ระบบบันทึกข้อมูลในการทดสอบเครื่องจักรกลเกษตรพื้นฐาน

โดย
นายภานุมาศ ทองตะนุนาม
นางสาวปฐมา เต็มวุฒิโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมเกษตร
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชา วิศวกรรมเกษตร

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบบันทึกข้อมูลในการทดสอบเครื่องจักรกลเกษตรพื้นฐาน

ผู้จัดทำ

1. นายภานุมาศ ทองตะนุนาม
2. นางสาวปฐมา เต็มวุฒิโรจน์

..... อาจารย์ที่ปรึกษา

(ผศ.ปานมนัส ศิริสมบรูณ์)

ปริญญานิพนธ์

ปีการศึกษา 2537

เรื่อง ระบบบันทึกข้อมูลในการทดสอบเครื่องจักรกลเกษตรพื้นฐาน

นายภานุมาศ ทองตะนุนาม
นางสาวปฐมา เต็มวุฒิโรจน์

อาจารย์ที่ปรึกษา
ผศ.ปานมนัส ศิริสมบูรณ์

บทคัดย่อ

ระบบแสดงและบันทึกผลข้อมูลจากการวัดที่สามารถทำงานได้ในขณะที่เคลื่อนที่ ถูกพัฒนาขึ้นเพื่อประยุกต์ใช้ในการทดสอบเครื่องจักรกลเกษตรขั้นพื้นฐาน ประกอบด้วยอุปกรณ์นำข้อมูลเข้าสู่คอมพิวเตอร์ โปรแกรมควบคุมการทำงาน และคอมพิวเตอร์แบบโน้ตบุ๊ค(Notebook Computer) ซึ่งอุปกรณ์นำข้อมูลเข้าสู่คอมพิวเตอร์เหล่านี้ใช้ IC ADC0809 เป็นตัวแปลงอนาลอกให้เป็นสัญญาณดิจิตอล ซึ่งจะทำให้สามารถรับสัญญาณอินพุท ได้พร้อมกัน 8 สัญญาณ และให้เอาท์พุทมีขนาด 8 บิท โดยใช้แรงดันอ้างอิง 0 ถึง 5 โวลท์ การควบคุมการสุ่มและการส่งข้อมูลเข้าคอมพิวเตอร์กระทำผ่านทางช่องต่อเครื่องพิมพ์ (Printer Port) ด้วยอัตราการสุ่ม 500 ข้อมูล (Data Record) ต่อวินาทีโดยโปรแกรมควบคุมการทำงานที่พัฒนาขึ้นมาด้วยภาษาซี(C language) การติดต่อสื่อสารกับผู้ใช้เป็นแบบเมนู (Menu User Interface) การบันทึกเป็นไปอย่างต่อเนื่องโดยสามารถเลือกอัตราการบันทึกได้ตั้งแต่ 1 ชุดข้อมูลต่อวินาที จนถึง 500 ชุดข้อมูลต่อวินาที สามารถเฉลี่ยข้อมูลในขณะที่บันทึกได้ โดยที่การแสดงผลจะเป็นแบบกราฟเส้นและตัวเลข ในขณะทำการบันทึก ข้อมูลจะถูกพักไว้ชั่วคราวในหน่วยความจำจนครบจำนวนชุดข้อมูลที่ต้องการจึงบันทึกลงบนแผ่นดิสก์ในรูปแบบคอมมาสเปรดชีท แวลู (Comma Spread Sheet Value Format) ซึ่งสามารถนำไปประมวลผลด้วยโปรแกรมสำเร็จรูปประเภทสเปรดชีทได้

Data Acquisition System for Basic Agricultural Machinery

Panumas Thongtanunam

Pathama Temvuthiroj

Advisor

Assist. Prof. Panmanas Sirisomboon

Abstract

Portable measured data presentation and recording system is developped for basic agricultural machine testing. The system is composed of computer's data logging unit, driving software ,and notebook computer. The computer's data loging unit has IC ADC 0809 to convert analog signal to digital signal, which enable it to recieve 8 input signal and also to send out 8 bit output data. It's reference voltage is 0 to 5 volt. The sampling and sending data to computer are performed through printer port which the sampling rate of 500 data record per second by the C language program developped. The user interface is menu drive. The user can select a datum record per minute to 500 data records per second of recording rate. The program has utility of averaging while recording. The data are continuously recorded and presented in both line graph and digits. The data are tempolary saved in memory during recording, till the required record's are approched. They will be saved on a diskette with Comma Spread Sheet Value format which can be imported into spread sheet software to analyze.

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

ระบบเก็บข้อมูลจากการวัด (Data Acquisition) ในอดีตใช้วิธีการจดบันทึกจากมาตรแสดงผล (Meter) ซึ่งอาจมีความคลาดเคลื่อนจากผู้บันทึก และไม่สามารถบันทึกด้วยความเร็วสูง ซึ่งปัจจุบันได้มีการพัฒนาอุปกรณ์ที่ใช้ในการเก็บข้อมูลที่สามารถบันทึกข้อมูลด้วยความเร็วสูงและถูกต้องอีกทั้งมีความสะดวกในการนำไปใช้งานกับคอมพิวเตอร์ได้

อย่างไรก็ตามอุปกรณ์ดังกล่าวจะมีราคาสูง ซึ่งไม่เหมาะสมที่จะติดตั้งกับระบบการวัดในการทดสอบเครื่องจักรกลเกษตรพื้นฐาน รวมทั้งยังขาดความรู้ความเข้าใจในการใช้งานระบบเก็บข้อมูล ดังนั้น จึงได้มีการศึกษาและสร้างระบบรวมทั้งได้จัดหาองค์ประกอบต่างๆเพื่อนำมาประกอบเป็นระบบเก็บข้อมูลที่นำไปใช้ในการทดสอบเครื่องจักรกลเกษตรพื้นฐาน โดยได้ระบบที่สามารถบันทึกผลทางฟิสิกส์ที่ได้จากการวัดและทดสอบ สามารถวัดและบันทึกข้อมูลในขณะที่มีการเคลื่อนที่ได้ ซึ่งประกอบด้วยอุปกรณ์นำสัญญาณจากการวัดเข้าสู่คอมพิวเตอร์หรือเรียกว่า คาต้าลอกเกอร์ (Data Logger) และโปรแกรมควบคุมการทำงานของคาต้าลอกเกอร์ โดยที่ระบบนี้ได้ถูกพัฒนาขึ้นมาบนเครื่องคอมพิวเตอร์แบบโน้ตบุ๊ค (Notebook Computer) ของคอมแพค รุ่น คอนทูร่า 3/20 (Compaq Contura 3/20)

การนำไปใช้งานจะต้องนำอุปกรณ์เหล่านี้ไปใช้ร่วมกับอุปกรณ์วัดสัญญาณที่มีการปรับสภาวะสัญญาณให้ออกมาเป็นสัญญาณแรงดันไฟฟ้า ขนาด 0-5 โวลท์

วัตถุประสงค์

เพื่อสร้างระบบบันทึกข้อมูลโดยคอมพิวเตอร์ซึ่งประกอบด้วยอุปกรณ์นำข้อมูลเข้าสู่คอมพิวเตอร์และโปรแกรมควบคุมการทำงาน โดยมีจุดมุ่งหมายดังนี้

1. อุปกรณ์ที่ได้มีราคาประหยัด
2. บันทึกข้อมูลจากการวัดได้หลายอย่างพร้อมกัน
3. สามารถบันทึกข้อมูลได้ด้วยความเร็วสูง
4. บันทึกข้อมูลปริมาณมากได้
5. โปรแกรมมีความสะดวกต่อการใช้งาน
6. ข้อมูลมีความสะดวกต่อการนำไปประมวลผลด้วยโปรแกรมสำเร็จรูปประเภทสเปรดชีท (Spread Sheet)

บทที่ 2

ทฤษฎีและหลักการ

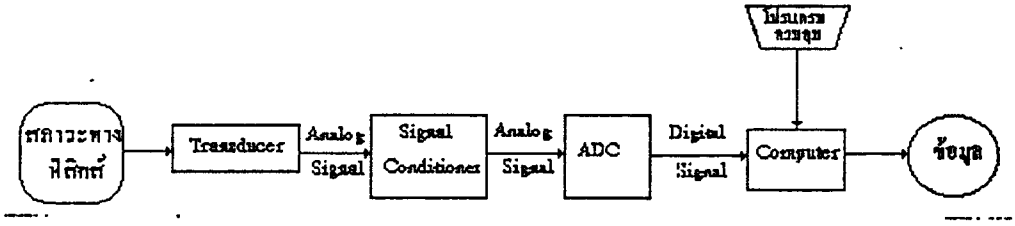
1. ระบบเก็บข้อมูล (Data Acquisition System)

ระบบเก็บข้อมูล คือระบบที่ใช้ทำงานเพื่อให้ได้มาซึ่งข้อมูลจากสภาวะสภาวะทางฟิสิกส์ที่สนใจ จุดประสงค์ของการใช้งานคือ ความต้องการนำข้อมูลจากสภาวะทางฟิสิกส์ นี้มาแสดงผลและ/หรือบันทึกผลเพื่อนำไปประมวลผล ซึ่งสามารถแบ่งโครงสร้างของระบบได้เป็น 2 ส่วน คือ ส่วนทำการวัด และส่วนแสดงผล/บันทึกผลเชื่อมต่อกันสัญญาณทางไฟฟ้า สามารถแสดงได้ด้วยแบบดังรูปที่ 2.1

รูปที่ 2.1 โครงสร้างพื้นฐานของระบบเก็บข้อมูล

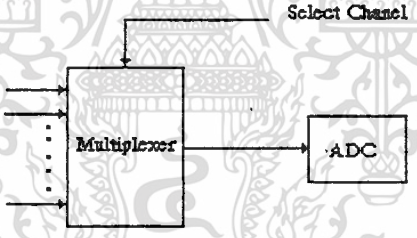
หน่วยทำการวัด เป็นอุปกรณ์ที่เปลี่ยนสภาวะทางฟิสิกส์ให้เป็นสัญญาณทางไฟฟ้า เรียกว่า ทรานสดิวเซอร์ (Transducer) ในส่วนของการแสดง/บันทึกผลนี้ ในปัจจุบันนิยมใช้คอมพิวเตอร์เข้ามาจัดการโดยมีโปรแกรมควบคุมการทำงานของคอมพิวเตอร์อีกที

เนื่องจากการติดต่อสื่อสารกับคอมพิวเตอร์ (Computer Interfacing) ต้องเป็นสัญญาณไฟฟ้าแบบสัญญาณดิจิตอล(Digital Signal) ซึ่งสัญญาณที่ได้จากทรานสดิวเซอร์เป็นสัญญาณไฟฟ้าแบบสัญญาณอนาลอก(Analog Signal) จึงต้องมีอุปกรณ์ที่แปลงสัญญาณอนาลอกเป็นดิจิตอล หรือ ADC (Analog to Digital Converter) และโดยทั่วไปสัญญาณที่ได้จากทรานสดิวเซอร์จะมีลักษณะที่ไม่เหมาะสมที่จะป้อนให้กับ ADC ดังนั้น จึงต้องใช้อุปกรณ์ปรับสภาวะสัญญาณ (Signal Conditioner) ทำการปรับสภาวะสัญญาณก่อน ดังนั้นแผนภาพระบบเก็บข้อมูลจึงมักจะมีรูปแบบดังรูปที่ 2.2



รูปที่ 2.2 โครงสร้างระบบเก็บข้อมูลที่บันทึกผลลงคอมพิวเตอร์

ระบบดังรูปที่ 2.2 นี้เป็นระบบที่มีช่องทาง (Channel) รับสัญญาณจากทรานสดิวเซอร์ช่องเดียว ในกรณีที่ระบบมีการเก็บข้อมูลจากสภาวะทางฟิสิกส์หลายอย่างพร้อมกัน (Multi Channal) จะใช้อุปกรณ์ ที่เรียกว่า มัลติเพลกเซอร์ (Muliplexer) ซึ่งจะทำหน้าที่เป็นสวิทช์ปิดเปิดให้สัญญาณผ่านทีละช่องทางตามรหัสเลือกช่องทาง อุปกรณ์นี้มักจะติดตั้งไว้ในตำแหน่งก่อนที่สัญญาณจะเข้าสู่ ADC ดังรูปที่ 2.3



รูปที่ 2.3 มัลติเพลกเซอร์

มีหลายบริษัทได้ผลิตอุปกรณ์ที่ใช้ในงานเก็บข้อมูล เรียกว่า คาต้าลอกเกอร์ (Data Logger) โดยรวมเอาอุปกรณ์ปรับสภาวะสัญญาณ,มัลติเพลกเซอร์ และ ADC เข้าไว้ด้วยกัน และมักจะมีอุปกรณ์พักข้อมูลชั่วคราวและ/หรืออุปกรณ์แสดงผลติดตั้งอยู่ด้วย การทำงานของส่วนต่าง ๆ ภายในเครื่องควบคุมด้วยไมโครโปรเซสเซอร์ (Microprocessor) ตัวอย่างของคาต้าลอกเกอร์ที่หาซื้อได้เช่น Hydra ของบริษัท Phillips/Fluke, ORP1200 ของบริษัท Yokogawa

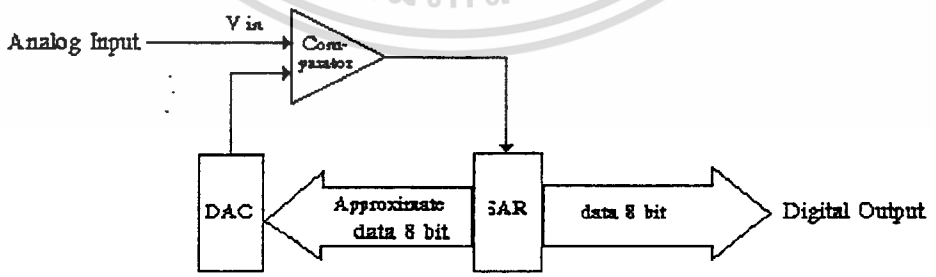
2. ทรานสดิวเซอร์

ทรานสดิวเซอร์เป็นอุปกรณ์ที่เปลี่ยนสภาวะทางฟิสิกส์ไปเป็นสัญญาณทางไฟฟ้า องค์ประกอบที่สำคัญของทรานสดิวเซอร์ คือ ตัวตรวจรู้ (Sensor) ที่ทำหน้าที่เปลี่ยนสภาวะทางฟิสิกส์เป็นสัญญาณทางไฟฟ้า โดยจะมีทั้งแบบพาสสีพ และ แบบแอคทีฟ โดยที่แบบแอคทีฟนี้ ตัวมันเองจะกำเนิดสัญญาณไฟฟ้าที่แปรตามสภาวะทางฟิสิกส์ หรือแบบพาสสีพที่ต้องป้อนไฟฟ้าเข้าไป แล้วทำหน้าที่เปลี่ยนแปลงลักษณะของไฟฟ้าตามสภาวะทางฟิสิกส์ ซึ่งในโครงงานนี้ได้จำลองสัญญาณด้วยเครื่องกำเนิดสัญญาณ และผลที่ได้จะถูกนำไปบันทึกและประมวลผลลงบนคอมพิวเตอร์

3. การแปลงสัญญาณ อนาลอกเป็นดิจิตอล

การจัดวงจร แปลงสัญญาณอนาลอกเป็นดิจิตอลมีหลายแบบ และอยู่ในวงจรรวม โดยจะกล่าวถึงวงจรต่าง ๆ โดยย่อ ได้แก่

1. Basic Conversion Method และ Counter Type ADC
2. Intergrating ADC แบ่งตามการทำงานได้ 2 แบบ คือ Single Slope และ Dual Slope
3. Parallel (Flash) ADC วงจรชนิดนี้เป็นวงจรที่มีความเร็วสูงสุด
4. Successive Approximate Converter การแปลงสัญญาณแบบนี้เป็นที่นิยมมากที่สุด และโครงงานนี้ได้ใช้การแปลงสัญญาณแบบนี้ด้วย ซึ่งจะกล่าวถึงการทำงานโดยสังเขป รูปที่ 2.4 ได้แสดงถึงบล็อคไดอะแกรมการทำงานของ 8 Bit A/D



รูปที่ 2.4 บล็อกไดอะแกรมการทำงาน 8 บิท A/D

หลักการทำงานของวงจรคือ วงจรจะทำการประมาณค่าสัญญาณอนาลอกเป็นรหัสเลขฐานสองแล้วนำไปเก็บในรีจิสเตอร์ที่เรียกว่า SAR (Successive Approximate Register) รหัสเลขฐานสองใน SAR จะถูกแปลงเป็นสัญญาณอนาลอกอีกครั้งโดย DAC (Digital to Analog Converter) เพื่อเปรียบเทียบกับสัญญาณอนาลอกที่อินพุท (Vin) ที่คอมพาราเตอร์ (Comparator) เอาท์พุทจากคอมพาราเตอร์จะไปควบคุมรหัสเลขฐานสองใน SAR จากนั้นจะกระทำซ้ำเช่นนี้จนครบ 8 ครั้ง ก็จะให้เอาท์พุทจากวงจร ADC เป็นสัญญาณดิจิตอลขนาด 8 บิท ออกมา

นอกจากนี้วงจร ADC ยังจำเป็นต้องมีวงจรสุ่มและคงค่าสัญญาณ (Sample and Hold) ก่อนที่สัญญาณจะเข้าสู่ ADC อันเนื่องมาจากการแปลงสัญญาณต้องใช้เวลาขณะหนึ่ง

สิ่งที่สำคัญมากสำหรับ ADC คือความเที่ยงตรงและความละเอียด (Accuracy and Resolution) ซึ่งจะได้กล่าวถึงโดยสังเขปดังนี้

ความแม่นยำ (Accuracy) คือค่าความแตกต่างที่มากที่สุดของอินพุทกับเอาท์พุท ขึ้นอยู่กับความแม่นยำของรีซีสเตอร์ที่ใช้ใน DAC และความแม่นยำของแรงดันอ้างอิง (Reference Voltage)

ความละเอียด (Resolution) คือค่าอินพุทที่เปลี่ยนแปลงอย่างน้อยที่สุดที่ทำให้เอาท์พุทเปลี่ยนแปลง ความละเอียดจะขึ้นอยู่กับจำนวนบิทของดิจิตอลเอาท์พุท

4. การติดต่อสื่อสารกับคอมพิวเตอร์

การเชื่อมต่อกับคอมพิวเตอร์ เพื่อการสื่อสารข้อมูลระหว่างอุปกรณ์ภายนอกกับคอมพิวเตอร์ จะกระทำผ่านพอร์ท(Port) ซึ่งมีหลายแบบ เช่น

- ผ่านช่องพอร์ทอนุกรม(Serial Communication Port) มักจะติดตั้งมากับคอมพิวเตอร์ มีรูปแบบการเชื่อมต่อตามมาตรฐาน RS 232-C

- ใช้อินเตอร์เฟสการ์ด (Interface Card)ที่ออกแบบตามมาตรฐานต่าง ๆ เช่น IEEE 488, IEEE 583, RS 422, RS 423 โดยนำการ์ดดังกล่าวเสียบลงในสลอตภายในเครื่องคอมพิวเตอร์ แล้วเชื่อมต่อกับพอร์ทบนการ์ดอีกที

- ผ่านทางพอร์ทเครื่องพิมพ์ ปรกติจะใช้ในการสื่อสารข้อมูลกับเครื่องพิมพ์ แต่ก็สามารถนำมาใช้สื่อสารกับอุปกรณ์อื่น ๆ ได้

5. โปรแกรมควบคุม

หน้าที่ของ โปรแกรม คือ

- สั่งการให้คอมพิวเตอร์ทำการรับส่งข้อมูลกับอุปกรณ์ภายนอก
- สั่งการให้คอมพิวเตอร์แสดงผลข้อมูลหรือบันทึกข้อมูล
- วิเคราะห์ข้อมูล

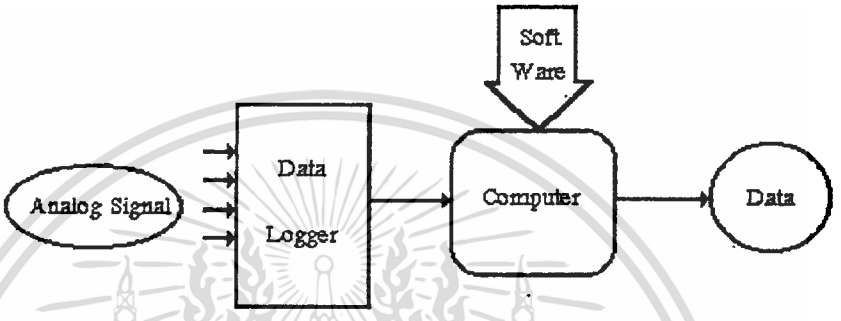
โปรแกรมนี้สามารถเขียนได้ด้วยภาษาเขียนโปรแกรม ซึ่งนอกจากจะต้องรู้หลักการของภาษาแล้ว ยังต้องรู้รายละเอียดโครงสร้างของพอร์ท และรูปแบบของการสื่อสารของอุปกรณ์ภายนอกที่มาเชื่อมต่อกับคอมพิวเตอร์ และต้องรู้วิธีการควบคุมอุปกรณ์ภายนอก

นอกจากนี้ยังมี Software สำเร็จรูป เช่น Labtech Notebook ของบริษัท Laboratory Technology, Test Team ของบริษัท Phillips/Fluke, Measure ของบริษัท National Instrument

บทที่ 3

การสร้าง

จากวัตถุประสงค์และทฤษฎี จึงได้ออกแบบโครงสร้างระบบไว้ดังรูปที่ 3.1 ซึ่งประกอบด้วย ดาต้าลอกเกอร์ คอมพิวเตอร์ และโปรแกรมควบคุมการทำงาน



รูปที่ 3.1 โครงสร้างของระบบที่สร้างขึ้นมา

1. ดาต้าลอกเกอร์ (Data Loggers)

เนื่องจากดาต้าลอกเกอร์เชื่อมต่อกับ คอมพิวเตอร์ทางพริ้นเตอร์พอร์ท จึงต้องศึกษารายละเอียดวงจรของ พริ้นเตอร์พอร์ท จากการศึกษารายละเอียดพบว่าประกอบด้วย พอร์ท พอร์ทย่อยๆ 3 พอร์ท คือ

1. พอร์ทเอาท์พุท ขนาด 8 บิท ทำหน้าที่ส่งข้อมูลแต่ละ บิท ทางขา 2 - 9 ของพริ้นร์ท ตามลำดับ ออกจากคอมพิวเตอร์ ดังนั้นจะใช้เป็นทางส่งข้อมูลจากคอมพิวเตอร์มายังดาต้าลอกเกอร์

2. พอร์ทอินพุท ขนาด 5 บิท ทำหน้าที่รับข้อมูลบิท ที่ 4, 5, 6, 7 และ 8 ทางขา 15, 13, 12, 10 และ 11ของปริ้นเตอร์พอร์ท ตามลำดับ ดังนั้นจะใช้เป็นทางรับข้อมูลจากดาต้าลอกเกอร์เข้าเกอร์เข้าสู่คอมพิวเตอร์

3. พอร์ทอิน/เอาท์พุท ขนาด 4 บิท ทำหน้าที่รับ/ส่งข้อมูลแต่ละบิท ทางขา 1, 14, 16 และ 17 ของพริ้นเตอร์พอร์ทตามลำดับ แต่เนื่องจากอุปกรณ์ภายในของพอร์ทนี้เป็นแบบโอเพ่นคอลเลกเตอร์(Open Colector)ข้อมูลจากภายนอกจะถูก AND กับข้อมูลจากภายในคอมพิวเตอร์ ทำให้มีความยุ่งยากจึงไม่นำมาใช้ จากนั้นมาออกแบบดาต้าลอกเกอร์ได้วงจรดังรูปที่ 3.2 ซึ่งประกอบด้วย

1. Connector DB25S เป็นช่องทางเชื่อมต่อดาต้าลอกเกอร์กับ พริ้นเตอร์พอร์ท ของคอมพิวเตอร์

2. IC 74LS244 และ 74LS245 ทำหน้าที่เป็น อินพุท และ เอาท์พุท บัฟเฟอร์ (Buffer) ตามลำดับ เพื่อป้องกันความผิดพลาดของสัญญาณระหว่าง ดาต้าลอกเกอร์ กับ พริ้นเตอร์พอร์ท

3. IC ADC0809 ทำการแปลงสัญญาณ อนาลอก เป็น ดิจิตอล ขนาด 8 บิท สามารถเลือกได้ 8 ช่อง โดยใช้สัญญาณ ดิจิตอล ขนาด 3 บิท ช่องสัญญาณอนาลอกที่ถูกเลือกจะเปิดให้สัญญาณอนาลอก เข้าสู่ ADC เมื่อ ADC ได้รับสัญญาณ HIGH ที่ขา 22 (ALE) ซึ่งจะเป็นการสั่งให้ ADC เริ่มการแปลง ด้วย เพราะขา 22 นี้ต่อพ่วงอยู่กับขา 6 (START) ซึ่งเลือกช่องสัญญาณ ขนาด 3 บิท และ เริ่มการแปลงขนาด 1 บิท นี้ถูกส่งมาจากขา 2, 3, 4 และ 9 ของ พริ้นเตอร์พอร์ท โดยผ่านคอนเนคเตอร์ และ บัฟเฟอร์ ที่ได้กล่าวไว้ในข้อ 2 และ 1 ตามลำดับ

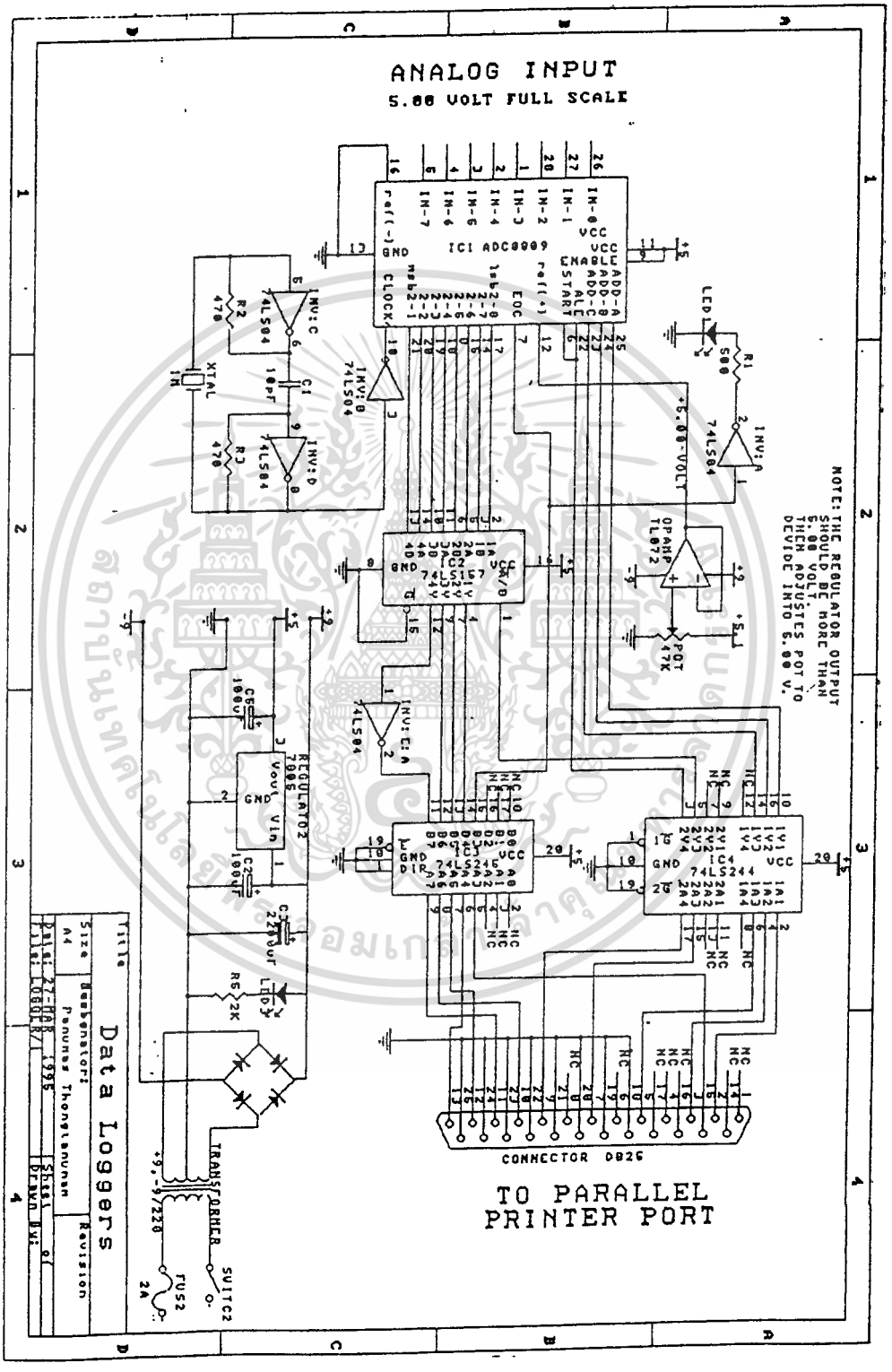
สำหรับขา 7(EOC) ต่อเข้ากับขา 15 ของ พริ้นเตอร์พอร์ท โดยผ่านบัฟเฟอร์ และคอนเนคเตอร์ เพื่อตรวจสอบสภาวะการทำงานของ ADC โดย คอมพิวเตอร์ และขา 7 นี้ยังต่อไปยัง LED1 เพื่อแสดงสภาวะการทำงานที่ตัวดาต้าลอกเกอร์ด้วย โดยต่อผ่าน IC 74LS04 อินเวอร์เตอร์เพื่อกลับสถานะสัญญาณก่อน เนื่องจากสัญญาณนี้จะมีสถานะเป็น LOW ในระหว่างที่ ADC ทำการแปลง ส่วนขาข้อมูล 8 ขาถูกต่อไปยังขา อินพุท ของ Multiplexer ซึ่งจะกล่าวถึงในหัวข้อถัดไป

4. IC 74LS157 Multiplexer ทำการสลับข้อมูลที่มาจาก ADC ให้ออกมาทีละ 4 บิท การเลือกว่าจะส่ง 4 บิทแรก หรือ 4 บิทหลัง ออกมานั้นทำได้โดยการป้อนสัญญาณ LOW หรือ HIGH ตามลำดับให้กับขา 1(SLT) สัญญาณนี้ถูกส่งมาจากขา 7 ของ พริ้นเตอร์พอร์ท ส่วนข้อมูลที่ IC ตัว นี้ส่งออกมาถูกป้อนให้กับ ขา 13, 12, 10 และ 11 ของ พริ้นเตอร์พอร์ท โดยบัฟเฟอร์ และ คอนเนคเตอร์ สำหรับข้อมูลที่เข้าสู่พริ้นเตอร์พอร์ท ทางขา 11 นั้นจะถูกสถานะโดยวงจรภายใน พริ้นเตอร์พอร์ท ดังนั้นข้อมูลบิทที่ 4 ที่ออกจาก IC ตัวนี้จึงถูกต่อผ่าน IC 74LS04 Inverter เพื่อกลับสถานะก่อน

5. วงจรแรงดันอ้างอิง ของ ADC ประกอบด้วย Potentiometer ทำการแบ่งแรงดันขนาด 5.1 โวลท์ ให้ได้ 5.0 โวลท์ และเพื่อเสถียรภาพของแรงดันอ้างอิงจึงใช้ ออปแอมป์ ต่อแบบโวล-เตจ ฟอลโลเวอร์(Voltage Follower)รับแรงดัน 5.0 โวลท์ นี้ก่อนส่งให้กับ ADC ส่วนแรงดันขนาด 5.1 โวลท์ ได้มาจาก Regulator 7805 ซึ่งตามหลักการแล้ว แรงดันที่ได้จาก Regulator 7805 นี้จะต้องเป็น 5.0 โวลท์ แต่ค่าที่ได้มักจะมากกว่า 5.0 โวลท์

6. วงจรกำเนิด สัญญาณนาฬิกา ขนาด 1 MHz ป้อนให้กับ ADC เพื่อใช้เป็นจังหวะของการทำงาน

7. วงจรจ่ายไฟเลี้ยง ประกอบด้วยวงจร Rectifier จาก AC 220 โวลท์ เป็น DC +9, -9 โวลท์ จ่ายให้กับ ออปแอมป์ และไฟ DC +9 โวลท์ จะถูก regulate ด้วย Regulator 7805 เป็น DC +5 โวลท์ แต่จะได้จริงๆ ประมาณ +5.1 โวลท์ ใช้เป็นไฟเลี้ยงไอซีต่างๆ ในวงจร



รูปที่ 3.2 วงจรของดาต้าลอกเกอร์

2. โปรแกรมควบคุมการทำงาน

โปรแกรมควบคุมนี้ พัฒนาขึ้นโดยใช้ภาษา ซี (C Language) โดยใช้คอมพลายเลอร์ (Compliler) ของเทอร์โบซี (Turbo C) เวอร์ชั่น 2 พัฒนาบนเครื่องคอมพิวเตอร์ แบบโน้ตบุ๊คของคอมแพครุ่น คอนทูร่า 3/20 (Compaq Contura 3/20 Notebook Computer) ซึ่งมี CPU เบอร์ 386SX การติดต่อสื่อสารกับผู้ใช้ของโปรแกรมนั้นเป็นแบบเมนู (Menu User Interfacing) ในโหมดการแสดงผล C80 (Color 80 text/line) โดยพัฒนารูทีน (Routine) ในการติดต่อสื่อสารกับผู้ใช้ขึ้นมาเอง ขณะที่โปรแกรมทำงานในขั้นตอนเก็บข้อมูล จะแสดงผลเป็นกราฟเส้นดังนั้นจึงใช้การแสดงผลในโหมดEGA (Entrance Graphic Adapter) มีการใช้ไทเมอร์อินเทอร์รัพท์ (Timer Interrupt) ดังนั้นจึงตั้งออปชั่น (Option) ของคอมพลายเลอร์ ไม่ให้อยู่ในมาตรฐาน ANSI และไม่มีการตรวจสอบ สแต็ก โอเวอร์โฟล์ว (Stack Over Flow) นอกจากนี้ตัวโปรแกรมยังแยกกันอยู่ในไฟล์หลายไฟล์ ดังนั้นในการคอมพลายจะต้องสร้างโปรเจกไฟล์ (Project File)

การทำงานที่สำคัญของโปรแกรมจะมีลักษณะดังนี้ คือ

1. การสุ่มข้อมูล ซึ่งขั้นตอนอย่างคร่าว ๆ คือ ควบคุมคาต้าลอกเกอร์ให้แปลงสัญญาณแล้วส่งข้อมูล (สัญญาณดิจิตอลที่ได้จากการแปลง) เข้าสู่คอมพิวเตอร์

2. จัดการกับข้อมูลที่เข้าสู่คอมพิวเตอร์ ได้แก่ การเฉลี่ยข้อมูล แสดงผล และเก็บข้อมูลไว้ชั่วคราวในหน่วยความจำ (RAM)

3. นำข้อมูลที่เก็บไว้ชั่วคราวมาบันทึกลงบนแผ่นดิสก์ (Diskette) เป็นการถาวร

การทำงานในขั้นตอนที่ 1 และ 2 จะมีความเกี่ยวเนื่องกัน ซึ่งจะต้องดำเนินไปพร้อม ๆ กัน ในขณะที่ทำการเก็บข้อมูล ส่วนขั้นตอนที่ 3 จะกระทำภายหลังจากที่เก็บข้อมูลครบจำนวนที่ต้องการแล้ว

เนื่องจากตัวโปรแกรมมีรายละเอียดมาก ดังนั้นจะกล่าวถึงเฉพาะโครงสร้างในตอนเริ่มโปรแกรม และตอนที่ทำงานในข้อ 1 และ 2 ในตอนเริ่มโปรแกรม ซึ่งโมดูลเมน (Main) จะทำหน้าที่ส่งการทำงานไปยังโมดูลย่อยอื่น ๆ ตามที่ผู้ใช้เลือก โมดูลย่อยต่างๆ ได้แก่

1. โมดูลคอนฟิก (Config Module) เป็นส่วนที่ให้ผู้ใช้ตั้งสภาวะต่าง ๆ ในการเก็บข้อมูล สภาวะเหล่านี้ได้แก่ หมายเลขช่องสัญญาณที่ใช้ ค่าพิกัดข้อมูลของแต่ละช่อง อัตราเร็วในการบันทึก จำนวนข้อมูลที่ใช้ในการเฉลี่ย และจำนวนชุดข้อมูลที่ต้องการ (Required Records)

2. โมดูลรัน (Run Module) เป็นโมดูลที่ดำเนินการเก็บข้อมูล ซึ่งจะกล่าวถึงโครงสร้างที่สำคัญของโมดูลนี้ในภายหลัง

3.โมดูลเซฟ (Save Module) ทำหน้าที่บันทึกข้อมูลลงแผ่นดิสก์ข้อมูลถูกจัดเก็บในรูปแบบของคอมม่าสเปรดชีทแวลู (Comma Spread Sheet Value Format) ซึ่งจะใช้ส่วนขยายชื่อไฟล์ (extension) เป็น .CSV ตัวอย่างแสดงไว้ในรูปที่ 3.3

```
Comment:, Demonstration of configuration,
Total Record:,300,
Recording Rate:, 10 rec/sec,
Average:,10, time
Resolution:, 1/256, of Span
Used channels:,3,
1,3,6,
Drawbar pull,Velocity, Torque,
500,60,10000,
0,0,0,
kgF,km/h,lb.in,
192,24,20,
198,24,21,
203,25,19,
206,26,18,
209,27,20,
```

รูปที่ 3.3 ตัวอย่างไฟล์ข้อมูลที่ได้จากการทำงานของโปรแกรม

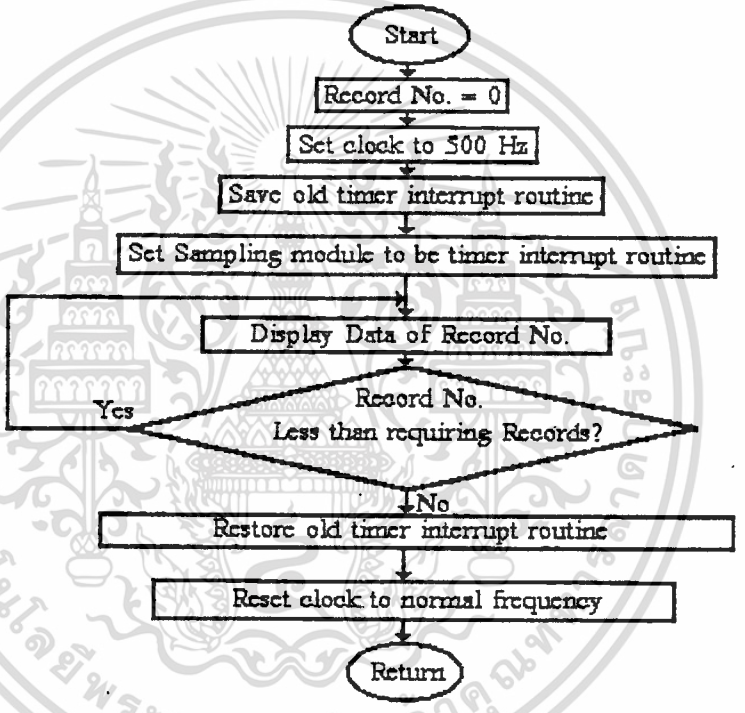
โดยที่ 11 บรรทัดแรกจะเป็นข้อมูลที่แสดงส่วนหัวของไฟล์ที่แสดงสภาวะในการเก็บข้อมูล บรรทัดต่อจากนั้นเป็นข้อมูลที่ถูกบันทึกไว้ในลักษณะรหัสตัวเลขระหว่าง 0 ถึง 255 เพื่อประหยัดเนื้อที่จัดเก็บ การหาค่าจริงทำได้โดยใช้สูตร

ค่าจริง = ค่ารหัส x (maximum - minimum) / 256 + minimum

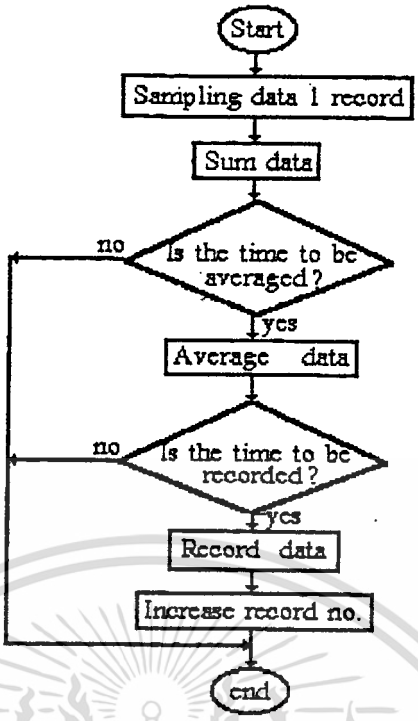
โดยค่าแมกซิมั่ม (maximum) และค่ามินิมั่ม (minimum) แสดงไว้ในบรรทัดที่ 9 และ 10 ตามลำดับ การแปลงค่ารหัสเป็นค่าจริงดังกล่าวสามารถกระทำได้โดยง่าย โดยโปรแกรมสำเร็จประเภทสเปรดชีท

4. โมดูลโอเอสเชลล์ (OS shell) ให้ผู้ใช้ติดต่อกับระบบปฏิบัติการ (Operating System)
5. โมดูลอเบ้าท์ (About) แสดงข้อความที่มาของโปรแกรม
6. โมดูลควิท (Quit) จบการทำงานของโปรแกรม

สำหรับโครงสร้างที่สำคัญของโมดูลรัน แสดงไว้ในรูปที่ 3.4 ซึ่งเป็นการกำหนดการกำหนดโมดูลแซมปลิ้ง (Sampling) ให้ถูกเรียกใช้ทุก ๆ 2 มิลลิวินาที โดยการตั้งสัญญาณไทเมอร์อินเทอร์รัพท์ให้มีความถี่ 500 Hz แล้วกำหนดโมดูลแซมปลิ้งให้เป็นไทเมอร์อินเทอร์รัพท์รูทีน โมดูลแซมปลิงเมื่อถูกเรียกจะทำการสุ่มเฉลี่ย และบันทึกข้อมูล โมดูลรันจะนำข้อมูลจากการสุ่มของโมดูลแซมปลิงมาแสดงผลจนครบจำนวนชุดข้อมูลที่ต้องการ ก็จะยกเลิกการใช้โมดูลแซมปลิง โครงสร้างของโมดูลแซมปลิง แสดงไว้ในรูปที่ 3.5



รูปที่ 3.4 บล็อคไดอะแกรมการทำงานของโมดูลรัน



รูปที่ 3.5 บล็อคไดอะแกรมการทำงานของโมดูลแซมปลิง

บทที่ 4

การทดลอง

การทดลอง หาค่าเอาท์พุทจากอินพุทแบบสุ่ม (Randomize Input Signal) ของคาต้าลอกเกอร์

วัตถุประสงค์ เพื่อหาค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐาน ของค่าความผิดพลาดระหว่างอินพุทกับเอาท์พุท ของคาต้าลอกเกอร์

เงื่อนไข การทดลองนี้ใช้ดิจิตอลมัลติมิเตอร์ของฟลุ้ค รุ่น 87 (FLUKE Digital Multimiter) เป็น อุปกรณ์อ้างอิง โดยใช้เป็นตัววัดสัญญาณอินพุท ซึ่งจะตั้งการทำงานไปที่ DCV พิกัด 0.000 กับ 4.000 โวลท์

วิธีการทดลอง

1. สุ่มตัวเลขที่มีค่าระหว่าง 0 ถึง 4000 จำนวน 50 ตัว

2. ตั้งสภาวะการทำงานของโปรแกรมควบคุมให้ทำการบันทึกสัญญาณที่ช่วง 1 มีค่าสูงสุด (Maximum) เท่ากับ 5000 และค่าต่ำสุด(Minimum) เท่ากับ 0 ใช้หน่วยเป็น มิลลิโวลท์ (Millivolt,mv) และใช้อัตราการบันทึก 10 ชุดข้อมูลต่อวินาที (10 rec/sec) จำนวนเฉลี่ยข้อมูลต่อครั้งเท่ากับ 1 (ไม่เฉลี่ย)

3. ที่อินพุทช่วง 1 ของคาต้าลอกเกอร์ ป้อนแรงดันที่มีค่าเป็นมิลลิโวลท์ เท่ากับตัวเลขสุ่ม 1 ตัว โดยใช้ดิจิตอลมัลติมิเตอร์ของฟลุ้ควัดแรงดันที่ป้อน

4. จดบันทึกตัวเลขเอาท์พุทที่อ่านได้จากจอแสดงผล (เนื่องจากโปรแกรมจะบันทึกได้เฉพาะแบบต่อเนื่องเท่านั้น ส่วนการทดลองนี้ต้องปรับแรงดันอินพุทแต่ละครั้ง ทำให้การบันทึกไม่ต่อเนื่อง)

5. หาความแตกต่าง (E) ของอินพุท (I) กับเอาท์พุท (O) แต่ละตัวโดยใช้สูตร

$$E_i = O_i - I_i \quad ; i=1,2,3,...,n,\ n=50$$

6. ทำซ้ำข้อ 3,4 และ 5 จนครบตัวเลขสุ่มจำนวน 50 ตัว

7. หาค่าเฉลี่ย จาก $\bar{E} = \frac{1}{n} \sum_{i=1}^{n} E_i$

และค่าเบี่ยงเบนมาตรฐาน จาก $\sigma_E = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (E_i - \bar{E})^2}$

ผลการทดลอง

ตารางบันทึกผลการทดลอง

ลำดับที่	อินพุท (I)	เอาท์พุท(O)	ค่าความผิดพลาด (E)	ลำดับที่	อินพุท (I)	เอาท์พุท(O)	ค่าความผิดพลาด (E)
1	346	351	5	26	3503	3496	-7
2	130	136	6	27	2245	2246	1
3	2980	2968	-12	28	2565	2558	-7
4	1090	1093	3	29	2776	2773	-3
5	3654	3652	-2	30	3812	3808	-4
6	3116	3105	-11	31	1676	1679	3
7	1591	1582	-9	32	3647	3632	-15
8	2414	2402	-12	33	1991	1992	1
9	2943	2929	-14	34	3588	3574	-14
10	3119	3105	-14	35	3734	3730	-4
11	1002	996	-6	36	1307	1308	1
12	2555	2539	-16	37	3979	3964	-15
13	3571	3554	-17	38	1990	1992	2
14	2874	2871	-3	39	3558	3554	-4
15	2488	2480	-8	40	88	97	9
16	1360	1367	7	41	2485	2480	-5
17	1411	1406	-5	42	3286	3281	-5
18	2715	2714	-1	43	459	468	9
19	2458	2460	2	44	662	664	2
20	1041	1035	-6	45	1891	1894	3
21	3113	3105	-8	46	1860	1855	-5
22	3434	3417	-17	47	2761	2753	-8
23	3189	3183	-6	48	1363	1367	4
24	1982	1972	-10	49	1635	1640	5
25	3207	3203	-4	50	1146	1152	6

วิเคราะห์ผลการทดลอง

$$\bar{E} = -4.16$$

$$\sigma_E = 7.17317$$

เปอร์เซนต์เทียบสแปน (Span = Maximum - Minimum)

$$\%\bar{E} = \frac{\bar{E}}{span} \times 100 = \frac{-4.16}{(5000 - 0)} \times 100 = -0.0832\ \%$$

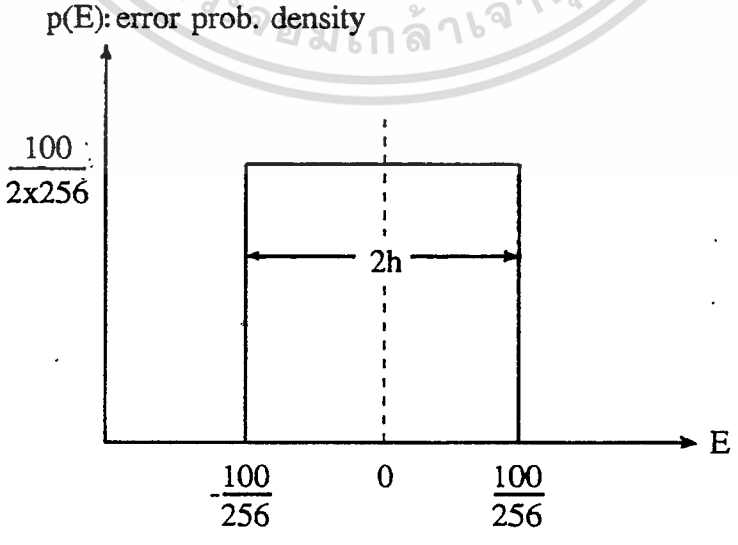
$$\%\sigma_E = \frac{\sigma_E}{span} \times 100 = \frac{7.17317}{(5000 - 0)} \times 100 = 0.1435\ \%$$

สรุปผลการทดลองและวิจารณ์

ส่วนเบี่ยงเบนมาตรฐานของ ADC 0809 หาได้โดยพิจารณาจากค่าความละเอียด (Resolution) ของตัวADC ซึ่งจากข้อมูล ADC 0809 ในภาคผนวก พบว่าการจัดความต้านทานภายใน ทำให้สแปนของสัญญาณฟูลสเกล ถูกแบ่งออกเป็น 256 ส่วน ทำให้มีความละเอียด เท่ากับ 100/256 % ดังนั้นควอนไตซ์เซชั่นเออเรอร์ (Quantization Error) มีค่าเท่ากับ $\pm 100/(2x256)$ % ซึ่งหมายความว่า สมการความหนาแน่นของความน่าจะเป็นสำหรับค่าความผิดพลาด (Error Probability Density Function) จะเป็นดังนี้ คือ

$$p(E) = \begin{cases} 0 & \text{เมื่อ } E < -\dfrac{100}{2\times256} \\ \dfrac{100}{2\times256} & \text{เมื่อ } -\dfrac{100}{2\times256} \le E \le \dfrac{100}{2\times256} \\ 0 & \text{เมื่อ } E > \dfrac{100}{2\times256} \end{cases}$$

ซึ่งจะเขียนเป็นกราฟได้ดังนี้



รูปที่ 4.1 การกระจายแบบสี่เหลี่ยม

จากกราฟเห็นได้ว่าจะเป็นการกระจายแบบสี่เหลี่ยม (Rectangular Distribution) ซึ่งค่าเบี่ยงเบนมาตรฐานของการกระจายแบบนี้ เมื่อมีความกว้าง 2h คือ $h/\sqrt{3}$

เพราะฉะนั้น ค่าเบี่ยงเบนมาตรฐานของ ADC 0890 มีค่าเท่ากับ

$$\sigma_{ADC} = \frac{100}{2 \times 256 \times \sqrt{3}} = 0.11276\ \%$$

ดังนั้น ค่าเบี่ยงเบนมาตรฐานของดาต้าลอกเกอร์หาได้จากการทดลอง จะมีค่ามากกว่าค่าเบี่ยงเบนมาตรฐานของ ADC 0809 เล็กน้อย ซึ่งสาเหตุอาจเนื่องมาจาก มีการเก็บตัวอย่างข้อมูลน้อยเกินไป และอุปกรณ์อ้างอิง คือ ฟลุ้ค ดิจิตอลมัลติมิเตอร์ มีความแม่นยำไม่เพียงพอ

ส่วนค่าความผิดพลาดเฉลี่ย มีค่าติดลบ หมายความว่า โดยเฉลี่ยแล้ว ข้อมูลที่อ่านได้จะมีค่าน้อยกว่าค่าจริง เท่ากับ 0.0832% ของค่าสแปน

อย่างไรก็ตาม การทดลองนี้ใช้ดิจิตอลมัลติมิเตอร์ของฟลุ้ครุ่น 87 เป็นอุปกรณ์อ้างอิง ซึ่งจะถือว่าได้รับการปรับเทียบมาจากอุปกรณ์ปรับเทียบแบบมาตรฐาน ดังนั้นค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานที่หาได้ เป็นค่าสัมพันธ์กับดิจิตอลมัลติมิเตอร์ดังกล่าว

อีกประการหนึ่งคือเนื่องจากพิกัดของดิจิตอลมัลติมิเตอร์ที่จะดูความละเอียดเป็นมิลลิโวลท์ได้นั้น จำกัดอยู่ที่ 4 โวลท์ การทดลองนี้จึงไม่ได้ทำที่มากกว่า 4 โวลท์จนถึง 8 โวลท์ แต่ก็อาจจะถือได้ว่า ค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานของดาต้าลอกเกอร์มีความสม่ำเสมอตลอดช่วงสแปน

บทที่ 5

บทสรุปและวิจารณ์

จากการทดลองในบทที่ 4 แสดงให้เห็นว่า ข้อมูลที่ได้จากการทำงานของคาต้าลอกเกอร์ มีมีความแม่นยำใกล้เคียงกับความแม่นยำของ ADC 0809 ซึ่งมีค่าเป็นที่น่าพอใจ มีความเหมาะสมที่จะนำไปใช้ในการทดสอบเครื่องจักรกลเกษตรพื้นฐาน ในกรณีที่ต้องการความแม่นยำสูงกว่านี้ ควรเปลี่ยน ADC ให้เป็นแบบที่ให้เอาท์พุทมีจำนวนบิทมากกว่าเดิม ซึ่งจะต้องออกแบบวงจรของคาต้าลอกเกอร์และโปรแกรมควบคุมการทำงานใหม่ทั้งหมด แต่ก็สามารถนำโครงงานนี้เป็นแนวทางได้ สิ่งที่ต้องปรับปรุงอีกประการของคาต้าลอกเกอร์ คือ การป้องกันขนาดของสัญญาณอินพุทให้อยู่ในช่วงที่กำหนด คือ 0 ถึง 5 โวลท์ เพราะถ้าหากเกินช่วงดังกล่าว ก็อาจก่อให้เกิดความเสียหายกับคาต้าลอกเกอร์ หรืออาจเสียหายถึงคอมพิวเตอร์ได้

สำหรับตัวโปรแกรมควบคุม มีความสะดวกต่อการใช้งานพอสมควร เนื่องจากได้ออกแบบ การติดต่อสื่อสารกับผู้ใช้ให้เป็นแบบเมนู สิ่งที่จะต้องปรับปรุงสำหรับโปรแกรมควบคุม คือ การพัฒนาให้สามารถควบคุมการบันทึกแบบไม่ต่อเนื่องได้ โดยให้ผู้ใช้สามารถควบคุมให้บันทึกข้อมูลในจังหวะที่ต้องการได้ รวมทั้งเพิ่มเติมโมดูลที่แสดงข้อความอธิบายการใช้งานหรือที่เรียกว่า เฮลป์ (Help) ตลอดจนให้มีการแสดงผลในโหมดภาษาไทย

ระบบนี้สามารถนำไปใช้เก็บข้อมูลในขณะเคลื่อนที่ได้ แต่ไฟเลี้ยงระบบต้องใช้แหล่งจ่ายไฟจากแบตเตอรี่ แล้วใช้อินเวอร์ทเตอร์แปลงไฟกระแสตรงเป็นกระแสสลับ สิ่งที่ต้องคำนึงถึงอีกประการเมื่อนำไปใช้งานแบบนี้คือ ความร้อน และการสั่นสะเทือนที่อาจทำความเสียหายให้กับคอมพิวเตอร์ ซึ่งในการพัฒนาต่อไปในอนาคตควรมีการนำอุปกรณ์วัดจริงมาติดตั้ง เช่น อุปกรณ์วัดแรง หรือ โหลดเซลล์ (Load Cell) ซึ่งต้องใช้อุปกรณ์ปรับสภาวะสัญญาณที่เรียกว่าสเตรนแอมพลิฟลายเออร์ (Strain Amplifier) ก่อนปรับสัญญาณให้กับคาต้าลอกเกอร์ แล้วทำการทดลองดังเช่นบทที่ 4 เพื่อหาความแม่นยำของทั้งระบบอีกที

อุปกรณ์วัดสัญญาณแบบอื่นนั้นสามารถนำมาใช้กับระบบบันทึกข้อมูลนี้ได้ โดยต้องมีการปรับสภาวะสัญญาณให้เหมาะสมก่อนปรับเข้าสู่คาต้าลอกเกอร์

สรุปผล

จากลักษณะของคาต้าลอกเกอร์ และโปรแกรมควบคุม ทำให้ได้ระบบบันทึกข้อมูลที่คุณสมบัติดังนี้

1. สามารถบันทึกข้อมูลจากเครื่องมือวัดได้พร้อมกัน 8 สัญญาณ

2. รับสัญญาณอินพุทแบบ Analog Voltage ในช่วง 0 ถึง 5 Volt

3. อัตราการสุ่มข้อมูล 500 ชุดข้อมูลต่อ วินาที

4. สามารถเลือกอัตราการบันทึกได้ตั้งแต่ 1 ชุดข้อมูลต่อนาที จนถึง 500 ชุดข้อมูลต่อวินาที

5. สามารถเฉลี่ยข้อมูลได้ถึง 250 ครั้ง ต่อข้อมูลที่บันทึก 1 ค่า

6. สามารถบันทึกได้ถึง 460,000 ข้อมูล

7. แสดงผลทางจอคอมพิวเตอร์เป็นกราฟเส้นและตัวเลข

8. บันทึกข้อมูลเป็น Text File ที่มีรูปแบบ Comma Spread Sheet Value (นามสกุล .CSV)

9. ข้อมูลที่บันทึกไว้สามารถนำไปประมวลผลด้วยซอฟท์แวร์สำเร็จประเภท Spread Sheet เช่น Excel, Lotus

10. ใช้กับคอมพิวเตอร์ IBM หรือ Compatible มี CPU เบอร์ 386 ขึ้นไป และสามารถแสดงผลในโหมด EGA

11. เชื่อมต่อกับคอมพิวเตอร์ทาง LPT1 (Printer Port)

และสามารถนำไปใช้กับอุปกรณ์วัดแรงฉุดลาก, ความเร็วในการเคลื่อนที่ ความเร็วเชิงมุม และแรงบิด ในการทดสอบเครื่องจักรเกษตรพื้นฐานได้

หนังสืออ้างอิง

1. Willis J. Tompkins and John G. Webster , "Interfacing Sensors to the IBM PC". Prentice Hall International, 1988, P. 145-150.

2. John P. Bently, "Principle of Measurement Systems" , Teside Polytechnic, P. 232-244,and P. 465-470.

3. Kent Porter, "Stretching Turbo C" , Brady New York ,P. 3-26 ,P. 91-222,and 365-377.

4. พูลพงษ์ บุณพราหมณ์, "คอมพิวเตอร์ช่วยงานอุตสาหกรรม", สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), พ.ศ. 2534, หน้า 164-216.

5. ธานินทร์ ถาวรศาสน์วงศ์ และ ทินกร คุ๊ก, "การอินเตอร์เฟส IBM PC" , ฟิสิกส์-เซนเตอร์, หน้า 139-156, 191-196

6. เปรมจิตร วิสุทธิศิริม," พื้นฐานวงจร เอทูดี-ดีทูเอ" , เซมิคอนดัคเตอร์ เล่มที่ 103, พ.ศ. 2533 , หน้า 308

กิตติกรรมประกาศ

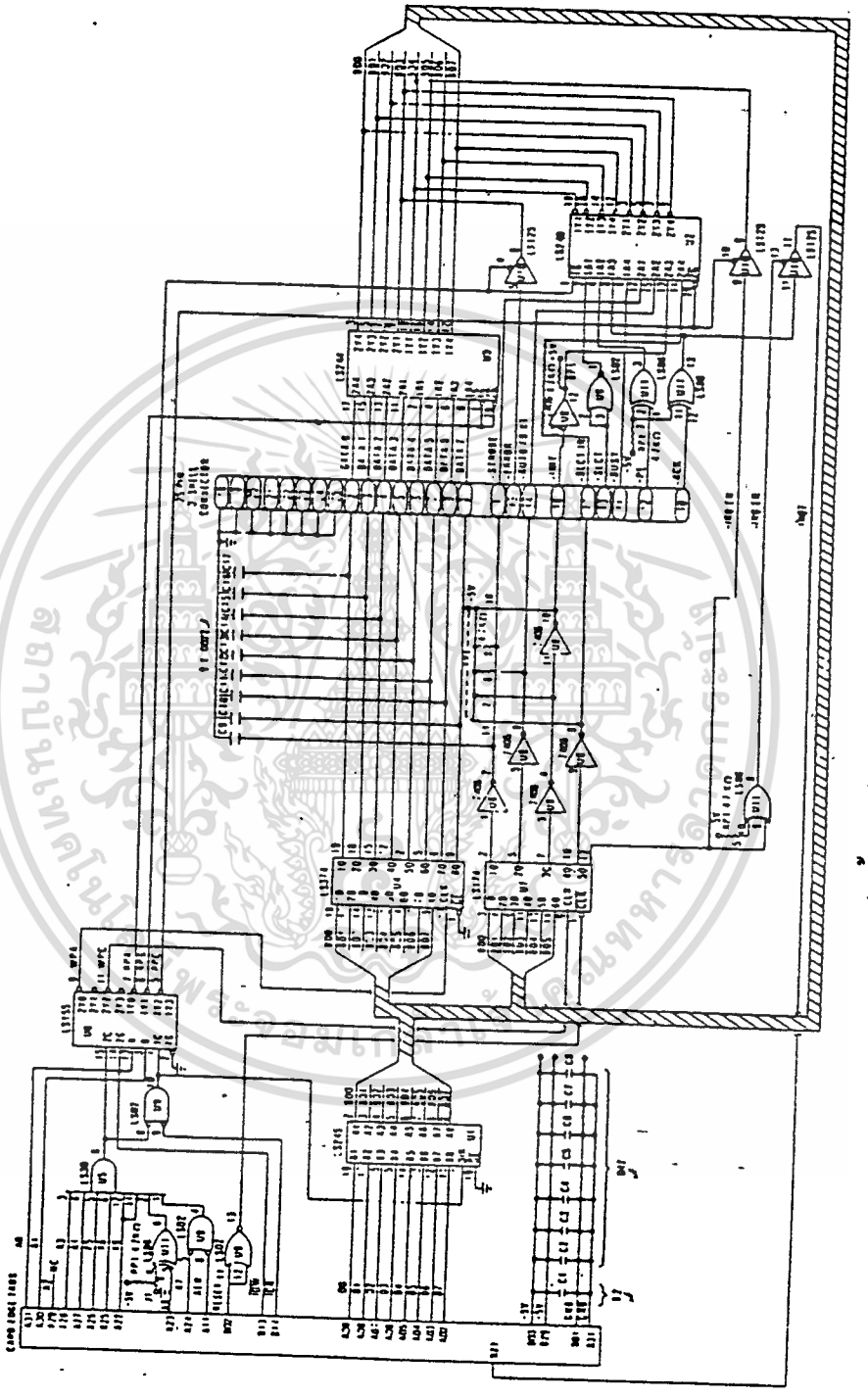
โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากความร่วมมือของหลายฝ่ายด้วยกัน

ขอขอบคุณคณาจารย์ และเจ้าหน้าที่ประจำโรงปฏิบัติงานภาควิชาวิศวกรรมเกษตร รวมทั้งเพื่อนนักศึกษาทุกคนที่ให้ความช่วยเหลือและเป็นกำลังใจ และ ขอขอบคุณเป็นพิเศษสำหรับกระทรวงวิทยาศาสตร์เทคโนโลยีและสิ่งแวดล้อม ซึ่งให้งบประมาณวิจัยเรื่องเครื่องเกี่ยวนวดถั่วเหลือง ซึ่งครอบคลุมโครงงานนี้ และขอขอบคุณผู้ที่มีส่วนช่วยเหลือโครงงานดังนี้

- พี่ศมิทธิ์ เอมสมบัติ ที่ให้ความรู้เกี่ยวกับเรื่องปริ้นเตอร์พอร์ท
- พี่สมภพ ภูริวิกรัยพงศ์ ที่ให้ความรู้และคำแนะนำเกี่ยวกับเรื่อง ADC
- ภาควิชาวิศวกรรมเครื่องกล ที่ให้ความสะดวกในการใช้อุปกรณ์และเครื่องมือต่าง ๆ สำหรับการทำชิ้นงาน และเจ้าหน้าในโรงปฏิบัติการเครื่องกล
- ขอขอบคุณ ดร. รัตติกร วรากูลศิริพันธ์ ให้ความอนุเคราะห์อุปกรณ์และสถานที่

สุดท้ายนี้ ขอกราบขอบพระคุณ *ผศ.ปานมนัส ศิริสมบูรณ์* อาจารย์ที่ปรึกษาโครงงาน ซึ่งเป็นผู้ที่มีความสำคัญที่สุด ที่ให้คำแนะนำและคอยติดตามผลงานด้วยความเอาใจใส่อย่างใกล้ชิดเสมอมา

ภาคผนวก



รูปพริ้นเตอร์อแดปเตอร์ (Printer Adapter)

National Semiconductor Corporation

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.)

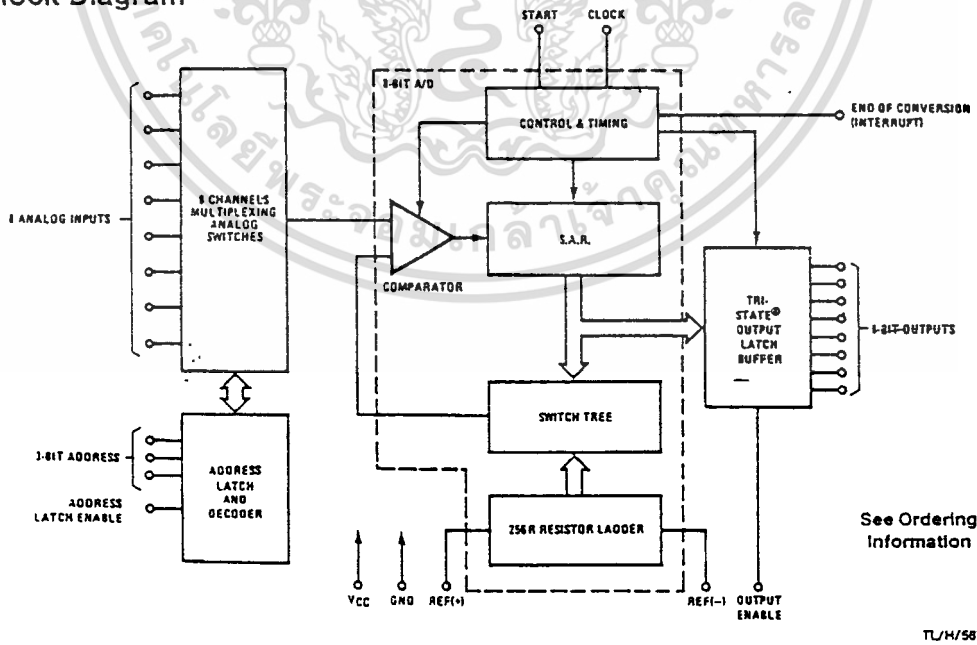
Features

- 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

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

TL/H/5672–1

Absolute Maximum Ratings (Notes 1 & 2)

If Military/Aerospace specified devices are required, 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 11)	400V

Operating Conditions (Notes 1 & 2)

Temperature Range (Note 1)	$T_{MIN} \le T_A \le T_{MAX}$
ADC0808CJ	$-55°C \le T_A \le +125°C$
ADC0808CCJ, ADC0808CCN, ADC0809CCN	$-40°C \le T_A \le +85°C$
ADC0808CCV, ADC0809CCV	$-40°C \le T_A \le +85°C$
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: ADC0808CJ $4.5V \le V_{CC} \le 5.5V$, $-55°C \le T_A \le +125°C$ unless otherwise noted ADC0808CCJ, ADC0808CCN, ADC0808CCV, ADC0809CCN and ADC0809CCV, $4.75 \le V_{CC} \le 5.25V$, $-40°C \le T_A \le +85°C$ 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

Electrical Characteristics (Continued)

Digital Levels and DC Specifications: ADC0808CJ $4.5V \le V_{CC} \le 5.5V$, $-55°C \le T_A \le +125°C$ unless otherwise noted. ADC0808CCJ, ADC0808CCN, ADC0808CCV, ADC0809CCN and ADC0809CCV, $4.75 \le V_{CC} \le 5.25V$, $-40°C \le T_A \le +85°C$ unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
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
DATA OUTPUTS AND EOC (INTERRUPT)						
$V_{OUT(1)}$	Logical "1" Output Voltage	$I_O=-360$ μA	$V_{CC}-0.4$			V
$V_{OUT(0)}$	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 $T_A=25°C$ 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	$R_S=0\Omega$ *(Figure 5)*		1	2.5	μS
t_{H1}, t_{H0}	OE Control to Q Logic State	$C_L=50$ pF, $R_L=10$k *(Figure 8)*		125	250	ns
t_{1H}, t_{0H}	OE Control to Hi-Z	$C_L=10$ pF, $R_L=10$k *(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, (Note 12)		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 on-chip 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 6)*. 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.

Functional Description

Multiplexer. The device contains an 8-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. Table I shows the input states for the address lines to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

TABLE I

SELECTED ANALOG CHANNEL	ADDRESS LINE		
	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

CONVERTER CHARACTERISTICS

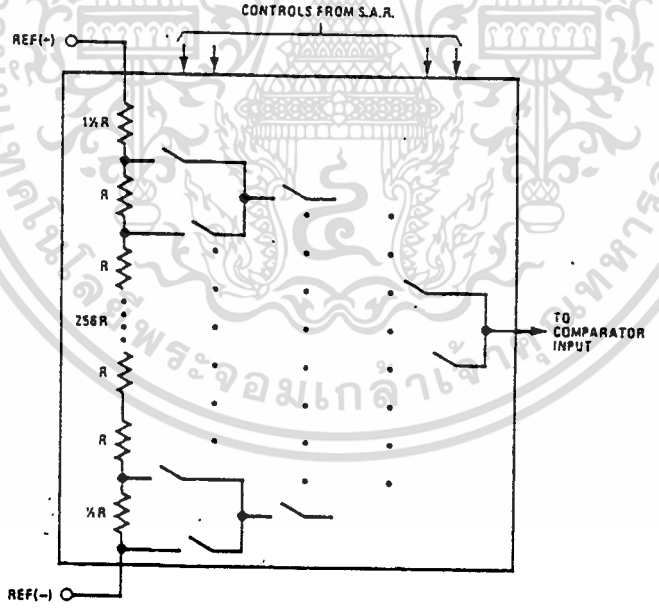
The Converter

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network, the successive approximation register, and the comparator. The converter's digital outputs are positive true.

The 256R ladder network approach *(Figure 1)* was chosen over the conventional R/2R ladder because of its inherent monotonicity, which guarantees no missing digital codes. Monotonicity is particularly important in closed loop feedback control systems. A non-monotonic relationship can cause oscillations that will be catastrophic for the system. Additionally, the 256R network does not cause load variations on the reference voltage.

The bottom resistor and the top resistor of the ladder network in *Figure 1* are not the same value as the remainder of the network. The difference in these resistors causes the output characteristic to be symmetrical with the zero and full-scale points of the transfer curve. The first output transition occurs when the analog signal has reached +½ LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

The successive approximation register (SAR) performs 8 iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter. *Figure 2* shows a typical example of a 3-bit converter. In the ADC0808, ADC0809, the approximation technique is extended to 8 bits using the 256R network.



TL/H/5672–2

FIGURE 1. Resistor Ladder and Switch Tree

Functional Description (Continued)

The A/D converter's successive approximation register (SAR) is reset on the positive edge of the start conversion (SC) pulse. The conversion is begun on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion may be accomplished by tying the end-of-conversion (EOC) output to the SC input. If used in this mode, an external start conversion pulse should be applied after power up. End-of-conversion will go low between 0 and 8 clock pulses after the rising edge of start conversion.

The most important section of the A/D converter is the comparator. It is this section which is responsible for the ultimate accuracy of the entire converter. It is also the comparator drift which has the greatest influence on the repeatability of the device. A chopper-stabilized comparator provides the most effective method of satisfying all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is then fed through a high gain AC amplifier and has the DC level restored. This technique limits the drift component of the amplifier since the drift is a DC component which is not passed by the AC amplifier. This makes the entire A/D converter extremely insensitive to temperature, long term drift and input offset errors.

Figure 4 shows a typical error curve for the ADC0808 as measured using the procedures outlined in AN-179.

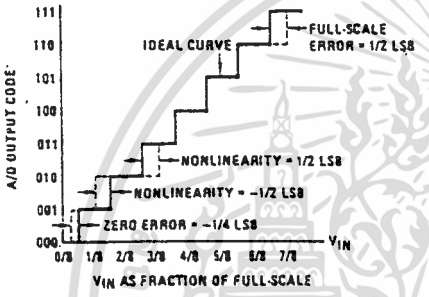


FIGURE 2. 3-Bit A/D Transfer Curve

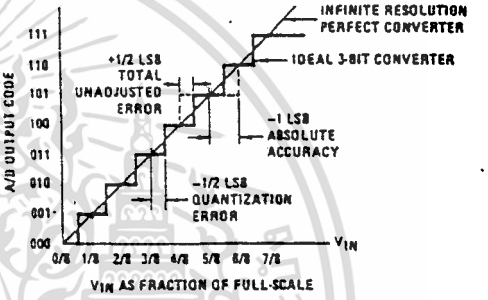
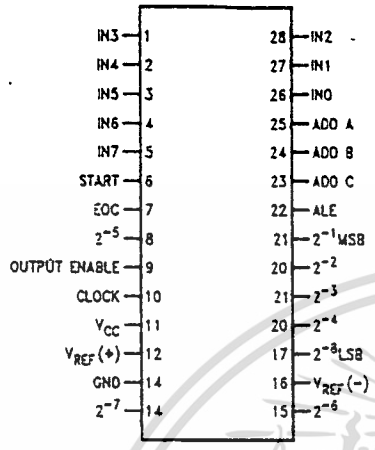


FIGURE 3. 3-Bit A/D Absolute Accuracy Curve

FIGURE 4. Typical Error Curve

Connection Diagrams

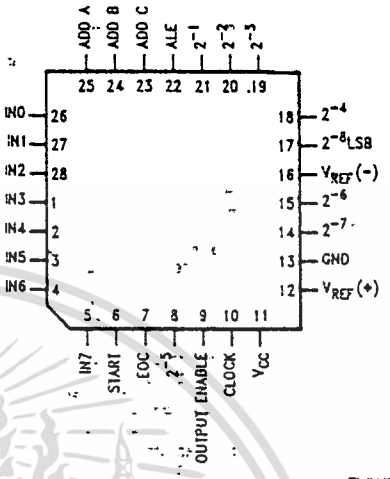
Dual-In-Line Package



TL/H/5672–11

Order Number ADC0808CCN, ADC0809CCN,
ADC0808CCJ or ADC0808CJ
See NS Package J28A or N28A

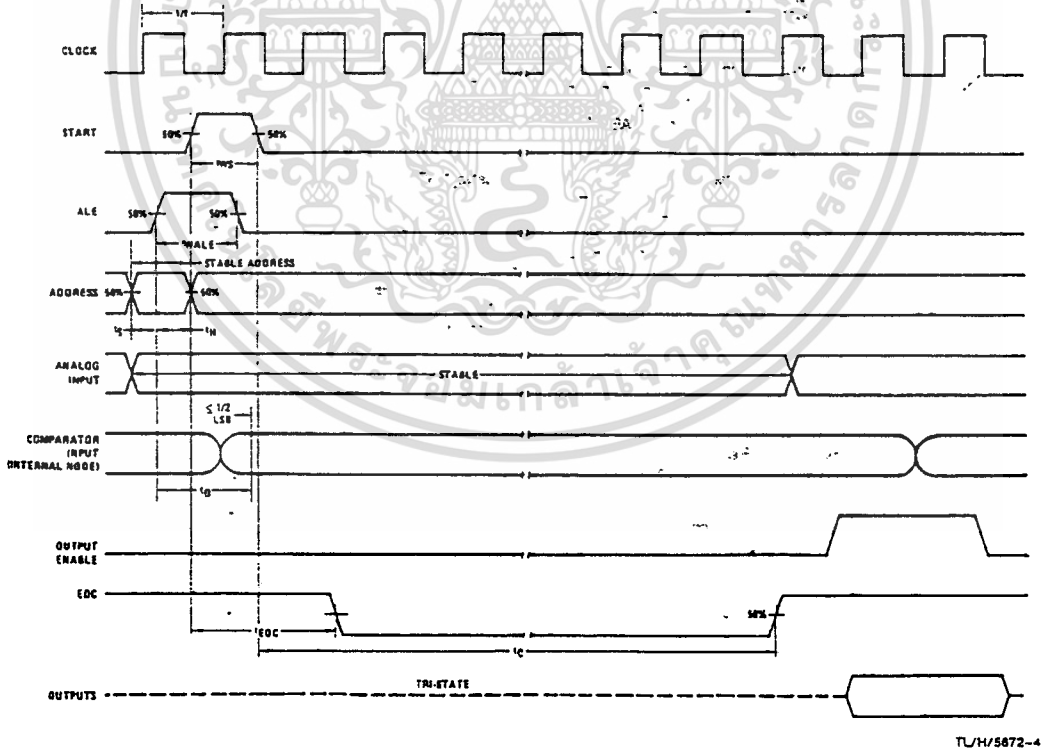
Molded Chip Carrier Package



TL/H/5672–12

Order Number ADC0808CCV or ADC0809CCV
See NS Package V28A

Timing Diagram



TL/H/5672–4

FIGURE 5

Typical Performance Characteristics

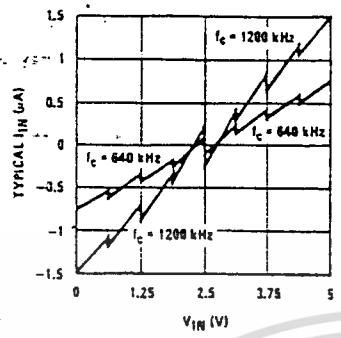
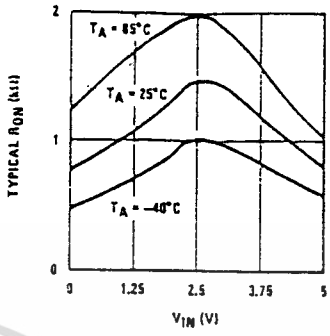


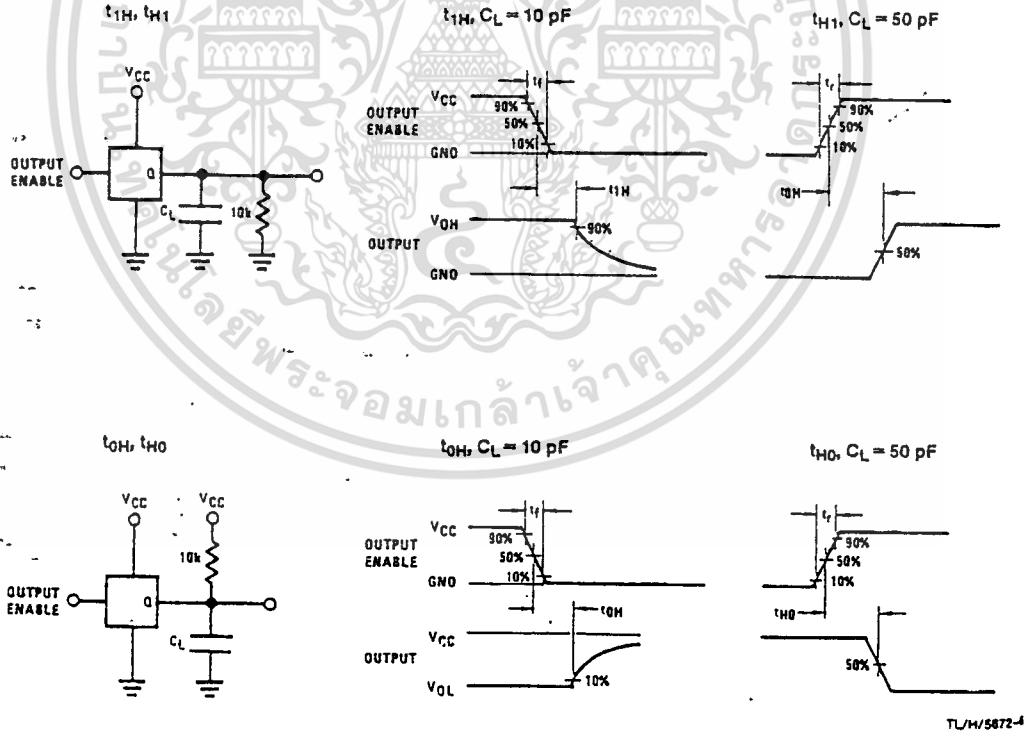
FIGURE 6. Comparator I_{IN} vs V_{IN} ($V_{CC} = V_{REF} = 5V$)



TL/H/5872–5

FIGURE 7. Multiplexer R_{ON} vs V_{IN} ($V_{CC} = V_{REF} = 5V$)

TRI-STATE Test Circuits and Timing Diagrams



TL/H/5872–4

FIGURE 8

Applications Information

OPERATION

1.0 RATIOMETRIC CONVERSION

The ADC0808, ADC0809 is designed as a complete Data Acquisition System (DAS) for ratiometric conversion systems. In ratiometric systems, the physical variable being measured is expressed as a percentage of full-scale which is not necessarily related to an absolute standard. The voltage input to the ADC0808 is expressed by the equation

$$\frac{V_{IN}}{V_{fs}-V_Z}=\frac{D_X}{D_{MAX}-D_{MIN}} \quad (1)$$

V_{IN} = Input voltage into the ADC0808

V_{fs} = Full-scale voltage

V_Z = Zero voltage

D_X = Data point being measured

D_{MAX} = Maximum data limit

D_{MIN} = Minimum data limit

A good example of a ratiometric transducer is a potentiometer used as a position sensor. The position of the wiper is directly proportional to the output voltage which is a ratio of the full-scale voltage across it. Since the data is represented as a proportion of full-scale, reference requirements are greatly reduced, eliminating a large source of error and cost for many applications. A major advantage of the ADC0808, ADC0809 is that the input voltage range is equal to the supply range so the transducers can be connected directly across the supply and their outputs connected directly into the multiplexer inputs. *(Figure 9)*.

Ratiometric transducers such as potentiometers, strain gauges, thermistor bridges, pressure transducers, etc., are suitable for measuring proportional relationships; however, many types of measurements must be referred to an absolute standard such as voltage or current. This means a system reference must be used which relates the full-scale voltage to the standard volt. For example, if $V_{CC}=V_{REF}=5.12V$, then the full-scale range is divided into 256 standard steps. The smallest standard step is 1 LSB which is then 20 mV.

2.0 RESISTOR LADDER LIMITATIONS

The voltages from the resistor ladder are compared to the selected into 8 times in a conversion. These voltages are coupled to the comparator via an analog switch tree which is referenced to the supply. The voltages at the top, center and bottom of the ladder must be controlled to maintain proper operation.

The top of the ladder, Ref(+), should not be more positive than the supply, and the bottom of the ladder, Ref(−), should not be more negative than ground. The center of the ladder voltage must also be near the center of the supply because the analog switch tree changes from N-channel switches to P-channel switches. These limitations are automatically satisfied in ratiometric systems and can be easily met in ground referenced systems.

Figure 10 shows a ground referenced system with a separate supply and reference. In this system, the supply must be trimmed to match the reference voltage. For instance, if a 5.12V is used, the supply should be adjusted to the same voltage within 0.1V.

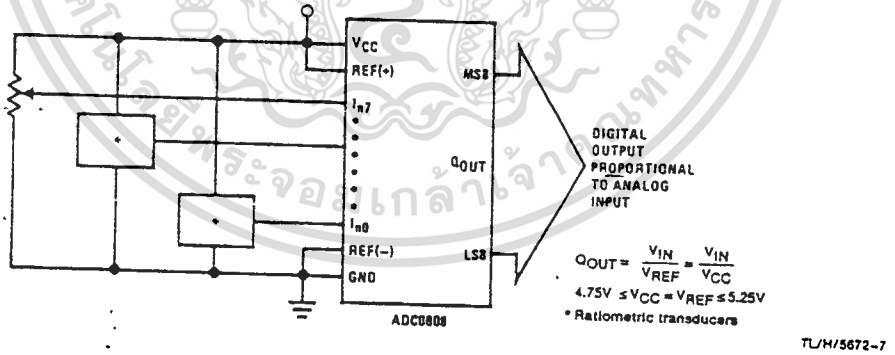


FIGURE 9. Ratiometric Conversion System

Applications Information (Continued)

The ADC0808 needs less than a milliamp of supply current so developing the supply from the reference is readily accomplished. In *Figure 11* a ground referenced system is shown which generates the supply from the reference. The buffer shown can be an op amp of sufficient drive to supply the milliamp of supply current and the desired bus drive, or if a capacitive bus is driven by the outputs a large capacitor will supply the transient supply current as seen in *Figure 12*. The LM301 is overcompensated to insure stability when loaded by the .10 μF output capacitor.

The top and bottom ladder voltages cannot exceed V_{CC} and ground, respectively, but they can be symmetrically less than V_{CC} and greater than ground. The center of the ladder voltage should always be near the center of the supply. The sensitivity of the converter can be increased, (i.e., size of the LSB steps decreased) by using a symmetrical reference system. In *Figure 13*, a 2.5V reference is symmetrically centered about $V_{CC}/2$ since the same current flows in identical resistors. This system with a 2.5V reference allows the LSB bit to be half the size of a 5V reference system.

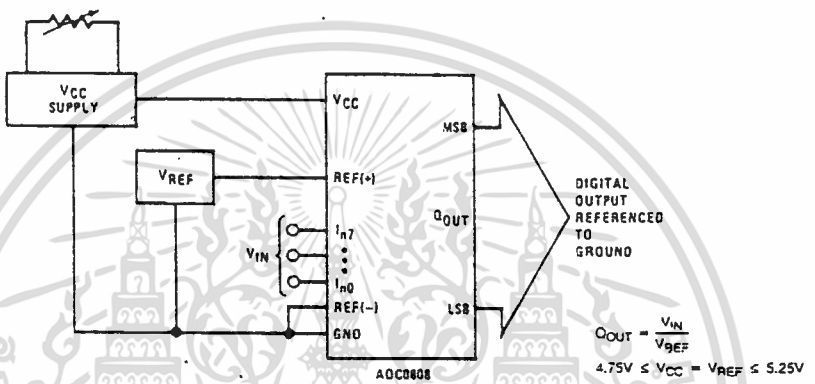


FIGURE 10. Ground Referenced Conversion System Using Trimmed Supply

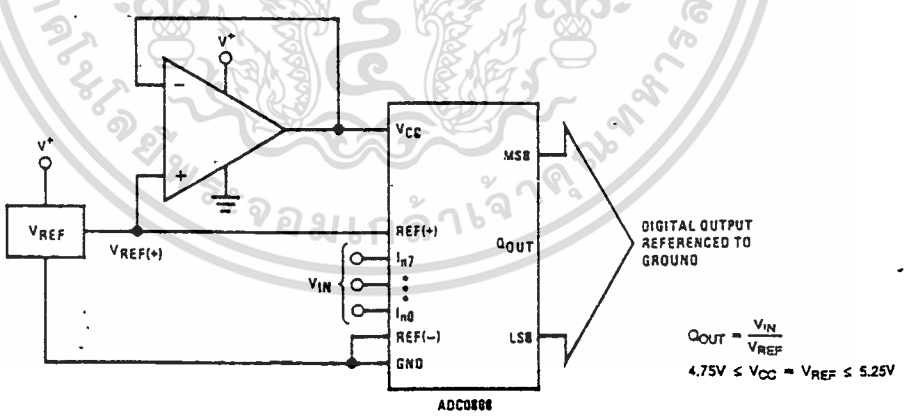


FIGURE 11: Ground Referenced Conversion System with Reference Generating V_{CC} Supply

Applications Information (Continued)

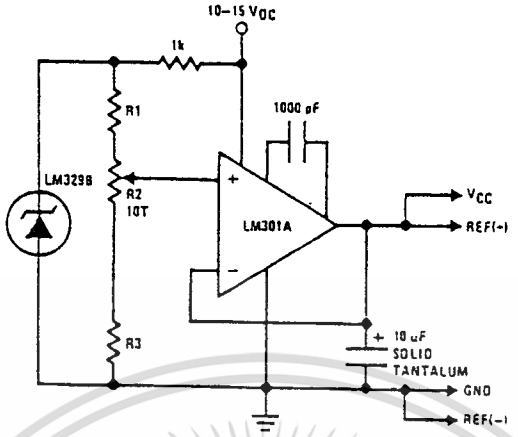
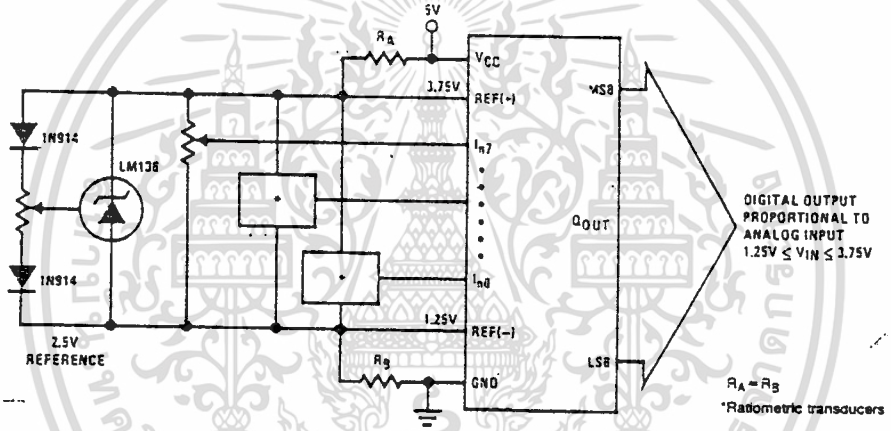


FIGURE 12. Typical Reference and Supply Circuit



TL/H/5672–9

FIGURE 13. Symmetrically Centered Reference

3.0 CONVERTER EQUATIONS

The transition between adjacent codes N and N + 1 is given by:

$$V_{IN} = \left\{ (V_{REF(+)} - V_{REF(-)}) \left[\frac{N}{256} + \frac{1}{512} \right] \pm V_{TUE} \right\} + V_{REF(-)} \quad (2)$$

The center of an output code N is given by:

$$V_{IN} \left\{ (V_{REF(+)} - V_{REF(-)}) \left[\frac{N}{256} \right] \pm V_{TUE} \right\} + V_{REF(-)} \quad (3)$$

The output code N for an arbitrary input are the integers within the range:

$$N = \frac{V_{IN} - V_{REF(-)}}{V_{REF(+)} - V_{REF(-)}} \times 256 \pm \text{Absolute Accuracy} \quad (4)$$

where: V_{IN} = Voltage at comparator input

$V_{REF(+)}$ = Voltage at Ref(+)

$V_{REF(-)}$ = Voltage at Ref(−)

V_{TUE} = Total unadjusted error voltage (typically $V_{REF(+)} \div 512$)

4.0 ANALOG COMPARATOR INPUTS

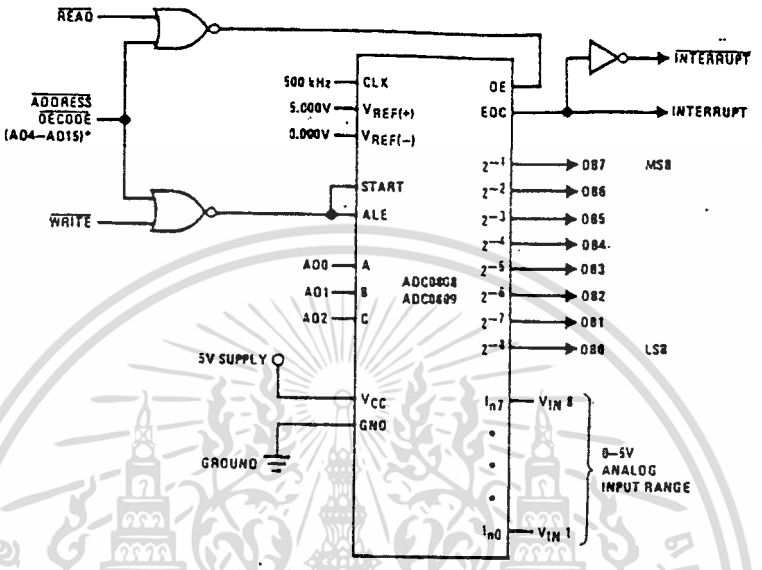
The dynamic comparator input current is caused by the periodic switching of on-chip stray capacitances. These are connected alternately to the output of the resistor ladder/switch tree network and to the comparator input as part of the operation of the chopper stabilized comparator.

The average value of the comparator input current varies directly with clock frequency and with V_{IN} as shown in *Figure 6.*

If no filter capacitors are used at the analog inputs and the signal source impedances are low, the comparator input current should not introduce converter errors, as the transient created by the capacitance discharge will die out before the comparator output is strobed.

If input filter capacitors are desired for noise reduction and signal conditioning they will tend to average out the dynamic comparator input current. It will then take on the characteristics of a DC bias current whose effect can be predicted conventionally.

Typical Application



TL/H/5672–10

*Address latches needed for 8085 and SC/MP interfacing the ADC0808 to a microprocessor

MICROPROCESSOR INTERFACE TABLE

PROCESSOR	READ	WRITE	INTERRUPT (COMMENT)
8080	MEMR	MEMW	INTR (Thru RST Circuit)
8085	RD	WR	INTR (Thru RST Circuit)
Z-80	RD	WR	INT (Thru RST Circuit, Mode 0)
SC/MP	NRDS	NWDS	SA (Thru Sense A)
6800	VMA•φ2•R/W	VMA•φ•R/W	IRQA or IRQB (Thru PIA)

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

NEWLOG.PRJ

main.c
save.c
openfile.c
diskerr.c
config.c
run.c
sampling.c
vcoords.c
tui.c
logicdrv.c

LOG.H

```
#include <stdio.h>
#include "tui.h"

typedef unsigned long  ulong;

#define  SetFlag(flag,bit)    flag=flag|bit
#define  ResetFlag(flag,bit)  flag=flag&~bit

#define  CNFGready       0x01
#define  DTsaved         0x80

#define  MAINB     BROWN
#define  MAINF     BLUE
#define  HTAB      BLACK
#define  BTAB      MAGENTA
#define  MARC      LIGHTGRAY
#define  ENAC      MAINF
#define  DISC      BROWN
#define  WARNF     YELLOW
#define  WARNB     BLUE

enum message { OK, FAIL };

#define  WTYPE  12
#define  WMAX   6
#define  WMIN   6
#define  WUNIT  8
#define  ON          1
#define  OFF         0

typedef struct
{  char   type[WTYPE+1];
   char   max[WMAX+1];
   char   min[WMIN+1];
   char   unit[WUNIT+1];
   uchar  stat;
}  CHANNEL;

typedef struct
{  uchar  hr;
   uchar  min;
   uchar  sec;
   uchar  msec;
}  TIME;

typedef struct
{  char    Comment[80];
   CHANNEL  Chan[8];
   uchar    openChan;
   uchar    aveNo;
   uchar    recRateNo;
   ulong    recAmount;
   TIME     recTime;
}  CONFIG;
```

```
/*-------------------------------------------------------------------*/
#define SecPerTick    0.002          /* seccond in one  clock tick */
#define TickPerSec    500       /* amount of ticks in 1 secound */
#define COUNTPERTICK  2386           /* 8253 count per tick */
#define USECPERCLOCK  0.838096515    /* Microsecound period in one clock */
#define COUNTREG      0x0040         /* First counter register */
#define CTRLCOUNT     0x0043         /* Timer control register */
/*-------------------------------------------------------------------*/
```

MAIN.C

```
#include <stdio.h>
#include <bios.h>
#include <stdarg.h>
#include <errno.h>
#include <stdlib.h>
#include <alloc.h>
#include <dir.h>
#include "tui.h"
#include "log.h"

TAB  mainMenu[] = {  { " File ",   'F',  4, 3,  6, ENABLE, NULL },
                     { " Run ",    'R', 12, 3,  5, DISABLE, NULL },
                     { " System ", 'S', 19, 3,  8, ENABLE, NULL }
                   };

TAB  fileMN[] = { { " Save..   ", 'S', 0, 0, 0, ENABLE,
                    " Save data from memory to file " },
                  { " DOS      ", 'D', 0, 0, 0, ENABLE,
                    " Suspend to DOS " },
                  { " Quit     ", 'Q', 0, 0, 0, ENABLE,
                    " Exit programe " }
                  };

TAB  runMN[] = { { "  Run..  ", 'R', 0, 0, 0, ENABLE,
                   " Perform operation " }
                 };

TAB  sysMN[] = { { " About     ", 'A', 0, 0, 0, ENABLE,
                   " System preface " },
                 { " Help..    ", 'H', 0, 0, 0, DISABLE,
                   " System Information " },
                 { " Config..  ", 'C', 0, 0, 0, ENABLE,
                   " Setup system configuration " }
                 };

CONFIG      cnfg;
uchar huge  *Data;


main()
{  uchar     ENVflag;
   TABCOLOR  TC = { HTAB, BTAB, MARC, ENAC, DISC };

   clrscr();
   InitVid();
   CursorOff();
   SetTabColor( &TC );
   DrawMainScreen();
   About();
   if( !BuiltMNBR(  3, mainMenu, 3, fileMN, 3, runMN, 1, sysMN, 3 ) )
     exit( 0 );
   SetFlag( ENVflag, DTsaved );
   ResetFlag( ENVflag, CNFGready );
   for( ; ; )
   {  switch( MenuBar() )
```

```
        { case  1: if( Save() == OK )
                     SetFlag( ENVflag, DTsaved );
                   break;
          case  2: OSshell();
                   break;
          case  4: RunManage( &ENVflag );
                   break;
          case  5: About();
                   break;
          case  6:
          case  7: ConfigManage( &ENVflag );
                   break;
          case  3:
          default: ExitManage( &ENVflag );
} } }

DrawMainScreen()
{  SetBackColor( 1, 1, 80, 25, MAINB );
   SetForeColor( 1, 1, 80, 25, MAINF );
   CenterString( 1, "\4 Data Logger \4" );
   SetBackColor( 1, 3, 80, 3, BTAB );
   PutMenu( mainMenu, 3 );
   PutPlate( 1, 4, 80, 24, 178 );
}

PutPlate( int ix, int iy, int ex, int ey, int style )
{  int  i, j;
   for( i = iy; (i <= ey) && (i < 26); i++ )
     for( j = ix; (j <= ex) && (i < 81); j++ )
          Putcxy( j, i, style );
}

RunManage( uchar *ENVflag )
{  if( !(*ENVflag & CNFGready ) )
   {  switch( Ask( 3, WARNF, WARNB, 2, "Configuration not ready !",
                                       " Set configuration before ? " ) )
      {  case  1: if( Config() == OK )
                    SetFlag( *ENVflag, CNFGready );
                  else
                    return;
          case  2: break;
          default: return;
   } }
   if( !(*ENVflag & DTsaved) )
   {  switch( Ask( 3, WARNF, WARNB, 3, " WARNING ",
                    " Old data not ready saved ! ",
                    " Old data will lost. Save data before ? " )   )
      {  case  1: if( Save() != OK )
                    return;
                  else
                    SetFlag( *ENVflag, DTsaved );
          case  2: break;
          default: return;
   } }
   farfree( (void far *)Data );
   Data = (uchar huge *)farmalloc( (ulong)cnfg.openChan*cnfg.recAmount );
   if( Data == NULL )
     return;
```

```
  if( Run() == OK )
    ResetFlag( *ENVflag, DTsaved );
}

ConfigManage( uchar *ENVflag )
{ if( !(*ENVflag & DTsaved) )
  { switch( Ask( 3, WARNF, WARNB, 3, " WARNING ",
                                " Old data not ready saved ! ",
                                " Save data before ? " )   )
    { case  1: if( Save() != OK )
                  return;
                else
                  SetFlag( *ENVflag, DTsaved );
          case  2: break;
          default: return;
  } }
  farfree( (void far *)Data );
  if( Config() == OK )
    SetFlag( *ENVflag, CNFGready );
}

ExitManage( uchar *ENVflag )
{ if( Ask( 2, WARNF, WARNB, 1, " Exit programe ? " ) == 1 )
  { if( !(*ENVflag & DTsaved) )
    { switch( Ask( 3, WARNF, WARNB, 3, " WANING ",
                                " Data not ready saved ! ",
                                " Save before exit ? "  ) )
        { case  1: if( Save() != OK )
                      return;
          case  2: break;
          default: return;
    } }
    farfree( (void far *)Data );
    clrscr ();
    CursorOn();
    exit(0);
} }

OSshell()
{ char  scrnBuf[ 80*25*2 ];

  gettext( 1, 1, 80, 25, scrnBuf );
  clrscr();
  CursorOn();
  printf( " Type EXIT to return... \n" );
  errno = EZERO;
  system( NULL );
  if( errno != EZERO )
  { printf( " %s : hit any key to return ", sys_errlist[errno] );
    getch();
  }
  CursorOff();
  puttext( 1, 1, 80, 25, scrnBuf );
}

About()
{ Ask( 1, BLACK, GREEN, 13,
    "Data Logger",
```

```
        NULL,
        " Software for driving Analog to Digital Converter through LPT1 port. ",
        NULL,
        "This Software and it's Hardware belong to",
        "Agricultural Engineering of KMITL",
        NULL,
        "Developed by",
        "Panumas Thongtanunam 34128027",
        "Patama  Temwootiroge 34128017",
        NULL,
        "Advisor",
        "Assit. Prof. Panmanus Sirisombroon"
    );
}
```

SAVE.C

```
#include <dir.h>
#include <dos.h>
#include "tui.h"
#include "log.h"

extern CONFIG      cnfg;
extern uchar huge  *Data;

long  RecAvailSave();

Save()
{  BOARD  brd = { 12, 7, 68, 9, DOUBLE, NORMAL, BLACK, BROWN, NULL };
   FILE  *fp;
   char   path[MAXPATH];
   uchar  flag;

   BuiltBoard( &brd );
   CenterString( brd.iy, "Save" );
   PutString( brd.ix + 1, brd.iy + 1, " File's name: " );
   flag = FileOpenWrite( &fp, strcpy( path, "*.CSV" ), brd.ix+15, brd.iy+1 );
   if( flag == OK )
   {  flag = SaveManage( fp, path );
      fclose( fp );
   }
   EraseBoard( &brd );
   return flag;
}

SaveManage( FILE *fp, char *path )
{  struct dfree  diskFree;
   long        freeSpace;

   getdfree( path[0] - 'A' + 1, &diskFree );
   if( (signed)diskFree.df_sclus == -1 )
   {  Ask( 1, BLUE, GREEN, 2, "ERROR", " Can't find disk free space " );
      return 0;
   }
   freeSpace = (long) diskFree.df_avail * diskFree.df_sclus * diskFree.df_bsec;
   if( SaveHead( fp, &freeSpace ) == FAIL )
      return FAIL;
   if( cnfg.recAmount == 0 )
      return OK;
   if( SaveData( fp, &freeSpace ) == FAIL )
      return FAIL;
   return OK;
}

#define  FILE_HEAD_LEN              208  /* LENGH of TOP HEAD count from start file
*/
#define  FILE_CHANFIELD_LEN          (WTYPE+1)
#define  FILE_CHAN_SIZE             FILE_CHANFIELD_LEN*5
#define  FILE_CHANTABLE_SIZE( nchan )  FILE_CHAN_SIZE*nchan+10

SaveHead( FILE *fp, long  *freeSpace )
{  *freeSpace -= ( FILE_HEAD_LEN + FILE_CHANTABLE_SIZE( cnfg.openChan ) );
```

```
    if( *freeSpace < 0L )
      return FAIL;
    WriteHead( fp );
    return OK;
  }

  WriteHead( FILE *fp )
  { uchar i;
    extern TAB rateMenu[];
    extern TAB aveMenu[];

    fprintf( fp, "      Comment:, %-60s,\n", cnfg.Comment );
    fprintf( fp, " Total Reccord:, %-6ld,\n", cnfg.recAmount );
    fprintf( fp, "Recording Rate:, %11s,\n", rateMenu[cnfg.recRateNo].item );
    fprintf( fp, "     Averaged:, %5s, time,\n", aveMenu[cnfg.aveNo].item );
    fprintf( fp, "   Resolution:, 1/256, of Span,\n" );
    fprintf( fp, " Used channels:, %d,\n", cnfg.openChan );
    for( i = 0; i < 8; i++ )
      if( cnfg.Chan[i].stat )
          fprintf( fp, "%*d,", WTYPE, i+1 );
    fprintf( fp, "\n" );
    for( i = 0; i < 8; i++ )
      if( cnfg.Chan[i].stat )
          fprintf( fp, "%*s,", WTYPE, cnfg.Chan[i].type );
    fprintf( fp, "\n" );
    for( i = 0; i < 8; i++ )
      if( cnfg.Chan[i].stat )
          fprintf( fp, "%*s,", WTYPE, cnfg.Chan[i].max );
    fprintf( fp, "\n" );
    for( i = 0; i < 8; i++ )
      if( cnfg.Chan[i].stat )
          fprintf( fp, "%*s,", WTYPE, cnfg.Chan[i].min );
    fprintf( fp, "\n" );
    for( i = 0; i < 8; i++ )
      if( cnfg.Chan[i].stat )
          fprintf( fp, "%*s,", WTYPE, cnfg.Chan[i].unit );
    fprintf( fp, "\n" );
  }

  SaveData( FILE *fp, long *freeSpace )
  { long recAvailSave;
    char msg[70];

    recAvailSave = RecAvailSave( freeSpace );
    if( recAvailSave < cnfg.recAmount )
    { sprintf( msg, " Only record 1 to %ld of %ld can save, continue ? ",
                    recAvailSave, cnfg.recAmount );
      if( Ask( 2, BLUE, GREEN, 2, "Not enought disk free space !", msg ) != 1 )
          return 0;
      else
          cnfg.recAmount = recAvailSave;
    }
    WriteData( fp );
    return OK;
  }

  long RecAvailSave( long *freeSpace )
  { long   i;
```

```
    register uchar  j;
    BOARD   brd = { 22, 10, 59, 12, SINGLE, NORMAL, BLUE, GREEN, NULL };

    BuiltBoard( &brd );
    PutString( brd.ix+1, brd.iy+1, " Checking file's size, please wait. " );
    for( i = 0L; i < cnfg.recAmount; i++ )
    {  for( j = 0; j < cnfg.openChan; j++ )
       {  if( Data[ i*cnfg.openChan+j ] < 10 )
             *freeSpace -= 2;              /* 1 digit */
          else if( Data[ i*cnfg.openChan+j] < 99 )
             *freeSpace -= 3;              /* 2 digit */
          else
             *freeSpace -= 4;           /* if more than 99, then 3 digit */
       }
       *freeSpace -= 2;              /* "\r\n"   */
       if( *freeSpace < 0 )
          break;
    }
    EraseBoard( &brd );
    return i;
}

WriteData( FILE *fp )
{  uchar  i;
   long   j;
   char   msg[35];
   int    breakFlag = 0;
   BOARD  brd = { 18, 10, 63, 15, SINGLE, FLOAT, BLUE, GREEN, NULL };

   BuiltBoard( &brd );
   PutString( brd.ix+19, brd.iy, " SAVING " );
   PutString( brd.ix+2,  brd.iy+2, " Record No.          % ");
   sprintf( msg, "of%7ld Record", cnfg.recAmount );
   PutString( brd.ix+27, brd.iy+2, msg );
   PutString( brd.ix+19, brd.ey, " B-Break " );
   SetBackColor( brd.ix+3, brd.iy+3, brd.ex-3, brd.iy+3, brd.back+1 );
   for( j = 0L; j < cnfg.recAmount; j++ )
   {  for( i = 0; i < cnfg.openChan; i++ )
         fprintf( fp, "%d,", Data[ (ulong)j*(ulong)cnfg.openChan + (ulong)i ] );
      fprintf( fp, "\n" );

      sprintf( msg, "%ld", j+1L );
      PutString( brd.ix+14, brd.iy+2, msg );

      sprintf( msg, "%3d", (int)( ( (float)(j+1L) / cnfg.recAmount ) *100 ) );
      PutString( brd.ix+22, brd.iy+2, msg );

      Putcxy( brd.ix+3+((float)j/cnfg.recAmount)*40, brd.iy+3, 178 );

      if( kbhit() && ( toupper( bioskey(0) ) == 'B' ) )
      {  sprintf( msg, " record %ld to %ld not saved ! ", j+2, cnfg.recAmount );
         if( Ask( 2, BLUE, GREEN, 3, "USER BREAK", msg, " Continue save ?" ) != 1 )
         {  j++;
            breakFlag = 1;    break;
   } } }

   cnfg.recAmount = j;
```

```
    if( !breakFlag )
    { PutString( brd.ix+11, brd.ey, " Hit any key to continue " );
      while( !kbhit() );
      bioskey(0);
    }
    EraseBoard( &brd );
}
```

OPENFILE.C

```
#include <stdio.h>
#include <dir.h>
#include <stdlib.h>
#include <errno.h>
#include <dos.h>
#include "tui.h"
#include "log.h"

typedef struct
{ char *p;
  char d[MAXDRIVE];
  char s[MAXDIR];
  char n[MAXFILE];
  char e[MAXEXT];
} FNAME;

extern int TellHardError();
void TellChangeDisk();

FileOpenRead( FILE **fp, char *path, int x, int y )
{ FNAME F;
  uchar flag;

  harderr( TellHardError );
  F.p = path;
  if( ArrangeFullPathName( &F ) == FAIL )
    return FAIL;
  for( ; ; )
  { if( GetFullPathName( &F, x, y ) == FAIL )
    { flag = FAIL;
        break;
    }
    flag = fnsplit( F.p, F.d, F.s, F.n, F.e );
    if( (flag & FILENAME) && !(flag & WILDCARDS) && (_chmod( F.p, 0 ) !=
FA_DIREC) )
    { flag = OK;
        break;
    }
    if( SelectFullPathName( &F, x, y ) == FAIL )
    { flag = FAIL;
        break;
    }
    flag = fnsplit( F.p, F.d, F.s, F.n, F.e );
    if( (flag & FILENAME) && !(flag & WILDCARDS) )
    { flag = OK;
        break;
  } }
  if( flag == FAIL )
    return FAIL;
  *fp = fopen( F.p, "r" );
  if( *fp != NULL )
    return OK;
  else
  { Ask( 1, BLUE, BROWN, 3, "!!!", sys_errlist[errno], "can't open file" );
    return FAIL;
} }
```

```
FileOpenWrite( FILE **fp, char *path, int x, int y )
{ FNAME  F;

  harderr( TellHardError );
  F.p = path;
  if( ArrangeFullPathName( &F ) == FAIL )
    return FAIL;
  if( GetFullPathName( &F, x, y ) == FAIL )
    return FAIL;
  if( searchpath( F.p ) != NULL )
  { if( Ask( 2, BLUE, BROWN, 3, "WARNING", F.p, "existing, over write ?" )!= 1 )
        return FAIL;
  }
  *fp = fopen( F.p, "w+" );
  if( *fp != NULL )
    return OK;
  else
  { Ask( 1, BLUE, BROWN, 3, F.p, sys_errlist[errno], "can't open file !" );
    return FAIL;
} }

GetFullPathName( FNAME *f, int x, int y )
{ if( !GetNChar( x, y, 40, f->p ) )
    return FAIL;
  if( ArrangeFullPathName( f ) == FAIL )
    return FAIL;
  ClearText( x, y, x+39, y );
  PutString( x, y, f->p );
  return OK;
}

ArrangeFullPathName( FNAME *f )
{ uchar  flag;
  char   tem[MAXDIR];

  strupr( f->p );
  flag = fnsplit( f->p, f->d, f->s, f->n, f->e );
  if( !(flag & DRIVE) )     /* if no spec. drive then current drive */
  { f->d[0] = getdisk() + 'A';
    f->d[1] = ':';
  }
  set_logical_drive( f->d[0], TellChangeDisk );

  if( !(flag & DIRECTORY) )
    if( getcurdir( f->d[0] - 'A' + 1, f->s ) == -1 )
        return FAIL;

  if( f->s[0] != '\\' )
  { strcpy( tem, f->s );
    strcpy( f->s, "\\" );
    strcat( f->s, tem );
  }
  if( f->s[ strlen( f->s ) - 1 ] != '\\' )
    strcat( f->s, "\\" );
  fnmerge( f->p, f->d, f->s, f->n, f->e );
  if( _chmod( f->p, 0 ) == FA_DIREC )
    if( f->p[ strlen( f->p ) - 1 ] != '\\' )
```

```
            strcat( f->p, "\\" );
    flag = fnsplit( f->p, f->d, f->s, f->n, f->e );
    if( !(flag & FILENAME) )
    {  strcpy( f->n, "*" );
       if( !(flag & EXTENSION) )
            strcpy( f->e, ".*" );
    }
    else
    {  if( !strcmp( f->n, "*" ) && !(flag & EXTENSION) )
            strcpy( f->e, ".*" );
    }
    fnmerge( f->p, f->d, f->s, f->n, f->e );
    return OK;
}

void  TellChangeDisk( char drive )
{  char  msg[] = "Insert new disk for drive  : and press Enter.";

   msg[26] = drive;
   Ask( 1, BLUE, BROWN, 1, msg );
}

SelectFullPathName( FNAME  *f, int x, int y )
{  char          tem[MAXFILE+MAXEXT+1];
   uchar         flag;
   char far      *oldDTA;
   int           i;
   int           n;
   char          **menu;
   TABCOLOR      oldTColor, TColor = { RED, BLUE, GREEN, MAGENTA, BLACK };
   BOARD         brd = { 0, 0, 0, 0, SINGLE, FLOAT, BLUE, BROWN, NULL };
   uchar         boardExist = 0;

   brd.ix = x, brd.iy = y + 1, brd.ex = x + 15, brd.ey = y + 12;
   SetScrollPage( brd.ix + 1, brd.iy + 1, 14, 10 );
   GetTabColor( &oldTColor );
   SetTabColor( &TColor );
   oldDTA = getdta();
   for(;;)
   {  ClearText( x, y, x+39, y ) ;
      PutString( x, y, f->p );
      n = CountMatch( f->p );
      if( !n )
      {  Ask( 1, BLUE, BROWN, 1, sys_errlist[errno] );
         flag = OK;
         break;
      }

      menu = (char **)malloc( sizeof( char * ) * n );
      for( i = 0; i < n; i++ )
         menu[i] = (char *)malloc( 15 );

      BuiltMatch( f->p, menu );

      if( !boardExist )
      {  BuiltBoard( &brd );
         boardExist = 1;
      }
```

```
        i = ScrollMenu( menu, n, 1 );
        strcpy( tem, (i > 0)? &menu[i-1][1] : NULL );
        flag = (i > 0)? 1 : 0;
        for( i = 0; i < n; i++ )
            free( menu[i] );
        free( menu );
        if( flag )
        {  if( tem[ strlen( tem ) -1 ] == '\\' )
            {  if( !strcmp( tem, "..\\" ) )  /* reduce directory */
               {  strrchr( f->s, '\\' )[0] = NULL;
                  strrchr( f->s, '\\' )[1] = NULL;
               }
               else   /* add directory */
                  strcat( f->s, tem );
               fnmerge( f->p, f->d, f->s, f->n, f->e );
            }
            else
            {  fnmerge( f->p, f->d, f->s, NULL, NULL );
               strcat( f->p, tem );
               flag = OK;
               break;
        } }
        else
        {  flag = OK;
            break;
        }
    }
    setdta( oldDTA );
    EraseBoard( &brd );
    SetTabColor( &oldTColor );
    return  flag;
}

CountMatch( char *path )
{  struct ffblk  f;
   int         n;
   int         flag;
   FNAME       tem;
   char        buf[MAXPATH];

   tem.p = buf;
   fnsplit( path, tem.d, tem.s, tem.n, tem.e );
   fnmerge( tem.p, tem.d, tem.s, "*", NULL );

   n = 0;
   flag =  findfirst( tem.p, &f, FA_DIREC );
   if( !flag )
   {  if( strcmp( f.ff_name, "." ) && (f.ff_attrib == FA_DIREC) )
          n++;
      while( flag == 0 )
      {  flag = findnext( &f );
          if( !flag && (f.ff_attrib == FA_DIREC) )
           n++;
   } }

   flag = findfirst( path, &f, FA_ARCH );
   if( !flag )
   {  n++;
```

```
        while( flag == 0 )
        {  flag = findnext( &f );
              if( !flag )
                n++;
      } }
      return n;
}

BuiltMatch( char *path, char **menu )
{  struct ffblk f;
   int        flag;
   int        i;
   FNAME      tem;
   char       p[MAXPATH];

   tem.p = p;
   fnsplit( path, tem.d, tem.s, tem.n, tem.e );
   fnmerge( tem.p, tem.d, tem.s, "*", NULL );

   i = 0;
   flag =  findfirst( tem.p, &f, FA_DIREC );
   if( !flag )
   {  if( strcmp( f.ff_name, "." ) && (f.ff_attrib == FA_DIREC ) )
      {  sprintf( menu[i], " %s\\", f.ff_name );
            i++;
      }
      while( flag == 0 )
      {  flag = findnext( &f );
            if( !flag && (f.ff_attrib == FA_DIREC) )
            {  sprintf( menu[i], " %s\\", f.ff_name );
               i++;
   } } }

   flag = findfirst( path, &f, FA_ARCH );
   if( !flag )
   {  sprintf( menu[i], " %s", f.ff_name );
      i++;
      while( flag == 0 )
      {  flag = findnext( &f );
            if( !flag )
            {  sprintf( menu[i], " %s", f.ff_name );
               i++;
   } } }
}
□
```

```
/*  This program will trap disk errors and prompt the user for action.
    Try running it with no disk in drive A: to invoke its functions.
*/
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include "frame.h"

#define IGNORE  0
#define RETRY   1
#define ABORT   2

/* define the error messages for trapping disk problems  */
static char *err_msg[] =
{   "write protect",
    "unknown unit",
    "drive not ready",
    "unknown command",
    "data error (CRC)",
    "bad request",
    "seek error",
    "unknown media type",
    "sector not found",
    "printer out of paper",
    "write fault",
    "read fault",
    "general failure",
    "reserved",
    "reserved",
    "invalid disk change"
};

#pragma warn -par
int handler( int errval, int ax, int bp, int si )
{  static char  msg[80];
   unsigned     di;
   int          drive;
   int          errorno;

   di = _DI;
   if( ax < 0 )  /* if this is not a disk error, then it was another device */
   {  printf( "Device error" ); /* report the error */
      getch();
      hardretn( ABORT );/* and return to the program directly requesting abort */
   }
   /* otherwise it was a disk error */
   drive = ax & 0x00FF;    /* = ah :drive */
   errorno = di & 0x00FF;
   sprintf( msg, "\r\n %s\r\n on drive %c", err_msg[errorno], 'A' + drive );
            /* report which error it was */

   result = Tellerr( 25, 9, msg )
   hardresume( result );/* return to the program via dos interrupt 0x23
                           with abort, retry, or ignore as input by the user.*/
```

```
  return( ABORT );
}
#pragma warn +par

Tellerr( int x, int y, char *warn )
{  struct  text_info   tinfo;
   char    screen[32*5*2];
   char   *select[]  = { " RETRY ", " ABORT " };
   int     choice;
   gettextinfo( &tinfo );
   gettext( x, y, x+31, y+4, screen);
   Frame(   x, y, x+31, y+4, SINGLE );
   Paintstr( x+13, y, " ERROR ", YELLOW, RED );
   window( x+1, y+1, x+30, y+1 );
   cprintf( "%s", warn );
   choice = getresp( x+8, y+2, select, "ra", 8, 2 );
   puttext( x, y, x+31, y+4, screen );
   window( tinfo.winleft, tinfo.wintop, tinfo.winright, tinfo.winbottom );
   return  ( choice == 1 ) ? 1 : 0;
}
```

LOGICDRV.C

```
#include <ctype.h>
#include <dir.h>
#include <dos.h>
#include <errno.h>
#include <stdio.h>

#define DOS_DATA_SEG  0x005
char  get_logical_drive();
char  set_logical_drive();


/* Attaches logical drive to block device, if not already attached */
char  set_logical_drive( char drive, void (*message)() )
{  char  cur_drive;     /* Current drive attached to block device */
   union REGS  regs;    /* Registers for interupt calls */

   drive = drive?toupper(drive):(getdisk()+'A'); /* Default to current drive */

   if( (cur_drive = get_logical_drive( drive=toupper( drive ) ) )
   != 0 && cur_drive != drive)
   {  message( drive );
      if( _osmajor > 3 || _osmajor == 3 && _osminor >= 20  )
      {  regs.x.ax = 0x440F; /* IOCTL request 0x0F ( set logical drive) */
            regs.h.bl = 1 + drive - 'A'; /* Drive No. ( 1=A, 2=B, etc ) is in BL */
            intdos( &regs, &regs );
            if( regs.x.cflag )    /* Carry flag set, error in _doserrno. */
               drive = 0;
      }
      else
      {  if( drive == 'A' || drive == 'B' )    /* For lower versions of DOS, only drive A and B are
interchaneable */
               pokeb( DOS_DATA_SEG, 0x0004, drive - 'A' );
   } }
   return( cur_drive );
}

/* Returns drive currently attached block device defined by 'drive' */
char get_logical_drive( char drive)
{  char cur_drive;       /* Current drive attached to block device */
   union REGS regs;      /* Registers for interupt calls */

   drive = drive ? toupper(drive) : (getdisk()+'A'); /* Default to current drive */

   if( !isalpha( drive ) )          /* Check for valid letter */
   {  cur_drive = 0;
      _doserrno = EINVDRV;
   }
   else
   {  /* Check for DOS version 3.20 or above */
      if( _osmajor > 3 || _osmajor == 3 && _osminor >= 20 )
      {  regs.x.ax = 0x440E;          /* IOCTL request 0x0E ( get logical drive */
            regs.h.bl = 1 + drive - 'A'; /* Drive No (1=A, 2=B, etc) is in BL */
            intdos( &regs, &regs);
            if( regs.x.cflag )          /* Carry flag set error in doserrno */
               cur_drive = 0;
```

```
            else
              /* Return code of 0 in AL means only one drive attached to block device */
              cur_drive = regs.h.al == 0 ? drive : regs.h.al + ('A'-1);
        }
        else
        { if( drive == 'A' || drive == 'B' ) /* For lower versions of DOS, only drives A and B are
interchangeable */
            { .int86( 0x11, &regs, &regs );
              if( regs.x.ax & 0x0001 )       /* Bit 0 in AX is 1 if diskette drives are present */
              { if( ((regs.x.ax & 0xC0) >> 6).== 0 )
                    cur_drive = drive == 'A' || drive == 'B'
                    ? peekb( DOS_DATA_SEG, 0X0004) + 'a' : drive; /* Only one diskette
installed */
                else
                    cur_drive = drive; /* More than one diskette installed */
              }
              else
              { cur_drive = 0;      /* No diskette drive */
                _doserrno = EINVDRV;
            } }
            else
              cur_drive = drive;
  } }
  return( cur_drive );
}
□
```

CONFIG.C

```
#include <stdlib.h>
#include <alloc.h>
#include <dir.h>
#include "tui.h"
#include "log.h"
#define CONFIG_FILE

#define DEFAULT_REC_RATE  0
#define DEFAULT_AVE_NO    0

#define XRATE     61
#define YRATE     14
#define N_RRT_MENU  7

TAB  rateMenu[N_RRT_MENU] =
{  { "500 rec/sec", NULL, XRATE+1, YRATE+1, 11, ENABLE, NULL },
   { " 50 rec/sec", NULL, XRATE+1, YRATE+2, 11, ENABLE, NULL },
   { " 10 rec/sec", NULL, XRATE+1, YRATE+3, 11, ENABLE, NULL },
   { "  1 rec/sec", NULL, XRATE+1, YRATE+4, 11, ENABLE, NULL },
   { "  2 rec/min", NULL, XRATE+1, YRATE+5, 11, ENABLE, NULL },
   { "  1 rec/min", NULL, XRATE+1, YRATE+6, 11, ENABLE, NULL },
   { "manual trig", NULL, XRATE+1, YRATE+7, 11, DISABLE, NULL }
};

#define  XAVE      61
#define  YAVE      16
#define  N_AVE_MENU  5
TAB  aveMenu[N_AVE_MENU] =
{  {"   1 ", NULL, XAVE+1, YAVE+1, 5, ENABLE, NULL },
   { "  10 ", NULL, XAVE+1, YAVE+2, 5, ENABLE, NULL },
   { "  50 ", NULL, XAVE+1, YAVE+3, 5, ENABLE, NULL },
   { " 100 ", NULL, XAVE+1, YAVE+4, 5, ENABLE, NULL },
   { " 250 ", NULL, XAVE+1, YAVE+5, 5, ENABLE, NULL }
};

extern CONFIG   cnfg;
extern ulong    rateSpan[];

Config()
{
#define  XCNFG   3
#define  YCNFG   6
#define  WCHAN  (WTYPE+WMAX+WMIN+WUNIT+11)
  TAB  menu[] =
  {  { NULL,    NULL, XCNFG+11,      YCNFG+1,  60, ENABLE, NULL },
     {"      ", NULL, XCNFG+WCHAN+7,  YCNFG+5,   6, ENABLE, NULL },
     { NULL,    NULL, XRATE+1,       YRATE+1,  11, ENABLE,  NULL },
     { NULL,    NULL, XAVE+1,        YAVE+1,    5, ENABLE,  NULL },

     { NULL,    NULL, XCNFG+1,  YCNFG+5,  WCHAN, ENABLE, NULL },
     { NULL,    NULL, XCNFG+1,  YCNFG+6,  WCHAN, ENABLE, NULL },
     { NULL,    NULL, XCNFG+1,  YCNFG+7,  WCHAN, ENABLE, NULL },
     { NULL,    NULL, XCNFG+1,  YCNFG+8,  WCHAN, ENABLE, NULL },
     { NULL,    NULL, XCNFG+1,  YCNFG+9,  WCHAN, ENABLE, NULL },
     { NULL,    NULL, XCNFG+1,  YCNFG+10, WCHAN, ENABLE, NULL },
```

```
    { NULL,    NULL, XCNFG+1,  YCNFG+11, WCHAN, ENABLE, NULL },
    { NULL,    NULL, XCNFG+1,  YCNFG+12, WCHAN, ENABLE, NULL },

    { "  OK  ", 'O',  XCNFG+19, YCNFG+14,  8, ENABLE, NULL },
    { " CANCEL ", 'C',  XCNFG+29, YCNFG+14,  8, ENABLE, NULL },
    { " Reset  ", 'R',  XCNFG+39, YCNFG+14,  8, ENABLE, NULL },
    { " Load.. ", 'L',  XCNFG+49, YCNFG+14,  8, ENABLE, NULL }
};
char tabChan[8][WCHAN+1];

BOARD  brd = { XCNFG, YCNFG, XCNFG+75, YCNFG+15, DOUBLE,
               NORMAL, BLUE, BROWN, NULL
             };
TABCOLOR oldTC, newTC = { BLUE, MAGENTA, WHITE, YELLOW, RED };

CONFIG tem;
ulong  Time;
char   TimeBuf[13];
ulong  recAvail;
char   recAvailBuf[7];
TIME   timeAvail;
int    choice, i;

memcpy( &tem, &cnfg, sizeof( CONFIG ) );

menu[0].item = tem.Comment;
for( i = 0; i < 8; i++ )
  menu[i+4].item = tabChan[i];

GetTabColor( &oldTC );
SetTabColor( &newTC );

BuiltBoard( &brd );
Hline( brd.ix, brd.iy+3,  brd.ex, SH );
Hline( brd.ix, brd.iy+13, brd.ex, SH );
Putcxy( brd.ix, brd.iy+3,  SBRD );
Putcxy( brd.ix, brd.iy+13, SBRD );
Putcxy( brd.ex, brd.iy+3,  SBLD );
Putcxy( brd.ex, brd.iy+13, SBLD );
Vline( brd.ix+WCHAN+1, brd.iy+3, brd.iy+13, SV );
Putcxy( brd.ix+WCHAN+1, brd.iy+3, SDB );
Putcxy( brd.ix+WCHAN+1, brd.iy+13, SUB );

PutString( (brd.ix+brd.ex-22) / 2, brd.iy, " System Configuration " );
PutString( menu[0].x-9,  menu[0].y,  "Comment: " );
PutString( brd.ix+WCHAN/2-2, brd.iy+3,  "Channels" );
PutString( brd.ix+2, brd.iy+4,
           "No.  type     maximum  minimum  unit" );
PutString( brd.ix+WCHAN+12, brd.iy+3, "Recording" );
PutString( brd.ix+WCHAN+7,  brd.iy+4, "Require    Available" );
PutString( brd.ix+WCHAN+8,  brd.iy+7, "(hr:min:sec:msec)" );
PutString( XRATE-5,        YRATE+1,  "Rate:" );
PutString( XAVE-8,         YAVE+1,   "Average:      time" );

for( choice = 1;;)
{  for( tem.openChan = 0, i = 0; i < 8; i++ )
       if( tem.Chan[i].stat )
          tem.openChan++;
```

```
recAvail  = (tem.openChan)? farcoreleft()/tem.openChan : 0;
tem.recAmount = (tem.recAmount > recAvail)? recAvail : tem.recAmount;
sprintf( menu[1].item, "%6ld", tem.recAmount );
sprintf( recAvailBuf,  "%6ld", recAvail );
PutString( brd.ix+WCHAN+20, brd.iy+5, recAvailBuf );

if( tem.recRateNo <= 2 )
{  for( i = 0; i < 5; i++ )
        aveMenu[i].stat = (i<=tem.recRateNo)? ENABLE:DISABLE;
   tem.aveNo = (tem.aveNo > tem.recRateNo)? tem.recRateNo : tem.aveNo;
}
else
      for( i = 0; i < 5; i++ )
        aveMenu[i].stat = ENABLE;
menu[2].item = rateMenu[tem.recRateNo].item;
menu[3].item = aveMenu[tem.aveNo].item;

Time = tem.recAmount*rateSpan[tem.recRateNo];
tem.recTime.msec = Time%500;
tem.recTime.sec  = (Time/500)%60;
tem.recTime.min  = (Time/30000)%60;
tem.recTime.hr   = Time/1800000L;
sprintf( TimeBuf, "%02d:%02d:%02d:%03d",
 tem.recTime.hr,  tem.recTime.min, tem.recTime.sec, tem.recTime.msec );
PutString( brd.ix+WCHAN+3, brd.iy+6, TimeBuf );

Time = recAvail*rateSpan[tem.recRateNo];
timeAvail.msec = Time%500;
timeAvail.sec  = (Time/500)%60;
timeAvail.min  = (Time/30000)%60;
timeAvail.hr   = Time/1800000L;
sprintf( TimeBuf, "%02d:%02d:%02d:%03d", timeAvail.hr,  timeAvail.min,
                                         timeAvail.sec, timeAvail.msec );
PutString( brd.ix+WCHAN+18, brd.iy+6, TimeBuf );

for( i = 0; i< 8; i++ )
{  sprintf( menu[i+4].item, " %c%d %-*s  %*s  %*s %-*s",
                tem.Chan[i].stat? 4 : ' ', i+1,
                WTYPE, tem.Chan[i].type, WMAX, tem.Chan[i].max,
                WMIN,  tem.Chan[i].min, WUNIT, tem.Chan[i].unit );
}

PutMenu( menu, 16 );
choice = MenuSelect( menu, 16, choice );
switch( choice )
{  case  1: GetNChar( menu[0].x, menu[0].y, menu[0].w, tem.Comment );
                break;
      case  2: if( menu[1].item[0] == ' ' )
                   strcpy( menu[1].item, strrchr( menu[1].item, ' ')+1 );
                if( GetDigitString( menu[1].x, menu[1].y, menu[1].w,
                                    menu[1].item ) )
                  tem.recAmount = (ulong)atol( menu[1].item );
                break;
      case  3: i =  PopupMenu( rateMenu, 7, tem.recRateNo+1 );
                tem.recRateNo = (i > 0)? i-1 : tem.recRateNo;
                break;
      case  4: i =  PopupMenu( aveMenu, 5, tem.aveNo+1 );
                tem.aveNo = (i > 0)? i-1 : tem.aveNo;
```

```
                    break;
            case 5: case 6: case 7: case 8: case 9: case 10: case 11:
            case 12: SetChan( &tem.Chan[choice-5], choice-5 );
                    break;
            case 15: ResetConfig( &tem );
                    break;
            case 16: LoadConfig( &tem );
                    break;
            case 13: memcpy( &cnfg, &tem, sizeof( CONFIG ) ); /* next to default */
            default: EraseBoard( &brd );
                    SetTabColor( &oldTC );
                    return (cnfg.openChan && cnfg.recAmount)? OK : FAIL;
} } }

#define XCHAN  27
#define YCHAN  10
SetChan( CHANNEL *chan, int no )
{ TAB menu[] =
  { { NULL,     NULL, XCHAN+12, YCHAN+1, WTYPE, ENABLE, NULL },
    { NULL,     NULL, XCHAN+12, YCHAN+2, WMAX, ENABLE, NULL },
    { NULL,     NULL, XCHAN+12, YCHAN+3, WMIN, ENABLE, NULL },
    { NULL,     NULL, XCHAN+12, YCHAN+4, WUNIT, ENABLE, NULL },
    { " CLOSE ", 'C', XCHAN+5, YCHAN+6, 7, ENABLE, NULL },
    { " OPEN ", 'O', XCHAN+16, YCHAN+6, 7, DISABLE, NULL }
  };
  BOARD brd = { XCHAN, YCHAN, XCHAN+WTYPE+14, YCHAN+7,
                SINGLE, FLOAT, YELLOW, RED, NULL };
  int  choice;
  char buf[4][WTYPE+1];

  BuiltBoard( &brd );
  Hline( brd.ix, brd.ey-2, brd.ex, SH );
  Putcxy( brd.ix, brd.ey-2, SRB );
  Putcxy( brd.ex, brd.ey-2, SLB );
  sprintf( buf[0], "Channel %d", no+1 );
  PutString( (brd.ix + brd.ex - 9) / 2 , brd.iy, buf[0] );
  PutString( menu[0].x-9, menu[0].y, "  type:" );
  PutString( menu[1].x-9, menu[1].y, "maximum:" );
  PutString( menu[2].x-9, menu[2].y, "minimum:" );
  PutString( menu[3].x-9, menu[3].y, "  unit:" );
  strcpy( buf[0], chan->type );
  strcpy( buf[1], chan->max );
  strcpy( buf[2], chan->min );
  strcpy( buf[3], chan->unit );
  menu[0].item = buf[0];
  menu[1].item = buf[1];
  menu[2].item = buf[2];
  menu[3].item = buf[3];

  for( choice = 1; ; )
  { menu[5].stat = (IsDigString(menu[1].item) && IsDigString(menu[2].item))?
                   ENABLE : DISABLE;
    PutMenu( menu, 6 );
    choice = MenuSelect( menu, 6, choice );
    switch( choice )
    { case 1: GetNChar( menu[0].x, menu[0].y, menu[0].w, menu[0].item );
              break;
      case 2: GetDigitString(menu[1].x,menu[1].y,menu[1].w,menu[1].item);
```

```
                    break;
            case  3: GetDigitString(menu[2].x,menu[2].y,menu[2].w,menu[2].item);
                    break;
            case  4: GetNChar( menu[3].x, menu[3].y, menu[3].w, menu[3].item );
                    break;
            case  6: chan->stat = 1;
                    strcpy( chan->type, menu[0].item );
                    strcpy( chan->max,  menu[1].item );
                    strcpy( chan->min,  menu[2].item );
                    strcpy( chan->unit, menu[3].item );
                    EraseBoard( &brd );
                    return OK;
            case  5: chan->stat = 0;
                    chan->type[0] = NULL;
                    chan->max[0]  = NULL;
                    chan->min[0]  = NULL;
                    chan->unit[0] = NULL;
            default: EraseBoard( &brd );
                    return FAIL;
} } }


ResetConfig( CONFIG *tem )
{  tem->recRateNo = DEFAULT_REC_RATE;
   tem->aveNo   = DEFAULT_AVE_NO;
   ResetAllChan( tem->Chan );
   tem->Comment[0] = NULL;
}

ResetAllChan( CHANNEL *Chan )
{  int    i;
   for( i = 0; i < 8; i++ )
   {  Chan[i].type[0] = NULL;
      Chan[i].max[0]  = NULL;
      Chan[i].min[0]  = NULL;
      Chan[i].unit[0] = NULL;
      Chan[i].stat    = OFF;
} }

LoadConfig( CONFIG *tem )
{  FILE   *fp;
   char   path[MAXPATH];
   uchar  flag;
   CONFIG  cnfgBuf;
   BOARD   brd = { 20, 7, 61, 9, SINGLE, FLOAT, BLUE, GREEN };

   ResetConfig( &cnfgBuf );
   BuiltBoard( &brd );
   if( FileOpenRead( &fp, strcpy( path, "*.CSV" ), brd.ix+1, brd.iy+1 ) == OK )
   {  if( ( LoadComment(   fp, &cnfgBuf ) == OK )  &&
          ( LoadRecAmount( fp, &cnfgBuf ) == OK )  &&
          ( LoadRecRateNo( fp, &cnfgBuf ) == OK )  &&
          ( LoadAveNo(     fp, &cnfgBuf ) == OK )  &&
          ( LoadChannel(   fp, &cnfgBuf ) == OK )
        )
      {  memcpy( tem, &cnfgBuf, sizeof( CONFIG ) );
         flag = OK;
      }
```

```
        else
        { Ask( 1, BLUE, RED, 2,
                    " Can't load configuration from file's heading. ",
                        "The heading area of file may be changed."    );
            flag = FAIL;
        }
        fclose( fp );
      }
      EraseBoard( &brd );
      return flag;
    }

    #define F_CMNT_START   17
    #define F_CMNT_LEN     60
    #define F_RAMNT_START  97
    #define F_RAMNT_LEN    6
    #define F_RRTNO_START  123
    #define F_RRTNO_LEN    11
    #define F_AVENO_START  154
    #define F_AVENO_LEN    5
    #define F_UCHAN_START  219
    #define F_UCHAN_LEN    1
    #define F_CHAN_START   223

    LoadComment( FILE *fp, CONFIG *tem )
    { fseek( fp, F_CMNT_START, SEEK_SET );
      return ( fgets( tem->Comment, F_CMNT_LEN+1, fp ) == NULL )? FAIL : OK;
    }

    LoadRecAmount( FILE *fp, CONFIG *tem )
    { char   buf[7];

      fseek( fp, F_RAMNT_START, SEEK_SET );
      if( fgets( buf, F_RAMNT_LEN+1, fp ) == NULL )
        return FAIL;
      tem->recAmount = atol( buf );
      return OK;
    }

    LoadRecRateNo( FILE *fp, CONFIG *tem )
    { char   buf[F_RRTNO_LEN+1];
      uchar  i;

      fseek( fp, F_RRTNO_START, SEEK_SET );
      if( fgets( buf, F_RRTNO_LEN+1, fp ) == NULL )
        return FAIL;
      for( i = 0; i < N_RRT_MENU; i++ )
      { if( !strcmp( buf, rateMenu[i].item ) )
        { tem->recRateNo = i;
            return OK;
      } }
      return FAIL;
    }

    LoadAveNo( FILE *fp, CONFIG *tem )
    { char   buf[F_AVENO_LEN+1];
      uchar  i;
```

```
    fseek( fp, F_AVENO_START, SEEK_SET );
    if( fgets( buf, F_AVENO_LEN+1, fp ) == NULL )
      return FAIL;
    for( i = 0; i < N_AVE_MENU; i++ )
    { if( !strcmp( buf, aveMenu[i].item ) )
      { tem->aveNo = i;
          return OK;
    } }
    return FAIL;
}

LoadChannel( FILE *fp, CONFIG *tem )
{ char  buf[WTYPE+1];
  uchar i;
  uchar chano;


    fseek( fp, F_UCHAN_START, SEEK_SET );
    if( fgets( buf, 1+1, fp ) == NULL )
      return FAIL;
    tem->openChan = atoi( buf );
    if( tem->openChan == 0 )
      return OK;
    for( i = 0; i < tem->openChan; i++ )
    { fseek( fp, F_CHAN_START + i*(WTYPE+1), SEEK_SET );
      if( fgets( buf, WTYPE+1, fp ) == NULL )
          return FAIL;
      chano = atoi( strrchr( buf, ' ' ) + 1 ) - 1;
      if( chano < 0 || chano > 7 )
          return FAIL;
      tem->Chan[chano].stat = ON;

      fseek( fp, F_CHAN_START + (WTYPE+1)*tem->openChan + 2 + i*(WTYPE+1),
SEEK_SET );
      if( fgets( buf, WTYPE+1, fp ) == NULL )
          return FAIL;
      CutStringFrontSpace( buf );
      strcpy( tem->Chan[chano].type, buf );

      fseek( fp, F_CHAN_START + 2*(WTYPE+1)*tem->openChan + 4 + i*(WTYPE+1),
SEEK_SET );
      if( fgets( buf, WTYPE+1, fp ) == NULL )
          return FAIL;
      CutStringFrontSpace( buf );
      strcpy( tem->Chan[chano].max, buf );

      fseek( fp, F_CHAN_START + 3*(WTYPE+1)*tem->openChan + 6 + i*(WTYPE+1),
SEEK_SET );
      if( fgets( buf, WTYPE+1, fp ) == NULL )
          return FAIL;
      CutStringFrontSpace( buf );
      strcpy( tem->Chan[chano].min, buf );

      fseek( fp, F_CHAN_START + 4*(WTYPE+1)*tem->openChan + 8 + i*(WTYPE+1),
SEEK_SET );
      if( fgets( buf, WTYPE+1, fp ) == NULL )
          return FAIL;
      CutStringFrontSpace( buf );
```

```
            strcpy( tem->Chan[chano].unit, buf );
        }
        return OK;
    }

    CutStringFrontSpace( char *str )
    {  while( str[0] == ' ' && str[0] != NULL )
           strcpy( str, &str[1] );
    }
```

RUN.C

```
#include <stdlib.h>
#include <dos.h>
#include <alloc.h>
#include <graphics.h>
#include <math.h>
#include "log.h"
#include "tui.h"
#include "vcoords.h"

#define SAMPLING_PAGE  0
#define AVERAGED_PAGE  1


extern volatile ulong  RecNo;
extern volatile uchar  errorFlag;
extern volatile uchar  SmpData[8];
extern volatile uchar  AveData[8];


extern unsigned  LPTsubB;
extern unsigned  LPTsubA;

uchar    NChan;
uchar    chno[8];
float    coef[8];
float    offs[8];
int      colE[8];
int      colI[8];

extern unsigned  SumData[8];
extern uchar     AveNo;
extern ulong     Span;
extern ulong     SmpCount;
extern ulong     RecCount;
extern ulong     NRec;

extern uchar huge  *Data;

extern void interrupt  (*OldTickISR)();
extern void interrupt  NewTickISR( void );
extern void interrupt  Sampling( void );

ulong   rateSpan[] = {   1, 10, 50, 500, 15000L,  30000L };
ulong   sumAmount[] = {  1, 10, 50, 100, 250 };
extern  CONFIG cnfg;

Run()
{  int  i, j;
   int  colE1;

   NChan = cnfg.openChan;

#define  COL_WI  100
   colE1  = 400 - NChan*COL_WI/2 + COL_WI;
   for( i = j = 0;  i < 8; i++ )
```

```
    { if( cnfg.Chan[i].stat )
      { chno[j] = i;
            offs[j] = atof( cnfg.Chan[i].min );
            coef[j] = ( atof( cnfg.Chan[i].max ) - offs[j] ) / 256.0;
            colE[j] = colE1 + COL_WI*j;
            colI[j] = colE[j] - COL_WI;
            SumData[j] = 0;
            j++;
    } }
    if( InitGraph() == FAIL )
      return FAIL;

    DrawRunScreen( SAMPLING_PAGE );
    DrawRunScreen( AVERAGED_PAGE );

    LPTsubA = peek( 0x0040, 0x0008 );
    LPTsubB = LPTsubA + 1;    /* input port address: 5 bit */
    AveNo = sumAmount[cnfg.aveNo];
    RecNo = 0L;
    NRec  = cnfg.recAmount;
    Span = rateSpan[cnfg.recRateNo];
    RecCount = Span;
    SmpCount = 0L;

    OldTickISR = getvect( 0x1C );
    SetClock( COUNTPERTICK );
/*  setvect( 0x1C, NewTickISR );
*/  setvect( 0x1C, Sampling );

    Display();

    setvect( 0x1C, OldTickISR );
    SetClock( 0xFFFFL );

    while( !kbhit() );
    bioskey(0);
    closegraph();
    DrawMainScreen();
    CursorOff();
    return OK;
}

InitGraph()
{ int       driver = EGA, mode = EGAHI, grResult;

    initgraph( &driver, &mode, "BGI" );
    grResult =  graphresult();
    if( grResult != grOk )
    { if( (driver != EGA) && (driver != VGA ) )
            Ask( 1, BLUE, GREEN, 2, "GRAPHICS ERROR", grapherrormsg( grResult ) );
      return FAIL;
    }
    setFactors();
    return OK;
}

#define GRPIX   101
#define GRPIY   231
```

```
#define GRPW   600
#define GRPH   256
#define GRPEX  (GRPIX+GRPW-1)
#define GRPEY  (GRPIY+GRPH-1)

#define DT_BOT   85
#define DT_TOP   100
#define DT_WIDE  70
#define DT_SIZE  4

#define REC_BOT  530
#define REC_TOP  550
#define REC_RIG  254
#define REC_LEF  204

DrawRunScreen( int page )
{ int  i;
  char  buf[40];
  extern TAB rateMenu[];
  extern TAB aveMenu[];

  setactivepage( page );
  setfillstyle( SOLID_FILL, BROWN );
  bar( Dx(1), Dy(550), Dx(800), Dy(600) );
  bar( Dx(1), Dy(1),   Dx(800), Dy(25) );

  settextjustify( CENTER_TEXT, CENTER_TEXT );
  settextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );
  outtextxy( Dx(400), Dy(580), "\4 Data Logger \4" );
  settextstyle( SMALL_FONT, HORIZ_DIR, 4 );
  outtextxy( Dx(200), Dy(15), "Q - Stop   G - Sampling/Average graph" );


  setfillstyle( INTERLEAVE_FILL, BLUE );
  bar( Dx(1), Dy(26), Dx(800), Dy(549) );

  setfillstyle( CLOSE_DOT_FILL, MAGENTA );
  bar( Dx(721), Dy(320), Dx(780), Dy(380) );
  outtextxy( Dx(750), Dy(360),
    (page == SAMPLING_PAGE)? "SAMPLING" : "AVERAGE" );
  outtextxy( Dx(750), Dy(340), "GRAPH" );

  setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
  rectangle( Dx( GRPIX-1 ), Dy( GRPIY-2 ), Dx( GRPEX+2 ), Dy( GRPEY+2 ) );

  settextstyle( SMALL_FONT, HORIZ_DIR, 2 );
  for( i = 0; i <= 600; i += 50 )
  { sprintf( buf, "%d", i/10 );
    outtextxy( Dx( GRPIX+i ), Dy( GRPIY-13 ), buf );
  }

  for( i = 0; i <= 256; i += 32 )
  { sprintf( buf, "%4.1f", (float)i/2.56 );
    outtextxy( Dx( GRPIX-15 ), Dy( GRPIY+i ), buf );
  }

  settextstyle( SMALL_FONT, VERT_DIR, 4 );
  outtextxy( Dx(GRPIX-35), Dy(GRPIY+GRPH/2), "% Full Scale" );
```

```
    settextstyle( SMALL_FONT, HORIZ_DIR, 4 );
    outtextxy( Dx(GRPEX+25), Dy(GRPIY-10), "sec" );
    outtextxy( Dx(150), Dy(REC_BOT+7), "Recording  No:" );
    sprintf( buf, "Total: %6ld", cnfg.recAmount );
    outtextxy( Dx(206), Dy(REC_BOT-15), buf );
    sprintf( buf, "%2d hr %2d min %2d sec %3d msec", cnfg.recTime.hr,
            cnfg.recTime.min, cnfg.recTime.sec, cnfg.recTime.msec );
    outtextxy( Dx(390), Dy(REC_BOT-15), buf );
    sprintf( buf, "   Rate: %s", rateMenu[cnfg.recRateNo] );
    outtextxy( Dx(625), Dy(REC_BOT+7), buf );
    sprintf( buf, "Average:%s time", aveMenu[cnfg.aveNo] );
    outtextxy( Dx(615), Dy(REC_BOT-15), buf );

#define ROW_HI 20
    settextjustify( RIGHT_TEXT, BOTTOM_TEXT );
    for( i = 0; i < cnfg.openChan; i++ )
    {  setcolor( 15 - i );
       line( Dx(colE[i]-50), Dy(DT_BOT+4*ROW_HI), Dx(colE[i]), Dy
(DT_BOT+4*ROW_HI) );
       setcolor( WHITE );
       sprintf( buf, "Chan. %d", i+1 );
       outtextxy( Dx(colE[i]), Dy(DT_BOT+3*ROW_HI), buf );
       outtextxy( Dx(colE[i]), Dy(DT_BOT+2*ROW_HI), cnfg.Chan[chno[i]].type );
       outtextxy( Dx(colE[i]), Dy(DT_BOT+ROW_HI), cnfg.Chan[chno[i]].max );
       outtextxy( Dx(colE[i]), Dy(DT_BOT-ROW_HI), cnfg.Chan[chno[i]].min );
       outtextxy( Dx(colE[i]), Dy(DT_BOT-2*ROW_HI), cnfg.Chan[chno[i]].unit );
    }
}


Display()
{  uchar    i, visualP;
   unsigned graphX;
   unsigned min;
   char     buf[20];

   visualP = SAMPLING_PAGE;
   for( ; RecNo < NRec; )
   {  graphX = (unsigned)(SmpCount/50)%600;
      if( graphX == 0 || graphX == 1 )
      {  min = (unsigned)(SmpCount/50)/600;
            sprintf( buf, "Time: %2d hr %2d min", min/60, min%60 );

            setactivepage( SAMPLING_PAGE );
            PutGraphPlate();
            bar( Dx(GRPIX+210), Dy(GRPIY-20), Dx(GRPIX+360), Dy(GRPIY-40) );
            outtextxy( Dx(GRPIX+360), Dy(GRPIY-40), buf );

            setactivepage( AVERAGED_PAGE );
            PutGraphPlate();
            bar( Dx(GRPIX+210), Dy(GRPIY-20), Dx(GRPIX+360), Dy(GRPIY-40) );
            outtextxy( Dx(GRPIX+360), Dy(GRPIY-40), buf );
      }

      graphX += GRPIX;
      for( i = 0; i < NChan; )
      {  setactivepage( SAMPLING_PAGE );
```

```
            putpixel(  Dx( graphX ), Dy( SmpData[ i ] + GRPIY ), 15 - i );
            setactivepage( AVERAGED_PAGE );
            putpixel(  Dx( graphX ), Dy( AveData[ i ] + GRPIY ), 15 - i );
            setactivepage( visualP );
            if( errorFlag )
              sprintf( buf, "-ERROR-" );
            else
              sprintf( buf, "%8.2f", (float)AveData[i]*coef[i] + offs[i] );
            bar( Dx( colI[i] ), Dy( DT_BOT ), Dx( colE[i]), Dy( DT_TOP ) );
            outtextxy( Dx( colE[i++] ), Dy( DT_BOT ), buf );
        }
    /*    setactivepage( AVERAGED_PAGE );
        for( i = 0; i < NChan; i++ )
            putpixel(  Dx( graphX ), Dy( AveData[ i ] + GRPIY ), 15 - i );

        setactivepage( visualP );
        for( i = 0; i < NChan; )
        {  if( errorFlag )
              sprintf( buf, "-ERROR-" );
            else
              sprintf( buf, "%8.2f", (float)AveData[i]*coef[i] + offs[i] );
            bar( Dx( colI[i] ), Dy( DT_BOT ), Dx( colE[i]), Dy( DT_TOP ) );
            outtextxy( Dx( colE[i++] ), Dy( DT_BOT ), buf );
        }
    */
        sprintf( buf, "%6ld", RecNo );
        bar( Dx(REC_LEF), Dy(REC_TOP), Dx(REC_RIG), Dy(REC_BOT) );
        outtextxy( Dx(REC_RIG), Dy(REC_BOT), buf );

        if( kbhit() )
        {  switch( toupper( getch() ) )
            {  case 'Q': return;
               case 'G': visualP = (visualP == 0)? 1: 0;
                         setvisualpage( visualP );
                         break;
      } } }
    }

    PutGraphPlate()
    { int  i;
      setfillstyle( SOLID_FILL, BLACK );
      bar( Dx(GRPIX), Dy(GRPIY), Dx(GRPEX), Dy(GRPEY) );
      setlinestyle( DOTTED_LINE, 0, NORM_WIDTH );
      for( i = 32; i <= 224; i += 32 )
        line( Dx( GRPIX ), Dy( GRPIY+i ), Dx( GRPEX ), Dy( GRPIY+i ) );
      for( i = 50; i <= 550; i += 50 )
        line( Dx( GRPIX+i ), Dy( GRPIY ), Dx( GRPIX+i ), Dy( GRPEY ) );
      setfillstyle( INTERLEAVE_FILL, BLUE );
    }
```

SAMPLING.C

```
#include <stdlib.h>
#include <dos.h>
#include <math.h>
#include "log.h"

volatile ulong   RecNo;
volatile uchar   errorFlag;
volatile uchar   SmpData[8];
volatile uchar   AveData[8];

unsigned  LPTsubB;
unsigned  LPTsubA;
extern uchar     NChan;
extern uchar     chno[8];
unsigned  SumData[8];
uchar     AveNo;
ulong     Span;
ulong     SmpCount;
ulong     RecCount;
ulong     NRec;

extern uchar huge   *Data;

void interrupt   (*OldTickISR)();
void interrupt   NewTickISR( void );
void interrupt   Sampling( void );
void SetClock( long clockcount );

void interrupt NewTickISR( void )
{  uchar   i;

  SmpCount++;

  for( i = 0 ; i<NChan; i++ )
  { if(rand()%2)
    if(rand()%2)
          SmpData[i] = (rand()%2)? ((SmpData[i]<255)? SmpData[i]+1:SmpData[i]):
                                   ((SmpData[i]>0)?  SmpData[i]-1:SmpData[i]);
    SumData[i] += SmpData[i];
  }
  if( !(SmpCount % AveNo) )
  {  for( i = 0; i < NChan; )
    {  AveData[i] = SumData[i] / AveNo;
          SumData[i++] = 0;
    }

    if( (SmpCount == RecCount) && (RecNo < NRec) )
    {  for( i = 0; i < NChan; i++ )
          Data[ RecNo*(ulong)NChan + (ulong)i ] =  AveData[i];
       RecCount += Span;
       RecNo++;
} } }

/*----------------------macro for A/D converter----------------------*/
#define  START     0x80     /* START and ALE */
```

```
#define  TAKEHIGH   0x40    /* control MULTIPLEXER out 4 high bit */
#define  HIGH       0xF0    /* mark 4 high bit */
#define  LOW        0x0F    /* mark 4 low bit */
#define  EOCbit     0x08    /* END OF CONVERSION  bit */

void interrupt Sampling( void )
{  uchar   i;
   uchar   againFlag;      /* flag for checks repeation when error occured */
   uchar   convFlag;       /* flag for checks conversion activity */

   SmpCount++;

   /*-----------Sampling Data----------------*/
   for( againFlag = errorFlag = i = 0 ; i<NChan; i++ )
   {
AGAIN:outportb( LPTsubA, chno[i] );      /* Important, must lead */
     outportb( LPTsubA, START|chno[i] ); /* clear old chanel address */
     outportb( LPTsubA, 0x00 );              /* conv. start on tailing */
     for( convFlag = 0;;)                 /* wait for already conv.. */
     {  if( EOCbit & inportb( LPTsubB ) )     /* if EOC go HIGH */
           {  if( convFlag )
                break;
              else  /* no conversion */
              {  if( againFlag )
                 {  errorFlag = 1;
                       break;
                 }
                 else
                 {  againFlag = 1;
                       goto AGAIN;
              } } }
             else
               convFlag = 1;
     }
     SmpData[i] = (inportb(LPTsubB)>>4);    /* take 4 low bit but get by 4 high bit from
LPTsubB */
     outportb( LPTsubA, TAKEHIGH );         /* control multiplexer for taking 4 high bit */
     SmpData[i] = ((HIGH&inportb(LPTsubB)) | SmpData[i] );
     SumData[i] += SmpData[i];
   }
   if( !(SmpCount % AveNo) )
   {  for( i = 0; i < NChan; )
     {      AveData[i] = SumData[i] / AveNo;
            SumData[i++] = 0;
     }
     if( (SmpCount == RecCount) && (RecNo < NRec) )
     {  for( i=0; i<NChan; i++ )
              Data[ RecNo*(ulong)NChan + (ulong)i ] = AveData[i];
           RecCount += Span;
           RecNo ++;
} } }

void SetClock( long clockcount )
{  outportb( CTRLCOUNT, 0x34 );
   outportb( COUNTREG,  clockcount&0x00FF );
   outportb( COUNTREG,  (clockcount&0xFF00)>>8 );
}
```

```
#ifndef  TUI_declare
#define  TUI_declare

#include <conio.h>
#include <string.h>
#include <ctype.h>
typedef  unsigned char  uchar;

#define  BIOSData_Seg  0x0040
#define  VidBufOfs     *((unsigned far *)MK_FP( BIOSData_Seg, 0x0076 ))
#define  VidBufSeg     0xB800
#define  VIDRAM        0xB8000000L

void  InitVid();

void  Cursor  ( char start, char end);

#define  CursorOn()   Cursor( 7, 6)
#define  CursorOff()  Cursor( 0, 1)

void  ClearText( int x, int y, int endx, int endy );
void  ClearAll( int x, int y, int endx, int endy );
void  SetBackColor( int x, int y, int endx, int endy, int back );
void  SetForeColor( int x, int y, int endx, int endy, int fore );
void  GetAttribute( int x, int y, int endx, int endy, char  *buf );
void  PutAttribute( int x, int y, int endx, int endy, char  *buf );

void  PutString( int x, int y, char *str );
void  PutNChar( int x, int y, int n, char *str );
void  CenterString( int y, char *str );

typedef  struct { int  ix;
                  int  iy;
                  int  ex;
                  int  ey;
                  int  edge;
                  int  elev;    /* FLOAT or NORMAL */
                  int  fore;
                  int  back;
                  char *scrn;
                } BOARD;
#define FLOAT  1
#define NORMAL  0

void  Hline( int x, int y, int endx, char line );
void  Vline( int x, int y, int endy, char line );
void  Putcxy( int x, int y, char c);
void  Frame( int lef, int top, int rig, int bot, int edge);
void  BuiltBoard( BOARD *b );
void  EraseBoard( BOARD *b );
/*  double broader marcro  */
#define DH   205   /* horizontal */
#define DV   186   /* vertical  */
#define DTL  201   /* top left corner */
#define DTR  187   /* top right corner */
```

```
#define DBL  200   /* bottom left corner */
#define DBR  188   /* bottom right corner */
#define DC   206   /* cross */
#define DDB  203   /* down branch */
#define DUB  202   /* up branch */
#define DRB  204   /* right branch */
#define DLB  185   /* left branch */

/*  single broader marcro */
#define SH   196   /* horizontal */
#define SV   179   /* vertical */
#define STL  218   /* top left corner */
#define STR  191   /* top right corner */
#define SBL  192   /* bottom left corner */
#define SBR  217   /* bottom right corner */
#define SC   197   /* cross */
#define SDB  194   /* down branch */
#define SUB  193   /* up branch */
#define SRB  195   /* right branch */
#define SLB  180   /* left branch */

#define SBDD 209   /* single branch down from double */
#define SBDU 207   /* single branch up from double */
#define SBRD 199   /* single branch right from double */
#define SBLD 182   /* single branch left from double */

#define NOEDGE   0
#define SINGLE   1
#define DOUBLE   2


typedef  struct{ char *item;    /* tab's item */
                 char  k;       /* short cut key , NULL = not used */
                 int   x;       /* x ordinate */
                 int   y;       /* y ordinate */
                 int   w;       /* tab's wide */
                 int   stat;    /* stutus : ENABLE , DISABLE */
                 char *help;    /* information */
               } TAB;

#define ENABLE  1
#define DISABLE 0

typedef struct
{  unsigned char  hilight;
   unsigned char  back;
   unsigned char  mark;
   unsigned char  enable;
   unsigned char  disable;
} TABCOLOR;

typedef struct
{  unsigned char  x;
   unsigned char  y;
} TABHELP;

void  SetTabMark( int color );
void  SetTabHilight( int color );
```

```
void  SetTabBack( int color );
void  SetTabEnable( int color );
void  SetTabDisable( int color );
void  GetTabColor( TABCOLOR *tabcolor );
void  SetTabColor( TABCOLOR *tabcolor );

void  PutItem( TAB tab );
void  PutMenu( TAB tab[], int n );

int   MenuSelect( TAB  Tab[], int TabAmount, int init  );
int   PopupMenu( TAB  tab[], int n, int init );

void  SetScrollPage( int x, int y, int w, int h );
int   ScrollMenu( char *menu[], int n, int init );

int   BuiltMNBR( int y, TAB *head, int n,... );
int   MenuBar( void );
int   headMNBR( void );
int   childBar( void );

int   Ask( int nChoice, int fore, int back, int n,... );
int   Hollow();

int   GetNChar( int x, int y, int n, char *str );

int   IsDigString( char *str );
int   GetDigitString( int x, int y, int ndigit, char *str );
void  Beep();

#define  Enter        '\r'
#define  Space        ' '
#define  Esc          0x1B
#define  Tab_         '\t'
#define  Bckspc       0x8
#define  Delete       0x5300
#define  Home         0x4700
#define  End          0x4F00
#define    UP_AIRROW   0x4800
#define   DOWN_AIRROW  0x5000
#define   LEFT_AIRROW  0x4B00
#define  RIGHT_AIRROW  0x4D00
#define  Page_Up      0x4900
#define  Page_Down    0x5100
#define  Ctrl_F10     26368


#endif
```

□

```
#include <stdio.h>
#include <dos.h>
#include <ctype.h>
#include "tui.h"

typedef union { int  i;  char c[2]; } UNION;

#define  BIOSData_SEG    0x0040
#define  VIDMODE_SEG    BIOSData_SEG
#define  VIDMODE_OFS    0x0049
#define  MONOVID_ADDR    (UNION far *)0xB0000000L
#define COLORVID_ADDR    (UNION far *)0xB8000000L

UNION  far  *Vid = MONOVID_ADDR;    /* default video ram address */
uchar       VidMode;

TABCOLOR  TabColor = { LIGHTGRAY, BLACK, MAGENTA, WHITE, BLUE };
TABHELP   TabHelp  = { 1, 25 };

struct  { uchar  nH;          /* amount of Head */
          uchar  idH;         /* Head index */
          TAB   *head;        /* Head object */
          uchar *wC;          /* array of Child wide */
          uchar *idC;         /* array of Child index */
          uchar *nC;          /* array of Child amount */
          TAB  **child;       /* array of Child menu pointer */
        } MB;  /* Menu Bar Object */

struct  {  uchar  x;
           uchar  y;
           uchar  w;
           uchar  h;
         }  Scroll;



enum  MessagE { ESC=0, ENTER, CLOSE, PULL, LEFT, RIGHT };

void  InitVid()
{  uchar far *vmode = (MK_FP( VIDMODE_SEG, VIDMODE_OFS ));
   VidMode = *vmode;
   Vid = ((VidMode == 2)||(VidMode == 7))? MONOVID_ADDR : COLORVID_ADDR;
}

void  Cursor(char start,char end)
{  union  REGS  r;
   r.h.ah = 1;              /* function assigned cursor' size */
   r.h.cl = start;
   r.h.ch = end;
   int86( 0x10, &r, &r );
}

void Hline( int x, int y, int endx, char line )
{  register uchar i;
```

```
    i = x-1;
    y--;
    y *= 80;
    for( ; i < endx; i++)
      Vid[i+y].c[0] = line;
  }

  /* draw vertical line  from (x,starty) to (x,endy) with character c */
  void Vline( int x, int y, int endy, char line )
  {  register uchar i;
    x--;
    for( i = y-1; i < endy; i++)
      Vid[(i*80)+x].c[0] = line;
  }

  /* put corner character ,c, at (x,y) */
  void Putcxy( int x, int y, char c)
  {  x--; y--;
    Vid[ x + y*80 ].c[0] = c;
  }

  void Frame( int lef, int top, int rig, int bot, int edge)
  {  uchar     f[6];
  #ifdef  DEBUG
    if( (lef>rig)||(top>bot)||(lef<1)||(rig>80)||(top<1)||(bot>25) )
      Abnormal(" Frame() : bad co-ordinate ! ");
  #endif
    switch( edge)
    {  case  0: f[0]=0; f[1]=0; f[2]=0; f[3]=0; f[4]=0; f[5]=0; break;
      case  2: f[0]=DH; f[1]=DV; f[2]=DTL; f[3]=DTR; f[4]=DBR; f[5]=DBL; break;
      default: f[0]=SH; f[1]=SV; f[2]=STL; f[3]=STR; f[4]=SBR; f[5]=SBL; break;
    }
    Hline( lef+1, top, rig-1, f[0] );
    Hline( lef+1, bot, rig-1, f[0] );
    Vline( lef, top+1, bot-1, f[1] );
    Vline( rig, top+1, bot-1, f[1] );
    Putcxy( lef, top, f[2] );
    Putcxy( rig, top, f[3] );
    Putcxy( rig, bot, f[4] );
    Putcxy( lef, bot, f[5] );
  }

  void  BuiltBoard( BOARD *b )
  {  b->scrn = (char *)malloc((b->ex-b->ix+b->elev*2+1)*(b->ey-b->iy+b->elev+1)*2 );
    gettext( b->ix, b->iy, b->ex + b->elev*2, b->ey + b->elev, b->scrn );
    ClearAll( b->ix, b->iy, b->ex, b->ey );
    Frame( b->ix, b->iy, b->ex, b->ey, b->edge );
    SetBackColor( b->ix, b->iy, b->ex, b->ey, b->back );
    SetForeColor( b->ix, b->iy, b->ex, b->ey, b->fore );
    if( b->elev )
    {  SetBackColor( b->ix+2, b->ey+1, b->ex+2, b->ey+1, BLACK );
      SetBackColor( b->ex+1, b->iy+1, b->ex+2, b->ey+1, BLACK );
  } }

  void  EraseBoard( BOARD *b )
  {  if( b->scrn == NULL )
      return;
    puttext( b->ix, b->iy, b->ex + b->elev*2, b->ey + b->elev, b->scrn );
```

```
  free( b->scrn );
  b->scrn = NULL;
}

void SetBackColor( int x, int y, int endx, int endy, int back )
{  register unsigned char i, j;

  back = ((VidMode==2)||(VidMode==7))? BLACK : back;
#ifdef DEBUG
  if( back > LIGHTGRAY )
    Abnormal( " BackColor() : bad background color ! ");
#endif
  for( i=x-1; i<endx; i++ )
    for( j=y-1; j<endy; j++ )
          Vid[ i+j*80 ].c[1] = (back<<4)|(Vid[ i+j*80].c[1]&0xf);
}

void SetForeColor( int x, int y, int endx, int endy, int fore )
{  register unsigned char i, j;

  fore = ((VidMode==2)||(VidMode==7))? WHITE : fore;
  for( i=x-1; i<endx; i++ )
    for( j=y-1; j<endy; j++ )
          Vid[ i+j*80 ].c[1] = (Vid[ i+j*80 ].c[1]&0xf0) | fore;
}

void PutString( int x, int y, char *str )
{  register unsigned char  i;
  x--; y--;
  for( i = 0; str[i]; i++)
    Vid[ x + y*80  + i ].c[0] = str[ i ];
}

void PutNChar( int x, int y, int n, char *str )
{  register unsigned char  i, len;
  x--; y--;
  len = strlen( str );
  for( i=0; i<n ; i++)
    Vid[ x + y*80 + i ].c[0] = (i < len)? str[ i ] : ' ';
}

void CenterString( int y, char *str )
{  uchar  len;
  len = strlen( str );
#ifdef  DEBUG
  if( len > 80 )
    Abnormal( " CenterString() : string not fit " );
#endif
  PutString( 41 - len/2, y, str );
}

void ClearText( int x, int y, int endx, int endy )
{  register unsigned char i, j;
  for( i=x-1; i<endx; i++ )
    for( j=y-1; j<endy; j++ )
          Vid[ i+j*80 ].c[0] = ' ';
}
```

```
void ClearAll( int x, int y, int endx, int endy )
{  window( x, y, endx, endy );
   clrscr();
   window( 1, 1, 80, 25 );
}

void  GetAttribute( int x, int y, int endx, int endy, char  *buf )
{  register unsigned char  i, j, k;
   for( i=x-1, k=0; i<endx; i++ )
      for( j=y-1; j<endy; j++, k++ )
            buf[ k ] = Vid[ i + j*80 ].c[1];
}

void  PutAttribute( int x, int y, int endx, int endy, char  *buf )
{  register unsigned char  i, j, k;
   for( i=x-1, k=0; i<endx; i++ )
      for( j=y-1; j<endy; j++, k++ )
            Vid[ i + j*80  ].c[1] = buf[ k ];
}

SetTabHelp( int x, int y )
{  TabHelp.x = x;
   TabHelp.y = y;
}

void  SetTabHilight( int color )
{  color = ((VidMode==2)||(VidMode==7))? LIGHTGRAY : color;
   TabColor.hilight = (uchar)color;
#ifdef  DEBUG
   if( color > LIGHTGRAY )
      Abnormal( " SetTabColor() : Improperly Tab color" );
#endif
}

void  SetTabBack( int color )
{  TabColor.back = color;
}

void  SetTabMark( int color )
{  TabColor.mark = (uchar)color;
}

void  SetTabEnable( int color )
{  TabColor.enable = (uchar)color;
}

void SetTabDisable( int color )
{  TabColor.disable = (uchar)color;
}

void GetTabColor( TABCOLOR *tabcolor )
{  tabcolor->hilight = TabColor.hilight;
   tabcolor->back    = TabColor.back;
   tabcolor->mark    = TabColor.mark;
   tabcolor->enable  = TabColor.enable;
   tabcolor->disable = TabColor.disable;
}
```

```
void SetTabColor( TABCOLOR *tabcolor )
{  TabColor.hilight = tabcolor->hilight;
   TabColor.back    = tabcolor->back;
   TabColor.mark    = tabcolor->mark;
   TabColor.enable  = tabcolor->enable;
   TabColor.disable = tabcolor->disable;
}

void  PutTab( TAB tab, char **Buf )
{  *Buf = (char *)malloc( sizeof( char )*tab.w );
#ifdef  DEBUG
   if( *Buf == NULL )
      Abnormal( " PutTab() : can't allowcate memmory " );
#endif
   GetAttribute( tab.x, tab.y, tab.x+tab.w-1, tab.y, *Buf );
   SetBackColor( tab.x, tab.y, tab.x+tab.w-1, tab.y, TabColor.hilight );
}

void  DelTab( TAB tab, char **Buf )
{
#ifdef  DEBUG
   if( *Buf == NULL )
      Abnormal( " DelTab() : *Buf == NULL " );
#endif
   PutAttribute( tab.x, tab.y, tab.x+tab.w-1, tab.y, *Buf );
   free( *Buf );
}

void  PutItem( TAB tab )
{  uchar  i;
   PutNChar( tab.x, tab.y, tab.w, tab.item );
   SetBackColor( tab.x, tab.y, tab.x+tab.w-1, tab.y, TabColor.back );
   if( tab.stat == ENABLE )
   {  SetForeColor( tab.x, tab.y, tab.x+tab.w-1, tab.y, TabColor.enable );
      for( i=0; i<tab.w && tab.item[i]; i++ )
      {  if( tab.item[i] == tab.k )
            {  SetForeColor( tab.x+i, tab.y, tab.x+i, tab.y, TabColor.mark );
               break;
   } } }
   else
      SetForeColor( tab.x, tab.y, tab.x+tab.w-1, tab.y, TabColor.disable );
}

void  PutMenu( TAB  tab[], int n )
{  uchar i;
   for( i=0; i<n; i++ )
      PutItem( tab[i] );
}

int Hollow()
{  return 1;
}

int   MenuSelect( TAB  tab[], int TabAmount, int init  )
{  union{  int  i;
           char c[2];
        } u;
```

```
  uchar p;
  uchar i;
  char *tabBuf;
  char keyid;

#ifdef DEBUG
  if( (init == 0) || (init > TabAmount) )
    Abnormal( " TabSelect() : invalid tab's initial positon (init) " );
#endif
  for( p = abs(init)-1; ; )
  { PutTab( tab[p], &tabBuf );
    PutString( TabHelp.x, TabHelp.y, tab[p].help );
    u.i = bioskey(0);
    ClearText( TabHelp.x, TabHelp.y, TabHelp.x+strlen(tab[p].help), TabHelp.y );
    DelTab( tab[p], &tabBuf );
    if( u.c[0] )
    { switch( u.c[0] )
        { case Enter: if( tab[p].stat )
                        return p+1;
                      else
                      { Beep();
                        break;
                      }
          case Space: p++;  break;
          case Tab_ : p++;  break;
          case Esc  : return -(p+1);
          default   : keyid = FindKey( tab, TabAmount, u.c[0] );
                      if( (keyid != -1) && tab[keyid].stat )
                        return keyid + 1;
        }
      p = ( p>=TabAmount )? 0 : p;
      p = ( p<0 )? TabAmount-1: p;
    }
    else
    { uchar j;
      switch( u.i )
      { case UP_AIRROW: /* find most top */
                        for( i=0, j=p; i<TabAmount; i++ )
                        { if( tab[i].y < tab[j].y )
                            j = i;
                        }
                        if( tab[j].y < tab[p].y )
                        /* find next up from current */
                        { for( i=0; i<TabAmount; i++ )
                          { if( (tab[i].y < tab[p].y) &&
                                (tab[i].y > tab[j].y) )
                              j = i;
                        } }
                        else
                        /* find most bottom */
                        { for( i=0; i<TabAmount; i++ )
                          { if( tab[i].y > tab[j].y )
                              j = i;
                        } }
                        /* find horizontal nearest */
                        for( i=0; i<TabAmount; i++ )
                        { if( (tab[i].y == tab[j].y) &&
                              (abs(tab[i].x-tab[p].x)) <
```

```
                    (abs(tab[j].x-tab[p].x)) )
                     j = i;
                }
                break;
        case DOWN_AIRROW: /* find most bottom */
                for( i=0, j=p; i<TabAmount; i++ )
                {  if( tab[i].y > tab[j].y )
                     j = i;
                }
                if( tab[j].y > tab[p].y )
                /* find next down from current */
                {  for( i=0; i<TabAmount; i++ )
                   {  if( (tab[i].y > tab[p].y) &&
                          (tab[i].y < tab[j].y) )
                        j = i;
                } }
                else
                /* find most top */
                {  for( i=0; i<TabAmount; i++ )
                   {  if( tab[i].y < tab[j].y )
                        j = i;
                } }
                /* find horizontal nearest */
                for( i=0; i<TabAmount; i++ )
                {  if( (tab[i].y == tab[j].y) &&
                       (abs(tab[i].x-tab[p].x)) <
                       (abs(tab[j].x-tab[p].x)) )
                     j = i;
                }
                break;
        case LEFT_AIRROW: /* find most left  */
                for( i=0, j=p; i<TabAmount; i++ )
                {  if( (tab[i].y == tab[p].y) &&
                       (tab[i].x <  tab[j].x) )
                    j = i;
                }
                if( tab[j].x < tab[p].x )
                {  /* find left next from current */
                                for( i=0; i<TabAmount; i++ )
                   {  if( (tab[i].y == tab[p].y) &&
                          (tab[i].x <  tab[p].x) &&
                          (tab[i].x >  tab[j].x) )
                        j = i;
                } }
                else
                {  /* find most right */
                   for( i=0; i<TabAmount; i++ )
                   {  if( (tab[i].y == tab[p].y) &&
                          (tab[i].x >  tab[j].x) )
                        j = i;
                } }
                break;
        case RIGHT_AIRROW: /* find most right  */
                for( i=0, j=p; i<TabAmount; i++ )
                {  if( (tab[i].y == tab[p].y) &&
                       (tab[i].x >  tab[j].x) )
                     j = i;
                }
```

```
                    if( tab[j].x > tab[p].x )
                    {  /* find right next from current */
                      for( i=0; i<TabAmount; i++ )
                      {  if( (tab[i].y == tab[p].y) &&
                            (tab[i].x >  tab[p].x) &&
                            (tab[i].x <  tab[j].x) )
                          j = i;
                    } }
                    else
                    {  /* find most left */
                      for( i=0; i<TabAmount; i++ )
                      {  if( (tab[i].y == tab[p].y) &&
                            (tab[i].x <  tab[j].x) )
                          j = i;
                    } }
                    break;
        }
        p = j;
} } }

int  PopupMenu( TAB  tab[], int n, int init )
{  BOARD   brd = { 80, 25, 1, 1, SINGLE, NORMAL, WHITE, BLACK, NULL };
   char   i;

   brd.fore = TabColor.enable;
   brd.back = TabColor.back;
   for( i = 0; i < n; i++ )
   {  brd.ix = (tab[i].x < brd.ix)? tab[i].x : brd.ix;
      brd.iy = (tab[i].y < brd.iy)? tab[i].y : brd.iy;
      brd.ex = (tab[i].x+tab[i].w-1 > brd.ex)? tab[i].x+tab[i].w-1 : brd.ex;
      brd.ey = (tab[i].y > brd.ey)? tab[i].y : brd.ey;
   }
   brd.ix--, brd.iy--, brd.ex++, brd.ey++;
   BuiltBoard( &brd );
   PutMenu( tab, n );
   i = MenuSelect( tab, n, init );
   EraseBoard( &brd );
   return i;
}

void SetScrollPage( int x, int y, int w, int h )
{  Scroll.x = x;
   Scroll.y = y;
   Scroll.w = w;
   Scroll.h = h;
}

int ScrollMenu( char  *menu[], int n, int i )
{  union  {  int   i;
             char  c[2];
          }  u;
   char    *tabBuf;
   TAB      tab;
   int      lstart;

   tab.x = Scroll.x, tab.w = Scroll.w;

   i = abs( i ) - 1;
```

```
lstart = (i / Scroll.h) * Scroll.h;
PutPage( menu, n, lstart );
for(;;)
{  tab.y = Scroll.y + i - lstart;
   PutTab( tab, &tabBuf );
   u.i = bioskey(0);
   DelTab( tab, &tabBuf );
   if( u.c[0] )
   {  switch( u.c[0] )
         {  case  Enter : return i + 1;
            case  Esc   : return 0 - i - 1;
   } }
   else
   {  switch( u.i )
         {  case    Page_Up : if( i )
                              {  i = ((i+1) > Scroll.h)? i - Scroll.h : 0;
                                 if( lstart > i )
                                 {  lstart = ((lstart+1) > Scroll.h)?
                                                lstart - Scroll.h : 0;
                                    PutPage( menu, n, lstart );
                              } }
                              break;
            case  Page_Down : if( (i+1) < n )
                              {  i = ((i+Scroll.h) < (n-1))? i+Scroll.h : n-1;
                                 if( (lstart+Scroll.h-1) < i )
                                 {  lstart = ((lstart+Scroll.h) < (n-1))?
                                                lstart+Scroll.h : n-1;
                                    PutPage( menu, n, lstart );
                              } }
                              else
                              {  lstart = n - 1;
                                 PutPage( menu, n, lstart );
                              }
                              break;
            case  UP_AIRROW : if( i )
                              {  if( i == lstart )
                                 {  lstart--;
                                    PutPage( menu, n, lstart );
                                 }
                                 i--;
                              }
                              break;
            case DOWN_AIRROW : if( (i+1) < n )
                              {  if( i == (lstart+Scroll.h-1) )
                                    {  lstart++;
                                       PutPage( menu, n, lstart );
                                    }
                                    i++;
                              }
                     break;
            case      Home : if( i )
                              {  i =  lstart = 0;
                                    PutPage( menu, n, lstart );
                              }
                              break;
            case      End  : if( (i+1) < n )
                              {  i = lstart = n - 1;
                                 PutPage( menu, n, lstart );
```

```
                                    }
                                    break;
} } } }

PutPage( char *menu[], int n, int start )
{ int i;
  ClearText( Scroll.x, Scroll.y, Scroll.x+Scroll.w-1, Scroll.y+Scroll.h-1 );
  for( i=0; i < Scroll.h; i++ )
    if( (start+i) < n )
        PutString( Scroll.x, Scroll.y+i, menu[start+i] );
}
/*
main()
{ char **menu;
  int  i, j;
  InitVid();
  SetScrollPage( 4, 10, 20, 10 );
  for( j = 1; j > 0; )
  { menu = (char **)malloc( sizeof( char * ) * 50 );
    for( i = 0; i < 50; i++ )
    {   menu[i] = (char *)malloc( 30 );
        sprintf( menu[i], " menu %2d ", i+1 );
    }
    j = ScrollMenu( menu, 50, j );
    for( i = 0; i < 50; i++ )
        free( menu[i] );
    free( menu );
  }
}
*/

int BuiltMNBR( int y, TAB *head, int n,... )
{ uchar    i, j;
  uchar    wide, childLen;
  va_list  argp;

  MB.nH   = n;
  MB.idH  = 0;
  MB.head = (TAB *)malloc( sizeof( TAB * ) );
  if( !MB.head )
        return 0;
  MB.head = head;
  for( i = 0; i < n; i++ )
  { MB.head[i].y   = y;
    MB.head[i].stat = ENABLE;
  }
  MB.nC  = (uchar *)malloc( sizeof( uchar )*n );
  MB.idC = (uchar *)malloc( sizeof( uchar )*n );
  MB.wC  = (uchar *)malloc( sizeof( uchar )*n );
  MB.child = (TAB **)malloc( sizeof( TAB ** ) );
  if( !MB.nC || !MB.idC || !MB.wC || !MB.child )
    return 0;
  va_start( argp, n );
  for( i = 0; i < MB.nH; i++ )
  { MB.child[i] = (TAB *)malloc( sizeof( TAB * ) );
    if( !MB.child[i] )
        return 0;
    MB.child[i] = va_arg( argp, TAB * );
```

```
        MB.nC[i]   = va_arg( argp, int );
        for( wide = 0, j = 0; j < MB.nC[i]; j++ )
        {  MB.child[i][j].x = MB.head[i].x + 1;
              MB.child[i][j].y = y + 2 + j;
              childLen = strlen( MB.child[i][j].item );
              wide = (childLen > wide)? childLen : wide;
        }
        for( j = 0; j < MB.nC[i]; j++ )
              MB.child[i][j].w = wide;
        MB.wC[i] = wide + 2;
        MB.idC[i] = 0;
        if( (MB.head[i].x + wide + 3) > 80 )
              return 0;
     }
     va_end( argp );
     PutMenu( MB.head, MB.nH );
     return 1;
}

int  MenuBarResult()
{  uchar  i, j;
   for( j = i = 0; i < MB.nH; i++ )
   {  if( i != MB.idH )
              j += MB.nC[i];
      else
      {  j += MB.idC[i];
              return j+1;
} } }

int  MenuBar( void )
{  uchar        prev;
   uchar    Direction;
   char      *tabBuf;

   PutTab( MB.head[MB.idH], &tabBuf );
   for( Direction = CLOSE;;)
   {  prev = MB.idH;
      switch( Direction )
      {  case  ESC:
                 DelTab( MB.head[MB.idH], &tabBuf );
                 return  -MenuBarResult();
            case  ENTER:
                 DelTab( MB.head[MB.idH], &tabBuf );
                 return  MenuBarResult();
            case  LEFT:
                 MB.idH = (MB.idH == 0)? MB.nH-1 : MB.idH-1;
                 Direction = PULL;
                 break;
            case  RIGHT:
                 MB.idH = (MB.idH == MB.nH-1)? 0 : MB.idH+1;
                 Direction = PULL;
                 break;
            case  PULL:
                 Direction = SelectChild();
                 break;
            case  CLOSE:
                 Direction = SelectHead();
                 break;
```

```
        }
        DelTab( MB.head[prev],  &tabBuf );
        PutTab( MB.head[MB.idH], &tabBuf );
    } }

    int  SelectHead()
    { union { int i;
              char c[2];
            } u;
      char  k;

      u.i = bioskey(0);
      if( u.c[0] )
      { switch( u.c[0] )
        { case Enter:
                return PULL;
          case Esc:
                return ESC;
          default: k = FindKey( MB.head, MB.nH, u.c[0] );
                   if( ( k != -1 ) && ( MB.head[k].stat == ENABLE ) )
                   { MB.idH = k;
                     return PULL;
                   }
                   return CLOSE;
      } }
      else
      { switch( u.i )
        { case DOWN_AIRROW:
                return PULL;
          case LEFT_AIRROW:
                MB.idH = (MB.idH == 0)? MB.nH-1 : MB.idH-1;
                return CLOSE;
          case RIGHT_AIRROW:
                MB.idH = (MB.idH == MB.nH-1)? 0 : MB.idH+1;
                return CLOSE;
          default:
                return CLOSE;
    } } }

    int SelectChild()
    { union { int  i;
              char c[0];
            } u;
      char  k;
      char *tabBuf;
      BOARD brd = { 0, 0, 0, 0, SINGLE, FLOAT, 0, 0, NULL };

      brd.ix   = MB.head[MB.idH].x;
      brd.iy   = MB.head[MB.idH].y + 1;
      brd.ex   = brd.ix + MB.wC[MB.idH] - 1;
      brd.ey   = brd.iy + MB.nC[MB.idH] + 1;
      brd.fore = TabColor.enable;
      brd.back = TabColor.back;
      BuiltBoard( &brd );

      SetBackColor( MB.head[MB.idH].x+MB.wC[MB.idH],  MB.head[MB.idH].y+1,
                    MB.head[MB.idH].x+MB.wC[MB.idH]+1, MB.head[MB.idH].y+1,
    BLACK );
```

```
    PutMenu( MB.child[MB.idH], MB.nC[MB.idH] );
    for(;;)
    { PutTab( MB.child[MB.idH][MB.idC[MB.idH]], &tabBuf );
      PutString( TabHelp.x, TabHelp.y, MB.child[MB.idH][MB.idC[MB.idH]].help );
      u.i = bioskey(0);
      ClearText( TabHelp.x, TabHelp.y, TabHelp.x+
                  strlen( MB.child[MB.idH][MB.idC[MB.idH]].help ), TabHelp.y );
      DelTab( MB.child[MB.idH][MB.idC[MB.idH]], &tabBuf );
      if( u.c[0] )
      { switch( u.c[0] )
          { case Enter:
                  if( MB.child[MB.idH][MB.idC[MB.idH]].stat == ENABLE )
                  { EraseBoard( &brd );
                    return ENTER;
                  }
                  else
                  { Beep();
                    break;
                  }
            case Esc:
                  EraseBoard( &brd );
                  return ESC;
            default:
                  k = FindKey( MB.child[MB.idH], MB.nC[MB.idH], u.c[0] );
                  if( (k != -1) && (MB.child[MB.idH][k].stat == ENABLE) )
                  { MB.idC[MB.idH] = k;
                    EraseBoard( &brd );
                    return ENTER;
      } }     }
      else
      { switch( u.i )
          { case  UP_AIRROW:
                  MB.idC[MB.idH] = (MB.idC[MB.idH] == 0)?
                                   MB.nC[MB.idH]-1 : MB.idC[MB.idH]-1;
                  break;
            case DOWN_AIRROW:
                  MB.idC[MB.idH] = (MB.idC[MB.idH] == MB.nC[MB.idH]-1)?
                                   0 : MB.idC[MB.idH]+1;
                  break;
            case RIGHT_AIRROW:
                  EraseBoard( &brd );
                  return RIGHT;
            case LEFT_AIRROW:
                  EraseBoard( &brd );
                  return LEFT;
    } } } }

    int  FindKey( TAB tab[], int n, char key )
    { uchar i;
      for( i=0; i<n; i++ )
      { if( tolower( key ) == tab[i].k || toupper( key ) == tab[i].k )
            return i;
      }
      return -1;
    }

    int GetNChar( int x, int y, int n, char *str )
    { char      buf[81], buf2[81];
```

```
    unsigned char   pos;
    unsigned char   len;
    union { char c[2];
            int  z;
          } u;

    len = strlen( str );
#ifdef DEBUG
    if( (x+n > 80) | (len > n) )
      Abnormal( "Getnchar() : x+n > 80 | strlen( str ) > n" );
#endif
    ClearText( x, y, x+n-1, y );
    PutString( x, y, strcpy( buf, str ) );
    pos = len==n? len-1 : len;
    gotoxy( x+pos, y );
    CursorOn();
    u.z = bioskey(0);
    if( isprint( u.c[0] ) )
    {  ClearText( x, y, x+n-1, y );
      len = pos = 0;
      buf[pos] = NULL;
    }
    for( ; ; )
    {  if( u.c[0] )
      {  switch( u.c[0] )
          {  case Esc   : CursorOff();
                          return  0;
             case Enter : buf[len] = NULL;
                          strcpy( str, buf );
                          CursorOff();
                          return  len;
             case Bckspc: if( pos > 0 )
                          {  pos--;
                           Putcxy( x+len-1, y, ' ' );
                           Putcxy( x+pos, y, ' ' );
                           PutString( x+pos, y, strcpy( &buf[pos], &buf[pos+1] ) );
                           len--;
                          }
                          break;
             default    : if( isprint( u.c[0]) )
                          {  if( len < n ) /* if increase lenght 1 char so less than n */
                            {  if( pos < len )        /* insert */
                                  strcpy( &buf[pos+1], strcpy( buf2, &buf[pos]));
                              buf[pos] = u.c[0];
                              buf[len+1] = NULL;
                              PutString( x+pos, y, &buf[pos] );
                              len++;
                              pos++;
                            }
                            else if( len == n ) /* if increase so equal */
                            {  if( pos < len )
                              {  buf[pos] = u.c[0];
                                    buf[len] = NULL;
                                    PutString( x+pos, y, &buf[pos] );
                                    pos++;
                              }
                              else  /* pos == len */
                              {  buf[pos] = u.c[0];
```

```
                                        buf[len] = NULL;
                                        PutString( x+pos, y, &buf[pos] );
        } }         } } }
        else
        { switch( u.z )
            { case LEFT_AIRROW : if( pos > 0 )  pos--;  break;
              case RIGHT_AIRROW : if( pos < len ) pos++; break;
              case Home       : pos = 0;          break;
              case End        : pos = len;        break;
              case Delete     : if( len > pos )
                                  { Putcxy( x+len-1, y, ' ' );
                                    PutString( x+pos, y,
                                        strcpy( &buf[pos], &buf[pos+1] ));
                                    len--;
        } }             }
        gotoxy( x+( pos = (pos==n)? pos-1 : pos ), y );
        u.z = bioskey(0);
    } }

    int Ask( int nChoice, int fore, int back, int n,... )
    { char    choice;
      uchar   wide, len;
      char    **inform;
      va_list  argp;
      TAB  ans[] = { { NULL, NULL, 0, 0, 8, ENABLE, NULL },
                     { NULL, NULL, 0, 0, 8, ENABLE, NULL },
                     { NULL, NULL, 0, 0, 8, ENABLE, NULL }
                   };
      char    *Item[] = { "  Yes  ", "  No  ", " Cancel ", "  OK  " };
      BOARD    brd = { 0, 0, 0, 0, SINGLE, FLOAT, 0, 0, NULL };

      inform = (char **)malloc( sizeof( char * )*n );
      wide = nChoice*10;
      va_start( argp, n );
      for( choice = 0; choice < n; choice++ )
      { inform[choice] = va_arg( argp, char * );
        len  = strlen( inform[choice] );
        wide = (wide < len)? len : wide;
      }
      va_end( argp );

      brd.fore = fore;
      brd.back = back;
      brd.ix = 40 - wide/2;
      brd.iy = 10 - n/2;
      brd.ex = brd.ix + wide + 1;
      brd.ey = brd.iy + n + 3;

      ans[0].y = ans[1].y = ans[2].y = brd.ey - 1;
      ans[0].x = 42 - nChoice*5;
      ans[1].x = ans[0].x + 10;
      ans[2].x = ans[1].x + 10;
      switch( nChoice )
      { case 1: ans[0].item = Item[3]; break;
        case 2: ans[0].item = Item[3], ans[1].item = Item[2]; break;
        case 3: ans[0].item = Item[0], ans[1].item = Item[1],
                ans[2].item = Item[2]; break;
      }
```

VCOORDS.C

```
/* Code for implementing 800x600 virtual display area in Turbo C graphics */

#include <graphics.h>

#define VH  600      /* height */
#define VW  800      /* width */

int     px, py;     /* physical x and y */
double  xf, yf;     /* x and y translation factor */

/*  compute translation factors, call once at start of program */
void  setFactors( void )
{  px = getmaxx();
   py = getmaxy();
   xf = (double)(px) / VW;
   yf = (double)(py) / VH;
}

/* translate virt x to device x, call at output time */
int dx( int vx )
{  return( (int)( xf * vx ) );
}

/* translate virt y device y */
int dy( int vy )
{  return( (int)( py - (yf * vy) ) );
}
□
```

```
  BuiltBoard( &brd );
  for( choice = 0; choice < n; choice++ )
    CenterString( brd.iy+1+choice, inform[choice] );
  PutMenu( ans, nChoice );
  choice = MenuSelect( ans, nChoice, 1 );
  EraseBoard( &brd );
  free( inform );
  return choice;
}
/*
main()
{  int i;
  InitVid();
  i = Ask( 3, BLACK, GREEN, 2, "WARNING", "testing warning BBBBBBBBBBB" );

  getch();
}
*/
int IsDigString( char *str )
{  unsigned char i;
  for( i=0; str[i]; i++)
    if( (isdigit( str[i] ) == 0) && ( str[i] != '.' ) )
        return 0;
  return (int)i;
}


int  GetDigitString( int x, int y, int ndigit, char *str )
{  for(;;)
   {  if( GetNChar( x, y, ndigit, str ) )
        {  if( !IsDigString( str ) )
             Beep();
           else
             return 1;
        }
        else
          return 0;
} }

void Beep()
{  sound( 50 );
  delay( 50 );
  nosound();
}

#ifdef DEBUG
Abnormal( char *msg )
{  printf( " Abnormal programe terminate ! \n" );
  printf( msg );
  getch(); exit(0);
}
#endif
□
```

Data Loggers

File Run System

Data Loggers

Software for driving Analog to Digital Converter through LPT1 port.

This Software and it's Hardware belong to
Agricultural Engineering of KMIT'l

Developed by
Panumas Thongtanunam 34128027
Patama Temwootiroge 34128017
Advisor
Assit. Prof. Panmanus Sirisombroon

OK

Data Loggers

File Run System

About
Help..
Config...

Setup system configuration

Data Loggers

File Run System

System Configuration

Comment: Testing configuration

Channels

No.	type	maxim
1		
2		
3		
4		
5		
6		
7		
8		

Channel 1

type: Load Cell
maximum: 500
minimum: 0
unit: KgF

CLOSE OPEN

Recording

mount Avail.
0 0
:00:000 00:00:00:000
(hr:min:sec:msec)

Rate: 10 rec/sec

verage: 50 time

OK CANCEL Reset Load..

Data Loggers

File Run System

System Configuration

Comment:

Channels

No.	type	maximum	minimum	unit
1	Load Cell	500	0	kgF
2				
3	Speed Wheel	60	0	km/hr
4				
5	Torque Meter	2000	0	lb.in
6				
7	Rot. Speed	5000	0	rpm
8				

Recording

Amount Avail.
36000 122234
01:00:00:000 03:23:43:200
(hr:min:sec:msec)

Rate: 10 rec/sec

Average: 50 time

OK CANCEL Reset Load..