การสร้างเครื่องวิเคราะห์ความแตกต่างทางความร้อน 2

นายเอกราช โล่ห์เพชรัตน์

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต
ภาควิชานิสิกส์ประยุกต์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2536/

CONSTRUCTION OF DIFFERENTIAL THERMAL ANALYZER 2

Mr. EKARAT LOPETCHARRAT

A Special Project Submitted in Partial Fulfillment of the
Requirement for the Degree of Bachelor of Science
Department of Applied Physics
Faculty of Science
King Mongkut 's Institute of Technology Ladkrabang
1993

หัวข้อโครงงานพิเศษ การสร้างเครื่องวิเคราะห์ความแตกต่างทางความร้อน 2

โดย นายเอกราช โล่ห์เพชรัตน์
ภาควิชา ฟิสิกส์ประยุกต์
อาจารย์ที่ปรึกษา ผศ.ดร. อารีย์ วิเชียรฉาย

ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับโครงงานพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

_____ หัวหน้าภาควิชาฟิสิกส์ประยุกต์

(ผศ. ปรีชา เทียนสมประสงค์)

คณะกรรมการโครงงานพิเศษ

_____ ประธานกรรมการ

(ผศ.ดร.อารีย์ วิเชียรฉาย)

_____ กรรมการ

(รศ.สุรพล รักวิจัย)

_____ กรรมการ

(ดร.ฐิตินัย แก้วแดง)

ลิขสิทธิ์ของภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อโครงงานพิเศษ การสร้างเครื่องวิเคราะห์ความแตกต่างทางความร้อน 2

นักศึกษา นายเอกราช โล่ห์เพชรัตน์
อาจารย์ที่ปรึกษา ผศ.ดร. อารีย์ วิเชียรฉาย
ภาควิชา ฟิสิกส์ประยุกต์
ปีการศึกษา 2536

บทคัดย่อ

โครงงานนี้ได้ทำการศึกษาการทำงานของ เครื่องวิเคราะห์ความแตกต่างทางความร้อน ต่อจากโครงงานเดียวกันนี้ในปีการศึกษา 2535 โดยเพิ่มเติมในส่วนของการเก็บข้อมูลและแสดงผลด้วยคอมพิวเตอร์ เพื่อทำหน้าที่แทนเครื่องบันทึกผลแบบ X-Y ซึ่งมีราคาแพง ทำให้ได้ผลจากการวัด DTA สะดวกและง่ายต่อการแปลผล อีกทั้งยังสามารถบันทึกข้อมูลที่วัดไว้ในจานแม่เหล็กเพื่อนำกลับมาใช้ในภายหลังได้

Special Project Title	Construction of Differential Thermal Analyzer 2
Name	Mr. Ekarat Lopetcharrat
Special Project Advisor	Asst.prof.Dr. Aree Wichianchai
Department	Applied Physics
Academic Year	1993

Abstract

This senior project is the continuous working, which follows the last academic year project in 1992, that had researched and constructed the Differential Thermal Analysis instrument ,DTA. It is used for testing a thermal property of solid state material by increase the temperature linearly to the sample and reference, which both are in the same condition.

The reserchers,in 1992,built furnace, crucibles, differential DC amplifier and linearly rate controller. A result of working on this instrument shows in X-Y recorder; the temperature,X-axis, vs the different temperature between sample and reference,Y-axis

This academic year, in 1993 , The signal interface circuit was built. The IBM PC is used for collecting data and result it in graph.

กิตติกรรมประกาศ

ในการทำโครงงานพิเศษชิ้นนี้ ผู้จัดทำได้รับความช่วยเหลือจากบุคคลหลายท่าน ทางผู้จัดทำขอขอบคุณสำหรับน้ำใจของทุกๆ ท่านไว้ ณ ที่นี้ด้วย

ขอขอบคุณ

ท่านอาจารย์อารีย์ วิเชียรฉาย ผู้ให้คำแนะนำในการทำโครงงาน

ท่านอาจารย์วิชิต ศิริโชติ ที่ได้คอยให้คำปรึกษาและให้การช่วยเหลือในการทำโครงงานหลายอย่าง

คุณสุรวุฒิ กิจสัมพันธ์ คุณจักรกฤษณ์ จูเจริญ ผู้ให้ความช่วยเหลือในการสร้างวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

คุณธวัชชัย เอื้อมมนัสกุล คุณทรงวิทย์ เจริญรวยวัฒนา ผู้ให้คำแนะนำช่วยเหลือในการเขียนโปรแกรมและแนะนำเอกสาร

คุณกิตติชัย อภินทนาพงศ์ ผู้จัดทำข้อมูลจำลองสำหรับการทดสอบโปรแกรม

คุณพรศรี ตั้งพัฒนากิจเจริญ ผู้ช่วยเหลือในศึกษาการทำงานของเครื่องวิเคราะห์ความแตกต่างทางความร้อน

คุณมานิตรา นิรมิตศิริพงศ์ ผู้ให้การช่วยเหลือในการเขียนบทคัดย่อ

เจ้าหน้าที่ปฏิบัติการและภาควิชาเคมีอุตสาหกรรม คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้อนุเคราะห์สารตัวอย่างที่ใช้ในการทดสอบ

และขอขอบคุณบุคคลท่านอื่นๆ ที่มิได้กล่าวไว้ในที่นี้ด้วย

ผู้จัดทำโครงงาน

สารบัญ

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

การสร้างและศึกษาเครื่องมือวัดเป็นปัจจัยสำคัญ ในอันที่จะพัฒนาความรู้และ ขยายความสามารถทางวิทยาศาสตร์และเทคโนโลยีของประเทศ ดังที่เห็นได้จากประเทศที่มีความก้าวหน้าทางวิทยาศาสตร์ จะมีความสามารถในการผลิตเครื่องมือวัดที่มีประสิทธิภาพสูงไว้ใช้งานได้เอง สำหรับประเทศของเรานั้นเป็นประเทศหนึ่ง ที่มีการขยายตัวทางอุตสาหกรรมในอัตราสูง ณ จุดนี้การศึกษาหลักการและวิธีการสร้างเครื่องมือวัดไว้ใช้งานเองจึงเข้ามามีบทบาทสำคัญเป็นก้าวหนึ่ง ที่ช่วยพัฒนาความสามารถในการเรียนรู้และสร้างเทคโนโลยีของตนเองทั้งในด้านกิจการอุตสาหกรรมและงานวิจัยทางวิทยาศาสตร์

สำหรับเครื่องมือวิเคราะห์ความแตกต่างทางความร้อน (Differential Thermal Analysis, DTA) เป็นเครื่องมือที่ใช้กันอย่างแพร่หลายในงานวิจัยเกี่ยวกับคุณสมบัติเชิงความร้อนของสาร เครื่องมือดังกล่าวนี้ได้ทำการศึกษาและสร้างขึ้นมาตั้งแต่ โครงงานพิเศษในปีการศึกษาที่แล้วและในปีการศึกษานี้ ได้ทำการพัฒนาเครื่องมือนี้ให้สามารถติดต่อกับเครื่องคอมพิวเตอร์เพื่อนำผลที่วัดได้ไปทำการวิเคราะห์

หลักการของเครื่องมือนี้อาศัยการวัดความแตกต่างระหว่างอุณหภูมิของ สารอ้างอิงกับสารตัวอย่างที่อยู่ในสภาวะแวดล้อมเดียวกัน และมีการเพิ่มอุณหภูมิอย่างเป็นเชิงเส้น การสร้างเครื่องมือดังกล่าวในปีการศึกษาที่ผ่านมาประกอบด้วย เตาความร้อน, เครื่องควบคุมอัตราการเพิ่มอุณหภูมิ และเครื่องขยายสัญญาณจากหัววัดอุณหภูมิ ซึ่งค่าที่เราได้จากการวัดจะบอกถึงคุณสมบัติต่างๆ ของสาร เช่น

- พลังงานความร้อนที่ใช้ในการเปลี่ยนแปลงสถานะของสาร
- ค่าความจุความร้อนจำเพาะ
- ความบริสุทธิ์ของสารที่ใช้ในการทดสอบ
- ชนิดของสารทดสอบ
- ปริมาณของส่วนประกอบของสาร
- อัตรการเกิดปฏิกิริยา
- จุดหลอมเหลว, จุดเดือด ,จุดที่สารเกิดการเปลี่ยนแปลงโครงสร้าง

ในปีการศึกษานี้ได้มีการพัฒนาเครื่องมือดังกล่าวต่อมา โดยสร้างในส่วนเชื่อมต่อสัญญาณจากวงจรขยายสัญญาณจากหัววัดอุณหภูมิมาเข้ากับคอมพิวเตอร์ เพื่อให้คอมพิวเตอร์ทำการบันทึกและทำการวิเคราะห์ข้อมูล

1.1 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาหลักการทำงานและวิธีการสร้างเครื่องมือวัดขึ้นใช้เอง
2. เพื่อพัฒนาเครื่องมือวัดที่มีอยู่เดิมให้มีการเก็บบันทึก วิเคราะห์และแสดงผลได้สะดวกและถูกต้องมากขึ้น
3. เพื่อนำผลจากการศึกษามาใช้ในการพัฒนาความรู้ขั้นต่อไปในอนาคต

1.2 วิธีการดำเนินการ

1. ศึกษาข้อมูลการสร้างจากโครงงานในปีการศึกษาที่ผ่านมา
2. ศึกษาข้อมูลเพิ่มเติมในด้านหลักการเชื่อมต่อเข้ากับคอมพิวเตอร์และ ด้านการวิเคราะห์ผลข้อมูลด้วยคอมพิวเตอร์
3. แบ่งโครงงานออกเป็น 2 ส่วนประกอบด้วย

3.1 ส่วนของฮาร์ดแวร์ ประกอบด้วย วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล วงจรจ่ายแรงดันและทำการดัดแปลงวงจรขยายสัญญาณจากหัววัดอุณหภูมิ

3.2 ส่วนของซอฟท์แวร์ ประกอบด้วยโปรแกรมย่อย ซึ่งจะทำหน้าที่ต่างๆ ไปและจะถูกเรียกใช้โดยโปรแกรมหลัก

4. สร้างส่วนต่างๆ ที่กล่าวไว้ในข้อ 3 ขึ้นมาแล้วทำการทดสอบเบื้องต้น โดยการจำลองภาวะการทำงานที่จะเกิดขึ้นให้กับส่วนประกอบแต่ละส่วน
5. นำส่วนประกอบที่ได้จากข้อ 4 มาประกอบรวมกันแล้วทดสอบการทำงานหลายๆ ครั้งแล้วทำการเฉลี่ยค่าที่ได้
6. นำค่าที่ได้มาเปรียบเทียบกับค่ามาตรฐานหากมีค่าผิดพลาดจากค่ามาตรฐานมากให้ทำการหาสาเหตุและแก้ไข แล้วทำตามขั้นตอนที่ 5 ซ้ำ หากค่าที่ได้มีค่ายอมรับได้ให้ไปทำขั้นตอนที่ 7
7. สรุปผลและปัญหาในการดำเนินงานเพื่อใช้ในการปรับปรุงต่อไป

1.3 ประโยชน์ที่ได้รับ

1. ข้อมูลที่ได้จากการทำโครงงานสามารถนำไปพัฒนาต่อให้มีประสิทธิภาพสูงขึ้น
2. สามารถเผยแพร่ความรู้เกี่ยวกับหลักการทำงานของเครื่องมือวัดชนิดนี้ให้กับผู้สนใจ
3. ผู้จัดทำโครงงานได้ทราบหลักการที่อยู่เบื้องหลังการทำงานของเครื่องมือ
4. ได้ทราบวิธีการหาข้อมูลเพื่อใช้ในทำโครงงาน
5. ได้ทราบวิธีการนำเสนอผลการดำเนินงานทั้งในรูปของการอภิปรายและในรูปของเอกสารรายงาน

บทที่ 2
หลักการพื้นฐานของเครื่องวิเคราะห์ความแตกต่างทางความร้อน

2.1. หลักการทำงานพื้นฐาน

เครื่องมือวิเคราะห์ความแตกต่างทางความร้อน(DTA) อาศัยหลักการวัดความแตกต่างของอุณหภูมิระหว่างสารตัวอย่าง(sample) และ สารอ้างอิง(reference)ที่อยู่ในสภาพแวดล้อมเดียวกันและได้รับพลังงานความร้อนเท่ากัน ภาชนะใส่สารตัวอย่างและสารอ้างอิงจะอยู่ในเตาไฟฟ้า โดยที่ภาชนะใส่สารทั้งสองวางอยู่ในตำแหน่งสมมาตรกันเพื่อให้สารตัวอย่างและสารอ้างอิงได้รับปริมาณความร้อนเท่ากัน

เครื่องมือนี้ใช้ในการศึกษาพลังงานความร้อนที่เกิดขึ้นในปฏิกิริยาเคมี การเปลี่ยนแปลงทางสถานะ พลังงานที่ใช้ในการเปลี่ยนแปลงโครงสร้างของสสาร เป็นต้น และนอกจากนี้ยังสามารถแสดงถึงปฏิกิริยาของสารตัวอย่างว่า เป็นปฏิกิริยาดูดความร้อนซึ่งมี ΔH เป็นบวกหรือปฏิกิริยาคายความร้อนซึ่งมี ΔH เป็นลบ การบันทึกผลการทดลองของเครื่องมือชนิดนี้จะบันทึกโดยการเขียนกราฟระหว่างอุณหภูมิแตกต่าง(ΔT) และ อุณหภูมิอ้างอิง(T)(ในบางเครื่องอาจจะใช้เวลาแทนอุณหภูมิอ้างอิง) จากหลักการพื้นฐานดังกล่าวมาข้างต้นสามารถแสดงเป็นแผนภาพได้ดังรูปที่ 2.1

ส่วนที่ 1 คือเครื่องควบคุมอัตราการเพิ่มของอุณหภูมิ ทำหน้าที่ควบคุมให้ความร้อนในเตาเพิ่มขึ้นอย่างสม่ำเสมอเป็นเชิงเส้น โดยใช้หัววัดอุณหภูมิเป็นตัวคอยตรวจสอบค่าของอุณหภูมิในเตาแล้วส่งให้เครื่องควบคุมรับรู้เพื่อส่งสัญญาณควบคุมออกมา

ส่วนที่ 2 คือเตาไฟฟ้าภายในเตาจะมีลักษณะทรงกระบอกกลวง มีภาชนะใส่สารตัวอย่างและสารอ้างอิงโดยภาชนะทั้งสองจะวางอยู่ในตำแหน่งที่สมมาตรกัน และมีหัววัดอุณหภูมิต่ออยู่กับภาชนะใส่สารทั้งสองเพื่อคอยตรวจสอบค่าของอุณหภูมิ

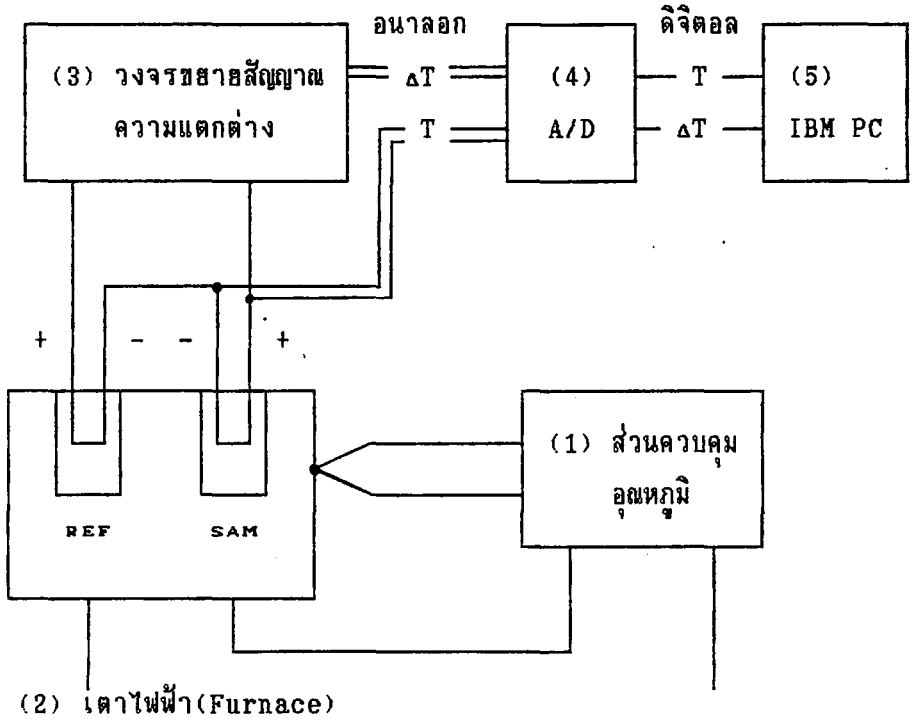
ส่วนที่ 3 คือส่วนขยายสัญญาณความแตกต่าง เนื่องจากสัญญาณที่ได้มีค่าน้อยมากจึงต้องนำมาขยายก่อนจึงสามารถนำมาแสดงผลได้

ส่วนที่ 4 คือส่วนแปลงสัญญาณอนาลอก ที่ได้มาจากส่วนขยายสัญญาณให้เป็นสัญญาณดิจิตอล(Analog to Digital Converter) เพื่อจะส่งข้อมูลไปแสดงผล จัดเก็บและทำการวิเคราะห์ที่เครื่องคอมพิวเตอร์ ซึ่งตามปกติเราสามารถแสดงผลข้อมูลออกมาได้ในโดยต่อกับเครื่องบันทึกผลแบบ X-Y (X-Y recorder) แล้วทำการวิเคราะห์เอาเอง

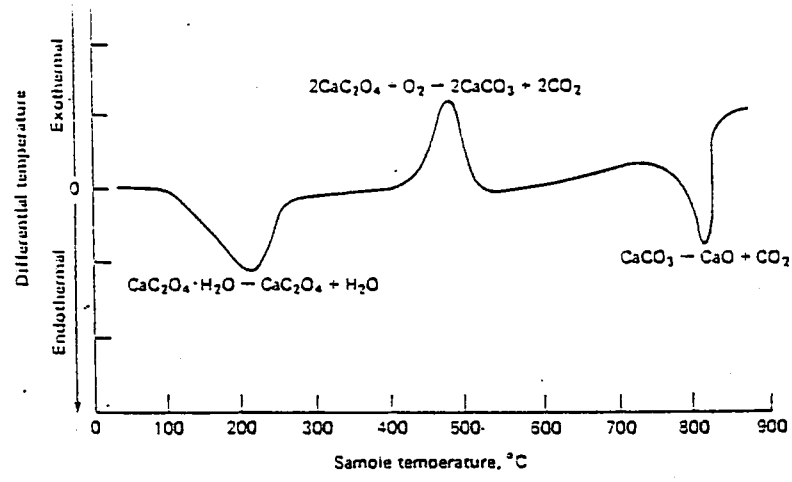
ส่วนที่ 5 คือคอมพิวเตอร์ที่ใช้สำหรับแสดงผลข้อมูลในรูปของกราฟ จัดเก็บข้อมูล และใช้ในการวิเคราะห์ข้อมูล ในส่วนนี้จะรับสัญญาณดิจิตอลจากส่วนที่ 4 ในลักษณะของนิกัด X-Y โดยที่ความแตกต่างอุณหภูมิจะอยู่ในแกนตั้ง ส่วนอุณหภูมิอ้างอิงจะอยู่ในแกนนอนจากนั้นจะนำค่าทั้งสองมาแสดงบนกราฟ และทำการเก็บลงบนจานแม่เหล็ก ซึ่งจะทำให้เราสามารถเรียกข้อมูลขึ้นมาแสดงซ้ำได้ตามต้องการ

ตัวอย่างของสัญญาณที่ได้จากเครื่องแสดงดังรูปที่ 2.2

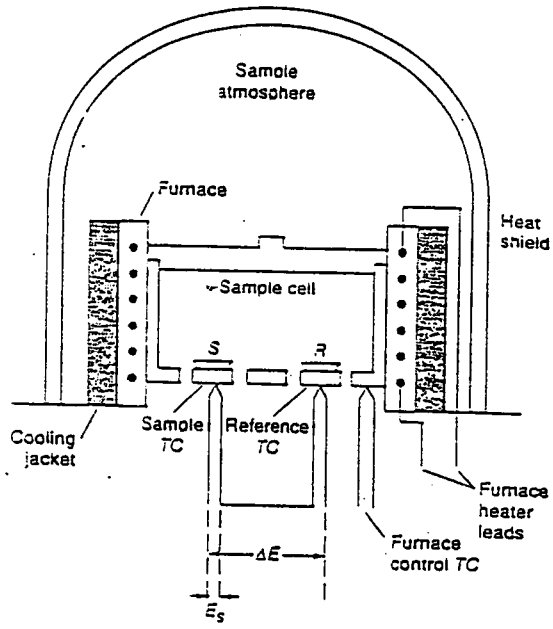
โครงสร้างภายในของเตาไฟฟ้าแสดงไว้ในรูปที่ 2.3



รูปที่ 2.1 ส่วนประกอบของเครื่องวิเคราะห์ความแตกต่างทางความร้อน [5]



รูปที่ 2.2 ตัวอย่างสัญญาณที่ได้จากเครื่องวิเคราะห์ความแตกต่างทางความร้อน [1]



รูปที่ 2.3 แสดงโครงสร้างภายในพื้นฐานของเตาไฟฟ้า [1]

ลักษณะของสัญญาณที่ได้แสดงไว้ในรูปที่ 2.2 จะสังเกตได้ว่ามีลักษณะเป็นพีค(peak)ซึ่งเป็นจุดที่เราสนใจเนื่องจากเป็นส่วนที่จะให้ข้อมูลกับเราหลายอย่าง เช่น

- ชนิดของปฏิกิริยาที่เกิดขึ้นเป็นแบบคายความร้อน(Exothermic)หรือ ดูดความร้อน(Endothermic) โดยดูได้จากลักษณะของพีค ถ้าพีคตั้งขึ้นก็เป็นปฏิกิริยาคายความร้อน ถ้าพีคลงก็เป็นปฏิกิริยาดูดความร้อน
- อุณหภูมิที่เกิดปฏิกิริยาหาได้จากอุณหภูมิที่ตรงกับยอดพีคเช่น จุดหลอมเหลว ฯลฯ
- พลังงานความร้อนที่ใช้ไปในการทำปฏิกิริยา(Enthalpy,ΔH) ซึ่งจะหาได้จากพื้นที่ใต้กราฟและต้องทราบค่าคงที่ของเครื่องก่อนแล้วจึงหาจากสมการ

$$\Delta H = \int_{T1}^{T2} K\ \Delta T dT$$

โดย T1 คืออุณหภูมิเริ่มต้นของการเกิดปฏิกิริยา
T2 คืออุณหภูมิสุดท้ายของการเกิดปฏิกิริยา

ตัวอย่างของอุณหภูมิ และพลังงานที่ใช้ในการทำปฏิกิริยาของสารชนิดต่างๆ ที่เป็นมาตราฐานแสดงไว้ในตารางที่ 2.1

เมื่อให้ความร้อนแก่สารแล้วสารเกิดการเปลี่ยนแปลงแบบดูดความร้อนนั้น มีสาเหตุมาจาก

- น้ำที่เป็นส่วนประกอบภายในของสารแตกตัวแยกออกไป
- โครงสร้างของสารเกิดการแตกสลาย
- เกิดการหลอมเหลว หรือ กลายเป็นไอ
- สารเกิดการเปลี่ยนแปลงโครงสร้างภายใน
- สภาพแม่เหล็กของสารเกิดการเปลี่ยนแปลงมักเกิดในสารพวกเฟอร์โรแมกเนติก

แต่ถ้าสารนั้นเกิดการเปลี่ยนแปลงแบบคายความร้อนจะมีสาเหตุมาจาก

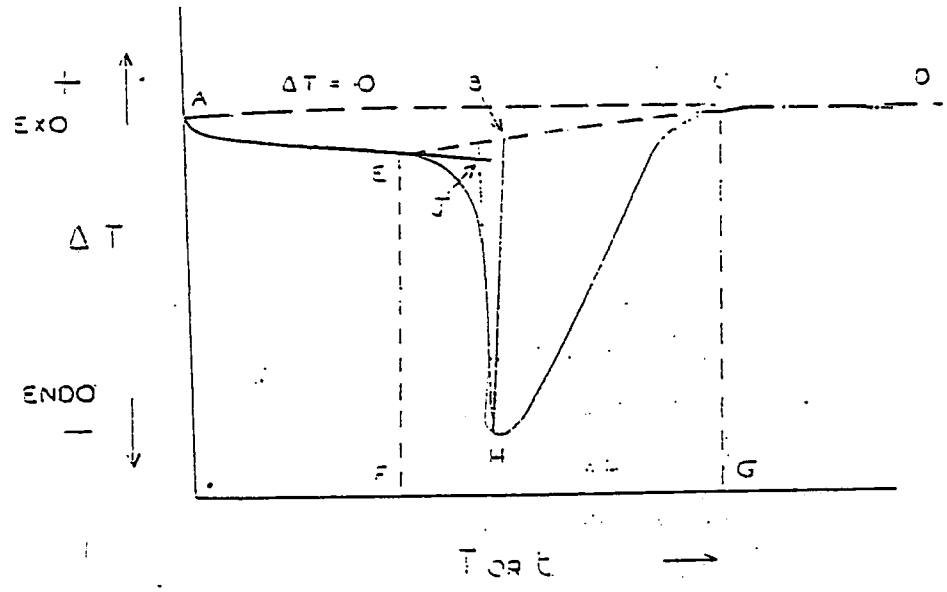
- เกิดผลึกขึ้นมาใหม่ภายในสาร
- เกิดการเผาไหม้

จากข้อมูลดังกล่าวสามารถใช้วิเคราะห์การเกิดปฏิกิริยาในสารตัวอย่างได้

ตารางที่ 2.1 ตัวอย่างของอุณหภูมิและพลังงานที่ใช้ในการทำปฏิกิริยาของสารชนิดต่างๆ [5]

Substance	Kind of transformation	Temperature (° C)	Heat of reaction (cal/g)	ΔH-fusion (kcal/Mole)
Gallium	Melting point	29.8		1.335
NH_4NO_3	Structural transf.	32		
NH_4NO_3	Structural transf.	85		
NH_4NO_3	Structural transf.	125		
KNO_3	Structural transf.	127.8		
Na_2SO_4	Structural transf.	147		
AgJ	Structural transf.	147		
$AgNO_3$	Structural transf.	160		
NH_4NO_3	Melting point	170		
$AgNO_3$	Melting point	212	16.7	
Na_2SO_4	Structural transf.	215		
Tin	Melting point	231.9 ± 0.001		1.68
Bismuth	Melting point	271.0		2.63
AgCl	Melting point	307		
$NaNO_3$	Melting point	314	45.3	
Lead	Melting point	327.4		1.141
KNO_3	Melting point	339	25.3	
Ag_2SO_4	Structural transf.	432		
AgJ	Melting point	552		
Na_3AlF_6 (cryolite)	Structural transf.	562.7		
K_2SO_4	Structural transf.	583.5		
Na_2MoO_4	Structural transf.	642		
Ag_2SO_4	Melting point	652		
Aluminium	Melting point	660.0 ± 0.2		2.57
Na_2MoO_4	Melting point	687		
KCl	Melting point	775		
NaCl	Melting point	801		
Witherite:				
$BaCO_3$	Structural transf.	810 ± 1.0		
Silver	Melting point	961	25.0	2.70
Norsethite. $BaMg(CO_3)_2$	Structural transf.	968		
Witherite:				
$BaCO_3$	Structural transf.	980		

2.2 นิยามต่างๆ ของรูปสัญญาณที่ได้จากเครื่องวิเคราะห์ความแตกต่างทางความร้อน



รูปที่ 2.4 รายละเอียดต่างๆ บนเส้นสัญญาณที่ได้จาก
เครื่องวิเคราะห์ความแตกต่างทางความร้อน [5]

จากรูปที่ 2.4 แสดงรายละเอียดต่างๆ บนเส้นสัญญาณ

- เส้นประ AD เรียกว่าเส้นฐาน(base line) คือเส้นที่แสดงว่า $\Delta T = 0$ ซึ่งหมายถึงอุณหภูมิของสารทดสอบไม่แตกต่างกับอุณหภูมิของสารอ้างอิง

- เส้น AE และ CD จะแสดงถึงช่วงอุณหภูมิที่สารทดสอบไม่เกิดปฏิกิริยา(ΔT มีค่าประมาณศูนย์) กราฟที่ได้จะประมาณเป็นเส้นฐาน

- ในขณะที่สารทดสอบเกิดปฏิกิริยา คายความร้อนหรือดูดความร้อน จะทำให้เส้นกราฟแยกตัวออกจากเส้นฐาน และเมื่อปฏิกิริยาสิ้นสุดลง เส้นกราฟจะกลับสู่เส้นฐานอีกครั้ง จากรูปที่ 2.4 เป็นเส้นสัญญาณของสารทดสอบที่เกิดปฏิกิริยาดูดความร้อน ซึ่งจะเห็นได้ว่าอุณหภูมิของสารทดสอบจะต่ำกว่าอุณหภูมิของสารอ้างอิง ซึ่งทำให้ค่า ΔT ติดลบ และถ้าหากว่าสารทดสอบเกิดปฏิกิริยาคายความร้อนสารทดสอบก็จะมีอุณหภูมิสูงกว่าสารอ้างอิงคือค่า ΔT เป็นบวก

- เส้นไข่ปลา EC หมายถึงความกว้างของรูปสัญญาณพีค ซึ่งจะถูกกำหนดในรูปของเวลาหรืออุณหภูมิจากจุดที่กราฟแยกออกจากเส้นฐาน จนถึงจุดที่กราฟกลับคืนสู่เส้นฐานอีกครั้ง

- ระยะ BH เป็นตัวบอกอุณหภูมิแตกต่างของสารตัวอย่างกับสารอ้างอิง โดยบอกรูปขององศาเซลเซียส หรือ องศาเคลวิน

- พื้นที่ปิด EHCE จะเป็นสัดส่วนกับพลังงานความร้อนที่ใช้ในการเกิดปฏิกิริยาหรือเอนทาลปี(Enthalpy) สามารถนำมาใช้ในการประมาณจำนวนสารที่ใช้ในการทดสอบได้ ในการหาค่าของพื้นที่ปิดนี้มักจะประมาณค่าโดยแทนพื้นที่ด้วยรูปสามเหลี่ยมแต่ถ้าจะให้ละเอียดต้องทำการประมาณค่าด้วยการนำข้อมูลที่ได้มาทำการวิเคราะห์เชิงตัวเลข

บทที่ 3
การแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล

เนื่องจากสัญญาณที่ได้จากวงจรขยายความแตกต่างและอุณหภูมิอ้างอิง เป็นสัญญาณอนาลอกแต่เครื่องคอมพิวเตอร์จะทำงานด้วยสัญญาณดิจิตอลเท่านั้น จากความแตกต่างนี้จึงจำเป็นต้องอาศัยเครื่องมือสำหรับทำหน้าที่แปลงสัญญาณจากอนาลอก(จากส่วนขยายสัญญาณ) มาเป็นสัญญาณทางดิจิตอลเพื่อส่งข้อมูลไปให้คอมพิวเตอร์ดำเนินการต่อ เครื่องมือที่ได้กล่าวมานี้ก็คือ "วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล (Analog to Digital Converter)" ซึ่งต่อไปนี้จะเรียกว่า "เอทูดี" ที่ใช้กันอยู่ก็มีให้เลือกมากมายหลายชนิดแต่ละชนิดก็มีคุณสมบติและหลักการทำงานแตกต่างกันออกไป ในที่นี้จะกล่าวถึงเอทูดีชนิดพื้นฐาน แบ่งตามความเร็วของการแปลงสัญญาณได้ 3 ชนิดดังต่อไปนี้

1. แบบใช้วงจรเปรียบเทียบขนาน หรือแบบ "แฟลช"(Parallel Comparator Simultaneous or "Flash" A/D converter) เป็นแบบที่เร็วที่สุดใช้เวลาแปลงอยู่ในระดับนาโนวินาที จึงมักจะใช้ในงานที่ต้องการแปลงสัญญาณให้ดูต่อเนื่อง เช่น สัญญาณภาพ แต่ราคาก็แพงที่สุดด้วยเช่นกัน

2. แบบใช้การประมาณค่า(Successive Approximation A/D converter) วงจรนี้มีความเร็วในการแปลงสัญญาณปานกลางอยู่ในระดับไมโครวินาที มีข้อได้เปรียบในด้านความละเอียด ราคาปานกลาง ใช้แปลงสัญญาณในวงจรเครื่องเสียงได้

3. แบบอินทิเกรตสัญญาณ วงจรเอทูดีที่ใช้หลักการนี้มีอยู่หลายชนิดแต่ในที่นี้จะกล่าวถึง วงจรเอทูดีแบบสโลปคู่(Dual Slope A/D converter) วงจรแบบนี้เป็นแบบที่ช้าที่สุด ใช้เวลาในการแปลงสัญญาณในระดับมิลลิวินาที(ช้ากว่าแบบแฟลชล้านเท่า) จึงมักจะใช้ในการแปลงที่ไม่ต้องการความเร็วมากเช่น ดิจิตอลมัลติมิเตอร์และเครื่องมือวัดอื่นๆ อีกหลายชนิด ข้อดีของวงจรนี้คือ ความถูกต้องสูง ราคาถูก เสถียรภาพทางด้านอุณหภูมิดี

ในโครงงานพิเศษนี้ใช้วงจรแปลงสัญญาณแบบสโลปคู่ เป็นวงจรรวม(ICL7109) ดังที่แสดงรายละเอียดไว้ในภาคผนวก

3.1 แบบใช้วงจรเปรียบเทียบขนานหรือแบบ "แฟลช"

วงจรเอทูดีแบบนี้ใช้หลักการง่ายๆ และเป็นวิธีที่รวดเร็วที่สุด คือใช้วงจรเปรียบ-

เทียบที่ต่อขนานกัน ดังรูปที่ 3.1 ประกอบด้วยออปแอมป์(Op Amp) ที่ต่อเป็นวงจรเปรียบ-เทียบและตัวต้านทานต่อไว้เพื่อแบ่งแรงดันที่ขาอินพุตแบบกลับ(inverting) ให้มีขนาดต่างๆ กัน

จากหลักการของวงจรเปรียบเทียบเมื่อแรงดันอินพุตแบบไม่กลับ(noninverting) มีค่าสูงกว่าที่ขาอินพุตแบบกลับ เอาต์พุตจะได้แรงดันค่าสูง ดูได้จากตารางที่ 3.1 จะทำให้เข้าใจได้ยิ่งขึ้นว่าที่แรงดันค่าต่างๆ มีผลต่อเอาต์พุตของวงจรเปรียบเทียบแต่ละตัวอย่างไร ซึ่งเอาต์พุตที่ได้จากวงจรเปรียบเทียบนี้จะนำไปเข้ารหัสให้เป็นเลขฐานสองต่อไป

เอาต์พุตของวงจรเปรียบเทียบที่ใช้ในวงจรนี้ ขึ้นอยู่กับขนาดของสัญญาณอนาลอกที่อินพุตจากวงจรรูปที่ 3.1 ถ้าแรงดันอินพุตมีค่าตั้งแต่ 0 ถึง 1 โวลต์ ไม่เพียงพอที่จะทำให้วงจรเปรียบเทียบตัวใดให้ค่าเอาต์พุตเป็น "1"

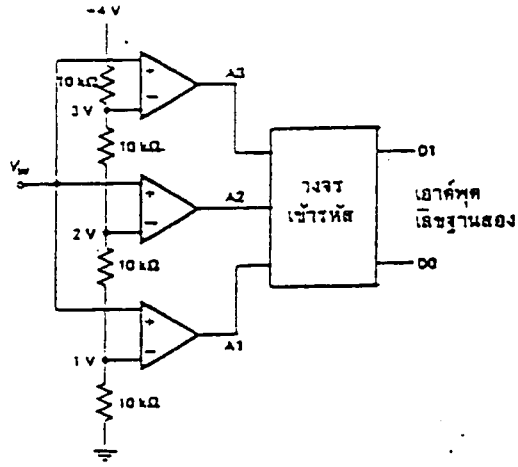
ที่แรงดันอินพุตระหว่าง 1 ถึง 2 โวลต์ วงจรเปรียบเทียบที่ระดับต่ำสุด(ในที่นี้คือ A1)จะให้เอาต์พุต "1" ออกมา

แรงดัน 2 ถึง 3 โวลต์ วงจรเปรียบเทียบ A1 และ A2 จะให้เอาต์พุตเป็น "1" ถ้าแรงดันที่อินพุตตั้งแต่ 3 โวลต์ขึ้นไปวงจรเปรียเทียบจะให้ค่าเป็น "1" ทั้งหมด

เมื่อต้องการวงจรที่มีความละเอียดสูงขึ้น จำเป็นต้องใช้วงจรเปรียบเทียบเพิ่มขึ้น เช่น ถ้าต้องการความละเอียด 3 บิตต้องใช้วงจรเปรียบเทียบ 7 ตัว ความละเอียด 4 บิต ต้องใช้วงจรเปรียบเทียบ 15 ตัว(16 ระดับ) หาจำนวนวงจรเปรียบเทียบได้จาก 2^N-1 เมื่อ N แทนจำนวนบิตหรือความละเอียดที่ต้องการ

จะเห็นว่าที่ความละเอียด 8 บิตต้องใช้วงจรเปรียบเทียบมาถึง 255 ตัว ซึ่งเป็นข้อเสียของวงจรเอทูดีแบบนี้

ข้อเสียอีกข้อคือ เอาต์พุตที่ได้จะไม่เป็นเลขฐานสองต้องเพิ่มเติมวงจรเข้ารหัสเข้าไปจึงจะใช้งานได้ ข้อดีของวงจรเอทูดีแบบนี้ก็คือ ความเร็วสูงมาก จึงเรียกวงจรเอทูดีแบบนี้ว่าแบบ "แฟลช" (Flash type converter) วงจรเอทูดีชนิดนี้ใช้เวลาในการแปลงน้อยมากในระดับนาโนวินาทีเท่านั้น



รูปที่ 3.1 แสดงวงจรเอทูดีแบบใช้วงจรขนานเปรียบเทียบ
หรือแบบ "แฟลช" [12]

แรงดันอินพุต V_{in} (โวลต์)	เอาต์พุตของวงจรเปรียบเทียบ			เอาต์พุตเลขฐานสอง	
	A1	A2	A3	D1	D2
0 - 1	0	0	0	0	0
1 - 2	1	0	0	0	1
2 - 3	1	1	0	1	0
>3	1	1	1	1	1

ตารางที่ 3.1 ตารางแสดงความสัมพันธ์ระหว่างอินพุตที่เป็นอนาลอก
กับเอาต์พุตที่เป็นดิจิตอล [12]

3.2 วงจรเอทูดีแบบใช้วิธีการอินทิเกรตสัญญาณชนิดสโลปคู่

จากรูปที่ 3.2 แสดงผังวงจรของวงจรเอทูดีแบบสโลปคู่ ส่วนแรกของวงจรคือวงจรกำเนิดสัญญาณแรมป์(Ramp) หรือวงจรอินทิเกรเตอร์(Integrator) นั่นเอง ที่อินพุตแบบกลับของออปแอมป์มีจะสภาพเป็นกราวน์เทียม(virtual ground) ถ้ามีแรงดันอินพุต 2 โวลต์ จะได้กระแสไหลผ่านตัวต้านทาน 10 กิโลโอห์ม เท่ากับ 0.2 มิลลิแอมป์ไปยังจุดรวม (summing point) เนื่องจากค่าความต้านทานอินพุตของออปแอมป์นั้นสูงมาก กระแสที่ไหลจึงเกิดขึ้นผ่านตัวเก็บประจุ

ขณะที่ตัวเก็บประจุกำลังทำการรับประจุ แรงดันที่เอาต์พุตของออปแอมป์ก็จะยิ่งเป็นลบมากขึ้นเรื่อยๆ เพื่อรักษาระดับกระแสให้คงที่ แรงดันคร่อมตัวเก็บประจุจึงได้เป็นสัญญาณแรมป์ที่เป็นเชิงเส้น(linear ramp)

ถ้าแรงดันอินพุตเป็นบวก วงจรอินทิเกรเตอร์จะให้เอาต์พุตเป็นสัญญาณแรมป์ทางลบดังแสดงไว้ในช่วง t_1 รูปที่ 3.3 หากแรงดันอินพุตเป็นลบก็จะทำให้เอาต์พุตได้แรมป์ทางบวก

ความชันของสัญญาณแรมป์ สามารถคำนวณได้จากความสัมพันธ์ของประจุ $q = cv$ และ $q = It$ โดยจับสองสมการมาเท่ากัน

$$\Delta V / \Delta T = I/C \qquad (3.1)$$

เมื่อทราบว่ากระแสเท่ากับ V_{in}/R เราก็รู้ว่า

$$\Delta V / \Delta T = V_{in}/RC \qquad (3.2)$$

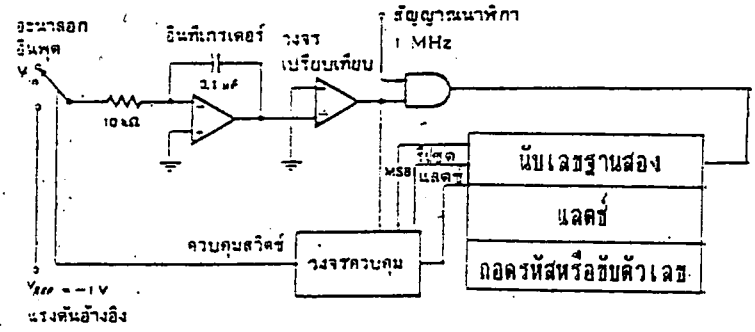
จากรูปให้แรงดันอินพุต +2 โวลต์ ก็จะได้ความชันของสัญญาณแรมป์ทางเอาต์พุตเท่ากับ -2 V/ms

จากวงจรในรูปที่ 3.2 อธิบายได้คือเมื่อสวิตช์ต่อกับสัญญาณอินพุตจะทำให้มีแรงดันบวกจากอินพุตป้อนเข้าสู่วงจรอินทิเกรเตอร์ ได้เอาต์พุตออกมาเป็นแรมป์ทางลบ

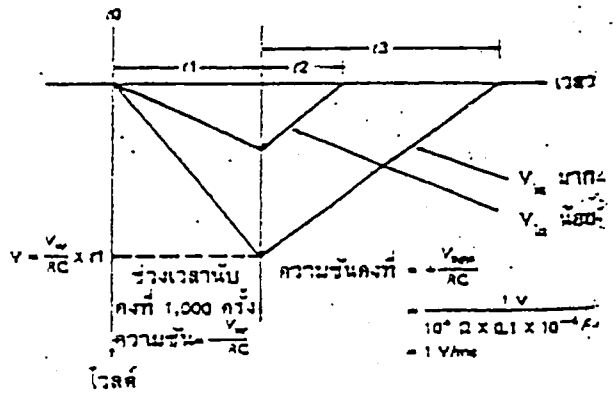
วงจรเปรียบเทียบก็จะได้แรงดันลบจากวงจรอินทิเกรเตอร์ แล้วให้เอาต์พุตเป็นบวกทำการเปิดแอนด์เกต(AND gate) ให้สัญญาณนาฬิกาผ่านเข้าไปสู่วงจรนับ วงจรนับจะนับไปยังค่าที่กำหนดไว้คงที่(t_1) แล้วทำการสับสวิตช์ต่อเข้ากันกับแรงดันอ้างอิง

ในช่วงที่วงจรนับ นับด้วยค่าที่คงที่นั้น วงจรอินทิเกรเตอร์จะให้สัญญาณแรมป์ทางลบที่มีค่าไต่สูงสุดตามแต่ระดับแรงดันอินพุต เมื่อทำการสวิตช์อินพุตของวงจรอินทิเกรเตอร์ให้ไปที่แรงดันอ้างอิงค่าลบ เอาต์พุตของวงจรจึงได้เป็นแรมป์ทางบวกช่วง t_2 รูปที่ 3.3 พร้อมๆ กับทำการรีเซต(reset) วงจรนับให้เป็นศูนย์เพื่อเริ่มนับใหม่

เมื่อเอาต์พุตของวงจรอินทิเกรเตอร์ไต่ขึ้นมา(ช่วง t_2) จนกระทั่งถึงแรงดันศูนย์ เอาต์พุตของวงจรเปรียบเทียบก็จะกลายเป็นลบ(หรือศูนย์) วงจรควบคุมจับการเปลี่ยนแปลงอันนี้ได้ก็ส่งสัญญาณสโตรบ(strobe) ให้เก็บค่าที่ได้ไว้ในวงจรแลตช์(latch) จากนั้นรีเซตวงจรนับให้เป็นศูนย์แล้วทำการสวิตช์วงจรอินทิเกรเตอร์ต่อกับแรงดันอินพุต เป็นการเริ่มทำการแปลงสัญญาณอีกครั้งหนึ่ง จำนวนที่นับได้ก็จะเป็นสัดส่วนโดยตรงกับแรงดันอินพุต V_{in}



รูปที่ 3.2 ผังวงจรของเอทูดีชนิดสโลปคู่ [12]



รูปที่ 3.3 เอาต์พุตของวงจรอินทิเกรเตอร์เทียบกับเวลา [12]

สัญญาณแรมป์ทางเอาต์พุตของวงจรอินทิเกรเตอร์ในช่วงเวลาคงที่ t_1 จะลดลงสู่แรงดัน V ซึ่ง

$$V = (V_{in}/RC) \times t_1 \qquad (3.3)$$

และเพื่อให้กลับไปสู่ระดับศูนย์ วงจรอินทิเกรเตอร์จึงต้องสร้างแรมป์ทางบวกให้มีค่าเพิ่มขึ้นเท่าๆ กันในช่วงเวลา t_2 (ซึ่งเกิดจากแรงดันอ้างอิง) จะได้แรงดัน V เท่ากับ

$$V = (V_{Ref}/RC) \times t_2 \qquad (3.4)$$

สมการทั้งสองสามารถจับมาเท่ากันได้เป็น

$$(V_{in}/RC) \times t_1 = (V_{ref}/RC) \times t_2$$
$$V_{in} \times t_1 = V_{ref} \times t2$$
$$t_2 = V_{in} \times (t_1/V_{ref}) \qquad (3.5)$$

เห็นได้ว่า RC ปรากฏอยู่ทั้งสองข้างของสมการจึงสามารถตัดทิ้งได้หมายถึง ช่วงเวลาในการอินทิเกรตสัญญาณอินพุตและช่วงเวลาในการอินทิเกรตสัญญาณอ้างอิง จะใช้ตัวต้านทานและตัวเก็บประจุตัวเดียวกัน ดังนั้นการเปลี่ยนแปลงค่าทั้งสอง จะไม่มีผลต่อความถูกต้องของสัญญาณเอาต์พุต จากสมการจะเห็นได้ว่าเอาต์พุตของวงจรนับในช่วงเวลา t_2 นั้นจะเป็นสัดส่วนโดยตรงกับแรงดันอินพุต V_{in} เมื่อ V_{Ref} และ t_1 คงที่

จากรูปที่ 3.2 และ 3.3 ให้ t_1 เท่ากับ 1000 รอบ เมื่อป้อนสัญญาณนาฬิกา 1 MHz และ V_{Ref} เท่ากับ -1 โวลต์

ถ้าให้สัญญาณอินพุตขนาด 2 โวลต์ จะได้ช่วงเวลา t_2 = 2000 รอบ(การนับ) ถ้าเราใส่จุดทศนิยมทางขวาของภาคแสดงผลก็จะได้ 2.000

กราฟในรูปที่ 3.3 แสดงว่าเมื่อสัญญาณอินพุตน้อยกว่านี้จะมีการเปลี่ยนแปลงเช่นไรบ้าง เช่น อินพุต 0.8 โวลต์ จะได้ t_2 = 800 รอบ ภาคแสดงผลก็จะแสดงค่าที่ได้เป็น 0.800 หลักการเช่นนี้ถูกนำไปใช้ในดิจิตอลมัลติมิเตอร์และเครื่องมือวัดหลายๆ ชนิด

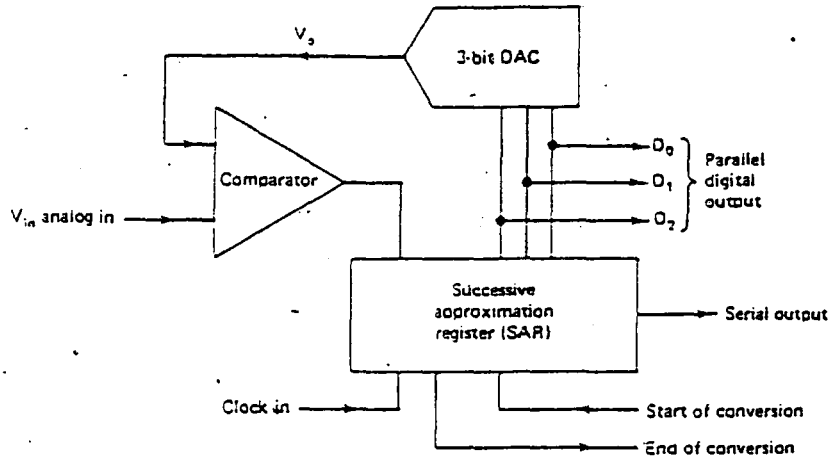
สรุปอีกครั้งได้ว่า แรงดันอินพุตที่ไม่ทราบค่าถูกป้อนเข้ามาในวงจรอินทิเกรเตอร์ เมื่อครบช่วงเวลา t_1 วงจรนับก็จะถูกรีเซตเป็นศูนย์ อินพุตของวงจรอินทิเกรเตอร์ก็จะถูก

สวิตช์ต่อกลับไปยังแรงดันอ้างอิง(ที่มีแรงดันคงที่) ให้ความชันของสัญญาณแรมป์เพิ่มค่าขึ้นไปจนถึงระดับศูนย์(ใช้เวลาไป t_2) ช่วงเวลา t_2 นี้ จะเป็นสัดส่วนโดยตรงกับแรงดันอินพุตที่เราไม่ทราบค่าพิจารณารูปที่ 3.3 เปรียบเทียบระหว่างแรงดันอินพุตสองค่าแล้วจะเข้าใจยิ่งขึ้น

ข้อดีของวงจรเอทูดีชนิดสโลปคู่ก็คือ ความถูกต้องสูง ราคาถูกและมีเสถียรภาพทางด้านอุณหภูมิดี(เพราะว่าการเปลี่ยนแปลงของค่าของตัวต้านทานและตัวเก็บประจุเนื่องมาจากอุณหภูมิจะไม่มีผลต่อความถูกต้องของวงจร)

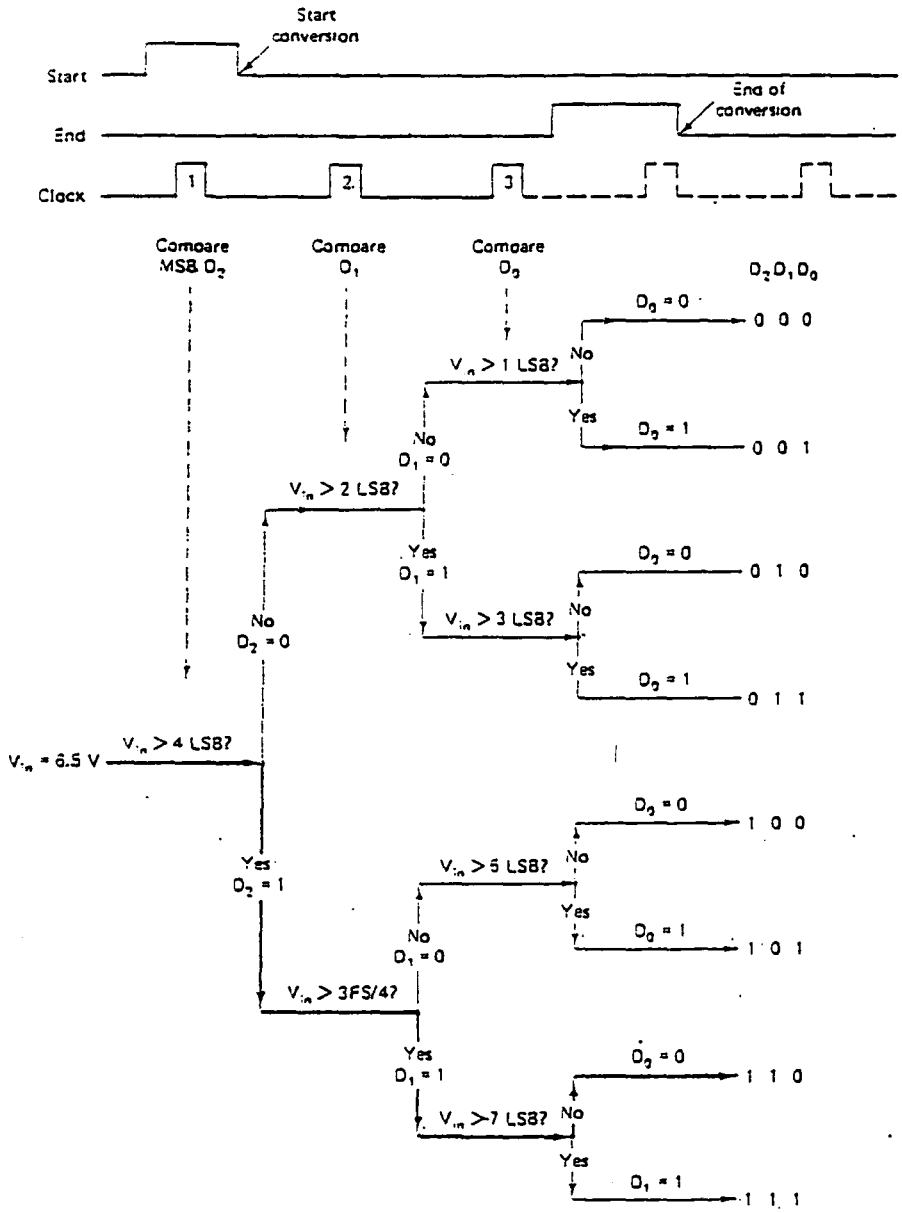
ข้อเสียของวงจรแบบนี้ก็คือความเร็วในการแปลงสัญญาณต่ำ(อยู่ในระดับมิลลิวินาที)

3.3 วงจรเอทูดีแบบใช้การประมาณค่า(Succesive Aproximation)



รูปที่ 3.4 ผังวงจรของเอทูดีแบบใช้การประมาณค่า(Successive Aproximation) ความละเอียด 3 บิต [4]

จากรูปที่ 3.4 แสดงผังของวงจรเอทูดีแบบใช้การประมาณค่าซึ่งมีขั้นตอนการทำงานดังนี้ เริ่มจากมีสัญญาณมากระตุ้นการทำงานเพื่อให้เริ่มแปลงสัญญาณ(สัญญาณจะเข้าที่ขา Start of conversion ดูรูปประกอบ) จากรูปรีจิสเตอร์สำหรับการประมาณค่า(SAR) จะต่อเข้ากับขาเอาต์พุตสัญญาณดิจิตอล(ขา D_0 D_1และD_2) ซึ่งแต่ละขาจะแทนบิตหนึ่งบิต อีกด้านหนึ่งของขาเอาต์พุตก็จะต่อเข้ากับตัวแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก(Digital-



รูปที่ 3.5 แสดงขั้นตอนการทำงานของวงจรเอทูดีแบบใช้การประมาณค่ามีจำนวนครั้งการเปรียบเทียบเท่ากับจำนวนบิต(3 บิต) [4]

to Analog Converter) ซึ่งจะทำหน้าที่แปลงสัญญาณจากเอาต์พุต(D_0 D_2และD_3) เป็นแรงดัน V_0 เพื่อนำไปเปรียบเทียบกับสัญญาณอินพุต V_{in} สัญญาณที่ได้จากวงจรเปรียบเทียบจะถูกส่งกลับไปที่รีจิสเตอร์เพื่อบอกผลของการเปรียบเทียบว่า แรงดันอินพุทมากกว่าหรือน้อยกว่าแรงดันเอาต์พุต ซึ่งการเปรียบเทียบเช่นนี้จะทำจนกระทั่งครบทุกบิตโดยเริ่มจากบิตที่มีนัยสำคัญสูงสุดก่อนจนถึงบิตที่มีนัยสำคัญต่ำสุด เมื่อการเปรียบเทียบเสร็จสิ้นแล้วรีจิสเตอร์ก็จะส่งสัญญาณสิ้นสุดการแปลง(end of conversion)ออกมา

เพื่อให้เห็นภาพการทำงานชัดเจนยิ่งขึ้นจึงเสนอตัวอย่างประกอบซึ่งจะใช้วงจรตามแบบรูปที่ 3.4 และ ขอให้ดูรูปที่ 3.5 ประกอบด้วย

สมมติให้สัญญาณอินพุตที่ต้องการแปลง(V_{in}) มีค่าเท่ากับ 6.5 โวลต์ เมื่อเริ่มทำการแปลง(ที่สัญญาณนาฬิกาลูกที่ 1)รีจิสเตอร์จะเซตค่าบิตที่มีนัยสำคัญสูงสุดให้เป็น "1" นอกนั้นเป็น "0" (จะได้ $D_2D_1D_0$ = 100) จากนั้นทำการแปลงเป็นสัญญาณอนาลอกเพื่อนำไปเปรียบเทียบกับสัญญาณอินพุต(ในที่นี้ได้ V_0 = 4 โวลต์) ถ้าหากว่า $V_{in} > V_0$ บิตที่ทำการเปรียบเทียบจะถูกเซตเป็น "1" ในทางตรงกันข้ามถ้า $V_{in} < V_0$ บิตที่ทำการทดสอบก็จะเป็นศูนย์ ในขณะนี้ $V_{in} > V_0$ ดังนั้นบิต D_2 จึงถูกเซตเป็น "1" จบสัญญาณนาฬิกาลูกแรก

คราวนี้มาดูที่สัญญาณนาฬิกาลูกที่สอง รีจิสเตอร์จะทำการเซตบิต D_1 ให้เป็น "1" (บิตที่นัยสำคัญต่ำกว่าจะเป็น "0" ส่วนบิตที่ทำการเปรียบเทียบไปแล้วจะคงค่าที่ได้จากการเปรียบเทียบไว้)ในขณะนี้จะได้ $D_2D_1D_0$ = 110 เมื่อผ่านการแปลงเป็นสัญญาณอนาลอกแล้วจะได้ V_0 = 6 โวลต์ ซึ่งยังน้อยกว่าสัญญาณอินพุต ดังนั้นบิต D_2 จึงถูกเซตเป็น "1"

ท้ายสุดนี้(สัญญาณนาฬิกาลูกที่สาม) รีจิสเตอร์จะทำการเซตบิตนัยสำคัญต่ำสุดให้เป็น "1" ส่วนบิตอื่นๆ คงเดิมไว้(ในขณะนี้ได้ $D_1D_2D_3$ = 111) เมื่อผ่านการแปลงเป็นอนาลอกจะได้ค่า V_0 = 7 โวลต์ ซึ่งมากกว่าสัญญาณอินพุต D_0 จะถูกเซตเป็น "0"

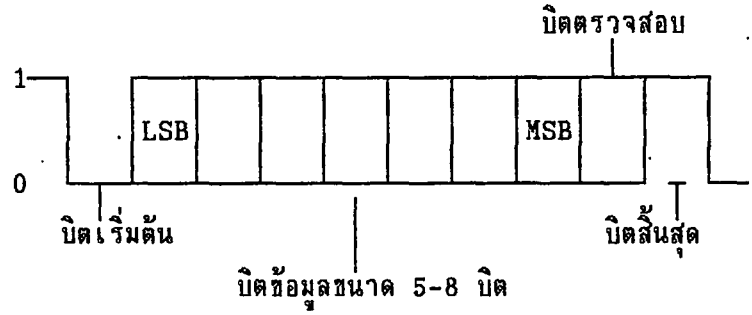
เมื่อทำการเปรียบเทียบมาจนถึงบิตสุดท้ายแล้ว รีจิสเตอร์ก็จะส่งสัญญาณสิ้นสุดการแปลงออกมา ผลลัพธ์ที่ได้จะเป็น $D_1D_2D_3$ = 110 จะเห็นว่าค่าที่ได้จะผิดไปจากค่าจริงบ้าง หากทำการเปรียบเทียบโดยใช้จำนวนบิตมากกว่านี้ก็จะได้ค่าที่ละเอียดขึ้น

บทที่ 4

การติดต่อผ่านพอร์ตอนุกรม(Serial Port)

โครงงานพิเศษนี้ได้ทำการแปลงสัญญาณอนาลอกมาเป็นสัญญาณดิจิตอล มาเก็บไว้ที่ไมโครคอนโทรลเลอร์ 8751 จากนั้นจึงให้ 8751 ส่งข้อมูลมายังเครื่องคอมพิวเตอร์โดยผ่านทางพอร์ตสื่อสารอนุกรม(Serial Port) ดังที่จะได้อธิบายต่อไป

ก่อนที่จะได้อธิบายการทำงานของพอร์ตอนุกรมนั้น ควรจะได้ทำความเข้าใจกับวิธีการสื่อสารระยะไกลซึ่งจะแบ่งออกเป็น 2 วิธีคือ แบบอซิงโครนัส(Asynchronous) และแบบซิงโครนัส(Synchronous) แต่ในที่นี้จะกล่าวถึงเฉพาะวิธีการติดต่อแบบอซิงโครนัส การรับส่งข้อมูลด้วยวิธีนี้เป็นการรับส่งโดยกำหนดให้มี บิตเริ่มต้น(Start Bit) บิตสิ้นสุด (Stop Bit) เป็นตัวบอกจุดเริ่มต้นและจุดสิ้นสุดของข้อมูล เมื่อใดที่มีการส่งข้อมูลออกไปจะส่งบิตเริ่มต้นออกไปก่อน แล้วจึงส่งข้อมูลตามไปติดๆ ซึ่งจำนวนบิตของข้อมูลเราสามารถกำหนดได้ว่าจะใช้เท่าไร(5-8 บิต) ในการส่งข้อมูลจะส่งบิตที่มีนัยสำคัญต่ำสุด(LSB)ออกไปก่อนจนถึงบิตที่มีนัยสำคัญสูงสุด(MSB) หลังจากนั้นจึงทำการส่งบิตตรวจสอบ(Parity Bit) เพื่อทำการทดสอบความถูกต้องของข้อมูล(สามารถกำหนดให้มีหรือไม่ก็ได้ หรือจะกำหนดให้เป็นบิตตรวจสอบชนิดคู่หรือคี่ก็ได้) ซึ่งปกติจะไม่ค่อยกำหนดให้มีการใช้เพื่อประหยัดเวลาในการส่งข้อมูล และท้ายสุดจะเป็นการส่งบิตสิ้นสุดเพื่อบอกจุดสิ้นสุดของข้อมูลซึ่งสามารถกำหนดขนาดของบิตสิ้นสุดได้ การส่งข้อมูลแบบนี้จะเป็นดังรูปที่ 4.1



รูปที่ 4.1 รูปแบบการส่งข้อมูลแบบอซิงโครนัส [11]

จากรูปจะเห็นว่าบิตเริ่มต้นจะมีสถานะต่ำ(0) ทั้งนี้เนื่องจากในสภาวะปกติสายส่งจะมีสถานะสูง(1) ส่วนบิตข้อมูลและบิตตรวจสอบจะเป็นสถานะใดก็ได้ จากนั้นจะปิดท้ายด้วยบิตสิ้นสุดซึ่งมีสถานะเป็นสูง(1)

บิตตรวจสอบ จะทำหน้าที่ตรวจสอบความถูกต้องของข้อมูล บิตตรวจสอบมี 2 ชนิด คือเป็น คู่ หรือ คี่(Even or Odd) ถ้าเป็นคู่หมายความว่าเมื่อรวมบิตตรวจสอบแล้วจำนวนของบิตข้อมูลที่มีค่าเป็น 1 จะเป็นจำนวนคู่ และถ้าเป็นคี่ หมายความว่าเมื่อรวมบิตตรวจสอบจำนวนของบิตข้อมูลที่เป็น 1 จะเป็นจำนวนคี่

อัตราการรับส่งข้อมูล(Baud rate) มีหน่วยเป็น บิตต่อวินาที(baud) ค่าต่ำสุดที่มีการใช้กันคือ 110 บิตต่อวินาที ซึ่งเราสามารถกำหนดอัตราการรับส่งข้อมูลเป็นค่าอื่นได้ดังที่แสดงไว้ในตารางที่ 4.4 ในโครงงานนี้จะใช้อัตราการรับส่งข้อมูลที่ 9600 บิตต่อวินาที การเขียนโปรแกรมสำหรับใช้งานพอร์ตอนุกรม(ในโครงงานนี้ใช้ภาษาปาสคาล) สามารถทำได้ 3 วิธีคือ ผ่านทางดอส(DOS,Disk Operating System) ผ่านทางไบออสและสุดท้ายคือการเขียนโปรแกรมควบคุมพอร์ตอนุกรมโดยตรง

การเรียกใช้พอร์ตอนุกรมผ่านทางดอสนั้น ไม่เหมาะสมเท่าใดนักเนื่องจากว่าดอสไม่มีวิธีที่จะตรวจสอบสถานะของการรับส่งและตัวพอร์ตอนุกรมว่าการรับส่งข้อมูลถูกต้องเพียงใด ดังนั้นในรายงานฉบับนี้จะกล่าวถึงเฉพาะการเรียกใช้พอร์ตอนุกรมโดยผ่านไบออส และการโปรแกรมควบคุมพอร์ตโดยตรง(ในโครงงานนี้เลือกใช้วิธีหลังเนื่องจากโปรแกรมทำงานได้เร็วกว่าเมื่อเปรียบเทียบที่อัตราการรับส่งข้อมูลเดียวกัน)

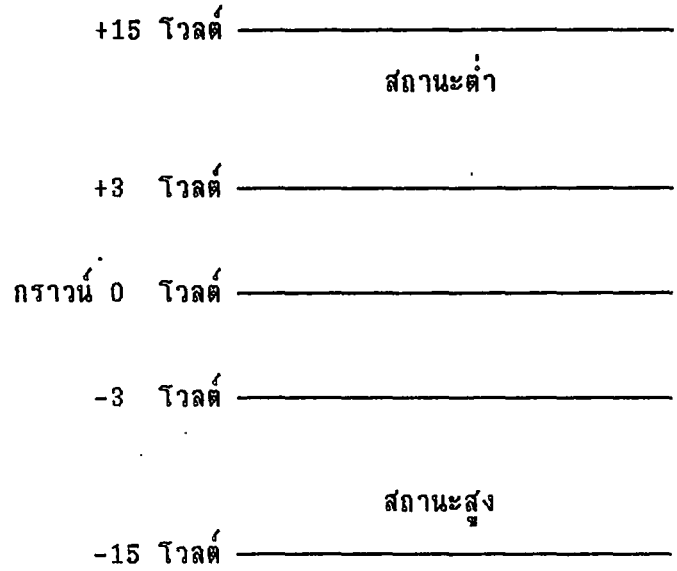
4.1. มาตรฐาน RS-232C

มาตรฐาน RS-232C คือมาตรฐานสำหรับเชื่อมต่อคอมพิวเตอร์ ซึ่งมาตรฐานนี้เป็นการส่งข้อมูลแบบอซิงโครนัส กำหนดขึ้นโดยสถาบัน Electronics Industrials Association(EIA) มาตรฐานนี้เป็นที่ยอมรับและนิยมใช้กันแพร่หลายในปัจจุบันนี้ และในโครงงานพิเศษนี้ก็ได้นำมาใช้ในการติดต่อกันระหว่าง 8751 กับ คอมพิวเตอร์

* **หมายเหตุ** มาตรฐานการเชื่อมต่อสัญญาณแบบอนุกรม มีการกำหนดขึ้นมาหลายมาตรฐาน ซึ่งมีคุณสมบัติแตกต่างกันไป สำหรับในที่นี้จะกล่าวเฉพาะการเชื่อมต่อสัญญาณตามมาตรฐาน RS-232C เท่านั้น

4.1.1. ลักษณะของสัญญาณ RS-232C

สัญญาณของ RS-232C จะตรงข้ามกับความเป็นจริงคือสถานะสูงมีระดับแรงดัน -3 ถึง -15 โวลต์ แต่ส่วนใหญ่จะใช้ -12 โวลต์ และ สถานะต่ำมีระดับแรงดันตั้งแต่ +3 ถึง +15 โวลต์(ดูรูปที่ 4.2 ประกอบ)



รูปที่ 4.2 แสดงลักษณะสัญญาณของ RS-232C [13]

4.1.2. ความเร็วในการส่งสูงสุด 20000 บิตต่อวินาที

4.1.3. ระยะส่ง ไม่เกิน 50 ฟุต

4.1.4. การกำหนดขั้วต่อของ RS-232C

มาตรฐาน RS-232C ไม่ได้กำหนดชนิดของขั้วต่อไว้ แต่ที่ใช้กันมากมักจะเป็นขั้วต่อแบบ D(D-Subminiature connector) 25 ขา หรือ 9 ขา ซึ่งมีการจัดตำแหน่งของขาสัญญาณดังตารางที่ 4.1 และ ตารางที่ 4.2

ขั้วต่อแบบ D 25 ขา	
ขาที่	สัญญาณ
1	GND(Chassis Ground)
2	Tx(Transmit Data)
3	Rx(Receive Data)
4	RTS(Request to Send)
5	CTS(Clear to Send)
6	DSR(Data set Ready)
7	Ground
8	DCD(Data Carier Detect)
9	NC
10	NC
11	NC
12	NC
13	NC
14	NC
15	NC
16	NC
17	NC
18	NC
19	NC
20	DTR(Data Terminal Ready)

NC = NO CONNECT(ไม่ใช้งาน)

ตารางที่ 4.1 ตำแหน่งของขาสัญญาณบนขั้วต่อแบบ D 25 ขา [14] (มีต่อ)

ขั้วต่อแบบ D 25 ขา	
ขาที่	สัญญาณ
21	NC
22	RI(Ring Indicator)
23	NC
24	NC
25	NC

NC = NO CONNECT(ไม่ใช้งาน)

ตารางที่ 4.1(ต่อ) ตำแหน่งของขาสัญญาณบนขั้วต่อแบบ D 25 ขา [14]

ขั้วต่อแบบ D 9 ขา	
ขาที่	สัญญาณ
1	Chassis Ground
2	Tx (Transmit Data)
3	Rx (Receive Data)
4	RTS (Request to Send)
5	CTS (Clear to Send)
6	DSR (Data set Ready)

NC = NO CONNECT(ไม่ใช้งาน)

ตารางที่ 4.2 การจัดตำแหน่งของสัญญาณบนขั้วต่อแบบ D 9 ขา [14] (มีต่อ)

ขั้วต่อแบบ D 9 ขา	
ขาที่	สัญญาณ
7	GND (Signal Ground)
8	DCD (Data Carier Detect)
9	NC

NC = NO CONNECT(ไม่ใช้งาน)

ตารางที่ 4.2(ต่อ) การจัดตำแหน่งของสัญญาณบนขั้วต่อแบบ D 9 ขา [14]

4.1.5. ความหมายของสัญญาณ

Tx (Transmit Data)

เป็นขาที่ส่งสัญญาณข้อมูลออกไปยังจุดอื่นๆ

Rx (Receive Data)

เป็นขาที่รับสัญญาณข้อมูลจากจุดอื่นๆ

RTS (Request to Send)

เป็นขาที่บอกต่อเครื่องอื่นๆ ว่าตัวเองพร้อมที่จะส่งข้อมูลแล้ว

CTS (Clear to Send)

เป็นขาที่บอกกับเครื่องอื่นว่าพร้อมจะรับข้อมูลแล้ว

DSR (Data Set Ready)

เป็นขาที่บอกไมโครคอมพิวเตอร์ว่า โมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยและพร้อมที่จะส่งได้

Signal Ground

เป็นขากราวน์(Ground) ของสัญญาณ

DTR (Data Terminal Ready)

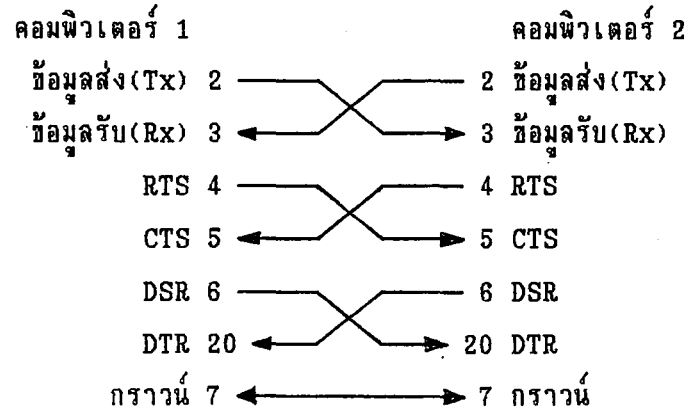
เป็นขาที่บอกโมเด็มว่าไมโครคอมพิวเตอร์พร้อมที่จะติดต่อด้วย

4.1.6. การเชื่อมต่อสัญญาณ

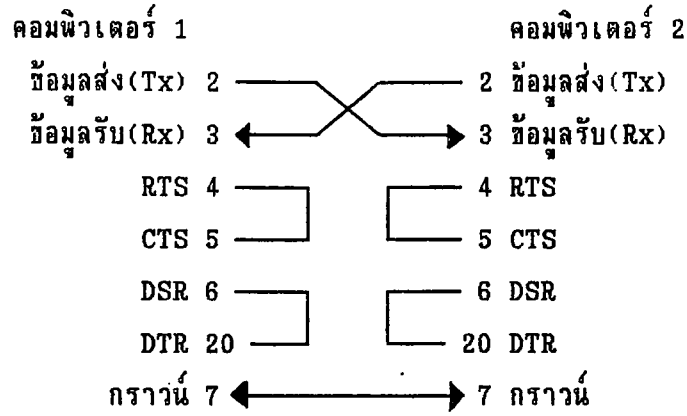
แต่เดิมนั้นพอร์ตอนุกรมถูกออกแบบมาเพื่อใช้งานร่วมกับโมเด็ม ดังนั้นเมื่อนำไปใช้กับอุปกรณ์อื่นบางสัญญาณจึงไม่จำเป็น เพราะสัญญาณเหล่านี้มีเพื่อใช้เป็นข้อตกลงระหว่างโมเด็มกับคอมพิวเตอร์ว่า คอมพิวเตอร์จะไม่ส่งหรือรับข้อมูลจนกว่าโมเด็มจะพร้อมเพื่อป้องกันความผิดพลาดในการรับส่งข้อมูลซึ่งมีชื่อเรียกของสัญญาณว่า RTS และ CTS ถ้าเป็นการติดต่อกันระหว่างคอมพิวเตอร์สองเครื่อง RTS ของเครื่องที่ 1 จะต่อกับ CTS ของเครื่องที่ 2 ดูได้จากรูปที่ 4.3 ลักษณะการตอบโต้แบบนี้เรียกว่า แฮนด์เชคกิ้ง(Handshaking)

เมื่อคอมพิวเตอร์ตัวรับพร้อมที่จะรับข้อมูลจะแจ้งสัญญาณไปที่ขา RTS (ซึ่งต่อกับ CTS คอมพิวเตอร์ตัวส่ง) จากนั้นคอมพิวเตอร์ตัวส่งจะตอบกลับที่ขา CTS(ซึ่งต่ออยู่กับขา RTSของคอมพิวเตอร์ตัวรับ) วิธีการดังที่กล่าวมาพอสังเขปนี้เรียกว่า ฮาร์ดแวร์แฮนด์เชคกิ้ง(hardware handshaking) นอกจากนี้ยังมีวิธีแฮนด์เชคกิ้งโดยใช้ซอฟท์แวร์เรียกว่า XON/XOF ซึ่งมีวิธีการคล้ายๆ กัน

วิธีการเชื่อมต่อสัญญาณที่ได้อธิบายข้างต้นเป็นวิธีการต่อที่มีฮาร์ดแวร์แฮนด์เชคกิ้ง ซึ่งมักจะใช้กับอุปกรณ์พวกโมเด็ม แต่ในโครงงานนี้จะใช้การเชื่อมต่ออีกแบบ ซึ่งเป็นแบบง่ายๆ ใช้เฉพาะสาย Tx RXและกราวน์ต่อกันระหว่างเครื่อง จากนั้นทำการลัดวงจรขา RTS กับขา CTS และลัดวงจรขา DSR กับ DTR เพื่อเป็นการตัดฮาร์ดแวร์แฮนด์เชคกิ้งดังที่แสดงไว้ในรูปที่ 4.4 แต่วิธีการเช่นนี้จะทำให้เกิดปัญหาข้อมูลถูกเขียนทับได้ ซึ่งต้องแก้ไขด้วยการใช้ซอร์ฟแวร์แฮนด์เชคกิ้งแทนฮาร์ดแวร์แฮนด์เชคกิ้ง



รูปที่ 4.3 แสดงการเชื่อมต่อสัญญาณโต้ตอบเพื่อป้องกันข้อผิดพลาดในการรับส่งข้อมูล [14]

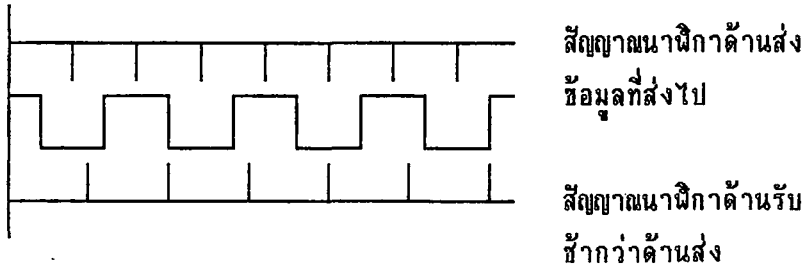


รูปที่ 4.4 แสดงการเชื่อมต่อสัญญาณแบบง่ายๆ ซึ่งใช้ในโครงงานนี้ [14]

4.1.7. ความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูล

4.1.7.1. ความผิดพลาดของกรอบข้อมูล(Framming Error)

ความผิดพลาดของกรอบข้อมูล คือ ความผิดพลาดของการส่งข้อมูลที่เกิดจากสัญญาณนาฬิกาที่ควบคุมการทำงานของอุปกรณ์ทั้ง 2 ด้านมีค่าไม่เท่ากัน เพราะว่าจากการทำงานของพอร์ตอนุกรมเมื่อพอร์ตได้รับบิตเริ่มต้นก็จะสุ่มอ่านค่าจากส่วนรับข้อมูล 1 ครั้งต่อ 1 รอบเพื่ออ่านบิตต่อไป ซึ่งระยะเวลาในการสุ่มอ่านแต่ละรอบนาฬิกาไม่ตรงกัน คอมพิวเตอร์ด้านรับก็จะอ่านข้อมูลจากส่วนรับข้อมูลของตน ช้าเกินไปหรือเร็วเกินไปก่อนที่ข้อมูลจะถูกส่งมาจากคอมพิวเตอร์ด้านส่งจึงทำให้เกิดความผิดพลาดขึ้น(พิจารณารูปที่ 4.5)



รูปที่ 4.5 แสดงความผิดพลาดของกรอบข้อมูล [2]

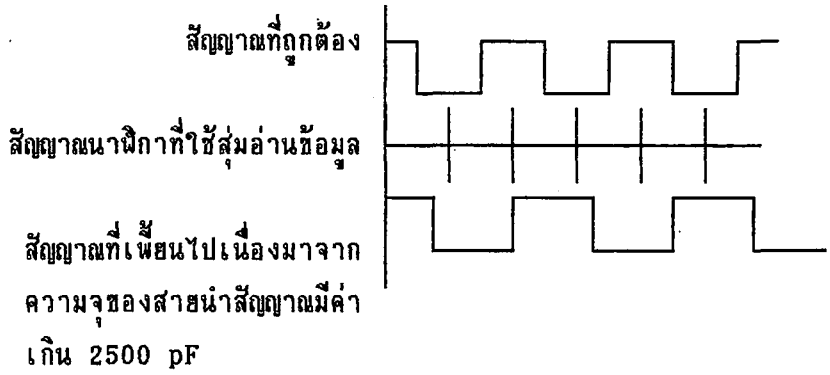
4.1.7.2. ความผิดพลาดจากการที่ข้อมูลถูกเขียนทับ(Overrun Error)

ความผิดพลาดชนิดนี้เกิดจากการที่คอมพิวเตอร์ด้านส่ง ทำการส่งข้อมูลมาที่คอมพิวเตอร์ด้านรับในขณะที่ด้านรับยังไม่พร้อมที่จะรับข้อมูล เนื่องจากยังไม่ได้อ่านข้อมูลเดิมเข้าไปเก็บมีผลทำให้ข้อมูลที่เข้ามาใหม่ทับลงไปที่ข้อมูลเดิม ข้อมูลเดิมก็จะหายไปซึ่งความผิดพลาดนี้สามารถป้องกันได้ด้วยวิธีการแฮนด์เชคกิ้ง

4.1.8. ข้อด้อยของการเชื่อมต่อตามมาตรฐาน RS-232C

1. ใช้ไฟเลี้ยง -15 โวลต์ นอกเหนือจากไฟเลี้ยงของสัญญาณทีทีแอล(TTL) ทำให้ยุ่งยากต่อการใช้งานเพราะต้องหาแหล่งจ่ายแรงดันเพิ่มเติม

2. ค่าความจุของสายนำสัญญาณ(Stay capacitance) จะต้องไม่เกิน 2500 pF สายที่นำมาใช้ส่งสัญญาณมักจะมีความจุประมาณ 50 pF/ฟุต ดังนั้นจึงเดินสายได้ยาวไม่เกิน 50 ฟุต เนื่องจากถ้าสายมีความจุเกิน 2500 pF จะทำให้เกิดความผิดพลาดของข้อมูลเกินกว่าที่จะยอมรับได้(4 เปอร์เซนต์)



รูปที่ 4.6 แสดงความเพี้ยนของสัญญาณอันเนื่องมาจากความจุของสายนำสัญญาณ [13]

3. ถ้ากราวน์ของด้านรับไม่ตรงกับกราวน์ของด้านส่ง จะทำให้การอ่านสถานะของสัญญาณผิดพลาด

4.2. การติดต่อกับพอร์ตอนุกรมผ่านไบออส(BIOS,Basic Input Output System)

อุปกรณ์แวดล้อมต่างๆ ที่เชื่อมต่อเข้ากับคอมพิวเตอร์จะมีโปรแกรมสำหรับควบคุมการทำงานที่เรียกว่า ไบออส ซึ่งจะประกอบด้วยโปรแกรมควบคุมฮาร์ดแวร์ต่างๆ ได้ 256 แบบ ซึ่งเราสามารถเรียกมาใช้ในการติดต่อกับอุปกรณ์ได้ ซึ่งคล้ายกับการเรียกโปรแกรมย่อยมาใช้งาน แต่การเรียกใช้ผ่านไบออสจะเรียกตามหมายเลขอินเทอรัพท์

*หมายเหตุ คำว่าอินเทอรัพท์(Interrupt) แปลเป็นภาษาไทยได้ว่า การขัดจังหวะ แต่ในที่นี้จะทับศัพท์ในกรณีที่หมายถึง โปรแกรมบริการที่เรียกใช้ผ่านไบออสซึ่งจะเรียกตามหมายเลขของอินเทอรัพท์ที่ต้องการจะใช้ (ในเครื่องไอบีเอ็ม พีซี(IBM PC)จะมีหมายเลขอินเทอรัพท์ได้ 256 หมายเลข)

4.2.1. การเตรียมสถานะเริ่มต้นของพอร์ต

ในการเตรียมสถานะของพอร์ตอนุกรมสามารถทำได้โดย ผ่านอินเทอรัพท์หมายเลข 14 ฟังก์ชันหมายเลข 0 โดยมีรีจิสเตอร์ AH เป็นตัวผ่านค่าหมายเลขของฟังก์ชัน(ในที่นี้คือเลข 0) รีจิสเตอร์ AL ผ่านค่ารหัสที่จะตั้งสถานะของพอร์ตอนุกรม โดยมีความหมายของแต่ละบิตดังตารางที่ 4.4

ตัวอย่างการใช้รหัสเพื่อเตรียมสถานะของพอร์ตอนุกรมเช่น ต้องการตั้งให้พอร์ตมีอัตราการรับส่งข้อมูล 9600 บิตต่อวินาที มีบิตตรวจสอบคู่(Even Parity) บิตสิ้นสุด 1 บิต และใช้บิตข้อมูล 8 บิตจะได้รหัสดังตารางที่ 4.3(ดูตารางที่ 4.4 ประกอบด้วย)

บิตที่	7	6	5	4	3	2	1	0
รหัส	1	1	1	1	1	0	1	1

ตารางที่ 4.3 ตัวอย่างการตั้งรหัสเพื่อเตรียมสถานะของพอร์ตอนุกรมโดยใช้รีจิสเตอร์ AL ผ่านค่ารหัส

หมายเลขประจำบิต	ความหมาย
7,6,5	อัตราการรับส่งข้อมูล 000 = 110 บิตต่อวินาที 001 = 150 บิตต่อวินาที 010 = 300 บิตต่อวินาที 011 = 600 บิตต่อวินาที 100 = 1200 บิตต่อวินาที 101 = 2400 บิตต่อวินาที 110 = 4800 บิตต่อวินาที 111 = 9600 บิตต่อวินาที
4,3	บิตตรวจสอบ 00 = ไม่มีบิตตรวจสอบ 10 = ไม่มีบิตตรวจสอบ 01 = บิตตรวจสอบชนิดคู่ 11 = บิตตรวจสอบชนิดคี่
2	จำนวนของบิตสิ้นสุด 0 = 1 บิตสิ้นสุด 1 = 2 บิตสิ้นสุด
1,0	จำนวนบิตข้อมูล 10 = 7 บิต 11 = 8 บิต

ตารางที่ 4.4 รหัสสำหรับการเตรียมสถานะเริ่มต้นของพอร์ต [2]

เครื่องคอมพิวเตอร์โดยทั่วไปจะมีพอร์ตอนุกรมได้มากถึง 7 พอร์ต(แต่จะมีพอร์ตมาตรฐานมาให้ 2 พอร์ตคือ COM1 และ COM2 ซึ่งต่อไปจะเขียนเป็นตัวอักษรภาษาอังกฤษเพื่อให้เห็นได้เด่นชัด) การกำหนดพอร์ตในกรณีเรียกผ่านไบออสจะให้ค่าหมายเลขพอร์ตผ่านรีจิสเตอร์ DX พอร์ตแรกมีหมายเลข 0 พอร์ตต่อไปคือหมายเลข 1 เช่นนี้ต่อไปเรื่อยๆ

*<u>หมายเหตุ</u> ในคอมพิวเตอร์บางเครื่องที่มีพอร์ตอนุกรมมาให้หลายๆ พอร์ตและมีการเชื่อมต่ออุปกรณ์เข้ากับพอร์ตเหล่านี้พร้อมๆ กันอาจจะทำให้เกิดปัญหาไม่สามารถใช้อุปกรณ์เหล่านี้ได้ในเวลาเดียวกัน ทั้งนี้เพราะพอร์ตอนุกรมมีการใช้การขัดจังหวะหมายเลขเดียวกันกล่าวคือได้มีการกำหนดข้อมูลพื้นฐานและการขัดจังหวะไว้ดังนี้

พอร์ต	แอดเดรส(Address)	หมายเลขการขัดจังหวะ(IRQ)
COM1	3F8h	IRQ4
COM2	2F8h	IRQ3
COM3	3E8h	IRQ4
COM4	2E8h	IRQ3

ตารางที่ 4.5 แสดงหมายเลขการขัดจังหวะสำหรับพอร์ตอนุกรม [2]

จากตารางจะเห็นว่า COM1 และ COM3 กับ COM2 และ COM4 มีหมายเลขการขัดจังหวะที่ซ้ำกันดังนั้นหากติดตั้งอุปกรณ์บน COM1 และ COM3 พร้อมๆ กัน(หรือ COM2 และ COM4) จะใช้งานไม่ได้ ในโปรแกรมที่ใช้ในโครงงานนี้จะกำหนดให้ใช้เฉพาะ COM1 หรือ COM2 เท่านั้น

4.2.2. การส่งข้อมูลขนาด 1 ไบท์ผ่านพอร์ตอนุกรม

จะใช้ฟังก์ชันที่ 1 ของอินเทอรัพท์หมายเลข 14 ของไบออสทำหน้าที่ส่งข้อมูลขนาด 1 ไบท์ผ่านทางพอร์ตอนุกรม โดยกำหนดหมายเลขพอร์ตผ่านรีจิสเตอร์ DX และข้อมูลที่จะส่ง

จะอยู่ในรีจิสเตอร์ AL เมื่อทำการส่งเรียบร้อยแล้ว สถานะของการส่งข้อมูลจะมาปรากฏที่รีจิสเตอร์ AH เพื่อใช้ตรวจสอบดูว่าการส่งข้อมูลถูกต้องหรือไม่ ถ้าบิตที่ 7 ของ AH เป็น 1 หลังจากเสร็จสิ้นการส่งข้อมูล แสดงว่าเกิดข้อผิดพลาดขึ้น จะต้องตรวจสอบสถานะของพอร์ตดูว่าเกิดข้อผิดพลาดอะไร ดังในหัวข้อต่อไป

4.2.3 การตรวจสอบสถานะของพอร์ตอนุกรม

ฟังก์ชันหมายเลข 3 ของอินเทอรัพท์ที่ 14 ของไบออส ใช้ตรวจดูสถานะของพอร์ตอนุกรม โดยผ่านหมายเลขพอร์ตทางรีจิสเตอร์ DX สถานะของรหัสจะผ่านมาทางรีจิสเตอร์ AH และ AL ดังตารางที่ 4.6 จะเห็นว่าสัญญาณสถานะต่างๆ ส่วนใหญ่จะใช้งานกับโมเด็ม ดังนั้นเมื่อนำมาใช้กับอุปกรณ์อื่น(เช่นที่ใช้ในโครงงานนี้) สัญญาณเหล่านี้จึงลดความสำคัญลง แต่ยังมีสัญญาณอีกสัญญาณหนึ่งที่มีความสำคัญคือ สัญญาณพร้อมรับข้อมูล(Data ready) ซึ่งเป็นตัวบอกว่าเมื่อไรที่ข้อมูลถูกรับเข้ามาและพร้อมที่จะอ่านเข้าไปเก็บ

สถานะของสายส่ง AH	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	บิตที่
Data Ready	0
Overrun Error	1
Parity Error	2
Framing Error	3
Break-Detect Error	4
Transfer hold-register empty	5
Tramsfer shift-register empty	6
Time-out Error	7

ตารางที่ 4.6 สถานะของพอร์ตอนุกรมในรีจิสเตอร์ AH และ AL [2] (มีต่อ)

สถานะของโมเด็ม AL	
ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	บิตที่
Change in clear-to-send	0
Change in data-set-ready	1
Trailing-edge ring detector	2
Change in line signal	3
Clear-to-send	4
Data-set-ready	5
Ring indicator	6
Line signal detector	7

ตารางที่ 4.6(ต่อ) สถานะของพอร์ตอนุกรมในรีจิสเตอร์ AH และ AL [2]

4.2.4. การรับข้อมูล 1 ไบท์ผ่านพอร์ตอนุกรม

การรับข้อมูลจากพอร์ตอนุกรมสามารถทำได้ โดยใช้ฟังก์ชันที่ 2 ของอินเทอรัพท์ที่ 14 รีจิสเตอร์ DX ใช้สำหรับกำหนดหมายเลขพอร์ต ข้อมูลที่อ่านได้จะเก็บไว้ในรีจิสเตอร์ AL เมื่ออ่านข้อมูลแล้วสามารถตรวจสอบได้ที่ บิต 7 ของ รีจิสเตอร์ AH

4.3. การเรียกใช้พอร์ตอนุกรมโดยการควบคุมพอร์ตโดยตรง

วิธีการนี้ไม่สะดวกเท่ากับวิธีติดต่อพอร์ตอนุกรมโดยผ่านไบออส แต่จากการทดลองเปรียบเทียบการทำงานของวิธีทั้งสอง วิธีนี้จะทำงานเร็วกว่าในโครงงานนี้เลือกใช้วิธีนี้ในการเรียกใช้พอร์ตอนุกรม

4.3.1. รีจิสเตอร์ของ 8250

ไอซี(IC) 8250 ทำหน้าที่ควบคุมการติดต่อสื่อสารผ่านทางโมเด็ม(รวมทั้งการติด

ต่อผ่านทาง RS-232 ด้วย) ไอซี 8250 มีรีจิสเตอร์ที่เอื้ออำนวยต่อการกำหนดค่าต่างๆ สำหรับการรับส่งข้อมูล การแสดงสถานะของการรับส่ง แสดงข้อผิดพลาด เป็นที่พักของข้อมูลก่อนและหลังการรับส่ง และเป็นตัวทำให้เกิดสัญญาณการขัดจังหวะเมื่อเกิดเหตุการณ์อย่างใดอย่างหนึ่งขึ้น รายละเอียดจะขอรวบรวมไว้ดังตารางที่ 4.7

หมายเลขพอร์ต		ลักษณะการใช้งาน	ชื่อรีจิสเตอร์
COM1	COM2		
3F8	2F8	Tx Buffer	RBR,DLL
3F8	2F8	Rx Buffer	RBR,DLL
3F8	2F8	Division Latch LSB	RBR,DLL
3F9	2F9	Division Latch MSB	DLM,IER
3F9	2F9	Interrupt Enable Register	DLM,IER
3FA	2FA	Interrupt Identification Register	IIR
3FB	2FB	Line Control Register	LCR
3FC	2FC	Modem Control Register	MCR
3FD	2FD	Line Status Register	LSR
3FE	2FE	Modem Status Register	MSR

ตารางที่ 4.7 หมายเลขพอร์ตและลักษณะการใช้งานรีจิสเตอร์ของ 8250 [11]

4.3.2. หน้าที่การทำงานของรีจิสเตอร์

รีจิสเตอร์ RBR(DLL) มีหน้าที่ 3 อย่างคือ เป็นที่พักของข้อมูลก่อนที่จะถูกส่งออกไปเป็นตัวรับข้อมูลที่ได้รับเข้ามาจากอนุกรมสื่อสาร และเป็นตัวกำหนดอัตราของการรับส่งข้อมูล(Baud rate) โดยเป็นตัวหารนัยสำคัญต่ำ ใช้งานร่วมกับรีจิสเตอร์ DLM ซึ่งเป็นตัว

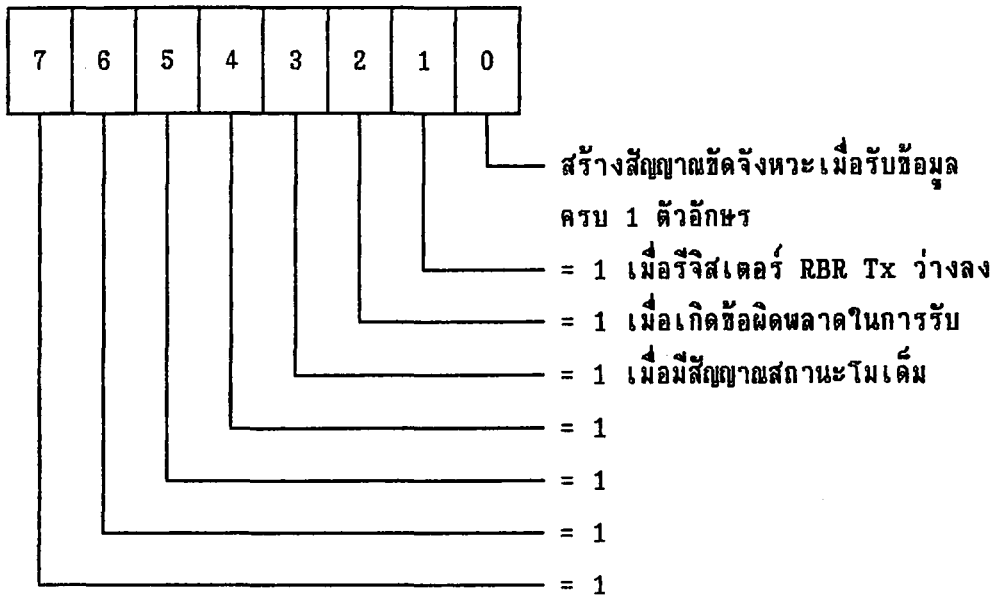
หารนัยสำคัญสูง ค่าของตัวหารนี้จะนำไปหารออกจากความถี่ 1.8432 MHz ซึ่งเป็นความถี่ของวงจรออสซิลเลเตอร์(Oscillator) ในอแดปเตอร์(Adaptor)สำหรับสื่อสาร ค่าที่ได้เป็น 16 เท่าของอัตราการรับส่งข้อมูล ค่าของตัวหารของความเร็วแสดงในตารางที่ 4.8 หรือจะหาจากสมการ

$$\text{ตัวหาร} = \frac{1843200}{16} = \frac{115200}{\text{อัตราการรับส่ง}}$$

อัตราการรับส่งข้อมูล	ตัวหาร(ฐาน 16)
300	180
600	0C0
1200	060
2400	030
4800	018
9600	00C

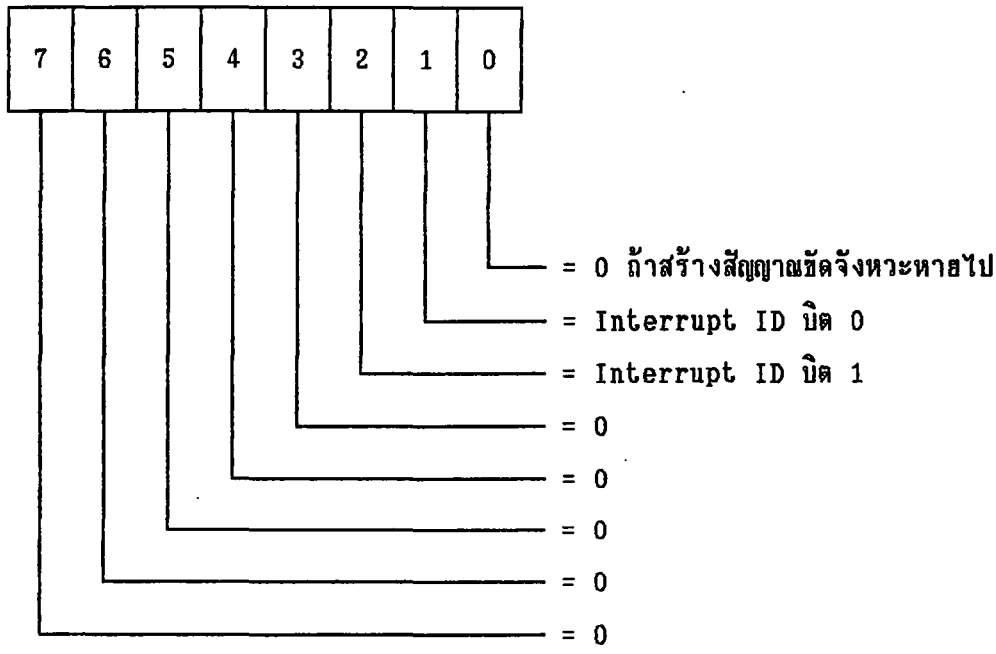
ตารางที่ 4.8 ตัวหาร 1.8432 MHz เพื่อได้ความถี่ 16 เท่าของอัตราการรับส่งข้อมูล (Baud rate) [11]

รีจิสเตอร์ DLM (IER) เป็นตัวหารนัยสำคัญสูงซึ่งใช้ร่วมกับรีจิสเตอร์ DLL ดังที่กล่าวมาแล้ว เช่นถ้าต้องการตั้งอัตราการรับส่งเป็น 2400 บิตต่อวินาที จะต้องใช้ตัวหาร 0030H ต้องตั้งค่าใน DLL เป็น 03H และ DLM เป็น 00H เป็นต้น หน้าที่อีกอย่างของรีจิสเตอร์ตัวนี้คือเป็นตัวควบคุมการเกิดการขัดจังหวะดังรูปที่ 4.7 ตัวอย่างเช่น ต้องการสร้างสัญญาณการขัดจังหวะเมื่อได้รับข้อมูลครบ 1 อักขระ โดยการเซต(Set) บิต 0 ของรีจิสเตอร์ IER ให้เป็น '1' เป็นต้น



รูปที่ 4.7 หน้าที่ของบิตต่างๆ ของรีจิสเตอร์ IER(Interrupt Enable Register) [11]

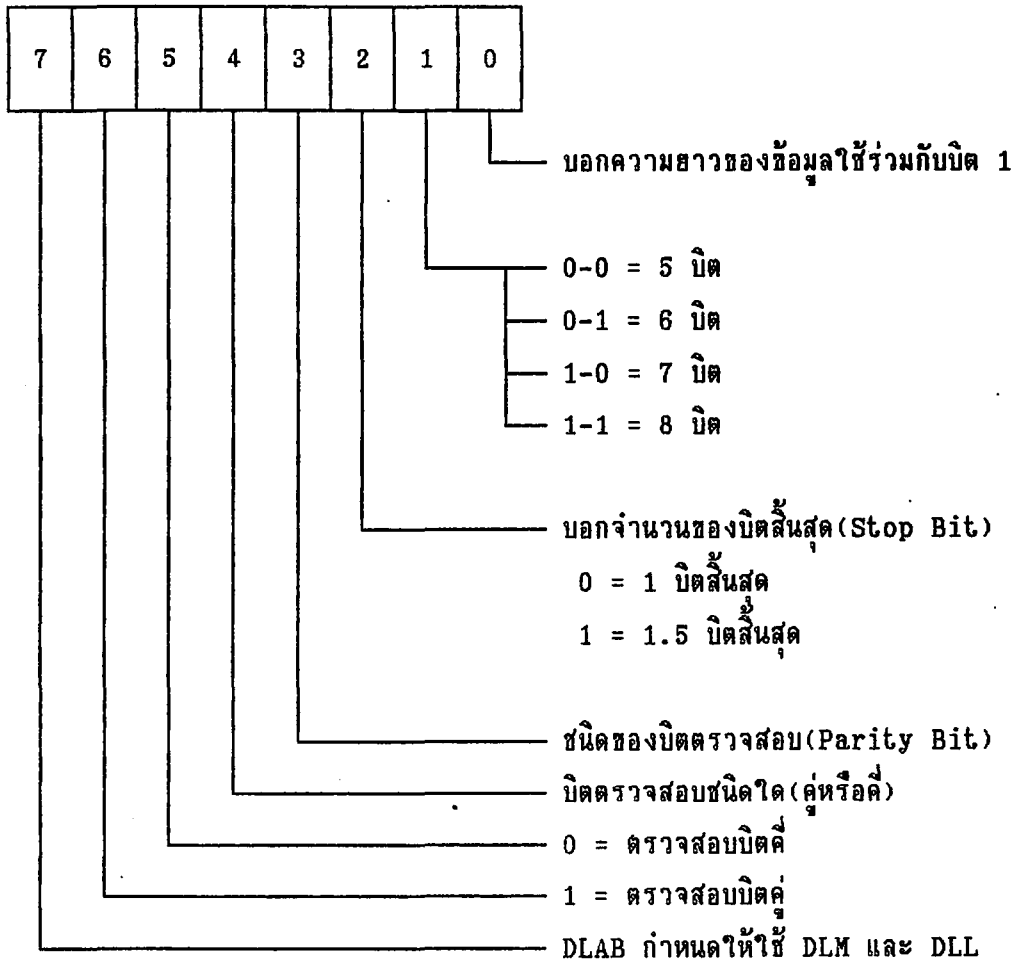
รีจิสเตอร์ IIR ใช้สำหรับบอกชนิดของการขัดจังหวะที่กำหนดโดยรีจิสเตอร์ IER ในโมดูลที่ทำหน้าที่จัดการเกี่ยวกับการขัดจังหวะควรตรวจสอบที่รีจิสเตอร์ตัวนี้ด้วย เพื่อจะได้รู้ว่าสาเหตุของการขัดจังหวะเพราะสาเหตุใด ดูจากรูปที่ 4.8 จะเห็นว่า ที่บิต 0 ของรีจิสเตอร์ IIR เป็นตัวแสดงสถานะว่า สัญญาณขัดจังหวะยังอยู่หรือหายไปแล้ว และบิตที่ 1 และ 2 เป็นตัวบอกสถานะของการขัดจังหวะ



รูปที่ 4.8 หน้าที่ของบิตต่างๆ ของรีจิสเตอร์ IIR
(Interrupt Identification Register) [11]

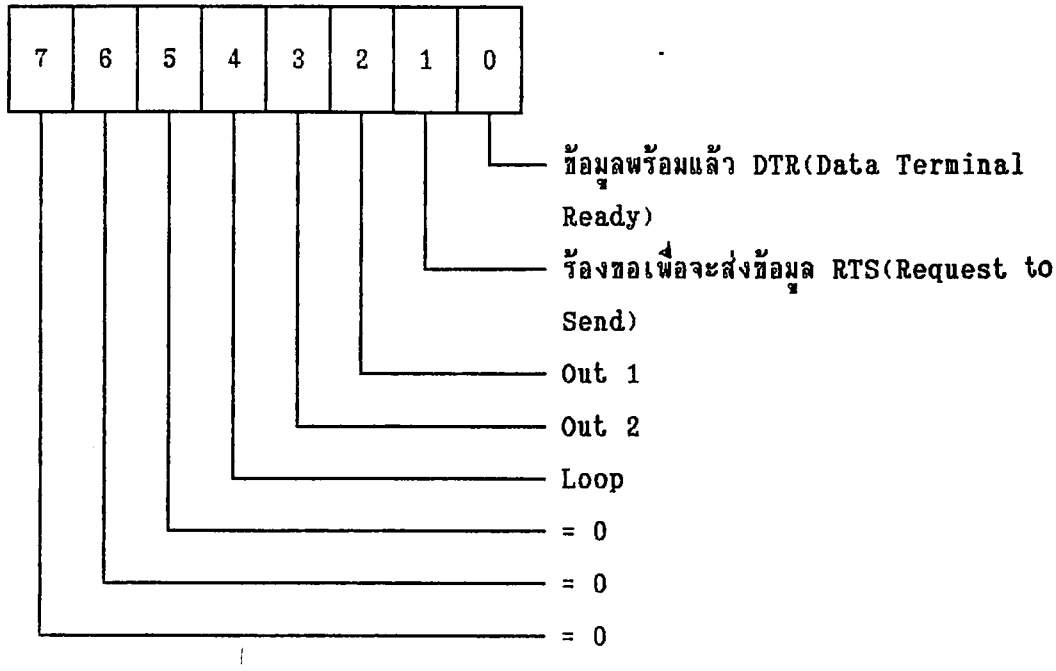
รีจิสเตอร์ LCR เป็นตัวกำหนดค่าพารามิเตอร์(Parameter) ต่างๆ ในการรับส่งข้อมูล ดูจากรูปที่ 4.9 สมมุติว่าต้องการรับส่งข้อมูลทีละ 8 บิต มีบิตสิ้นสุด 1 บิต ไม่มีบิตตรวจสอบ จะต้องให้ค่ากับรีจิสเตอร์ LCR คือ 00000011B บิต 7 ของ LCR เป็นตัวบอกว่าขณะนี้เป็นการกำหนดค่าพารามิเตอร์หรือบอกว่าเป็นการใช้งาน DLM และ DLL ถ้าเซตเป็น '0' ข้อมูลที่ส่งไปนั้นเป็นตัวกำหนดพารามิเตอร์ต่างๆ ถ้าเซตให้เป็น '1' ค่าที่ให้กับ DLM และ DLL ก็จะเป็นตัวหารนัยสำคัญสูงและต่ำนั่นเอง (ที่ต้องทำเช่นนี้เพราะรีจิสเตอร์ DLM กับ DLL มีหลายหน้าที่)

รีจิสเตอร์ MCR รีจิสเตอร์ตัวนี้ใช้สำหรับควบคุมการขัดจังหวะกับโมเด็ม รายละเอียดของแต่ละบิตดังรูปที่ 4.10 โดยมีบิตที่สำคัญคือ บิต 1,2 และ 3 (บิตที่ 3 หรือ OUT2 เป็นตัวอีนาเบิล(enable) สำหรับการขัดจังหวะให้กับไอซี 8259 ซึ่งเป็นไอซีที่ควบคุมการขัดจังหวะโดยเฉพาะซึ่งจะไม่กล่าวถึงรายละเอียดในที่นี้

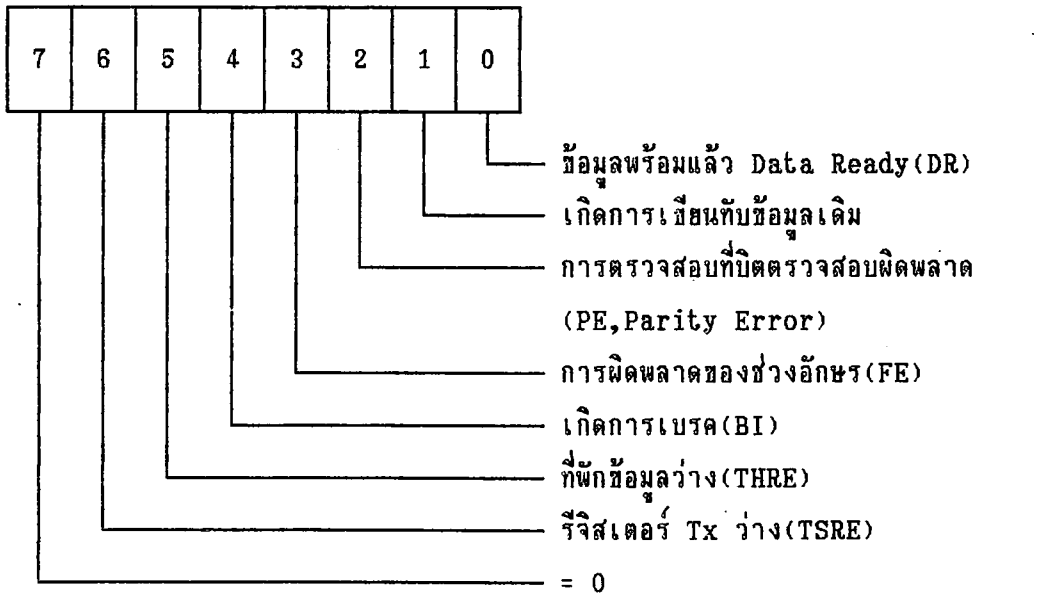


รูปที่ 4.9 หน้าที่ของบิตต่างๆ ของรีจิสเตอร์ LCR(Line Control Register) [11]

รีจิสเตอร์ LSR เป็นรีจิสเตอร์ที่ใช้บอกสถานะของการรับส่งข้อมูลถ้าต้องการรู้ว่าการรับส่งข้อมูลเป็นอย่างไรบ้าง มีข้อผิดพลาดอะไรเกิดขึ้น หรือตรวจสอบว่าที่พักข้อมูลในรีจิสเตอร์ RBR นั้นว่างหรือไม่ สามารถตรวจสอบได้ที่รีจิสเตอร์ตัวนี้ ดังรายละเอียดในรูปที่ 4.11

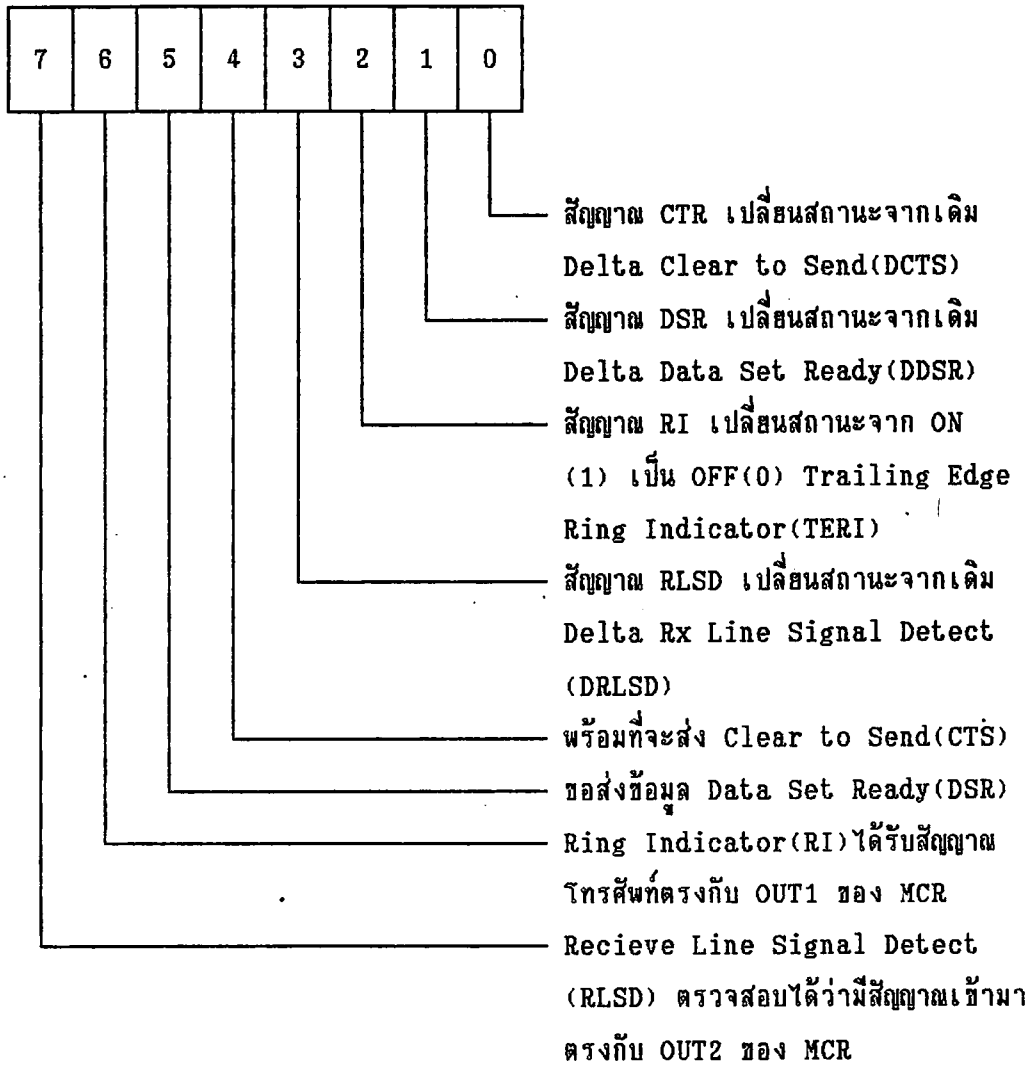


รูปที่ 4.10 หน้าที่ของบิตต่างๆ ของรีจิสเตอร์ MCR(Modem Control Register) [11]



รูปที่ 4.11 หน้าที่ของบิตต่างๆ ของรีจิสเตอร์ LSR(Line Status Register) [11]

รีจิสเตอร์ MSR เป็นตัวบอกสถานะต่างของโมเด็ม เราสามารถตรวจสอบสถานะของโมเด็มได้ที่รีจิสเตอร์ตัวนี้ รายละเอียดดังรูปที่ 4.12



รูปที่ 4.12 หน้าที่ของบิตต่างๆ ของรีจิสเตอร์ MSR(Modem Status Register) [11]

บทที่ 5
การพิมพ์ภาพกราฟฟิก

การแสดงผลนอกจากจะแสดงทางจอภาพแล้ว ยังสามารถคัดลอกภาพบนจอออกมายังกระดาษได้ หลักการลอกภาพก็คือใช้วิธีตรวจดูว่ามีการลงจุดหรือไม่(ถ้าไม่มีการลงจุดจะได้ค่าเป็น 0 แต่ถ้าหากไม่มีการลงจุดจะได้ค่าอื่นที่ไม่เท่ากับ 0)ซึ่งในภาษาปาสคาลมีคำสั่งเกี่ยวกับการทดสอบการลงจุดมาให้อยู่แล้วนั่นคือคำสั่ง

```
GetPixel(x,y : Integer)
```

คำสั่งนี้จะทำการตรวจคู่ลำดับบนจอภาพที่ x และ y แล้วทำการส่งค่าเป็นตัวเลขมาให้ถ้าหากไม่มีการลงจุดก็จะได้ค่าเป็น 0

5.1. การพิมพ์ของเครื่องพิมพ์

ก่อนที่จะเข้าสู่การอธิบายขอทำความเข้าใจไว้ก่อนว่า เครื่องพิมพ์ในที่นี้ หมายถึงเครื่องพิมพ์แบบหัวเข็ม 9 เข็ม ถ้าเป็นแบบ 24 หัวเข็มจะมีวิธีการที่ต่างกันออกไป

รหัสที่ใช้ในการควบคุมการทำงานของครื่องพิมพ์นั้นเราเรียกว่ารหัสเอสเคป(Escape code) ซึ่งที่รหัสที่สำคัญมีดังนี้

- ESC 3 ใช้สำหรับจัดระยะบรรทัดของเครื่องพิมพ์ให้ตรงกับความต้องการคำสั่งในภาษาปาสคาลมีดังนี้

```
Writeln(Lst,#27+'3'+chr(n));
```

เมื่อ n หมายถึง ตัวเลขที่เราต้องการให้ได้ระยะบรรทัด (n/216 นิ้ว สำหรับเครื่องพิมพ์แบบ 9 เข็ม n/180 นิ้ว สำหรับเครื่องพิมพ์แบบ 24 เข็ม)สำหรับการพิพ์แบบกราฟิกนี้เราจะต้องกำหนดค่า n = 24 เพื่อให้การพิมพ์มีลักษณะต่อเนื่องกัน

- ESC * สำหรับการกำหนดรูปแบบการพิมพ์ให้แก่เครื่องพิมพ์ โดยกำหนดเป็นรหัสขนาด 8 บิต คำสั่งที่ใช้คือ

```
Writeln(Lst,#27+'*'+chr(m)+chr(n1)+chr(n2));
```

โดยค่า m มีค่าจาก 0 ถึง 7 ส่วนค่า n1 และ n2 นั้นจะหาได้จากการคำนวณจากจำนวนจุดที่เราต้องการพิมพ์ลงเครื่องพิมพ์

- ESC l สำหรับกำหนดระยะกั้นซ้าย(left margin) โดยมีรูปแบบคำสั่งดังนี้

```
Writeln(Lst,#27+'l'+chr(n));
```

ค่า n จะขึ้นอยู่กับชนิดของเครื่องพิมพ์ว่าเป็นแบบแคร่สั้นหรือแคร่ยาว หากกำหนดเกินขอบเขตของเครื่องพิมพ์ ก็จะได้ค่ากั้นซ้ายสูงสุดที่เครื่องพิมพ์นั้นๆ จะทำได้

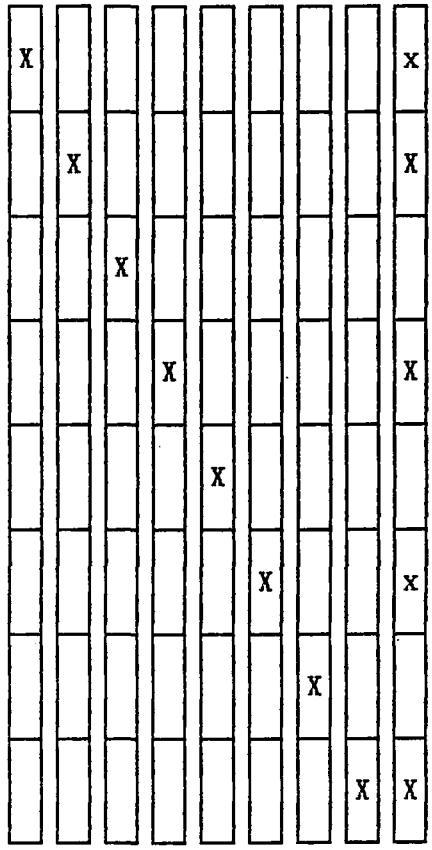
ในโครงงานนี้จะไม่ใช้คำสั่ง writeln(lst,'') ของปาสคาล แต่จะใช้วิธีการส่งค่าให้แก่อินเทอรัพท์(Interrupt) หมายเลข 17(ฐานสิบหก) ซึ่งควบคุมการพิมพ์โดยตรง เพราะจะช่วยให้เราสามารถควบคุมเครื่องพิมพ์ได้ดีกว่า โดยจะสร้างชุดคำสั่งเพื่อทำหน้าที่นี้ขึ้นมา ซึ่งการใช้งานอินเทอรัพท์นี้มีข้อกำหนดดังนี้

หมายเลขฟังก์ชัน(ในที่นี้ใช้ฟังก์ชัน 0) ใส่ค่าไว้ที่ รีจิสเตอร์ AH

รหัส 8 บิตจะส่งให้ กับ รีจิสเตอร์ AL ต้องส่งในรูปตัวเลขจำนวนเต็ม

หมายเลขเครื่องพิมพ์จะส่งให้กับ รีจิสเตอร์ DX ซึ่งตามปกติจะเป็นเลข 0 (หมายถึงเครื่องพิมพ์เครื่องที่หนึ่ง เครื่องอื่นก็ใช้หมายเลขต่อๆ กันไปแต่ตามปกติจะต่อเครื่องพิมพ์เพียงเครื่องเดียว)

ต่อจากนี้เรามาดูกันว่าค่าที่เราส่งให้แก่เครื่องพิมพ์นั้น สามารถทำงานได้อย่างไร ขอให้ดูรูปที่ 5.1



รูปที่ 5.1 แสดงค่าที่กำหนดให้แก่รีจิสเตอร์เพื่อใช้ควบคุมหัวเข็มแต่ละเล่ม [6]

ถ้าต้องการให้เข็มของเครื่องพิมพ์ทำงานตามแถว A จะต้องให้ค่าแก่เครื่องพิมพ์เป็น 128+64+16+4+1 = 213 ในทำนองเดียวกันถ้าเราต้องการพิมพ์แบบกลับขาวดำเราจะต้องให้ค่าเป็น 255-213 = 42 จะสังเกตมีแถวจากบนลงล่าง 8 แถวแต่ละแถวจะใช้แทน หัวเข็ม 1 หัว(หัวที่ 9 จะไม่ใช้งาน)

5.2. ศึกษาการสร้างภาพกราฟิก

สมมติว่าเราใช้จอภาพขนาด 750x350 จุด เราจะกำหนดรหัสที่จะส่งให้เครื่องพิมพ์อย่างไร

หลักการมีดังนี้

1. แถวทั้งหมดมี 350 แถว แบ่งออกเป็นบรรทัดโดยใช้บรรทัดละ 8 แถว(8 จุด) มีเศษเหลือไม่เป็นไร

2. ส่วนแบ่งแนวตั้งจากซ้ายไปขวามี 720 จุด เราจึงแบ่งได้เป็น 720 คอลัมน์

3. ตรวจการลงจุดเริ่มจากบรรทัดที่ 1 คอลัมน์ที่ 1 (ตรวจการลงจุด 8 จุด ตามแนวดิ่งที่มุมบนซ้าย) แล้วคำนวณเป็นรหัสดังที่ยกตัวอย่างในรูปที่ 5.1

4. ทำการตรวจการลงจุดคอลัมน์ที่ 2 เรื่อยไปจนครบ 720 คอลัมน์ จากนั้นจึงส่งข้อมูล 1 บรรทัดไปให้เครื่องพิมพ์ทำการพิมพ์(หมายความว่าแถวหนึ่งจะมีข้อมูล 720 ไบท์)

5. เมื่อหมดบรรทัดแรกแล้วก็ทำการตรวจรหัสบรรทัดต่อไป แล้วนำไปพิมพ์ ทำเช่นนี้จนกระทั่งครบทุกบรรทัดก็จะได้ภาพ 1 จอภาพตามต้องการ

5.3 การเลือกโหมด(mode)การพิมพ์

ในการเลือดโหมดการพิมพ์เราจะกำหนดโดยพิจารณาจากจำนวนจุดที่เราต้องการ โหมดการพิมพ์จะเป็นดังตารางที่ 5.1

จากตารางที่ 5.1 นั้นจะเห็นว่าหากต้องพิมพ์ภาพบนจอลงบนกระดาษกว้าง 8 นิ้ว แล้วให้มีลักษณะที่เหมือนกับภาพบนจอจะต้องเข้าใจการทำงานด้วย เช่น เราแสดงภาพบนจอขนาด 640x350 จุดจะพิจารณาได้ดังนี้

1. จำนวนจุดทั้งหมดในหนึ่งแถวมี 640 จุด

2. ความกว้างของกระดาษ 8 นิ้ว

3. จำนวนจุดที่เหมาะสมหาได้จาก จำนวนจุดบนจอภาพ/ความกว้าง กระดาษ นำตัวเลขที่ได้ในข้อ 1 กับ 2 มาคำนวณได้ดังนี้

จำนวนจุดต่อนิ้ว = 640/8 = 80 จุดต่อนิ้ว

4. นำค่าที่คำนวณได้ไปพิจารณาโหมดการพิมพ์ได้เป็นโหมด 4

รหัส	โหมด	จำนวนจุดต่อนิ้ว
0	Normal density	60
1	Dual density	120
2	Double-speed, Dual density	120
3	Quadruple-density	240
4	CRT graphics	80
5	Plotter graphics	72
6	CRT graphics II	90
7	Dual density plotter graphics	144

ตารางที่ 5.1 แสดงโหมดต่างๆ ของการพิมพ์แบบกราฟิก [6]

บทที่ 6
การวิจัยและดำเนินงาน

ในการดำเนินงานขั้นแรก เริ่มจากทำการศึกษาหลักการการทำงานและวิธีการใช้เครื่องวิเคราะห์ความแตกต่างทางความร้อน จากเอกสารรายงานของผู้จัดทำโครงงานในปีการศึกษา 2535 ได้มีข้อเสนอแนะอยู่ในบทสรุปผลการทดลองไว้ดังนี้

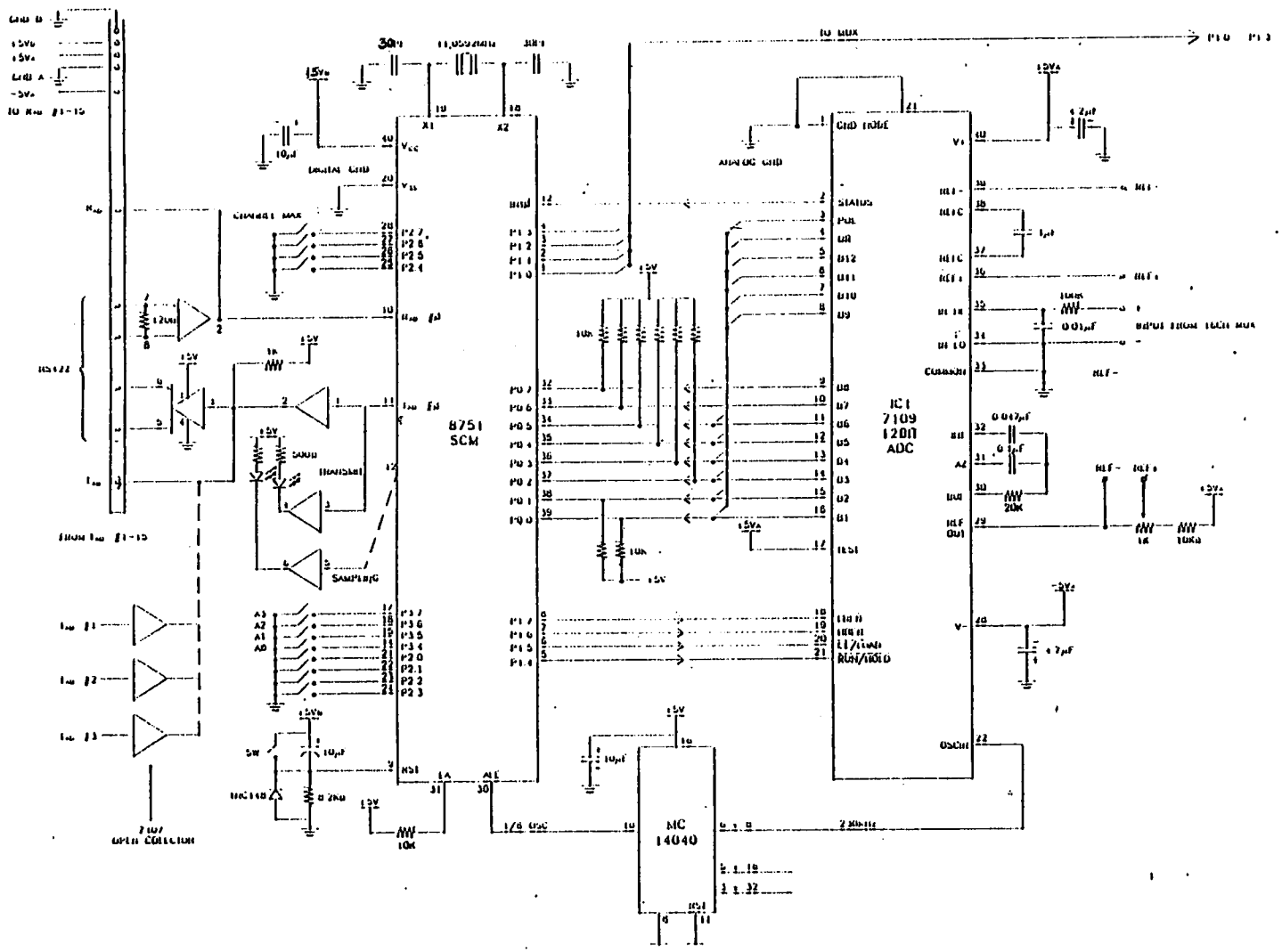
จากการทดลองหาค่าคงที่ของเครื่องมือ(K) แล้วนำค่าที่ได้ไปหามวลของสารที่ใช้ทดสอบปรากฏว่ามีค่าผิดพลาดจากค่าที่ทราบไปมาก ซึ่งมีข้อสันนิษฐานว่าความคาดเคลื่อนนี้มาจากการประมาณค่าพื้นที่ใต้พีค(peak) ด้วยพื้นที่ของสามเหลี่ยม(ดังที่จะได้อธิบายต่อไป) ทำให้ค่าที่คำนวณได้คลาดเคลื่อนไป จึงได้ทำการเสนอแนะวิธีการปรับปรุงความถูกต้องในการวัดของเครื่องไว้โดยให้นำสัญญาณที่ได้จากเครื่องมือนี้ เชื่อมต่อเข้ากับคอมพิวเตอร์แล้วใช้คอมพิวเตอร์ทำการประมาณค่าพื้นที่ใต้พีคซึ่งจะได้ค่าที่ละเอียดใกล้เคียงกับค่าพื้นที่จริง

จากข้อเสนอแนะดังกล่าว จึงได้ทำการสร้างวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลขึ้นมาเพื่อใช้สำหรับเชื่อมต่อสัญญาณจาก วงจรขยายสัญญาณของหัววัดอุณหภูมิเข้ากับเครื่อง-คอมพิวเตอร์ โดยผ่านไอซี MAX232 เพื่อเปลี่ยนจากสัญญาณทีทีแอล(TTL) เป็นสัญญาณแบบ RS-232 ซึ่งงานส่วนนี้เป็นงานทางด้านฮาร์ดแวร์(Hardware) วงจรที่ทำหน้าที่ดังกล่าวแสดงไว้ในรูปที่ 6.1

ในส่วนของซอฟท์แวร์(Software) ใช้สำหรับเก็บข้อมูลที่วงจรแปลงสัญญาณส่งมาให้แล้วจะนำไปแสดงผลในรูปของกราฟ(ลักษณะของข้อมูลที่เข้ามาจะเป็นคู่อันดับแกน X กับ Y) และจัดเก็บข้อมูลลงในจานแม่เหล็ก นอกจากนี้ยังใช้สำหรับวิเคราะห์ข้อมูลซึ่งจะแสดงผลข้อมูลออกมาได้สองทางคือทางจอภาพกับทางเครื่องพิมพ์

6.1. การประมาณค่าพื้นที่ใต้พีค

ในหัวข้อนี้จะทำการอธิบายวิธีการประมาณค่าพื้นที่ใต้พีคสองวิธีคือ วิธีประมาณค่าพื้นที่ใต้พีคด้วยพื้นที่ของสามเหลี่ยมซึ่งเป็นวิธีการประมาณค่าพื้นที่อย่างง่ายๆ แต่ค่าที่ได้จะไม่ใกล้เคียงกับค่าพื้นที่จริงกับ วิธีการประมาณค่าวิธีสี่เหลี่ยมคางหมูซึ่งเป็นวิธีที่ให้ค่าได้ ใกล้-



รูปที่ 6.1 วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล 12 บิต 16 ช่องสัญญาณ

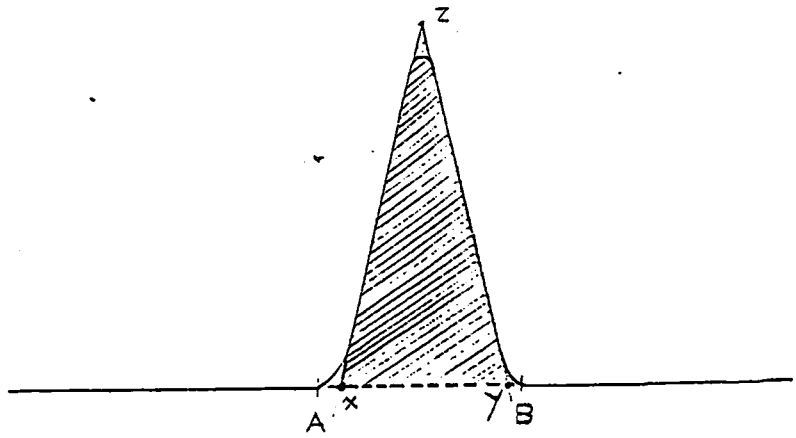
เคียงกับค่าพื้นที่จริงมากกว่าแต่การประมาณค่าด้วยวิธีนี้ จะต้องใช้คอมพิวเตอร์เข้ามาช่วยในการประมาณค่าจึงจะทำได้

6.1.1. การประมาณค่าพื้นที่ใต้พีคด้วยพื้นที่สามเหลี่ยม

พิจารณารูปที่ 6.2 ส่วนต่างๆ ของกราฟเป็นดังต่อไปนี้

- จุด A คือ จุดเริ่มต้นของปฏิกิริยา
- จุด B คือ จุดสิ้นสุดของปฏิกิริยา
- เส้นตรง AB คือ เส้นฐาน
- สามเหลี่ยม XYZ คือพื้นที่สามเหลี่ยมที่ใช้สำหรับประมาณค่าพื้นที่ใต้พีค

จากรูปจะเห็นว่าพื้นที่ใต้พีคจะไม่พอดีกับพื้นที่สามเหลี่ยมทำให้เกิดความคลาดเคลื่อนในการประมาณค่า



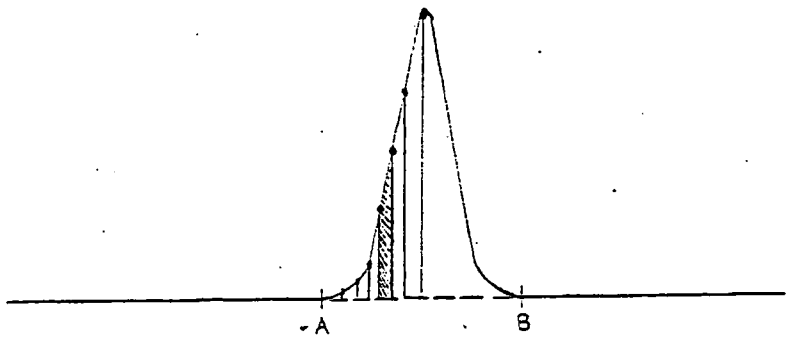
รูปที่ 6.2 แสดงการประมาณค่าพื้นที่ใต้พีคโดยการประมาณด้วยพื้นที่สามเหลี่ยม

6.1.2. การประมาณค่าพื้นที่ด้วยวิธีสี่เหลี่ยมคางหมู

จากรูปที่ 6.3 ส่วนประกอบของกราฟเป็นดังต่อไปนี้

- เส้นตรง AB คือเส้นฐาน
- จุด A คือจุดเริ่มต้นของปฏิกิริยา
- จุด B คือจุดสิ้นสุดของปฏิกิริยา
- พื้นที่ที่แรเงา คือ พื้นที่สี่เหลี่ยมคางหมูส่วนหนึ่งที่ใช้ในการประมาณค่าพื้นที่ใต้พีค

เมื่อดูจากภาพจะเห็นว่าถ้าเราทำการแบ่งพื้นที่ใต้พีคเป็นส่วนย่อยหลายๆ ส่วนแล้วแทนที่พื้นที่ส่วนย่อยๆ เหล่านั้นด้วยพื้นที่สี่เหลี่ยมคางหมู เมื่อนำพื้นที่ของสี่เหลี่ยมคางหมูในแต่ละส่วนมารวมกันจะได้ค่าที่ใกล้เคียงกับพื้นที่ใต้พีค ยิ่งทำการแบ่งพื้นที่ให้เป็นส่วนละเอียดเท่าใดพื้นที่ที่ได้ก็จะยิ่งใกล้เคียงกับค่าพื้นที่จริงเท่านั้น

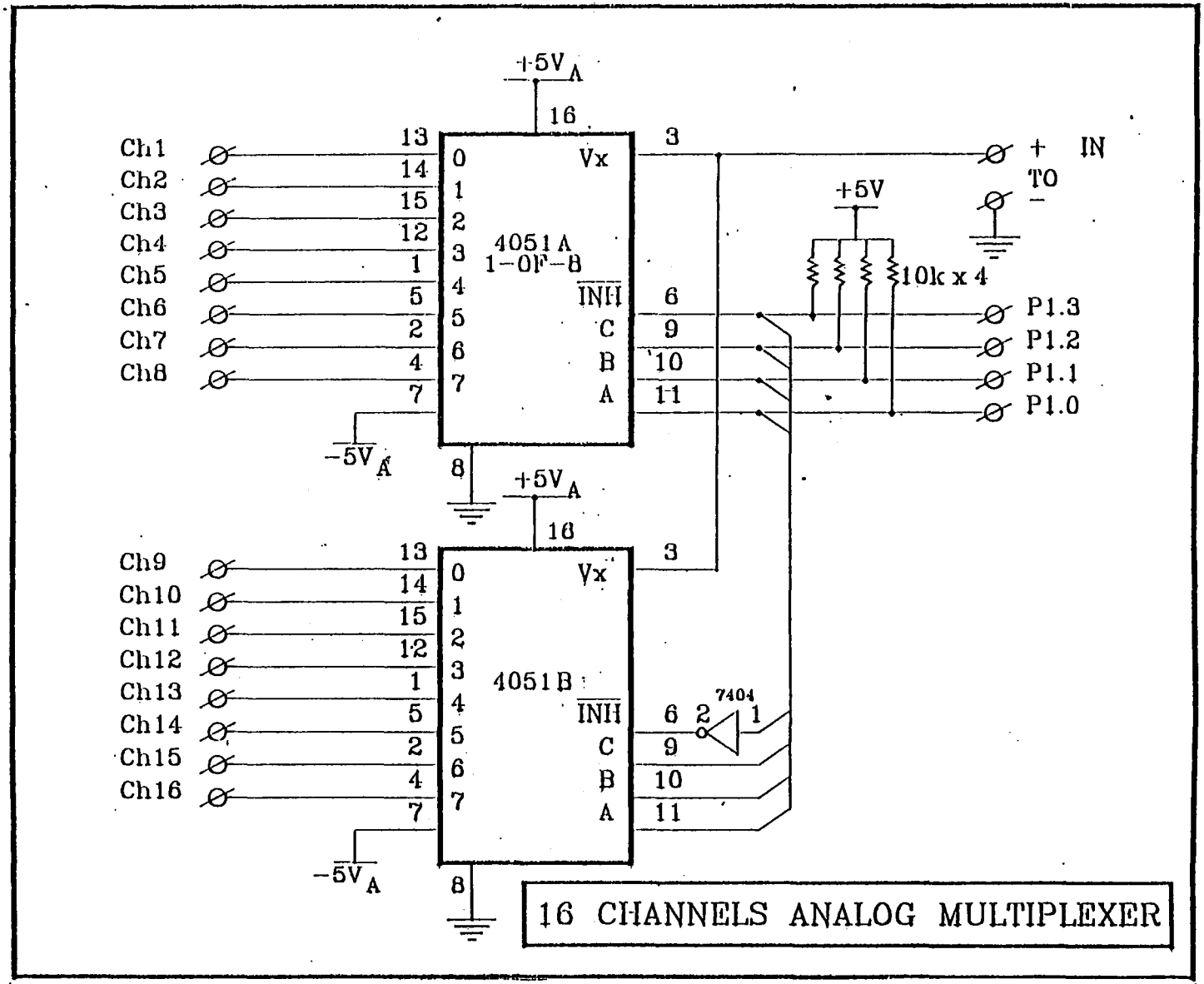


รูปที่ 6.3 แสดงการประมาณค่าพื้นที่ใต้พีคด้วยวิธีสี่เหลี่ยมคางหมู

6.2. ส่วนที่ทำหน้าที่แปลงสัญญาณอนาลอกเป็นดิจิตอล

จากรูปที่ 6.1 วงจรจะมีส่วนประกอบต่างๆ ดังต่อไปนี้

1. ส่วนที่ทำหน้าที่มัลติเพลกซ์(Multiplex) สัญญาณจากวงจรขยายสัญญาณที่ได้มาจากหัววัดอุณหภูมิ เนื่องจากสัญญาณที่เราต้องการมีลักษณะเป็นคู่อันดับ เพื่อใช้สำหรับแสดงผลในรูปของกราฟ ดังนั้นจึงต้องส่งสัญญาณเข้ามาคราวละ 2 ครั้ง โดยจะส่งสัญญาณจากแกนใดแกนหนึ่งเข้ามาก่อนแล้วจึงส่งสัญญาณจากอีกแกนตามมา วงจรที่ทำหน้าที่ในส่วนนี้คือ



รูปที่ 6.4 วงจรมัลติเพลกซ์ขนาด 16 ช่องสัญญาณ

ไอซี 4051 จำนวน 2 ตัว สามารถมัลติเพลกซ์สัญญาณได้ 16 ช่องสัญญาณ แต่ในที่นี้จะใช้เพียง 2 ช่องสัญญาณ คือสัญญาณความแตกต่างระหว่างอุณหภูมิสารอ้างอิงกับสารตัวอย่าง (แกนตั้ง) และสัญญาณอุณหภูมิอ้างอิง(แกนนอน) สัญญาณที่ผ่านวงจรมัลติเพลกซ์ไปแล้วนั้นจะถูกส่งต่อไปยัง วงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล(รูปที่ 6.4)

2. ไอซี ICL7109 เป็นวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลชนิดสโลปคู่ มีความละเอียดขนาด 12 บิต กำหนดให้มีแรงดันอ้างอิง บวกลบ 400 มิลลิโวลต์ วงจรนี้จะทำหน้าที่แปลงสัญญาณอนาลอกจากวงจรขยายสัญญาณ และจะทำการส่งค่าแปลงแล้วไปให้กับไมโครโปรเซสเซอร์ 8751 อีกที

3. ไมโครคอนโทรลเลอร์(Microcontroller) 8751 จะรับสัญญาณมาจากไอซี ICL7109 แล้วจะจัดการข้อมูลที่ได้ให้อยู่ในรูปของอักขระเพื่อที่จะทำการส่งข้อมูลนี้ไปให้กับเครื่องคอมพิวเตอร์ นอกจากนี้ในการเซต(set) วงจรจะต้องทำการเซตที่ขาต่างๆ ของ 8751 ดังที่แสดงไว้ในตารางที่ 6.1 ถึง 6.5

P2.7	P2.6	P2.5	P2.4	SCAN
0	0	0	0	CH1
0	0	0	1	CH1-CH2
0	0	1	0	CH1-CH3
0	0	1	1	CH1-CH4
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
1	1	1	1	CH1-CH16

ตารางที่ 6.1 การกำหนดจำนวนช่องสัญญาณโดยกำหนดที่ขา P2.4 ถึง P2.7 ของ 8751

P3.7	P3.6	P3.5	P3.4	บอร์ด
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
1	1	1	1	F

ตารางที่ 6.2 กำหนดหมายเลขบอร์ด(Board) ที่ขา P3.4 ถึง P3.7 ของ 8751ในโครงงานนี้เลือกหมายเลข 0

P2.0	อัตราการรับส่งข้อมูล
0	9600
1	19200

ตารางที่ 6.3 อัตราการรับส่งข้อมูลกำหนดที่ขา P2.0 ของ 8751 ในที่นี้เลือก 9600 บิตต่อวินาที

P2.1	วิธีการรับส่งข้อมูล
0	ส่งด้วยความเร็วสูงสุด โดยไม่รอสัญญาณกระตุ้น
1	รอสัญญาณกระตุ้นก่อนจึงค่อยส่งข้อมูลไป 1 ไบท์

ตารางที่ 6.4 เลือกวิธีการรับส่งข้อมูล ในโครงงานนี้เลือก 1

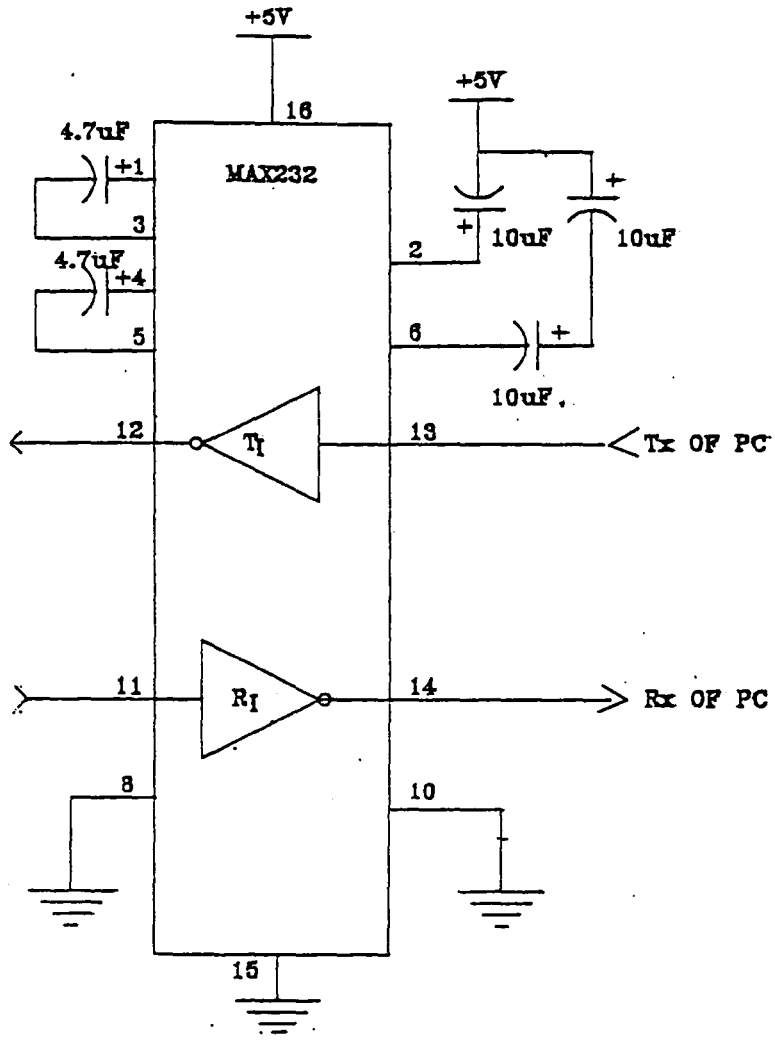
P2.3	ลักษณะของสัญญาณที่ทำการส่ง
0	ส่งสัญญาณในแบบไบโพลา(Bipolar)
1	ส่งสัญญาณในแบบยูนิโพลา(Unipolar)

ตารางที่ 6.5 เลือกลักษณะของสัญญาณที่ส่งออกไป(เลือก 0)

4. ไอซี MAX232 ใช้สำหรับการแปลงสัญญาณแบบทีทีแอล(TTL ลักษณะของสัญญาณจะเป็นคลื่นสี่เหลี่ยม ที่สถานะต่ำจะเป็น 0 โวลต์ สถานะสูงจะเป็น +5 โวลต์) ไปเป็นสัญญาณแบบ RS-232(ลักษณะของสัญญาณได้อธิบายไว้แล้วในบทที่ 4) เพื่อเชื่อมต่อสัญญาณเข้ากับพอร์ตอนุกรมของคอมพิวเตอร์

5. แหล่งจ่ายแรงดันของวงจรส่วนนี้จะเป็น แรงดันด้านบวกและลบ 5 โวลต์ สำหรับสัญญาณอนาลอก แรงดัน +5 โวลต์ สำหรับสัญญาณดิจิตอล ส่วนกราวน์จะใช้ร่วมกัน

การต่อสัญญาณจากหัววัดอุณหภูมิเข้าสู่วงจรแปลงสัญญาณนั้นจำเป็นต้องทำการเปลี่ยนแปลงอัตราการขยายให้ เพื่อให้ขนาดของสัญญาณสอดคล้องกับแรงดันอ้างอิงของวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล



รูปที่ 6.5 ไอซี MAX232

6.3. ด้านซอฟท์แวร์

เราแบ่งซอฟท์แวร์ที่ทำออกเป็นส่วนๆ ตามลักษณะการทำงานได้ดังนี้

1. **ซอฟท์แวร์ส่วนรับข้อมูล** เป็นซอฟท์แวร์สำหรับติดต่อกับพอร์ตอนุกรม โดยจะทำการส่งสัญญาณกระตุ้นไปบอกให้ 8751 ส่งข้อมูลเข้ามาครั้งละ 1 ไบท์ ซึ่งอยู่ในรูปของรหัสแอสกี(ASCII) แล้วทำการกระตุ้นและรับข้อมูลอีกจนกว่าจะได้ข้อมูลครบ เมื่อได้ข้อมูลครบแล้วจึงทำการเปลี่ยนรูปแบบข้อมูลจากข้อมูลตัวอักษรเป็นตัวเลขจำนวนเต็มแล้วจึงนำตัวเลขที่ได้ไปทำการคำนวณเป็นค่าอุณหภูมิกับความแตกต่างของอุณหภูมิอีกที

เพื่อความเข้าใจขอยกตัวอย่างประกอบดังนี้

ก. คอมพิวเตอร์ส่งสัญญาณไปกระตุ้นให้ 8751 ส่งสัญญาณที่ผ่านการแปลงมาแล้วกลับไปให้คอมพิวเตอร์จนกระทั่งได้ข้อมูลครบ สมมติว่าได้ข้อมูลมาดังในเครื่องหมายคำพูดนี้ "+4095,+0000" ข้อมูลเหล่านี้เป็นชนิดสายอักษร(String) ข้อมูลที่อยู่หน้าเครื่องหมายจุลภาคจะเป็นแกนอุณหภูมิอ้างอิง ส่วนที่อยู่หลังเครื่องหมายจุลภาคจะเป็นข้อมูลของความแตกต่างทางอุณหภูมิ

ข. ทำการแยกข้อมูลออกเป็นสองส่วนคือส่วนอุณหภูมิอ้างอิงกับส่วนความแตกต่างทางอุณหภูมิเราจะได้ข้อมูลเป็น "+4095" กับ "+0000" ขณะนี้ข้อมูลยังเป็นชนิดสายอักษรอยู่ซึ่งใช้ในการคำนวณไม่ได้ จึงต้องทำการเปลี่ยนชนิดของข้อมูลให้เป็นตัวเลขก่อน เมื่อทำการเปลี่ยนชนิดของข้อมูลแล้วเราจะได้ข้อมูลที่เป็นเลขจำนวนเต็มมา 2 จำนวน

ค. นำเลขจำนวนเต็มทั้งสองจำนวน ไปทำการคำนวณ ให้เป็นค่าอุณหภูมิจะได้อุณหภูมิอ้างอิงเท่ากับ 400 องศาเซลเซียส และ ความแตกต่างทางอุณหภูมิเป็น 0 องศาเซลเซียส ข้อมูลนี้จะถูกเก็บลงในจานแม่เหล็กและแสดงผลทางจอภาพ

2. **ส่วนการแสดงผลทางจอภาพ** จะนำข้อมูลซึ่งเป็นคู่อันดับที่ได้จากส่วนที่ 1 หรือที่อ่านจากจานแม่เหล็ก มาแสดงเป็นกราฟทางจอภาพ

3. **ส่วนวิเคราะห์ผลข้อมูล** จะนำข้อมูลที่ได้มาคำนวณหาพื้นที่ใต้พีคตามวิธีการที่ได้กล่าวไว้ตอนต้นบท แล้วนำค่าพื้นที่ที่ได้มาคำนวณหาเอนทาลปีของสารอีกที หรืออีกทางหนึ่งนำค่าพื้นที่คำนวณได้กับค่าเอนทาลปีมาตราฐานมาคำนวณหาค่าคงที่ของเครื่อง

4. **ส่วนแสดงผลข้อมูลทางเครื่องพิมพ์** จะทำงานตามที่ได้กล่าวไว้ในบทที่ 5 คือทำการลอกจอภาพตามแนวนอนส่งไปให้เครื่องพิมพ์ทำการพิมพ์หน้าจอจากบนลงล่างจนกระทั่งได้ภาพสมบูรณ์

5. **ส่วนรับค่าจากแป้นพิมพ์** ใช้ในการอ่านค่าพารามิเตอร์(parameter) เช่น

ชื่อของสารทดสอบ น้ำหนักของสาร ชื่อไฟล์ เป็นต้น โปรแกรมในส่วนนี้ที่เขียนขึ้นมาแบ่งตามชนิดข้อมูลของตัวแปรที่รับค่าได้สามชนิดคือ

- ตัวแปรของข้อมูลชนิดสายอักษร เช่น ชื่อของสารทดสอบ
- ตัวแปรของข้อมูลชนิดจำนวนจริง เช่น น้ำหนักของสาร
- ตัวแปรของข้อมูลชนิดจำนวนเต็ม ซึ่งไม่ได้ใช้ในโปรแกรมนี้แต่ได้เขียนเอาไว้

บทที่ 7
การทดสอบเครื่องมือที่สร้างขึ้น

ในบทนี้จะกล่าวถึงการทดสอบเครื่องมือที่ได้สร้างเพิ่มเติมขึ้นมา โดยจะแบ่งการทดลอง ออกเป็น 2 ตอนดังนี้

1. การทดลองการรับข้อมูล เพื่อทดสอบการทำงานของวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลและทดสอบโปรแกรมการรับส่งข้อมูล

2. การทดลองการทำงานของโปรแกรมวิเคราะห์ข้อมูล เป็นการทดสอบทางด้านซอฟท์แวร์โดยเฉพาะ โดยทดลองให้โปรแกรมทำการวิเคราะห์ข้อมูลที่ได้จำลองขึ้นมาแล้วดูผลที่ได้ว่าถูกต้องหรือไม่

วิธีการทดลอง

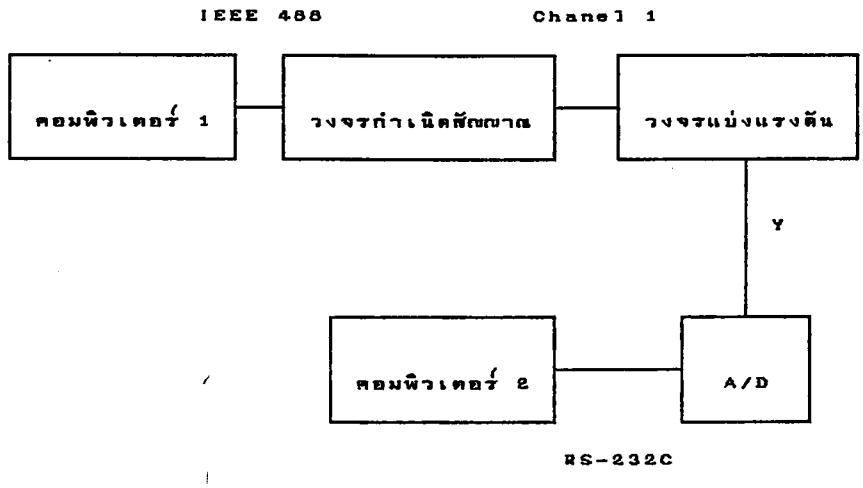
ตอนที่ 1 ทดลองการรับข้อมูล

ในการทดลองนี้จะใช้คอมพิวเตอร์ 2 เครื่อง โดยให้เครื่องหนึ่ง(คอมพิวเตอร์ 1) ทำหน้าที่ควบคุมวงจรกำเนิดสัญญาณเพื่อจำลองสัญญาณอนาลอก ลักษณะของสัญญาณที่ได้จะเป็นสัญญาณรูปสามเหลี่ยม ดังรูปที่ 7.2 มีค่าแรงดันสูงสุดที่ +10 โวลต์ ต่ำสุดที่ -10 โวลต์

สำหรับคอมพิวเตอร์อีกเครื่อง(คอมพิวเตอร์ 2) จะทำหน้าที่รับข้อมูลซึ่งออกมาจากวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล มาวาดกราฟ และทำการวิเคราะห์

ขั้นตอนในการทดลอง

1.1 เชื่อมต่อสัญญาณจากวงจรกำเนิดสัญญาณเพื่อส่งข้อมูลไปยังคอมพิวเตอร์ 2 ดังรูปที่ 7.1 จากรูปจะเห็นว่าต้องเพิ่มวงจรแบ่งแรงดันเข้ามาเนื่องจาก สัญญาณจากวงจรกำเนิดสัญญาณมีขนาดใหญ่กว่าแรงดันอ้างอิงของวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (แรงดันอ้างอิงอยู่ที่ +400 มิลลิโวลต์ ถึง -400 มิลลิโวลต์) ดังนั้นจึงต้องทำการแบ่งแรงดันของสัญญาณให้อยู่ในช่วงของแรงดันอ้างอิง



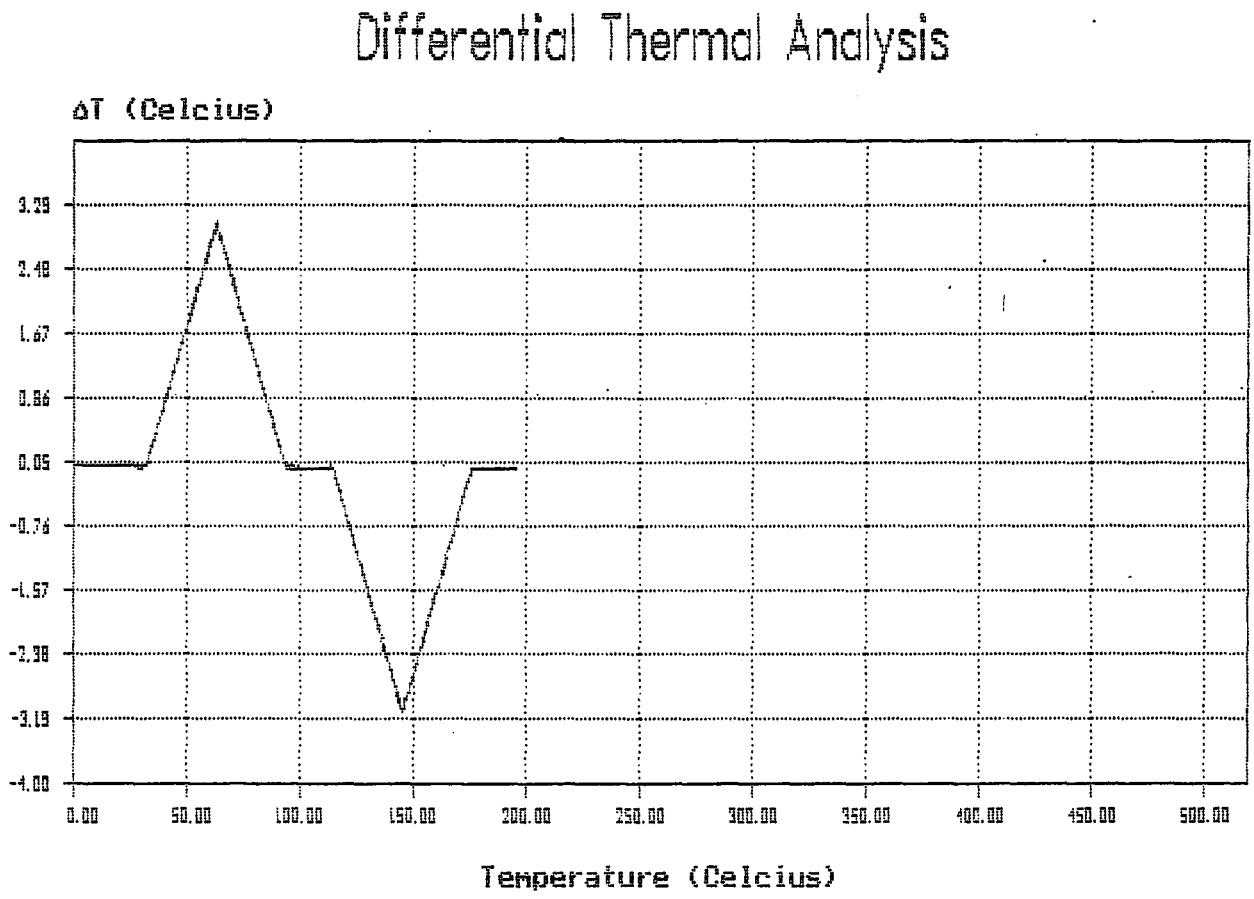
รูปที่ 7.1 แสดงการเชื่อมต่อสัญญาณเพื่อทดลองรับข้อมูล

1.2 จากรูปที่ 7.1 ข้อมูลที่ต่อเข้ากับคอมพิวเตอร์ 2 จะมีเฉพาะแกน Y เท่านั้น (จะมีสัญญาณเปลี่ยนแปลงเฉพาะแนวแกน Y) ดังนั้นจึงให้โปรแกรมสำหรับทดลองทำการให้ค่ากับแกน X โดยจะเพิ่มค่าทางแกน X ในลักษณะเชิงเส้น(ค่าที่เพิ่มทางแกน X ต่อเวลาจะคงที่) ซึ่งผลที่ได้จะเป็นดังรูปที่ 7.2

ตอนที่ 2 ทดลองวิเคราะห์ข้อมูล

การทดลองในตอนที่ 2 จะให้คอมพิวเตอร์ทำการวิเคราะห์ข้อมูล โดยให้ทำการหาจุดยอดของพีคตามแนวแกน X (อุณหภูมิที่เกิดปฏิกิริยา) คำนวณพื้นที่ใต้พีค และคำนวณค่าของพลังงานที่ใช้ในการทำปฏิกิริยา(Enthalpy,ΔH) โดยสมมติให้ค่าคงที่ของเครื่องวิเคราะห์ความแตกต่างทางความร้อน(K) มีค่าเท่ากับ 1 โดยข้อมูลที่ใช้ในการทดสอบการวิเคราะห์จะใช้ข้อมูลจากที่ได้ทดลองในตอนที่ 1(รูปที่ 7.2) ซึ่งผลที่ได้จากการวิเคราะห์บอกให้เราทราบว่าการตรวจจับพีคมีข้อผิดพลาด เนื่องจากได้จำนวนพีคไม่เท่ากับที่เห็นในรูป ได้พื้นที่ใต้

พีคไม่เท่ากับที่คำนวณได้ด้วยมือและนอกจากนี้ค่าเริ่มต้น ค่าของอุณหภูมิที่เกิดปฏิกิริยาและ ค่าสิ้นสุดของปฏิกิริยาได้ค่าไม่ตรงกับค่าวัดได้จากกราฟ ค่าที่ได้จากการวิเคราะห์นี้ขึ้นอยู่กับการกำหนดค่าพารามิเตอร์ ในส่วนของการวิเคราะห์(ในโปรแกรมหลัก)



รูปที่ 7.2 ผลที่ได้จากการทดลองตอนที่ 1

สรุปผลการทดลอง

จากผลการทดลองสามารถสรุปได้ดังนี้

1. วิธีการวิเคราะห์พีคมีความผิดพลาดจำเป็นต้องหาวิธีการวิเคราะห์ใหม่แล้วจึงนำ วจึงนำไปทดสอบกับข้อมูลที่ได้จากเครื่องวิเคราะห์ความแตกต่างทางความร้อน เพื่อเปรียบเทียบกับค่ามาตรฐาน

2. เปลี่ยนการแสดงผลจากที่ใช้ในขณะนี้คือ ใช้โหมดตัวอักษร(Text mode) ผสมกับโหมดกราฟิก(Graphic mode) ไปเป็นโหมดกราฟิกล้วนๆ เนื่องจากมีความหลากหลายในการแสดงผลและไม่ต้องเปลี่ยนโหมดการแสดงผลไปมา

เอกสารอ้างอิง

1. Douglas A. Skoog, Principles of Intrumental analysis, 3rd Ed., Colt-Sannders International Editions, 1985.

2. Phoenix Technologies Ltd., System BIOS for IBM PC/XT/AT computer and compatibles, Addison-Wesley Publishing Company,Inc., 1989.

3. Stephen K. O' Brien, Turbo pascal 6 The complete reference, BORLAND OSBORNE.

4. Robert F. Coughlin, Frederick F. Driscoll, Operational Amplifiers & Linear Integrated Circuits, 4th Ed., Prentice-Hall International, Inc.,1991.

5. พุฒิพงษ์ กนกบรรฒากร, สุรชัย สายพิมพ์ "การสร้างเครื่องดิฟเฟอเรนเชียลเทอร์มอลอนาไลซิส" วิทยานิพนธ์ระดับปริญญาตรี ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

6. อาจหาญ สัตยารักษ์, เทคนิคการเขียนกราฟิกบนเทอร์โบปาสคาล, บริษัท ซีเอ็ดยูเคชั่นจำกัด (มหาชน).

7. สุรศักดิ์ สงวนพงษ์, เทคนิคการเขียนโปรแกรมขั้นสูง แอดวานซ์เทอโบปาสคาล Version 4, บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน).

8. บุญเลิศ เอี่ยมทัศนา, คู่มือเทอร์โบปาสคาลรุ่น 4.0-5.0, บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน).

9. นุกูล กระจาย, การเขียนโปรแกรมกราฟิกและเกมคอมพิวเตอร์ด้วยเทอร์โบปาสคาล, บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน).

10. มงคล อัศวโกวิทกรณ์, การเขียนโปรแกรมกราฟิก, บริษัท ดวงกมลสมัย จำกัด, 2534.

11. นาวิน แดนราช, คชเดช แพงศุข, ภูเบศ เสถียรพจน์, นิทัศน์ กำแพงพันธ์ "การเขียนออนไลน์เกม" คอมพิวเตอร์รีวิว 89(ม.ค. 2535):181-190.

12. เปรมจิตร วิสุทธิศิริ "พื้นฐานวงจร เอทูดี,ดีทูเอ ตอน 2 วงจรแปลงอนาลอกเป็นดิจิตอล" เซมิคอนดัคเตอร์ อิเลคทรอนิกส์ 103(ธ.ค. 2533):302-305.

13. นต.ดร.ไพศาล สงวนหมู่, รส.ยืน ภู่วรวรรณ, การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวอร์ค ,บริษัท ซีเอ็ดยูเคชัน จำกัด (มหาชน), 2532.

14. เอกชัย สันกำแพง "วงจรแปลง RS-232 เป็น RS-422" เซมิคอนดัคเตอร์อิเลคทรอนิกส์ 115(มีนาคม 2535):132-136.

ภาคผนวก

PROGRAM LISTING

```
(********************)
(* Unit DTADIR.TPU *)
(********************)

Unit DTAdir;
interface
uses dos,crt,screen01,dtakey,Errormsg;

Procedure Readdir(Xwin,Ywin:byte; msk: string; var path,name:string);

implementation
type Dirptr    = ^dirlist;
     dirname   = string[12];
     dirlist   = record
                   name  : Dirname;
                   attr  : byte;
                   left,
                   right : Dirptr;
                 end;

var Dircount     : byte;
    DirInfo      : SearchRec;
    Headnode     : Dirptr; {Pointer to Header Node}
    DPtr         : Dirptr; {Current pointer}
    Xwint,Ywint  : byte;

const Xdim = 3; {Max number of file name per row}
      Ydim = 6;
var  namearr : array [1..ydim,1..xdim] of dirname;
```

```
Function GetNode :DirPtr;
var P :DirPtr;
begin
  New(p);
  GetNode := P;
end;

Procedure FreeDir(var P:DirPtr);
begin
  Dispose(HeadNode);
end;

Procedure InitHeader;
begin
  HeadNode        := Getnode;
  HeadNode^.left  := HeadNode;
  HeadNode^.right := HeadNode;
end;

Procedure InsertRight(AtNode :DirPtr; Name:DirName; Attr:byte);
var NewNode,Temp :Dirptr;
begin
  NewNode         := Getnode;
  NewNode^.Name   := Name;
  NewNode^.Attr   := Attr;
  Temp            := AtNode^.Right;
  Temp^.Left      := NewNode;
  NewNode^.Right  := Temp;
  NewNode^.Left   := AtNode;
  AtNode^.Right   := NewNode;
```

```
end;

Procedure InsertAfter(name:dirname; Attr:byte);
var Last :Dirptr;
begin
  last := Headnode^.left; {Pointer of last node}
  InsertRight(Last,Name,Attr);
end;

Procedure SearchDir(Name :DirName; var Post :byte; var Pt :DirPtr);
{ Count and move to current directory }
begin
  Post := 0;
  pt   := Headnode;
  repeat
    post := Post+1;
    pt   := pt^.right
  until ((name = pt^.name) or (pt = headnode));
  if pt = headnode then post :=0;
  { if not found name then position is 0 }
end;


Procedure DisplayDir(Xindex,Yindex:byte);
var Tail : char;
    Col,Row:byte;
begin
  col := (Xindex-1)*14+2;
  row := Yindex;
  Gotoxy(Col,Row);
```

```
  if dptr^.attr = Directory then tail :='\' else tail := ' ';
  Write(dptr^.Name,tail,' ':11- length(dptr^.name));
end;

Procedure PutDirtolist;
begin
  With dirinfo do
  begin
    if ((attr and sysfile)= 0) and
       ((attr and volumeid)=0) then
    begin
      Insertafter(name,attr);
      dircount:= dircount+1;
    end;
  end;
end;

Procedure Listdir(mask : string);
begin
  initheader;
  dircount := 0;
  findfirst(mask,anyfile,dirinfo);
  while doserror = 0 do
  begin
    findnext (dirinfo);
    if (doserror = 0) then putdirtolist;
  end;
end;

Procedure MaptoMatrix(size,post :byte; var x,y:byte);
```

```
var i,j,k :byte;
begin
     x := (post mod size);
     y := (post div size);
     if (x = 0) then x := size;
     if (post mod size) <> 0 then y := y + 1;
     dptr := headnode;
     for i := 1 to post do dptr := dptr^.right;
end;

Procedure Dir(mask : string);
var xindex,yindex :byte;
    post          :byte;
begin
  ListDir(Mask);
  for Post := 1 to Dircount do
  begin
    maptomatrix(xdim,post,xindex,yindex);
    displaydir(xindex,yindex);
  end;
end;

(*-------------- Function Getmatrix ------------------*)

Function Getmatrix(xdim,num,post : byte; var pt:dirptr): dirname;
var xindex,yindex :byte; { Current status XY }
    Key           :char; { Read to it       }
    Currentpost   :byte;
    SelectOK      :boolean; { Run Flag }
    OldX,OldY     :byte;
```

```
    currname      !dirname;
    Oldname       !dirname;
    Tail          !string;

Procedure nameup;
begin
  if post > xdim then
  begin
    currentpost != post;
    post        != Post - Xdim;
  end;
end;

Procedure NameDN;
begin
  if (post + xdim <= Num) then
  begin
    currentpost != post;
    post        != post+xdim;
  end;
end;

Procedure NameLF;
begin
  if post <> 1 then
  begin
    currentpost != post;
    post        != post-1;
  end
end;
```

```
Procedure NameRT;
begin
  if ((Post+1) <= num) then
  begin
    currentpost := post;
    post        := post+1;
  end;
end;

Procedure FirstName;
begin
  currentPost := Post;
  Post := 1;
end;

Procedure LastName;
begin
  currentpost := post;
  post := num;
end;

Procedure Returnvalue;
var tail : string;
begin
  pt         := dPtr;
  getmatrix  := dPtr^.name;
  selectOK   := true;
end;
```

```
Procedure ExitESC;
begin
  Getmatrix := #27;
  SelectOK  := true;
end;

procedure namematrix(inpt:dirptr);
var m,n : integer;
    Ptr : dirptr;
begin
  ptr := inpt;
  for m := 1 to ydim do begin
  for n := 1 to xdim do begin
  if ptr <> headnode then begin
     if (ptr^.attr and Directory) = Directory
        then tail := '\' else tail := ' ';
     if (ptr^.attr and VolumeId) <> VolumeId
        then namearr[m,n] := ptr^.name+tail;
     if (ptr^.attr and VolumeId) = VolumeId
        then begin
             n := n-1;
             if n=0 then begin
                n := Xdim; m := m-1 ;
                if m = 0 then m:=1;
             end;
        end;
     ptr := ptr^.right;
  end
  else
     namearr[m,n] := '              ';
```

```
  end; end;
end;

procedure rowup;
var rptr : dirptr;
    i,j  : byte;
begin
    rptr := dptr;
    i := (Xdim-1)-(Xdim-Xindex);
    if i <> 0 then begin
    For j := 1 to i do
    begin
        rptr := rptr^.left;
    end;
    end;
    namematrix(rptr);
end;

procedure rowdown;
var rptr : dirptr;
    i,j  : byte;
begin
    rptr := dptr;
    i := (Xdim-1)-(Xdim-Xindex)+(Xdim*(Ydim-1));
    for j := 1 to i do
    begin
        rptr := rptr^.left;
    end;
    namematrix(rptr);
end;
```

```
procedure disprow;
var i,j,x,y :byte;
begin
    for i := 1 to Ydim do
    begin
    for j := 1 to Xdim do
    begin
        x := (xwint)+((j-1)*14)+1;
        y := ywint+i;
        setattr(Reverselow);
        directwr(x,y,'               ');
        directWr(x,y,' '+Namearr[i,j]);
    end;
    end;
end;

Procedure dispname;
var i,j :byte;
    col,row :byte;
    tail    :string;
begin
    For i := 1 to Ydim do begin
    For j := 1 to Xdim do begin
      if namearr[i,j] = currname  then begin
        col := j; row := i;
        Oldx := j; Oldy := i;
      end;
    end; end;
    row := row + ywint;
```

```
    col := ((col-1)*14) + xwint+1;
    setattr(Highdisplay);
    directwr(col,row,' '+currname+' ');
end;

Function scroll:boolean;
var i,j : byte;
    tail : char;
begin
    scroll := true;
    For i := 1 to Ydim do begin
    For j := 1 to Xdim do begin
    if namearr[i,j] = currname then scroll := false;
    end; end;
end;

(*---------------- Main of Getmatrex ---------------*)
begin
  setattr(lowdisplay);
  currentpost := post;
  maptomatrix(xdim,post,xindex,yindex);
  if dptr^.attr = Directory then tail := '\' else tail := ' ';
  currname := dptr^.name+tail;
  rowup;
  disprow;
  dispname;
  selectOK := False;
  repeat
    readfunckey(key);
    if Funckey then
```

```
    begin
      case key of
        UP      : NameUP;
        Lt      : NameLF;
        Rt      : NameRt;
        Dn      : NameDn;
        Home    : Firstname;
        End_Key : Lastname;
      end;
      maptomatrix(Xdim,Post,Xindex,Yindex);
      if dptr^.attr = Directory then tail := '\' else tail := ' ';
      currname := dptr^.name+tail;
      if (Post > currentpost) and (scroll=true) then begin
         rowdown; end;
      if (post < currentpost) and (scroll=true) then begin
         rowup; end;
      disprow;
      dispname;
    end
    else
      case Key of
        CR    : ReturnValue;
        ESC   : ExitESC;
      end;
  until selectOK;
end; {of Getmatrix}

(*-------- Getpath ---------*)
function getpath(n : byte) :string;
var path : string;
```

```
begin
  getdir(N,path);
  if path[length(path)] = '\' then
     getpath := path
  else
     getpath := path +'\';
end;

(*--------Readdir-------*)
Procedure Readdir(Xwin,Ywin:byte; msk: string; var path,name:string);
{ Read directory and return
       Path --> Current Path
       Name --> File name
}
type String80 = string[80];
var  post          :byte;     { Position of DIRECTORY in list }
     P             :byte;     { Locate string postion of directory name }
     Drv           :byte;     { Number of drv to getpath }
     Xindex,Yindex :byte;     { Position of File name }
     Xwinb ,Ywinb  :byte;
     N             :STRING;   {DirName} { Hold Select Name }
     Curdir        :String80; { Holds current directory name }
     StartDir      :String;   { Holds Starting directory }
     Pt            :DirPtr;   { Points to current node }
     Finished      :Boolean;  { Run Flag }

Procedure Upcasestr(strIN:string;var strOUT:string);
var i : byte;
   tem : string;
begin
```

```
  Tem := '';
  For i := 1 to length(strIN) do
  begin
    Tem := Tem + Upcase(strIN[i]);
  end;
  strOUT := Tem;
end;


begin
  CursorOFF;
  GetDir(0,startdir);
  GetDir(0,curdir);
  Upcasestr(msk,msk);
  curdir := msk;
  chdir(curdir);
  curdir := curdir+'\';
  Xwint  := Xwin;
  Ywint  := Ywin;
  Xwinb  := Xwint+2+(14*Xdim);
  Ywinb  := Ywint+1+Ydim;
  post := 1;
  finished := false;
  listdir(msk);
  Repeat
      if Dircount = 0 then
      listdir('*.*');
      Setattr(ReverseLow);
      background(' ',xwint,ywint,xwinb,ywinb);
      messagebox(xwint,ywint,xwinb,ywinb,1);
```

```
directwr(xwint+2,ywint,' '+Curdir);
{Get name}
Searchdir(curdir,post,pt);
if post = 0 then post := 1;
N := getmatrix(xdim,dircount,post,pt);
case N[1] of
  #27 : begin
          Name   := #27;
          Finished :=  true; {Force to exit}
        end;
  ',' : begin
          getdir(0,curdir);
          p := length(curdir);
          repeat p := p-1; until curdir[p] = '\';
          curdir := copy(curdir,1,p);
          chdir('..');
          dircount := 0;
        end
else
    begin
      if (dptr^.attr and Directory) = Directory then
      begin
        chdir(dptr^.name);
        curdir := curdir + dptr^.name + '\';
        dircount := 0;
      end
      else
      begin
        name := N;
        path := curdir; {effect path to path}
```

```
            finished := true;
          end;
        end;
      end;
    Freedir(headnode);
  until Finished;
  Chdir(startdir); {restore old directory}
  clrdisp(Xwint,Ywint,Xwinb,Ywinb);
end;

{main}
begin
end.
```

```
(**********************)
(* Unit SCREEN01.TPU *)
(**********************)

Unit Screen01;
interface

uses Crt,Dos;

const  NoDisplay     = $00;
       LowDisplay    = $07;
       HighDisplay   = $0F;
       UnderLineLow  = $01;
       UnderLineHigh = $09;
       ReverseLow    = $70;
       ReverseHigh   = $78;
       BlinkLow      = $87;
       BlinkHigh     = $8F;
       UndBlinkLow   = $81;
       UndBlinkHigh  = $89;
       RevBlinkLow   = $F0;
       RevBlinkHigh  = $F8;
       ColorSeg      = $B800;
       MonoSeg       = $B000;
var    VideoSeg      : word;
       Crttype       : byte Absolute $0040:$0049;

Procedure SetAttr(Attrib: byte);
Procedure SetCursor(Top,Bottom :byte);
Procedure Cursoron;
```

```
Procedure Cursoroff;

{ End of interface section }

implementation

var Regs :registers;

Procedure SetAttr(Attrib: byte);
begin
  Textattr := Attrib;
end;

Procedure SetCursor(Top,Bottom : byte);
begin
  Regs.AH := 1;    {Function Cursor mode}
  Regs.CH := Top;
  Regs.CL := Bottom;
  intr($10,Regs);
end;

Procedure CursorOn;
begin
  Regs.AH := 2;
  Regs.DH := 0;
  Regs.DL := 0;
  Regs.BH := 0;
  Intr($10,Regs);
end;
```

```
Procedure CursorOff;
begin
  Regs.AH := 2;
  Regs.DH := 50;
  Regs.DL := 1;
  Regs.BH := 0;
  Intr($10,Regs);
end;

Procedure IdentifyCrt;
begin
  case CrtType of
       0..3     : Videoseg := Colorseg;
       7        : Videoseg := Monoseg;
  end;
end;

begin
  IdentifyCrt;
end.{of Unit}
```

```
(******************)
(* Unit CRIAL.TPU *)
(******************)

Unit Crial;
Interface
uses serial,dos,crt;

Function DATAin(timeout:longint):string;
Function DATAx(strIN:string):integer;
Function DATAy(strIN:string):integer;
Function int2realX(Xint_IN : integer):real;
Function int2realY(Yint_IN : integer):real;
Function real2intX(Xreal_IN : real):integer;
Function real2intY(Yreal_IN : real):integer;

Implementation

Function DATAin(timeout:longint):string;
var  crial : integer;
     strIn : string;
begin
    strIN := '';
    crial := setserial(com1,9600,0,8,1);
    crial := serialout('0');
    repeat
      crial := serialin(timeout);
      if crial <> -1 then begin
         strIN := strIN + chr(crial);
      end;
```

```
    until (crial = -1);
    DATAin := copy(strIN,1,length(strIN));
end;

Function DATAx(strIN:string):integer;
var res,data  : integer;
    str       : string ;
begin
   str := copy(strIN,1,5);
   val(str,DATA,res);
   dataX := data;
end;

Function DATAy(strIN:string):integer;
var data,res : integer;
    str      : string ;
begin
   str := copy(strIN,7,5);
   val(str,DATA,res);
   dataY := data;
end;

Function int2realX(Xint_IN : integer):real;
Begin
   int2realX := Xint_IN/10.2375;
end;

Function int2realY(Yint_IN : integer):real;
Begin
   int2realY := Yint_IN/409.5;
```

```
end;

Function real2intX(Xreal_IN : real):integer;
begin
   Xreal_IN := Xreal_IN * 10.2375;
   real2intX := round(Xreal_IN);
end;

Function real2intY(Yreal_IN : real):integer;
begin
   Yreal_IN := Yreal_IN * 409.5;
   real2intY := round(Yreal_IN);
end;

begin {main}
end.
```

```
(********************)
(* Unit SERIAL.TPU *)
(********************)

Unit Serial;

Interface

  const COM1 = 0;
        COM2 = 1;
        COM3 = 3;
        COM4 = 4;

        COM1BASE = $3F8;
        COM2BASE = $2F8;
        COM3BASE = $3E8;
        COM4BASE = $2E8;

        TXR = 0;
        RXR = 0;
        IER = 1;
        IIR = 2;
        LCR = 3;
        MCR = 4;
        LSR = 5;
        MSR = 6;
        SRP = 7;
        DLL = 0;
        DLH = 1;
```

```
RX_INT  = $01;
TBE_INT = $02;
ERR_INT = $04;
RS_INT  = $08;

RX_ID   = $04;
RX_MASK = $07;

NO_PARITY   = $00;
EVEN_PARITY = $018;
ODD_PARITY  = $08;
DLAB        = $80;

DTR = $01;
RTS = $02;
GPO1 = $04;
GPO2 = $08;
EN_INT = $08;

RCVRDY  = $01;
OVRERR  = $02;
PATYERR = $04;
FRMERR  = $08;
BRKERR  = $10;
XMTRDY  = $20;
XMTRSR  = $40;
TIMEOUT = $80;

D_CTS   = $01;
D_DSR   = $02;
```

```
        D_RI    = $04;
        D_DCD   = $08;
        CTS     = $10;
        DSR     = $20;
        RI      = $40;
        DCD     = $80;

        IMR     = $21;
        ICR     = $20;

        EOI     = $20;

        IRQ4    = $EF;
        IRQ3    = $F7;

        SBUFSIZ = 4000;
var portbase:Integer;

Function Setserial(comport:integer;baud:longint;parity,databit,stopbit:integer)
Function Serialin(Time_out:LongInt):integer;
Function Serialout(ch:char):integer;

Implementation


Function Setserial(comport:integer;baud:longint;parity,databit,stopbit:integer)
var  divisor,currentV,setting,orpara:integer;

Begin
  case comport of
```

```
   com1  : portbase := com1base;
   com2  : portbase := com2base;
   com3  : portbase := com3base;
   com4  : portbase := com4base;
   Else Begin
        SetSerial := -1;
        exit;
        End;
  End;
  If ((baud<=0) or (baud>115200)) Then
  Begin
    SetSerial := -2;
    exit;
  End Else divisor := trunc(115200/baud);
  currentV := Port[Portbase+ LCR];
  port[portbase+LCR] := (currentV or DLAB);
  port[portbase+DLL] := (divisor and $00ff);
  divisor := divisor shl 8;
  port[portbase+DLH] := (divisor and $00ff);
  port[portbase+LCR] := currentV;
  setting := databit -5;
  if (stopbit =1) then  orpara := $00 else orpara := $04;
  setting := setting Or orpara;
  setting := setting or parity;
  port[portbase+LCR] := setting;
  setserial := 0;
End;

Function Serialin(Time_out:LongInt):integer;
var value : Integer;
```

```
    start,check : Byte;
    LSRcheck    : Integer;
    count  : Word;
Begin
  count := $0000;
  repeat
  inc(count);
  LSRcheck := port[portbase + LSR];
  until ((LSRcheck and RCVRDY) <> 0) or
        (count = Time_out);
  value := port[portbase+RXR];
  if count = Time_out Then serialin := -1
        else  serialin := value;
End;

Function SerialOut(ch:char) : Integer;
var value : Integer;
Begin
  value := ord(ch);
  Repeat Until ( ( port[portbase+LSR] and XMTRDY) <> 0);
  port[portbase + TXR ] := value;
End;
begin
End.
```

```
(************************)
(*  Unit DTATITLE.TPU  *)
(************************)


Unit Dtatitle;
Interface

uses dtakey,screen01,crt,dos;

Procedure showtitle(x,y:byte);
Procedure showmain;

Implementation
(******************** Title ********************)

Procedure showtitle(x,y:byte);
var c : char;
begin
  disp(x,y,x+26,y+6);
  directwr(x+1,y+1,'    DTA. Data Analyzer   ');
  directwr(x+1,y+2,'        Version 1.0      ');
  directwr(x+1,y+3,'  By Ekarat Lopetcharrat ');
  directwr(x+1,y+4,'     Applied Physics     ');
  directwr(x+1,y+5,'        K.M.I.T.L        ');
  c := readkey;
  clrdisp(x,y,x+26,y+6);
end;
```

```
(***************** Main menu ******************)

Procedure mainbound;
var i : byte;
begin
  setattr(Reverselow);
  directwr(2,2,#201);
  directwr(2,23,#200);
  directwr(79,2,#187);
  directwr(79,23,#188);
  directwr(2,3,#186);
  directwr(79,3,#186);
  directwr(2,4,#204);
  directwr(79,4,#185);
  directwr(2,5,#186);
  directwr(79,5,#186);
  directwr(2,6,#199);
  directwr(79,6,#182);
  i := 0;
  repeat
    i := i+1;
    directwr(2,6+i,#186);
    directwr(79,6+i,#186);
  until (6+i = 22);
  i := 0;
  repeat
    i := i+1;
    directwr(2+i,2,#205);
    directwr(2+i,23,#205);
    directwr(2+i,4,#205);
```

```
    directwr(2+i,6,#196);
  until ( 2 + i = 78);
end;

Procedure mainmessage(x,y:byte);
begin
  directwr(x,y,'                        ');
  directwr(x+23,y,'Differential Thermal Analysis');
  directwr(x+52,y,'                          ');
  directwr(x,y+2,'      File                      Data Analysis ');
  directwr(x+45,y+2,'                    Print      ');
  setattr(Highdisplay);
  directwr(2,25,'F2');
  directwr(17,25,'Alt X');
  setattr(Lowdisplay);
  directwr(4,25,'-Graph/menu');
  directwr(22,25,'-Exit');
end;

Procedure showmain;
var loop : byte;
begin
  mainbound;
  setattr(lowdisplay);
  for loop := 1 to 80 do begin
      directwr(loop,1,' ');
      directwr(loop,24,' ');
      directwr(loop,25,' ');
  end;
  setattr(Reverselow);
```

```
  mainmessage(3,3);
  setattr(highdisplay);
  background(#177,3,7,78,22);
  cursoroff;
end;
end.
```

```
(*********************)
(* Unit DTASETUP.TPU *)
(*********************)

Unit DtasetUP;
interface
uses DOS,CRT,DTAKEY,SCREEN01;

Procedure Dta_SetUp(X,Y: Byte; SamIN : string;
                    massIN,ScaleY,OrgY : real;
                    Var SamOut   : string;
                    Var massOut  : real;
                    Var ScaleOut : real;
                    Var OrgOut   : real);

Implementation
Procedure Dta_SetUp(X,Y   : Byte;
                    SamIN : string;
                    massIN,ScaleY,OrgY : real;
                    Var SamOut   : string;
                    Var massOut  : real;
                    Var ScaleOut : real;
                    Var OrgOut   : real);
var Left    : byte;
    Para    : byte;
    res     : integer;
    ch      : char;

const Blanks = '          ';
      MaxRow = 4;
```

```
var   row_arr : array[1..MaxRow] of string;

      Function real2str(realIN:real):string;
      var st : string;
          dummy : real;
          i  : byte;
      begin
        dummy := realIN;
        str(dummy:7:3,st);
        i := 1;
        While (st[i] = ' ') do
        begin
           Inc(i);
        end;
        Real2str := copy(st,i,length(st));
      end;

      Procedure Readsamp(x1,y1:byte);
      begin
        setattr(Highdisplay);
        Directwr(X1+10,Y1,' Sample : ');
        setattr(Lowdisplay);
        GotoXY(X1+1,Y1+1);
        write(samOUT);
        readstr(samOUT,30,samOUT);
        if length(samOUT) > 10 then row_arr[1] := copy(samOUT,1,10)
        else row_arr[1] := samOUT;
        Left := 1;
      end;
```

```
Procedure Readmass(x1,y1:byte);
begin
  setattr(Highdisplay);
  Directwr(X1+9,Y1,' Mass(Gram) : ');
  setattr(Lowdisplay);
  GotoXY(X1+1,Y1+1);write(row_arr[2]);
  ReadReal(massOUT,6,3,massOUt,res);
  row_arr[2] := real2str(massOUT);
end;

Procedure Readscale(x1,y1:byte);
begin
  setattr(Highdisplay);
  Directwr(X1+9,Y1,' Scale(Y) : ');
  setattr(Lowdisplay);
  GotoXY(X1+1,Y1+1); write(row_arr[3]);
  ReadReal(scaleOUT,6,3,scaleOUt,res);
  row_arr[3] := real2str(scaleOUT);
end;

Procedure ReadorgY(x1,y1:byte);
begin
  setattr(Highdisplay);
  Directwr(X1+9,Y1,' Origin(Y) : ');
  setattr(Lowdisplay);
  GotoXY(X1+1,Y1+1); write(row_arr[4]);
  ReadReal(orgOUT,6,3,orgOUt,res);
  row_arr[4] := real2str(orgOUT);
end;
```

```
    Procedure initsetup;
    var i : byte;
    begin
      CursorOff;
      disp(X,Y,X+26,Y+1+MaxRow);
      setattr(lowdisplay);
      row_arr[1] := ' Sample     : '+ Blanks;
      row_arr[2] := ' Mass(gram) : '+ Blanks;
      row_arr[3] := ' Scale(Y)   : '+ Blanks;
      row_arr[4] := ' Origin(Y)  : '+ Blanks;
      For i := 1 to MaxRow do begin
        directwr(X+1,Y+i,row_arr[i]);
      end;
      if length(samIN) > 10 then row_arr[1] := copy(samIN,1,10)
      else row_arr[1] := samIN;
      row_arr[2] := real2str(massOUT);
      row_arr[3] := real2str(scaleOUT);
      row_arr[4] := real2str(orgOUT);
      setattr(highdisplay);
      For i := 1 to MaxRow do begin
        directwr(X+15,Y+i,row_arr[i]);
      end;
      setAttr(ReverseLow);
      directwr(X+15,Y+1,row_arr[1]);
      setAttr(lowdisplay);
    end;
begin
  para := 1; Left := 1;
  samOUT := samIN;
```

```
massOUT := massIN;
scaleOUT := scaleY;
orgOUT := orgY;
initsetup;
Repeat
  Ch := Readkey;
  case Ch of
   #72 : Begin  (* Up Arrow *)
          setattr(Highdisplay);
          directwr(X+15,Y+para,Row_arr[para]);
          Dec(para); if para = 0 then para := maxRow;
          setattr(Reverselow);
          directwr(X+15,Y+para,Row_arr[para]);
          setattr(lowdisplay);
         end;
   #80 : Begin  (* Down Arrow *)
          setattr(Highdisplay);
          directwr(X+15,Y+para,Row_arr[para]);
          Inc(para); if para = maxRow+1 then para := 1;
          setattr(Reverselow);
          directwr(X+15,Y+para,Row_arr[para]);
          setattr(lowdisplay);
         end;
   #77 : Begin (* Right Arrow *)
          if para = 1 then begin
            if (length(samOUT)-Left) >= 10 then begin
              Inc(Left);
              row_arr[1] := copy(samOUT,Left,10);
            end;
            setattr(Reverselow);
```

```
          directwr(X+15,Y+para,Row_arr[1]);
          setattr(lowdisplay);
        end;
      End;
#75 : Begin (* Left Arrow *)
        if para = 1 then begin
          if Left > 1 then begin
            Dec(Left);
            row_arr[1] := copy(samOUT,Left,10);
          end;
          setattr(Reverselow);
          directwr(X+15,Y+para,row_arr[1]);
          setattr(lowdisplay);
        end;
      End;
#13 : Begin (* Enter Key *)
        disp(X-9,Y+1+MaxRow,X+24,Y+3+MaxRow);
        setattr(lowdisplay);
        CursorON;
        Case (para) of
          1 : Readsamp(x-9,y+1+maxrow);
          2 : Readmass(x-9,y+1+maxrow);
          3 : Readscale(x-9,y+1+maxrow);
          4 : ReadorgY(x-9,y+1+maxrow);
        end; (* Sub Case *)
        CursorOFF;
        Clrdisp(X-9,Y+1+MaxRow,X+24,Y+3+Maxrow);
        directwr(X+15,Y+para,blanks);
        setattr(reverselow);
        directwr(X+15,Y+para,row_arr[para1);
```

```
            setattr(Highdisplay);
            messagebox(X,Y,X+26,Y+MaxRow+1,2);
            setattr(lowdisplay);
            shadow(X,Y,X+26,Y+MaxRow+1);
          end;
    end; (* Case *)
  Until Ch = #27; {Press ESC to EXIT}
  clrdisp(X,Y,X+26,Y+MaxRow+1);
end;

begin (* main *)
end.
```

```
(********************)
(* Unit DTAGRAF.TPU *)
(********************)

Unit Dtagraf;
Interface

uses crt,dos,graph;
const minimumX = 0;

var   pictpoint : pointer;
      Memused   : Word; (* byte of memory *)
      ch        : char;

Procedure Opengraf(path:string);
Procedure Closegraf;
Procedure Axislength(var Xlong,Ylong:integer);
Procedure Scalex(var PxperuX,UpertickX:real);
Function  UpertickY(datain : real):real;
Procedure YScale(datain : real; var UpY,PxY:real);
Function  minx(datain:real):real;
Function  miny(datain:real):real;
Procedure orgxy(var orgx,orgy:integer);
Procedure getpict(x1,y1,x2,y2:integer;var pictptr:pointer;
                                    var memo   :word  );
Procedure Putpict(x1,y1:integer);
Procedure Axis(ScaleY,minvY:real);
Procedure Dotdata(DataX,DataY,ScaleY,minvY:real;color:integer);

Implementation
```

```
Procedure Opengraf(path:string);
var grdrv,grmode : integer;
begin
    Detectgraph(grdrv,grmode);
    case (grdrv) of
        1  :grmode := 4;
      2,8  :grmode := 5;
    3,4,9  :grmode := 1;
        5  :grmode := 3;
    7,6,10 :grmode := 0;
       -2  :begin
              clrscr;
              gotoxy(1,1);
              write ('No graphic system.');
            end;
    end;{case}
    initgraph(grdrv,grmode,path);
end;{Procedure Opengraph}

Procedure Closegraf;
begin
    closegraph;
end;

(*------- Initial value for DTAgraph -------*)
Procedure Axislength(var Xlong,Ylong:integer);
var grdrv,grmode : integer;
begin
```

```
    Detectgraph(grdrv,grmode);
    if (grdrv = 1 ) then
    begin
        Xlong := 550;
        Ylong := 150;
    end
    else
    begin
        Xlong := 550;
        Ylong := 200;
    end;
end;

Procedure Scalex(var PxperuX,UpertickX:real);
begin {Pixel per Unit,Unit per tick}
    PxperuX  := 1.0625;
    UpertickX := 50;
end;

Function UpertickY(datain : real):real;
begin
  if datain = 0 then datain :=0.36;
  UpertickY := datain;
end;

Procedure YScale(datain : real; var UpY,PxY:real);
begin {Pixel per Unit,Unit per tick}
    UpY := UpertickY(datain);
    PxY := 20/UpY;
end;
```

```
Function minx(datain:real):real;
begin {minimum value of x}
   minx := datain;
end;

Function miny(datain:real):real;
begin {minimum value of y}
   miny := datain;
end;

Procedure orgxy(var orgx,orgy:integer);
var Xlng,Ylng : integer;
begin
    Axislength(Xlng,Ylng);
    orgx := round((getmaxx-Xlng)/2)+15;
    orgy := round(((getmaxy-Ylng)/2)+Ylng);
end;

procedure getpict(x1,y1,x2,y2:integer;var pictptr:pointer;
                                        var memo   :word  );
begin
   Memo := Imagesize(x1,y1,x2,y2);
   getmem(pictptr,memo);
   getimage(x1,y1,x2,y2,pictptr^);
end;

Procedure Putpict(x1,y1:integer);
begin
   putimage(x1,y1,pictpoint^,normalput);
```

```
end;

(*------------ Drawing Axis -----------*)
Procedure Axis(ScaleY,minvY:real);{Show Axis}
(* Consist of subprocedure:
           Xtick;
           Ytick;
           Axisname(Xname,Yname); *)
var Xlngs,Ylngs,orX,orY  : integer;
    PuX,PuY,UtX          : real;
    minvX                : real;

Procedure Xtick;
var i,pxpertick :  integer;
    j  :  real;
    s  :  string;
begin
    pxpertick := round(PuX*UtX);
    j  := minvX; i := 0;
    settextstyle(Smallfont,Horizdir,2);
    settextjustify(Centertext,Centertext);
    repeat
       setlinestyle(UserBitln,$ffff,NormWidth);{ Tick line }
       line (orX+i,orY,orX+i,orY+4);
       setlinestyle(UserBitln,$aaaa,NormWidth);{ Grid line }
       line (orX+i,orY,orX+i,orY-Ylngs);
       str(j:6:2,s);
       outtextxy(orX+i,orY+10,s);
       j := j+UtX;
       i := i + pxpertick;
```

```
    until ((orX+i) > (orX+Xlngs));
end;

Procedure Ytick;
var i,pxpertick  :  integer;
    j  :  real;
    s  :  string;
begin
    pxpertick := round(PuY*ScaleY);
    j  := minvY; i := 0;
    settextstyle(Smallfont,Horizdir,2);
    settextjustify(Righttext,Centertext);
    repeat
       setlinestyle(UserBitln,$ffff,NormWidth);{ Tick line }
       line (orX,orY-i,orX-4,orY-i);
       setlinestyle(UserBitln,$aaaa,NormWidth);{ Grid line }
       line (orX,orY-i,orX+Xlngs,orY-i);
       str(j:6:2,s);
       outtextxy(orX-10,orY-i,s);
       j := j+ScaleY;
       i := i + pxpertick;
    until ((orY-i) <= (orY-Ylngs));
end;

Procedure Axisname(Xname,Yname:string);
var j : integer;
begin
    settextstyle(5,Horizdir,1);
    settextjustify(Centertext,Centertext);
    j := round(orX+Xlngs/2);
```

```
    outtextxy(j,orY+30,Xname);
    settextjustify(lefttext,Bottomtext);
    j := orY-Ylngs-5;
    outtextxy(orX,j,Yname);
end;

Procedure Headname(Head:string);
var j : integer;
begin
    settextstyle(3,Horizdir,2);
    settextjustify(Centertext,Centertext);
    j := round(orX+Xlngs/2);
    outtextxy(j,orY-(Ylngs+35),Head);
end;

Begin {Axis}
    Axislength(Xlngs,Ylngs);
    ScaleX(PuX,UtX);
    YScale(ScaleY,ScaleY,PuY);
    minvX := minx(minimumX);
    orgXY(orX,orY);
    line(orX,orY,orX+Xlngs,orY);
    line(orX,orY,orX,orY-Ylngs);
    line(orX+Xlngs,orY-Ylngs,orX,orY-Ylngs);
    line(orX+Xlngs,orY,orX+Xlngs,orY-Ylngs);
    Xtick;
    Ytick;
    Headname('Differential Thermal Analysis');
    Axisname('Temperature (Celcius)',#127+'T (Celcius)');
end;
```

```
(*------------------- Plot Graph ---------------------*)
Procedure Utopix(PperU,Units:real; var pix:integer);
begin
    pix := round(PperU*Units);
end;

Procedure Dotdata(DataX,DataY,ScaleY,minvY:real;color:integer);
var orX,orY,Xlngs,Ylngs,Xpix,Ypix : integer;
    Pux,Puy,UtX,minvX             : real;
begin
    Axislength(Xlngs,Ylngs);
    ScaleX(PuX,UtX);
    YScale(ScaleY,ScaleY,PuY);
    minvX := minx(minimumX);
    orgXY(orX,orY);
    Utopix(PuX,DataX-minvX,Xpix);
    Xpix := Xpix+orX;
    Utopix(PuY,DataY-minvY,Ypix);
    Ypix := orY-Ypix;
    if (Xpix > orX) and (Xpix < (orX+Xlngs)) and
       (Ypix < orY) and (Ypix > (orY-Ylngs)) then
       putpixel(Xpix,Ypix,color);
end;

begin {main}
end.
```

```
(**********************)
(*  Unit DTAKEY.TPU  *)
(**********************)

Unit DtaKey;
Interface

Uses Crt,Dos,Screen01;
var    Crttype : byte Absolute $0049:0040;
       Scrnseg : longint;
       c : char;
       Funckey : boolean;
const  Monoseg = $b000;
       Colorsg = $b800;
       CR = #13;  BS = #8 ; SP = #32; ESC = #27;
       UP = #72;  LT = #75; RT = #77; DN  = #80;
       HOME = #71; END_Key = #79;

(*---------------- Part 1 : Crt -------------------*)
Procedure directwr(col,row:byte;st : string);
Function GetchXY(col,row:byte):char;
Function GetAttrXY(col,row:byte):byte;
Procedure background(ch:string;x1,y1,x2,y2:byte);
Procedure shadow(x1,y1,x2,y2:byte);
Procedure messagebox(x1,y1,x2,y2,style:byte);

(*---------- Extention of CRT part ---------------*)
Procedure Disp(X1,Y1,X2,Y2:byte);
Procedure ClrDisp(X1,Y1,X2,Y2:byte);
```

```
(*------------ Part 2 : Keyboard -----------------*)
Procedure ReadFunckey(var key:char);
Procedure Readchr(var ch  :char ;
                  var func:byte);
Procedure myRtrim(strIN:string; var strOUT:string);
Procedure myLtrim(strIN:string; var strOUT:string);
Function  testch(var ch:char):char;
Procedure Readstr(bstr:string ;loop :byte ; var astr : string);
Procedure ReadInt(bint:integer;loop :byte ; var aint : integer;
                                          var code : integer);
Procedure Readreal(breal:real;loop,Deci:byte ; var areal : real;
                                             var code : integer);

Implementation

(*------------ Part 1 : Crt --------------*)
(************ Direct video ram ***************)
(* set attribute by procedure setattr(attr) *)

Procedure directwr(col,row:byte;st : string);
var Pos : byte;
    Ofset : word;
begin
    Ofset := 160*(row-1)+2*(col-1);
    For Pos := 1 to length(st) do
    begin
      Mem[videoseg:ofset] := Ord(st[pos]);
      Mem[videoseg:ofset+1] := textattr;
      Ofset := Ofset + 2;
    end;
```

```
end;

Function GetchXY(col,row:byte):char;
var Pos : byte;
    Ofset : word;
begin
    Ofset := 160*(row-1)+2*(col-1);
    GetchXY := chr(Mem[videoseg:ofset]);
end;

Function GetAttrXY(col,row:byte):Byte;
var Pos : byte;
    Ofset : word;
begin
    Ofset := 160*(row-1)+2*(col-1);
    GetAttrXY := (Mem[videoseg:ofset+1]);
end;

Procedure background(ch:string;x1,y1,x2,y2:byte);
var i,j : byte;
begin
  j := 0;
  repeat
     i := 0;
     repeat
        directwr(x1+i,y1+j,ch);
        i := i+1;
     until (x1+i > x2);
     j := j+1;
  until (y1+j > y2);
```

```
end;

Procedure shadow(x1,y1,x2,y2:byte);
var i : byte;
begin
  i:= 0;
  repeat
    directwr(x2+1,y1+1+i,GetchXY(x2+1,y1+1+i));
    directwr(x2+2,y1+1+i,GetchXY(x2+2,y1+1+i));
    i:= i+1;
  until (y1+1+i > y2+1);
  i:= 0;
  repeat
    directwr(x1+2+i,y2+1,GetchXY(x1+2+i,y2+1));
    i:= i+1;
  until (x1+1+i > x2);
end;

Procedure messagebox(x1,y1,x2,y2,style:byte);
var i : byte;
begin
  case (style) of
  2 : begin
        directwr(x1,y1,chr(201));
        directwr(x2,y1,chr(187));
        directwr(x1,y2,chr(200));
        directwr(x2,y2,chr(188));
        i := 1;
        While (y1+i <= y2-1) do
        begin
```

```
        directwr(x1,y1+i,chr(186));
        directwr(x2,y1+i,chr(186));
        Inc(i);
      end;
      i := 1;
      While (x1+i <= x2-1) do
      begin
        directwr(x1+i,y1,chr(205));
        directwr(x1+i,y2,chr(205));
        Inc(i);
      end;
    end;
  1 : begin
      directwr(x1,y1,chr(218));
      directwr(x2,y1,chr(191));
      directwr(x1,y2,chr(192));
      directwr(x2,y2,chr(217));
      i := 1;
      While (y1+i <= y2-1) do
      begin
        directwr(x1,y1+i,chr(179));
        directwr(x2,y1+i,chr(179));
        i:= i+1;
      end;
      i := 1;
      While (x1+i <= x2-1) do
      begin
        directwr(x1+i,y1,chr(196));
        directwr(x1+i,y2,chr(196));
        i:= i+1;
```

```
        end;
      end;
  end;{case}
end;

(*---------- Extention of CRT part ----------*)

Procedure Disp(X1,Y1,X2,Y2:byte);
begin
    SetAttr(highdisplay);
    BackGround(' ',X1,Y1,X2,Y2);
    messagebox(X1,Y1,X2,Y2,2);
    SetAttr(lowdisplay);
    shadow(X1,Y1,X2,Y2);
end;

Procedure ClrDisp(X1,Y1,X2,Y2:byte);
begin
  SetAttr(highdisplay);
  BackGround(chr(177),X1,Y1,X2,Y2);
  shadow(X1,Y1,X2,Y2);
end;

(*---------- Part 2 : Keyboard ----------*)

Procedure ReadFunckey(var key:char);
begin
  key := readkey;
  if key <> #00 then
      Funckey := false
```

```
  else begin Funckey := true;
       key := readkey; end;
end;

Procedure Readchr(var ch  :char;
                  var func:byte);
var regs:registers;
begin
  regs.AH := $00;
  intr($16,regs);
  ch := chr(regs.AL);
  func := regs.AH;
end;

Procedure myRtrim(strIN:string; var strOUT:string);
var i,j : integer;
    quit: boolean;
begin
  i := length(strIN)+1;
  quit := false;
  while quit = false do begin
    i := i-1;
    if strIN[i] <> ' ' then quit := true;
  end;
  strOUT := copy(strIN,1,i);
  strOUT[0] := chr(length(strOUT));
end;

Procedure myLtrim(strIN:string; var strOUT:string);
var i,j : integer;
```

```
    quit : boolean;
begin
  i := 0;
  quit := false;
  while quit = false do  begin
    i := i+1;
    if strIN[i] <> ' ' then quit := true;
  end;
  strOUT    := copy(strIN,i,length(strIN)-i+1);
  strOUT[0] := chr(length(strOUT));
end;

Function testch(var ch:char):char;
var reg : registers;
begin
  repeat
    reg.ah := 1; intr(22,reg);
  until (reg.flags and Fzero) = 0;
    ch := chr(reg.al); testch := ch;
    if ch < #32 then ch:= readkey;
    if ch = #00 then ch:= readkey;
end;

Procedure Readstr(bstr:string;loop :byte ; var astr : string);
var   i    : byte;
      scan : byte;
      ch   : char;
      quit : boolean;
     x,y   : byte;
```

```
Procedure Delete(strIN:string;var strOUT:string;
                              var ptr,x,y : byte);
begin
  ptr := length(strIN)+1;
  if ptr <> 1 then
  begin
     ptr := ptr-1;
     write(ch);
     write(' ');
     write(ch);
     strOUT := copy(strIN,1,length(strIN)-1);
  end
  else
    write(^G);
  x := wherex; y := wherey;
end;

Procedure Character(loop:byte; strg : string;
                              var strg2   : string;
                              var ptr,x,y : byte);
begin
  ptr := length(strg)+1;
  if ptr > loop then
     Write(^G)
  else
  begin
     write(ch);
     Ptr   := ptr+1;
     strg2 := strg+ch;
  end;
```

```
  x := wherex; y := wherey;
end;

Procedure EnterKey(strg:string; var strg2 :string;
                                var quit  :boolean);
begin
  if strg = '' then begin strg := #13; quit := true; end
  else begin strg2:=strg; quit := true; end;
end;

Procedure Escape(var quit:boolean);
begin
  quit := true;
end;

begin
    x := wherex ; y := wherey;
    astr := bstr;
    i := length(bstr) + 1;
    quit := false;
    repeat
      readchr(ch,scan);
      if (ch = cr) and  (scan = $1c) then Enterkey(bstr,astr,quit);
      if (ch = esc) and (scan = $01) then Escape(quit);
      if (ch = bs)  and (scan = $0e) then
         delete(bstr,bstr,i,x,y);
      if (ch <> #00) and (ch >= #32) then
         character(loop,bstr,bstr,i,x,y);
      gotoxy(x,y);
    until quit;
```

```
end;

Procedure ReadInt(bint:integer;loop :byte ; var aint : integer;
                                          var code : integer);
var   i    : byte;
      scan : byte;
      ch   : char;
      quit,minusflag : boolean;
      x,y  : byte;
      transb : string;
      oldval : string;

Procedure Delete(strIN:string;var strOUT:string;
                              var ptr,x,y : byte);
begin
  ptr := length(strIN)+1;
  if ptr <> 1 then
  begin
     ptr := ptr-1;
     write(ch);
     write(' ');
     write(ch);
     strOUT := copy(strIN,1,length(strIN)-1);
  end
  else
    write(^G);
  x := wherex; y := wherey;
end;

Procedure Intnum(loop:byte; strg : string;
```

```
                    var strg2   : string;
                       var ptr,x,y : byte);
begin
  ptr := length(strg)+1;
  if strg[1] = '-' then
  begin
  if ptr > loop then Write(^G)
  else
  begin
     write(ch);
     Ptr   := ptr+1;
     strg2 := strg+ch;
  end;
  end;

  if strg[1] <> '-' then
  begin
  if ptr > loop-1 then write(^G)
  else
  begin
     write(ch);
     ptr  := ptr+1;
     strg2 := strg+ch;
  end;
  end;
  x := wherex; y := wherey;
end;

Procedure EnterKey(strg:string; var strg2 :string;
                                var quit  :boolean);
```

```
begin
  if strg = '' then begin strg2 := oldval; quit := true; end
  else begin strg2:=strg; quit  := true; end;
end;

Procedure Escape(var quit:boolean);
begin
  quit := true;
end;

Procedure minus(strIN : string;var strOUT  : string;
                              var ptr,x,y : byte);
begin
  ptr := length(strIN)+1;
  if (minusflag) and (ptr = 1) then
  begin
    write(ch);
    ptr := ptr+1;
    strOUT := strIN+ch;
    minusflag := false;
  end
  else write(^G);
  x := wherex;y:=wherey;
end;

begin
    x := wherex ; y := wherey;
    str(bint:loop,transb);
    myLtrim(transb,transb);
    oldval := transb;
```

```
    i := length(transb) + 1;
    quit := false;
    minusflag := false;
    repeat
      readchr(ch,scan);
      if (ch = cr) and  (scan = $1c) then
      begin
        Enterkey(transb,transb,quit);
      end;
      if (ch = esc) and (scan = $01) then
      begin
        Escape(quit); transb := oldval;
      end;
      if (ch = bs)  and (scan = $0e) then
         delete(transb,transb,i,x,y);
      if (ch <> #00) and (ch >= #32) then
         case ch of
         '-': begin minusflag := true; minus(transb,transb,i,x,y);
              gotoxy(x,y);
              end;
         '0'..'9' : intnum(loop,transb,transb,i,x,y);
         else write(^G);
         end;
      gotoxy(x,y);
    until quit;
    val(transb,aint,code);
    if code <>0 then code:= 255;
end;

Procedure Readreal(breal:real;loop,Deci :byte ; var areal : real;
```

```
                                        var code : integer);
var   i    : byte;
      scan : byte;
      ch   : char;
      quit,minusflag,pointflag : boolean;
      x,y  : byte;
      transb : string;
      oldval : string;
      tempst : string;

Procedure Delete(strIN:string;var strOUT:string;
                              var ptr,x,y : byte);
begin
  ptr := length(strIN)+1;
  if ptr <> 1 then
  begin
     ptr := ptr-1;
     write(ch);
     write(' ');
     write(ch);
     strOUT := copy(strIN,1,length(strIN)-1);
  end
  else
    write(^G);
  x := wherex; y := wherey;
end;

Procedure Realnum(loop:byte; strg : string;
                  var strg2   : string;
                     var ptr,x,y : byte);
```

```
begin
 ptr := length(strg)+1;
  if strg[1] = '-' then
  begin
  if ptr > loop+1 then Write(^G)
  else
  begin
     write(ch);
     Ptr  := ptr+1;
     strg2 := strg+ch;
  end;
  end;

  if strg[1] <> '-' then
  begin
  if ptr > loop then write(^G)
  else
  begin
     write(ch);
     ptr  := ptr+1;
     strg2 := strg+ch;
  end;
  end;
  x := wherex; y := wherey;
end;

Procedure EnterKey(strg:string; var strg2 :string;
                               var quit  :boolean);
begin
  if strg = '' then begin strg2 := oldval; quit := true; end
```

```
  else begin strg2:=strg; quit  := true; end;
end;

Procedure Escape(var quit:boolean);
begin
  quit := true;
end;

Procedure point(strIN : string;var strOUT  : string;
                                var ptr,x,y : byte);
var j : byte;
begin
  ptr := length(strIN)+1;
  j := 0;
  repeat
    if strIN[j] = '.' then pointflag := false;
    j := j + 1;
  until (j = ptr) or (pointflag = false);
  if pointflag then
  begin
    write(ch);
    ptr := ptr+1;
    strOUT := strIN+ch;
    pointflag := false;
  end
  else write(^G);
  x := wherex ; y := wherey;
end;

Procedure minus(strIN : string;var strOUT  : string;
```

```
                                var ptr,x,y : byte);
begin
  ptr := length(strIN)+1;
  if (minusflag) and (ptr = 1) then
  begin
    write(ch);
    ptr := ptr+1;
    strOUT := strIN+ch;
    minusflag := false;
  end
  else write(^G);
  x := wherex;y:=wherey;
end;

begin
    str(breal:loop:Deci,transb);
    myLtrim(transb,transb);
    oldval := transb;
    tempst := '0';
    i := length(transb) + 1;
    quit := false;
    x := wherex; y := wherey;
    gotoxy(x,y);
    minusflag := false;
    pointflag := false;
    repeat
      readchr(ch,scan);
      if (ch = cr) and  (scan = $1c) then
      begin
        Enterkey(transb,transb,quit);
```

```
      end;
      if (ch = esc) and (scan = $01) then
      begin
        Escape(quit); transb := oldval;
      end;
      if (ch = bs)  and (scan = $0e) then
         delete(transb,transb,i,x,y);
      if (ch <> #00) and (ch >= #32) then
         case ch of
         '.': begin pointflag := true; point(transb,transb,i,x,y);
              gotoxy(x,y);
              end;
         '-': begin minusflag := true; minus(transb,transb,i,x,y);
              gotoxy(x,y);
              end;
         '0'..'9' : realnum(loop,transb,transb,i,x,y);
         else write(^G);
         end;
      gotoxy(x,y);
    until quit;
    if transb[1] = '.' then transb := '0'+transb;
    if transb[length(transb)] = '.' then transb := transb+'0';
    if (transb[1] = '-') and (transb[2] = '.') then
      insert('0',transb,2);
    val(transb,areal,code);
    if code <> 0 then code:= 255;
end;

begin{main}
end.
```

DATA SHEET

HARRIS SEMICONDUCTOR

ICL7109
12-Bit μP-Compatible A/D Converter

GENERAL DESCRIPTION

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

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

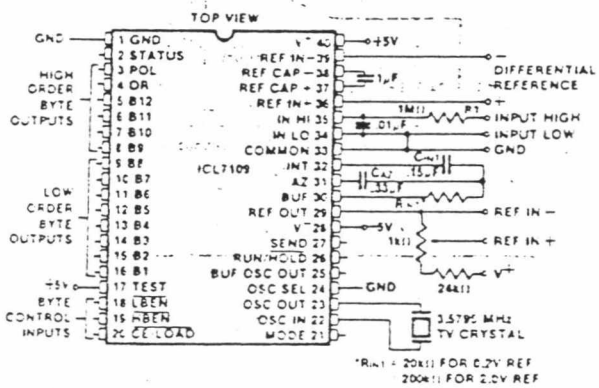
The ICL7109 provides the user with the high accuracy, low noise, low drift, versatility and economy of the dual-slope integrating A/D converter. Features like true differential input and reference, drift of less than 1μV/°C, maximum input bias current of 10pA, and typical power consumption of 20mW make the ICL7109 an attractive per-channel alternative to analog multiplexing for many data acquisition applications.

FEATURES

- 12 Bit Binary (Plus Polarity and Overrange) Dual Slope Integrating Analog-to-Digital Converter
- Byte-Organized TTL-Compatible Three-State Outputs and UART Handshake Mode for Simple Parallel or Serial Interfacing to Microprocessor Systems
- RUN/HOLD Input and STATUS Output Can Be Used to Monitor and Control Conversion Timing
- True Differential Input and Differential Reference
- Low Noise — Typically 15μV p-p
- 1pA Typical Input Current
- Operates At Up to 30 Conversions Per Second
- On-Chip Oscillator Operates With Inexpensive 3.58MHz TV Crystal Giving 7.5 Conversions Per Second for 60Hz Rejection. May Also Be Used With An RC Network Oscillator for Other Clock Frequencies

ORDERING INFORMATION

Part Number	Temp. Range	Package
ICL7109MDL	−55°C to +125°C	40-Pin Ceramic DIP
ICL7109IDL	−25°C to +85°C	40-Pin Ceramic DIP
ICL7109IJL	−25°C to +85°C	40-Pin CERDIP
ICL7109CPL	0°C to 70°C	40-Pin Plastic DIP



(See Figure 2 for typical connection to a UART or Microcomputer)

Figure 1: Pin Configuration and Test Circuit

301655-003

NOTE: All typical values have been characterized but are not tested.

ABSOLUTE MAXIMUM RATINGS

Positive Supply Voltage (GND to V+) ..... +6.2V
Negative Supply Voltage (GND to V−) ..... −9V
Analog Input Voltage (Lo or Hi) (Note 1) ..... V+ to V−
Reference Input Voltage (Lo or Hi) (Note 1) ..... V+ to V−
Digital Input Voltage ..... V− − 0.3V
(Pins 2-27) (Note 2) ..... GND − 0.3V

Power Dissipation (Note 3)
Ceramic Package ..... 1.3W @ +85°C
Plastic Package ..... 500mW @ +70°C
Operating Temperature
Ceramic Package (MDL) ..... −55°C to +125°C
Ceramic Package (IDL) ..... −25°C to +85°C
Plastic Package (CPL) ..... 0°C to +70°C
Storage Temperature ..... −65°C to +150°C
Lead Temperature (Soldering, 10sec) ..... +300°C

NOTE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

($V^+ = +5V$, $V^- = -5V$, GND = 0V, $T_A = 25°C$, $f_{CLK} = 3.58$ MHz, unless otherwise indicated.) Test circuit as shown on first page of this data sheet.

ANALOG SECTION

Symbol	Parameter	Test Conditions	Min	Typ	Max	Unit
	Zero Input Reading	V_{IN} = 0.0000V V_{REF} = 204.8 mV	−0000	±0000	+0000	Counts
	Ratiometric Error(4)	$V_{IN} = V_{REF}$ = 204.8 mV	−3		0	Counts
	Non-Linearity (Max deviation from best straight line fit)	Full Scale = 409.6mV to 2.048V Over full operating temperature range. (Note 4), (Note 6)	−1	±.2	+1	Counts
	Roll-over Error (difference in reading for equal pos. and neg. inputs near full scale)	Full Scale = 409.6mV to 2.048V (Note 5), (Note 6)	−1	±.2	+1	Counts
CMRR	Common Mode Rejection Ratio	V_{CM} = 1V, V_{IN} = 0V Full Scale = 409.6mV		50		µV/V
VCMR	Input Common Mode Range	Input Hi, Input Lo, Common (Note 4)	V− + 1.5		V+ − 1.0	V
e_n	Noise (p-p value not exceeded 95% of time)	V_{IN} = 0V Full Scale = 409.6mV		15		µV
I_{ILK}	Leakage current at Input	V_{IN} = 0 All devices at 25°C ICL7109CPL 0°C ≤ T_A ≤ +70°C (Note 4) ICL7109IDL −25°C ≤ T_A ≤ +85°C (Note 4) ICL7109MDL −55°C ≤ T_A ≤ +125°C		1 20 100 2	10 100 250 5	pA pA pA nA
	Zero Reading Drift	V_{IN} = 0V R_1 = 0Ω (Note 4)		0.2	1	µV/°C
	Scale Factor Temperature Coefficient	V_{IN} = 408.9mV = >7770$_8$ reading Ext. Ref. 0 ppm/°C (Note 4)		1	5	ppm/°C
I^+	Supply Current V+ to GND	V_{IN} = 0, Crystal Osc 3.58MHz test circuit		700	1500	µA
I_{SUPP}	Supply Current V− to V−	Pins 2-21, 25, 26, 27, 29, open		700	1500	µA
V_{REF}	Ref Out Voltage	Referred to V+, 25kΩ between V+ and REF OUT	−2.4	−2.8	−3.2	V
	Ref Out Temp. Coefficient	25kΩ between V+ and REF OUT		80		ppm/°C

NOTE: All typical values have been characterized but are not tested.

ELECTRICAL CHARACTERISTICS ($V^+ = +5V$, $V^- = -5V$, GND = 0V, $T_A = 25°C$, unless otherwise indicated.) Test circuit as shown on first page of this data sheet. (Continued)

DIGITAL SECTION

Symbol	Parameter		Test Conditions	Min	Typ	Max	Unit
V_{OH}	Output High Voltage		$I_{OUT} = 100\mu A$ Pins 2-16, 18, 19, 20	3.5	4.3		V
V_{OL}	Output Low Voltage		$I_{OUT} = 1.6mA$		0.2	0.4	V
	Output Leakage Current		Pins 3-16 high impedance		±.01	±1	μA
	Control I/O Pullup Current		Pins 18, 19, 20 $V_{OUT} = V^+ - 3V$ MODE input at GND		5		μA
	Control I/O Loading		$\overline{HBEN}$ Pin 19 $\overline{LBEN}$ Pin 18 (Note 4)			50	pF
V_{IH}	Input High Voltage		Pins 18-21, 26, 27 referred to GND	3.0			V
V_{IL}	Input Low Voltage		Pins 18-21, 26, 27 referred to GND			1	V
	Input Pull-up Current		Pins 26, 27 $V_{OUT} = V^+ - 3V$		5		μA
	Input Pull-up Current		Pins 17, 24 $V_{OUT} = V^+ - 3V$		25		μA
	Input Pull-down Current		Pin 21 $V_{OUT} = GND + 3V$		5		μA
O_{OH}	Oscillator Output	High	$V_{OUT} = 2.5V$		1		mA
O_{OL}	Current	Low	$V_{OUT} = 2.5V$		1.5		mA
BO_{OH}	Buffered Oscillator	High	$V_{OUT} = 2.5V$		2		mA
BO_{OL}	Output Current	Low	$V_{OUT} = 2.5V$		5		mA
t_W	MODE Input Pulse Width		(Note 4)	50			ns

NOTES: 1. Input voltages may exceed the supply voltages provided the input current is limited to ±100μA.
2. Due to the SCR structure inherent in the process used to fabricate these devices, connecting any digital inputs or outputs to voltages greater than V^+ or less than GND may cause destructive device latchup. For this reason it is recommended that no inputs from sources other than the same power supply be applied to the ICL7109 before its power supply is established, and that in multiple supply systems the supply to the ICL7109 be activated first.
3. This limit refers to that of the package and will not be obtained during normal operation.
4. This parameter is not production tested, but is guaranteed by design.
5. Roll-over error for $T_A = -55°C$ to $+125°C$ is ± 3 counts maximum.
6. A full scale voltage of 2.048V is used because a full scale voltage of 4.096V exceeds the devices Common Mode Voltage Range.
7. For Cerdip package the hadronemic error can be −4 (Min.).

NOTE: All typical values have been characterized but are not tested.

TABLE 1: Pin Assignment and Function Description

Pin	Symbol	Description		
1	GND	Digital Ground, 0V. Ground return for all digital logic.		
2	STATUS	Output High during integrate and deintegrate until data is latched. Output Low when analog section is in Auto-Zero configuration		
3	POL	Polarity — Hi for Positive input		All three state output data bits
4	OR	Overrange — Hi if Overranged.		
5	B12	Bit 12	(Most Significant Bit)	
6	B11	Bit 11		
7	B10	Bit 10		
8	B9	Bit 9		
9	B8	Bit 8		
10	B7	Bit 7	Hi = true	
11	B6	Bit 6		
12	B5	Bit 5		
13	B4	Bit 4		
14	B3	Bit 3		
15	B2	Bit 2		
16	B1	Bit 1	(Least Significant Bit)	
17	TEST	Input High — Normal Operation. Input Low — Forces all bit outputs high. Note: This input is used for test purposes only. Tie high if not used.		
18	$\overline{LBEN}$	Low Byte Enable — With Mode (Pin 21) low, and $\overline{CE/LOAD}$ (Pin 20) low, taking this pin low activates low order byte outputs B1 — B8. — With Mode (Pin 21) high, this pin serves as a low byte flag output used in handshake mode. See Figures 8, 9, 10.		
19	$\overline{HBEN}$	High Byte Enable — With Mode (Pin 21) low, and $\overline{CE/LOAD}$ (Pin 20) low, taking this pin low activates high order byte outputs B9 — B12, POL, OR. — With Mode (Pin 21) high, this pin serves as a high byte flag output used in handshake mode. See Figures 8, 9, 10.		
20	$\overline{CE/LOAD}$	Chip Enable Load — With Mode (Pin 21) low, $\overline{CE/LOAD}$ serves as a master output enable. When high, B1 — B12, POL, OR outputs are disabled. — With Mode (Pin 21) high, this pin serves as a load strobe used in handshake mode. See Figures 8, 9, 10.		

Pin	Symbol	Description
21	MODE	Input Low — Direct output mode where $\overline{CE/LOAD}$ (Pin 20), $\overline{HBEN}$ (Pin 19) and $\overline{LBEN}$ (Pin 18) act as inputs directly controlling byte outputs. Input Pulsed High — Causes immediate entry into handshake mode and output of data as in Figure 10. Input High — Enables $\overline{CE/LOAD}$ (Pin 20), $\overline{HBEN}$ (Pin 19), and $\overline{LBEN}$ (Pin 18) as outputs, handshake mode will be entered and data output as in Figures 8 and 9 at conversion completion.
22	OSC IN	Oscillator Input
23	OSC OUT	Oscillator Output
24	OSC SEL	Oscillator Select — Input high configures OSC IN, OSC OUT, BUF OSC OUT as RC oscillator — clock will be same phase and duty cycle as BUF OSC OUT. — Input low configures OSC IN, OSC OUT for crystal oscillator — clock frequency will be 1/58 of frequency at BUF OSC OUT.
25	BUF OSC OUT	Buffered Oscillator Output
26	RUN/$\overline{HOLD}$	Input High — Conversions continuously performed every 8192 clock pulses. Input Low — Conversion in progress completed, converter will stop in Auto-Zero 7 counts before integrate.
27	SEND	Input — Used in handshake mode to indicate ability of an external device to accept data. Connect to + 5V if not used.
28	V⁻	Analog Negative Supply — Nominally −5V with respect to GND (Pin 1).
29	REF OUT	Reference Voltage Output — Nominally 2.8V down from V⁺ (Pin 40).
30	BUFFER	Buffer Amplifier Output
31	AUTO-ZERO	Auto-Zero Node — Inside foil of C_{AZ}
32	INTEGRATOR	Integrator Output — Outside foil of C_{INT}
33	COMMON	Analog Common — System is Auto-Zeroed to COMMON.
34	INPUT LO	Differential Input Low Side
35	INPUT HI	Differential Input High Side
36	REF IN +	Differential Reference Input Positive
37	REF CAP +	Reference Capacitor Positive
38	REF CAP −	Reference Capacitor Negative
39	REF IN −	Differential Reference Input Negative
40	V⁺	Positive Supply Voltage — Nominally + 5V with respect to GND (Pin 1).

Note: All digital levels are positive true.

NOTE: All typical values have been characterized but are not tested.

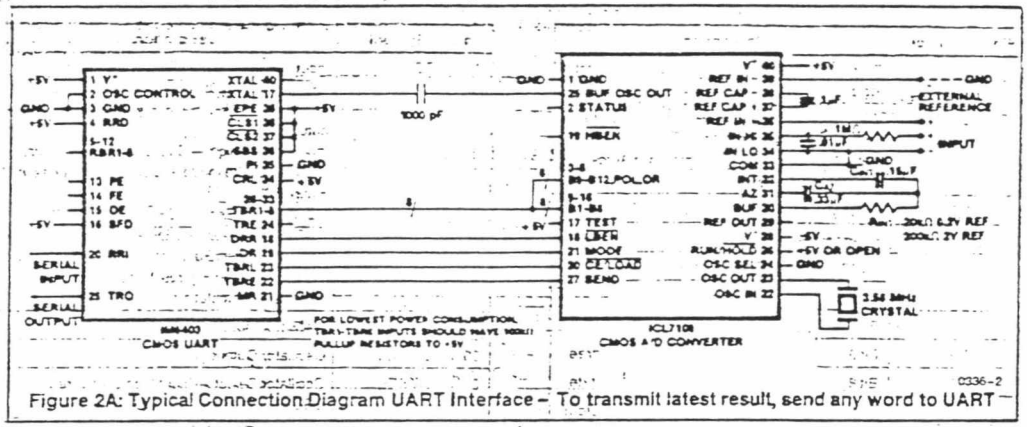


Figure 2A: Typical Connection Diagram UART Interface – To transmit latest result, send any word to UART

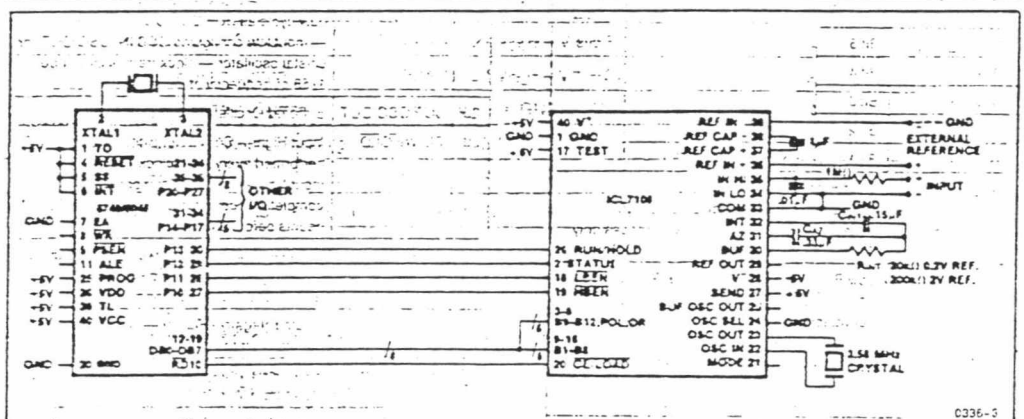


Figure 2B: Typical Connection Diagram Parallel Interface With 8048 Microcomputer

DETAILED DESCRIPTION

Analog Section

Figure 3 shows the equivalent circuit of the Analog Section of the ICL7109. When the RUN/HOLD input is left open or connected to V^+, the circuit will perform conversions at a rate determined by the clock frequency (8192 clock periods per cycle). Each measurement cycle is divided into three phases as shown in Figure 4. They are (1) Auto-Zero (AZ), (2) Signal Integrate (INT) and (3) Deintegrate (DE).

Auto-Zero Phase

During auto-zero three things happen. First, input high and low are disconnected from their pins and internally shorted to analog COMMON. Second, the reference capacitor is charged to the reference voltage. Third, a feedback loop is closed around the system to charge the auto-zero capacitor C_{AZ} to compensate for offset voltages in the buffer amplifier, integrator, and comparator. Since the comparator is included in the loop, the AZ accuracy is limited only by the noise of the system. In any case, the offset referred to the input is less than 10μV.

NOTE: All typical values have been characterized but are not tested.

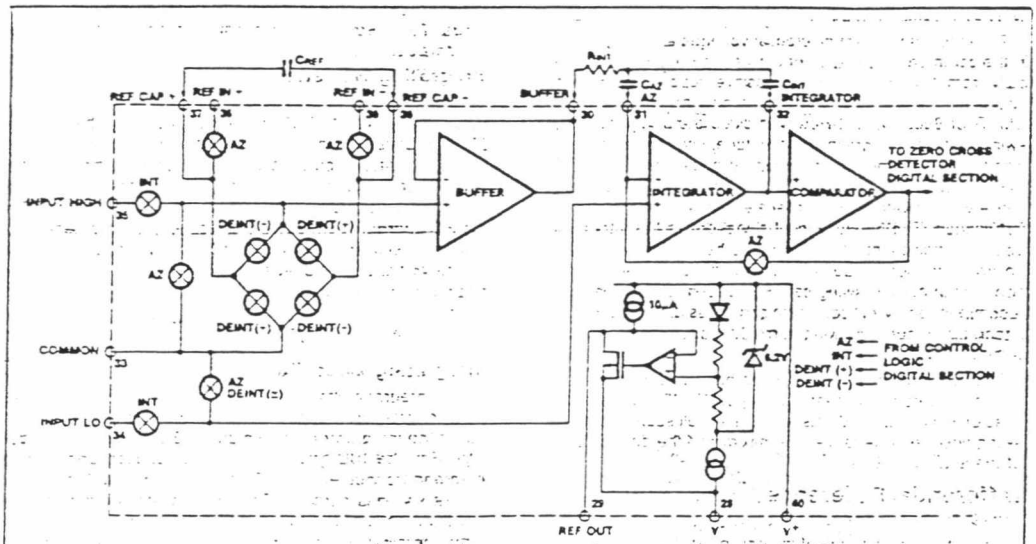


Figure 3: Analog Section

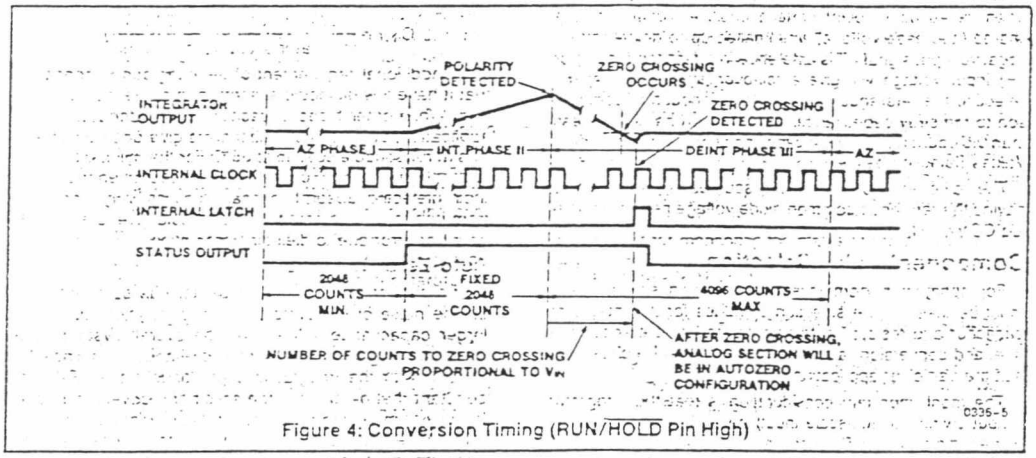


Figure 4: Conversion Timing (RUN/$\overline{HOLD}$ Pin High)

Signal Integrate Phase

During signal integrate the auto-zero loop is opened, the internal short is removed and the internal high and low inputs are connected to the external pins. The converter then integrates the differential voltage between IN HI and IN LO for a fixed time of 2048 clock periods. Note that this differential voltage must be within the common mode range of the inputs. At the end of this phase, the polarity of the integrated signal is determined.

De-Integrate Phase

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

NOTE: All typical values have been characterized but are not tested.

Differential Input

The input can accept differential voltages anywhere within the common mode range of the input amplifier, or specifically from 1.0 volts below the positive supply, to 1.5 volts above the negative supply. In this range the system has a CMRR of 86dB typical. However, since the integrator also swings with the common mode voltage, care must be exercised to assure the integrator output does not saturate. A worst case condition would be a large positive common mode voltage with a near full-scale negative differential input voltage. The negative input signal drives the integrator positive when most of its swing has been used up by the positive common mode voltage. For these critical applications the integrator swing can be reduced to less than the recommended 4V full scale with some loss of accuracy. The integrator output can swing within 0.3 volts of either supply without loss of linearity.

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

Differential Reference

The reference voltage can be generated anywhere within the power supply voltage of the converter. The main source of common mode error is a roll-over voltage caused by the reference capacitor losing or gaining charge to stray capacity on its nodes. If there is a large common mode voltage, the reference capacitor can gain charge (increase voltage) when called up to deintegrate a positive signal but lose charge (decrease voltage) when called up to deintegrate a negative input signal. This difference in reference for (+) or (−) input voltage will give a roll-over error. However, by selecting the reference capacitor large enough in comparison to the stray capacitance, this error can be held to less than 0.5 count for the worst case condition (see Component Values Selection below).

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

Component Value Selection

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

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

Integrating Resistor

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

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

Integrating Capacitor

The integrating capacitor C_{INT} should be selected to give the maximum integrator output voltage swing without saturating the integrator (approximately 0.3 volt from either supply). For the ICL7109 with ±5 volt supplies and analog common connected to GND, a ±3.5 to ±4 volt integrator output swing is nominal. For 7-½ conversions per second (61.72kHz clock frequency) as provided by the crystal oscillator, nominal values for C_{INT} and C_{AZ} are 0.15μF and 0.33μF, respectively. If different clock frequencies are used, these values should be changed to maintain the integrator output voltage swing. In general, the value of C_{INT} is given by

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

An additional requirement of the integrating capacitor is that it have low dielectric absorption to prevent roll-over errors. While other types of capacitors are adequate for this application, polypropylene capacitors give undetectable errors at reasonable cost up to 85°C. For the military temperature range, Teflon® capacitors are recommended. While their dielectric absorption characteristics vary somewhat from unit to unit, selected devices should give less than 0.5 count of error due to dielectric absorption.

Auto-Zero Capacitor

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

For optimal rejection of stray pickup, the outer foil of C_{AZ} should be connected to the R-C summing junction and the inner foil to pin 31. Similarly the outer foil of C_{INT} should be connected to pin 32 and the inner foil to the R-C summing junction. Teflon®, or equivalent, capacitors are recommended above 85°C for their low leakage characteristics.

NOTE: All typical values have been characterized but are not tested.

Reference Capacitor

A 1µF capacitor gives good results in most applications. However, where a large reference common mode voltage exists (i.e. the reference low is not at analog common) and a 409.6mV scale is used, a larger value is required to prevent roll-over error. Generally 10µF will hold the roll-over error to 0.5 count in this instance. Again, Teflon®, or equivalent capacitors should be used for temperatures above 85°C for their low leakage characteristics.

Reference Voltage

The analog input required to generate a full scale output of 4096 counts is $V_{IN} = 2V_{REF}$. Thus for a normalized scale, a reference of 2.048V should be used for a 4.095V full scale, and 204.8mV should be used for a 0.4095V full scale. However, in many applications where the A/D is sensing the output of a transducer, there will exist a scale factor other than unity between the absolute output voltage to be measured and a desired digital output. For instance, in a weighing system, the designer might like to have a full scale reading when the voltage from the transducer is 0.682V. Instead of dividing the input down to 409.6mV, the input voltage should be measured directly and a reference voltage of 0.341V should be used. Suitable values for integrating resistor and capacitor are 33kΩ and 0.15µF. This avoids a divider on the input. Another advantage of this system occurs when a zero reading is desired for non-zero input. Temperature and weight measurements with an offset or tare are examples. The offset may be introduced by connecting the voltage output of the transducer between common and analog high, and the offset voltage between common and analog low, observing polarities carefully. However, in processor-based systems using the ICL7109, it may be more efficient to perform this type of scaling or tare subtraction digitally, using software.

Reference Sources

The stability of the reference voltage is a major factor in the overall absolute accuracy of the converter. The resolution of the ICL7109 at 12 bits is one part in 4096, or 244ppm. Thus if the reference has a temperature coefficient of 80ppm/°C (onboard reference) a temperature difference of 3°C will introduce a one-bit absolute error.

For this reason, it is recommended that an external high-quality reference be used where the ambient temperature is not controlled or where high-accuracy absolute measurements are being made.

The ICL7109 provides a REFerence OUTput (pin 29) which may be used with a resistive divider to generate a suitable reference voltage. This output will sink up to about 20mA without significant variation in output voltage, and is provided with a pullup bias device which sources about 10µA. The output voltage is nominally 2.8V below V+, and has a temperature coefficient of ±80ppm/°C typ. When using the onboard reference, REF OUT (Pin 29) should be connected to REF− (pin 39), and REF+ should be connected to the wiper of a precision potentiometer between REF OUT and V+. The circuit for a 204.8mV reference is shown in the test circuit. For a 2.048mV reference, the fixed resistor should be removed, and a 25kΩ precision potentiometer between REF OUT and V+ should be used.

Note that if pins 29 and 39 are tied together and pins 39 and 40 accidentally shorted (e.g., during testing), the reference supply will sink enough current to destroy the device. This can be avoided by placing a 1kΩ resistor in series with pin 39.

DETAILED DESCRIPTION

Digital Section

The digital section includes the clock oscillator and scaling circuit, a 12-bit binary counter with output latches and TTL-compatible three-state output drivers, polarity, overrange and control logic, and UART handshake logic, as shown in Figure 5.

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

MODE Input

The MODE input is used to control the output mode of the converter. When the MODE pin is low or left open (this input is provided with a pulldown resistor to ensure a low level when the pin is left open), the converter is in its "Direct" output mode, where the output data is directly accessible under the control of the chip and byte enable inputs. When the MODE input is pulsed high, the converter enters the UART handshake mode and outputs the data in two bytes, then returns to "direct" mode. When the MODE input is left high, the converter will output data in the handshake mode at the end of every conversion cycle. (See section entitled "Handshake Mode" for further details).

STATUS Output

During a conversion cycle, the STATUS output goes high at the beginning of Signal Integrate (Phase II), and goes low one-half clock period after new data from the conversion has been stored in the output latches. See Figure 4 for details of this timing. This signal may be used as a "data valid" flag (data never changes while STATUS is low) to drive interrupts, or for monitoring the status of the converter.

RUN/HOLD Input

When the RUN/$\overline{\text{HOLD}}$ input is high, or left open, the circuit will continuously perform conversion cycles, updating the output latches after zero crossing during the Deintegrate (Phase III) portion of the conversion cycle (See Figure 4). In this mode of operation, the conversion cycle will be performed in 8192 clock periods, regardless of the resulting value.

If RUN/$\overline{\text{HOLD}}$ goes low at any time during Deintegrate (Phase III) after the zero crossing has occurred, the circuit will immediately terminate Deintegrate and jump to Auto-Zero. This feature can be used to eliminate the time spent in Deintegrate after the zero-crossing. If RUN/$\overline{\text{HOLD}}$ stays or goes low, the converter will ensure minimum Auto-Zero time, and then wait in Auto-Zero until the RUN/$\overline{\text{HOLD}}$ input goes high. The converter will begin the Integrate (Phase II) portion of the next conversion (and the STATUS output will go high) seven clock periods after the high level is detected at RUN/$\overline{\text{HOLD}}$. See Figure 6 for details.

NOTE: All typical values have been characterized but are not tested.

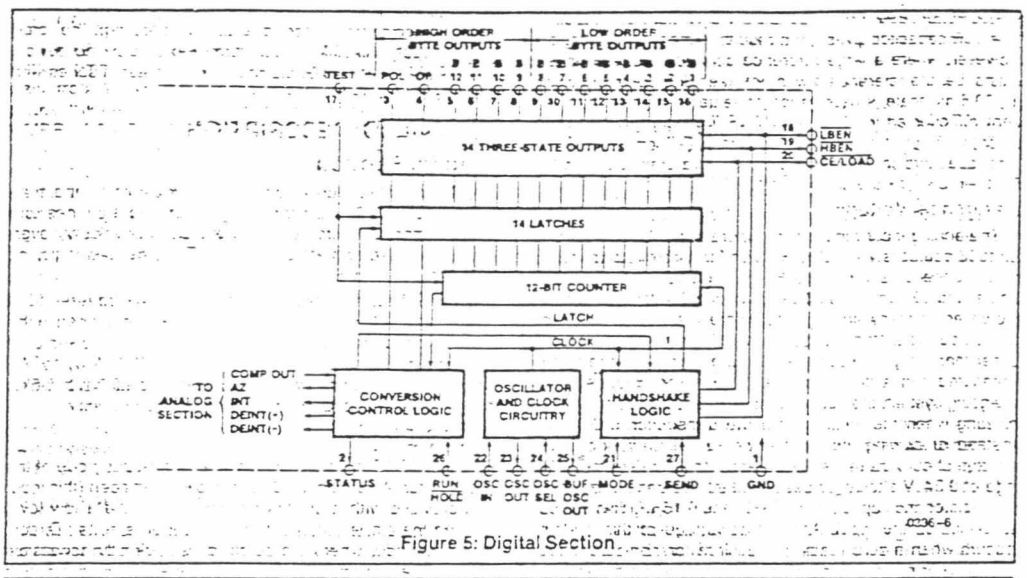


Figure 5: Digital Section

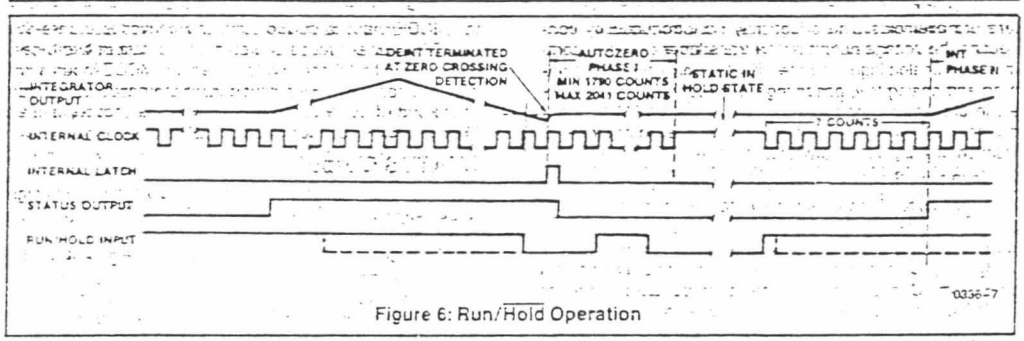


Figure 6: Run/$\overline{\text{Hold}}$ Operation

Using the RUN/$\overline{\text{HOLD}}$ input in this manner allows an easy "convert on demand" interface to be used. The converter may be held at idle in auto-zero with RUN/$\overline{\text{HOLD}}$ low. When RUN/$\overline{\text{HOLD}}$ goes high the conversion is started, and when the STATUS output goes low the new data is valid (or transferred to the UART — see Handshake Mode). RUN/$\overline{\text{HOLD}}$ may now be taken low which terminates deintegrate and ensures a minimum Auto-Zero time before the next conversion.

Alternately, RUN/$\overline{\text{HOLD}}$ can be used to minimize conversion time by ensuring that it goes low during Deintegrate, after zero crossing, and goes high after the hold point is reached. The required activity on the RUN/$\overline{\text{HOLD}}$ input can be provided by connecting it to the Buffered Oscillator Output. In this mode the conversion time is dependent on the input value measured. Also refer to Harris Application Bulletin A032 for a discussion of the effects this will have on Auto-Zero performance.

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

Direct Mode

When the MODE pin is left at a low level, the data outputs (bits 1 through 8 low order byte, bits 9 through 12, polarity and over-range high order byte) are accessible under control of the byte and chip enable terminals as inputs. These three inputs are all active low, and are provided with pullup resistors to ensure an inactive high level when left open. When the chip enable input is low, taking a byte enable input low will allow the outputs of that byte to become active (three-stated on). This allows a variety of parallel data accessing techniques to be used, as shown in the section entitled "Interfacing." The timing requirements for these outputs are shown in Figure 7 and Table 2.

NOTE: All typical values have been characterized but are not tested.

Table 2 — Direct Mode Timing Requirements
(See Note 4 of Electrical Characteristics)

SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNIT
t_{BEA}	Byte Enable Width	350	220		ns
t_{DAB}	Data Access Time from Byte Enable		210	350	ns
t_{DHB}	Data Hold Time from Byte Enable		150	300	ns
t_{CEA}	Chip Enable Width	400	260		ns
t_{DAC}	Data Access Time from Chip Enable		260	400	ns
t_{DHC}	Data Hold Time from Chip Enable		240	400	ns

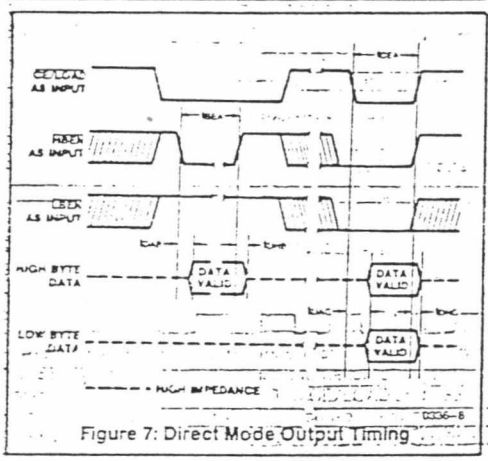


Figure 7: Direct Mode Output Timing

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

Handshake Mode

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

Entry into the handshake mode is controlled by the MODE pin. When the MODE terminal is held high, the ICL7109 will enter the handshake mode after new data has been stored in the output latches at the end of a conversion (See Figures 8 and 9). The MODE terminal may also be used to trigger entry into the handshake mode on demand. At any time during the conversion cycle, the low to high transition of a short pulse at the MODE input will cause immediate entry into the handshake mode. If this pulse occurs while new data is being stored, the entry into handshake mode is delayed until the data is stable. While the converter is in the handshake mode, the MODE input is ignored, and although conversions will still be performed, data updating will be inhibited (See Figure 10) until the converter completes the output cycle and clears the handshake mode.

When the converter enters the handshake mode, or when the MODE input is high, the chip and byte enable terminals become TTL-compatible outputs which provide the control signals for the output cycle (See Figures 8, 9, and 10).

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

Figure 8 shows the sequence of the output cycle with SEND held high. The handshake mode (internal MODE high) is entered after the data latch pulse, and since MODE remains high the $\overline{CE}/LOAD$, $\overline{LBEN}$ and $\overline{HBEN}$ terminals are active as outputs. The high level at the SEND input is sensed on the same high to low internal clock edge that terminates the data latch pulse. On the next low to high internal clock edge the $\overline{CE}/LOAD$ and the $\overline{HBEN}$ outputs assume a low level, and the high-order byte (bits 9 through 12, POL, and OR) outputs are enabled. The $\overline{CE}/LOAD$ output remains low for one full internal clock period only, the data outputs remain active for 1-½ internal clock periods, and the high byte enable remains low for two clock periods. Thus the $\overline{CE}/LOAD$ output low level or low to high edge may be used as a synchronizing signal to ensure valid data, and the byte enable as an output may be used as a byte identification flag. With SEND remaining high the converter completes the output cycle using $\overline{CE}/LOAD$ and $\overline{LBEN}$ while the low order byte outputs (bits 1 through 8) are activated. The handshake mode is terminated when both bytes are sent.

Figure 9 shows an output sequence where the SEND input is used to delay portions of the sequence, or handshake to ensure correct data transfer. This timing diagram shows the relationships that occur using an industry-standard IM6402/3 CMOS UART to interface to serial data channels. In this interface, the SEND input to the ICL7109 is driven by the TBRE (Transmitter Buffer Register Empty) output of the UART, and the $\overline{CE}/LOAD$ terminal of the ICL7109 drives the TBRL (Transmitter Buffer Register Load) input to the UART. The data outputs are paralleled into the eight Transmitter Buffer Register inputs.

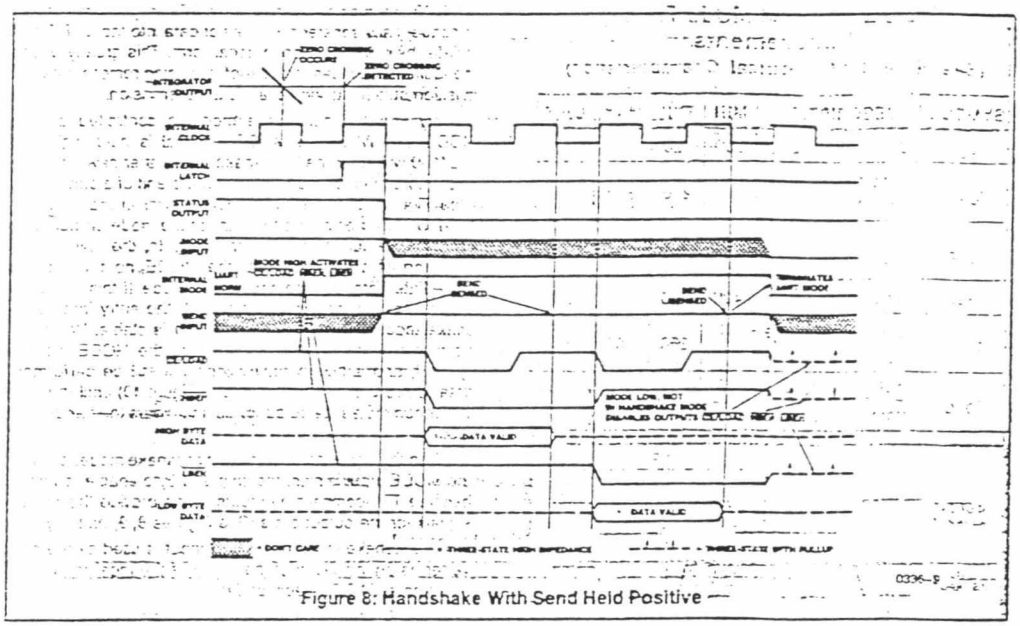


Figure 8: Handshake With Send Held Positive

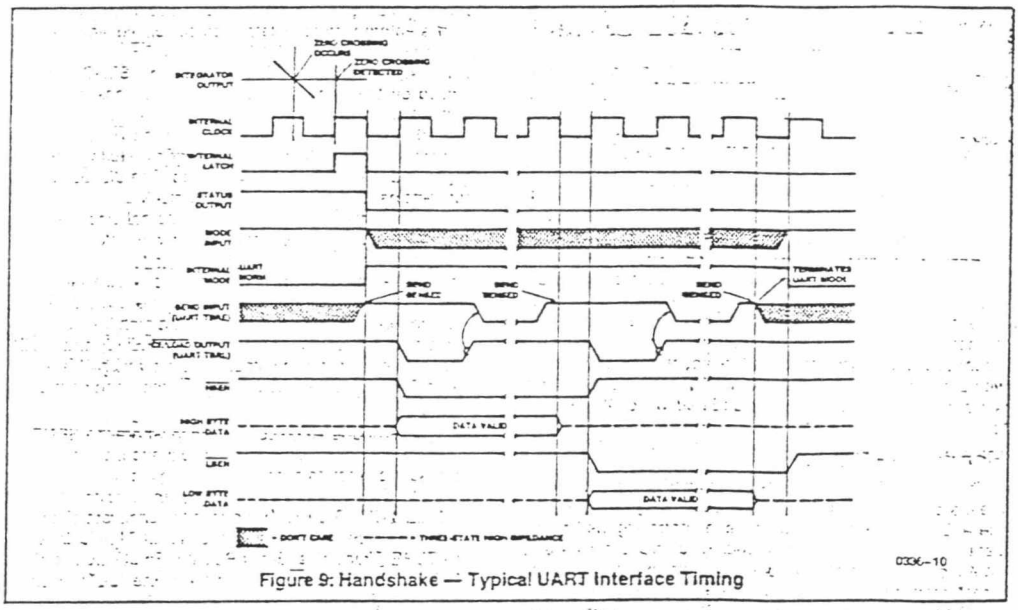


Figure 9: Handshake — Typical UART Interface Timing

NOTE: All typical values have been characterized but are not tested.

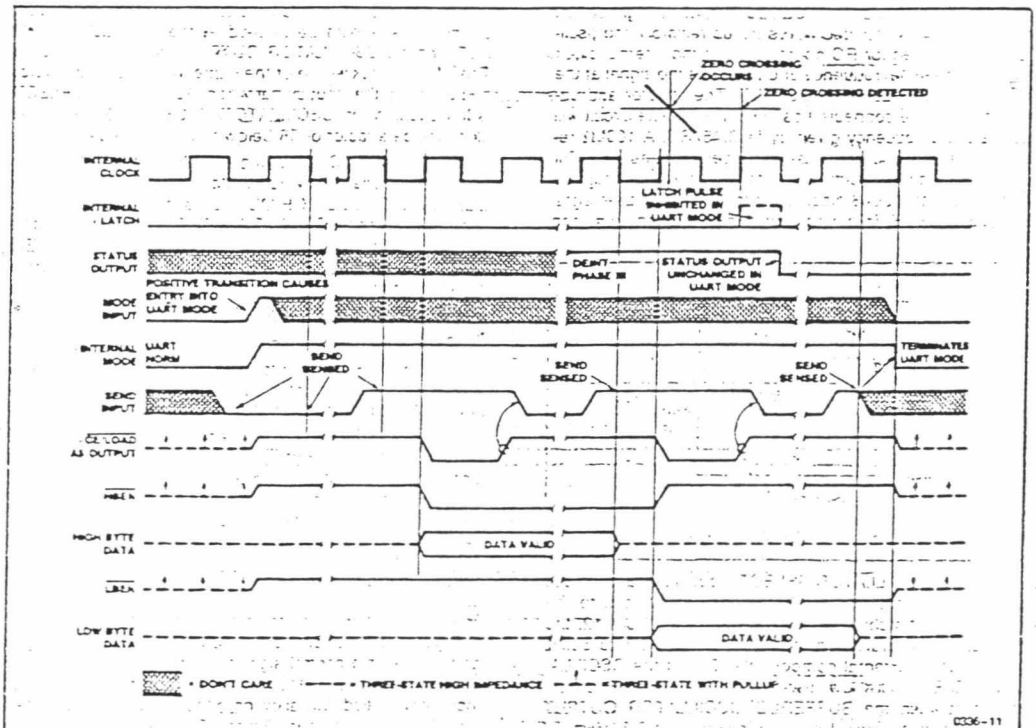


Figure 10: Handshake Triggered By Mode

Assuming the UART Transmitter Buffer Register is empty, the SEND input will be high when the handshake mode is entered after new data is stored. The $\overline{CE/LOAD}$ and $\overline{HBEN}$ terminals will go low after SEND is sensed, and the high order byte outputs become active. When $\overline{CE/LOAD}$ goes high at the end of one clock period, the high order byte data is clocked into the UART Transmitter Buffer Register. The UART TBRE output will now go low, which halts the output cycle with the $\overline{HBEN}$ output low, and the high order byte outputs active. When the UART has transferred the data to the Transmitter Register and cleared the Transmitter Buffer Register, the TBRE returns high. On the next ICL7109 internal clock high to low edge, the high order byte outputs are disabled, and one-half internal clock later, the $\overline{HBEN}$ output returns high. At the same time, the $\overline{CE/LOAD}$ and $\overline{LBEN}$ outputs go low, and the low order byte outputs become active. Similarly, when the $\overline{CE/LOAD}$ returns high at the end of one clock period, the low order data is clocked into the UART Transmitter Buffer Register, and TBRE again goes low. When TBRE returns to a high it will be sensed on the next ICL7109 internal clock high to low edge, disabling the data outputs. One-half internal clock later, the handshake mode will be cleared, and the $\overline{CE/LOAD}$, $\overline{HBEN}$, and $\overline{LBEN}$ terminals return high and stay active (as long as MODE stays high).

With the MODE input remaining high as in these examples, the converter will output the results of every conversion except those completed during a handshake operation. By triggering the converter into handshake mode with a low to high edge on the MODE input, handshake output sequences may be performed on demand. Figure 10 shows a handshake output sequence triggered by such an edge. In addition, the SEND input is shown as being low when the converter enters handshake mode. In this case, the whole output sequence is controlled by the SEND input, and the sequence for the first (high order) byte is similar to the sequence for the second byte. This diagram also shows the output sequence taking longer than a conversion cycle. Note that the converter still makes conversions, with the STATUS output and RUN/HOLD input functioning normally. The only difference is that new data will not be latched when in handshake mode, and is therefore lost.

Oscillator

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

NOTE: All typical values have been characterized but are not tested.

When the OSCILLATOR SELECT input is high or left open (the input is provided with a pullup resistor), the oscillator is configured for RC operation, and the internal clock will be of the same frequency and phase as the signal at the BUFFERED OSCILLATOR OUTPUT. The resistor and capacitor should be connected as in Figure 11. The circuit will oscillate at a frequency given by $f = 0.45/RC$. A 100kΩ resistor is recommended for useful ranges of frequency. For optimum 60Hz line rejection, the capacitor value should be chosen such that 2048 clock periods is close to an integral multiple of the 60Hz period (but should not be less than 50pF).

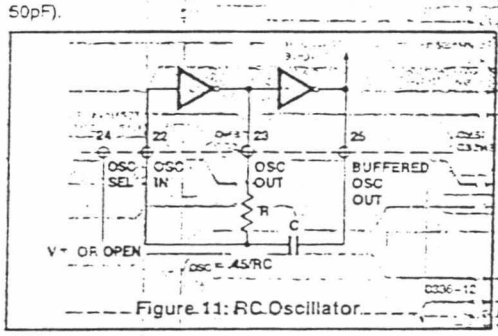


Figure 11: RC Oscillator

When the OSCILLATOR SELECT input is low a feedback device and output and input capacitors are added to the oscillator. In this configuration, as shown in Figure 12, the oscillator will operate with most crystals in the 1 to 5MHz range with no external components. Taking the OSCILLATOR SELECT input low also inserts a fixed ÷58 divider circuit between the BUFFERED OSCILLATOR OUTPUT and the internal clock. Using an inexpensive 3.58MHz TV crystal, this division ratio provides an integration time given by:

$$T_{INT} = (2048 \text{ clock periods}) \times (T_{CLOCK}) = 33.18 \text{ ms}$$

$$\text{where } T_{CLOCK} = \frac{58}{3.58 \text{ MHz}}$$

This time is very close to two 60Hz periods or 33.33ms. The error is less than one percent, which will give better than 40dB 60Hz rejection. The converter will operate reliably at conversion rates of up to 30 per second, which corresponds to a clock frequency of 245.5kHz.

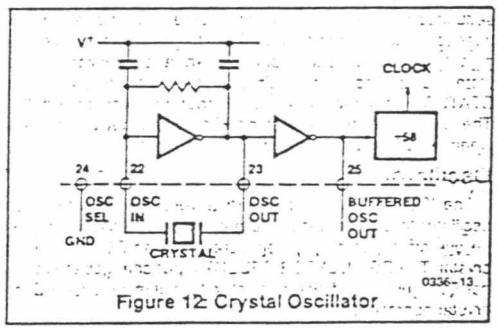


Figure 12: Crystal Oscillator

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

When using the ICL7109 with the TM6403 UART, it is possible to use one 3.58MHz crystal for both devices. The BUFFERED OSCILLATOR OUTPUT of the ICL7109 may be used to drive the OSCILLATOR INPUT of the UART, saving the need for a second crystal. However, the BUFFERED OSCILLATOR OUTPUT does not have a great deal of drive capability, and when driving more than one slave device, external buffering should be used.

Test Input

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

When the TEST input is connected to GND, the counter outputs are all forced into the high state, and the internal clock is disabled. When the input returns to the ½ (V+ – GND) voltage (or to V+) and one clock is applied, all the counter outputs will be clocked to the low state. This allows easy testing of the counter and its outputs.

INTERFACING

Direct Mode

Figure 13 shows some of the combinations of chip enable and byte enable control signals which may be used when interfacing the ICL7109 to parallel data lines. The CE/LOAD input may be tied low, allowing either byte to be controlled by its own enable as in Figure 13A. Figure 13B shows a configuration where the two byte enables are connected together. In this configuration, the CE/LOAD serves as a chip enable, and the HBEN and LBEN may be connected to GND or serve as a second chip enable. The 14 data outputs will all be enabled simultaneously. Figure 13C shows the HBEN and LBEN as flag inputs, and CE/LOAD as a master enable, which could be the READ strobe available from most microprocessors.

Figure 14 shows an approach to interfacing several ICL7109s to a bus, ganging the HBEN and LBEN signals to several converters together, and using the CE/LOAD inputs (perhaps decoded from an address) to select the desired converter.

NOTE: All typical values have been characterized but are not tested.

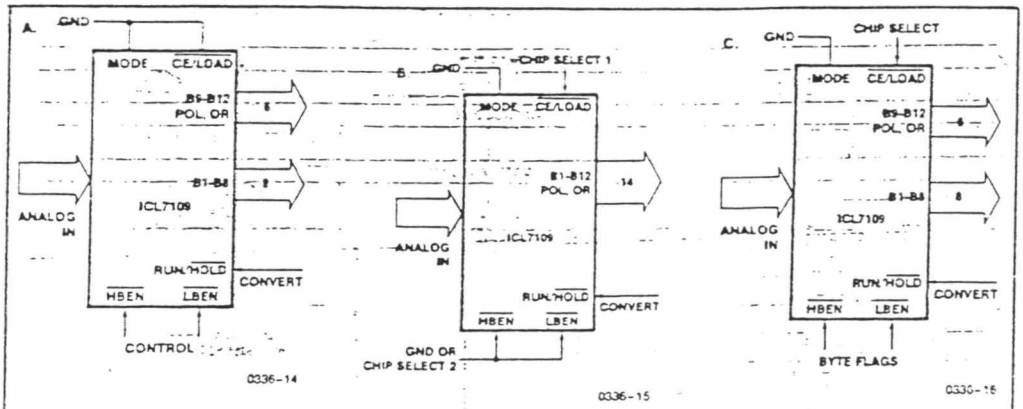


Figure 13: Direct Mode Chip and Byte Enable Combinations

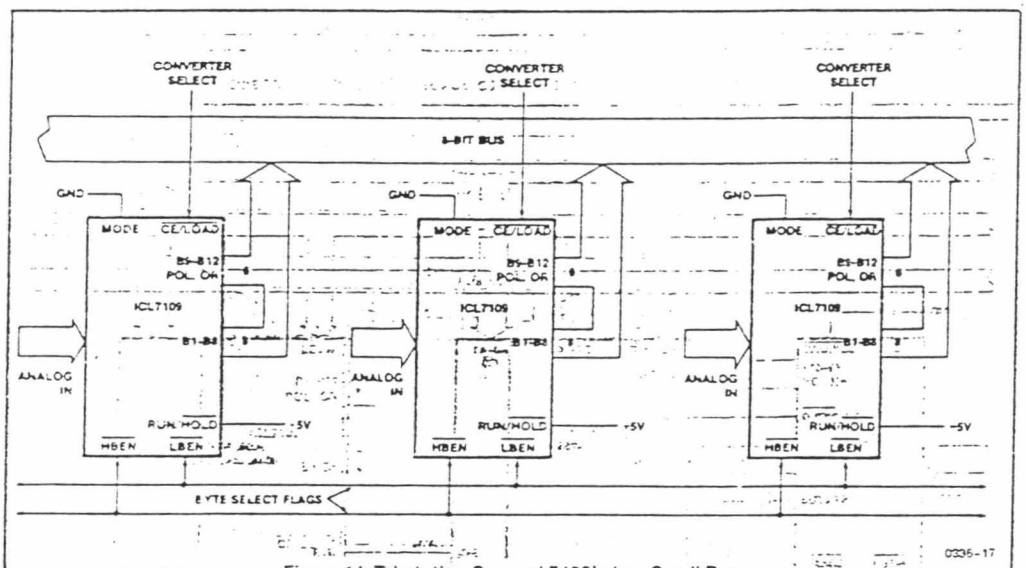


Figure 14: Tri-stating Several 7109's to a Small Bus

Some practical circuits utilizing the parallel three-state output capabilities of the ICL7109 are shown in Figures 15 through 20. Figure 15 shows a straightforward application to the Intel 8048/80/85 microprocessors via an 8255PPI, where the ICL7109 data outputs are active at all times. The I/O ports of an 8155 may be used in the same way. This interface can be used in a read-anytime mode, although a read performed while the data latches are being updated will lead to scrambled data. This will occur very rarely, in the proportion of setup-skew times to conversion time. One way to overcome this is to read the STATUS output as well, and if it is high, read the data again after a delay of more than ½ converter clock period. If STATUS is now low, the second reading is correct, and if it is still high, the first reading is correct. Alternatively, this timing problem is completely avoided by using a read-after-update sequence, as shown in Figure 16. Here the high to low transition of the STATUS output drives an interrupt to the microprocessor causing it to access the data latches. This application also shows the RUN/$\overline{\text{HOLD}}$ input being used to initiate conversions under software control.

A similar interface to Motorola MC6800 or Rockwell R650X systems is shown in Figure 17. The high to low transition of the STATUS output generates an interrupt via the

NOTE: All typical values have been characterized but are not tested.

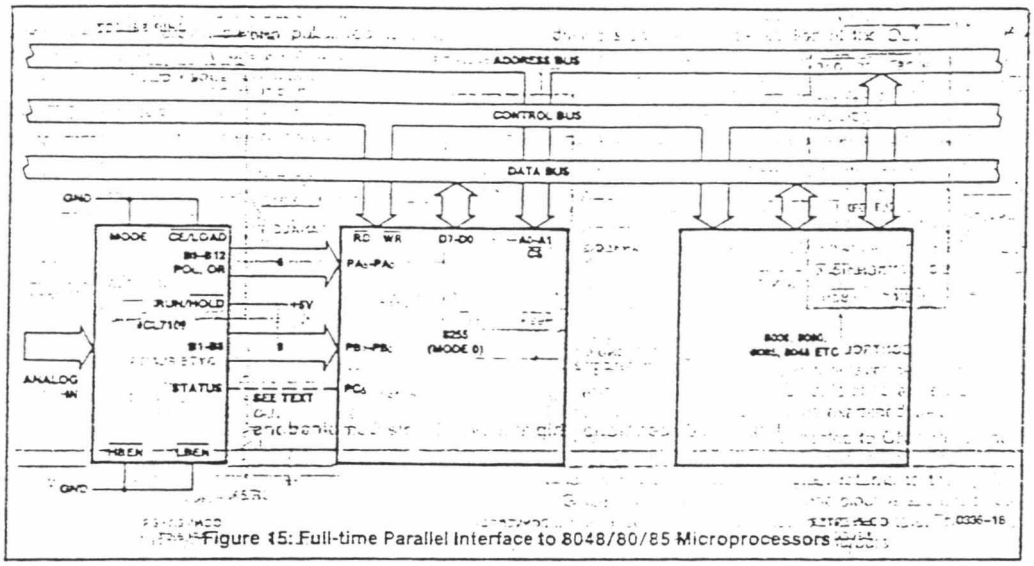


Figure 15: Full-time Parallel Interface to 8048/80/85 Microprocessors

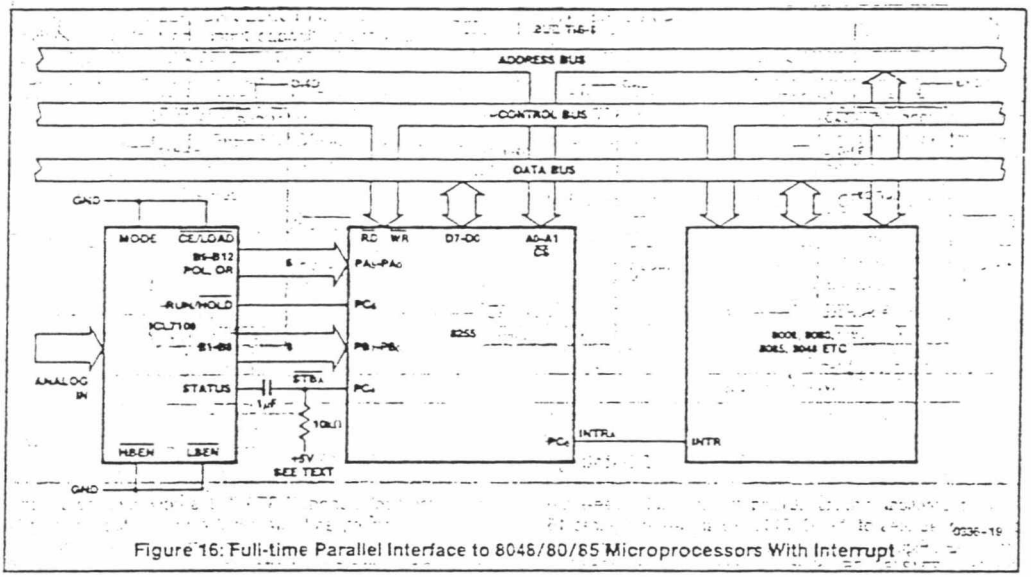


Figure 16: Full-time Parallel Interface to 8048/80/85 Microprocessors With Interrupt

Control Register B CB1 line. Note that CB2 controls the RUN/HOLD pin through Control Register B, allowing software-controlled initiation of conversions in this system as well.

The three-state output capability of the ICL7109 allows direct interfacing to most microprocessor busses. Examples of this are shown in Figures 18 and 19. It is necessary to carefully consider the system timing in this type of interface, to be sure that requirements for setup and hold times, and minimum pulse widths are met. Note also the drive limitations on long buses. Generally this type of interface is only favored if the memory peripheral address density is low so that simple address decoding can be used. Interrupt handling can also require many additional components, and using an interface device will usually simplify the system in this case.

NOTE: All typical values have been characterized but are not tested.

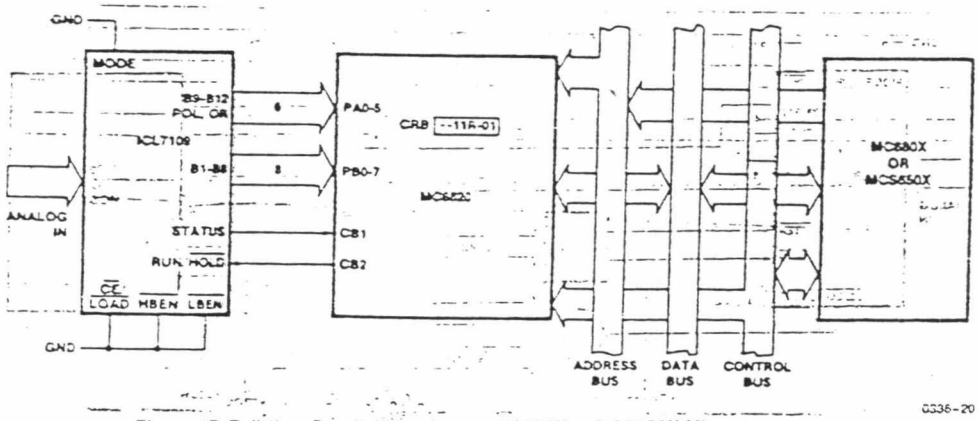


Figure 17: Full-time Parallel Interface to MC680X or MC6500X Microprocessors

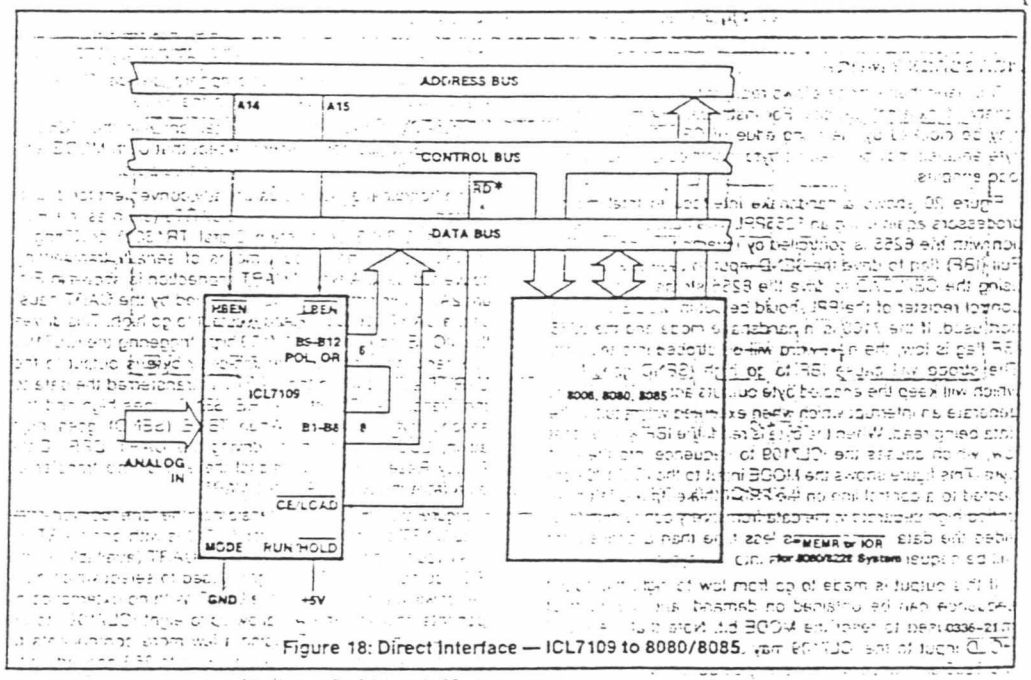


Figure 18: Direct Interface — ICL7109 to 8080/8085.

3

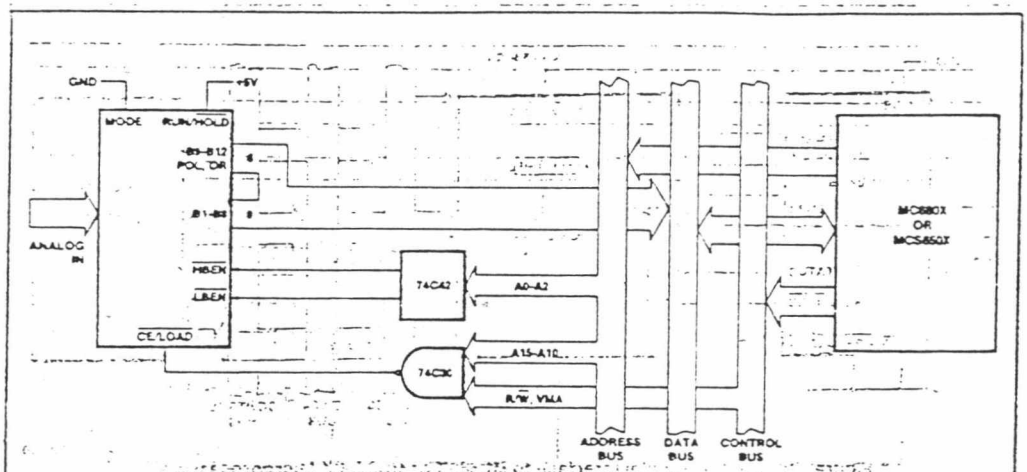


Figure 19. Direct ICL7109 — MC680X Bus Interface

Handshake Mode

The handshake mode allows ready interface with a wide variety of external devices. For instance, external latches may be clocked by the rising edge of CE/LOAD, and the byte enables may be used as byte identification flags or as load enables.

Figure 20 shows a handshake interface to Intel microprocessors again using an 8255PPI. The handshake operation with the 8255 is controlled by inverting its Input Buffer Full (IBF) flag to drive the SEND input to the ICL7109, and using the CE/LOAD to drive the 8255 strobe. The internal control register of the PPI should be set in MODE 1 for the port used. If the 7109 is in handshake mode and the 8255 IBF flag is low, the next word will be strobed into the port. The strobe will cause IBF to go high (SEND goes low), which will keep the enabled byte outputs active. The PPI will generate an interrupt which when executed will result in the data being read. When the byte is read, the IBF will be reset low, which causes the ICL7109 to sequence into the next byte. This figure shows the MODE input to the ICL7109 connected to a control line on the PPI. If this output is left high, or tied high separately, the data from every conversion (provided the data access takes less time than a conversion) will be sequenced in two bytes into the system.

If this output is made to go from low to high, the output sequence can be obtained on demand, and the interrupt may be used to reset the MODE bit. Note that the RUN/HOLD input to the ICL7109 may also be driven by a bit of the 8255 so that conversions may be obtained on command under software control. Note that one port of the 8255 is not used, and can service another peripheral device. The same arrangement can also be used with the 8155.

Figure 21 shows a similar arrangement with the MC6800 or MCS650X microprocessors, except that both MODE and RUN/HOLD are tied high to save port outputs.

The handshake mode is particularly convenient for directly interfacing to industry standard UARTs (such as the Harris IM5402/6403 or Western Digital TR1602) providing a minimum component count means of serially transmitting converted data. A typical UART connection is shown in Figure 2A. In this circuit, any word received by the UART causes the UART DR (Data Ready) output to go high. This drives the MODE input to the ICL7109 high, triggering the ICL7109 into handshake mode. The high order byte is output to the UART first, and when the UART has transferred the data to the Transmitter Register, TBRE (SEND) goes high and the second byte is output. When TBRE (SEND) goes high again, LBEN will go high, driving the UART DRR (Data Ready Reset) which will signal the end of the transfer of data from the ICL7109 to the UART.

Figure 22 shows an extension of the one converter — one UART scheme to several ICL7109s with one UART. In this circuit, the word received by the UART (available at the RBR outputs when DR is high) is used to select which converter will handshake with the UART. With no external components, this scheme will allow up to eight ICL7109s to interface with one UART. Using a few more components to decode the received word will allow up to 256 converters to be accessed on one serial line.

NOTE: All typical values have been characterized but are not tested.

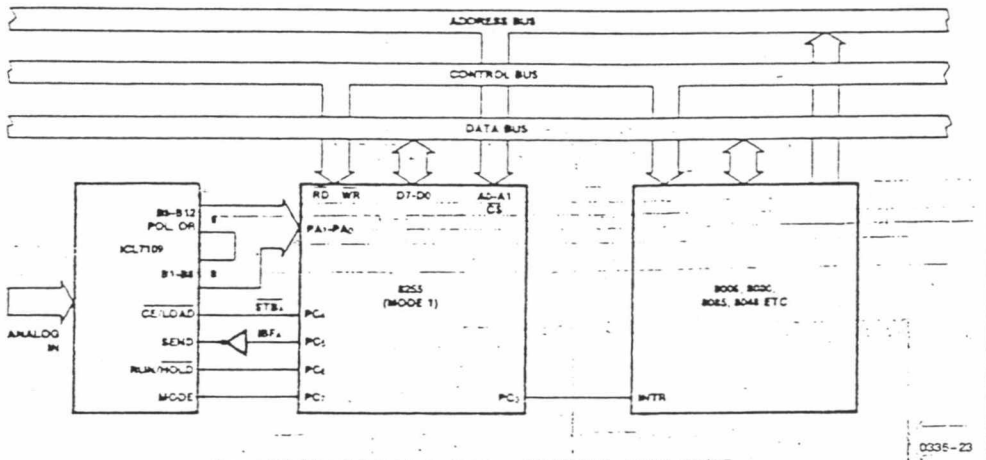


Figure 20: Handshake Interface — ICL7109 to 8048, 80/85

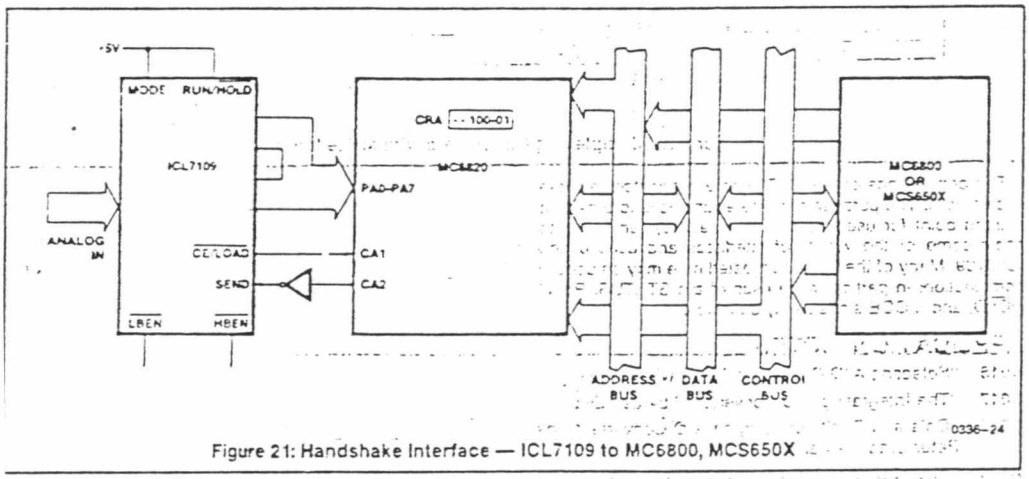


Figure 21: Handshake Interface — ICL7109 to MC6800, MCS650X

3

NOTE: All typical values have been characterized but are not tested.

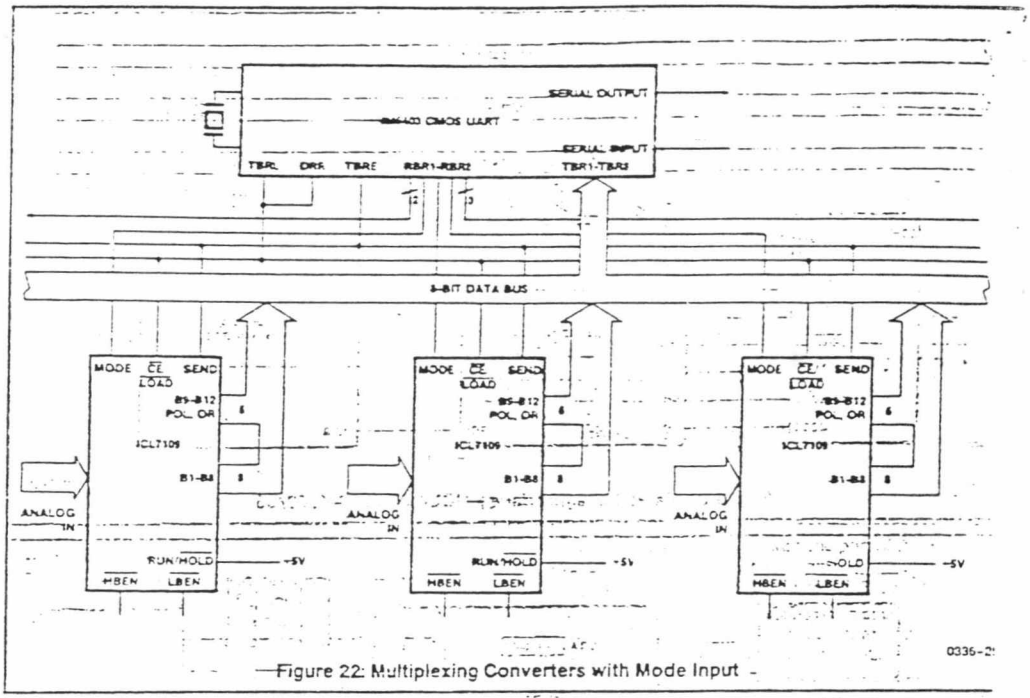


Figure 22: Multiplexing Converters with Mode Input

The applications of the ICL7109 are not limited to those shown here. The purpose of these examples is to provide a starting point for users to develop useful systems, and to show some of the variety of interfaces and uses of the ICL7109. Many of the ideas suggested here may be used in combination, in particular the uses of the STATUS, RUN/HOLD, and MODE signals may be mixed.

APPLICATION NOTES

A016 "Selecting A/D Converters," by David Fullagar

A017 "The Integrating A/D Converters," by Lee Evans

A018 "Do's and Don'ts of Applying A/D Converters," by Peter Bradshaw and Skip Osgood

A030 "The ICL7104 — A Binary Output A/D Converter for Microprocessors," by Peter Bradshaw

A032 "Understanding the Auto-Zero and Common Mode Performance of the ICL7106 Family," by Peter Bradshaw

R005 "Interfacing Data Converters & Microprocessors," by Peter Bradshaw et al, Electronics, Dec. 9, 1976.

NOTE: All typical values have been characterized but are not tested.

ICL7109
INTEGRATING A/D CONVERTER
EQUATIONS

Oscillator Frequency
$f_{OSC} = 0.45/RC$
$C_{OSC} > 50$ pF; $R_{OSC} > 50$ kΩ
f_{OSC} typ. = 60 kHz
or
f_{OSC} = 3.58 MHz Crystal

Oscillator Period
$t_{OSC} = RC/0.45$
$t_{OSC} = 1/3.58$ MHz (Crystal)

Integration Clock Frequency
$f_{CLOCK} = f_{OSC}$ (RC Mode)
$f_{CLOCK} = f_{OSC}/58$ (Crystal)
$t_{CLOCK} = 1/f_{CLOCK}$

Integration Period
$t_{INT} = 2048 \times t_{CLOCK}$

60/50 Hz Rejection Criterion
$t_{INT}/t_{60\ Hz}$ or $t_{INT}/t_{50\ Hz}$ = Integer

Optimum Integration Current
$I_{INT} = 20.0\ \mu A$

Full Scale Analog Input Voltage
V_{INFS} Typically = 200 mV or 2.0V

Integrate Resistor

$$R_{INT} = \frac{V_{INFS}}{I_{INT}}$$

Integrate Capacitor

$$C_{INT} = \frac{(t_{INT})(I_{INT})}{V_{INT}}$$

Integrator Output Voltage Swing

$$V_{INT} = \frac{(t_{INT})(I_{INT})}{C_{INT}}$$

V_{INT} Maximum Swing:
$(V^- - 0.5V) < V_{INT} < (V^+ - 0.5V)$
V_{INT} Typically = 2.0V

Display Count

$$COUNT = 2048 \times \frac{V_{IN}}{V_{REF}}$$

Conversion Cycle
$t_{CYC} = t_{CLOCK} \times 8192$
(In Free-Run Mode, Run/HOLD = 1)
when f_{CLOCK} = 60 kHz, t_{CYC} = 133 ms

Common Mode Input Voltage
$(V^- + 1.0V) < V_{IN} < (V^+ - 0.5V)$

Auto Zero Capacitor
0.01 μF < C_{AZ} < 1.0 μF

Reference Capacitor
0.1 μF < C_{REF} < 1.0 μF

V_{REF}
Biased between V^+ and V^-
$V_{REF} = V^+ - 2.8V$
Regulation lost when V^- to V^- ≤ 6.4V
If V_{REF} is not used, float output pin.

Power Supply: Dual ±5.0V
V^+ = +5.0 to GND
V^- = −5.0 to GND

Output Type:
Binary Amplitude with Polarity and Overrange Bits

Tips: Always tie TEST pin HIGH.
Don't leave any inputs floating.