ไมโครเทอร์มินอล

นาย ณรงค์ แสงแก้ว
นาย ปฐมสรรค ศิริหาญยากร

โครงงานพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาฟิสิกส์ประยุกต์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
2535

MICROTERMINAL

Mr. Narong Saengkaew

Mr. Pathomsak Sirihanyakorn

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

1992

หัวข้อโครงงานพิเศษ	ไมโครเทอร์มินอล
โดย	นาย ณรงค์ แสงแก้ว
	นาย ปฐมสรรค ศิริหาญยากร
ภาควิชา	ฟิสิกส์ประยุกต์
อาจารย์ที่ปรึกษา	อาจารย์ วิชิต ศิริโชติ

ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับโครงงานพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

---------- หัวหน้าภาควิชาฟิสิกส์ประยุกต์

(ดร. เสน่ห์ เอกะวิภาค)

คณะกรรมการโครงงานพิเศษ

---------- ประธานกรรมการ

(ผศ.ดร. ปรีชา เทียนสมประสงค์)

---------- กรรมการ

(ผศ.ดร. บุญส่ง ศิวโมกขธรรม)

---------- กรรมการ

(อาจารย์ วิชิต ศิริโชติ)

หัวข้อโครงงานพิเศษ	ไมโครเทอร์มินอล
นักศึกษา	นาย ณรงค์ แสงแก้ว
	นาย ปฐมสรรค ศิริหาญยากร
อาจารย์ที่ปรึกษา	อาจารย์ วิชิต ศิริโชติ
ภาควิชา	ฟิสิกส์ประยุกต์
ปีการศึกษา	พ.ศ. 2534

บทคัดย่อ

ไมโครเทอร์มินอลออกแบบเพื่อใช้ในการเก็บข้อมูล ข้อมูลจะเก็บบันทึกโดยผ่านทางคีย์บอร์ด และแสดงผลที่ส่วนแสดงผล LCD 16x4 บรรทัด ข้อมูลจะเก็บอยู่ในลักษณะของไฟล์ จำนวนเรคคอร์ดสูงสุดที่เก็บได้คือ 128 เรคคอร์ด ข้อมูลจะส่งผ่านการสื่อสารแบบอนุกรม ตามมาตรฐาน RS 422A ไปยังไมโครคอมพิวเตอร์ โดยมี LOTUS MEASURE เป็นโปรแกรมรับข้อมูล นอกจากนี้ ไมโครเทอร์มินอลยังสามารถบันทึกข้อมูลที่เป็นสัญญาณอนาลอก

Special Project Title	**Microterminal**
Name	**Narong Saengkaew**
	Pathomsak Sirihanyakorn
Spacial Project Advisor	**Wichit Sirichote**
Department	**Applied Physics Ladkrabang**
Acadamic Year	**1991**

Abstract

A microterminal is designed for data storage. Data can be fed into the terminal via a numeric keypad and display on a 16x4 lines LCD display. Data is recorded as sequential files up to 128 records.An EIA RS422A is used to communicate with a PC microcomputer for sending the files to a LOTUS MEASURE software.

In addition , the microterminal can also be recorded an analog voltage.

กิตติกรรมประกาศ

โครงงานพิเศษชิ้นนี้เป็นผลสำเร็จลงได้ ต้องขอขอบคุณ

คุณพ่อ คุณแม่ ผู้เป็นทั้งผู้ให้กำเนิด กำลังใจ ตลอดจนเงินสนับสนุนในด้านต่าง ๆ

อาจารย์ วิชิต ศิริโชติ ผู้ให้คำปรึกษา และจัดหาอุปกรณ์บางอย่างที่ใช้ในโครงงาน

คุณสมภพ ภูริวิกรัยพงศ์ และ คุณไพบูลย์ ศิริะพัฒน์ ในคำแนะนำที่ให้

คุณสมยศ แสงแก้ว ผู้สร้างกล่องของเครื่องไมโครเทอร์มินอล

น้องกี้ ที่พ่นสีของกล่อง ทำให้เกิดความสวยงามมากขึ้น

น้องแอน น้องชัย น้องอู๊ด และน้องซัง ในการพิมพ์ต้นฉบับ

คุณพงศ์สิริ อ่อนศรี จากความอนุเคราะห์ในเครื่องพิมพ์

เพื่อน ๆ ในความช่วยเหลือทั้งปวง

และขอขอบคุณ น้อง ๆ ผู้หญิงทั้งหลาย ที่นำความชุ่มชื่นใจให้กับผู้เขียน

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ 7.4 เมนูหลักของ RS 232 Module 91
รูปที่ 7.5 ตัวอย่างค่าต่าง ๆ ที่ตั้งไว้สำหรับ INKEY Mode 92
รูปที่ 7.6 ตัวอย่างข้อมูลที่รับได้จาก INKEY Mode 93
รูปที่ 7.7 ตัวอย่างข้อมูลที่รับได้จาก ADC Mode 94

บทที่ 1

1.1 บทนำ

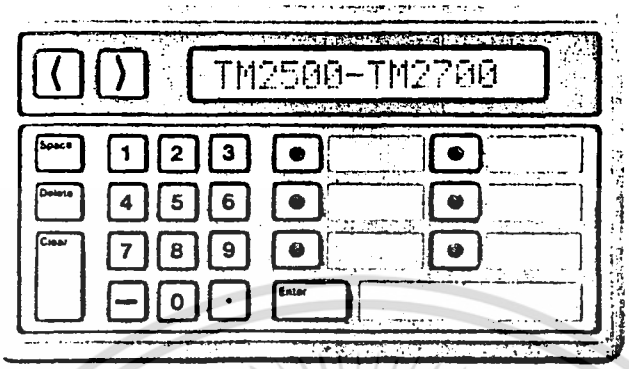
ในอดีตที่ผ่านมา การบันทึกข้อมูลหรือเก็บข้อมูลเพื่อนำไปใช้งานต่าง ๆ เช่น การพล็อตกราฟ มนุษย์จะเป็นผู้ดำเนินการเองทั้งหมด ซึ่งถ้าในกรณีที่ข้อมูลนั้นมีจำนวนมาก การเก็บข้อมูลและการพล็อตกราฟจะทำได้ลำบาก เสียเวลา และยังผิดพลาดได้ง่าย

จากความยากลำบากดังกล่าว ทำให้มีผู้สร้างอุปกรณ์ในการจัดเก็บ และประมวลผลข้อมูล ในลักษณะต่าง ๆ เช่น Data logger เป็นตัวเก็บข้อมูลจากตัว Sensor โดยตรง แล้วนำข้อมูลที่ได้มาประมวลผลโดย เครื่องไมโครคอมพิวเตอร์ เครื่องไมโครเทอร์มินอลเป็นตัวเก็บข้อมูลอีกลักษณะหนึ่ง ซึ่งผู้ใช้ต้องป้อนข้อมูลที่ต้องการเก็บผ่านคีย์บอร์ด ตัวเก็บข้อมูลชนิดนี้จะมีจอภาพแสดงข้อมูลที่เราจัดเก็บ และยังสามารถติดต่อกับเครื่องไมโครคอมพิวเตอร์ เพื่อนำข้อมูลนั้นมาแสดงผลในลักษณะกราฟ เป็นต้น ตัวเก็บข้อมูลดังกล่าวมีผู้ผลิตออกขายในลักษณะต่าง ๆ ผู้ใช้สามารถเลือกใช้ให้เหมาะสมกับงานของตนเอง โดยส่วนใหญ่แล้วจะเป็นผลิตภัณฑ์จากต่างประเทศ เช่น เครื่องไมโครเทอร์มินอลของ บริษัท เบอร์บราวน์ (Burr Brown) ดังรูปที่ 1.1 ซึ่งคุณสมบัติของเครื่องไมโครเทอร์มินอลของบริษัท เบอร์บราวน์ มีดังต่อไปนี้

- ส่วนแสดงผล LCD ขนาดใหญ่ คมชัด แบบ 16 ตัวอักษร
- ขนาดเล็ก 4.102" x 7.102" x 1.060"
- น้ำหนัก 10.5 ออนซ์
- อัตราการส่งข้อมูล 300,1200,9600
- การสื่อสารใช้มาตราฐาน RS 422A สามารถส่งได้ไกลถึง 1 กิโลเมตร
- คีย์บอร์ด 24 คีย์

BURR-BROWN®

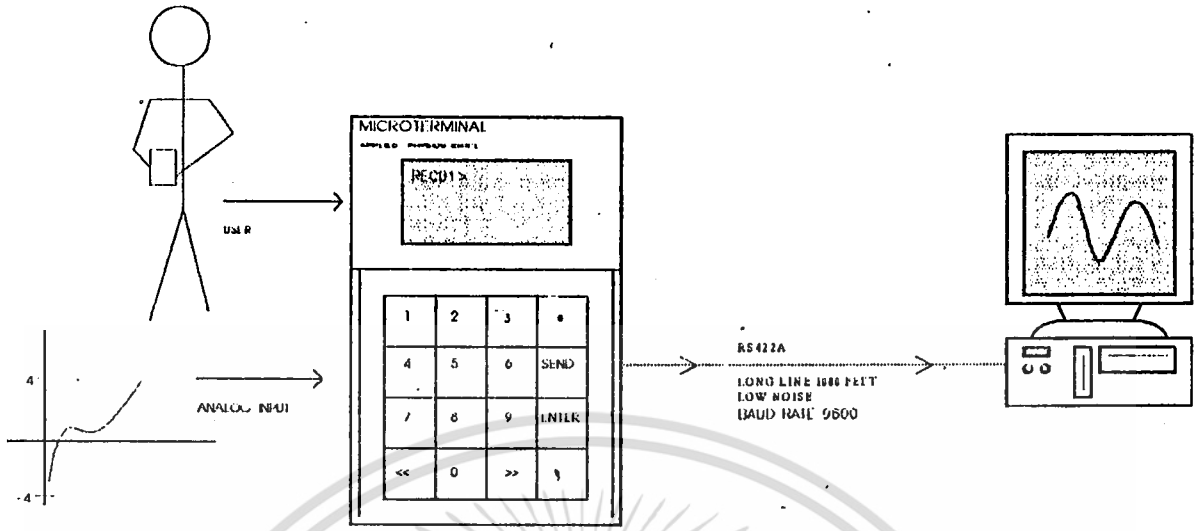
TM2500
TM2700



รูปที่ 1.1 เครื่องไมโครเทอร์มินอลของบริษัท เบอร์บราว์น

จากคุณสมบัติดังกล่าว ทำให้ผู้ใช้ได้รับความสะดวกในการใช้งาน แต่ก็มีราคาสูงตามคุณภาพด้วยเช่นกัน จากลักษณะและความสามารถของเครื่องไมโครเทอร์มินอลของ บริษัท เบอร์บราว์น ทำให้เกิดแรงดลใจในการที่จะสร้างเครื่องไมโครเทอร์มินอลให้มีคุณสมบัติเหมือนหรือใกล้เคียง กับเครื่องจากต่างประเทศ เพื่อเป็นแนวทางในการผลิตเครื่องมือขึ้นใช้เองในประเทศ และยังเป็นการลดการขาดดุลทางการค้ากับต่างประเทศอีกด้วย

ไมโครเทอร์มินอล(Microterminal) ในโครงงานพิเศษนี้เป็นเครื่องมือใช้บันทึกข้อมูล หรือเก็บข้อมูลจากสัญญาณอนาลอก โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมการทำงาน ข้อมูลต่าง ๆ ที่นำมาบันทึกจะแสดงผลทางจอภาพให้เห็นได้ทันที และยังสามารถส่งผ่านข้อมูลขึ้นไป ที่เครื่องคอมพิวเตอร์เพื่อช่วยในการพล็อตกราฟได้ รูปแบบโครงสร้างการทำงานแสดงได้ ดังรูป 1.2



รูปที่ 1.2 รูปแบบโครงสร้างของระบบ

ผู้ใช้สามารถบันทึกข้อมูลได้โดยการกดคีย์บอร์ดบน ไมโครเทอร์มินอล ข้อมูลจะแสดงผลให้เห็นทางจอภาพแสดงผล LCD ซึ่งมีขนาดใหญ่ แสดงตัวอักษรได้ 4.8 ตัว สามารถแก้ไขข้อมูล และเก็บลงหน่วยความจำได้ นอกจากนี้ ไมโครเทอร์มินอล ยังเก็บข้อมูลที่เป็นสัญญาณทางอนาลอกได้โดยตรง ข้อมูลจะส่งผ่านทางสายสัญญาณตามมาตรฐาน RS422A ซึ่งมีคุณสมบัติ คือ สามารถส่งผ่าน ข้อมูลได้ไกลถึง 1 กิโลเมตร ในอัตรา 9600 บิทต่อวินาที ข้อมูลจะส่งเข้าสู่ไมโครคอมพิวเตอร์ที่โปรแกรมสำเร็จรูป LOTUS MEASURE ซึ่งเป็นโปรแกรมที่ช่วยในการเขียนกราฟ ดังนั้นการเก็บข้อมูลและการพล๊อตกราฟจึงสามารถทำได้รวดเร็วกว่าในสมัยก่อนมาก อีกทั้งยังปราศจากข้อผิดพลาดในการเก็บข้อมูลและพล๊อตกราฟ

1.2 วัตถุประสงค์

1. เพื่อนำไปใช้ในการรับข้อมูลที่เป็นสัญญาณอนาลอก หรือ ข้อมูลที่บันทึกผ่านทางคีย์บอร์ด และส่งไปยังไมโครคอมพิวเตอร์เพื่อทำการวิเคราะห์

2. เพื่อนำไปใช้ในการสื่อสารข้อมูลแบบอนุกรมที่ตัวส่งอยู่ไกลจากตัวรับ และอยู่ในสภาวะที่มีสัญญาณรบกวน

1.3 ขอบเขตการดำเนินงาน

1. เชื่อมต่อไมโครคอนโทรเลอร์ 8032 เข้ากับจอภาพแสดงผล LCD และคีย์บอร์ด
2. สร้างวงจรแปลงสัญญาณอนาลอกไปเป็นสัญญาณดิจิตอล
3. สร้างวงจรติดต่อสื่อสารข้อมูลแบบอนุกรมระยะไกลตามมาตรฐาน RS422A
4. พัฒนาโปรแกรมภาษาเบสิกควบคุมการทำงานของระบบ
5. ส่งข้อมูลผ่านเข้าไมโครคอมพิวเตอร์โดยใช้โปรแกรม LOTUS MEASURE เป็นตัวรับ

บทที่ 2
การสื่อสารข้อมูลทางคอมพิวเตอร์

2.1 การสื่อสารข้อมูล

การสื่อสารข้อมูล คือ การส่งถ่ายข้อมูลจากแหล่งข้อมูลหนึ่งไปสู่อีกแหล่งข้อมูลหนึ่ง โดยอาศัยตัวกลางเป็นตัวส่งผ่านข้อมูล หรือที่เรียกว่า ช่องสัญญาณ (Transmission link) ดังแผนภูมิข้างล่าง

รูปที่ 2.1 รูปแสดงการสื่อสารข้อมูลผ่านตัวกลาง

การสื่อสารอาจแบ่งได้เป็น 2 ประเภทใหญ่ ๆ ตามลักษณะของตัวกลางหรือชองสัญญาณ ดังนี้

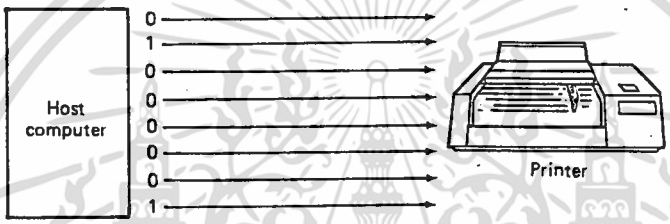
1) การสื่อสารวิทยุ เป็นการส่งผ่านข้อมูลโดยอาศัยคลื่นแม่เหล็กไฟฟ้าเป็นตัวส่งผ่านสัญญาณไปทางบรรยากาศ

2) การสื่อสารตามสาย เช่น การใช้สายไฟเป็นตัวกลางในการส่งผ่านสัญญาณหรือข้อมูล

2.2 การสื่อสารข้อมูลตามสายแบบต่าง ๆ

2.2.1 การสื่อสารข้อมูลแบบขนาน

เป็นการส่งผ่านข้อมูลโดยอาศัยสายในการส่งผ่านข้อมูลจำนวนหลายเส้น โดยข้อมูลแต่ละเส้นจะถูกส่งจากเครื่องภายในเวลาเดียวกัน เช่น การส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์จะใช้สายจำนวน 8 เส้น โดยแต่ละเส้นจะแทนข้อมูล 1 บิต ดังรูปที่แสดงข้างล่าง



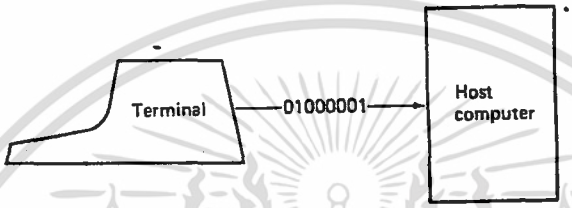
รูปที่ 2.2 การสื่อสารข้อมูลแบบขนาน

นอกจากนี้อาจจะมีสายควบคุมเพิ่มเข้ามาอีกก็ได้ เช่น บิตพาริตี้(Parity)ของสัญญาณ เพื่อเป็นการตรวจสอบความผิดพลาดของข้อมูล

ระยะทางระหว่างเครื่องส่งและเครื่องรับไม่ควรจะเกิน 100 ฟุต เพราะจะเกิดปัญหาสัญญาณสูญหายไปกับความต้านทานของสาย และระดับกราวนด์ในทางไฟฟ้าที่จุดรับผิดไปจากจุดส่ง ทำให้เกิดความผิดพลาดในการรับส่ง

2.2.2 การสื่อสารข้อมูลแบบอนุกรม

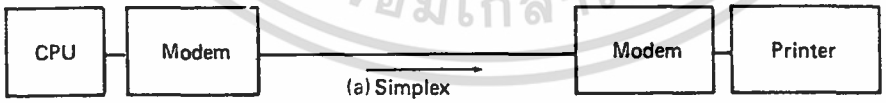
ข้อมูลถูกส่งออกมาทีละบิตระหว่างจุดส่งและจุดรับ จะเห็นว่าการสื่อสารข้อมูลแบบอนุกรมนี้จะช้ากว่าการสื่อสารข้อมูลแบบขนาน และใช้จำนวนสายในการสื่อสารเพียงเส้นเดียวหรือเพียงคู่เดียว ทำให้ค่าใช้จ่ายถูกกว่าการสื่อสารข้อมูลแบบขนาน ในกรณีที่ต้องการสื่อสารระยะทางไกล ๆ แสดงได้ดังรูป



รูปที่ 2.3 การสื่อสารข้อมูลแบบอนุกรม

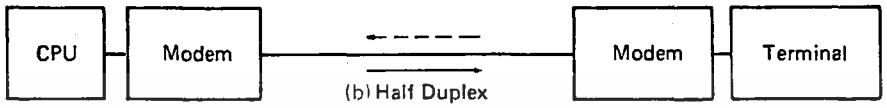
การสื่อสารแบบอนุกรมแบ่งตามรูปลักษณะได้ 3 แบบ คือ

1. แบบซิมเพล็กซ์ (Simplex communication) ข้อมูลจะส่งได้ในทิศทางเดียวเท่านั้น บางครั้งอาจเรียกว่า การส่งทิศทางเดียว (Unidirectional data bus)



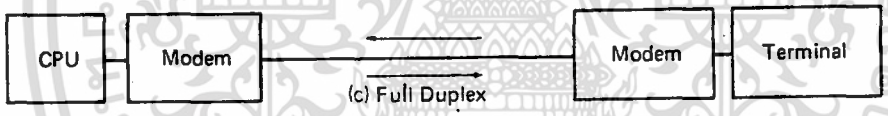
รูปที่ 2.4 การสื่อสารแบบซิมเพล็กซ์

2. แบบฮาล์ฟดูเพล็กซ์ (Half duplex communication) ข้อมูลจะสามารถส่งได้ทั้งสองสถานี แต่จะต้องผลัดกันส่งและผลัดกันรับ จะส่งและรับพร้อมกันไม่ได้



รูปที่ 2.5 การสื่อสารแบบฮาล์ฟดูเพล็กซ์

3. แบบฟูลดูเพล็กซ์ (Full duplex communication) ข้อมูลจะสามารถส่งและรับจากสถานีได้ในเวลาเดียวกัน

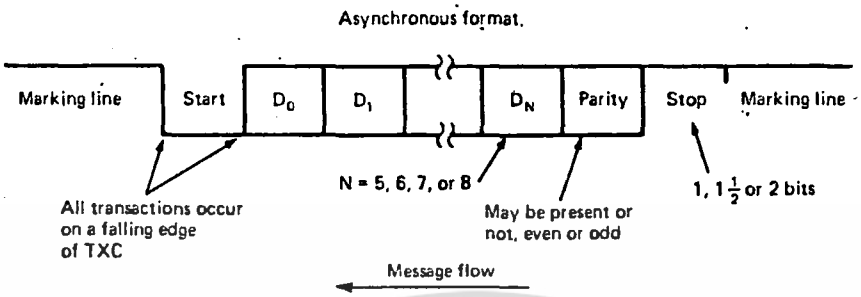


รูปที่ 2.6 การสื่อสารแบบฟูลดูเพล็กซ์

นอกจากนี้ การสื่อสารข้อมูลแบบอนุกรมยังสามารถจำแนกรูปแบบในการส่งข้อมูลออกเป็น

1. การสื่อสารแบบอะซิงโครนัส (Asynchronous transmission)

ลักษณะของสัญญาณจะเป็นดังรูป



รูปที่ 2.7 ลักษณะสัญญาณของการสื่อสารแบบอะซิงโครนัส

เมื่อเริ่มส่งข้อมูล สัญญาณของอะซิงโครนัสจะเป็น " ลอจิก 0 " หนึ่งช่วงสัญญาณนาฬิกา บิตนี้จะเรียกว่า บิตเริ่มต้น หรือ Start bit ตามหลังจากบิตเริ่มต้นจะเป็นข้อมูลซึ่งมีขนาดตั้งแต่ 5 บิตถึง 8 บิต โดยบิตนัยสำคัญต่ำ หรือ LSB จะถูกส่งออกมาก่อนไล่ไปจนถึงบิตนัยสำคัญสูง หรือ MSB และอาจจะมีบิตที่เป็นตัวตรวจสอบความถูกต้องของข้อมูลหรือที่เรียกว่า บิตพาริตี้ (Parity bit) ตามมาก็ได้ หลังจากบิตพาริตี้ก็จะเป็นบิตสิ้นสุด (Stop bit) ความกว้างของบิตสิ้นสุดอาจจะเป็น 1,1.5 หรือ 2 สัญญาณนาฬิกาก็ได้ ดังนั้นการส่งข้อมูลแบบนี้จะต้องตั้งค่าต่าง ๆ ได้แก่

1 ความเร็วในการส่ง

2 ความยาวของข้อมูล

3 บิตตรวจสอบ

4 ความกว้างของบิตสิ้นสุด (Stop bit)

ข้อดีของการสื่อสารแบบนี้คือ จะไม่ขึ้นอยู่กับสัญญาณนาฬิกา หมายถึงบิตเริ่มต้นของแต่ละไบต์จะเริ่มขึ้นที่เวลาใดก็ได้

2. การสื่อสารแบบซิงโครนัส (Synchronous transmission) ในการสื่อสารแบบนี้ ข้อมูลจะส่งออกมาแบบต่อเนื่อง จะไม่มี บิตเริ่มต้น, บิตพาริตี้ และบิตสิ้นสุดเลย โดยข้อมูลทั้งหมดจะมีไบต์ควบคุมปิดด้านหัวและด้านท้ายของชุดข้อมูล เรียกว่า การซิงโครไนส์ (Synchronization) หรือ Syn bytes ดังรูป

รูปที่ 2.8 ลักษณะสัญญาณของการสื่อสารแบบซิงโครนัส

Syn bytes จะเป็นตัวบอกให้กับสถานีรับข้อมูลว่า ข้อมูลเริ่มต้นและจบลงตรงที่ใด

2.3 อุปกรณ์ที่ใช้ในการสื่อสารแบบอนุกรม

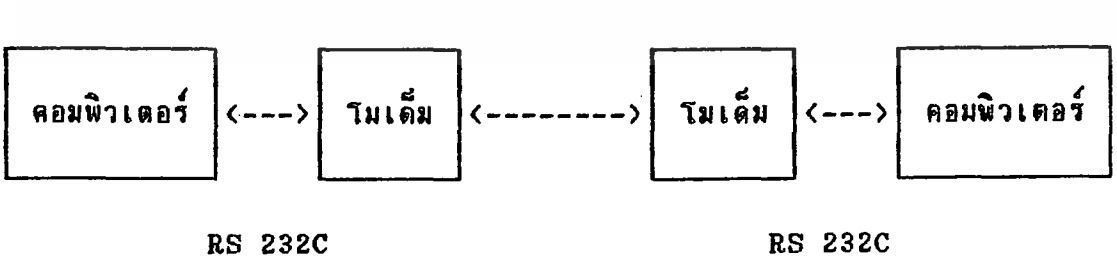
ในการติดต่อสื่อสารกับไมโครคอมพิวเตอร์ มักจะใช้การสื่อสารข้อมูลแบบอะซิงโครนัส ดังที่กล่าวในบทที่แล้ว ในบทนี้จะขอกล่าวถึงอุปกรณ์ที่ใช้ในการสื่อสารแบบนี้

2.3.1 มาตรฐาน RS 232C

โดยปกติในไมโครคอมพิวเตอร์จะมีพอร์ตที่เป็นแบบอนุกรม หรือ ที่เรียกกันว่า RS 232C อยู่ในตัวเอง พอร์ต RS 232C มีหน้าที่รับและส่งข้อมูลในแบบอนุกรม ความจริงมาตรฐานในการส่งข้อมูลแบบอนุกรมมีหลายมาตรฐาน แต่ที่นิยมใช้กันมากที่สุดสำหรับไมโครคอมพิวเตอร์ ก็คือ มาตรฐาน RS 232C

มาตรฐาน RS 232C ได้จัดพิมพ์ขึ้นเมื่อปี ค.ศ.1969 โดยสมาคมผู้ผลิตอุปกรณ์อิเลกทรอนิกส์แห่งสหรัฐอเมริกา หรือ The Electronic Industries Association (EIA) คำว่า RS ย่อมาจาก Recomended Standard ส่วนเลข 232 เป็นตัวที่บ่งบอกถึงมาตรฐานตัวนี้ ส่วน C เป็นหมายเลขของฉบับท้ายสุดของมาตรฐานตัวนี้ จุดประสงค์ของมาตรฐานตัวนี้ เพื่อการเชื่อมต่อระหว่างอุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment DTE)กับอุปกรณ์สื่อสารข้อมูล(Data Communication Equipment DCE)

ตัวอย่างอุปกรณ์ประเภท DTE และ DCE คือในการติดต่อระหว่างคอมพิวเตอร์ 2 ตัวผ่านสายโทรศัพท์ ซึ่งจำเป็นที่จะต้องมีโมเด็ม (Modem) เพื่อเป็นตัวฝากข้อมูล ในที่นี้ DTE ก็คือ คอมพิวเตอร์ และ DCE ก็คือ โมเด็ม ดังรูป



รูปที่ 2.9 ตัวอย่างของอุปกรณ์ประเภท DTE และ DCE

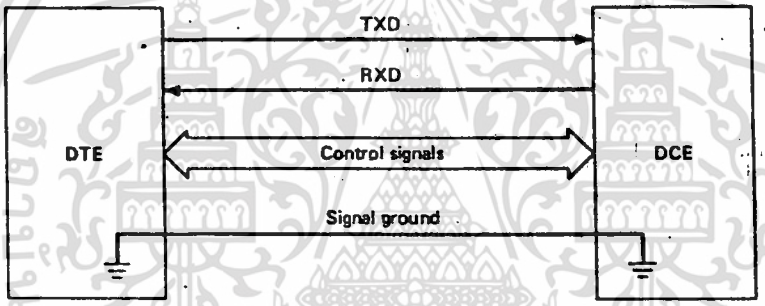
RS 232C สามารถเชื่อมต่อการถ่ายข้อมูลได้จาก 0 ถึง 20,000 บิตต่อวินาที ซึ่งเพียงพอสำหรับไมโครคอมพิวเตอร์ที่มีอัตราบอด (Baud rate)ตั้งแต่ 110 ถึง 9600 แต่ความยาวของสายเชื่อมต่อจำกัดอยู่ไม่เกิน 50 ฟุต ถ้าระยะทางในการโอนถ่ายข้อมูลยาวกว่า 50 ฟุต จะเกิดการผิดพลาดในการถ่ายข้อมูลได้

ถ้าไม่ต้องการมีการตรวจสอบสัญญาณกันระหว่างตัวส่งและตัวรับ สามารถใช้เพียงแค่ 3 ขา ก็สามารถติดต่อรับส่งข้อมูลกันได้แล้ว ซึ่งมีดังนี้

1 Transmit data (Tx) เป็นตัวส่งข้อมูลออกจาก DTE หรือ คอมพิวเตอร์

2 Receive data (Rx) เป็นตัวรับข้อมูลเข้าสู่ DTE

3 Signal ground เชื่อมต่อเพื่อให้สองสถานีเป็นระดับกราวนด์เดียวกัน

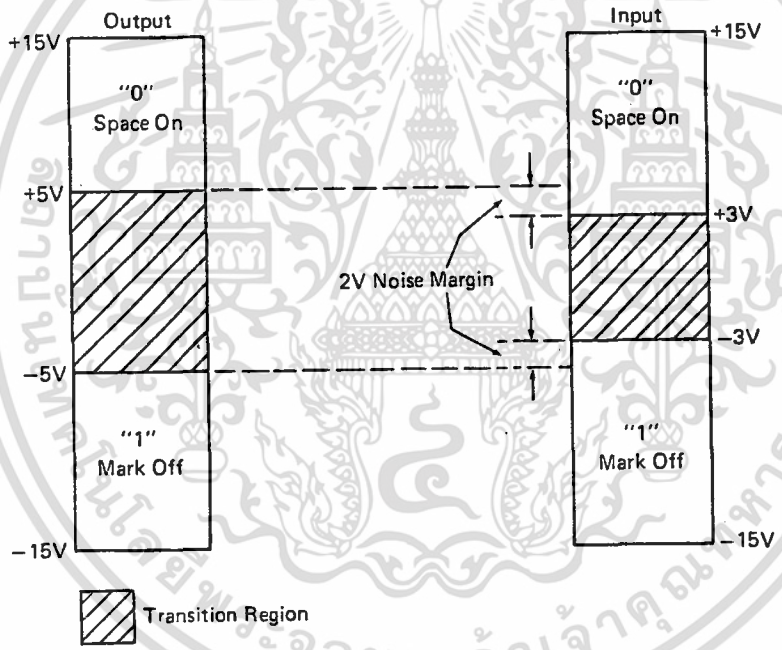


รูปที่ 2.10 ขาสัญญาณที่ใช้เชื่อมต่อ DTE และ DCE

ระดับความต่างศักย์ของสัญญาณของ RS 232C

เมื่อมีการส่งข้อมูลจากด้านตัวส่งจะกำหนด " ลอจิก 1 " หรือที่เรียกว่า mark ที่ระดับความต่างศักย์ -5 V ถึง -15 V และกำหนด " ลอจิก 0 " หรือ ที่เรียกว่า space ที่ระดับความต่างศักย์ 5 V ถึง 15 V

ทางด้านตัวรับจะกำหนด " ลอจิก 1 " ที่ระดับความต่างศักย์ -3 V ถึง -15 V และกำหนดว่าเป็น " ลอจิก 0 " ที่ระดับความต่างศักย์ 3 V ถึง 15 V จะเห็นว่าระดับช่วงความต่างศักย์ของตัวรับจะมากกว่าระดับช่วงความต่างศักย์ของตัวส่งอยู่ 2 V เพื่อที่ว่าป้องกันความผิดพลาดอันเนื่องมาจากการส่ง ซึ่งสัญญาณอาจจะสูญเสียไปให้กับสายส่ง ดังรูป

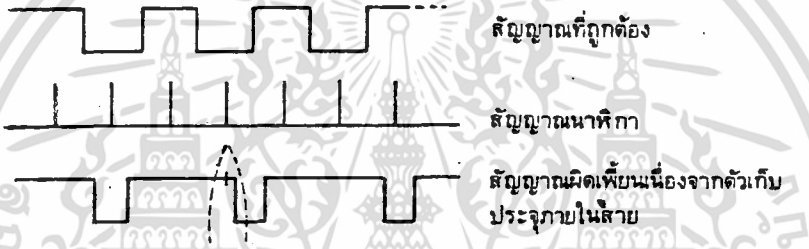


รูปที่ 2.11 ระดับความต่างศักย์ของสัญญาณของ RS 232C

ช่วง Transition region คือ ช่วงที่ไม่สามารถกำหนดได้ว่าเป็น "ลอจิก 1 " หรือ " ลอจิก 0 "

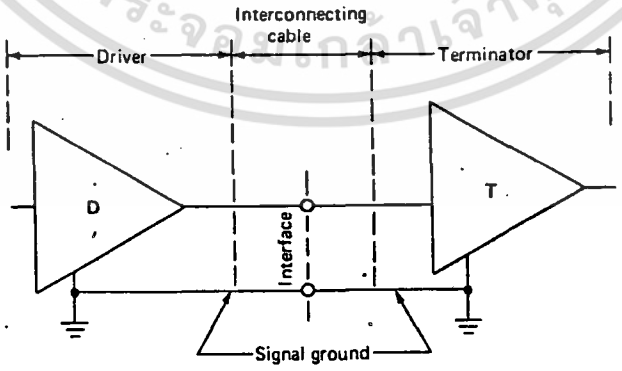
ข้อเสียของมาตรฐาน RS 232C

1 ค่าตัวเก็บประจุของอุปกรณ์รับสัญญาณ RS 232C รวมทั้งตัวเก็บประจุสเตย์ (Stay capacitance) ในสายจะต้องไม่มากกว่า 2500 pF สายที่รวมกันหลาย ๆ สาย ส่วนมากจะมีตัวเก็บประจุสเตย์ประมาณ 40-50 pF ต่อหนึ่งฟุต ดังนั้นสายนี้จะต่อได้ยาวสุด 50 ฟุต เพราะค่าตัวเก็บประจุสเตย์จะมากกว่า 2500 pF ถ้าหากตัวเก็บประจุสเตย์มากกว่าที่กำหนดนี้ ช่วงเวลาการเปลี่ยนแปลงระดับของสัญญาณจะมากกว่า 4% ตามที่ยอมให้ได้ในมาตรฐาน RS 232C ทำให้ฝ่ายรับตีความสัญญาณผิดไปจากความจริงเช่น Mark bit อาจจะยาวกว่า Space bit หรือ Space bit ยาวกว่า Mark bit ดังรูป



รูปที่ 2.12 สัญญาณที่ผิดเพี้ยนเนื่องจากสายยาวเกินไป

2 มาตรฐาน RS 232C จะมีรูปแบบการเชื่อมต่อระหว่างตัวส่งกับตัวรับเป็นแบบ Unbalanced interface ดังรูป



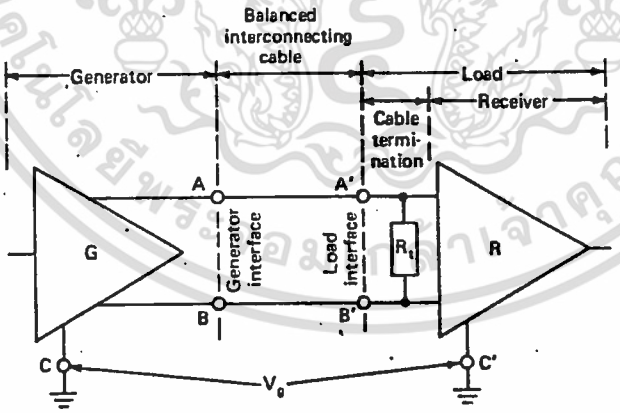
รูปที่ 2.13 Unbalance interface

ซึ่งจะมีปัญหาเรื่องระดับแรงดันกราวนด์ที่แตกต่างกันระหว่างเครื่องส่งกับเครื่องรับอันเนื่องมาจากสัญญาณรบกวนผ่านเข้ามาที่สายกราวนด์ หรือ สถานีรับและสถานีส่งมีระบบไฟฟ้าที่กราวนด์แตกต่างกัน สมมติว่า ทางด้านสถานีส่งมีระดับแรงดันกราวนด์มากกว่าทางด้านสถานีรับ 2 V ถ้าทางด้านส่งส่งแรงดันเข้าไป 5 V ทางฝ่ายรับจะมองเห็นแค่ 3 V เท่านั้น ซึ่งทำให้เกิดความผิดพลาดของข้อมูลได้

เนื่องจากทางสมาคมผู้ผลิตอุปกรณ์อิเลกทรอนิกส์แห่งสหรัฐอเมริกา หรือ The Electronic Industries Association (EIA) ได้ตระหนักถึงปัญหาข้างต้นจึงได้ออกมาตรฐานขึ้นใหม่เพื่อแก้ปัญหาดังกล่าว มาตรฐานที่ EIA เสนอออกมาก็คือ มาตรฐาน RS 422A ซึ่งจะกล่าวในหัวข้อถัดไป

2.3.2 มาตรฐาน RS 422A

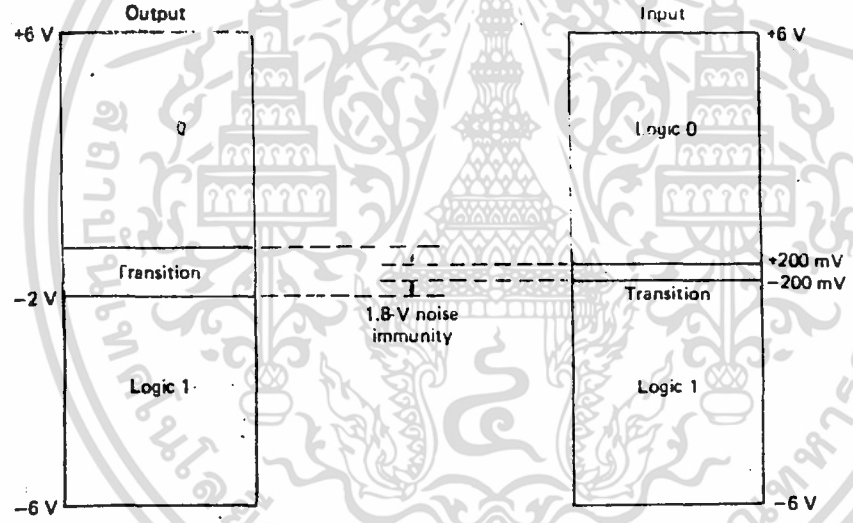
เนื่องจากปัญหาเรื่องระดับแรงดันกราวนด์ที่แตกต่างกันของด้านส่งและด้านรับ EIA จึงเสนอมาตรฐาน RS 422A ขึ้นมา ซึ่งมีรูปแบบการเชื่อมต่อระหว่างด้านส่งและด้านรับเป็นแบบ Balanced interface ดังรูป



รูปที่ 2.14 Balanced interface

เป็นการแก้ปัญหาโดยใช้หลักการความแตกต่างของสัญญาณของสายทั้งสอง หรือ Differential receiver ระดับสัญญาณที่ส่งจะเป็นความแตกต่างของสายทั้งสอง คือ (V_a-V_b) ที่จุด A จะมีความต่างศักย์ V_a ส่วนที่จุด B มีความต่างศักย์เป็น V_b ถ้าระหว่างจุดส่งและจุดรับมีระดับแรงดันกราวน์แตกต่างกันเป็น X ดังนั้นที่ด้านรับ จุด A' จะมีความต่างศักย์เป็น (V_a+X) และที่จุด B' มีความต่างศักย์เป็น (V_b+X) ดังนั้นทางด้านรับจะรับสัญญาณที่แตกต่างกัน(Differential receiver) คือ (V_a+X)-(V_b+X) ดังนั้นสัญญาณที่ได้คือ (V_a- V_b) ซึ่งเหมือนกับสัญญาณที่ส่งออกมาจากทางด้านส่ง ทำให้สามารถส่งได้ไกล อีกทั้งยังป้องกันสัญญาณรบกวนได้ดี

ระดับความต่างศักย์ของ RS 422A



รูปที่ 2.15 ระดับความต่างศักย์ของ RS 422A

RS 422A จะส่ง " ลอจิก 0 " ที่ระดับความต่างศักย์ 2 V ถึง 6 V และส่ง " ลอจิก 1 " ที่ระดับความต่างศักย์ -2 V ถึง -6 V และจะรับ " ลอจิก 0 " ที่ระดับความต่างศักย์ 200 mV ถึง 6 V ส่วน " ลอจิก 1 " จะรับที่ระดับความต่างศักย์ 200 mV ถึง 6V และเพื่อป้องกันการผิดพลาดในการส่งอันเนื่องมาจากการสูญเสียภายในสาย ระดับความต่างศักย์ของทางด้านส่งจะมากกว่าทางด้านรับอยู่ 1.8 V

ข้อแตกต่างระหว่างมาตรฐาน RS 422A และ RS 232C

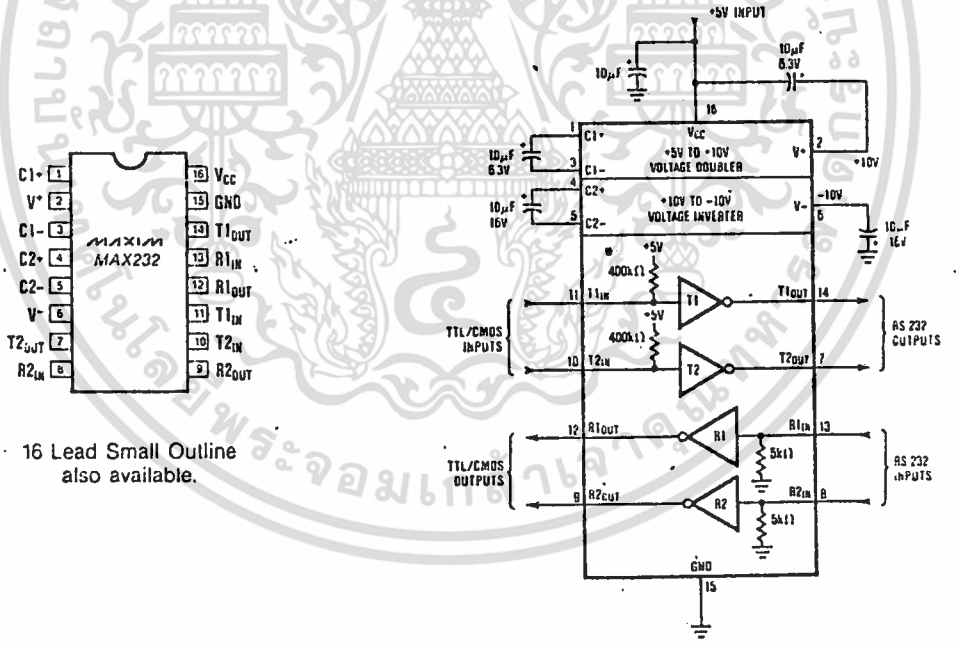
PARAMETER	RS 232C	RS 422A
Mode of operation	Single-ened	Differential
Number of drivers and receivers allowed	1 Driver 1 Receiver	1 Driver 10 Receiver
Maximum cable length (ft)	50	4000
Maximum data rate bits per second	20 k	10 M
Maximum common-mode voltage	±25 v	6 v -0.25 v
Driver output	±5 v min ±15 v max	±2 v min
Driver load	3 k to 7 k	100 min
Receiver input resistance	3 k to 7 k	4 k
Receiver sensitivity	±3 v	±200 mv

รูปที่ 2.16 ข้อแตกต่างระหว่างมาตรฐาน RS 232C และ RS 422A

จะเห็นว่ามาตรฐาน RS 422A สามารถสื่อสารข้อมูลได้ไกลถึง 4,000 ฟุต หรือ ประมาณ 1 กิโลเมตร ซึ่งมาตรฐาน RS 232C สามารถติดต่อได้แค่ระยะทางไม่เกิน 50 ฟุต นอกจากนี้มาตรฐาน RS 422A สามารถส่งถ่ายข้อมูลจากตัวส่งตัวเดียวไปยังตัวรับหลาย ๆ ตัวได้

2.3.3 MAX 232

เนื่องจากมาตรฐาน RS 232C จะใช้สัญญาณในการส่งและรับที่ระดับความต่างศักย์ ± 12 V และในคอมพิวเตอร์ส่วนใหญ่จะมีพอร์ตอนุกรมเป็นมาตรฐาน RS 232C อยู่ ดังนั้นถ้าต้องการติดต่อกับอุปกรณ์ที่รับส่งสัญญาณที่ระดับความต่างศักย์ ± 5 V เช่น มาตรฐาน RS 422A จึงต้องมีอุปกรณ์ที่มาแปลงระดับสัญญาณให้สามารถใช้ด้วยกันได้ ซึ่งก็คือ MAX 232 ซึ่งมีโครงสร้าง ดังรูป



รูปที่ 2.17 โครงสร้างของ MAX 232

สัญญาณที่ส่งออกมาจาก RS 232 C จะเข้าสู่ MAX 232 ที่ขา RS 232 อินพุท MAX 232 จะแปลงระดับสัญญาณให้เป็นระดับความต่างศักย์ $\pm$ 5 V ส่วนสัญญาณที่ส่งออกมาจากอุปกรณ์ภายนอกจะถูกแปลงโดย MAX 232 ให้เป็นระดับความต่างศักย์ $\pm$ 12 V ที่ขา RS 232 เอาท์พุท จะเห็นว่า MAX 232 มีตัวรับสัญญาณและตัวส่งสัญญาณถึง 2 ตัวด้วยกัน

บทที่ 3
ไมโครคอนโทรลเลอร์

เมื่อ 2 ทศวรรษที่ผ่านมา ได้มีการแนะนำเทคโนโลยีใหม่ซึ่งทำให้เกิดการเปลี่ยนแปลงในการวิเคราะห์ และ ควบคุมโลกรอบ ๆ ตัวเรา การเกิดอย่างควบคู่กันของ การพัฒนาสถาปัตยกรรม ทางไมโครคอมพิวเตอร์ และ การสร้างวงจรรวม ทำให้ไมโครคอมพิวเตอร์ หรือ คอมพิวเตอร์บนชิพเกิดขึ้นเป็นครั้งแรกในทางการค้า ในปี 1971 โดยการแนะนำคอมพิวเตอร์ 4 บิท เบอร์ 4004 จากบริษัทที่ไม่มีใครรู้จัก ชื่อ อินเทล คอเปอร์เรชั่น (Intel Coporation) ผลิตภัณฑ์อื่นที่เป็นที่รู้จักกันดีคือ อุปกรณ์ประเภทสารกึ่งตัวนำ โดยการบุกเบิกทางเทคโนโลยีของอินเทล ในปลายปี 1970 พวกเขาได้ทำการวิจัยและพัฒนาระบบไมโครโปรเซสเซอร์ขึ้น

ในปี 1970 จะเห็นว่าการเจริญเติบโตของจำนวนผู้ใช้คอมพิวเตอร์ส่วนบุคคลมีมากขึ้น รวมไปถึงใน งานธุรกิจ อุตสาหกรรม รัฐบาล การทหาร ของเล่นเด็ก การศึกษา และ ผู้ใช้ทั่ว ๆ ไป

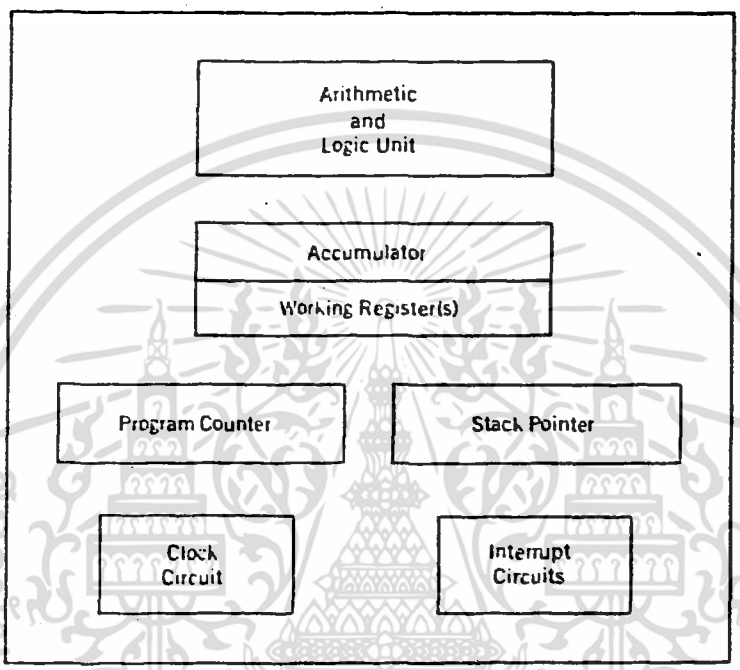
โดยการพัฒนาผลิตภัณฑ์ไมโครโปรเซสเซอร์ไปเป็นไมโครคอนโทรลเลอร์ด้วยเทคนิคการสร้าง และ แนวความคิดทางโปรแกรมที่เหมือนกัน ทำให้มีความเป็นไปได้ที่ไมโครโปรเซสเซอร์จุดประสงค์ทั่วไป จะเข้ากันได้กับไมโครคอนโทรลเลอร์ โดยทั่วไปไมโครคอนโทรลเลอร์ไม่ค่อยเป็นที่รู้จัก ไม่เหมือนกับไมโครโปรเซสเซอร์ซึ่งเป็นที่รู้จักมาก อย่างไรก็ตาม เป็นที่รู้กันดีว่ามีบางลักษณะของ ไมโครคอนโทรลเลอร์ ซึ่งสามารถตอบสนองการทำงานในการควบคุม เครื่องซักผ้า , เครื่องอบ , วีดีโอเกมส์ , โทรศัพท์ , ไมโครเวฟ , เครื่องรับโทรทัศน์ และรวมไปถึงเครื่องมืออื่น ๆ ที่สามารถโปรแกรมการทำงานได้ บริษัทอินเทล ตื่นตัวในเรื่องการแข่งขันในยุคของไมโครชิพ ซึ่งมีความต้องการผลิตภัณฑ์ของพวกเขา และ เครื่องจักรซึ่งพวกเขาใช้สร้างผลิตภัณฑ์เหล่านี้

3.1 ไมโครคอนโทรลเลอร์และไมโครโปรเซสเซอร์

ไมโครโปรเซสเซอร์ และ ไมโครคอนโทรลเลอร์ เกิดจากแนวความคิดพื้นฐานเดียวกัน สร้างโดยกลุ่มคนเดียวกัน และ ขายโดยระบบผู้ออกแบบและผู้เขียนโปรแกรมเดียวกัน อะไรคือความแตกต่างของไมโครคอนโทรลเลอร์กับไมโครโปรเซสเซอร์

3.1.1 ไมโครโปรเซสเซอร์

ไมโครโปรเซสเซอร์เป็นที่รู้จักกันดีว่าเป็นหน่วยประมวลผลกลาง (Central Processing Unit) แม้ว่ามันจะเป็นคอมพิวเตอร์บนชิพ แต่ไมโครโปรเซสเซอร์ก็ยังไม่ใช้ดิจิตอลคอมพิวเตอร์ที่สมบูรณ์แบบ



รูปที่ 3.1 แผนภาพแสดงส่วนต่าง ๆ ภายในไมโครโปรเซสเซอร์

จากรูปที่ 3.1 แสดงให้เห็นแผนภาพโครงสร้างของไมโครโปรเซสเซอร์ซึ่งประกอบไปด้วย หน่วยคณิตศาสตร์และตรรกศาสตร์ (ALU) , โปรแกรมเคาน์เตอร์ , สแตกพอยน์เตอร์ , รีจีสเตอร์บางส่วน ,วงจรเวลานาฬิกา ,วงจรอินเตอร์รัพท์

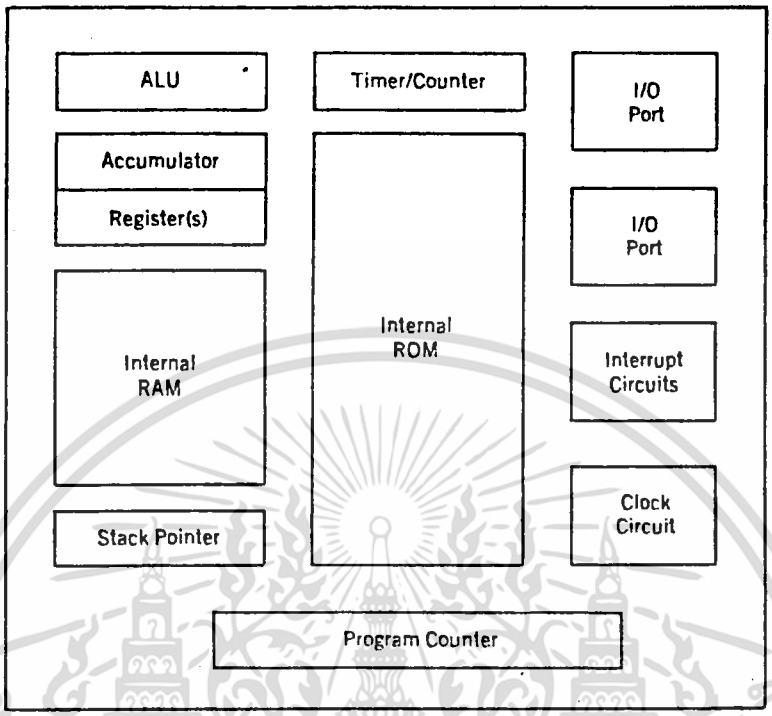
เพื่อที่จะสร้างไมโครคอมพิวเตอร์ที่สมบูรณ์เราจะต้องเพิ่มหน่วยความจำ ปกติจะเป็นหน่วยความจำประเภทที่สามารถอ่านได้เพียงอย่างเดียว (ROM) และหน่วยความจำประเภททั้งอ่านและเขียนได้ (RAM) , ตัวถอดรหัสหน่วยความจำ (Memory Decoder), ตัวสร้างสัญญาณนาฬิกา (Oscillator) และ อุปกรณ์ที่ติดต่อภายนอกทั้งส่วนรับ และ ส่งข้อมูล (I/O Port) ตัวอย่างเช่น พอร์ทส่งข้อมูลขนาน และ อนุกรม นอกจากนั้นยังมี

อุปกรณ์พิเศษอีก เช่น ตัวอินเตอร์รัพท์ หรือ เคาน์เตอร์ซึ่งจะช่วย CPU ในการจัดการกับเวลา หรือ แบ่งเวลา อุปกรณ์ที่ใช้เก็บข้อมูล สำหรับไมโครคอมพิวเตอร์ ปกติจะเป็นฟลอปปี้ดิสก์ (Floppy Disk) และ อุปกรณ์รับและส่งข้อมูลอื่น ๆ เช่นคีย์บอร์ด และ จอแสดงผล (CRT) เป็นที่ยอมรับกันว่าคอมพิวเตอร์ขนาดเล็ก จะสามารถประยุกต์ใช้งานกับโปรแกรมทั่วไปได้กว้างขวางมาก

เหตุผลสำคัญที่ใช้อธิบายการออกแบบไมโครโปรเซสเซอร์คือ สามารถใช้ในงานทั่ว ๆ ไปได้กว้าง การออกแบบฮาร์ดแวร์ ของไมโครโปรเซสเซอร์จะถูกจัดอยู่ในระบบเล็กหรือใหญ่ ขึ้นอยู่กับโครงสร้างของอุปกรณ์รอบ CPU ซึ่งบ่งบอกถึงการประยุกต์ใช้งาน สถาปัตยกรรมภายในCPUใช้รหัสการทำงานในระดับเครื่อง ซึ่งสามารถใช้งานได้กว้างขวางมาก แต่ก็มีความยืดหยุ่นเท่าที่จะเป็นไปได้

หลักในการทำงานของไมโครโปรเซสเซอร์คือ การเฟ็ทข้อมูล (Fetch Data) ทำการคำนวณข้อมูลและเก็บข้อมูลในอุปกรณ์จัดเก็บ หรือ แสดงผลให้กับผู้ใช้ โปรแกรมจะถูกใช้โดยไมโครโปรเซสเซอร์ ซึ่งจะเก็บอยู่ในอุปกรณ์จัดเก็บ และ นำออกมาลงใน แรม ตามคำสั่งของผู้ใช้ บางโปรแกรมจะถูกเก็บอยู่ในรอม โปรแกรมที่อยู่ในรอม เหล่านี้เป็นโปรแกรมพื้นฐานเล็ก ๆ ซึ่งจะทำงานทั่วไป และ ควบคุมอุปกรณ์อื่น ๆ ที่เชื่อมต่อกับระบบ การออกแบบไมโครโปรเซสเซอร์จะต้องทำให้มันสามารถขยายการทำงานได้มากเท่าที่จะเป็นไปได้ และมีราคาต้นทุนต่ำ

3.1.2 ไมโครคอนโทรลเลอร์



รูปที่ 3.2 แผนภาพแสดงส่วนต่าง ๆ ภายในไมโครคอนโทรลเลอร์

รูป 3.2 แสดงถึงแผนภาพของไมโครคอนโทรลเลอร์ ซึ่งเป็นคอมพิวเตอร์บนชิพที่แท้จริง การออกแบบจะรวบรวมส่วนต่าง ๆ ที่พบในไมโครโปรเซสเซอร์ เช่น CPU , หน่วยคณิตศาสตร์และตรรกศาสตร์ , โปรแกรมเคาน์เตอร์ , สแตกพอยน์เตอร์ และรีจิสเตอร์ นอกจากนี้ยังเพิ่มส่วนอื่น ๆ ขึ้นอีก เพื่อให้เป็นคอมพิวเตอร์ที่สมบูรณ์ยิ่งขึ้น คือ รอม ,แรม, พอร์ทรับส่งข้อมูล ทั้งแบบขนานและอนุกรม , วงจรนับ และ วงจรเวลา

เช่นเดียวกับไมโครโปรเซสเซอร์ คือ ไมโครคอนโทรลเลอร์เป็นอุปกรณ์ที่ใช้งานทั่วไป แต่การเพิ่มข้อมูลที่มาคำนวณ และ การควบคุมอุปกรณ์จากภายนอกมีข้อจำกัด หลักในการใช้งาน ไมโครคอนโทรลเลอร์ คือ การควบคุมการทำงานของเครื่องจักรที่ใช้โปรแกรมซึ่งเก็บอยู่ในรอม และไม่มีการเปลี่ยนแปลงโปรแกรมควบคุมการทำงาน ตลอดการทำงานของเครื่องจักร

การออกแบบ ไมโครคอนโทรลเลอร์ มีลักษณะใกล้เคียงกับไมโครโปรเซส-เซอร์มากจนเหมือนภาพสะท้อน การออกแบบไมโครโปรเซสเซอร์ประสพผลสำเร็จ เนื่อง

จากความยืดหยุ่น และ ชุดคำสั่งมากมายที่จะนำมาใช้งาน ชุดคำสั่งเหล่านี้ทำงานร่วมกับโครงสร้างฮาร์ดแวร์ ซึ่งจะติดต่อกับหน่วยความจำและหน่วยรับและส่งข้อมูล การทำงานของไมโครโปรเซสเซอร์สามารถที่จะเคลื่อนย้ายข้อมูล จากหน่วยความจำ ภายนอกมายังตัวมันได้ สถาปัตยกรรมที่ทำหน้าที่ของหน่วยความจำสำรอง (Register) ภายใน CPU สามารถถูกโปรแกรม ย้ายไปยังส่วนประมวลผล และ ชุดคำสั่งทำให้เกิดความสะดวก ในการพัฒนาระบบ ขาที่ต่อกับ ไมโครโปรเซสเซอร์ ไปยังหน่วยความจำภายนอกมีเพียงหน้าที่เดียวในแต่ละขา ข้อมูลอยู่ในรูปของไบท์ และ มีขนาดใหญ่

การออกแบบไมโครคอนโทรลเลอร์ใช้ชุดคำสั่งจำกัดอย่างมาก เพียงหนึ่งหรือสองไบท์ ซึ่งใช้เคลื่อนย้ายข้อมูลจากหน่วยความจำภายในไปยัง หน่วยคณิตศาสตร์ และตรรกศาสตร์ ขาไอซีจะถูกโปรแกรมให้มีหลายๆหน้าที่ในขาเดียวกัน ทั้งนี้ขึ้นอยู่กับโปรแกรมที่ผู้ใช้สร้างขึ้น ไมโครคอนโทรลเลอร์ รับและส่งข้อมูล ภายในขาเดียวกัน สถาปัตยกรรมและชุดคำสั่งจะมีความสามารถสูงสุด ในการจัดการกับข้อมูลในขนาด บิท และ ไบท์

3.1.3 การเปรียบเทียบไมโครคอนโทรลเลอร์กับไมโครโปรเซสเซอร์

ลักษณะที่ตรงข้ามกัน อย่างชัดเจน ระหว่าง ไมโครโปรเซสเซอร์ และไมโครคอนโทรลเลอร์ คือ ไมโครโปรเซสเซอร์ จะมีคำสั่งมากมาย ในการเคลื่อนย้ายข้อมูลจากหน่วยความจำภายนอกมายังCPU ส่วนไมโครคอนโทรลเลอร์อาจจะมีคำสั่งเพียงหนึ่ง หรือ สองคำสั่งเท่านั้น อีกลักษณะหนึ่งคือ ไมโครโปรเซสเซอร์จะมีเพียงหนึ่ง หรือสองคำสั่งเท่านั้นที่ทำงานในระดับบิท ส่วนไมโครคอนโทรลเลอร์นั้นมีมากมาย

กล่าวโดยสรุป คือ ไมโครโปรเซสเซอร์ จะเกี่ยวข้องกับ การย้ายข้อมูลจากหน่วยความจำภายนอกมายังตัว CPU ส่วนไมโครคอนโทรลเลอร์จะเกี่ยวข้องกับการเคลื่อนย้ายระดับบิทภายในชิพ ไมโครคอนโทรลเลอร์สามารถทำหน้าที่ในฐานะของ ไมโครคอมพิวเตอร์ โดยไม่ต้องใช้วงจรภายนอกเพิ่มขึ้น ส่วนไมโครโปรเซสเซอร์จะมีอุปกรณ์เชื่อมต่อภายนอกมากกว่า

เพื่อจะให้เห็น ความแตกต่างระหว่าง ไมโครโปรเซสเซอร์ และไมโครคอนโทรลเลอร์ ตามตารางที่ 3.1 จะแสดงให้เห็นถึงโครงสร้างขา และสถาปัตยกรรม และ ชุดคำสั่งของไมโครโปรเซสเซอร์ 8 บิทยอดนิยม ของ Zilog คือ Z-80 เปรียบเทียบกับไมโครคอนโทรลเลอร์ 8 บิทของ อินเทล เบอร์ 8051

ตารางที่ 3.1 เปรียบเทียบคุณสมบัติระหว่าง Z80 กับ 8051

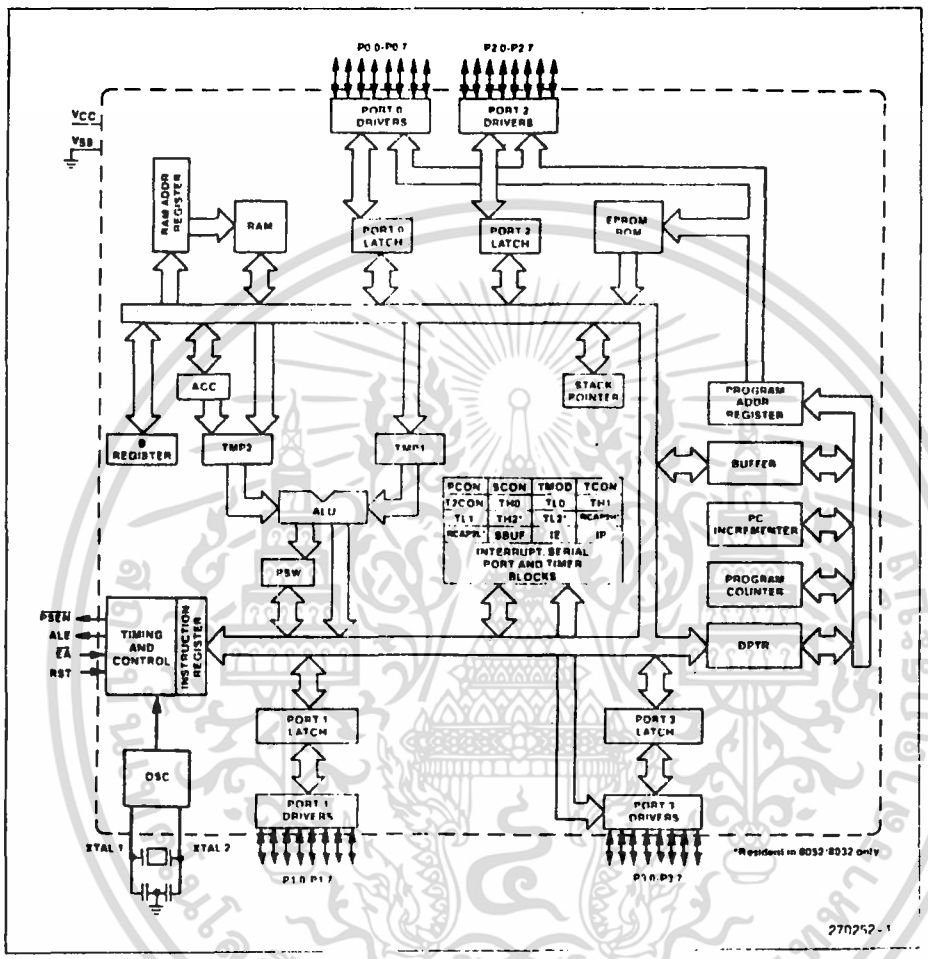
	Z80	8051
โครงสร้างภายนอกของขาไอซี		
จำนวนขาทั้งหมด	40	40
ขาแอดเดรส	16	16
ขาข้อมูล	8	8
ขาอินเทอร์รัพท์	2	2
ขาอินพุทและเอาท์พุท	0	32
สถาปัตยกรรม		
8 บิท รีจีสเตอร์	20	34
16 บิท รีจีสเตอร์	4	2
ขนาดของสแตกค์	64 K	128
รอมภายใน	0	4 K
แรมภายใน	0	128
หน่วยความจำภายนอก	64 K	128 K
แฟลก	6	4
ไทม์เมอร์	0	2
พอร์ทขนาน	0	4
พอร์ทอนุกรม	0	1

	Z80	8051
ชุดคำสั่ง		
External moves	4/17	2/6
Block moves	2/4	0
Bit manpulate	4/4	12/12
Jump on bit	0	3/3
Stack	3/15	2/2
Single byte	203	49
Multi-byte	409	62

จากตารางจะพบว่า ไมโครโปรเซสเซอร์ Z80 และไมโครคอนโทรลเลอร์ 8051 มีคุณสมบัติที่เป็นส่วนดี และข้อจำกัดที่แตกต่างกัน ดังนั้นจะบอกไม่ได้ว่าตัวใดจะดีกว่ากัน การออกแบบจึงมีแนวโน้มที่จะใช้ประโยชน์ในจุดประสงค์ที่ต่างกัน และในทางที่ต่างกัน ตัวอย่างเช่น Z-80 มีชุดคำสั่งมากมาย ข้อเสียในการที่มีคำสั่งมากมาย คือ ต้องใช้คำสั่งหลาย ๆ ไบท์จำนวนมาก ประมาณ 71 % ของคำสั่งทั้งหมด แต่ละไบท์ของคำสั่งจะต้องเฟ็ทจากหน่วยความจำ และ แต่ละการเฟ็ทจะต้องใช้เวลา ผลดังกล่าวทำให้โปรแกรมยาวขึ้น และ การทำงานของโปรแกรมช้าลง ตรงข้ามกับคำสั่งไบท์เดียวของ 8051 มีคำสั่งหลายไบท์อยู่ 62 % โปรแกรมของ 8051 จะกระทัดรัดกว่า และทำให้การทำงานของโปรแกรมรวดเร็วกว่า โดยทำงานได้ผลลัพธ์เช่นเดียวกัน

ข้อเสีย ของการใช้ชุดคำสั่ง ใน 8051 คือ ไม่สะดวกที่จะเขียนรหัสคำสั่ง ซึ่งข้อเสียนี้สามารถแก้ไขได้ โดยการใช้ภาษาระดับสูง เช่น Basic หรือ C แทนซึ่งภาษาทั้งสองเป็นที่นิยมมาก สำหรับนักพัฒนาระบบ 8051 ทำให้ลดเวลาในขั้นตอนการพัฒนาโปรแกรมได้มาก

3.2 สถาปัตยกรรมของไอซีตระกูล 51



รูปที่ 3.3 แผนภาพแสดงสถาปัตยกรรมของMCS51

คุณสมบัติของ MCS 51 มีดังต่อไปนี้

1. CPU 8 บิท ควบคุมได้ง่าย

2. มีการทำงานแบบ Single step

3. สายอินพุท และเอาท์พุท มีจำนวน 32 เส้น ใช้เลือกแอดเดรส แยกต่างหากจากกันได้

4. มีแรมบรรจุภายใน ขนาด 128 ไบท์ หรือ 256 ไบท์

5. วงจรตั้งเวลา/วงจรนับมีขนาด 16 บิท จำนวน 2/3 ตัว

6. กำหนดเป็น UART (Universal Synchronous Asynchronous Receiver Transmitter) ที่รับส่งข้อมูลอนุกรม ชนิด Full duplex ได้

7. การอินเตอร์รัพท์ แบ่งเป็น 2 ระดับจาก 5 หรือ 6 แหล่ง

8. มีวงจรกำเนิดสัญญาณนาฬิกา(โดยการต่อ X-TAL ภายนอกเท่านั้น) อยู่ภายในตัว

9. มีหน่วยความจำชนิดรอม สำหรับเก็บข้อมูลภายในขนาด 8 หรือ 9 กิโลไบท์ (EPROM 8751 และ 8752)

10. มีแอดเดรสของหน่วยความจำสำหรับเก็บโปรแกรมภายนอก จำนวนทั้งหมด 64 กิโลไบท์

11. มีแอดเดรสของหน่วยความจำสำหรับเก็บข้อมูลภายนอก จำนวนทั้งหมด 64 กิโลไบท์

12. คำสั่งทั้งหมดมี 111 คำสั่ง

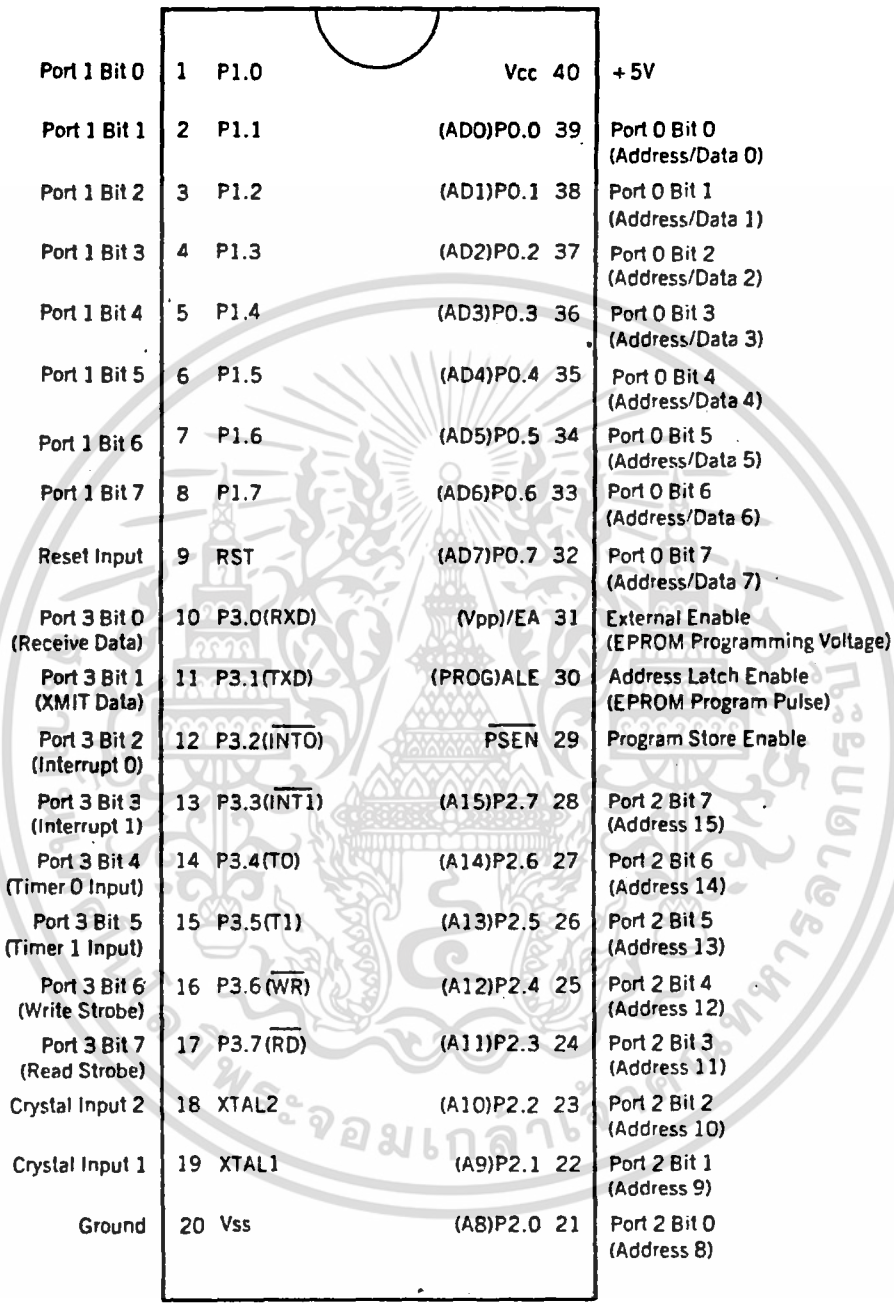
13. ทำงานด้วยเลขฐาน 10 และ ฐาน 16

14. ตัวแปลภาษา เบสิก มี ขนาด 8 กิโลไบท์ (8052AH-BASIC)

15. มีคำสั่งควบคุมการเขียนข้อมูล ลงใน EPROM ภายในตัวด้วยภาษาเบสิก (8052AH-BASIC)

16. มีคำสั่งเฉพาะของภาษาเบสิก ที่ใช้สำหรับ อินพุท และ เอาท์พุท และ อินเตอร์เฟส

8051 DIP Pin Assignments



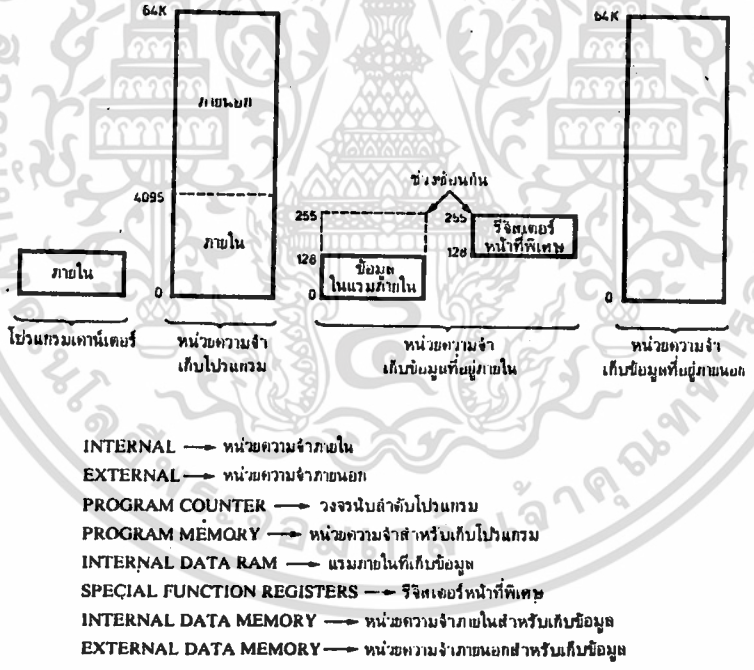
รูปที่ 3.4 แสดงโครงสร้างภายนอกของไอซี MCS 51

8051, 8051AH และ 8052AH เป็นไมโครคอนโทรลเลอร์ที่ทำงานโดยการควบคุมจากโปรแกรมในหน่วยความจำ (ROM)ซึ่งจะถ่ายโปรแกรมต้นแบบ ลงบนรอมจำนวนหลาย ๆ ตัว จึงเหมาะกับการผลิตเพื่อใช้งานควบคุมจำนวนมากๆ (8051 หน่วยความจำรอมขนาด 4 กิโลไบท์ ส่วน 8052AH จะมีหน่วยความจำเพิ่มขึ้นโดยสร้างจาก HMOSII)

มีอีก 2 เบอร์ ที่ทำงานคล้ายกันคือ 8031AH และ 8032AH ที่ใช้แทน 8051 และ 8052AH ได้ตามลำดับ โดยไม่ต้องส่งให้โรงงานโปรแกรมให้ เพราะเราอาจเขียนและทดสอบโปรแกรมด้วยหน่วยความจำภายนอกแทน(ไม่มีรอมภายใน)

เบอร์ 8751 และ 8752 สามารถป้องกันการอ่านจากภายนอก ซึ่งมีEPROM ภายใน ที่จัดค่าเวลาในการอ่านข้อมูล เพื่อกันการลอกเลียนโปรแกรม

เบอร์ 8052AH BASIC มีตัวแปลภาษา เบสิก ซึ่งเป็นรอมอยู่ภายใน



รูปที่ 3.5 แสดงโครงสร้างภายในหน่วยความจำของMCS51

3.2.1 หน่วยความจำภายใน

จากรูปที่ 3.5 เป็นหน่วยความจำภายในตัว MCS 51 หน่วยความจำนี้แบ่งได้ 2 กลุ่มคือ หน่วยความจำสำหรับเก็บโปรแกรม และ หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำแรกมี แอดเดรสที่ต่ำกว่า 4 หรือ 6 กิโลไบท์ บรรจุอยู่ใน รอม ส่วน MCS 51 ที่ไม่มีรอมภายใน จะใช้หน่วยความจำภายนอก ซึ่งอาจเป็น รอม , แรม หรือ EPROM แทนก็ได้

ตารางที่ 3.2 แสดงไมโครคอนโทรลเลอร์ ตระกูลMCS 51

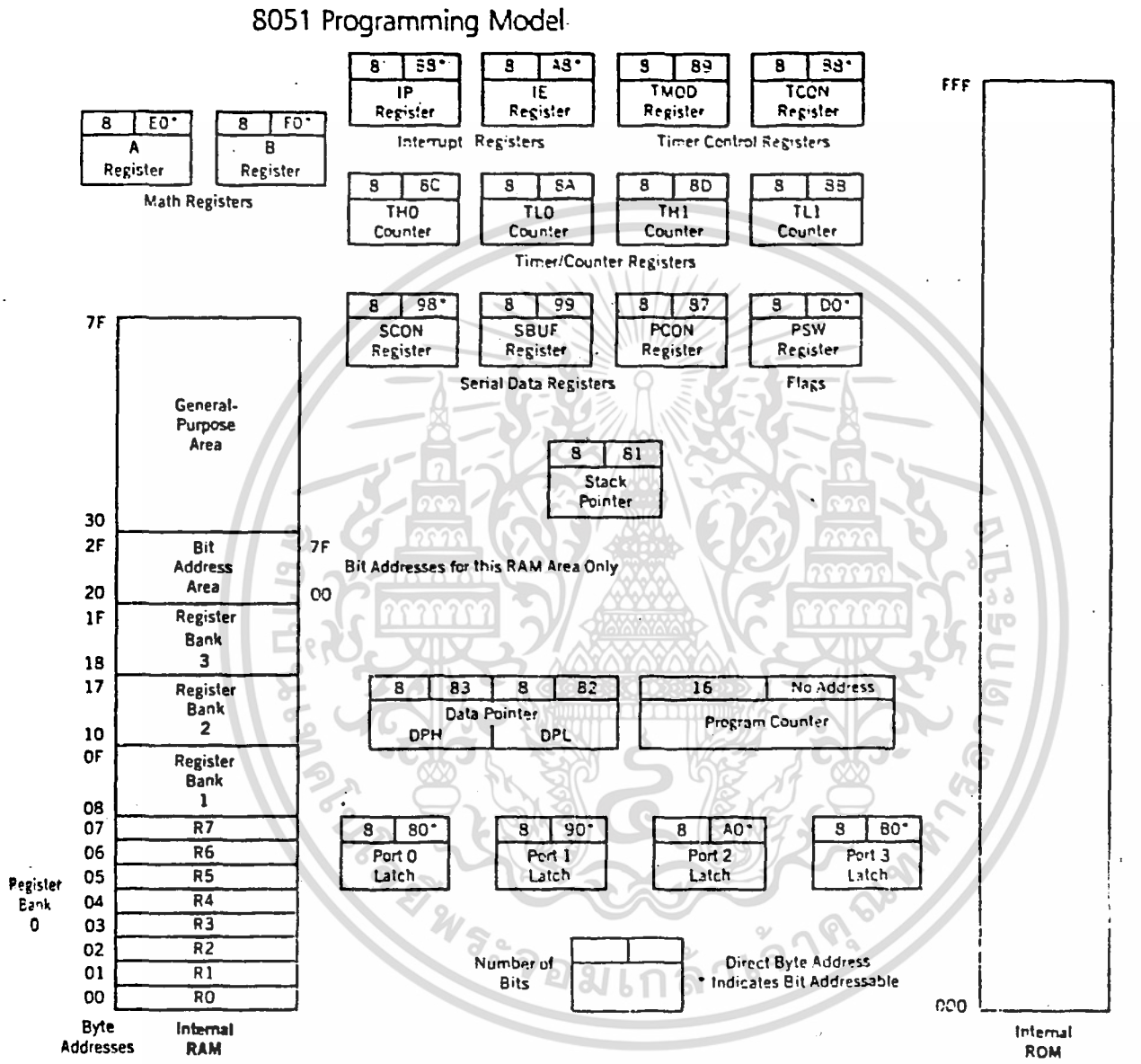
เบอร์	หน่วยความจำภายใน		ตั้งเวลา/นับ	อินเตอร์รัพต์ หมายเลข
	โปรแกรม	ข้อมูล		
8052AH	8K×8 ROM	256×8 RAM	3×16-Bit	6
8051AH	4K×8 ROM	128×8 RAM	2×16-Bit	5
8051	4K×8 ROM	128×8 RAM	2×16-Bit	5
8032AH	ไม่มี	256×8 RAM	3×16-Bit	6
8031AH	ไม่มี	128×8 RAM	2×16-Bit	5
8031	ไม่มี	128×8 RAM	2×16-Bit	5
8751H	4K×8 EPROM	128×8 RAM	2×16-Bit	5
8751H-12	4K×8 EPROM	128×8 RAM	2×16-Bit	5

MCS 51 จะอ่านหน่วยความจำสำหรับเก็บโปรแกรม เข้ามาเป็นภาษาเครื่องตามลำดับ ส่วนหน่วยความจำสำหรับเก็บข้อมูล จะใช้เป็นที่เก็บตัวแปรการคำนวณหาผลลัพธ์ การจัดการกับข้อมูลที่มีขนาด 16 บิท , ตารางที่ใช้ค้นหาค่าต่างๆ , และหน้าที่อื่นคล้ายกัน

3.2.2 รีจีสเตอร์ ภายใน

MCS 51 มีรีจีสเตอร์ที่อำนวยความสะดวกในการใช้งาน ควบคุมคำสั่งต่าง ๆ ประกอบด้วย แอคคิวมูเลเตอร์ รีจีสเตอร์ B ที่ใช้ในการคูณและหาร รีจีสเตอร์สถานะ สแตกพอยน์เตอร์ ดาต้าพอยน์เตอร์ (2x8 บิท หรือ 1x16 บิท) พอร์ทหมายเลขศูนย์ถึงพอร์ทหมายเลขสาม รีจีสเตอร์แบบคู่ ซึ่งใช้ส่งและรับข้อมูลชนิดอนุกรม รีจีสเตอร์16บิทที่เป็นวงจรตั้งเวลาและวงจรนับ รีจีสเตอร์ซึ่งจองไว้สำหรับใช้นับ รีจีสเตอร์คำสั่งสำหรับ

หน้าที่พิเศษ (เช่น การอินเตอร์รัพท์ RTC:Real Time Clock)และ เอาท์พุทแบบอนุกรม



รูปที่ 3.6 แสดงรีจีสเตอร์ภายใน MCS 51

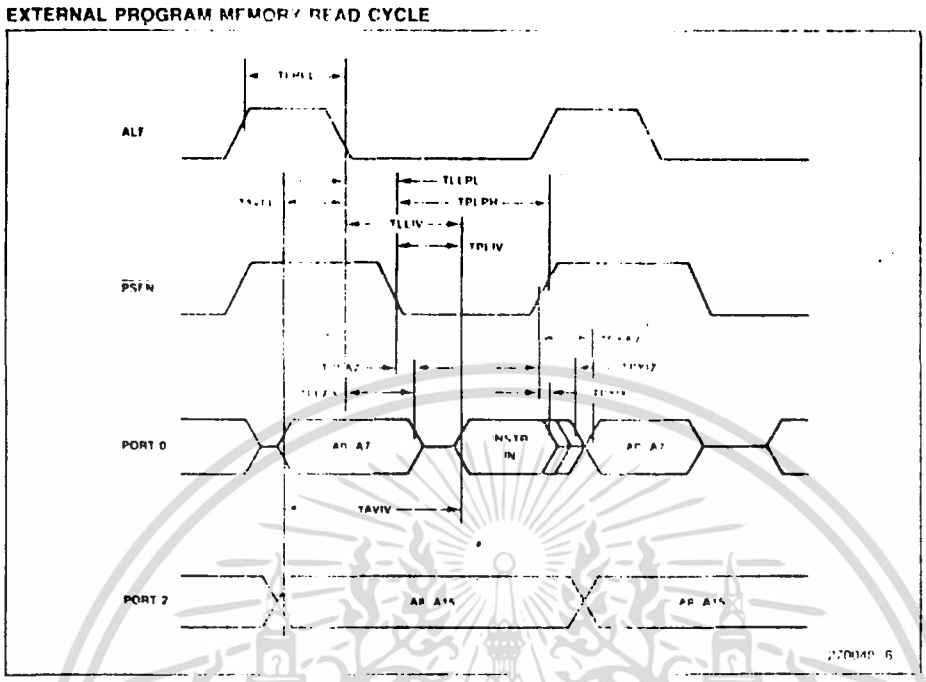
3.2.3 สายต่าง ๆ ของบัสและพอร์ท

จะเห็นโครงสร้างภายในของ MCS 51 ในรูป 3.3 เราจะสมมติให้มีบัสสองทิศทาง 4 เส้น และพอร์ทขนาด 8 บิท ตามทฤษฎี แต่ที่จริงนั้นจะใช้ได้ก็ต่อเมื่อมีการใช้หน่วยความจำภายในตัว (รอมหรือแรม) เมื่อไม่ใช้หน่วยความจำภายใน พอร์ท 0 และ 2 จะถูกใช้เป็นบัสของข้อมูล และแอดเดรส ดังนั้นพอร์ท 2 พอร์ท ยังคงใช้งานเป็นอินพุทและเอาท์พุท พอร์ท 2 ทำหน้าที่ เป็นสายสัญญาณแอดเดรส $A_{15},\ldots,A_8$ ส่วนพอร์ท 0 ทำหน้าที่เป็นสายสัญญาณแอดเดรส $A_7,\ldots,A_0$ และสัญญาณข้อมูล $D_7,\ldots,D_0$

เอาท์พุทของขา $\overline{RD}$ และ $\overline{WR}$ มาจากสายเอาท์พุทของพอร์ท 3 โดย โปรแกรมภายใน ใช้สัญญาณ $\overline{RD}$ และ $\overline{WR}$ เพื่อการเขียนและอ่านข้อมูล กับข้อมูลของหน่วยความจำภายนอก

3.2.3.1 ขา $\overline{PSEN}$

ขา $\overline{PSEN}$ เป็นขารับสัญญาณสำหรับเปิดให้มีการอ่านหน่วยความจำภายนอก ถ้าสังเกตทุก ๆ รอบคำสั่ง ระหว่างการทำงานด้วยโปรแกรมในรอมหรือ EPROM จะเห็นว่าสัญญาณ $\overline{PSEN}$ ทำงานถึงสองครั้งเหมือนสัญญาณ ALE เพราะว่ามีการอ่านข้อมูล จำนวน 2 ไบท์ ในแต่ละคำสั่ง ขา $\overline{PSEN}$ นี้ไม่ทำงานเมื่อมีภาษาเครื่องเก็บอยู่ในหน่วยความจำภายใน และ ถ้าหากหน่วยความจำภายนอก ไม่มีข้อมูลบรรจุอยู่ $\overline{PSEN}$ ก็จะไม่ทำงานด้วยเช่นกัน ส่วน 8052AH BASIC ไม่ใช้สัญญาณ $\overline{PSEN}$ เลยเพราะใช้รอมภายในตัว ซึ่งมีหน้าที่เป็นตัวแปลภาษาเบสิก



รูปที่ 3.7 แสดงตารางเวลา (Timing Diagram)

3.2.3.2 ขา EA

เป็นขาอินพุท ที่ใช้ร่วมกับแอดเดรสภายนอก โดยจะมีค่าลอจิก "0" เมื่อโปรเซสเซอร์อ่านค่าคำสั่งจากหน่วยความจำภายนอก (ปกติโปรเซสเซอร์ จะอ่านค่าหน่วยความจำภายในน้อยกว่าอ่านจาก หน่วยความจำภายนอก) ขา EA ยังมีหน้าที่เป็นอินพุทสำหรับป้อนแรงดัน 21 โวลต์ เพื่อโปรแกรมให้กับ EPROM (สำหรับกรณีที่ใช้ 8751 หรือ 8752)

3.2.4 วงจรนับ/วงจรตั้งเวลา

จากตาราง 3.2 จะเห็นว่า 8051 มีวงจรนับ และวงจรตั้งเวลา ขนาด 16 บิท มากกว่า 8051 อยู่หนึ่งตัว ต่อไปเราจะศึกษาการทำงานอย่างย่อ ๆ ของวงจรนับและวงจรตั้งเวลา

3.2.4.1 เมื่อทำงานเป็นวงจรตั้งเวลา

รีจีสเตอร์ที่ทำหน้าที่ตั้งเวลา จะเพิ่มค่าขึ้นหนึ่ง ทุกๆ รอบคำสั่งของเครื่อง และจะนับด้วยอัตราสูงสุดที่ 1/12 ของความเร็วสัญญาณนาฬิกาของโปรเซสเซอร์

3.2.4.2 เมื่อทำงานเป็นวงจรนับ

รีจีสเตอร์วงจรนับจะเพิ่มขึ้นหนึ่ง เมื่อมีสัญญาณป้อนให้อินพุท T_0, T_1 หรือ T_2 (T_2 มีเฉพาะ8052,8032) เป็นของสัญญาณขาลง อัตราการนับสูงสุดคือ 1/24 ของความเร็วสัญญาณนาฬิกาของโปรเซสเซอร์

วงจรนับและวงจรตั้งเวลา 0 และ 1 มีวิธีโปรแกรมให้ทำงานได้ต่างกันถึง 4 แบบ รวมทั้งการทำงานเป็น 8 หรือ 16 บิท และการบรรจุค่าพรีเซตหนึ่งค่าได้เองอย่างอัตโนมัติ

วงจรนับและวงจรตั้งเวลาที่ 2 (เฉพาะ 8052,8032 เท่านั้น) มีการทำงานย่อยได้ อีก 3 ชนิด

1. วงจรนับ 16 บิท ที่สามารถโหลดค่ากลับคืนเองอย่างอัตโนมัติ
2. วงจรนับที่จองไว้ ชนิด 16 บิท
3. วงจรกำเนิดสัญญาณของการส่งบิท เพื่อใช้อินเตอร์เฟสชนิดอนุกรม

3.2.5 พอร์ทชนิดอนุกรม

ไมโครคอนโทรลเลอร์ทั้งหมดในตระกูล MCS 51 เป็นชิพที่มีอินเตอร์เฟสอนุกรมชนิดสองทิศทาง ทำให้รับและส่งข้อมูลพร้อมกัน ตัวรับข้อมูลชนิดอะซิงโครนัส (Asynchronus Receiver) มีบัฟเฟอร์สำหรับข้อมูลเป็นพิเศษเพื่อความเร็วในการสื่อสาร พอร์ทชนิดอนุกรมนั้นเราสามารถเลือกโปรแกมเพื่อเลือกใช้การทำงานแบบใดแบบหนึ่งใน 4 แบบ ด้วยการใช้โปรแกรมควบคุมอัตราการส่งข้อมูลและรูปแบบของข้อมูล

อัตราการส่งข้อมูลที่เลือกใช้ได้จะสูงถึง 19200 บิท/วินาที ด้วยความเร็วของสัญญาณนาฬิกา 1 เมกกะเฮิรตซ์ สำหรับใช้ในระบบเครือข่าย (Network) และระบบการสื่อสารของโปรเซสเซอร์ หลายตัวร่วมกันเราเลือกความเร็วของสัญญาณนาฬิกาด้วยวงจรนับและวงจรตั้งเวลา

3.2.6 อินเตอร์รัพท์และชุดคำสั่ง

8051 รับการอินเตอร์รัพท์ได้ 5 แหล่ง ส่วน 8052 รับอินเตอร์รัพท์จากอุปกรณ์อื่นๆ ได้ถึง 6 แหล่งคือ ขา $\overline{INT0}$ และ $\overline{INT1}$ (กำหนดให้ใช้ระดับพัลส์ หรือขอบขาพัลส์ก็ได้) วงจรนับและวงจรตั้งเวลาที่ 0และ1 (สำหรับ 8052 เพิ่มวงจรตั้งเวลา/วงจรนับตัวที่ 2 และตัวสุดท้ายจากพอร์ทอนุกรม เราสามารถกำหนดลำดับความสำคัญของอินเตอร์รัพท์ได้ 2 ระดับ โดยไม่ต้องอาศัยวงจรภายนอกช่วย แต่ละแหล่งของอินเตอร์รัพท์ 5 หรือ 6 แหล่ง นั้นจะกำหนดเป็นเวกเตอร์เฉพาะ (ตัวชี้แอดเดรส)

ดังนั้นเมื่อมีอินเตอร์รัพท์เข้ามาแล้ว ตัวโปรเซสเซอร์จะกระโดดไปที่ส่วนของโปรแกรมที่ทำงานตามวัตถุประสงค์ของอินเตอร์รัพท์เท่านั้น หลังจากเก็บข้อมูลต่างๆ ของโปรแกรมเคาน์เตอร์ลงในสแต็ดชุดคำสั่งทั้งหมดของตัวคอนโทรลเลอร์ในตระกูล MCS แสดงไว้ในภาคผนวก

3.3 ระบบไมโครคอนโทรลเลอร์

สิ่งที่มีความจำเป็นสำหรับไมโครคอนโทรลเลอร์ที่ชาญฉลาดได้แก่การควบคุมแบบ Real Time และ การประยุกต์ใช้งาน เป็นตัวเก็บข้อมูล ตัวควบคุมจะเป็นส่วนหนึ่งของระบบเครือข่ายที่เป็นเอกเทศน์ ซึ่งจะเชื่อมต่อกับคอมพิวเตอร์ศูนย์กลาง (Host Computer) ด้วยการสื่อสารแบบอนุกรม ตัวควบคุมจะถูกสร้างให้มีขนาดเล็ก มันอาจจะต้องมีไม่น้อยกว่าหนึ่งพันหน่วยสำหรับงานพิเศษ และ ราคาจะต้องถูกด้วย

ตระกูล 8051 ถูกเลือกด้วยเหตุผลดังต่อไปนี้

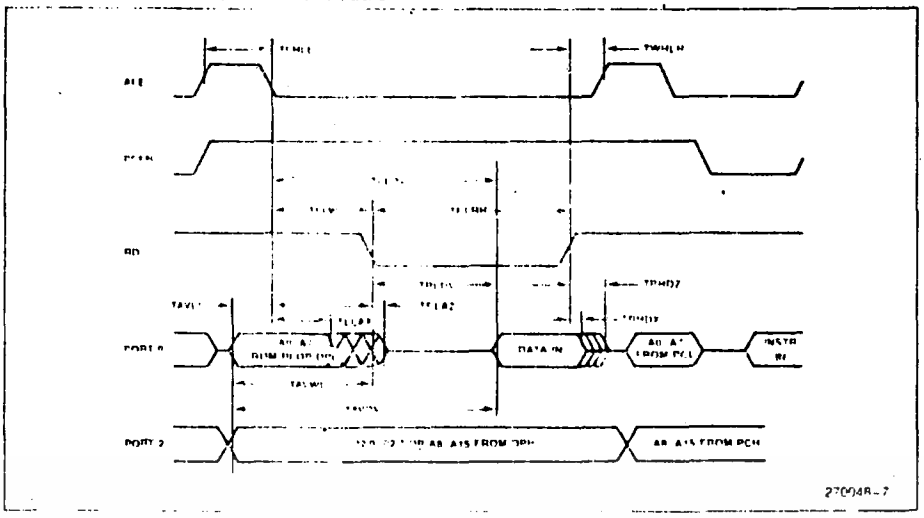
- ราคาถูก
- มีผู้ผลิตจำนวนมาก
- เทคโนโลยีที่ผลิตมีทั้งชนิด CMOS และ NMOS
- มีซอฟแวร์เป็นเครื่องมือช่วย และ ราคาไม่แพง
- สามารถใช้กับภาษาระดับสูงได้

สามข้อแรก เป็นส่วนสำคัญมากในการผลิตซึ่งมีจุดยืนในเรื่องราคา ซอฟแวร์จะเป็นตัวช่วยในการลดราคาในตอนต้น และ ยังจะทำให้โครงงานมีความสมบูรณ์ขึ้น

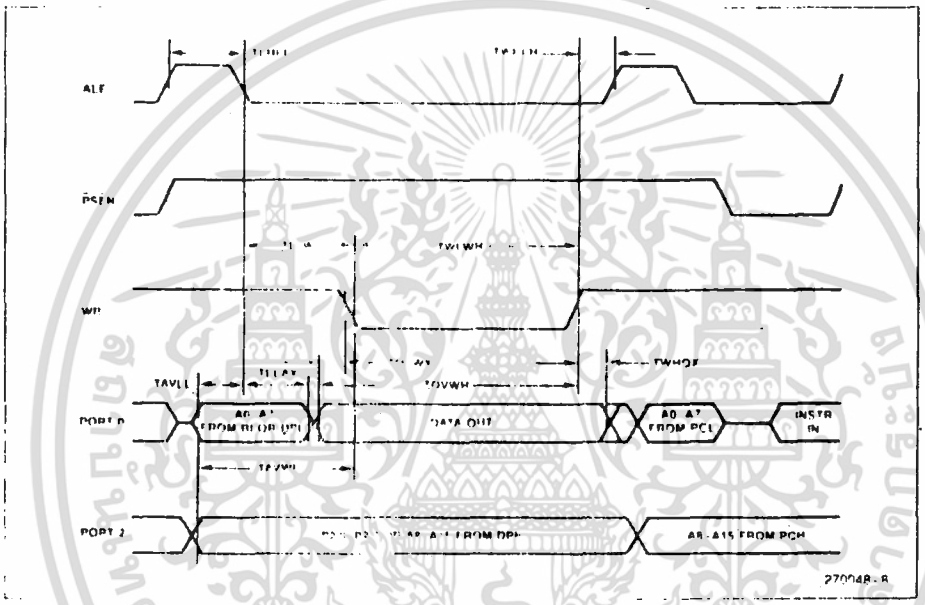
ความต้องการในการผลิตให้มีขนาดเล็ก และ ความจำเป็นในการเปลี่ยนโปรแกรม เพื่อที่จะสร้างโปรแกรมที่เหมาะสมกับงานเฉพาะอย่าง จึงจำเป็นต้องใช้หน่วยความจำภายนอกชนิดสามารถโปรแกรมได้ (EPROM) เพื่อที่จะเก็บโปรแกรมที่เราสร้างขึ้น ดังนั้นพอร์ท 0 (AD0-AD7) และ พอร์ท 2 (A8-A15) จะต้องใช้ในการติดต่อกับหน่วยความจำภายนอก (ROM) และจะไม่สามารถใช้เป็นส่วนรับและส่งข้อมูล (I/O) ได้อีก

เนื่องจากความเป็นไปได้อีกสิ่งหนึ่ง คือ การใช้ไมโครคอนโทรลเลอร์เป็นตัวเก็บข้อมูล แรมภายในที่ใช้อาจจะมากเกินจากหน่วยความจำภายใน ส่วนความจำภายนอกจะสามารถช่วยได้ในกรณีนี้ เมื่อเราต่อ แรม ใช้งาน พอร์ท 3 บิทที่ 6 ($\overline{WR}$) และบิทที่ 7 ($\overline{RD}$) จำเป็นสำหรับการติดต่อกับหน่วยความจำภายนอก จึงไม่สามารถใช้เป็นหน่วยรับส่งสัญญาณ (I/O)ได้อีก หน่วยความจำภายนอกตามมาตรฐานจะมี 28 ขา ซึ่งจะมีหน่วยความจำมากที่สุด 64 K ที่จะใส่ในชอคเก็ต 28 ขา

EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE



รูปที่ 3.8 แสดงตารางเวลาในการอ่านและเขียนข้อมูลภายนอก

ในทางการค้า EPROM ที่มีขายในท้องตลาดจะมีขนาดเป็น 2 เท่า โดยเริ่มจาก 2K EPROM ขนาดน้อยที่สุดที่ถูกเลือกคือ 8K ส่วนขนาดใหญ่ที่สุดได้แก่ 128K ตัวเลือกเหล่านี้จะบ่งบอกถึงจำนวนมากที่สุดเท่าที่มีอยู่จากผู้ผลิต และส่วนที่จะเริ่มผลิตจำนวนมาก ๆ

ข้อมูลอนุกรมจะถูกจัดการโดยวงจรพอร์ทอนุกรมภายในตัว 8051 อีกครั้งหนึ่ง ที่พอร์ท 3 ขาอินพุท/เอาท์พุทจะถูกใช้ ได้แก่ บิทที่ 3.0 (RXD) และ บิทที่ 3.1 (TXD) เราจะใช้พอร์ทที่หนึ่งทั้งพอร์ทเพื่อเป็นส่วนรับส่งข้อมูลทั่วไป และ พอร์ทที่ 3 ตั้งแต่ขาที่ 2-5 เป็นส่วนรับส่งข้อมูลทั่วไปหรือเพื่อใช้เป็นส่วนอินเทอร์รัพท์ภายนอกและส่วนรับสัญญาณเวลา (Timing Input)

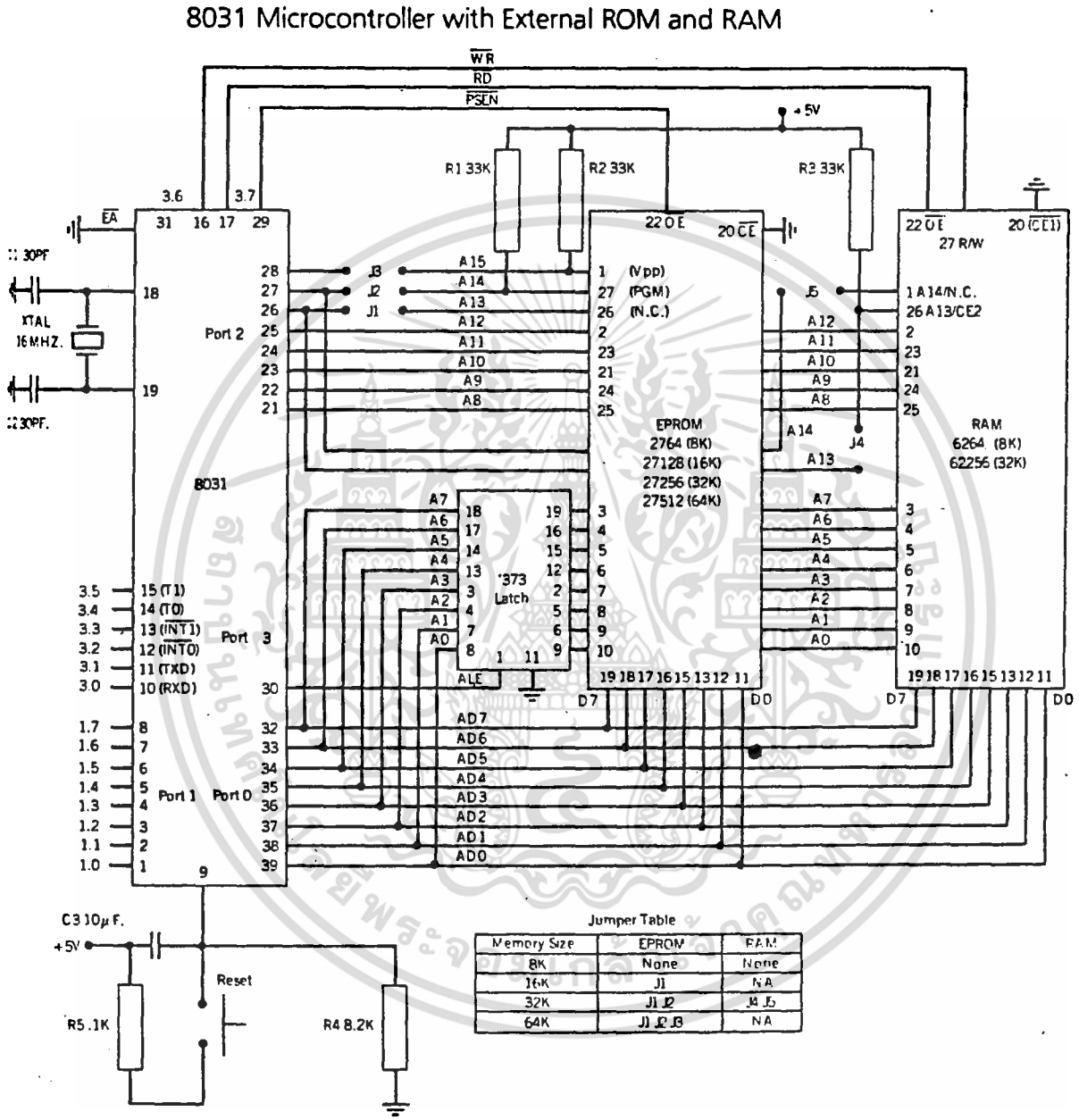
จะสังเกตได้ว่า มีข้อผิดพลาดของส่วนรับส่งข้อมูลเกิดขึ้น ขณะที่ใช้งานสลับหน้าที่กัน และควรจะคำนึงถึงข้อผิดพลาดที่จะมีผลต่อโปรแกรม

อย่างไรก็ตาม วิธีการแก้ไขจะไม่ยากจนเกินไป มี 2 วิธี ซึ่งใช้การขยายความสามารถในการรับและส่งข้อมูลของการประยุกต์ใช้งานคอมพิวเตอร์ ได้แก่ พอร์ท I/O และ Memory Mapped I/O

สุดท้ายเราเลือกคริสตอล 16 MHz จะทำให้อุปกรณ์มีความเร็วสูงสุดเท่าที่จะเป็นไปได้ และ จะทำให้ได้คุณสมบัติที่สมบูรณ์แบบ โดยสรุปเราจะได้ระบบดังต่อไปนี้

- 80c31 (ROM less) Microcontrollers
- 64 Kbytes of External EPROM
- 32 KBytes of External RAM
- 8 General-purpose I/O lines
- 4 General-purpose or Programmable I/O lines
- 1 Full-duplex serial port
- 16 MHz Crystal Clock

วงจรดังรูป 3.9 แสดงให้เห็นถึงวงจรขนาดเล็กที่สุด ซึ่งสามารถทำงานได้ง่าย และมีประสิทธิภาพสูงสุด

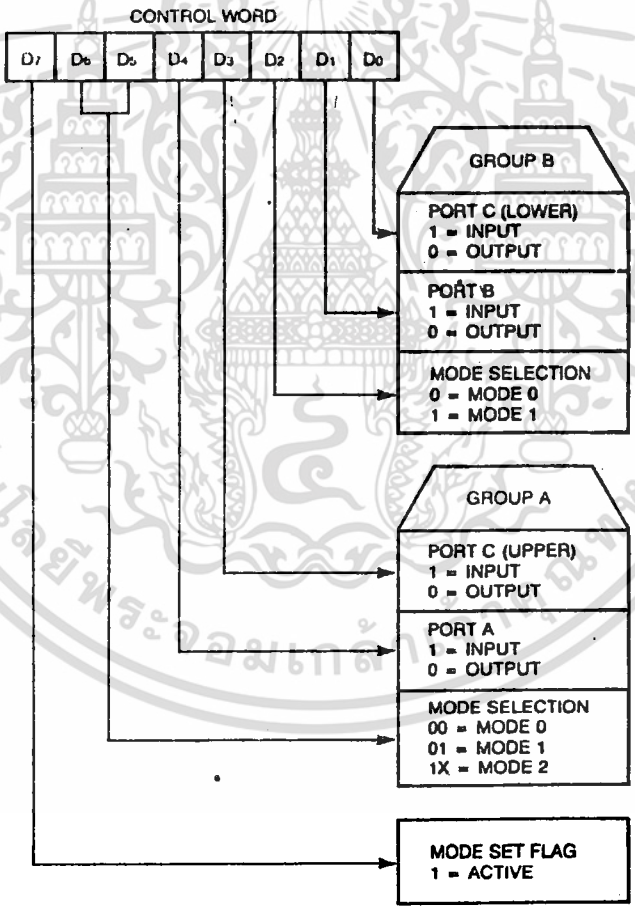


รูปที่ 3.9 แสดงระบบไมโครคอนโทรลเลอร์ที่ต่อกับแรมและรอมภายนอก

3.4 การขยายส่วนรับส่งข้อมูล

3.4.1 พอร์ทขนานที่สามารถโปรแกรมได้

ไอซี 8255 เป็นพอร์ทอินพุทเอาท์พุทแบบขนานซึ่งสามารถโปรแกรมได้ พอร์ทแต่ละพอร์ทมีขนาด 8 บิท มีอยู่ด้วยกัน 3 พอร์ท คือ พอร์ท A , B และ C ซึ่งพอร์ท C สามารถแยกเป็น 4 บิท 2 พอร์ทได้อีกตามที่ผู้ใช้โปรแกรมกำหนด วิธีการโปรแกรม 8255 และ รายละเอียดของขาทั้งหมด 40 ขา ดังรูป 3.10



รูปที่ 3.10 แสดงวิธีการโปรแกรม และโครงสร้างภายนอกของ 8255

ก่อนที่ 8255 จะถูกใช้งาน จะต้องโปรแกรมก่อนโดยการเขียนบิท ควบคุมไปที่รีจีสเตอร์ควบคุม พอร์ททั้งหมดของมัน ซึ่งสามารถถูกโปรแกรมได้โดย 8051 8255 ใช้แอดเดรส A0 และ A1 เพื่อที่จะกระทำกับรีจีสเตอร์ควบคุม และ พอร์ททั้งสาม สัญญาณควบคุม RD , WR และ CS จะถูกใช้งานโดยการถอดรหัสของระบบ 8051 ผลลัพธ์จากค่าสถานะของสัญญาณควบคุม และ สัญญาณแอดเดรส จะเป็นดังนี้

ตารางที่ 3.3 สถานะของสัญญาณควบคุม 8255

A1	A0	RD	WR	CS	ผลลัพธ์
0	0	L	H	L	อ่านค่าจากพอร์ท A
0	1	L	H	L	อ่านค่าจากพอร์ท B
1	0	L	H	L	อ่านค่าจากพอร์ท C
0	0	H	L	L	เขียนค่าไปยังพอร์ท A
0	1	H	L	L	เขียนค่าไปยังพอร์ท B
1	0	H	L	L	เขียนค่าไปยังพอร์ท C
1	1	H	L	L	เขียนค่าไปยังรีจีสเตอร์ควบคุม
X	X	X	X	H	บัสข้อมูลมีสถานะอิมพีแดนซ์สูง

3.4.1.1 การโปรแกรม 8255

ไบท์ควบคุมจะถูกเขียนไปยังรีจิสเตอร์ควบคุมโดยแต่ละบิทที่ใช้จะไปโปรแกรมลักษณะการทำงานของ 8255

เมื่อบิทที่ 7 เป็น 1 แล้ว พอร์ทที่ถูกโปรแกรมจะเป็นดังนี้

ตารางที่ 3.4 การโปรแกรม 8255

บิท	สถานะ	ผลลัพธ์
7	1	โปรแกรมพอร์ทสำหรับโหมด และส่วนรับส่งข้อมูล
7	0	เซต หรือ รีเซตบิทอิสระของพอร์ท C
6,5	00	เซตพอร์ท A และ C4-C7 เป็น I/O โหมด 0
6,5	01	เซตพอร์ท A และ C4-C7 เป็น I/O โหมด 1
6,5	10	เซตพอร์ท A และ C4-C7 เป็น I/O โหมด 2
4	0	เซตพอร์ท A เป็น เอาท์พุท
4	1	เซตพอร์ท A เป็น อินพุท
3	0	เซต C4-C7 เป็น เอาท์พุท
3	1	เซต C4-C7 เป็น อินพุท
2	0	เซตพอร์ท B และ C0-C3 เป็น I/O โหมด 0
2	1	เซตพอร์ท B และ C0-C3 เป็น I/O โหมด 1
1	0	เซตพอร์ท B เป็น เอาท์พุท
1	1	เซตพอร์ท B เป็น อินพุท
0	0	เซต C0-C3 เป็น เอาท์พุท
0	1	เซต C0-C3 เป็น อินพุท

3.4.1.2 โหมดการทำงาน ของ 8255

พอร์ท A และ พอร์ท C บน สามารถใช้ได้ 3 โหมด พอร์ท B และ พอร์ท C ล่างสามารถใช้ได้ 2 โหมด

โหมด 0 - I/O พื้นฐาน ข้อมูลที่ถูกเขียนไปยังพอร์ทจะถูกแลทช์ ข้อมูลที่อ่านจากพอร์ทเป็นข้อมูลที่อ่านจากขาอินพุท

โหมด 1 - I/O สโตรบ โหมดนี้เป็นโหมดแฮนด์เช็คโดยใช้พอร์ท A และพอร์ท B เป็นพอร์ทรับและส่งข้อมูล และ พอร์ท C เพื่อสร้างสัญญาณแฮนด์เช็คไปยังอุปกรณ์ที่ต่อกับพอร์ท A และ B และ สัญญาณอินเทอร์รัพท์ไปยัง Host Microcontrollers

โหมด 2 - I/O สโตรบ 2 ทิศทาง โหมดนี้จะคล้ายกับโหมด 1 สามารถใช้พอร์ท A เป็นบัสข้อมูลสองทิศทาง

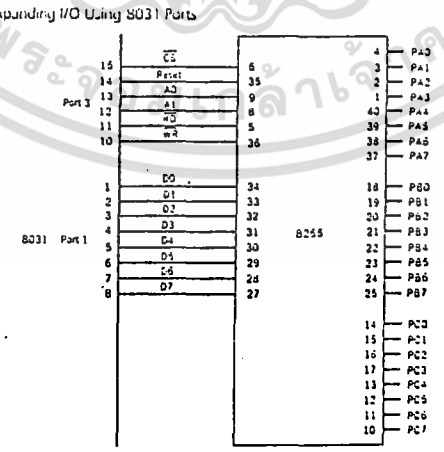
โหมด 1 และ 2 - ต้องเซตบิทอินเทอร์รัพท์ในพอร์ท C ซึ่งเป็นรีจิสเตอร์ข้อมูล โหมดเหล่านี้ ควรจะนำมาใช้กับอุปกรณ์ เช่น พรินเตอร์

3.4.1.3 สภาวะการรีเซต

เมื่อรีเซต พอร์ทข้อมูลแลทช์ทั้งหมด และ รีจิสเตอร์ควบคุมจะเคลียร์ให้เป็นศูนย์ พอร์ททั้งหมดจะอยู่ในโหมดอินพุท

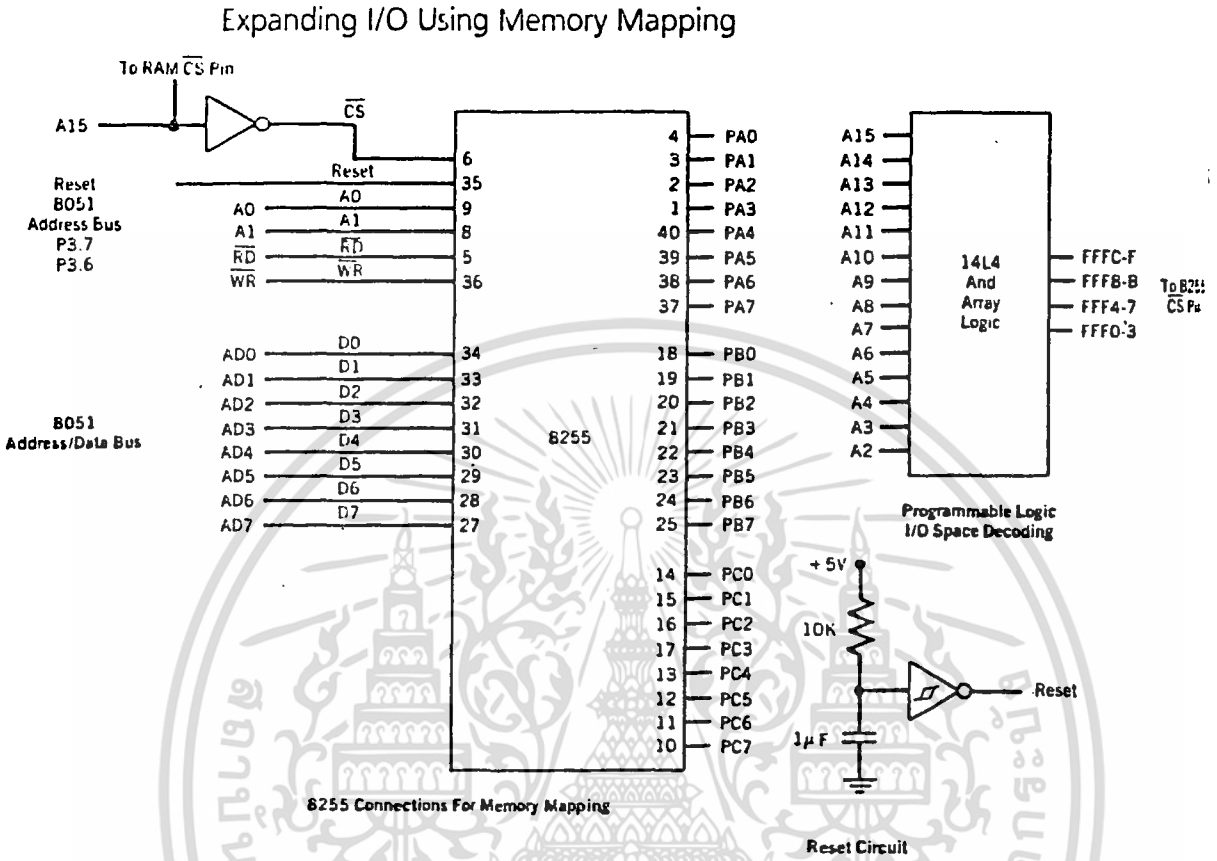
3.4.2 การต่อ 8255 กับ ไมโครคอนโทรลเลอร์โดยตรง

พอร์ท 1 และ 3 สามารถใช้เป็นส่วนควบคุมขนาดเล็ก และ บัสข้อมูล 2 ทิศทาง บัสข้อมูลสามารถจะต่อกับวงจรภายนอกเพิ่มเพื่อช่วยในการขยายส่วนอินพุท และ เอาท์พุท



รูปที่ 3.11 การขยายพอร์ท รับและส่งข้อมูลด้วยพอร์ท 8031

3.4.3 การขยายพอร์ทโดยวิธี Memory Mapped I/O



รูปที่ 3.12 Memory Mapped I/O

8255 สามารถจะนำมาใช้เพิ่มในส่วนของหน่วยความจำและที่ว่าง ดังในรูปที่ 3.12 ในการออกแบบที่ใช้หน่วยความจำ 32K จากที่สามารถใช้ได้ถึง 64K ดังนั้นจะมีที่ว่างด้านบนถึง 32K พอร์ทชิพสามารถจะใช้ขาแอดเดรส A15 ซึ่งปกติจะมีสถานะ "1" (8000H หรือมากกว่านั้น) และ แรม 32K จะอ้างแอดเดรสได้เมื่อไรก็ตามที่ A15 มีสถานะ "0" (7FFFH หรือ น้อยกว่านั้น) ลักษณะการถอดรหัส (decode) แบบนี้ต้องการเพียงเพิ่มอินเวอร์เตอร์เพื่อที่จะถอดรหัสหน่วยความจำที่ว่างสำหรับ แรม และหน่วยรับและส่งข้อมูล

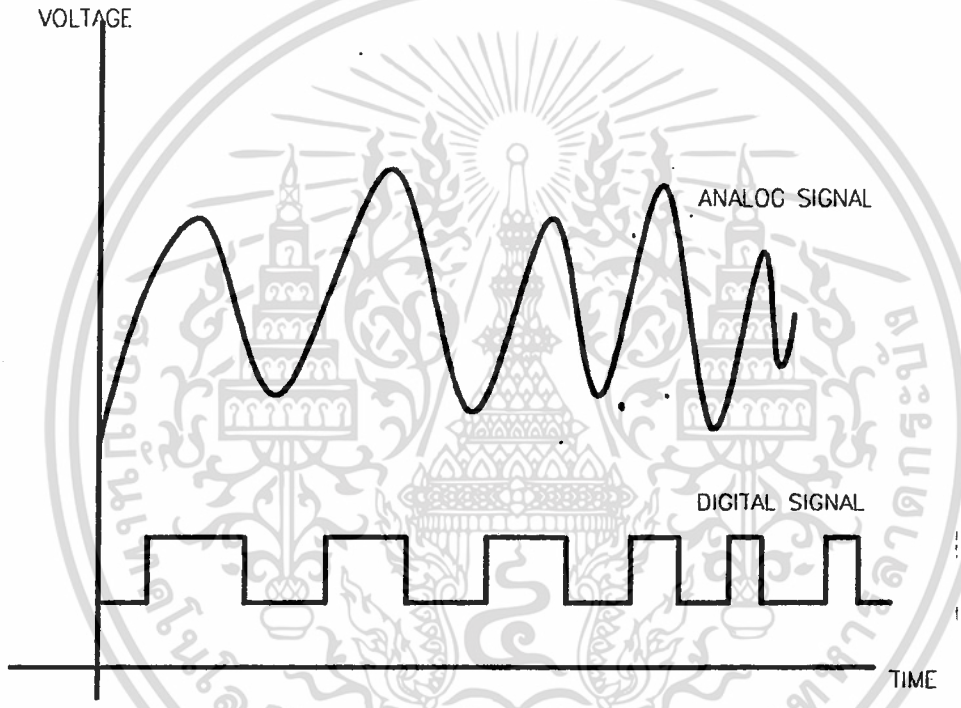
ถ้าจำเป็นต้องมีการเพิ่มแรม ในการออกแบบ ลักษณะการถอดรหัสหน่วยความจำเป็นต้องใช้ตัวถอดรหัส (Decoder) แบบอาเรย์ที่สามารถโปรแกรมได้ เพื่อจะสงวนบางส่วนของหน่วยความจำให้ว่างเพื่อใช้เป็นพอร์ทของส่วนรับและส่งข้อมูล ดังรูปที่ 3.12 แสดงการออกแบบซึ่งเพิ่ม memory mapped พอร์ทบิทถึง 3 ตัวที่ตำแหน่ง FFF0H-FFF3H , FFF4H-FFF7H , FFF8H-FFFBH , FFFCH-FFFFH แรมมีแอดเดรสจาก 0000H - FFEFH

Memory Mapped I/O มีข้อดีที่ไม่จำเป็นต้องใช้พอร์ทใด ๆ ของ 8051 แต่ก็มีข้อเสียคือ ทำให้เกิดการสูญเสียหน่วยความจำที่ว่างสำหรับแรม ซึ่งถูกใช้ไปเป็นส่วนรับส่งข้อมูล หรือ ต้องเพิ่มชิพสำหรับการถอดรหัสหน่วยความจำ ตามข้อจำกัดของแอดเดรสว่างที่สูญเสียไปของแรม การโปรแกรมในระดับสูง เช่นเดียวกับพอร์ทรับส่งข้อมูลเนื่องจากความไม่เหมาะสมของคำสั่ง MOVX อาจจะถูกใช้ในการกระทำกับ Memory Mapped I/O

บทที่ 4

การแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

โลกของการศึกษาทางไฟฟ้า จะแบ่งการพิจารณา สัญญาณทางไฟฟ้าเป็น 2 ลักษณะ ได้แก่ สัญญาณอนาลอก และ สัญญาณดิจิตอล สัญญาณอนาลอกเป็นสัญญาณที่มีความเป็นเชิงเส้น คือ จะมีการเปลี่ยนแปลงสัญญาณ เมื่อเทียบกับเวลาเป็น ลักษณะเชิงเส้น ส่วนสัญญาณดิจิตอล จะมีลักษณะการเปลี่ยนแปลงสัญญาณเพียง 2 สถานะคือ สถานะ "1" และ สถานะ "0"

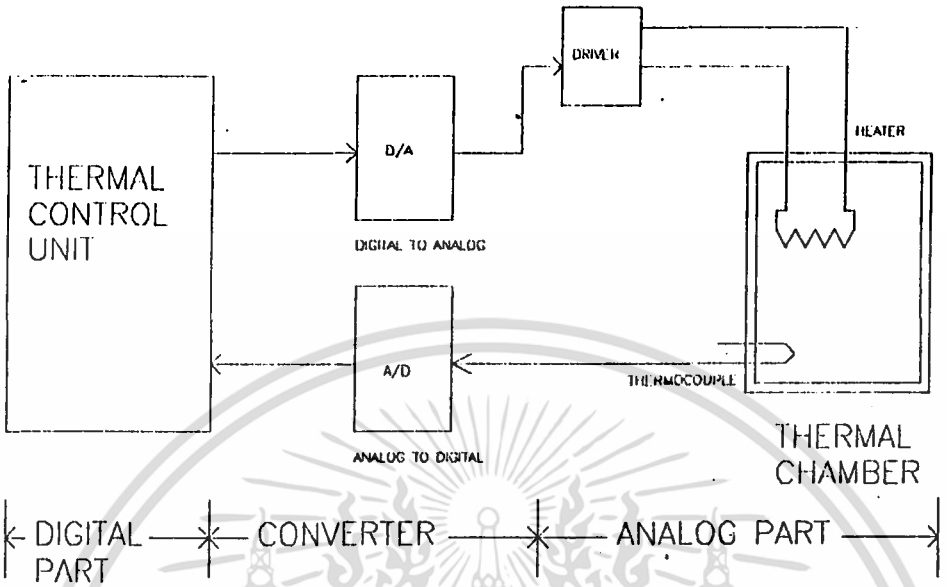


รูปที่ 4.1 แสดงลักษณะของสัญญาณอนาลอกและสัญญาณดิจิตอล

4.1 การแปลงสัญญาณ

การแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก และการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลเป็นขบวนการที่มีความจำเป็นอย่างมากในวงจรไฟฟ้าทั่วไป เนื่องจากอุปกรณ์ต่าง ๆ ที่ใช้งานจะมีลักษณะการทำงานที่เกี่ยวข้องกับทั้งสองสัญญาณ ตัวอย่างเช่น ในระบบควบคุมอุณหภูมิอัตโนมัติ อุณหภูมิที่ต้องการวัดจะถูกเปลี่ยนเป็นสัญญาณไฟฟ้า (ซึ่งเป็นสัญญาณอนาลอก) สัญญาณที่ได้จะถูกแปลงเป็นสัญญาณดิจิตอลเพื่อประมวลผลด้วยระบบ

ไมโครโปรเซสเซอร์ จากนั้นไมโครโปรเซสเซอร์ก็จะส่งสัญญาณมาควบคุมตัวทำความร้อน ให้เพิ่มอุณหภูมิ หรือ ลดอุณหภูมิ ตามที่ผู้ใช้ได้โปรแกรมไว้



รูป 4.2 แสดงลักษณะของการแปลงสัญญาณ

การแปลงสัญญาณจะต้องใช้รหัส ซึ่งจะแสดงสถานะเปรียบเทียบกันระหว่างสัญญาณทั้งสองรหัสที่ใช้ในการแปลงรหัสไบนารี ดังตารางที่ 4.1

ตารางที่ 4.1 แสดงจำนวนเต็มและเศษส่วนเปรียบเทียบรหัสไบนารี

Decimal Fraction	Code: Binary Fraction	MSB (×1/2)	Bit 2 (×1/4)	Bit 3 (×1/8)	Bit 4 (×1/16)	Binary Integer	Decimal Integer
0	0.0000	0	0	0	0	0 0 0 0	0
1/16 = 2^{-4} (LSB)	0.0001	0	0	0	1	0 0 0 1	1
2/16 = 1/8	0.0010	0	0	1	0	0 0 1 0	2
3/16 = 1/8 + 1/16	0.0011	0	0	1	1	0 0 1 1	3
4/16 = 1/4	0.0100	0	1	0	0	0 1 0 0	4
5/16 = 1/4 + 1/16	0.0101	0	1	0	1	0 1 0 1	5
6/16 = 1/4 + 1/8	0.0110	0	1	1	0	0 1 1 0	6
7/16 = 1/4 + 1/8 + 1/16	0.0111	0	1	1	1	0 1 1 1	7
8/16 = 1/2 (MSB)	0.1000	1	0	0	0	1 0 0 0	8
9/16 = 1/2 + 1/16	0.1001	1	0	0	1	1 0 0 1	9
10/16 = 1/2 + 1/8	0.1010	1	0	1	0	1 0 1 0	10
11/16 = 1/2 + 1/8 + 1/16	0.1011	1	0	1	1	1 0 1 1	11
12/16 = 1/2 + 1/4	0.1100	1	1	0	0	1 1 0 0	12
13/16 = 1/2 + 1/4 + 1/16	0.1101	1	1	0	1	1 1 0 1	13
14/16 = 1/2 + 1/4 + 1/8	0.1110	1	1	1	0	1 1 1 0	14
15/16 = 1/2 + 1/4 + 1/8 + 1/16	0.1111	1	1	1	1	1 1 1 1	15

Table 7.1 Integer and fractional binary codes.

นอกจากนี้ยังใช้รหัสอื่น ๆ อีก เช่น รหัส BCD (Binary Code Decimal BCD) รหัสเกรย์ (Gray Code) เป็นต้น

4.2 คุณลักษณะของการแปลงสัญญาณ

การใช้งานตัวแปลงสัญญาณ (Converter) เราจำเป็นต้องรู้ถึง คุณลักษณะบางประการของมัน ซึ่งจะช่วยให้เราสามารถตัดสินใจเลือกใช้ วงจรแปลงสัญญาณ ที่เหมาะสมกับงานของเรา

4.2.1 ความถูกต้องสัมบูรณ์(Absolute Accuracy)เป็นค่าของผลต่างระหว่างค่าของสัญญาณอนาลอกจริงกับค่าของสัญญาณดิจิตอลที่สร้างขึ้น

4.2.2 ออโตเมติกซีโร (Automatic Zero)เป็นการปรับค่าความคลาดเคลื่อนเพื่อชดเชยจากค่าผิดพลาดของการแปลงสัญญาณ (Drift error)

4.2.3 ค่าเต็มสเกล(Full scale)ช่วงค่าสูงสุดของสัญญาณอนาลอกที่จะแปลงเป็นสัญญาณดิจิตอล เช่น $\pm$ 4.098 V

4.2.4 ความละเอียดในการแปลงสัญญาณ (Resolustion) เป็นค่าซึ่งแสดงถึงการแปลงสัญญาณจะสามารถแยกความแตกต่างของสัญญาณที่ใกล้เคียงกันได้มากที่สุด เช่น ตัวแปลงสัญญาณที่มีจำนวน 12 บิทจะมีค่าความละเอียดสูงกว่า ตัวแปลงสัญญาณที่มีจำนวน 4 บิท

4.2.5 อัตราการแปลงสัญญาณ (Rating) เป็นค่าซึ่งแสดงถึงความเร็วในการแปลงสัญญาณภายในหนึ่งหน่วยเวลา

4.2.6 ค่าความผิดพลาดจากอุณหภูมิ (Temperature error) ค่าที่แสดงถึงความผิดพลาดเมื่อมีการเปลี่ยนแปลงอุณหภูมิ

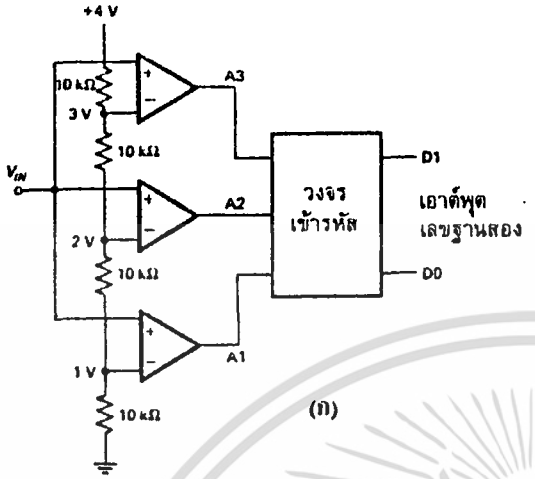
4.3 แบบต่าง ๆ ของการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

วงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลมีมากมายหลายแบบ ดังต่อไปนี้

4.3.1 แบบใช้วงจรเปรียบเทียบ (Flash A/D Converter)

วงจรแปลงแบบนี้ใช้หลักการของการเปรียบเทียบสัญญาณจากวงจรเปรียบเทียบที่ต่อขนานกัน

สัญญาณอนาลอกจะเข้ามายังวงจรเปรียบเทียบ แล้วผ่านไปยังวงจรเข้ารหัส ดังรูปที่ 4.3



(ก)

แรงดันอินพุต V_{in} (โวลต์)	เอาต์พุตของวงจรเปรียบเทียบ			เอาต์พุตเลขฐานสอง	
	A1	A2	A3	D1	D0
0 - 1	0	0	0	0	0
1 - 2	1	0	0	0	1
2 - 3	1	1	0	1	0
3 - 4	1	1	1	1	1

(ข)

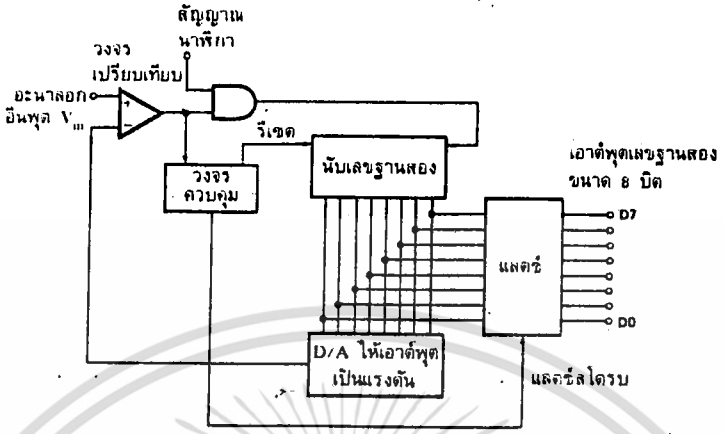
รูปที่ 4.3 ก.แสดงการต่อวงจร Parallel comparator A/D converter
ข.ตารางแสดงความสัมพันธ์ระหว่างแรงดันอินพุทที่เป็นอนาลอกกับเอาท์พุท ที่เป็นดิจิตอล

ดังนั้นวงจร A/D Converter แบบนี้ จึงเป็นวงจรที่มีความเร็วในการแปลงสัญญาณสูงมาก แต่ก็ยังมีข้อเสีย คือเมื่อต้องการสัญญาณที่มีความละเอียดสูง วงจรจะมีขนาดใหญ่ เช่น ถ้าต้องการความละเอียด 8 บิท จะต้องใช้วงจรเปรียบเทียบถึง 255 ตัว (จำนวนของวงจรเปรียบเทียบเท่ากับ 2^n-1 โดยที่ n เป็นจำนวนบิทของข้อมูล)

4.3.2 แบบใช้วงจรนับและวงจร D/A Converter ประกอบกัน

วงจร A/D Converter ชนิดนี้จะใช้วงจร D/A Converter เป็นตัวเปลี่ยนสัญญาณดิจิตอล (ที่ได้จากวงจรนับ)ไปเป็นสัญญาณอนาลอก เพื่อนำไปเปรียบเทียบกับสัญญาณอินพุท

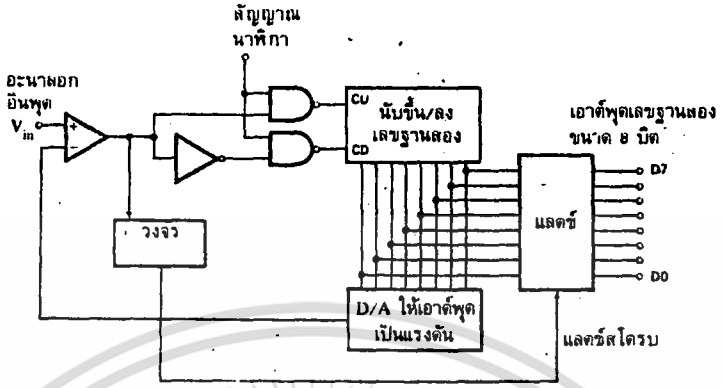
4.3.2.1 แบบวงจรนับเดี่ยว (Single counter)



รูปที่ 4.4 วงจร A/D Converter แบบวงจรนับเดี่ยวที่สร้างขึ้นโดยวงจรนับขั้น และ วงจร A/D Converter

สัญญาณแรมป์เชิงเส้น(Linear ramp) จะประกอบด้วยสัญญาณขั้นบันไดเล็กๆ จำนวนมากที่เกิดจากการต่อเอาท์พุทของ วงจรนับ เข้ากับ วงจรแปลง D/A Converter โดยขนาด ของขั้นบันไดแต่ละขั้นขึ้นอยู่กับ จำนวนบิท หรือ ความละเอียดของวงจร D/A Converter รูปที่ 4.4 แสดงการกำเนิดสัญญาณแรมป์เดี่ยว ด้วยวงจรนับ และวงจร D/A Converter (แทนวงจรอินทิเกรเตอร์)

4.3.2.2 แบบแทร็กกิ้ง (Tracking A/D Converter)



รูปที่ 4.5 วงจร A/D Converter ที่สร้างขึ้นจากวงจรนับขึ้น/ลง และวงจร D/A Converter

การทำงานของวงจร A/D Converter ชนิดนี้จะคล้ายกับวงจรนับเดี่ยว แต่จะไม่เริ่มนับที่ศูนย์ จะเริ่มนับขึ้นหรือลงจากค่าสุดไปยังค่าใหม่ แล้วแต่ค่าแรงดันอินพุทในรอบใหม่มีค่าสูงกว่าหรือต่ำกว่าค่าที่แล้วข้อดีของวงจร A/D Converter แบบนี้คือ ทำงานได้เร็วขึ้น

4.3.3 แบบใช้การประมาณค่า (Successive approximation)

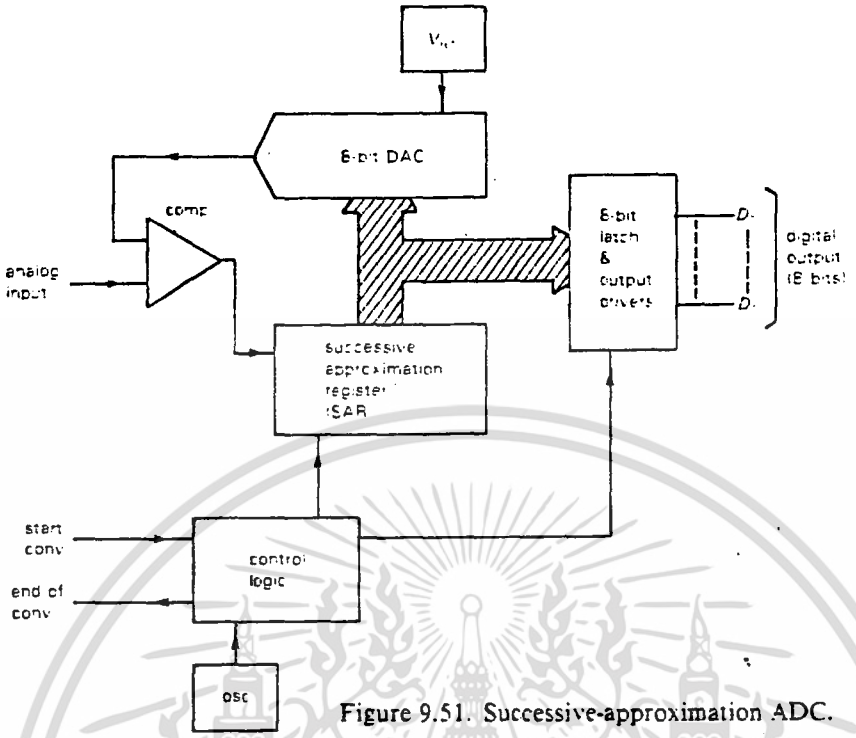


Figure 9.51. Successive-approximation ADC.

รูปที่ 4.6 วงจรแปลงสัญญาณ A/D Converter แบบการประมาณค่า

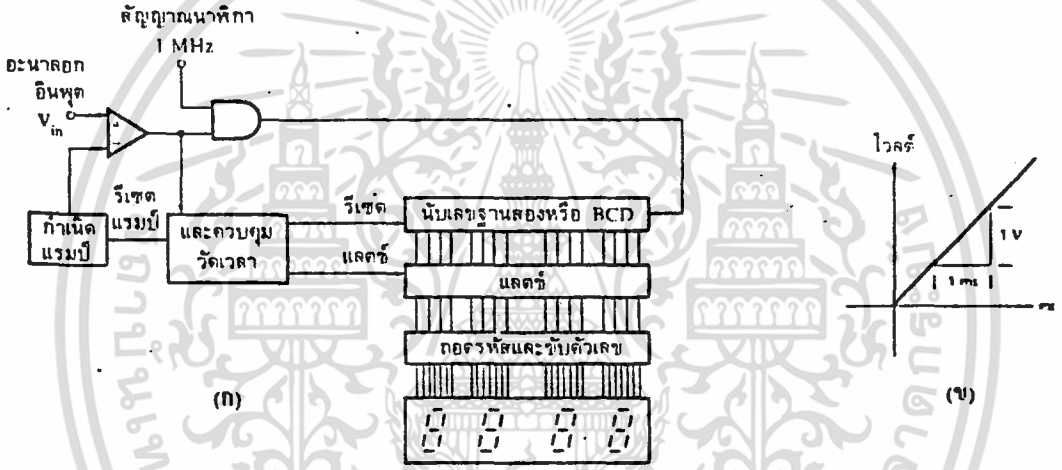
วงจร A/D Converter แบบนี้มีข้อดีคือ มีความละเอียดสูง ซึ่งสามารถกำหนดได้จากสัญญาณ พัลส์นาฬิกา เช่น วงจรแปลงขนาด 8 บิทต้องใช้พัลส์สัญญาณ 8 ลูก ในขณะที่วงจรนับต้องใช้พัลส์ถึง 256 ลูกหัวใจสำคัญของวงจร A/D Converter ชนิดนี้ได้แก่ Successive Approximation Register (SAR) เมื่อเริ่มเปลี่ยนสัญญาณ พัลส์ลูกแรกจะส่งบิทที่มีนัยสำคัญสูงสุดไปยัง วงจร D/A Converter โดยSARจะรอสัญญาณจากวงจรเปรียบเทียบ ซึ่งจะตรวจสอบว่าเอาท์พุทของวงจร D/A Converter มากหรือน้อยกว่าแรงดันดินพุท V_{in} ถ้าเอาท์พุทของวงจรเปรียบเทียบมี สถานะ "1" เอาท์พุทของวงจร D/A Converter จึงต่ำกว่า V_{in} SAR จะเก็บบิทที่มีนัยสำคัญสูงสุดไว้ ถ้าเอาท์พุทของวงจรเปรียบเทียบมีสถานะ "0" เอาท์พุทของ D/A Converter จึงมากกว่า V_{in} SAR จะรีเซต บิทที่มีนัยสำคัญสูงสุดนั้น

พัลส์ลูกต่อมาก็ทำเช่นเดียวกัน โดยบิทที่ได้คือบิทที่มีนัยสำคัญรองลงมา SAR จะทำงานแบบนี้ ไปจนถึงบิทที่มีนัยสำคัญต่ำสุด เมื่อแต่ละบิทใช้สัญญาณนาฬิกาครบทุกบิทแล้ว SAR จะส่งสัญญาณให้หยุดการแปลง (End of Conversion,EOC) ออกไป สัญญาณนี้

จะเป็นตัวบอกว่ามีข้อมูลอยู่ที่อินพุทครบแล้ว แต่ถ้าสัญญาณนี้ถูกต่อไปยังอินพุท ที่เป็นจุดเริ่มการเปลี่ยนแปลงสัญญาณ การเปลี่ยนแปลงสัญญาณก็จะเกิดขึ้นอย่างต่อเนื่อง วงจรดังในรูปที่ 4.6 ออปแอมป์จะเป็นตัวเปลี่ยนกระแสไปเป็นแรงดัน วงจรแบบนี้สามารถรับสัญญาณอนาลอกที่เป็น รูปซายน์ (มีค่าแรงดัน จาก -5 ถึง +5 V) ได้ และยังมีความละเอียดสูง

4.3.4 แบบที่ใช้วงจรอินทิเกรต

4.3.4.1 แบบสโลปเดี่ยว (Single ramp หรือ Single slope)



รูปที่ 4.7 วงจรA/D Converterแบบสโลปเดี่ยว

ก)แสดงบล็อกไดอะแกรม

ข)ความชันของสัญญาณแรมป์

วงจร A/D Converter แบบนี้แสดงดังรูปที่ 4.7 ประกอบด้วยวงจรกำเนิดสัญญาณแรมป์, วงจรเปรียบเทียบ, วงจรนับ BCD หรือ นับเลขฐานสอง เมื่อเริ่มทำการเปลี่ยนสัญญาณ สัญญาณแรมป์และวงจรนับจะถูกรีเซตให้เป็นศูนย์ แรงดันอนาลอกถูกป้อนไปยังวงจรเปรียบเทียบทางขาอินพุทแบบไม่ย้อนกลับ วงจรเปรียบเทียบก็จะให้เอาท์พุทเป็นสถานะ "1" ทำให้แอนด์เกต ปล่อยสัญญาณนาฬิกา ผ่านไปยังวงจรนับได้ และทำให้เริ่มเกิดสัญญาณแรมป์ จนกระทั่งสัญญาณแรมป์มีแรงดันเพิ่มขึ้นมากกว่า แรงดันอินพุท เอาท์พุท

ของวงจรเปรียบเทียบจะเปลี่ยนสถานะเป็น "0" ปิดแอนด์เกต ทำให้ไม่มีสัญญาณผ่านไปยังวงจรนับ วงจรนับจะหยุดนับและค่าที่นับได้ จะถูกแลตช์ไว้ จากนั้นจึงทำการรีเซ็ตวงจรนับและวงจรสร้างสัญญาณแรมป์

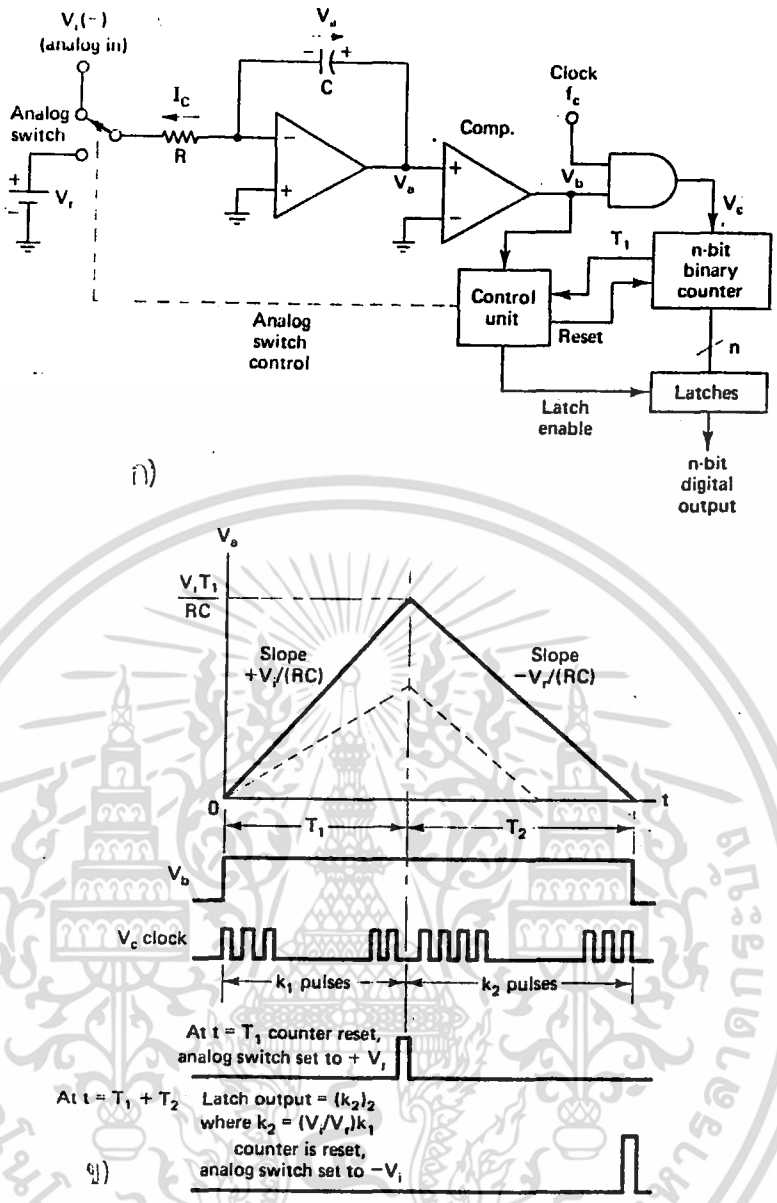
วงจรนี้เป็นหลักการเบื้องต้นของดิจิตอลโวลต์มิเตอร์ ซึ่งถ้าใช้วงจรนับเลขฐานสอง แทนแบบ BCD เอาท์พุทก็จะอ่านได้ค่าเลขฐานสองโดยตรง ข้อเสียของวงจร A/D Converter ชนิดนี้คือจะไม่สามารถใช้กับงานที่ต้องการความถูกต้องสูงได้ เนื่องจากวงจรมีผลตอบสนองต่ออุณหภูมิ

4.3.4.2 แบบสโลปคู่ (Dual-Slope A/D converter)

วงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลแบบสโลปคู่ อินทิเกรชันนี้ ข้อมูลที่แปลงได้เป็นจำนวนบิท หรือแบบ BCD การแปลงสัญญาณแบบสโลปคู่จะไม่เปรียบเทียบแรงดันที่ต้องการตรงๆ แต่จะนำไปเปรียบเทียบกับค่าแรงดันอ้างอิงค่าหนึ่ง โดยใช้เวลาเป็นตัวกำหนดการทำงาน การทำงานจะเริ่มต้นจาก สวิทช์ S_1 รับอินพุทเข้ามาสู่วงจรอินทิเกรชันด้วยเวลาคงที่ T_1 เมื่อครบเวลา T_1 สวิทช์ S_1 จะถูกสับกลับมาที่แรงดันอ้างอิงที่มีศักย์ตรงข้ามกับแรงดันอินพุท วงจรนับจะเริ่มต้นนับเวลาไปจนกระทั่ง เอาท์พุทของวงจรอินทิเกรเตอร์ โดยที่ T_1 เป็นเวลาที่กำหนดคงที่ และ T_2 เป็นเวลาแปรตามแรงดันอินพุท

ซึ่งมีความสัมพันธ์

$$T_2 = T_1 * V_{in}/V_{ref} \quad \text{.....(4.1)}$$

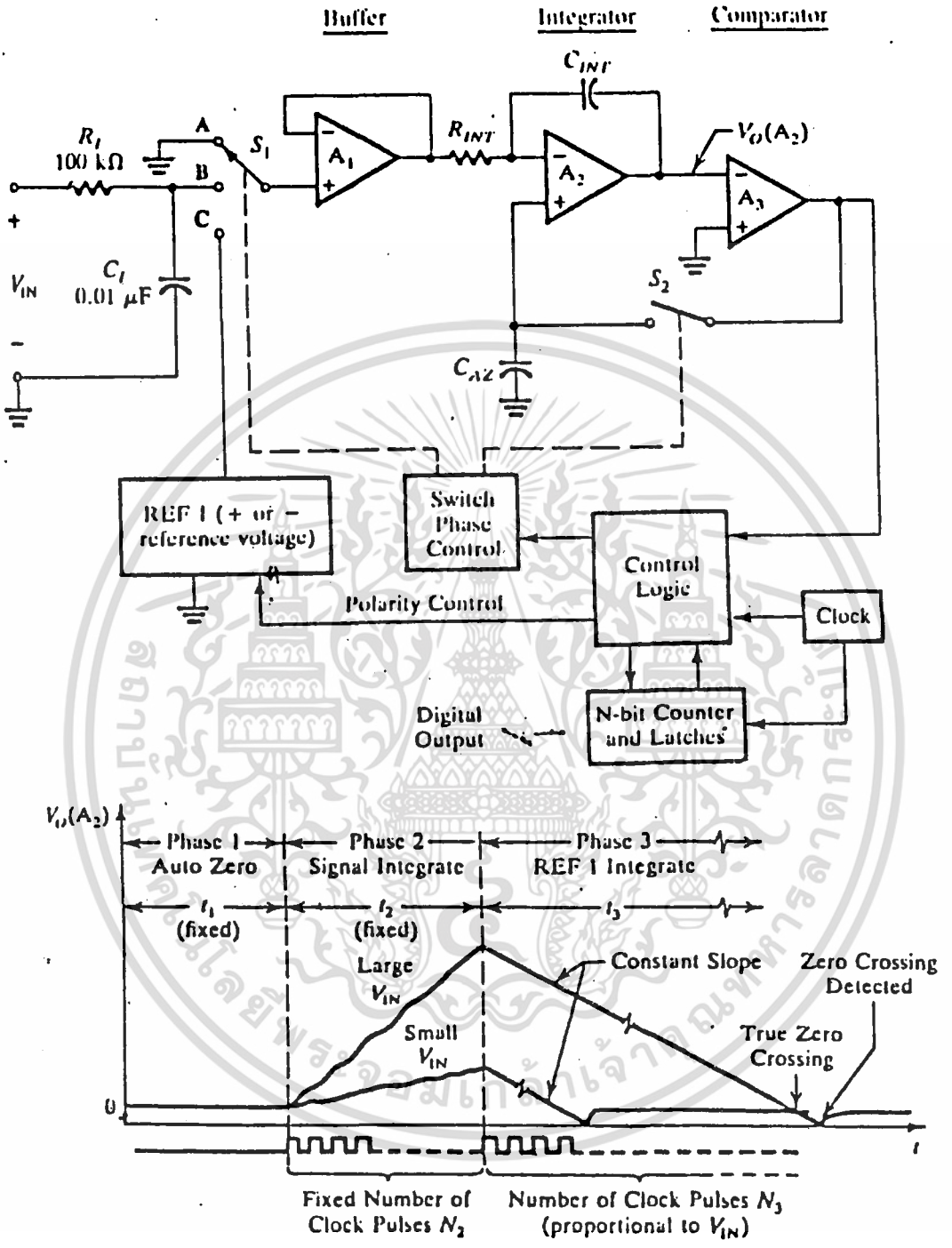


รูปที่ 4.8 วงจร A/D Converter แบบสโลปคู่

ก) แสดงบล็อกไดอะแกรม

ข) เอาท์พุทของวงจรอินทิเกรเตอร์เมื่อเทียบกับเวลา

ผลของการนับด้วยวงจรนับในเวลา T_2 จะอยู่ในรูปของจำนวนพัลส์ต่อโวลต์ ส่วนเอาท์พุทนั้นจะขึ้นอยู่กับวงจรรวมแต่ละตัว ส่วนแสดงผล จะแสดงผลของการนับออกเป็นค่าโวลต์ หรือเป็นจำนวนบิท ถ้าผลการนับแสดงออกเป็นโวลต์ จะอยู่ในลักษณะของจำนวนหลัก เช่น สามหลักครึ่ง สี่หลักครึ่ง ส่วนข้อมูลที่เป็นจำนวนบิท หรือเลขฐานสอง จะเป็น 8 บิท , 12 บิท



รูปที่ 4.9 ก) แสดงวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลแบบสโลปคู่ พร้อมส่วนออโต-ซีโร

ข) แสดงชนิดของสัญญาณแรงดันไฟฟ้าที่เอาท์พุทออปแอมป์ A_2 สำหรับสัญญาณอินพุทเป็นลบ และสัญญาณอ้างอิงเป็นบวก

จากวงจรสโลปคู่ที่ได้กล่าวมาแล้วได้มีการพัฒนาวงจรดังกล่าวโดยเพิ่มส่วนออโต-ซีโร (Auto-Zero)

ซึ่งใช้ออปแอมป์เป็นตัวเปรียบเทียบแรงดันแบบ ซีโร-ครอสซิ่ง มีการชาร์จสัญญาณจากตัวเก็บประจุใน วงจรอินทิเกรเตอร์ มีส่วนที่เป็นสัญญาณนาฬิกา และส่วนที่เป็นวงจรนับ ส่วนที่มีการเพิ่มเติมเข้าไปใหม่ คืออนาลอกสวิทช์และวงจรออโต-ซีโร เพื่อลดแรงดันออฟเซทดีซี ที่เกิดจากวงจรอินทิเกรเตอร์ และวงจรเปรียบเทียบแรงดัน วงจรแปลงสัญญาณแบบนี้สามารถแปลงสัญญาณ มีค่าความถูกต้องถึงหกหลักครึ่ง (20 บิท) แต่ใช้เวลานานถึง 100 มิลลิวินาที

จากรูป 4.9 ก)และ ข) ในส่วนของไซเคิลแรก (เฟส1) เริ่มจากส่วนอินพุทของวงจรขยายสัญญาณแรงดันไฟฟ้าแบบเป็นบัฟเฟอร์ต่อกับกราวด์ (สวิทช์ S_1 อยู่ที่ตำแหน่ง A) ขณะที่สวิทช์ S_2 จะปิดวงจรเพื่อชาร์จสัญญาณเข้าตัวเก็บประจุออร์โต-ซีโร C_z ด้วยเหตุนี้ผลการคำนวณค่าแรงดันออฟเซทดีซีที่ผิดพลาดของออป์แอมป์ A_1 และ A_2 และวงจรเปรียบเทียบแรงดัน A_2 จะถูกเก็บไว้ใน C_z เนื่องจากวงจรเปรียบเทียบแรงดัน A_3 รวมอยู่ในลูป จะแก้ไขแรงดันออฟเซทที่เป็นสัญญาณรบกวนที่เกิดขึ้นเองในวงจร โดยปกติ จะมีค่าประมาณ 10 ไมโครโวลต์ เวลา t ที่กำหนดค่าไว้แล้วเป็นเวลาที่ใช้ในโหมดออร์โต-ซีโร แต่ในกรณีอื่นอาจจะถูกเซทให้มีค่ามากกว่า 20 เท่าของเวลาการชาร์จ ที่มีผลต่อการรวมสัญญาณที่เลื่อนไปเทอมอื่น ๆ

ส่วนเริ่มของเฟสที่สอง ลูปของออร์โต-ซีโร จะเปิดออก (สวิทช์ S_2 เปิดออก) และสวิทช์อินพุท S_1 จะต่อเข้ากับจุด B (ต่อกับสัญญาณอินพุท V_{in}) วงจรอินทิเกรเตอร์ A_2 จะอินทิเกรทสัญญาณอินพุทในเวลา t_2 ที่กำหนดค่าไว้แล้ว ในขณะเดียวกัน วงจรนับจะถูกทริกที่จุดเริ่มต้นของเฟสสอง และจะมีค่าเป็นไปตามจำนวนเลขที่ถูกกำหนดไว้ของพัลส์สัญญาณนาฬิกา ที่จุดสุดท้ายของ t_2 อยู่ในตำแหน่งจุด C ซึ่งถูกกำหนดไว้กับแหล่งจ่ายสัญญาณแรงดันอ้างอิง จะมีลักษณะสัญญาณที่ตรงข้ามกับสัญญาณอินพุท ด้วยเหตุนี้ระหว่างที่อยู่ในเฟสสอง ความสำคัญของการคำนวณ V_{in} จะมีค่าที่เซ็ตค่าแรงดันอ้างอิง ให้มีความสำคัญเดียวกัน

เนื่องจากแรงดันอ้างอิงได้ถูกกำหนดค่าไว้แล้ว สโลปของสัญญาณเอาท์พุทของออปแอมป์ A_2 จะมีผลดีอินทิเกรตสัญญาณอินพุทกลับไปยังศูนย์ตามรูป 3.8 สิ่งสำคัญมากที่ใช้ในส่วนของแรงดันอ้างอิง ต้องมีค่ามากกว่าค่ามากที่สุดของ สัญญาณอินพุท

วงจรเปรียบเทียบแรงดัน A_3 เป็นแบบ ชีโร-ครอสชิ่ง สัญญาณอินพุทจนจบสามเฟส และหยุดวงจรนับ และรีเซตที่จุดเริ่มต้นของเฟสสาม ด้วยเหตุนี้ สัญญาณแรงดันไฟฟ้าที่เปลี่ยนแปลงไปที่เอาท์พุทของออปแอมป์ A_2 เป็นจนสุดท้ายของ t_2 มีค่า

$$V_{o\ (2)}\ (t=t_2) = -[\ V_{in}/(R_{INT} \times C_{INT})\ ] \quad \ldots\ldots\ldots\ldots (4.2)$$

และที่จุดสุดท้ายของ t_3

$$V_{o\ (2)}\ (t=t_3) = -[\ (V_{REF}/(R_{INT} \times C_{INT}))xt_2\ ] \quad \ldots\ldots (4.3)$$

เนื่องจากค่าแรงดันไฟฟ้าที่เปลี่ยนแปลงเท่ากัน,ค่าเวลา t_3 มีค่า (หรือค่า N_3 คือ จำนวนพัลส์ทั้งหมดในระหว่าง t_3

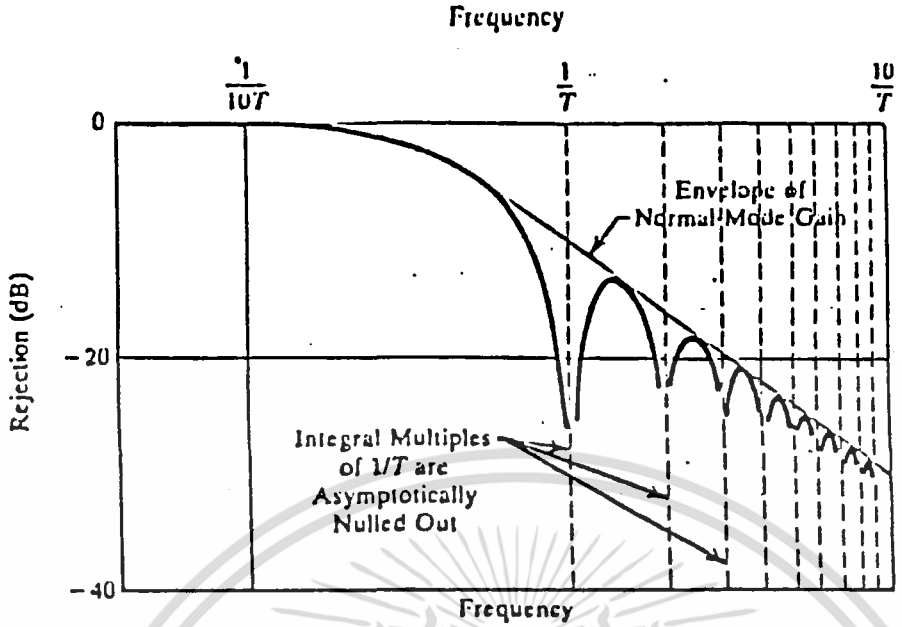
$$t_3 = V_{in} \times (t_2/V_{REF}) \quad \ldots\ldots\ldots\ldots (4.4)$$

ค่าในเทอมของจำนวนพัลส์

$$N_3 = V_{in} \times (N_2/V_{REF}) \quad \ldots\ldots\ldots\ldots (4.5)$$

จากสมการข้างต้นจะพบว่าความถูกต้องของ การแปลงข้อมูลไม่ได้ขึ้นอยู่กับค่าความถูกต้องของแรงดันอ้างอิง และความถูกต้องของสัญญาณนาฬิกาเท่านั้น แต่ยังขึ้นกับเสถียรภาพของตัวเก็บประจุในวงจรอินทิเกรเตอร์ C_{int} ที่เปลี่ยนแปลง ยกเว้นค่าที่ไม่เปลี่ยนแปลงในระหว่างการแปลงข้อมูลในแต่ละไซเคิล

ข้อดีของวงจรแปลงสัญญาณแบบนี้ การอินทิเกรทสัญญาณคาบเวลา t_2 สามารถเซตให้เท่ากับค่าจำนวนเต็มของค่าสัญญาณอินพุท ที่มีสัญญาณความถี่ผสมกันหลายความถี่ จะถูกอินทิเกรทเพื่อให้ได้ค่าเฉลี่ยเป็นศูนย์ที่เอาท์พุทของวงจรอินทิเกรเตอร์ การขจัดนี้เรียกว่า โหมดนอร์มอล (normal mode or line frequency rejection) แสดงในรูป 4.10



รูป 4.10 แสดงการขจัดสัญญาณในโหมดนอร์มอลของวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลแบบ สโลปคู่อินทิเกรชันพร้อมด้วยส่วนออร์โต-ซีโร ($t_2 = T$)

แม้ว่าวงจรนี้จะสามารถแปลงสัญญาณอนาลอกในย่านความถี่ต่ำได้ดี แต่ก็ยังมีข้อผิดพลาดที่เกิดจากกระแสรั่วที่สวิทช์ S_2 มีค่าประมาณน้อยกว่า 10 พิโคแอมป์ ซึ่งส่งผลไปลดค่าแรงดันไฟฟ้าที่ถูกเก็บไว้ใน C_2 และ C_{int} และปัญหาที่เกิดจากสวิทช์ S_1 และ S_2 ก็ยังมีผลต่อออปแอมป์ A_1 และ A_2 ทำให้เกิดความไม่เป็นเชิงเส้นที่เอาท์พุท ซึ่งเหมือนกับผลของค่าสลู-เรท ในออปแอมป์ A_1 ขณะที่สวิทช์ S_1 ได้จากแหล่งจ่ายแรงดันอ้างอิงกราวด์ ตัวเก็บประจุ C_{int} และ C_2 จะมีค่าการดูดกลืนไดอิเลคตริกเกิดขึ้น แต่เราสามารถทำให้ลดลงได้โดยการใช้ตัวเก็บประจุแบบโพลีสไตลีน โพลิโพไพลีน นอกจากนี้ยังมีการดีเลย์ (Delay) ในระหว่างการทำซีโร-ครอสซิ่งที่เอาท์พุทของออปแอมป์ A_2 กับการทำซีโร-ครอสซิ่ง โดยส่วนควบคุมลอจิก ซึ่งจะมีผลให้เกิดความแตกต่างในจำนวนของการนับ N_3 เท่าๆกับค่าความผิดพลาดส่วนที่เหลือใน ออร์โต-ซีโร จึงจำเป็นต้องเก็บผลรวมของค่าความผิดพลาดทั้งหมดให้มีค่าน้อยกว่า 1/2 LSB สำหรับการแปลงข้อมูลเอาท์พุท

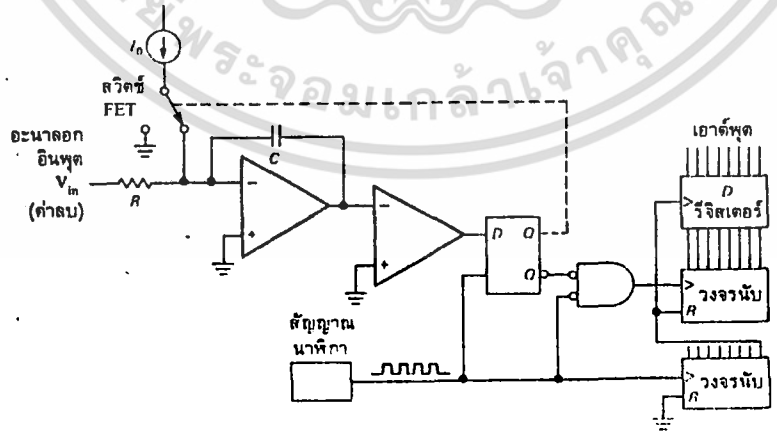
4.3.4.3 แบบชาร์จบาลานซ์ (Charge Balance A/D Converters)

วงจรเปลี่ยนสัญญาณ A/D Converter แบบชาร์จบาลานซ์ ใช้วงจรสำคัญคล้ายกับแบบสโลปคู่นั่นเอง แต่แทนที่จะให้อินพุทสวิทช์ไปมาระหว่างแรงดัน ที่ไม่รู้ค่ากับแรงดันอ้างอิง ก็ทำการแทรกพัลส์ของกระแสอ้างอิงมาตรงๆ ที่จุดรวม (summing point) ของวงจรอินทิเกรเตอร์ในช่วงเวลาที่คงที่ โดยที่จำนวนของพัลส์จะเป็นสัดส่วนโดยตรงกับแรงดันอินพุทที่ไม่รู้ค่า

ประโยชน์ของเทคนิคนี้ คือ แรงดันตกคร่อม ตัวเก็บประจุ ของวงจรอินทิเกรเตอร์จะมี ค่าใกล้เคียง 0 V ดังนั้น จึงไม่เกิดความผิดพลาด จากผลของกระแสรั่วไหล A/D Converter ชนิดนี้จึงมีความถูกต้องสูงกว่าแบบสโลปคู่

4.3.4.4 แบบเดลต้า-ซิกมา (Delta-Sigma A/D Converters)

จากวงจรรูปที่ 4.11 เมื่อมีแรงดันอินพุทป้อนเข้าไปที่วงจรอินททิเกรเตอร์จะให้เอาท์พุทไปเข้าวงจรเปรียบเทียบ เปรียบเทียบกับแรงดันคงที่ (จากรูปคือ กราวด์) พัลส์ของกระแสที่ได้ขึ้นอยู่กับเอาท์พุทของวงจรเปรียบเทียบ โดยสวิทช์ที่ทำขึ้นจากเฟสจะควบคุมให้กระแสเข้าไปยังที่จุดรามหรือลงกราวด์ไป ส่วนวงจรนับ จะนับจำนวนพัลส์ด้วยหลักการที่คล้ายกัน

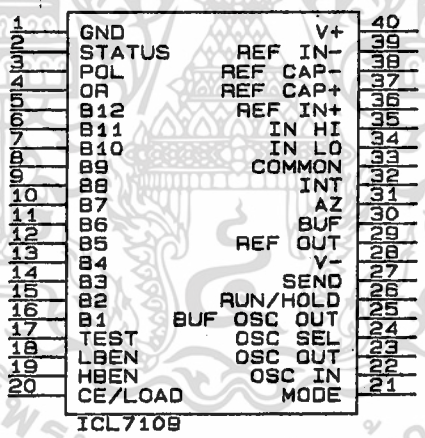


รูปที่ 4.11 วงจรเปลี่ยนสัญญาณแบบเดลต้า-ซิกมา

ข้อสรุปของ A/D Converter แบบอินทิเกรตสัญญาณ

จุดสำคัญ ของอินทิเกรติ้งเทคนิค คือ อินพุทที่ให้กับวงจรอินทิเกรเตอร์ ต้องเป็นกระแส ไอซีคอนเวอร์เตอร์บางตัวอาจมีอินพุทให้สองขา แต่จะมีขาหนึ่งต่อตรงกับจุด summing point ใช้กับอุปกรณ์ที่เป็นแหล่งจ่ายกระแสโดยตรง ถ้าให้อินพุทเป็นกระแสก็ไม่ต้องคำนึงถึง แรงดันออฟเซตของ วงจรอินทิเกรเตอร์ แต่ถ้าหากใช้กับ อินพุทที่เป็นแรงดัน(ต้องมีตัวต้านทานต่ออนุกรมอยู่ เพื่อให้ได้เป็นกระแส) ต้องปรับ ออฟเซตของออปแอมป์เสียก่อน การใช้อินพุทเป็นกระแส ทำให้ง่ายการใช้งานทางไฟสลับกว้างไอซีแบบชาร์จ-บาลานซ์มักประกอบด้วย วงจรแปลงแรงดันเป็นความถี่อยู่ด้วย ดังนั้นถ้าหากต้องการเอาท์พุทเป็นความถี่ก็สามารถเลือกได้

4.4 ไอซีวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลที่ใช้ในโครงงาน



รูปที่ 4.12 แสดงโครงสร้างภายนอกของไอซีเบอร์ ICL7109

ไอซีเบอร์ ICL7109 เป็นไอซีชนิด CMOS ที่มีคุณภาพสูง ใช้กระแสไฟฟ้าต่ำในการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล และยังถูกออกแบบให้ง่ายต่อการเชื่อมต่อกับไมโครโปรเซสเซอร์

เอาท์พุทข้อมูลมีขนาด 12 บิท มีขาควบคุมให้เลือกใช้งานทั้งแบบ 2 ไบท์ (LBEN,HBEN) และแบบตัวเลือกชิพ (Chip select) ทำให้ง่ายต่อการเชื่อมต่อแบบขนาน มีโหมดสำหรับการตรวจสอบการทำงาน (Handshaking)กับ UART (Universal Synchronous Asynchronous Receiver Transmitter) พร้อมกับการส่งข้อมูลในแบบอนุกรม ซึ่งสามารถประยุกต์ใช้งานในด้านการเก็บข้อมูล (Data logging) ระยะไกล อินพุท RUN/HOLD และเอาท์พุท STATUS จะบ่งบอกถึงสถานะการทำงานและควบคุมเวลาในการแปลงสัญญาณ

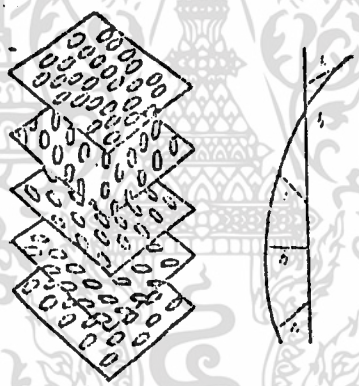
ไอซีเบอร์ ICL7109 มีความถูกต้องในการแปลงสัญญาณสูง, สัญญาณรบกวนต่ำ,(Low noise) ดริ๊ฟท์มีค่าต่ำ (Low drift) ใช้การแปลงสัญญาณอนาลอกเป็นดิจิตอลแบบอินทิเกรทสโลปคู่ (Dual Slope Integration) มีสัญญาณอินพุทและสัญญาณอ้างอิงที่แตกต่างกัน ความคลาดเคลื่อนจากการเปลี่ยนแปลงอุณหภูมิมีน้อยกว่า 1μV/c กระแสอินพุทสูงสุด 10 pA ใช้พลังงาน 20 mW ลักษณะต่างๆเหล่านี้ของ ICL7109 ทำให้มันเหมาะสมกับงานเก็บข้อมูลอนาลอกแบบมัลติเพลกซ์

บทที่ 5

จอภาพแสดงผล LCD

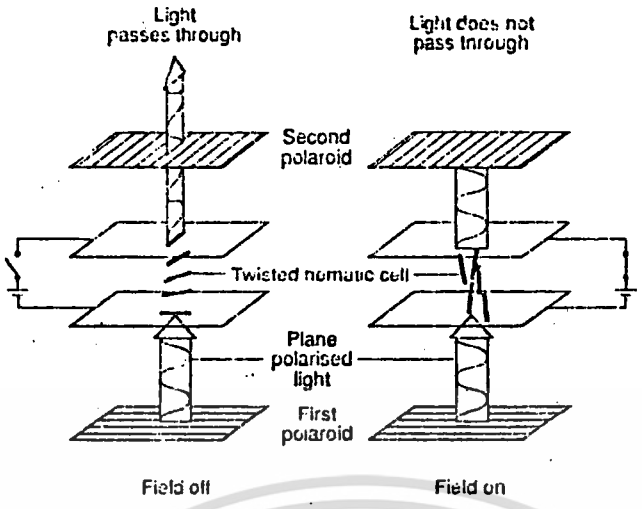
5.1 หลักการของจอภาพแสดงผล LCD

สารที่ใช้ทำจอภาพแสดงผล LCD คือ ผลึกเหลว (Liquid crustals) ซึ่งเป็นสถานะหนึ่งของสสารที่อยู่ระหว่างสถานะของผลึก กับสถานะของของเหลว และผลึกเหลวแบบที่นำมาใช้ในจอแสดงผล LCD เป็นผลึกเหลวแบบโคเลสเตอริก (Cholesteric liquid crystal) โมเลกุลจะเรียงตัวกันเป็นชั้น ๆ โดยในแต่ละชั้นโมเลกุลจะเรียงตัวไปทางเดียวกันคล้ายผลึกเหลวแบบเนมาติก แต่ละโมเลกุลในชั้นหนึ่ง ๆ จะเบี่ยงเบนไปจากชั้นอื่นเล็กน้อยอย่างต่อเนื่องคล้ายเป็นเกลียววนขึ้นไป ดังนั้นจึงมีชื่ออีกอย่างหนึ่งว่า Twisted nematics ดังรูป



รูปที่ 5.1 Cholesteric liquid crystal

จอแสดงผล LCD จะประกอบด้วยชั้นของผลึกเหลวโคเลสเตอริก โดยวางตัวเป็นชั้นให้บิดไปเป็นมุม 90 องศา และจะมีแผ่นกระจกประกบผลึกเหลวไว้ โดยที่แผ่นกระจกจะเคลือบไว้ด้วยชั้นของอินเดียมทินออกไซด์ (Indium tin Oxide In_2O_3) เพื่อทำหน้าที่เป็นขั้วไฟฟ้าที่โปร่งใส โดยจัดวางเป็นรูปแบบที่ต้องการจะแสดง และมีสารที่เป็นโพลารอยด์ประกบกระจกไว้อีกที โดยโพลารอยด์ทั้งสองวางแนวโพลาไรซ์ 90 องศาต่อกัน เพื่อเป็นการปิดเปิดแสง ดังรูป



รูปที่ 5.2 ส่วนประกอบของจอภาพแสดงผล LCD

เมื่อไม่มีการป้อนสนามไฟฟ้าแสงจะผ่านจากโพลารอยด์ตัวแรกเข้าสู่ผลึกเหลว ซึ่งผลึกเหลวจะทำให้แสงเปลี่ยนมุมไป 90 องศา แสงจึงผ่านโพลารอยด์ตัวที่สองออกไปได้ ทำให้เห็นเป็นโปร่งแสงไม่มีภาพปรากฏ และถ้ามีการป้อนสนามไฟฟ้าเข้าไปที่ขั้วไฟฟ้าตรงที่จะแสดงผล ผลึกเหลวจะมีการจัดเรียงตัวขึ้นใหม่เป็นเส้นตรง ทำให้แสงจากโพลารอยด์ตัวแรกผ่านขึ้นตรงเข้าสู่ผลึกเหลวโดยไม่มีการเปลี่ยนมุม แสงจึงไม่สามารถผ่านโพลารอยด์ตัวที่สองได้ จึงปรากฏเป็นสักษณะมืดตามรูปแบบของรูปที่ต้องการแสดง

5.2 DOT MATRIX LCD MODULE

อุปกรณ์ในปัจจุบันนี้ในส่วนของการแสดงผลนั้นจะใช้ LCD เสียเป็นส่วนใหญ่ไม่ว่าจะเป็น เครื่องเล่นวิดีโอ, เครื่องถ่ายเอกสาร, เครื่องคอมพิวเตอร์, เครื่องมือวัดควบคุมต่าง ๆ เนื่องจากเป็นอุปกรณ์ที่กินเนื้อที่น้อย และยังประหยัดไฟกว่าอุปกรณ์แสดงผลชนิดอื่นประมาณ 20 - 30 เท่า เราพอแบ่ง Dot matrix LCD module นี้ออกได้เป็นพวก ๆ ดังนี้

1. Character LCD module
2. Graphic LCD module
3. Segment display type LCD module

โดยในแต่ละแบบนี้ก็จะมีส่วนประกอบใหญ่ ๆ แบ่งได้เป็น

1. Dot matrix LCD เป็นตัวแสดงผลให้เรามองเห็นในลักษณะปิดและเปิดตัวเองกับแสงก็คือ ส่วนของที่เป็นตัวกระจกบรรจุผลึก

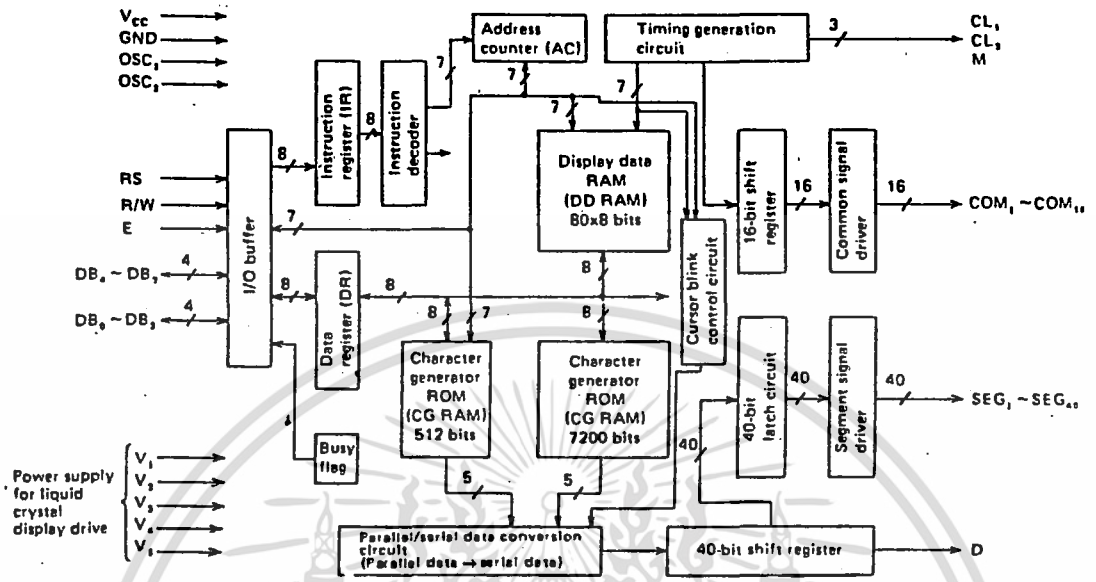
2. Driver เป็นตัวรับสัญญาณจากตัวควบคุมมาขับผลึก LCD อีกทีหนึ่งโดยมีเบอร์ที่นิยมใช้ใน LCD module เช่น HD44100H, MSM5259

3. Controller เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกเข้ามาและจัดการควบคุม LCD module ให้ทำงานแสดงผลต่าง ๆ เช่น การลบจอภาพ, การเกิดตัวอักษร เป็นต้น โดยมีเบอร์ IC ที่นิยมใช้กันคือ HD44780 ซึ่งมักจะใช้ในแบบ Character LCD module เป็นส่วนใหญ่ และเบอร์ IC HD61830 จะใช้ในแบบ Graphic LCD module

ในการศึกษาการทำงานและใช้งาน LCD module นั้นไม่ใช่เรื่องยากเลยถ้าเราสามารถทำความเข้าใจในส่วนของ Controller ได้ก็เพียงพอแล้ว และโดยส่วนมาก LCD module ในแต่ละบริษัทแล้วจะใช้ตัว Controller ที่มีหลักการทำงานเหมือน ๆ กันเป็นส่วนใหญ่ และใน LCD module แต่ละขนาดจำนวนตัวอักษรหรือจำนวนบรรทัดก็มีหลักการทำงานแบบเดียวกันทั้งหมด IC ที่นิยมมากที่สุดตัวหนึ่งที่เป็นตัว Controller LCD คือ เบอร์ HD44780 โดยรูปแบบการทำงานของมันได้เป็นมาตรฐานให้กับ Controller LCD ตัวอื่น ๆ ด้วย

HD44780 เป็นไอซี LSI ตัวหนึ่งที่ใช้ควบคุม LCD ในการแสดงผลในรูปของตัวอักษร หรือ สัญลักษณ์ต่าง ๆ ตัวมันเองสามารถต่อใช้งานแบบ 4 บิต หรือ 8 บิต ก็ได้

โดยถ้าเราต่อแบบ 4 บิต จะต่อใช้งานที่ DB7-DB4 เท่านั้น โดยข้อมูลครั้งแรกที่ส่งเข้ามานั้น HD44780 จะถือเป็นข้อมูล 4 บิตบน และข้อมูลที่ส่งต่อมานั้นเป็นข้อมูล 4 บิตล่าง



รูปที่ 5.3 โครงสร้างภายในของ HD44780

ขาต่าง ๆ ในการต่อใช้งาน HD44780

1. RS (Register Selection) จะเป็นขาเลือก Register ภายในซึ่งมีอยู่ 2 ตัว คือ Instruction Register (IR) และ Data Register(DR)โดยถ้าเป็น " ลอจิก 1 " จะเป็นการเลือก Data และถ้าเป็น " ลอจิก 0 " จะเป็นการเลือก Instruction

2. R/W (Read/Write) เป็นตัวเลือกว่าจะเขียน หรือจะอ่านข้อมูลจากตัว IC โดยถ้าอ่านข้อมูลจะเป็น "ลอจิก 1" แต่ถ้าเป็นการเขียนข้อมูลจะเป็น "ลอจิก 0"

3. E (Enable signal) เป็นขากำหนดสภาพการรับของการเขียนและการอ่านข้อมูล

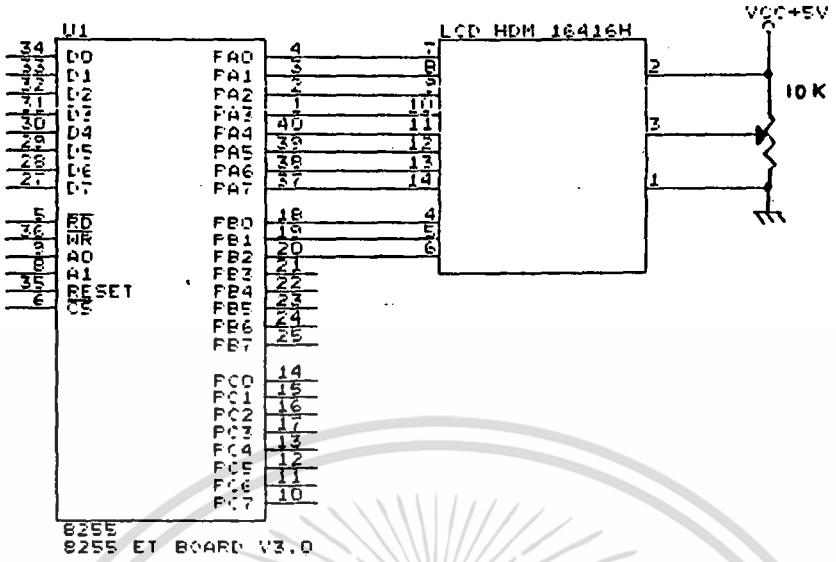
4. DB0-DB7 เป็นขารับส่งข้อมูลจากตัว IC

5. VDD ไฟเลี้ยงวงจร

6. VSS เป็นขากราวนด์

7. VO เป็นขารับไฟเลี้ยงในการขับ LCD ให้สว่างหรือมืด

ตัวอย่างการต่อใช้งาน



รูปที่ 5.4 ตัวอย่างการต่อใช้งาน

จากวงจรเป็นการต่อ IC 8255 (เป็นพอร์ตให้กับระบบไมโครโปรเซสเซอร์ทั่วไป) เข้ากับ LCD โดยเราจะจำลองสัญญาณต่าง ๆ ขึ้นมา โดยการใช้ พอร์ต A และพอร์ต B โดยพอร์ต A นั้นเราให้เป็นพอร์ตของข้อมูล และพอร์ต B นั้นเราใช้เป็นสัญญาณควบคุม

เมื่อเราเริ่มเปิดไฟป้อนให้ HD44780 มันจะทำการ Reset ตัวมันเองโดยจะใช้เวลาประมาณ 10 ms หลังจากไฟ VDD ถึง 4.5 Volt แล้ว โดยจะทำการเซ็ตตัวเอง ดังนี้

1. Display clear จะทำการลบข้อมูลจอภาพ LCD
2. Function set โดยจะตั้งค่าเซ็ตค่าภายใน
   - DL = 1 : เป็นการเซ็ตให้ติดต่อแบบ 8 บิต
   - N = 0 : เซ็ตให้เป็นการแสดงผล 1 บรรทัด
   - F = 0 : 5 X 7 จุดต่อหนึ่งตัวอักษร
3. Display on/off
   - D = 0 : Display off
   - C = 0 : Cursor off
   - B = 0 : Blink off

4. Enter mode set I/D = 1 : เพิ่มค่าตัวนับ (COUNTER) ขึ้น 1

S = 0 : No shift

ในการควบคุมการทำงานทำได้โดยส่งคำสั่งเข้าไป ซึ่งดูได้จากตาราง

Instruction	Code										Description	Execution time (when fosc is 250 kHz) Note 1	Execution time (when fosc is 160 kHz) Note 2
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	82 μs ~ 1.64 ms	120 μs ~ 4.9 ms
Return home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	40 μs ~ 1.6 ms	120 μs ~ 4.8 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 μs	120 μs
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40 μs	120 μs
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents	40 μs	120 μs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F).	40 μs	120 μs
Set CG RAM address.	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μs	120 μs
Set DD RAM address	0	0	1	A_{DD}							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μs	120 μs
Read busy flag & address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	1μs	1 μs
Write data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 μs	120 μs
Read data to CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 μs	120 μs
	I/D = 1: Increment (+1) I/D = 0: Decrement (−1) S = 1: Accompanies display shift. S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right. R/L = 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5 x 10 dots F = 0: 5 x 7 dots BF = 1: Internally operating BF = 0: Can accept instruction										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM address A_{DD}: DD RAM address Corresponds to cursor address. AC: Address counter used for both of DD and CG RAM address.	Execution time changes when frequency changes. (Example) When fosc is 270 kHz: $40\ \mu s \times \frac{250}{270} = 37\ \mu s$	

*No effect

Notes 1. Applied to models driven by 1/8 duty or 1/11 duty.

2. Applied to models driven by 1/16 duty.

รูปที่ 5.5 ตารางคำสั่ง HD44780

บทที่ 6
วงจรเครื่องไมโครเทอร์มินอล

6.1 Control pack ANT32

ANT32 เป็นบอร์ดที่สร้างขึ้นจำหน่ายโดยบริษัท ศิรารีเสิร์ช ใช้ไมโครคอนโทรลเลอร์ เบอร์ 8032 เป็นตัวประมวลผล มีขนาดเล็ก 5.25" x 3.9" สามารถนำไปประยุกต์ใช้ในงานควบคุม ได้กว้างขวางมาก มีพอร์ทรับส่งข้อมูลแบบขนาน เบอร์ 8255 ถึง 2 ตัว สามารถพัฒนาโปรแกรมโดยใช้ภาษาเบสิก ของ BASIC52 โดยไม่ต้องใช้ไอซี เบอร์ 8052 AH BASIC เนื่องจากได้นำเอาตัวแปลภาษาเบสิก ที่อยู่ใน ROM ของ 8052 AH BASIC บรรจุลงใน EPROM วงจร ANT32 แสดงดังรูป 6.1

รายละเอียดการใช้งาน แสดงไว้ในภาคผนวก

6.2 วงจรส่วนแสดงผลและคีย์บอร์ด

จอแสดงผล LCD และคีย์บอร์ดจะต่อเข้ากับไอซี 8255 ตัวที่ 1 (User1) ของ Control pack ANT32 โดยขาข้อมูลของจอแสดงผล LCD จะต่อเข้ากับพอร์ต A ส่วนขาควบคุมต่อเข้ากับพอร์ต B ที่บิท 0 ถึง บิท 3

ในส่วนของคีย์บอร์ดซึ่งเป็นเมมเบรนสวิทช์ (Membrain switch) ซึ่งมีการจัดเรียงตัวแบบเมตริกซ์ โดยขา 5 ถึงขา 8 จะพูลไฮด์ (Pull high) ไว้พร้อมกับต่อเข้าที่พอร์ต C บน ส่วนขา 1 ถึงขา 4 จะต่อเข้ากับไอซีดีโคดเดอร์ (Decorder) เบอร์ 74LS145 ซึ่งจะส่งให้ขา 1 ถึงขา 4 แอคทีฟทีละขา เมื่อมีการกดคีย์ สามารถตำแหน่งที่กดได้โดยการรับค่า (In port) ที่พอร์ต C ล่าง รูปวงจรแสดงดังรูปที่ 6.2

6.3 วงจรส่วนแปลงสัญญาณจากอนาลอกเป็นดิจิตอล

ไอซี 7109 เป็นไอซีแปลงสัญญาณอนาลอกเป็นดิจิตอลขนาด 12 บิท ดังนั้นจึงมีขาข้อมูลต่อเข้ากับไอซี 8255 ตัวที่สอง (User 2) 12 ขา โดยเป็นพอร์ต A และพอร์ต B ที่บิท 0 ถึงบิต 4 เมื่อมีการทริกของทริกเกอร์ (Trigger) สวิทช์ S2 สัญญาณอนาลอกจะเข้าที่ขา 34 และขา 35 เพื่อทำการแปลงเป็นดิจิตอล และในการแปลงสัญญาณจะต้องมีการป้อนสัญญาณอ้างอิง (Reference) เข้าที่ขา 36 และขา 39 ค่าสัญญาณอ้างอิงปรับได้โดย VR2 และเนื่องจากไอซี 7109 ต้องรับไฟเลี้ยงที่เป็นระดับลบด้วย จึงต้องมีไอซีเบอร์ 7660 มาแปลงไฟเลี้ยงจากด้านบวกให้เป็นด้านลบแล้วป้อนให้ไอซี 7109 ที่ขา 28 (V-) รูปวงจรแสดงดังรูป 6.3

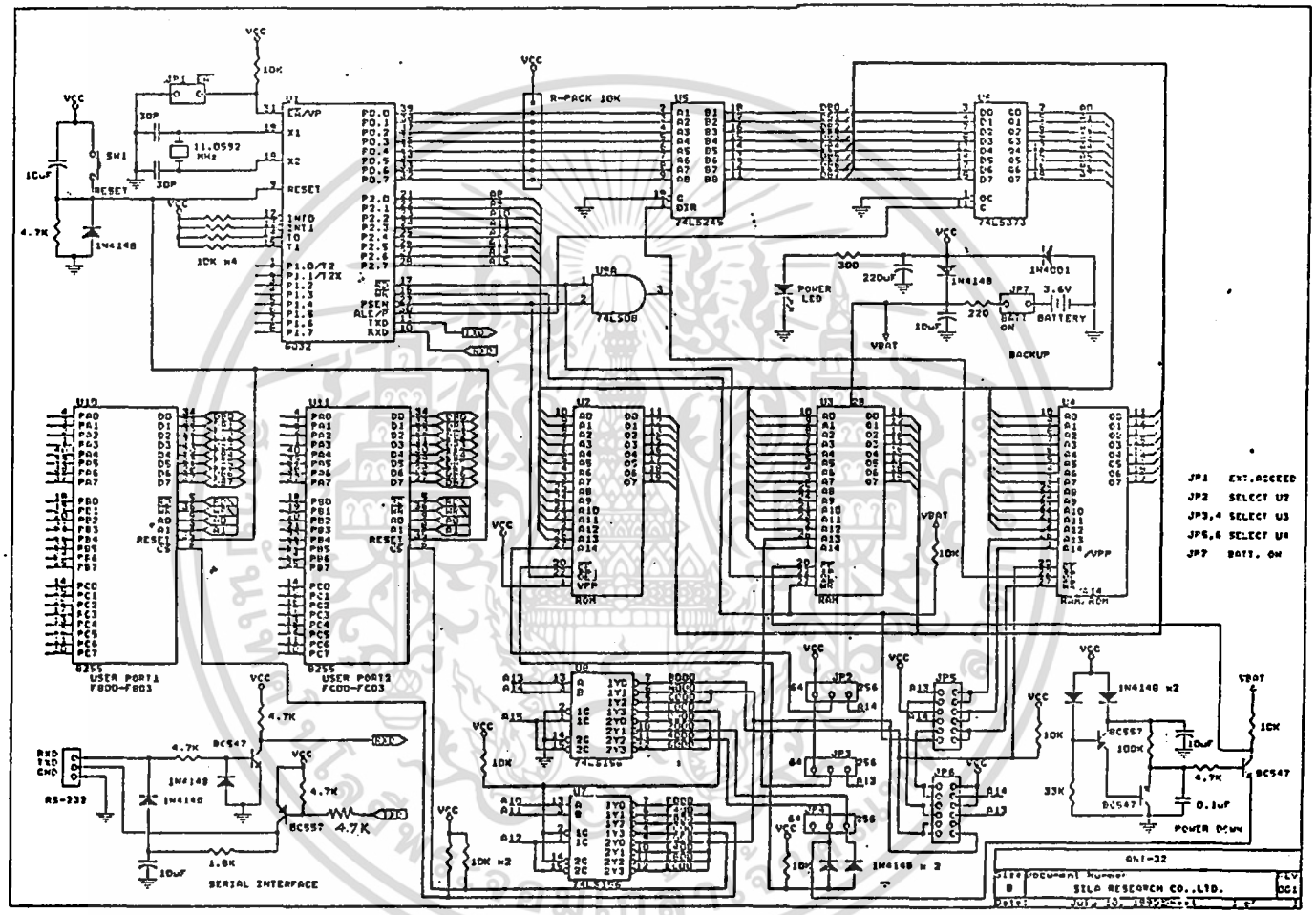
6.4 วงจรส่วนการเชื่อมต่อกับคอมพิวเตอร์และส่วนแปลงสัญญาณให้เข้ากัน

ขา Tx จากเครื่องไมโครเทอร์มินอลต่อเข้ากับขา 2 ของไอซี 75179 ซึ่งเป็นไอซีที่ส่งและรับสัญญาณตามมาตรฐาน RS 422A สัญญาณจะส่งผ่านตามสายไปยังไอซี 75179 อีกตัวคอยรับสัญญาณแล้วส่งผ่านไปยังขา 10 ของไอซีแปลงระดับสัญญาณ MAX 232 ให้เป็นระดับสัญญาณตามมาตรฐาน RS 232C เพื่อเข้าไปยังขา Rx ที่อยู่ในพอร์ตอนุกรม RS 232C ซึ่งมีในเครื่องคอมพิวเตอร์ทั่วไป

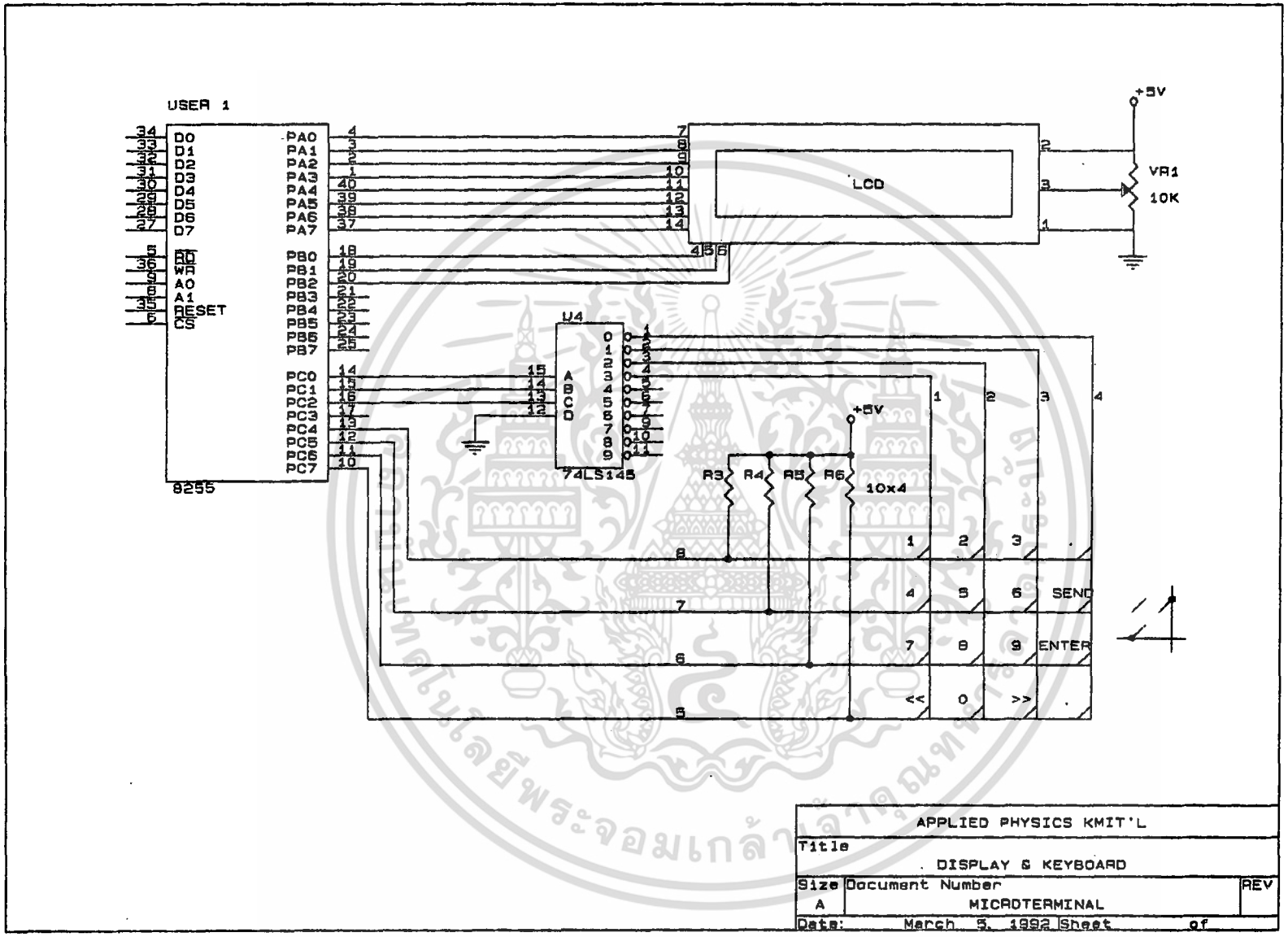
ส่วนขา Rx ในเครื่องไมโครเทอร์มินอลจะเป็นขารับสัญญาณจากขา Tx ของเครื่องคอมพิวเตอร์ โดยสัญญาณจากขา Tx ของเครื่องคอมพิวเตอร์จะถูกแปลงโดย MAx 232 ให้เป็นสัญญาณในระดับทีทีแอลแล้วส่งผ่านไปยังส่วน RS 422A Interface เข้าสู่ขา Rx ของเครื่องไมโครเทอร์มินอล รูปวงจรแสดงดังรูป 6.4

6.5 วงจรรวมส่วนต่าง ๆ

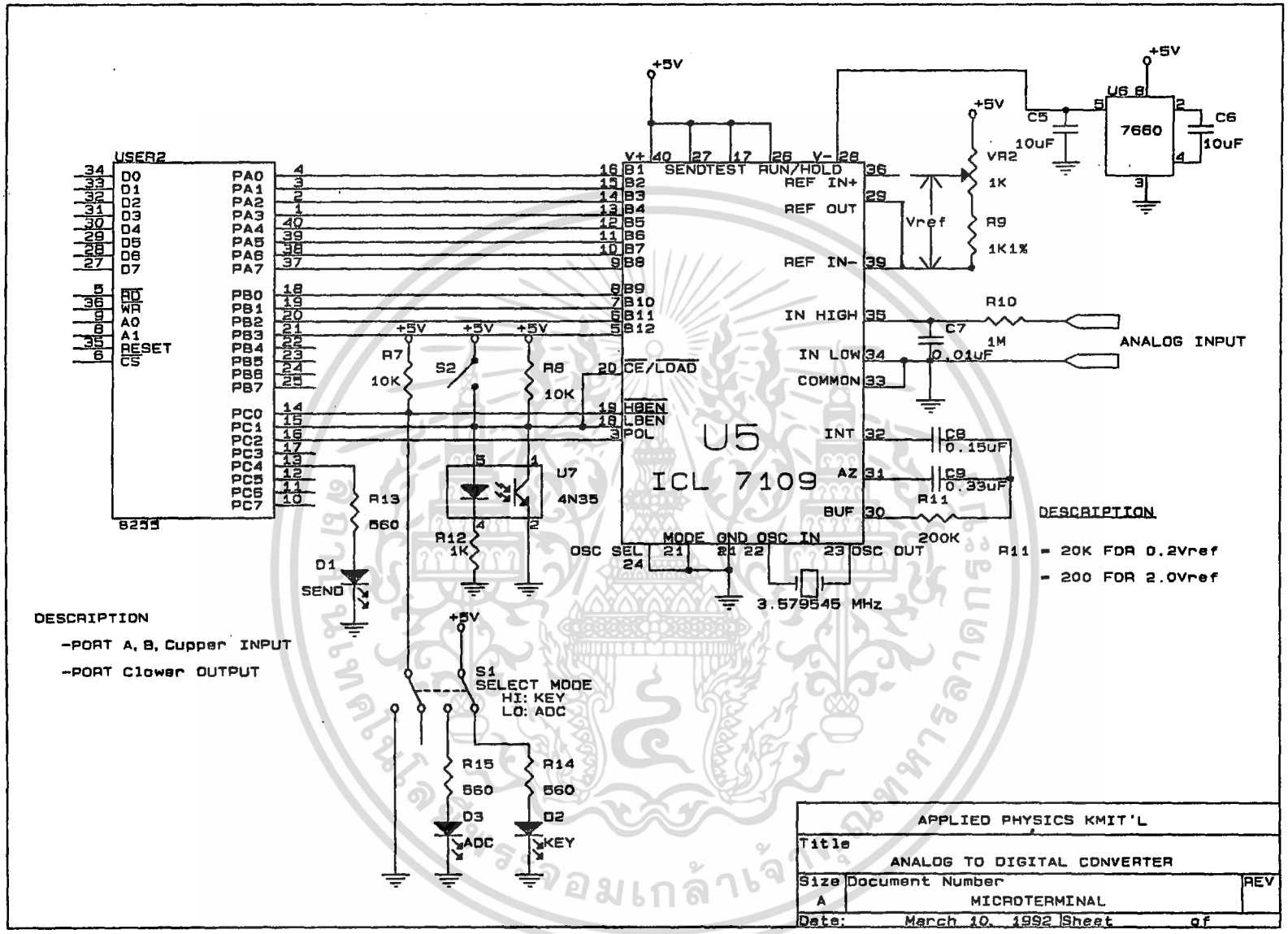
รูปวงจรแสดงให้เห็นในรูป 6.5 เส้นประล้อมกรอบจะเป็นส่วนของ Control pack ANT32 โดยแสดงให้เห็นเฉพาะส่วนที่ต่อกับภายนอก สวิทช์ S1 เป็นสวิทช์สำหรับเลือกโหมดการทำงานว่าเป็น ADC Mode หรือ INKEY Mode โดยเมื่อมีการเลือกโหมดจะมีการสว่างของ LED ประจำโหมดนั้น ส่วน LED 1 (SEND) จะสว่างเมื่อมีการส่งข้อมูลไปยังคอมพิวเตอร์



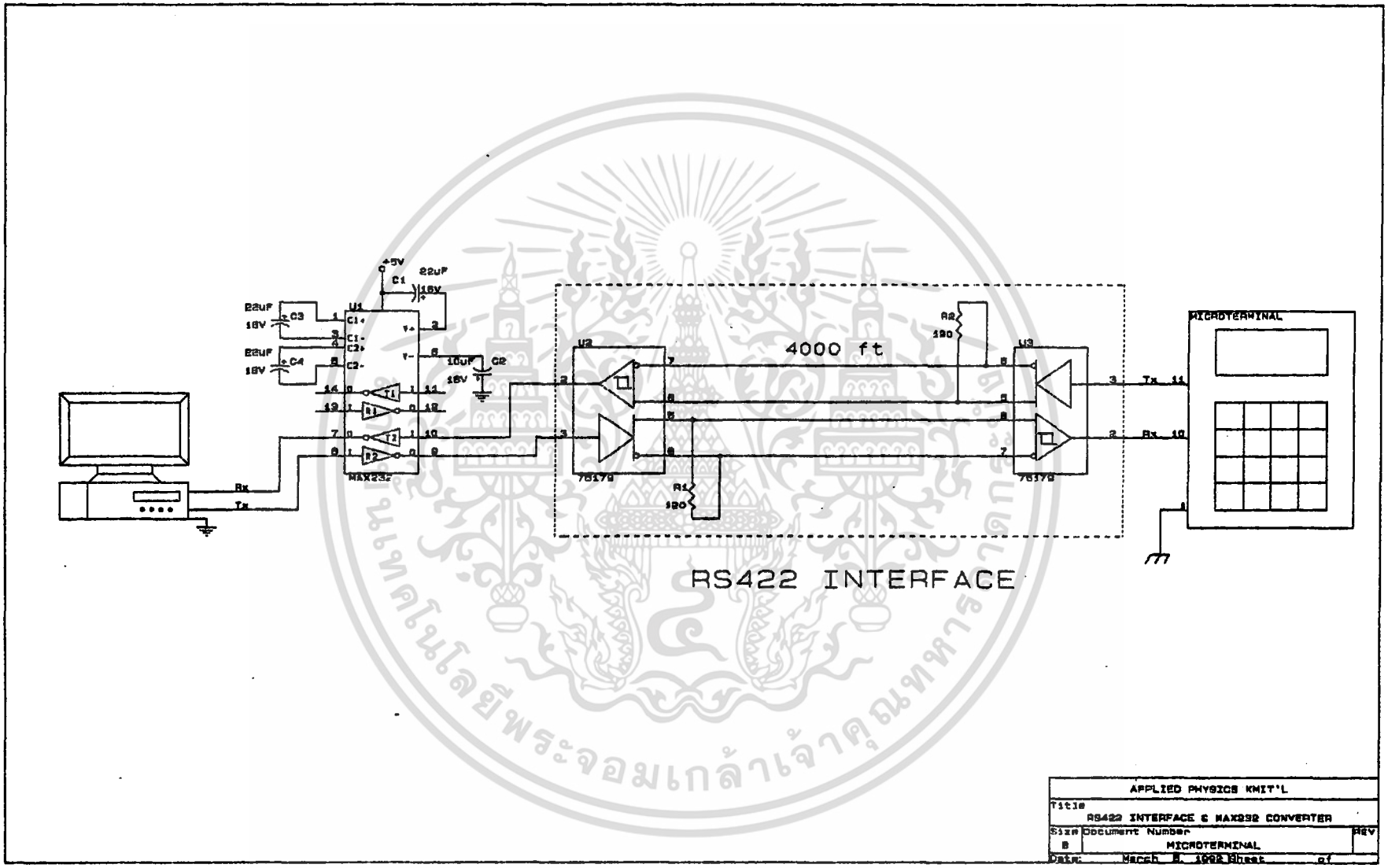
รูปที่ 6.1 วงจรส่วน Control pack ANT32



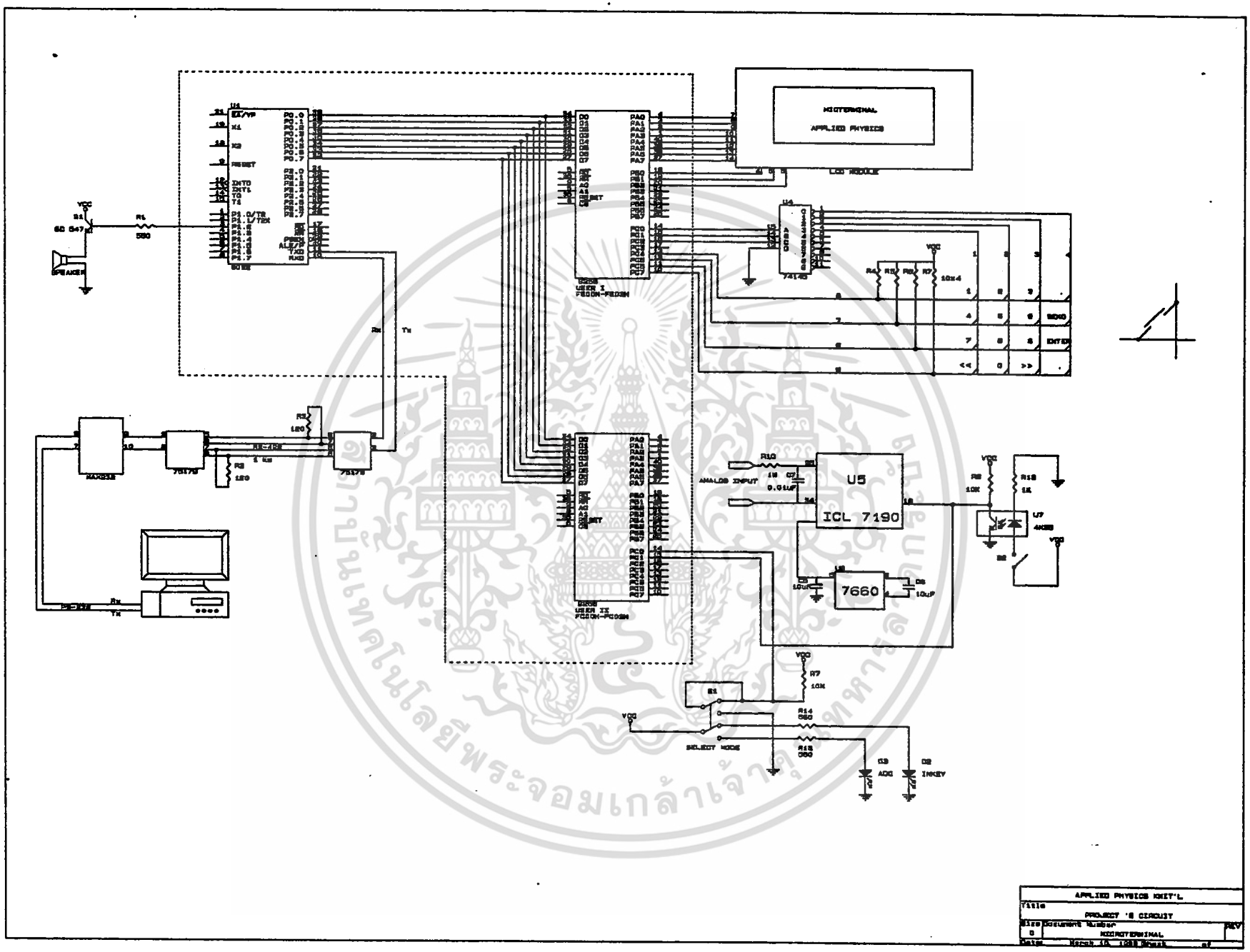
รูปที่ 6.2 วงจรส่วนแสดงผลและคีย์บอร์ด



รูปที่ 6.3 วงจรส่วนแปลงสัญญาณจากอนาลอกเป็นดิจิตอล



6.4 วงจรส่วนการเชื่อมต่อกับคอมพิวเตอร์และส่วนแปลงสัญญาณให้เข้ากัน



รูปที่ 6.5 วงจรรวมส่วนต่าง ๆ

บทที่ 7

โปรแกรมควบคุมระบบ

โปรแกรมการทำงานของเครื่องไมโครเทอร์มินอล จะควบคุมการทำงานของส่วนแสดงผล LCD การสแกนคีย์บอร์ด การส่งข้อมูลเข้าเครื่องไมโครคอมพิวเตอร์ และ ควบคุมการรับสัญญาณอนาลอก แล้วนำค่าที่ได้มาแสดงผลที่จอ LCD ซึ่งโปรแกรมควบคุมดังกล่าว เขียนด้วยภาษาเบสิก 52 ข้อมูลที่ได้จากไมโครเทอร์มินอลจะถูกเก็บและนำไปประมวลผลโดย เครื่องไมโครคอมพิวเตอร์ซึ่งใช้ซอฟแวร์สำเร็จรูป LOTUS MEASURE

7.1 โปรแกรมควบคุมการทำงานของเครื่องไมโครเทอร์มินอล

7.1.1 ภาษาเบสิก 52

ภาษาเบสิก 52 สร้างขึ้นมาเพื่อใช้ในงานควบคุมโดยเฉพาะ นอกจากจะมีคำสั่งพื้นฐาน เหมือนภาษาเบสิกทั่วไปแล้ว ยังมีคำสั่งพิเศษที่ช่วยในงานควบคุมเพิ่มขึ้นอีก ดังนั้นผู้ใช้จึงสามารถพัฒนาโปรแกรมใช้งานได้สะดวกในเวลาอันรวดเร็ว เนื่องจากการทำงานของภาษาเบสิกค่อนข้างช้า เมื่อเทียบกับภาษาแอสเซมบลี้ ดังนั้นเบสิก 52 จึงมีคำสั่งซึ่งสามารถติดต่อกับ โปรแกรมย่อยที่เป็นภาษาแอสเซมบลี้ได้ เพื่อใช้ในงานที่ต้องการความเร็วสูง เช่น การสแกนภาคแสดงผล 7 เซกเมนต์ เป็นต้น

7.1.1.1 คุณสมบัติพิเศษของภาษาเบสิก 52

MCS BASIC 52 เป็นเป็นตัวแปลภาษาเบสิกที่มีคำสั่งและฟังก์ชันพื้นฐาน คล้ายภาษาเบสิกทั่วไปแต่ MCS BASIC 52 ได้เพิ่มคำสั่งเพื่อความสะดวกในการพัฒนาโปรแกรมควบคุมระบบ เช่น

- สามารถโปรแกรม EPROM โดยคำสั่ง "PROG" ซึ่งจะนำโปรแกรมเบสิก ที่ใช้อยู่ขณะนั้น บันทึกลง EPROM ในลักษณะของไฟล์

- สามารถเรียกโปรแกรมเบสิกที่เก็บไว้ใน EPROM ออกมาทำงานได้ โดยคำสั่ง "ROM"

- สามารถย้ายโปรแกรมเบสิกที่เก็บอยู่ใน EPROM มาไว้ใน RAM ด้วยคำสั่ง "XFER"

- สามารถเรียกโปรแกรมย่อย ภาษาแอสเซมบลี้ ด้วยคำสั่ง "CALL"

- โปรแกรมย่อยภาษาแอสเซมบลี้ สามารถเรียกใช้ฟังก์ชันภาษาเบสิกได้

- มีคำสั่งควบคุมไทม์เมอร์/เคาว์เตอร์ ที่อยู่ในตัว ด้วยคำสั่ง "TIMER0, TIMER1,TCON,TMOD" เป็นต้น

- ระบบรีลไทม์คล้อก (Real time clock) ที่เที่ยงตรง ควบคุมด้วยคำสั่ง "CLOCK0,CLOCK1,TIMER"

- สามารถรับรู้ และควบคุม การอินเทอร์รัพท์ ด้วยคำสั่ง "ONE1,RETI, CLEARI,IDLE" เป็นต้น

- อ่านและเขียนค่าพอร์ทในตัวด้วยคำสั่ง "PORT1"

- คำสั่งควบคุมพอร์ทอนุกรมในตัว เช่น "BAUD,RCAP2"

- สามารถติดต่อกับหน่วยความจำภายใน,ภายนอก ด้วยคำสั่ง "CBY(), DBY(),XBY()"

- คำสั่งควบคุมลูป "DO...WHILE,DO...UNTIL"

- ฟังก์ชันคณิตศาสตร์ที่สมบูรณ์

7.1.1.2 การจัดหน่วยความจำ

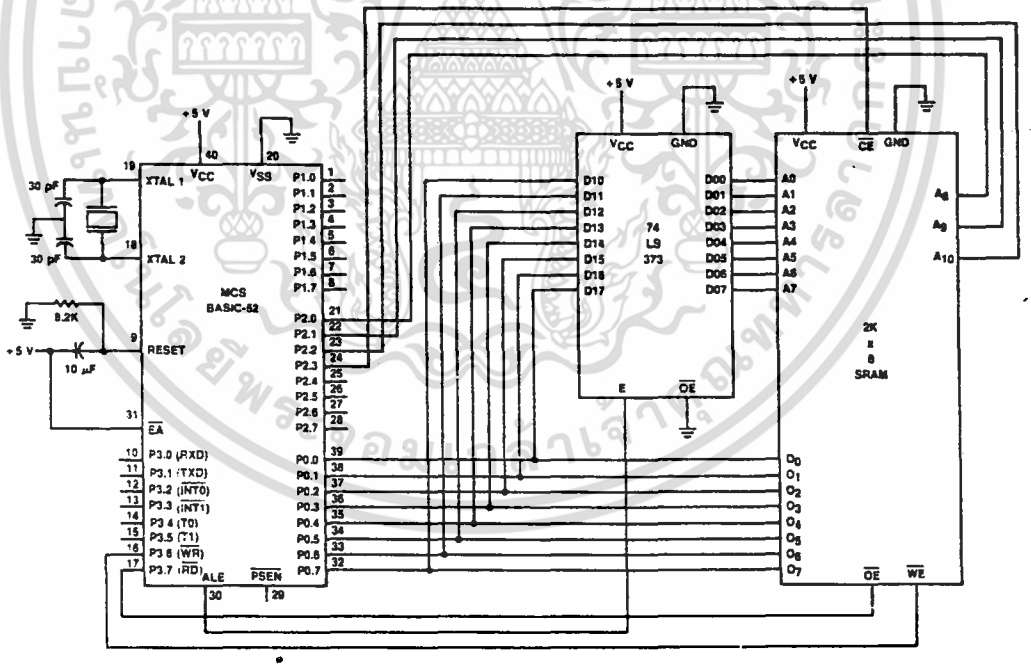
ตัวแปลภาษาเบสิกจะถูกบรรจุอยู่ใน ROM ขนาด 8 กิโลไบท์ ซึ่งอยู่ภายในไอซีเบอร์ 8052 AH BASIC ที่ตำแหน่ง 0000H-1FFFH (Program memory) ในระหว่างการทำงาน มันต้องใช้ RAM ภายนอกอย่างน้อย 1 กิโลไบท์ หลังจากการรีเซ็ท MCS BASIC 52 จะหาขนาดของ RAM ภายนอกซึ่งอยู่ที่ตำแหน่ง 0000H-0DFFH (Data memory) ข้อมูลที่ผ่านการทดสอบเพื่อหาขนาดจะมีค่าเป็น 00H ถ้าขนาดของ

RAM ไม่ถึง 1กิโลไบท์ MCS BASIC 52 จะไม่สามารถทำงานได้ ที่แอดเดรส 0E00H-0FFFFH (Data memory) จะสงวนไว้ให้ผู้ใช้สำหรับอินพุท เอาท์พุท และโปรแกรมภาษาแอสเซมบลี้ ข้อมูลในระหว่างแอสเดรสนี้จะไม่ถูกเคลียร์เป็นศูนย์

8052 AH BASIC แบ่งการจัดหน่วยความจำ ตามโครงสร้างทางฮาร์แวร์ที่ผู้ใช้สร้างขึ้น เป็น 2 โหมด ได้แก่

1. โหมด RAM

การทำงานในโหมดนี้ สามารถต่อ RAM ภายนอกได้ถึง 64 กิโลไบท์ ที่ตำแหน่ง 0000H-0FFFFH โหมดนี้ไม่สามารถโปรแกรม EPROM ได้ โดยทั่วไปแล้ว ใช้สำหรับตรวจสอบความถูกต้องของฮาร์ดแวร์ และเพื่อเป็นการทำความเข้าใจ 8052 AH BASIC กับระบบที่ง่ายที่สุด ดังในรูปที่ 7.1



รูปที่ 7.1 แสดงการต่อวงจรของ8052ในโหมด RAM

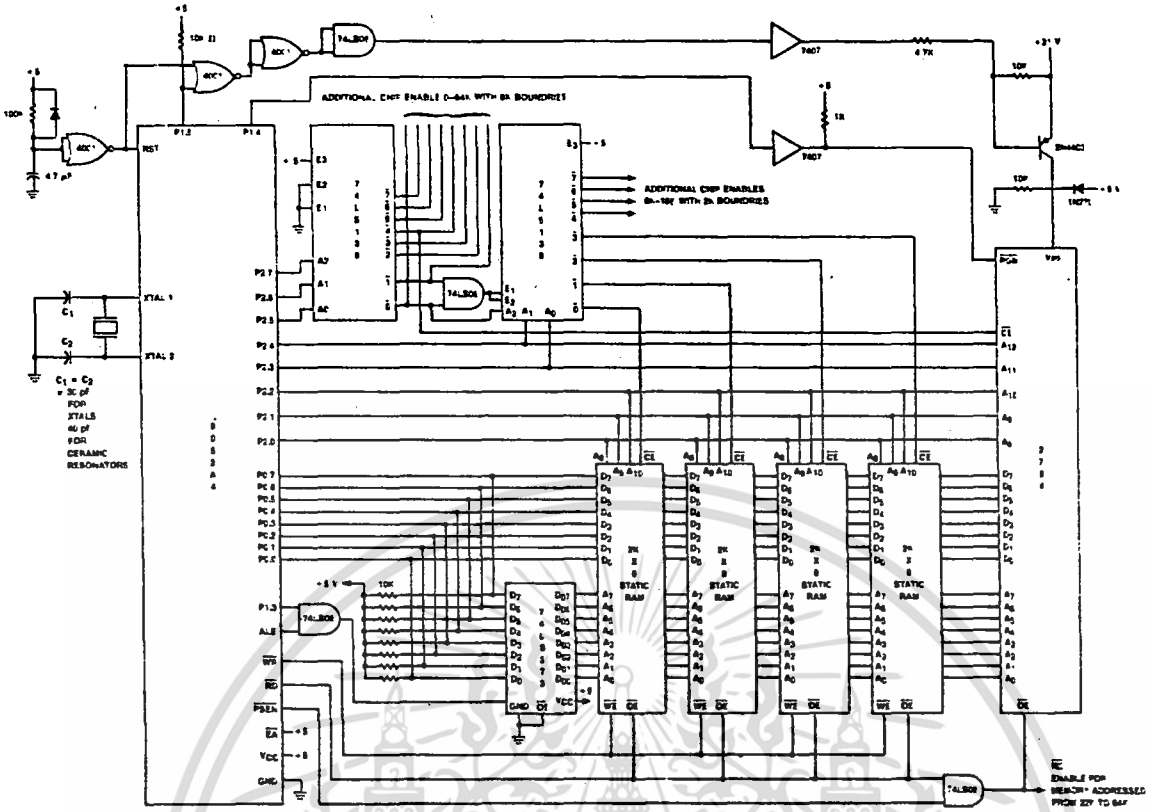
2. โหมด RAM/EPROM

โหมดนี้จะเป็นการทำงานอย่างสมบูรณ์แบบ ของ 8052 AH BASIC ซึ่งสามารถติดต่อกับหน่วยความจำภายนอกทั้ง RAM และ EPROM ได้ และยังสามารถโปรแกรม EPROM ได้อีกด้วย ตำแหน่งของข้อมูลสำหรับ RAM , EPROM , สถานะ และขนาด ดังแสดงในตารางที่ 7.1

ตารางที่ 7.1 แสดงการจัดหน่วยความจำในโหมด RAM/EPROM

ชนิดของหน่วยความจำ	ตำแหน่งข้อมูล	สถานะ	ขนาด
ROM (ภายใน)	0000H-1FFFH	Program memory	8KB
ROM (ภายนอก)	2000H-7FFFH	Program memory	24KB
RAM (ภายนอก)	0000H-7FFFH	Data memory	32KB
ROM และ/หรือ RAM	8000H-0FFFFH	Program/Data memory	32KB

การโปรแกรม EPROM จะใช้คำสั่ง "PROG" โดยมีแอดเดรสที่ 8000H-0FFFFH โปรแกรมเบสิกที่ถูกอัดลงบนตัว EPROM จะอยู่ที่ตำแหน่ง 8010H ส่วนที่ตำแหน่ง 8000H-800FH จะเก็บค่าบอดเรด(Baud rate) และค่าเซทอัพ(setup) ระบบ ที่ผู้ใช้ต้องการ ระบบฮาร์ดแวร์ในโหมดนี้ดังรูปที่ 7.2



รูปที่ 7.2 แสดงระบบฮาร์ดแวร์ในโหมด RAM/EPROM

7.1.1.3 การรับส่งข้อมูล

8052 AH BASIC มีพอร์ทสื่อสารอนุกรม อยู่ภายในตัว สามารถติดต่อกับเทอร์มินอลในโหมด 8 บิท UART ซึ่งมีข้อมูล 8 บิท บิทเริ่มต้น 1 บิท บิทหยุด (stop bit) 1 บิท และไม่มีบิทคู่(parity bit) สัญญาณจาก 8052 เป็นแบบ TTL จะต้องแปลงสัญญาณ ให้อยู่ในมาตราฐานของ RS 232 จึงสามารถติดต่อกับเทอร์มินอลได้ วงจรการแปลงสัญญาณจาก TTL ไปเป็น RS 232 แสดงในรูปที่ 7.3

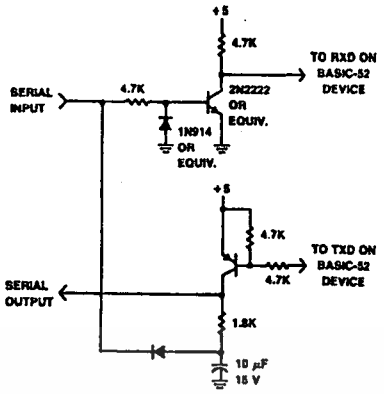


Figure 6A.
TWO TRANSISTORS TO IMPLEMENT RS-232. THE "NEGATIVE" SUPPLY FOR THE SERIAL OUTPUT LINE IS TAKEN FROM THE SERIAL INPUT LINE. NO ±12 VOLT SUPPLY IS REQUIRED.

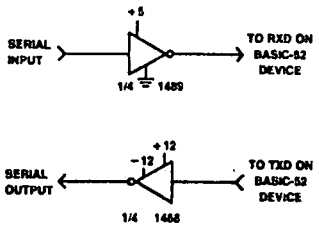


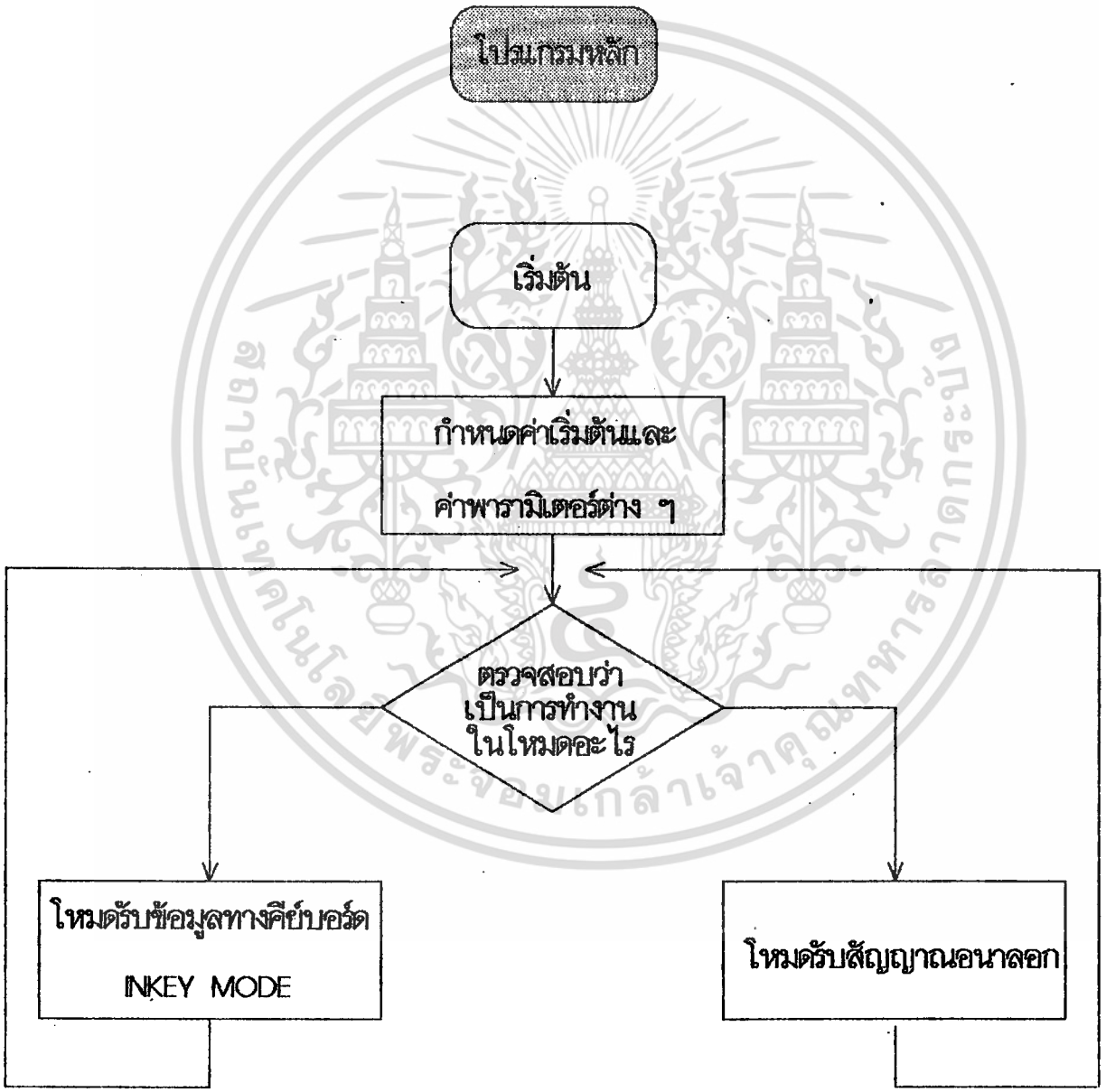
Figure 6B.
USING THE STANDARD 1489 AND 1488 LINE RECEIVERS AND DRIVERS, ±12 VOLTS IS NEEDED WITH THIS IMPLEMENTATION.

รูปที่ 7.3 แสดงวงจรการแปลงสัญญาณระหว่าง TTL กับ RS 232

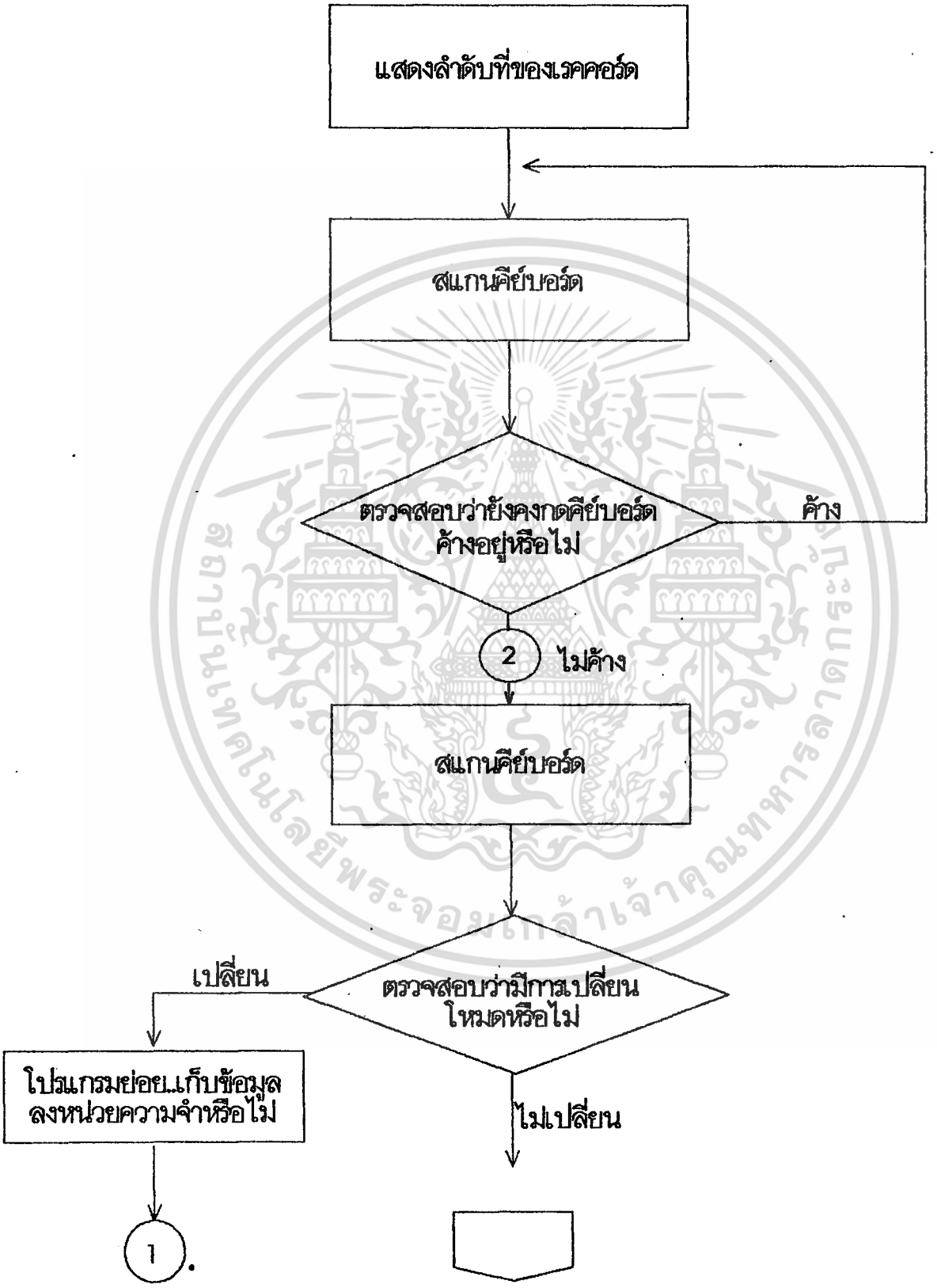
7.1.1.4 การเชื่อมต่อกับเทอร์มินอล

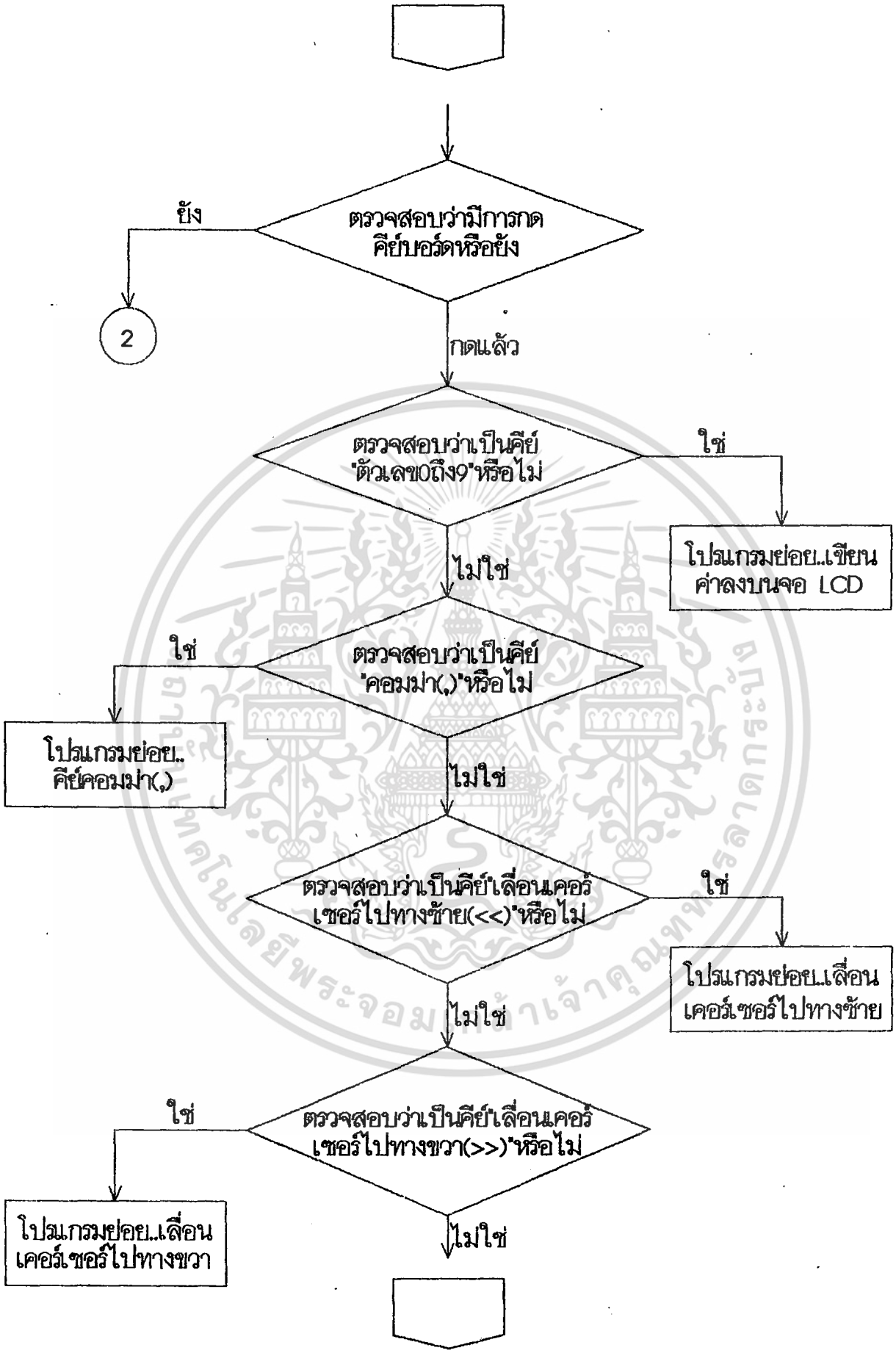
8052 AH BASIC สามารถใช้เครื่องไมโครคอมพิวเตอร์ PC/xT เป็นเทอร์มินอลได้ โดยผ่านการสื่อสารแบบอนุกรม RS 232 และใช้โปรแกรมที่ช่วยในการสื่อสาร เช่น Crosstalk,Procom,BTK เป็นต้น ข้อมูลที่รับส่งจะอยู่ในรูปของรหัส ASCII

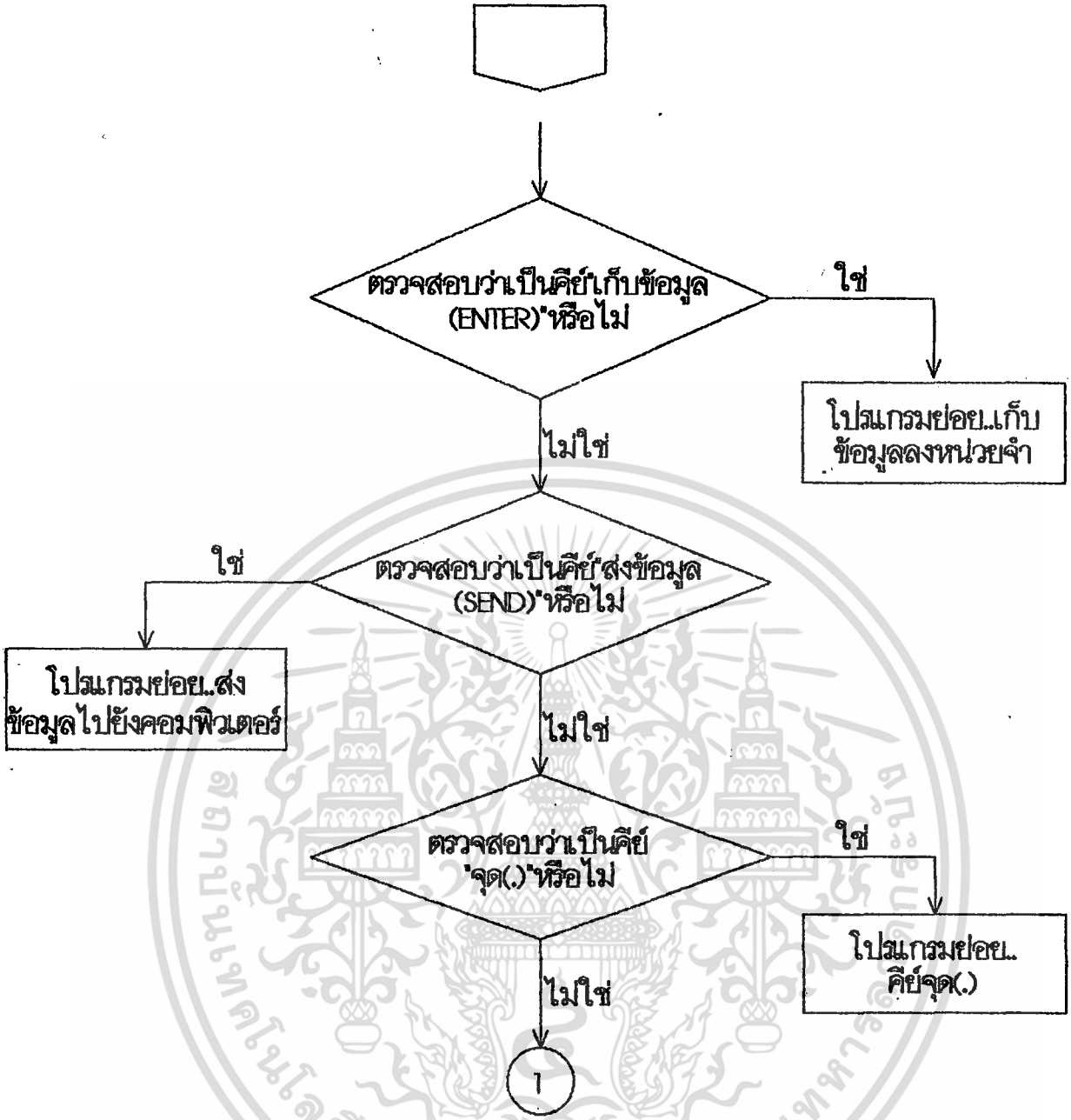
7.1.2 ลำดับขั้นการทำงานของโปรแกรมควบคุมระบบ

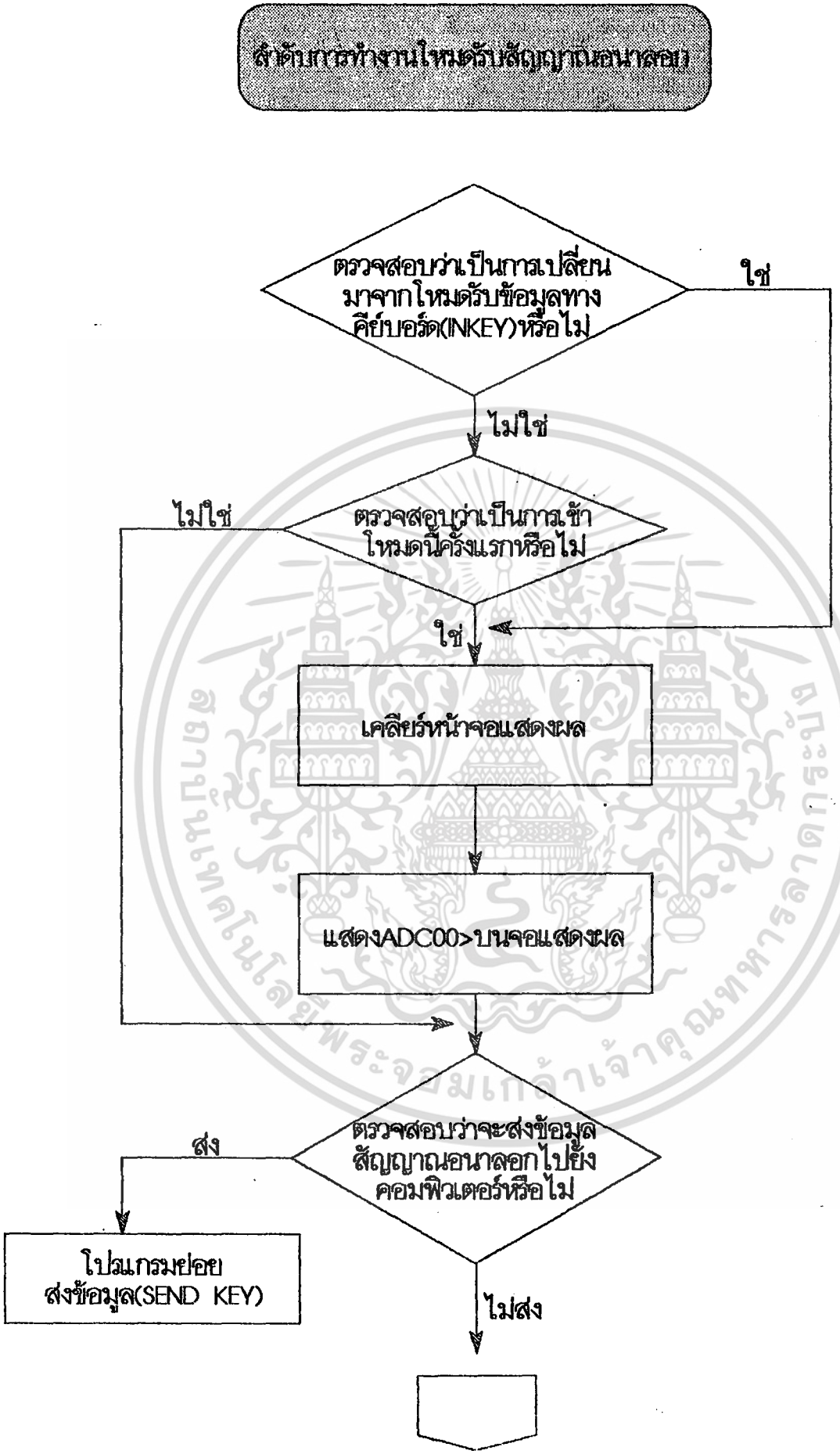


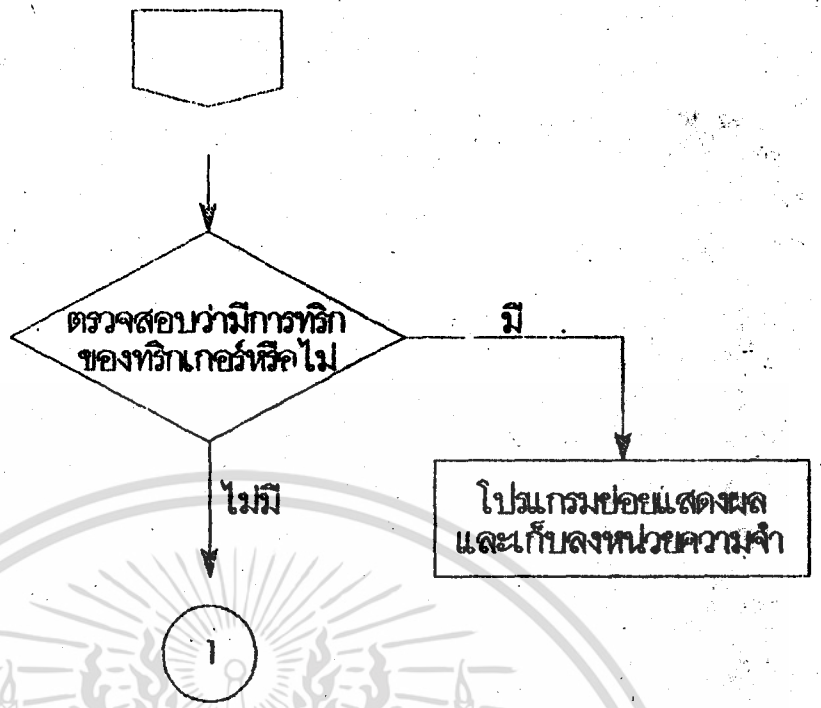
ลำดับการทำงานโหมดรับสัญญาณทางคีย์บอร์ด

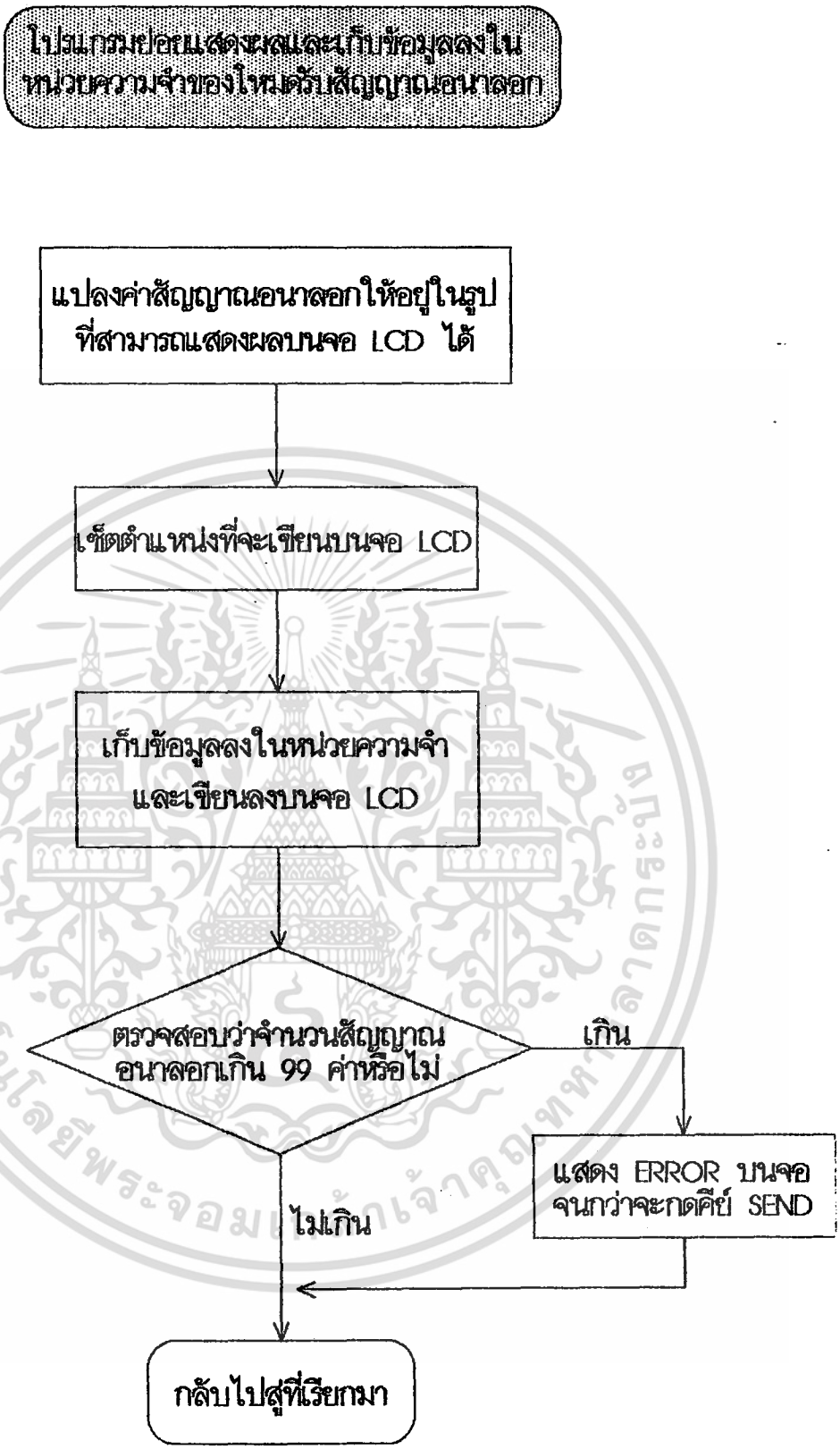












7.2 โปรแกรมการสื่อสาร และวิเคราะห์ข้อมูล (LOTUS MEASURE)

ขั้นแรกจะต้องมีแผ่นดิสค์ที่มีไฟล์ RS 232 Module และ LOTUS 123 อยู่ โดยทำเป็นแบทช์ไฟล์ (Batch file) ชื่อ Measure.bat เวลาเรียกใช้งาน สามารถเรียก Measure ได้เลย หลังจากที่เมนูของโลตัสขึ้นบนจอภาพ กดคีย์ "Alt" กับ "F9" พร้อมกัน เมนูหลักของ RS 232 Module จะปรากฏ ดังรูป

Settings	Capture-Range	Transmit-Range	Break	Quit

รูปที่ 7.4 เมนูหลักของ RS 232 Module

จากนั้นก็เซ็ทค่าเริ่มต้นต่าง ๆ ก่อน โดยเลือก Setting แล้วตั้งค่าดังนี้

Interface

Com-Port : เลือกพอร์ทอนุกรมที่เราต่อใช้งาน Com1

Baud : เลือกความเร็วในการรับส่งข้อมูล 9600

Stop-Bits : 1

Parity : None

Length : 8

Xon/Xoff

Inbound : Disabled

Outbound : Disabled

Duratiion

Break(ms) : ช่วงเวลาที่ต้องการหยุดทำงาน ซึ่งจะไม่มีการรับส่งข้อมูล

Timeout(sec) : ช่วงเวลาในการรับส่งข้อมูล

Capture

Capture-Range : ขอบเขตของข้อมูลที่ต้องการจะรับส่ง เช่น มีข้อมูล 2 เรคคอร์ด แต่ละข้อมูลมี 20 ค่า ก็สามารถตั้งเป็น A1..B20 หรืออาจกำหนดให้มีค่ามากกว่าก็ได้

End-of-line : \042 สำหรับ INKEY Mode

\044 สำหรับ ADC Mode

Phase-Method : \044

Width table : Default

Include table : Default

Response :

Value : Yes

และเนื่องจากเป็นการรับข้อมูลเพียงอย่างเดียว จึงไม่จำเป็นต้องตั้งค่าในส่วนของ Transmit ค่าต่าง ๆ ที่เซ็ทไว้สามารถเก็บลงไฟล์แล้วเรียกมาใช้ใหม่ได้

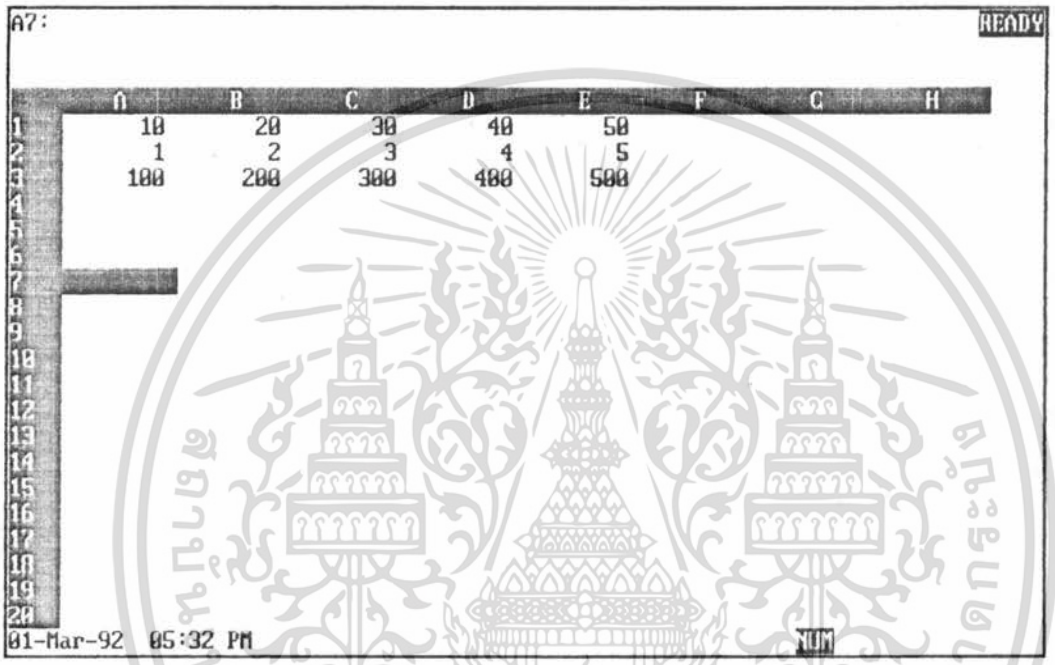
```
C3:                                                              MENU
Interface  Xon/Xoff  Duration  Capture  Transmit  Restore  Name  Quit
Enter settings for speed and type of transmission

 Interface                          Capture
    Comm-Port:        COM1            Capture-Range:      A1..A716
    Baud:             9600            End-of-Line:        \042
    Stop-Bits:        1               Parse-Method:       Char  \044
    Parity:           None              Width Table:      Default
    Length:           8                 Include Table:    Default
                                      Response:
 XON/XOFF                             Values:             Yes
    Inbound:          Disabled
    Outbound:         Disabled       Transmit
                                       Transmit-Range:    No
 Duration                              End-Of-Line:       \m
    Break (ms):       60              Response:          \j
    Timeout (sec):    10              Delay:             0
                                      Lag:                60
                                      Concurrent-Capture: No
                                    RS232 Settings: D:\LOTUS\MIC1.RCF

01-Mar-92  05:12 PM                                        NUM
```

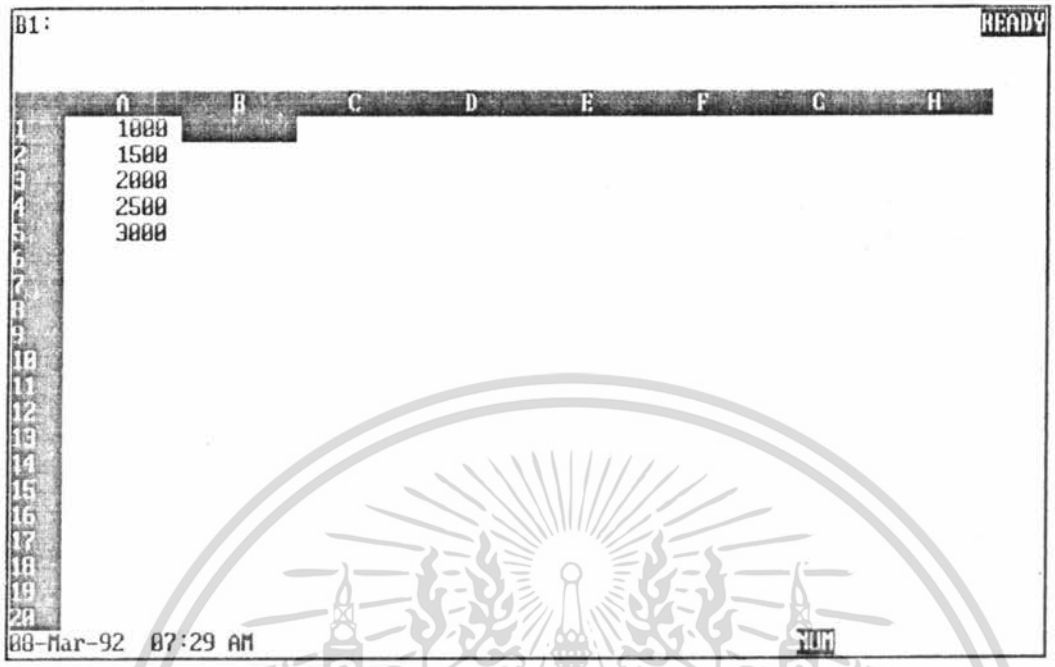
รูปที่ 7.5 ตัวอย่างค่าต่าง ๆ ที่ตั้งไว้สำหรับ INKEY Mode

เมื่อต้องการจะรับข้อมูล ทำได้โดยกลับไปเมนูหลัก แล้วเลือก Capture-Range เครื่องไมโครคอมพิวเตอร์ จะรับข้อมูลที่ส่งมาจากเครื่องไมโครเทอร์มินอลภายในเวลาที่ตั้งไว้ (Timeout) เมื่อมีการส่งข้อมูลเข้ามา ก็จะเห็นข้อมูลปรากฏบนจอภาพ กดคีย์"ENTER" จะเข้าสู่เมนูของ LOTUS 123 ซึ่งข้อมูลจะจัดเรียงอย่างอัตโนมัติ โดยถ้าเป็นข้อมูล ในโหมดของการกดคีย์ (INKEY) ข้อมูลจะเรียงในลักษณะ ดังรูป



รูปที่ 7.6 ตัวอย่างข้อมูลที่รับได้จาก INKEY Mode

ส่วนกรณีโหมดรับสัญญาณนาลอก จะมีการจัดเรียงดังนี้



รูปที่ 7.7 ตัวอย่างข้อมูลที่รับได้จาก ADC Mode

ข้อมูลที่รับมาสามารถนำไปพล๊อตกราฟได้ โดยเข้าสู่เมนูหลักของ LOTUS 123 ก่อน ซึ่งต้องกดคีย์ "7" เมนูหลักของ LOTUS 123 จะปรากฏขึ้นมาดังรูป

```
Worksheet Range Copy Move File Print Graph Data System Quit
```

รูปที่ 7.8 เมนูหลักของ Lotus 123

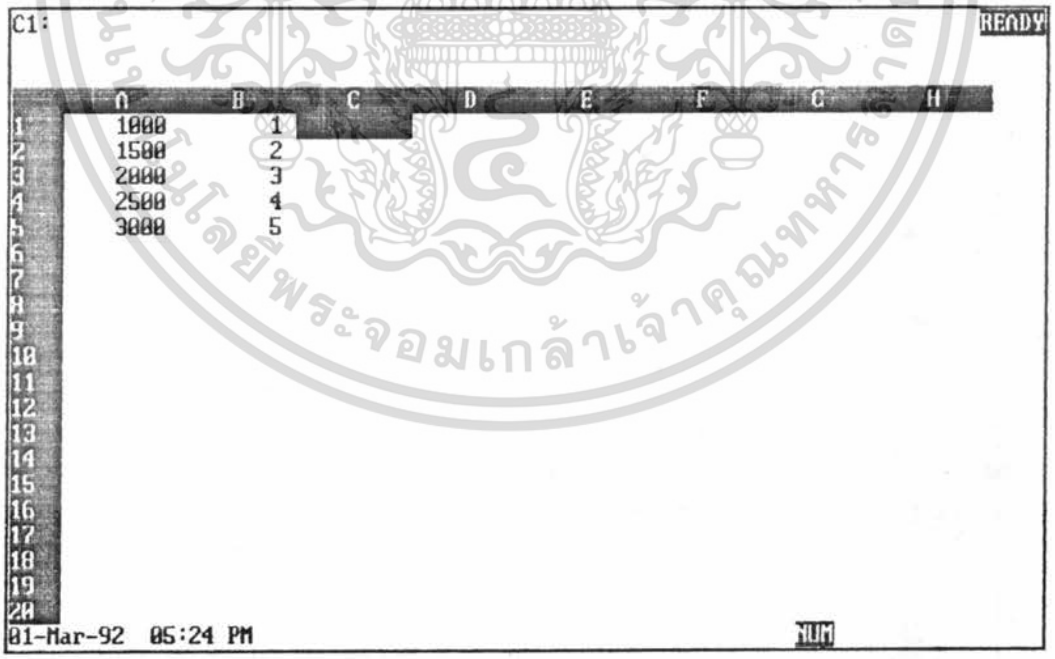
เลื่อนเคอร์เซอร์ไปที่ Graph แล้วกด "ENTER" เมนูของ Graph จะปรากฏดังรูป

Type X A B C D E F Reset View Save Option Name Quit

รูปที่ 7.9 เมนูของ Graph

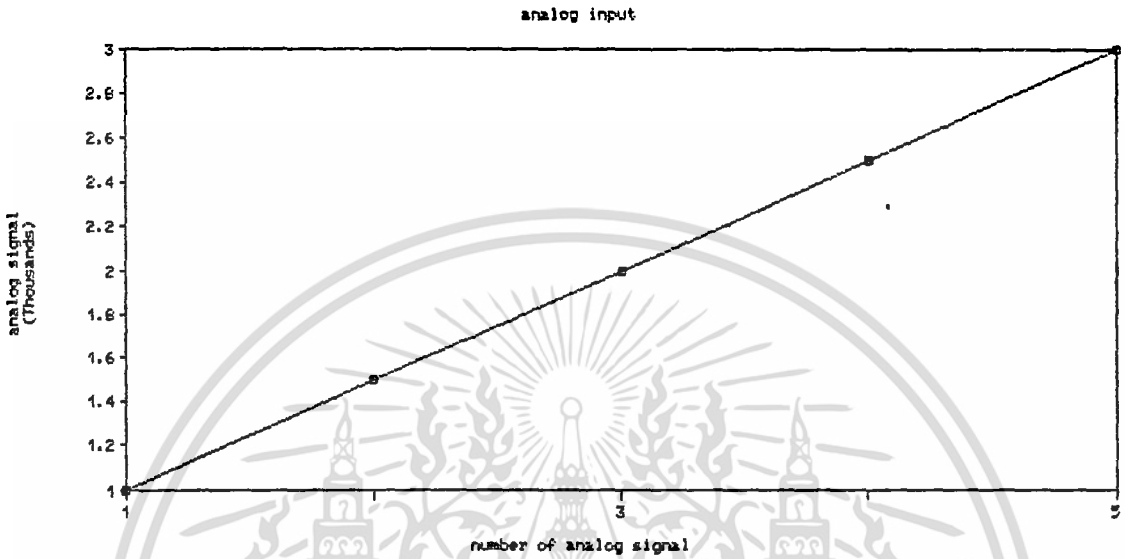
เลือก Type จะเป็นการเลือกชนิดของกราฟ เมื่อเลือกชนิดของกราฟแล้ว ตั้งขอบเขตของข้อมูล โดยการเลือก X แล้วใส่ขอบเขตของข้อมูลในแนวแกน x เช่น ถ้าต้องการพล็อตกราฟระหว่าง จำนวนของสัญญาณอนาลอก (แกน x) กับค่าสัญญาณอนาลอก (แกน y) ก็ทำได้ดังนี้

1. เลือกชนิดของกราฟ
2. เลือก A เพื่อทำการกำหนดขอบเขตของสัญญาณอนาลอกเป็น A1..A5
3. เลือกแกน x โดยในที่นี้ จะให้ตรงกับคอลัมน์ B จึงได้เป็น B1..B5
4. ป้อนเลข 1 ถึงเลข 5 ลงในคอลัมน์ B เพื่อเป็นเป็นจำนวนสัญญาณอนาลอก



รูปที่ 7.10 ตัวอย่างข้อมูลที่เตรียมพล็อตกราฟ

เลือก View เพื่อพล๊อตกราฟ กราฟจะแสดงขึ้นมาดังรูป



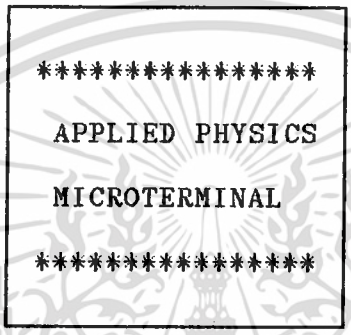
รูปที่ 7.11 ตัวอย่างกราฟของข้อมูลจาก ADC Mode

บทที่ 8

ผลการวิจัย

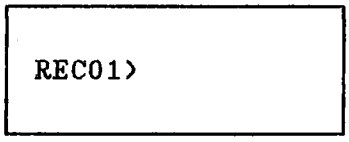
การแสดงผลบนจอ LCD

เมื่อเริ่มเปิดเครื่องไมโครเทอร์มินอล จอ LCD จะแสดงผลดังรูป

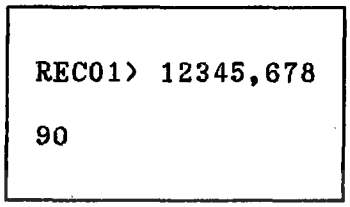


INKEY MODE

ถ้าในการเปิดเครื่องเข้ามาครั้งแรกอยู่ในโหมด INKEY จอ LCD จะแสดงผลดังรูป

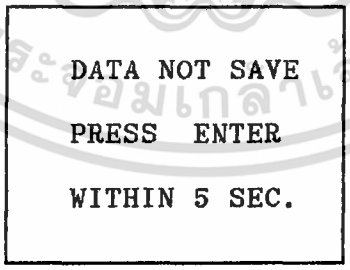


เมื่อทำการกดคีย์บอร์ด เช่น คีย์ 1 2 3 4 5 , 6 7 8 9 0 จอ LCD จะแสดงผลดังรูป



ถ้าต้องการจะเก็บข้อมูล ทำได้โดยการกดคีย์ ENTER จอ LCD จะแสดงผลเป็นเรคคอร์ดใหม่สำหรับรอรับข้อมูลชุดใหม่ ดังรูป

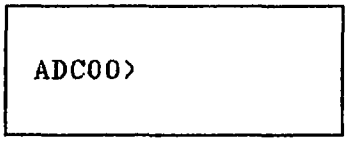
เมื่อทำการเปลี่ยนโหมดไปสู่ ADC MODE ถ้าข้อมูลในเรคคอร์ดยังไม่มีการจัดเก็บ (กดคีย์ ENTER) จอภาพ LCD จะแสดงผลเพื่อถามว่า จะจัดเก็บข้อมูลหรือไม่ ดังรูป



ถ้าต้องการจัดเก็บก็กดคีย์ ENTER ภายในเวลา 5 วินาที ถ้าไม่ต้องการจัดเก็บก็รอจนครบ 5 วินาที เครื่องจะทำการเปลี่ยนโหมดไปสู่ ADC MODE

ADC MODE

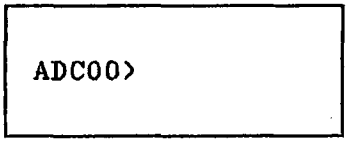
ถ้าเป็นการเข้ามาโหมดนี้ครั้งแรก จอ LCD จะแสดงผลดังรูป



เมื่อทริกเกอร์มีการทริกจะมีการรับสัญญาณอนาลอกเข้ามา แล้วแสดงผลบนจอ LCD ซึ่งจะมีการนับจำนวนสัญญาณด้วยว่าเป็นลำดับที่เท่าไร เช่น ถ้าสัญญาณอนาลอกที่เข้ามามีค่า 2000 mV เป็นสัญญาณแรก จอ LCD จะแสดงผลดังรูป

ถ้าสัญญาณอนาลอกมีจำนวนเกิน 99 ค่า โดยยังไม่มีการส่งไปยังคอมพิวเตอร์ เครื่องจะแสดง ERROR ขึ้นที่จอ LCD พร้อมกับเสียงดังยาวตลอด จนกว่าจะส่งข้อมูลไปยังคอมพิวเตอร์ (กดคีย์ SEND) ดังรูป

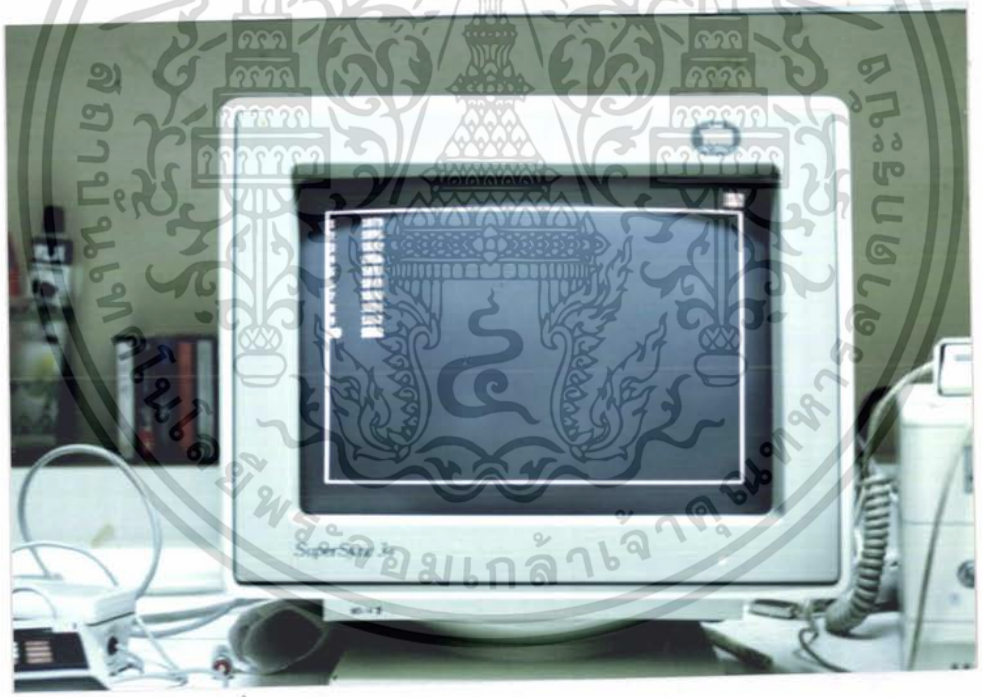
เมื่อได้กดคีย์ SEND แล้ว เครื่องไมโครเทอร์มินอลจะส่งข้อมูลค่าสัญญาณทั้ง 99 ค่าไปยังคอมพิวเตอร์ แล้วกลับมารอรับสัญญาณชุดต่อไปใหม่ ดังรูป



ในการเปลี่ยนโหมดจาก INKEY MODE ไปยัง ADC MODE หรือ จาก ADC MODE ไปยัง INKEY MODE เมื่อทำการเปลี่ยนกลับไปยังโหมดเดิมอีกที ค่าต่าง ๆ เช่น ลำดับสัญญาณอนาลอก หรือ ลำดับเรกคอร์ดในโหมดเดิมจะยังคงอยู่ และจะแสดงขึ้น LCD ให้เห็นดังเดิมเหมือนกับก่อนการเปลี่ยนโหมดไป

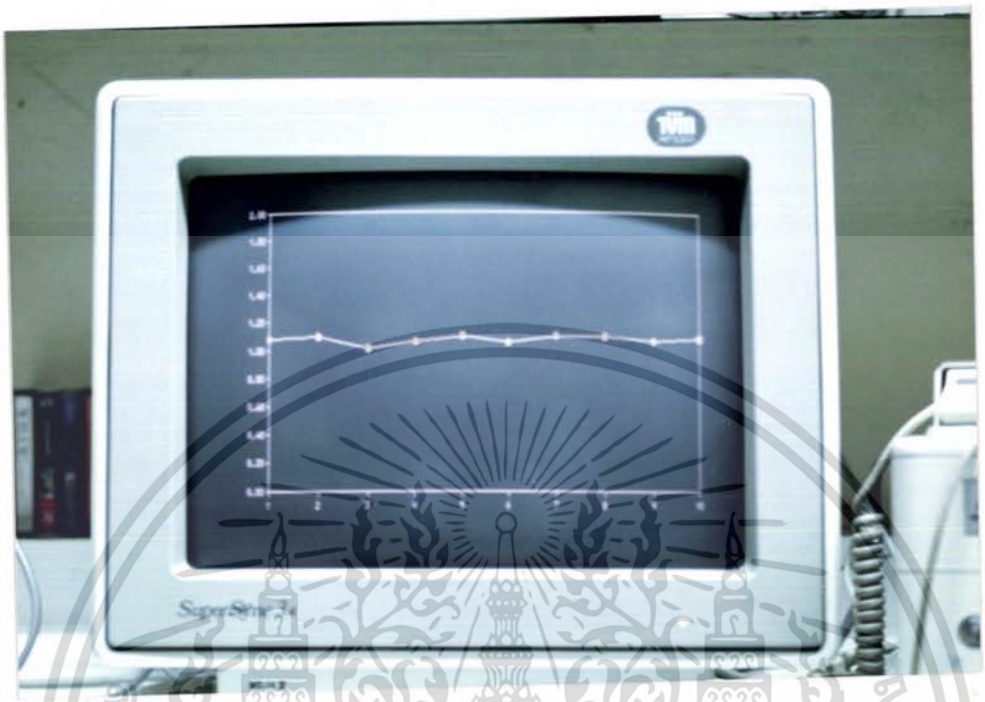
การรับข้อมูลของเครื่องคอมพิวเตอร์

ข้อมูลที่ส่งมาจากเครื่องไมโครเทอร์มินอล จะรับโดยเครื่องคอมพิวเตอร์ที่โปรแกรม LOTUS MEASURE ซึ่งได้กล่าวถึงในหัวข้อ 7.2 แล้ว ถ้ามีการส่งข้อมูล เช่น ข้อมูลที่เป็นสัญญาณอนาลอก โปรแกรม LOTUS MEASURE ที่เครื่องคอมพิวเตอร์จะแสดงข้อมูลที่รับได้ ดังรูป

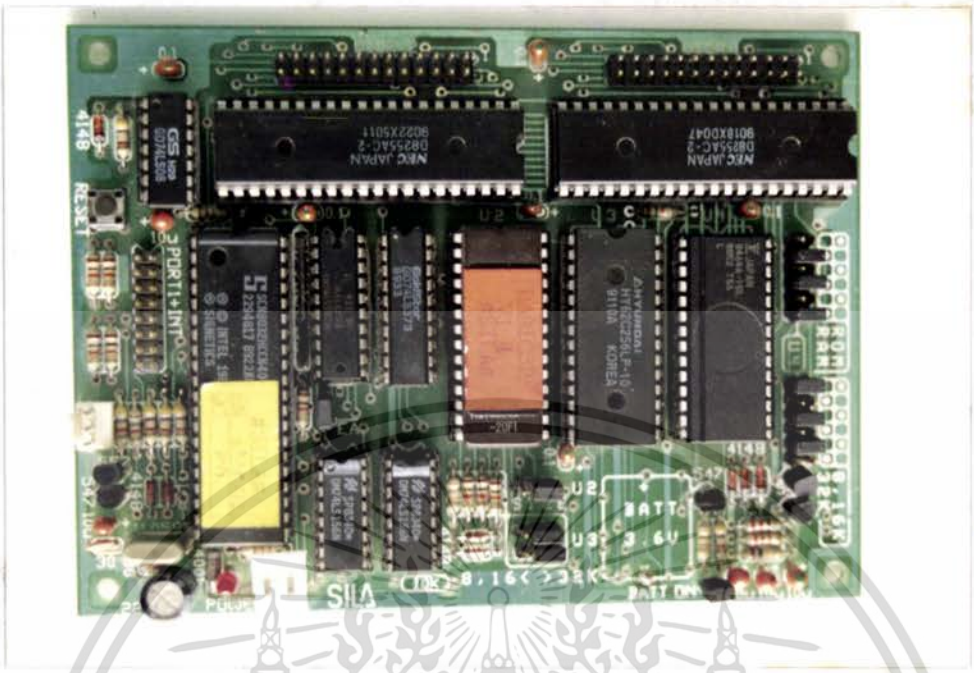


รูปที่ 8.1 ข้อมูลที่รับได้ทางโปรแกรม LOTUS MEASURE

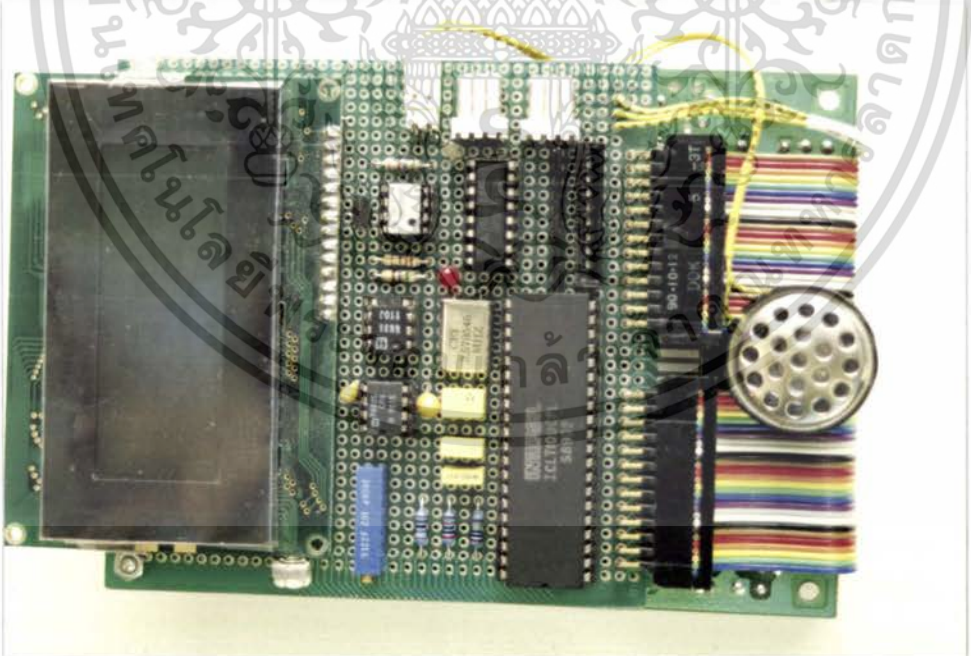
เมื่อนำไปทำการพล็อตกราฟ จะได้กราฟลักษณะดังรูป



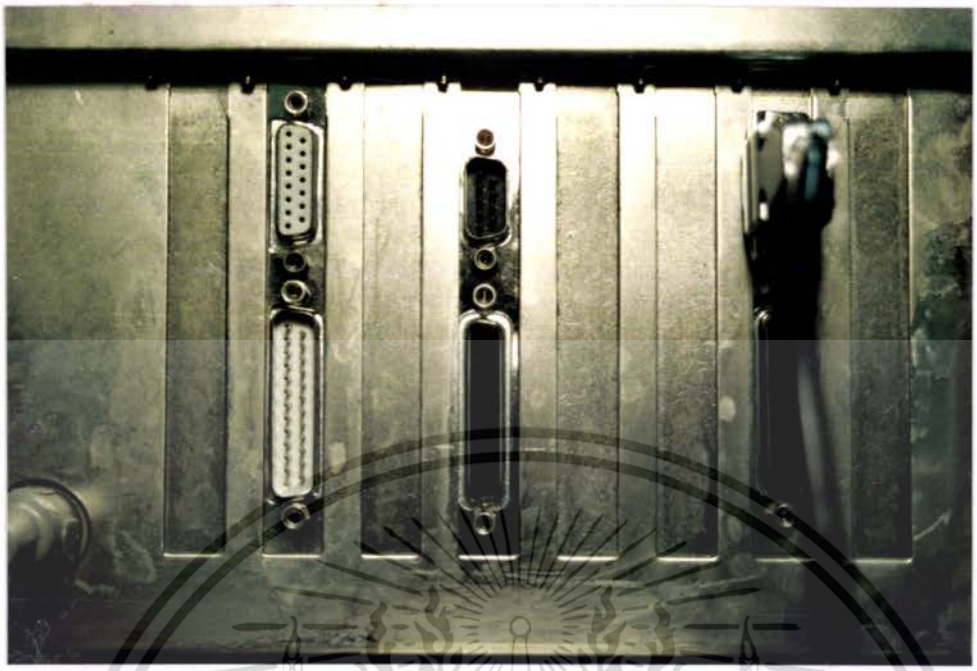
รูปที่ 8.2 ลักษณะกราฟของข้อมูลที่ทำการพล็อตโดยโปรแกรม LOTUS MEASURE



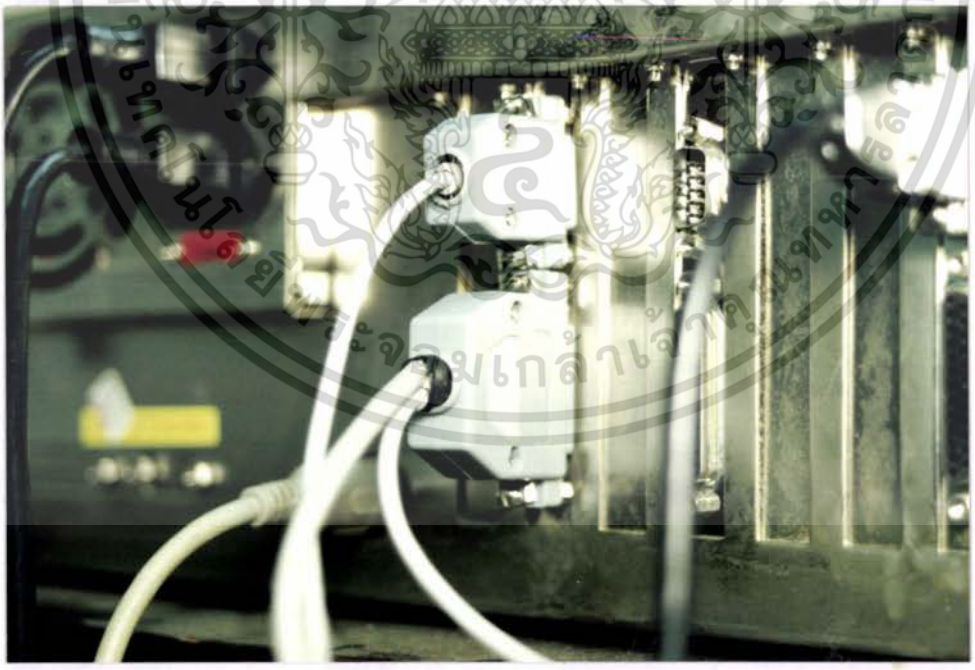
รูปที่ 8.3 Control pack ANT32



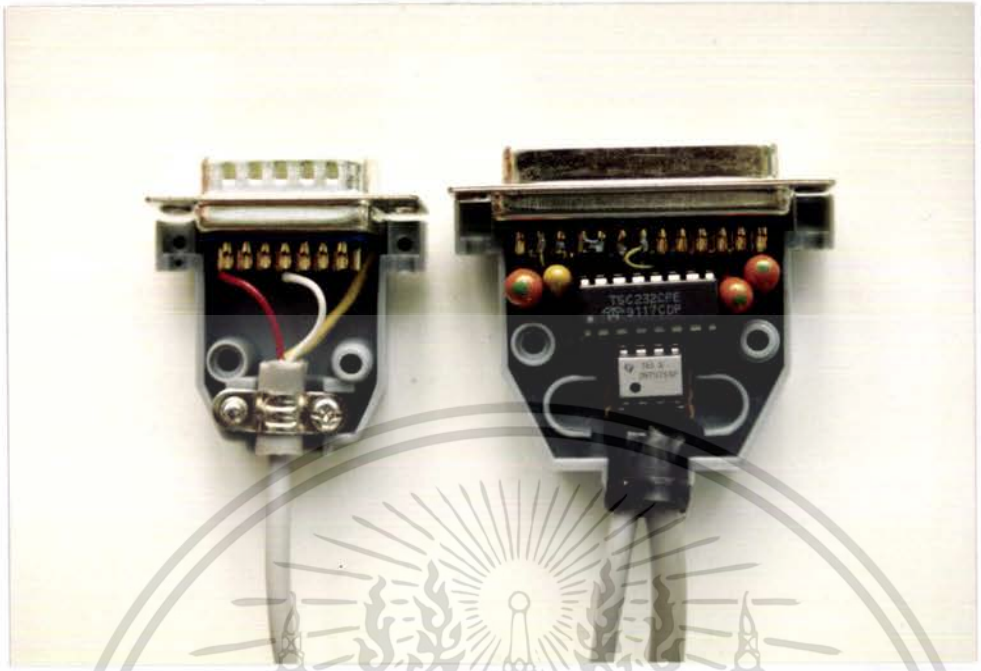
รูปที่ 8.4 การจัดอุปกรณ์ของเครื่องไมโครเทอร์มินอล



รูปที่ 8.5 ส่วนการเชื่อมต่อ RS 232C ของเครื่องคอมพิวเตอร์



รูปที่ 8.6 การเชื่อมต่อ RS 422A เข้ากับ RS 232C ของเครื่องคอมพิวเตอร์



รูปที่ 8.7 การจัดวางวงจรเชื่อมต่อ RS 422A ภายในขั้วต่อ DB25

รูปที่ 8.8 ลักษณะของสัญญาณที่ผ่านการสื่อสารอนุกรม RS 422A

รูปที่ 8.9 เครื่องไมโครเทอร์มินอล

รูปที่ 8.10 ส่วนการเชื่อมต่อของเครื่องไมโครเทอร์มินอล

บทที่ 9

สรุปและข้อเสนอแนะ

9.1 บทสรุป

โครงการพิเศษนี้ได้พัฒนาขึ้นมาเพื่อตอบสนอง ความต้องการในการบันทึกข้อมูล โดยมีบอร์ด ANT32 ตัวควบคุมหลักของระบบ ซึ่งบอร์ดนี้ใช้ไมโครคอนโทรลเลอร์เบอร์ 8032 เป็นตัวประมวลผล โปรแกรมควบคุมการทำงานของระบบเขียนด้วย ภาษา BASIC 52

การทำงานแยกเป็น 2 โหมด คือ

1. โหมดรับข้อมูลทางคีย์บอร์ด (INKEY Mode)
2. โหมดรับข้อมูลที่เป็นสัญญาณอนาลอก (ADC Mode)

การส่งผ่านข้อมูลเป็นแบบอนุกรมตามมาตราฐาน RS 422A โดยคอมพิวเตอร์ จะรับข้อมูลผ่านทางโปรแกรมสำเร็จรูป LOTUS MEASURE เพื่อประมวลผล

ความสามารถพิเศษของเครื่องไมโครเทอร์มินอล มีดังนี้

1. สามารถส่งข้อมูลไปยังคอมพิวเตอร์ได้ไกลถึง 1 กิโลเมตร
2. สามารถป้องกันสัญญาณรบกวนในระหว่างการสื่อสารข้อมูล
3. สามารถแก้ไขข้อมูลได้

9.2 ข้อเสนอแนะ

โครงงานพิเศษนี้เป็นการพัฒนาเพียงเบื้องต้น สามารถที่จะพัฒนาเพิ่มเติมให้สมบูรณ์แบบยิ่งขึ้นได้ ดังนี้

1. จากคุณสมบัติของ RS 422A ทำให้สามารถใช้เครื่องไมโครเทอร์-มินอลหลายตัว ต่อเข้ากับเครื่องคอมพิวเตอร์เพียงเครื่องเดียวในการประมวลผลข้อมูลที่ได้รับจากเครื่องไมโครเทอร์มินอลทั้งหลาย

2. ควรจะเขียนโปรแกรมในการรับข้อมูลของคอมพิวเตอร์ขึ้นเองให้ใช้งานง่าย มีการตรวจสอบสัญญาณของการส่งและการรับ (Handsbaking)

3. อาจจะพัฒนาจากการสื่อสารตามสายไปเป็นแบบไร้สาย เช่น การสื่อสารสารโดยใช้สัญญาณอินฟราเรด สัญญาณวิทยุ เป็นต้น เพื่อเกิดความสะดวกสำหรับผู้ใช้งาน

4. ในโหมดการรับข้อมูลทางคีย์บอร์ด ควรจะสามารถรับข้อมูลที่เป็นอักขระได้

ภาคผนวก ก

โปรแกรมจัดระบบ

```
80 ************************************************************
84 *                                                          *
86 *                    INITIAL PROGRAM                       *
88 *                                                          *
90 ************************************************************

            ;INITIAL FOR SET LCD,STRING,CONTROL PORT,DIMENSION
            P          : 8255 USER 1
            P1         : 8255 USER 2
            LINE 1000 : EPULSE ROUTINE

99 STRING 150,8
110 P=0F800H
115 P1=0FC00H
120 XBY(P+3)=88H            ;PORT A,B,CLOW ARE OUTPUT
                            ;AND CHI IS INPUT
125 XBY(P1+3)=93H           ;PORT A,B,CLO ARE INPUT
                            ;AND CHI IS OUTPUT
126 ************************SET LCD*********************************
130 XBY(P+1)=00H
140 XBY(P)=38H              ;FUNCTION SET
150 GOSUB 1000
160 XBY(P)=0CH              ;DISPLAY AND CURSOR OFF
170 GOSUB 1000
172 XBY(P)=06H              ;ENTRY MODE SET
174 GOSUB 1000
180 XBY(P)=01H              ;CLEAR ALL DISPLAY
190 GOSUB 1000
                ********************************
            INDEX    : VARIABLE FOR RUN TO SAVE IN RAM
            COUNTER : VARIABLE FOR POSITION ON LCD
```

```
                W,X      : NUMBER RECORD OF INKEY MODE
                W1,W2    : NUMBER SIGNAL OF ADC MODE
                           W,W1 -> HIGH ORDER
                           X,W2 -> LOW ORDER
                ASK      : VARIABLE FOR CHECK NOT SAVING IN RAM
                R        : ADRESS POSITION "INKEY MODE"
                R1       : ADRESS POSITION "ADC MODE"
                MN       : VARIABLE FOR CHECK FIRST TIME "INKEY MODE"
                MN2      : VARIABLE FOR CHECK FIRST TIME "ADC MODE"
                MN3      : VARIABLE FOR CHECK CHANGE MODE
                           MN3 = 00  INKEY MODE -> ADC MODE
                           MN3 = 01  ADC MODE   => INKEY MODE

199 INDEX=0
200 COUNTER=0
201 ASK=01                      ;SET FOR NOT SAVE " TITLE IN RAM "
202 W1=30H : W2=30H             ;SET FIRST ADC "01"
203 W=30H : X=31H               ;SET FIRST RECORD " 01 "
205 MN=00H:MN2=01H:MN3=01H      ;SET FOR INITIAL
207 R=1000H                     ;FIRST RAM ADDRESS FOR INKEY MODE
209 R1=7B00H                    ;FIRST RAM ADDRESS FOR ADC MODE
210 $(4)=" " : $(5)="3" : $(6)="6" : $(7)="9" : $(9)="2" : $(10)="5"
211 $(11)="8" : $(12)="0" : $(13)="1" : $(14)="4" : $(15)="7"
                                ;CHAR FOR NUMBER KEY
220 DIM A(200)                  ;SET DIMENSION
221 ********************** DISPLAY TITLE ***************************

                ;SHOW1 " ****************************** "
                       "         APPLIED PHYSICS        "
                       "          MICROTERMINAL         "
                       " ****************************** "
```

```
;SHOW2 "\          DESIGNED BY
       "              NARONG
       "               AND
       "            PATHOMSAK

      LINE  975 : WRITE BYTE ROUTINE
            1000 : EPULSE ROUTINE
            2300 : DELAY ROUTINE

FOR I=1 TO 192
READ N
IF I>64 THEN DISP=N : GOSUB 975
                      ;WRITE SHOW1 TO LCD
IF I=128 THEN GOSUB 2300 : XBY(P)=01H : GOSUB 1000
                      ;CLEAR LCD AND WRITE SHOW2 TO LCD
NEXT I : GOSUB 2300 : XBY(P)=01H : GOSUB 1000
                      ;DELAY AND CLEAR LCD
W=30H : X= 31H        ;SET FIRST RECORD " RE(01) "
ASK=00                ;TO NORMAL CONDITION
INDEX=0

'*************************************************************
'*                                                           *
'*                  END OF INITIAL PROGRAM                   *
'*                                                           *
'*************************************************************
```

```
300 ************************************************************
310 **********                MAIN PROGRAM                ***********
320 ************************************************************

              ;MO = 00H   : ADC MODE
              ;MO = 01H   : INKEY MODE
              LINE  400 : INKEY MODE
                    6900 : ADC MODE ROUTINE


330 MO=XBY(P1+2)                ;IN VALUE FOR CHECK MODE
335 MO=MO.AND.01H
340 IF MO=00H THEN MN=00H : GOSUB 6900
                                ;CALL ADC MODE
345 IF MO=01H THEN MN2=01H : GOSUB 400
                                ;CALL INKEY MODE
350 GOTO 300                    ;GOTO MAIN

    ************************************************************
    **********          END OF MAIN PROGRAM          ***********
    ************************************************************
```

```
399 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
400 +++++                  MAIN OF INKEY MODE                      +++++
401 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

             ;INKEY MODE GET DATA BY PRESS KEYBOARD
             LINE 1000 : EPULSE ROUTINE
                 1100 : SHOW " REC__> " ROUTINE
                 2000 : DELAY ROUTINE
                 6500 : ASK " ENTER? " ROUTINE


402 XBY(P+1)=00H : XBY(P)=0FH : GOSUB 1000
                              ;CURSOR ON
403 IF MN=MO THEN GOTO 410    ;GO LINE 410 , IF NOT FIRST TIME
404 GOSUB 1100
405 PWM 440,440,100           ;BEEP
406 PWM 233,233,80
410 GOSUB 2000
415 MN=MO                     ;SET FOR LINE 403 IS TRUE(NEXT TIME)
417 MN3=00H                   ;SET FOR THIS IS INKEY MODE
420 IF INKEY<>0F0H THEN GOTO 410
                              ;STILL PRESS ?
430 GOSUB 2000
433 MO=XBY(P1+2)
434 MO=MO.AND.01H
435 IF MO=00H THEN GOSUB 6500 : GOTO 540
                              ;CHANGE MODE? YES CALL # ENTER? #
440 IF INKEY=0F0H THEN GOTO 430
                              ;NO PRESS ?
450 ******************* GET INTERNAL CODE ************************
455 RESTORE
```

```
460 FOR I=0 TO POSI
470 READ N                          ;GET INTERNAL CODE
480 NEXT I

500 ************************CHECK NUMBER****************************

                LINE  975 : WRITE BYTE ROUTINE
                      2200 : EDIT LEFT KEY ROUTINE
                      2250 : EDIT RIGHT KEY ROUTINE
                      2500 : SEND KEYROUTINE
                      3500 : ENTER KEY ROUTINE

510 IF N<=9 THEN DISP=ASC($(I),1) :PWM 330,330,75 : GOSUB 975
                                    ;NUMBER KEY
520 IF N=10 THEN DISP=2CH : PWM 330,330,75 : GOSUB 975
                                    ;COMMA KEY
525 IF N=14 THEN GOSUB 2200  ;EDIT LEFT KEY
527 IF N=15 THEN GOSUB 2250  ;EDIT RIGHT KEY
528 IF N=11 THEN GOSUB 3500  ;ENTER KEY
530 IF N=12 THEN GOSUB 2500  ;SEND KEY
535 IF N=13 THEN DISP=2EH : PWM 330,330,75 : GOSUB 975
                                    ;POINT KEY
540 RETURN                          ;RETURN TO MAIN PROGRAM

545 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
550 +++++           END OF MAIN INKEY MODE                        +++++
555 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
950 %%%%%%%%%%%%%%%%%%%% SET DD RAM ROUTINE %%%%%%%%%%%%%%%%%%%%

              ;SET POSITION ON LCD
              LINE 1000 : EPULSE ROUTINE

955 DDADDS=DDADDS.OR.80H      ;SET BIT 7 = 1
960 XBY(P)=DDADDS
965 XBY(P+1)=00H              ;RS=0,R/W=0
970 GOSUB 1000
974 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

975 %%%%%%%%%%%%%%%%%%%% WRITE BYTE ROUTINE %%%%%%%%%%%%%%%%%%%%

              ;WRITE CHARACTER TO LCD
              LINE  950 : SET DD RAM ROUTINE
                   1000 : EPULSE ROUTINE
                   3000 : SAVE DATA ROUTINE

976 IF COUNTER=64 THEN XBY(P)=01H : XBY(P+1)=00H : GOSUB 1000 :
COUNTER=0 : EDIT=00H          ;CLEAR LCD TO NEXT PAGE
980 XBY(P+1)=01H              ;CONTROL WRITE DATA
985 XBY(P)=DISP               ;OUT CHARACTER TO LCD
987 GOSUB 1000
988 COUNTER=COUNTER+1
989 IF ASK=01 THEN GOTO 993   ;NOT SAVE TITLE " ENTER 7 " IN RAM
991 INDEX=INDEX+1
992 GOSUB 3000
993 IF COUNTER=16 THEN DDADDS=40H : GOSUB 950
994 IF COUNTER=32 THEN DDADDS=10H : GOSUB 950
995 IF COUNTER=48 THEN DDADDS=50H : GOSUB 950
```

```
996 IF COUNTER=64 THEN DDADDS=5FH : GOSUB 950
                                ;SET POSITION ON LCD
                                ;SHOW CURSOR ON LAST CHAR
998 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

1000 %%%%%%%%%%%%%%%%%%%% EPULSE ROUTINE %%%%%%%%%%%%%%%%%%%%%%%

              ;ENABLE LCD TO ACCEPT COMMAND

1010 E=XBY(P+1)               ;IN PORT B
1020 E=E.OR.04H               ;SET BIT 2 = 1
1030 XBY(P+1)=E               ;OUT HIGH EDGE
1040 FOR J=1 TO 10            ;DELAY
1050 NEXT J
1060 E=E.AND.0FBH             ;SET BIT 2 = 0
1070 XBY(P+1)=E               ;OUT LOW EDGE
1080 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

1100 %%%%%%%%%%%%%%%% SHOW " REC__> " ROUTINE %%%%%%%%%%%%%%%%%%

              ;SET REC__> FOR SHOW ON LCD
              LINE 1000 : EPULSE ROUTINE
                   1500 : WRITE ____> ROUTINE

1102 XBY(P)=01H : XBY(P+1)=00H
                                ;CLEAR LCD
1103 GOSUB 1000
1105 Y=52H                      ; " R "
1110 GOSUB 1500
```

```
1115 Y=45H                      ; " E "
1120 GOSUB 1500
1125 Y=43H                      ; " C "
1130 GOSUB 1500
1131 Y=W                        ; " HIGH NUMBER "
1132 GOSUB 1500
1135 Y=X                        ; " LOW NUMBER "
1140 GOSUB 1500
1145 Y=3EH                      ; " > "
1150 GOSUB 1500
1152 COUNTER=6
1153 EDIT=3EH
1155 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

1500 %%%%%%%%%%%%%%%%%% WRITE _____> ROUTINE %%%%%%%%%%%%%%%%%%

          ;WRITE " _____> " TO LCD
          LINE 1000 : EPULSE ROUTINE

1505 XBY(P+1)=01H
1506 XBY(P)=Y                   ;WRITE TO LCD
1508 GOSUB 1000
1510 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

2000 %%%%%%%%%%%%%%%%%% SCAN KEY ROUTINE %%%%%%%%%%%%%%%%%%%%%

2020 CODE=0
2030 COL=00H
2040 DO : XBY(P+2)=COL          ;OUT COLUMN
```

```
2050 INKEY=XBY(P+2)          ;IN PORT C
2060 INKEY=INKEY.AND.0F0H    ;MARK PCLOW IS 0
2070 IF INKEY=0E0H THEN POSI=CODE : GOTO 2170
2080 CODE=CODE+1
2090 IF INKEY=0D0H THEN POSI=CODE : GOTO 2170
2100 CODE=CODE+1
2110 IF INKEY=0B0H THEN POSI=CODE : GOTO 2170
2120 CODE=CODE+1
2130 IF INKEY=70H THEN POSI=CODE : GOTO 2170
2140 CODE=CODE+1
2150 COL=COL+1
2160 WHILE COL<4
2170 RETURN
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

2200 %%%%%%%%%%%%%%%%%% EDIT LEFT KEY ROUTINE %%%%%%%%%%%%%%%%%%

          ;TO EDIT LEFT CHARACTER
          LINE  950 : SET DD RAM ROUTINE
               1000 : EPULSE ROUTINE

2205 PWM 660,660,75
2207 IF COUNTER=6.AND.EDIT=3EH THEN GOTO 2240
                             ;DON'T EDIT LEFT
2210 IF COUNTER=0 THEN GOTO 2240
                             ;DON'T EDIT FIRST POSITION ON LCD
2215 IF COUNTER=16 THEN DDADDS=10H : GOSUB 950
                             ;LINE2 TO LINE1
2220 IF COUNTER=32 THEN DDADDS=50H : GOSUB 950
                             ;LINE3 TO LINE2
```

```
2225 IF COUNTER=48 THEN DDADDS=20H : GOSUB 950
                                ;LINE4 TO LINE3
2230 XBY(P)=10H
2231 XBY(P+1)=00H               ;SHIFT LEFT
2231 GOSUB 1000
2233 INDEX=INDEX-1
2235 COUNTER=COUNTER-1
2237 FOR I=1 TO 50
2239 NEXT I
2240 XBY(P+2)=03H : RE=XBY(P+2)
2241 RE=RE.AND.0F0H
2242 IF RE=70H THEN GOTO 2200
                                ;CHECK REPEAT KEY ?
2244 RETURN
                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

2250 %%%%%%%%%%%%%%%%%% EDIT RIGHT KEY ROUTINE %%%%%%%%%%%%%%%%

             ;TO EDIT RIGHT CHARACTER
             LINE  950 : SET DD RAM ROUTINE
                  1000 : EPULSE ROUTINE

2252 PWM 660,660,75
2255 IF COUNTER=15 THEN DDADDS=40H : GOSUB 950 : GOTO 2280
                                ;LINE1 TO LINE2
2260 IF COUNTER=31 THEN DDADDS=10H : GOSUB 950 : GOTO 2280
                                ;LINE2 TO LINE3
2265 IF COUNTER=47 THEN DDADDS=50H : GOSUB 950 : GOTO 2280
                                ;LINE3 TO LINE4
2270 IF COUNTER=64 THEN GOTO 2285
                                ;DON'T EDIT LAST POSITION
```

```
2275 XBY(P)=14H
2276 XBY(P+1)=00H              ;SHIFT RIGHT
2277 GOSUB 1000
2280 COUNTER=COUNTER+1
2283 INDEX=INDEX+1
2285 FOR I=1 TO 50
2290 NEXT I
2292 XBY(P+2)=01H : RE=XBY(P+2)
2293 RE=RE.AND.0F0H
2294 IF RE=70H THEN GOTO 2250
                               ;CHECK REPEAT KEY ?
2296 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

2300 %%%%%%%%%%%%%%%%%%%% DELAY ROUTINE %%%%%%%%%%%%%%%%%%%%

2305 FOR Z=1 TO 1000
2310 NEXT Z
2315 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

2500 %%%%%%%%%%%%%%%%%%%% SEND KEY ROUTINE %%%%%%%%%%%%%%%%%%%%

              ;FOR SEND DATA IN RAM TO COMPUTER
              ;MO = 00 : SEND FOR ADC MODE
              ;MO = 01 : SEND FOR INKEY MODE

2501 PWM 196,196,75
2502 XBY(P1+2)=10H             ;LED ON
2503 IF MO=00H THEN V=R1 : V=V-7B00H : S=7B00H
                               ;SEND FOR ADC MODE
```

```
2275 XBY(P)=14H
2276 XBY(P+1)=00H              ;SHIFT RIGHT
2277 GOSUB 1000
2280 COUNTER=COUNTER+1
2283 INDEX=INDEX+1
2285 FOR I=1 TO 50
2290 NEXT I
2292 XBY(P+2)=01H : RE=XBY(P+2)
2293 RE=RE.AND.0F0H
2294 IF RE=70H THEN GOTO 2250
                               ;CHECK REPEAT KEY ?
2296 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%

2300 %%%%%%%%%%%%%%%%%%%% DELAY ROUTINE %%%%%%%%%%%%%%%%%%%%

2305 FOR Z=1 TO 1000
2310 NEXT Z
2315 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%

2500 %%%%%%%%%%%%%%%%%%%% SEND KEY ROUTINE %%%%%%%%%%%%%%%%%%%%

              ;FOR SEND DATA IN RAM TO COMPUTER
              ;MO = 00 : SEND FOR ADC MODE
              ;MO = 01 : SEND FOR INKEY MODE

2501 PWM 196,196,75
2502 XBY(P1+2)=10H             ;LED ON
2503 IF MO=00H THEN V=R1 : V=V-7B00H : S=7D00H
                               ;SEND FOR ADC MODE
```

```
2504 IF MO=01H THEN V=R : V=V-1000H : S=1000H
                                ;SEND FOR INKEY MODE
2505 FOR K=1 TO V
2507 A1=XBY(S)                  ;GET DATA IN RAM TO DIMENSION
2509 S=S+1
2510 PRINT CHR(A1)
2515 NEXT K
2516 PWM 262,262,200
2517 PWM 196,196,200
2518 PWM 330,330,200
2519 XBY(P1.2)=00H              ;LED OFF
2520 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

3000 %%%%%%%%%%%%%%%%%%%% SAVE DATA ROUTINE %%%%%%%%%%%%%%%%%%%%

3005 A(INDEX)=DISP              ;SAVE DATA IN DIMENSION
3010 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

3500 %%%%%%%%%%%%%%%%%%%% ENTER KEY ROUTINE %%%%%%%%%%%%%%%%%%%%

              ;ENTER FOR SAVE DATA IN RAM
              ;AND INCREASE REC__>
              LINE 1000 : EPULSE ROUTINE
                   1100 : SHOW " REC__> " ROUTINE

3501 PWM 262,262,100
3502 IF INDEX=0 THEN GOTO 3537
3505 FOR L=1 TO INDEX
3510 XBY(R)=A(L)                ;SAVE DATA FROM DIMENSION TO RAM
3515 R=R+1
```

```
2504 IF MO=01H THEN V=R : V=V-1000H : S=3000H
                                ;SEND FOR INKEY MODE
2505 FOR K=1 TO V
2507 A1=XBY(S)                  ;GET DATA IN RAM TO DIMENSION
2509 S=S+1
2510 PRINT CHR(A1)
2515 NEXT K
2516 PWM 262,262,200
2517 PWM 196,196,200
2518 PWM 330,330,200
2519 XBY(P1+2)=00H              ;LED OFF
2520 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

3000 %%%%%%%%%%%%%%%%%% SAVE DATA ROUTINE %%%%%%%%%%%%%%%%%%

3005 A(INDEX)=DISP              ;SAVE DATA IN DIMENSION
3010 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

3500 %%%%%%%%%%%%%%%%%% ENTER KEY ROUTINE %%%%%%%%%%%%%%%%%%

               ;ENTER FOR SAVE DATA IN RAM
               ;AND INCREASE REC__>
               LINE 1000 : EPULSE ROUTINE
                    1100 : SHOW " REC__> " ROUTINE

3501 PWM 262,262,100
3505 FOR L=1 TO INDEX
3510 XBY(R)=A(L)                ;SAVE DATA FROM DIMENSION TO RAM
3515 R=R+1
```

```
3517 NEXT L
3521 A(L)=2AH                    ;END OF RECORD
3522 XBY(R)=A(L)
3523 R=R+1
3524 X=X+1                       ;INCREASE NUMBER OF RECORD
3526 IF X=3AH THEN X=30H : W=W+1
                                 ;CHANGE HIGH NUMBER
3527 XBY(P)=01H : XBY(P+1)=00H : GOSUB 1000
                                 ;CLEAR LCD
3528 IF MO=00H THEN GOTO 3530
3529 GOSUB 1100
3530 INDEX=0
3532 PWM 262,262,100
3533 PWM 196,196,200
3537 RETURN
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

6500 %%%%%%%%%%%%%%%%%% ASK ENTER ? ROUTINE %%%%%%%%%%%%%%%%%%

          ;ASK WHEN CHANGE MODE FROM INKEY TO ADC MODE
          ;IF PRESS ENTER KEY THEN SAVE DATA IN RAM
          ;IF NOT PRESS THEN NOT SAVE DATA IN RAM
          LINE  950 : SET DDRAM ROUTINE
                975 : WRITE BYTE ROUTINE
               1000 : EPULSE ROUTINE
               3500 : ENTER KEY ROUTINE

6501 XBY(P+1)=00H
6502 XBY(P)=0CH                  ;CURSOR OFF
6503 GOSUB 1000
6504 RESTORE
```

```
6505 IF COUNTER<7 THEN GOTO 6580
                                ;NO DATA
6506 ASK=01 : COUNTER=0
                                ;SET FOR NOT SAVE " TITLE " IN RAM
6507 XBY(P+1)=00H
6510 XBY(P)=01H                 ;CLEAR LCD
6511 GOSUB 1000
6512 DDADDS=00H                 ;SET DDRAM
6513 GOSUB 950
6515 FOR I=1 TO 63
6520 READ N
6522 DISP=N
6525 IF I>16 THEN GOSUB 975  ;SHOW TITLE
6530 NEXT I
6535 FOR F=1 TO 10
6540 FOR I=1 TO 20
6545 XBY(P+2)=00H               ;ENTER KEY
6550 EN=XBY(P+2)
6555 IF EN=0B0H THEN GOSUB 3500 : GOTO 6580
                                ;IF PRESS ENTER KEY THEN CALL ROUTINE
6560 NEXT I
6565 PWM 330,330,75
6570 NEXT F
6580 ASK=00                     ;RETRIVE VALUE
6583 MN=00H
6585 RETURN
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```
6505 IF COUNTER<7 THEN GOTO 6580
                                ;NO DATA
6506 ASK=01 : COUNTER=0
                                ;SET FOR NOT SAVE " TITLE " IN RAM
6507 XBY(P+1)=00H
6510 XBY(P)=01H                 ;CLEAR LCD
6511 GOSUB 1000
6512 DDADDS=00H                 ;SET DDRAM
6513 GOSUB 950
6515 FOR I=1 TO 63
6520 READ N
6522 DISP=N
6525 IF I>16 THEN GOSUB 975  ;SHOW TITLE
6530 NEXT I
6535 FOR F=1 TO 10
6540 FOR I=1 TO 20
6545 XBY(P+2)=00H               ;ENTER KEY
6550 EN=XBY(P+2)
6555 IF EN=0B0H THEN GOSUB 3500 : GOTO 6580
                                ;IF PRESS ENTER KEY THEN CALL ROUTINE
6560 NEXT I
6565 PWM 330,330,75
6570 NEXT F
6580 ASK=00                     ;RETRIVE VALUE
6585 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
6899 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
6900 +++++                    MAIN OF ADC MODE                     +++++
6901 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

              ;OFF CURSOR
              ;WAIT FOR TRIGGER
              ;THEN GET ANALOG SIGNAL AND SAVE IN RAM
              LINE 1000 : EPULSE ROUTINE
                   2500 : SEND KEY ROUTINE
                   7010 : GET SIGNAL & SAVE IN RAM ROUTINE
                   8000 : SHOW ADC__> ROUTINE

6902 XBY(P+1)=00H
6903 XBY(P)=0CH                ;CURSOR OFF
6904 GOSUB 1000
6905 IF MN3=00H THEN COUNTER=6 : MN3=01H : GOTO 6920
                               ;CHECK INKEY-->ADC MODE
6910 IF MN2=MO THEN GOTO 6927
                               ;CHECK FIRST TIME OF ADC MODE ?
6915 COUNTER=6
6920 XBY(P)=01H : XBY(P+1)=00H : GOSUB 1000
                               ;CLEAR LCD
6922 GOSUB 8000
6925 PWM 555,555,100
6926 PWM 440,440,80
6927 XBY(P+2)=00H : SEN=XBY(P+2)
6928 IF SEN=0D0H THEN GOSUB 2500
                               ;CHECK PRESS SEND KEY ?
6929 MN2=MO                    ;SET FOR NOT FIRST TIME
6930 TRIG=XBY(P1+2)
6932 TRIG=TRIG.AND.02H
```

```
6935 IF TRIG=00H THEN GOSUB 7010
                                ;CHECK TRIG SIGNAL ?
6945 RETURN                     ;RETURN TO MAIN PROGRAM

6946 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
6947 +++++                 END OF MAIN ADC MODE                  +++++
6948 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

7010 %%%%%%%%%%%% GET SIGNAL & SAVE IN RAM ROUTINE %%%%%%%%%%%%

7011 *************** CONVERT 12 BITS TO 16 BITS ******************

              ;CONVERT FOR CAN SHOW ON LCD
              ;INCREASE NUMBER OF ANALOG SIGNAL
              ;NUMBER OF ANALOG SIGNAL NOT EXCESS 99
              ;IF EXCESS THEN SHOW ERROR AND BEEP
              ;SHOULD SEND TO COMPUTER
              LINE  950 : SET DD RAM ROUTINE
                   7200 : WRITE ANALOG VALUE ON LCD ROUTINE
                   7300 : DISPLAY ERROR ROUTINE
                   7700 : SET DD RAM FOR ADC MODE ROUTINE
                   8200 : WRITE NUM ROUTINE

7012 AN1=XBY(P1)
7015 AN2=XBY(P1+1)
7017 AN2=AN2.AND.0FH
7018 AN2=AN2*(16**2)
7019 AN3=AN1+AN2
7023 N1=INT(AN3/1000)
7025 A(0)=N1+30H                ;" THOUSANDTH UNIT "
7029 N1=AN3-(N1*1000)
```

```
7032 N2=INT(N1/100)
7034 A(1)=N2+30H              ;" HUNDREDTH UNIT "
7038 N2=N1-(N2*100)
7040 N3=INT(N2/10)
7042 A(2)=N3+30H              ;" TENTH UNIT "
7046 N3=N2-(N3*10)
7048 A(3)=N3+30H              ;" FIRST UNIT "
7049 A(4)=2CH                 ;" , "
                         **********************
7050 GOSUB 7700
7055 FOR M=0 TO 4
7060 DISP=A(M)
7065 XBY(R1)=A(M)             ;SAVE IN RAM
7070 R1=R1+1
7075 GOSUB 7200
7080 NEXT M
7085 W2=W2+1                  ;INCREASE LOW NUM
7090 IF W2=3AH THEN W2=30H : W1=W1+1
                              ;INCREASE HIGH NUM
7095 IF W1>39H THEN GOSUB 7300 : GOTO 7099
                              ;EXCESS " ADC99> " WRITE ERROR
7096 MN3=01H : DDADDS=03H
7097 GOSUB 950
7098 GOSUB 8200
7099 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
7200 %%%%%%%%%%%% WRITE ANALOG VALUE ON LCD ROUTINE %%%%%%%%%%%%
```

LINE 1000 : EPULSE ROUTINE

```
7205 XBY(P+1)=01H
7207 XBY(P)=DISP                ;WRITE TO LCD
7210 GOSUB 1000
7215 COUNTER=COUNTER+1
7220 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7300 %%%%%%%%%%%%%%%%% DISPLAY ERROR ROUTINE %%%%%%%%%%%%%%%%%

             ;SET TO SHOW ERROR WHEN ANALOG SIGNAL EXCESS 9!
             LINE  950 : SET DD RAM ROUTINE
                   1000 : EPULSE ROUTINE
                   1500 : WRITE _____> ROUTINE
                   2300 : DELAY ROUTINE
                   2500 : SEND KEY ROUTINE

7302 XBY(P+1)=00H
7303 XBY(P)=01H                 ;CLEAR LCD
7304 GOSUB 1000
7305 DDADDS=45H                 ;FIRST POSITION
7310 GOSUB 950
7315 Y=45H                      ;" E "
7320 GOSUB 1500
 325 Y=52H                      ;" R "
7330 GOSUB 1500
7335 Y=52H                      ;" R "
7340 GOSUB 1500
7345 Y=4FH                      ;" O "
7350 GOSUB 1500
 355 Y=52H                      ;" R "
7360 GOSUB 1500
```

```
7362 GOSUB 2300
7363 PWM 330,330,75
7364 XBY(P+2)=00H
7365 SEN=XBY(P+2)
7366 IF SEN<>0D0H THEN GOTO 7363
                              ;CHECK PRESS SEND KEY ?
7368 GOSUB 2500
7369 R1=7B00H
7370 W1=30H : W2=30H          ;" ADC00> "
7372 MN2=01H : MN3=01H
7374 XBY(P+1)=00H : XBY(P)=01H
                              ;CLEAR LCD
7375 GOSUB 1000
7376 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

7700 %%%%%%%%%%%%% SET DD RAM FOR ADC MODE ROUTINE %%%%%%%%%%%%%

            ;SET POSITION ON LCD
            LINE  950 : SET DD RAM ROUTINE
                  1000 : EPULSE ROUTINE
                  8000 : SHOW ADC__> ROUTINE

7705 IF COUNTER=61 THEN XBY(P)=01H : XBY(P+1)=00H : GOSUB 1000 :
COUNTER=6 : GOSUB 8000        ;CLEAR LCD TO NEXT PAGE
7710 IF COUNTER<15 THEN DDADDS=COUNTER-00H : GOSUB 950
                              ;SET LINE 1
7715 IF COUNTER>15.AND.COUNTER<31 THEN DD1=COUNTER-16 :
DDADDS=DD1+40H : GOSUB 950
                              ;SET LINE 2
7720 IF COUNTER>30.AND.COUNTER<46 THEN DD1=COUNTER-31 :
DDADDS=DD1+10H : GOSUB 950
```

```
7362 GOSUB 2300
7363 PWM 330,330,75
7364 XBY(P+2)=00H
7365 SEN=XBY(P+2)
7366 IF SEN<>0D0H THEN GOTO 7363
                              ;CHECK PRESS SEND KEY 7
7368 GOSUB 2500
7370 W1=30H : W2=30H          ;" ADC00> "
7372 MN2=01H : MN3=01H
7374 XBY(P+1)=00H : XBY(P)=01H
                              ;CLEAR LCD
7375 GOSUB 1000
7376 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

7700 %%%%%%%%%%%%% SET DD RAM FOR ADC MODE ROUTINE %%%%%%%%%%%%%%

            ;SET POSITION ON LCD
            LINE  950 : SET DD RAM ROUTINE
                  1000 : EPULSE ROUTINE
                  8000 : SHOW ADC__> ROUTINE

7705 IF COUNTER=61 THEN XBY(P)=01H : XBY(P+1)=00H : GOSUB 1000 :
COUNTER=6 : GOSUB 8000        ;CLEAR LCD TO NEXT PAGE
7710 IF COUNTER<15 THEN DDADDS=COUNTER-00H : GOSUB 950
                              ;SET LINE 1
7715 IF COUNTER>15.AND.COUNTER<31 THEN DD1=COUNTER-16 :
DDADDS=DD1+40H : GOSUB 950
                              ;SET LINE 2
7720 IF COUNTER>30.AND.COUNTER<46 THEN DD1=COUNTER-31 :
DDADDS=DD1+10H : GOSUB 950
```

```
                                ;SET LINE 3
7725 IF COUNTER>45.AND.COUNTER<61 THEN DD1=COUNTER-46 :
DDADDS=DD1+50H : GOSUB 950
                                ;SET LINE 4
7730 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

8000 %%%%%%%%%%%%%%%%%% SHOW ADC__> ROUTINE %%%%%%%%%%%%%%%%%%%%

                ;SET TO SHOW ADC__> TO LCD
                LINE 8100 : WRITE ADC__> ROUTINE

8005 Y1=41H                     ;" A "
8010 GOSUB 8100
8015 Y1=44H                     ;" D "
8020 GOSUB 8100
8025 Y1=43H                     ;" C "
8030 GOSUB 8100
8035 Y1=W1                      ;" LOW NUM "
8040 GOSUB 8100
8045 Y1=W2                      ;" HIGH NUM "
8050 GOSUB 8100
8055 Y1=3EH                     ;" > "
8060 GOSUB 8100
8065 RETURN
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
8100 %%%%%%%%%%%%%%%%%% WRITE ADC__> ROUTINE %%%%%%%%%%%%%%%%%%

                LINE 1000 : EPULSE ROUTINE

8101 XBY(P+1)=01H
8102 XBY(P)=Y1                  ;WRITE TO LCD
8103 GOSUB 1000
8115 RETURN
                        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

8200 %%%%%%%%%%%%%%%%%% WRITE NUM ROUTINE %%%%%%%%%%%%%%%%%%%%

                LINE 1000 : EPULSE ROUTINE

8205 XBY(P+1)=01H
8210 Y1=W1 : XBY(P)=Y1          ;WRITE HIGH NUM
8212 GOSUB 1000
8215 Y1=W2 : XBY(P)=Y1
8217 GOSUB 1000                 ;WRITE HIGH NUM
8220 RETURN
                        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
8300 %%%%%%%%%%%%%%% DATA OF INTERNAL CODE %%%%%%%%%%%%%%%%%%%%

8310 DATA 13,12,11,10,3,6,9,15,2,5,8,0,1,4,7,14

                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

8320 %%%%%%%%%%%%%%%%%%%% DATA OF ENTER 7 %%%%%%%%%%%%%%%%%%%%

8330 DATA 0A0H,0A0H,44H,41H,54H,41H,0A0H,4EH
8335 DATA 4FH,54H,0A0H,53H,41H,56H,45H,0A0H
8340 DATA 0A0H,0A0H,50H,52H,45H,53H,53H,0A0H
8345 DATA 0A0H,45H,4EH,54H,45H,52H,0A0H,0A0H8350 DATA 0A0H
8350 DATA 0A0H,0A0H,57H,49H,54H,48H,49H,4EH
8360 DATA 0A0H,31H,30H,0A0H,53H,45H,43H,0A0H

                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

8370 %%%%%%%%%%%%%%%%%%%% DATA OF TITLE %%%%%%%%%%%%%%%%%%%%

8380 DATA 2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH
8385 DATA 2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH
8390 DATA 0A0H,0A0H,4DH,49H,43H,52H,4FH,54H
8395 DATA 45H,52H,4DH,49H,4EH,41H,4CH,0A0H
8400 DATA 0A0H,41H,50H,50H,4CH,49H,45H,44H
8405 DATA 0A0H,50H,48H,59H,53H,49H,43H,53H
8410 DATA 2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH
8415 DATA 2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH

8420 DATA 0A0H,0A0H,50H,52H,4FH,44H,55H,43H
8430 DATA 45H,44H,0A0H,0A0H,42H,59H,0A0H,0A0H
8440 DATA 0A0H,0A0H,0A0H,0A0H,0A0H,4EH,41H,52H
```

```
8450 DATA 4FH,4EH,47H,0A0H,0A0H,0A0H,0A0H,0A0H
8460 DATA 0A0H,0A0H,0A0H,0A0H,0A0H,0A0H,41H,4EH
8470 DATA 44H,0A0H,0A0H,0A0H,0A0H,0A0H,0A0H,0A0H
8480 DATA 0A0H,0A0H,0A0H,50H,52H,41H,54H,48H
8490 DATA 4FH,4DH,53H,41H,4BH,0A0H,0A0H,0A0H
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ภาคผนวก ข

คู่มือและการประยุกต์ใช้เครื่องไมโครเทอร์มินอล

คู่มือและการประยุกต์ใช้งาน

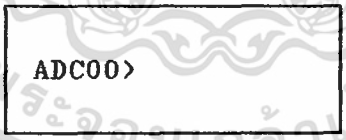
เครื่องไมโครเทอร์มินอลเป็นเครื่องมือที่ใช้เก็บข้อมูล สามารถเก็บข้อมูลผ่านทางคีย์บอร์ด และข้อมูลที่เป็นสัญญาณอนาลอกได้ และยังส่งข้อมูลไปเก็บที่เครื่องไมโครคอมพิวเตอร์ ผ่านทางการส่งข้อมูลแบบอนุกรม RS422A ด้วยอัตราการส่ง 9600 บิทต่อวินาที โดยใช้โปรแกรมสำเร็จรูป LOTUS MEASURE เป็นตัวรับ

การใช้งานเครื่องไมโครเทอร์มินอล

หลังจากป้อนไฟเลี้ยงให้กับเครื่องไมโครเทอร์มินอลแล้ว จะปรากฏตัวอักษรตามโหมดที่ตั้งไว้ จากสวิทช์เลือกโหมด ซึ่งอยู่ทางด้านขวาของจอแสดงผล

โหมด INKEY

โหมด ADC



รูปแสดงตัวอักษรที่ปรากฏบนจอ LCD ตามโหมดที่ได้เลือกไว้

การใช้งานเครื่องไมโครเทอร์มินอลทั้ง 2 โหมดมีดังต่อไปนี้

การใช้งานโหมด INKEY

การใช้งานไมโครเทอร์มินอลในโหมดนี้ จะเก็บข้อมูลในลักษณะของเรคคอร์ด โดยผู้ใช้สามารถป้อนค่าต่าง ๆ เป็นตัวเลขจากคีย์ 0-9 และสามารถแก้ไขข้อมูลที่ผิดโดยการกดคีย์ "<" หรือ ">" เพื่อเลื่อนเคอร์เซอร์ไปที่ตำแหน่งตัวเลข ที่ต้องการแก้ไข การแก้ไขจะเป็นแบบพิมพ์ทับเท่านั้น หลังจากการแก้ไขแล้ว ผู้ใช้ต้องเลื่อนเคอร์เซอร์กลับมาที่ตำแหน่งสุดท้ายของข้อมูลที่ต้องการเก็บ

การแบ่งข้อมูลภายในเรคคอร์ด จะใช้คีย์ "," เครื่องหมายดังกล่าวจะทำให้ LOTUS MEASURE เห็นความแตกต่างของข้อมูลแต่ละข้อมูลในเรคคอร์ด หลังจากที่ป้อนข้อมูลที่ต้องการแล้ว กดคีย์ "ENTER" เพื่อเก็บข้อมูลในเรคคอร์ดนั้น ๆ เมื่อข้อมูลในเรคคอร์ดแรกบันทึกแล้ว เครื่องไมโครเทอร์มินอลจะแสดงเรคคอร์ดต่อไป

เรคคอร์ดที่ 1

```
REC01> 12345,678
90
```

หลังจากกด "ENTER" LCD แสดงเรคคอร์ดใหม่

```
REC02>
```

รูปแสดงการบันทึกข้อมูลในโหมด INKEY

นอกจากนี้เราสามารถป้อนข้อมูลที่เป็นจุดทศนิยมได้โดยการใช้คีย์ "."

เมื่อเราต้องการเก็บข้อมูลที่ได้จากเครื่องไมโครเทอร์มินอล ลงบนเครื่องไมโครคอมพิวเตอร์ สามารถทำได้โดยการกดคีย์ "SEND" ข้อมูลที่ส่งไปยังเครื่องไมโครคอมพิวเตอร์จะเริ่มตั้งแต่เรคคอร์ดที่ 1 ไปจนถึงเรคคอร์ดสุดท้าย ทั้งนี้ก่อนที่จะกดคีย์ "SEND" ผู้ใช้จะต้องเปิดโปรแกรม LOTUS MEASURE ที่เครื่องไมโครคอมพิวเตอร์ให้พร้อมที่จะรับข้อมูลเสียก่อน

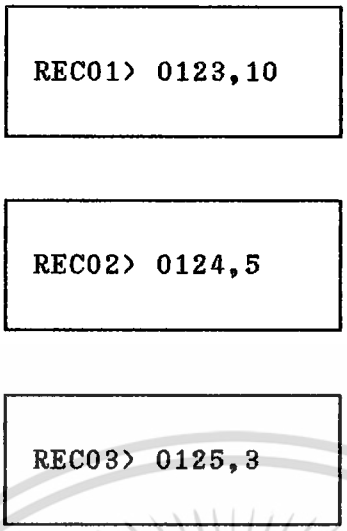
การประยุกต์ใช้งานในโหมด INKEY

สมมุติว่าเราต้องการตรวจสอบ จำนวนสินค้าที่มีทั้งหมดในคลังสินค้าของเรา โดยสินค้าแต่ละชนิดจะมีรหัสประสินค้า

รายการสินค้า และจำนวนมีดังต่อไปนี้

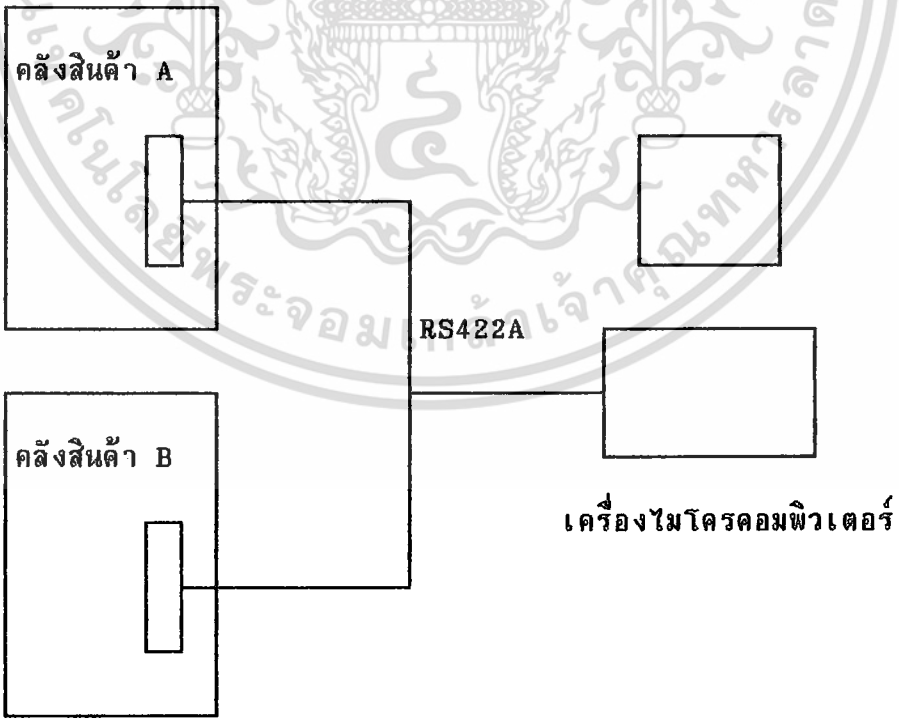
	รหัสสินค้า	จำนวน
1.	0123	10
2.	0124	5
3.	0125	3

เราสามารถป้อนตัวเลขดังกล่าวได้ โดยให้แต่ละเรคคอร์ดมีข้อมูล 2 ชุด ชุดแรกจะเป็นรหัสสินค้า และชุดที่ 2 เป็นจำนวนสินค้า



รูปแสดงการบันทึกข้อมูลในแต่ละเรคคอร์ด

เมื่อส่งค่าที่ได้ไปยังเครื่องไมโครคอมพิวเตอร์ แล้วเราสามารถที่จะแสดงข้อมูลที่ได้ในลักษณะกราฟได้ โดยใช้ LOTUS MEASURE



รูปแสดงการประยุกต์ใช้งานเครื่องไมโครเทอร์มินอล
ในการเก็บข้อมูลจากคลังสินค้า

การใช้งานในโหมด ADC

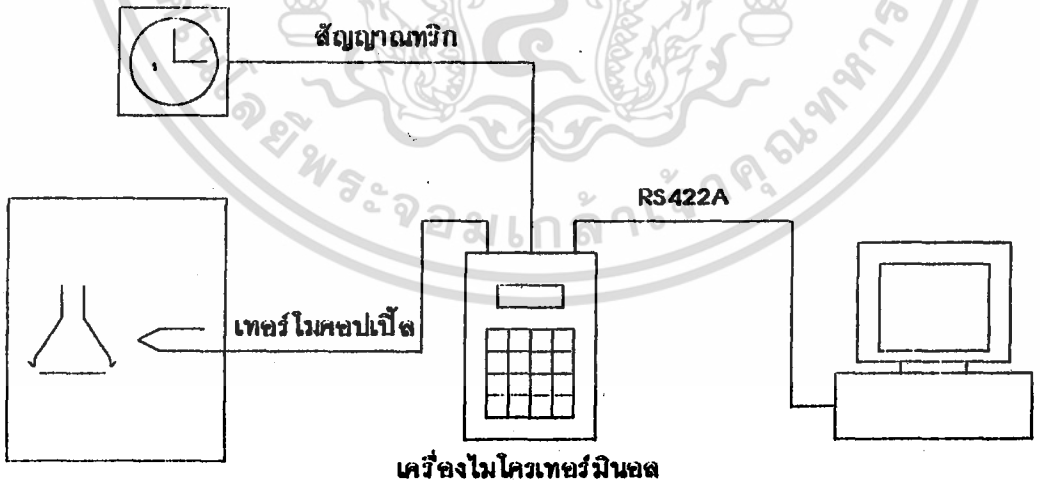
ในโหมดนี้จะสามารถรับสัญญานอนาลอกมีค่า 0000-4000 mV ได้ถึง 99 ค่า ในการรับข้อมูลแต่ละครั้งจะต้องได้รับสัญญานทริกจากภายนอกเสียก่อน

เมื่อเก็บข้อมูลครบแล้ว สามารถส่งข้อมูลไปเก็บที่เครื่องไมโครคอมพิวเตอร์ได้เช่นเดียวกับโหมด INKEY

```
ADC05> 1234,1235
1236,1327,1328,
1328
```

รูปแสดงการเก็บข้อมูลในโหมด ADC

การประยุกต์ใช้งานโหมด ADC



รูปแสดงการประยุกต์ใช้เครื่องไมโครเทอร์มินอลในการบันทึก
การเปลี่ยนแปลงอุณหภูมิของระบบ

ภาคผนวก ค

คู่มือ ANT32

รู้จักกับ ANT-32

ANT-32 คือบอร์ดไมโครคอนโทรลเลอร์ขนาดเล็ก ที่สามารถใช้ทั้ง ภาษาเบสิค และ แอสเซมบลี้ สำหรับการพัฒนาระบบควบคุมอัตโนมัติ และใช้งานในลักษณะเป็น EMBEDDED CONTROLLER โดยใช้ชิพไมโครคอนโทรลเลอร์ เบอร์ 8032 ซึ่งอยู่ในตระกูล MCS-51 ของอินเทล ทำหน้าที่เป็น CPU

ในการพัฒนาโปรแกรม ANT-32 สามารถใช้เครื่องคอมพิวเตอร์ PC ทั่วไป เป็นเครื่องมือช่วยในการพัฒนา ซึ่งจะทำให้งานของท่านสะดวกเป็นอย่างมาก มีความรวดเร็วและง่ายต่อการทดสอบแก้ไข ท่านสามารถมองเห็นโปรแกรม และข้อมูลในหน่วยความจำบนจอภาพ และท่านยังสามารถเขียนโปรแกรมสั่งงานภาษาเบสิค ด้วยคีย์บอร์ดของเครื่องคอมพิวเตอร์ PC ทั้งนี้การพัฒนาโปรแกรมจะกระทำโดยผ่านพอร์ตสื่อสารอนุกรม RS232C โดยที่เครื่องคอมพิวเตอร์ PC จะใช้ COMMUNICATION SOFTWARE ทั่วไปเช่น PROCOM, CROSSTALK จัดการให้เครื่องคอมพิวเตอร์ PC ทำหน้าที่เสมือนเป็น จอภาพ และคีย์บอร์ดของ ANT-32

ANT-32 มีภาษาเบสิค เป็น MONITOR PROGRAM ทำให้ท่านสามารถเขียน แก้ไข หรือ ทดสอบโปรแกรมภาษาเบสิค ซึ่งเป็นภาษาระดับสูง ง่ายต่อการเรียนรู้ และใช้งาน อีกทั้งใน BASIC MONITOR นี้ ท่านยังสามารถทำการ DOWNLOAD และ UPLOAD ข้อมูลหรือโปรแกรมภาษาเบสิค ระหว่าง ANT-32 และเครื่องคอมพิวเตอร์ PC ทั่วไปได้

ANT-32 เกิดขึ้นเพื่อช่วยให้นักพัฒนาไมโครมีความสะดวกยิ่งขึ้น และทำให้งานของท่านเป็นเรื่องที่ไม่ยุ่งยากอีกต่อไป พร้อมที่จะก้าวไปกับความก้าวหน้าทางเทคโนโลยี่อันเป็นผลดีต่อประเทศชาติของเรา ศิลา...หนทางที่ยังต้องก้าวต่อไปอีก เพื่อช่วยให้วงการไมโครเชิงวิศวกรรมในบ้านเราก้าวหน้ามากยิ่งขึ้น

POWER ON RESET

ศิลา...มิถุนายน 2533

แนะนำ MCS BASIC-52

MCS BASIC-52 คือตัวแปลภาษาเบสิคใน 8052AH BASIC มีระบบการจัดการติดต่อกับเทอมินอลในตัว เพื่อสะดวกในการเขียน แก้ไข และทดสอบโปรแกรมภาษาเบสิค โปรแกรมที่เขียน สามารถเก็บลงใน EPROM ที่ต่อภายนอก และนำโปรแกรมที่เก็บไว้ใน EPROM ภายนอกมาทำงานได้ ระบบ REAL TIME CLOCK และการอินเตอร์รัพต์สามารถรับรู้และควบคุมด้วยโปรแกรมเบสิค มีคำสั่งสำหรับการติดต่อกับ รีจิสเตอร์ หน่วยความจำภายใน ภายนอก และพอร์ต ด้วยภาษาเบสิค

MCS BASIC-52 สร้างขึ้นมา สำหรับการเขียน และพัฒนาโปรแกรม เพื่องานควบคุมระบบอัตโนมัติ ที่ใช้ไมโครคอนโทรลเลอร์ สามารถพัฒนาโปรแกรมได้ง่ายขึ้น และใช้เวลาน้อยลง เพราะการพัฒนาโปรแกรมด้วยภาษาแอสเซมบลี้ บางครั้งยาก และน่าเบื่อ เช่น โปรแกรมที่ต้องมีการคำนวณทางคณิตศาสตร์ การสร้างฐานข้อมูลของเครื่องบันทึกข้อมูล การเขียนโปรแกรมเกี่ยวกับการติดต่อกับจอภาพและคีย์บอร์ดของเครื่องคอมพิวเตอร์เพื่อใช้ในการแสดงผล และรับคำสั่ง ให้กับชุดฮาร์ดแวร์ของระบบไมโครคอนโทรลเลอร์ ซึ่งโปรแกรมเหล่านี้จะง่ายขึ้นทันที ถ้าเขียนด้วยภาษาเบสิค

เป็นที่แน่นอนว่า โปรแกรมภาษาแอสเซมบลี้ ย่อมทำงานได้เร็วกว่าโปรแกรมภาษาเบสิค MCS BASIC-52 จึงอนุญาตให้ผู้ใช้ติดต่อกับภาษาแอสเซมบลี้ได้ ทำให้งานบางอย่างที่โปรแกรมเบสิคทำไม่ทัน เช่นการสแกนภาคแสดงผล 7 เซคเม้นต์แบบมัลติเพล็กซ์ ซึ่งต้องใช้ความเร็วในการสแกนสูง ก็สามารถเขียนเป็นโปรแกรมย่อยภาษาแอสเซมบลี้ และเรียกใช้จากโปรแกรมเบสิค ด้วยคำสั่ง CALL

คุณสมบัติพิเศษของ MCS BASIC-52

MCS BASIC-52 เป็นตัวแปลภาษาเบสิค ที่มีคำสั่งและฟังก์ชั่นพื้นฐานคล้ายภาษาเบสิคทั่วไป แต่ MCS BASIC-52 ได้เพิ่มคำสั่งเพื่อความสะดวกในการพัฒนาโปรแกรมควบคุมระบบ เช่น...

- สามารถโปรแกรม EPROM ได้ โดยคำสั่ง "PROG" จะนำโปรแกรมเบสิคที่ใช้อยู่ขณะนั้น บันทึกลงใน EPROM ในลักษณะของ FILE
- สามารถเรียกโปรแกรมเบสิคที่เก็บไว้ใน EPROM ออกมาทำงานได้ โดยคำสั่ง"ROM"
- สามารถย้ายโปรแกรมเบสิคที่อยู่ใน EPROM มาไว้ใน RAM ด้วยคำสั่ง "XFER"
- สามารถเรียกโปรแกรมย่อย ภาษาแอสเซมบลี้ โดยใช้คำสั่ง "CALL"
- โปรแกรมย่อยภาษาแอสเซมบลี้ สามารถเรียกใช้ฟังก์ชั่นภาษาเบสิคได้
- มีคำสั่งควบคุม TIMER/COUNTER ที่อยู่ในตัว ด้วยคำสั่ง "TIMERØ,TIMER1, TIMER2,TCON,TMOD" เป็นต้น
- มีระบบ REAL TIME CLOCK ที่เที่ยงตรง ควบคุมด้วยคำสั่ง "CLOCKØ,CLOCK1, TIMER"
- สามารถรับรู้และควบคุมการอินเตอร์รัพท์ ด้วยคำสั่ง "ONEX1,RETI,CLEARI, IDLE" เป็นต้น
- สามารถอ่านและเขียนค่าพอร์ตในตัวได้ ด้วยคำสั่ง "PORT1"
- สามารถอ่านและเขียนค่าพอร์ตภายนอก (EXTERNAL PORT) ด้วยคำสั่ง "XBY()"
- มีคำสั่งควบคุมพอร์ตอนุกรมในตัว เช่นคำสั่ง "BAUD,RCAP2"
- สามารถติดต่อกับหน่วยความจำภายในและภายนอกด้วยคำสั่ง"CBY(),DBY(),XBY()"
- มีคำสั่งควบคุมการลูป "DO WHILE,DO UNTIL"
- มีฟังก์ชั่นทางคณิตศาสตร์ที่สมบูรณ์

การใช้งาน MCS BASIC-52 บนบอร์ด ANT-32 สามารถใช้งานได้ทุกคำสั่งยกเว้นคำสั่ง PGM,PROG,FPROG ซึ่งเป็นคำสั่งสำหรับการโปรแกรม EPROM สาเหตุที่ใช้คำสั่งเหล่านี้ไม่ได้เนื่องจาก วงจรฮาร์ดแวร์ของ ANT-32 ไม่ได้จัดไว้สำหรับการโปรแกรม EPROM

เพื่อทดแทนคำสั่งเกี่ยวกับการโปรแกรม EPROM เหล่านี้ ANT-32 MON BASIC MONITOR จึงได้เพิ่มคำสั่งเกี่ยวกับการโปรแกรม EEPROM เช่นคำสั่ง EPROG และได้เพิ่มความสามารถในการ UPLOAD และ DOWNLOAD โปรแกรมเบสิคและข้อมูล ระหว่าง ANT-32 กับเครื่องคอมเตอร์ PC โดยรายละเอียดของชุดคำสั่งพิเศษที่เพิ่มเติมนี้ อ่านเพิ่มเติมได้ในหัวข้อ "ANT-32MON BASIC MONITOR" ซึ่งอยู่ในคู่มือเล่มนี้ และรายละเอียดของชุดคำสั่ง MCS BASIC-52 อ่านได้ในหนังสือคู่มือ MCS BASIC-52 USER'S MANUAL

ANT-32MON BASIC MONITOR

ANT-32MON คือโปรแกรมมอนนิเตอร์ สำหรับบอร์ด ANT-32 โดยมีตัวแปลภาษาเบสิค (Basic Interpreter) MCS BASIC-52 รวมอยู่ด้วย และได้เพิ่มชุดคำสั่งพิเศษสำหรับเพิ่มความสะดวกในการพัฒนาโปรแกรม เช่นการ DOWNLOAD ,UPLOAD ระหว่างคอมพิวเตอร์ PC กับ ANT-32 เป็นต้น

ชุดคำสั่งพิเศษ และวิธีใช้ มีรายละเอียดดังนี้

XDOWN [OFFSET]

สำหรับการ DOWN LOAD ข้อมูลในลักษณะ INTEL-HEX FILE ลงใน RAM ทั้งนี้เพื่อให้ผู้ใช้โหลดส่วนที่เป็นข้อมูล หรือส่วนที่แปลมาจาก COMPILER ตัวอื่นๆได้ คำสั่งนี้จะรับข้อมูล และบรรจุลงในหน่วยความจำข้อมูล (DATA MEMORY) ตามค่า ADDRESS ที่อยู่ใน FILE เมื่อใช้คำสั่งนี้ ANT-32 จะรอรับข้อมูลทางสาย RS232 ให้ผู้ใช้ส่งข้อมูลจากเครื่องคอมพิวเตอร์ PC ไปยัง ANT-32 โดยใช้คำสั่ง SE ในโปรแกรม XTALK ได้ทันที ดังตัวอย่างต่อไปนี้

```
>XDOWN                    (กดคีย์ ENTER)
Wait for DOWNLOAD Intel Hex file.......
```

จากนั้นกด Ctrl-A เพื่อเข้า COMMAND MODE ของโปรแกรม XTALK แล้วใช้คำสั่ง SE เพื่อส่งข้อมูลโดยที่ ANT-32 จะรอรับ HEX FILE จากเครื่องคอมพิวเตอร์ PC และเมื่อรับจนจบไฟล์แล้วข้อมูลจะถูกเก็บไว้ใน หน่วยความจำข้อมูล (DATA MEMORY) ตามตำแหน่งที่ระบุใน HEX FILE นั้น

ถ้าผู้ใช้ต้องการให้ข้อมูลเก็บที่แอดเดรสอื่นๆ ที่ไม่ตรงกับค่าแอดเดรสที่ระบุใน HEX FILE นั้น ให้ใช้คำสั่ง XDOWN โดยระบุค่า OFFSET วิธีการคำนวนค่า OFFSET มีดังนี้

OFFSET = แอดเดรสเริ่มต้นที่ต้องการ LOAD-แอดเดรสเริ่มต้นของ HEX FILE

เช่นที่ HEX FILE ระบุแอดเดรสเริ่มต้นเป็น 2000H และผู้ใช้ต้องการ DOWNLOAD ข้อมูลมาเก็บไว้ที่ ANT-32 โดยมีแอดเดรสเริ่มต้นเป็น 8000H จะคำนวนค่า OFFSET ดังนี้

OFFSET = 8000H - 2000H = 6000H

```
> XDOWN 6000        ( กดคีย์ Enter)
Wait for DOWNLOAD Intel Hex file...
```

XUP Start,end

สำหรับการ UPLOAD ข้อมูลในหน่วยความจำข้อมูล (DATA MEMORY) ขึ้นบนเครื่องคอมพิวเตอร์ PC ในลักษณะของ INTEL-HEX FILE โดยผู้ใช้ต้องระบุแอดเดรสเริ่มต้นและแอดเดรสสุดท้ายที่ต้องการ UPLOAD ดังนี้

Start = แอดเดรสเริ่มต้น

end = แอดเดรสสุดท้าย + 1

เช่นถ้าต้องการ ส่งข้อมูลขึ้นบนเครื่องคอมพิวเตอร์ PC โดยแอดเดรสเริ่มต้นที่ต้องการส่งเป็น 1000H และแอดเดรสสุดท้ายที่ต้องการส่งเป็น 103AH

Start = 1000H

end = 103AH + 1 = 103BH

```
>XUP 1000, 103B
Wait 10 Second for UPLOAD Intel Hex file...
```

ANT-32 จะรอเป็นเวลา 10 วินาที ก่อนที่จะส่งข้อมูลขึ้นบนเครื่องคอมพิวเตอร์ PC ขณะที่ ANT-32 รอนั้นให้กด CTRL-A เพื่อเข้า COMMAND MODE ของโปรแกรม XTALK แล้วใช้คำสั่ง CA ON เพื่อให้เครื่องคอมพิวเตอร์ PC เตรียมตัวรับข้อมูลเก็บเข้า BUFFER

เมื่อ ANT-32 รอจนครบ 10 วินาที จะส่งข้อมูลขึ้นบนเครื่องคอมพิวเตอร์ PC ทันที เมื่อส่งเรียบร้อยแล้ว ANT-32 จะหยุดคอยเพื่อให้ผู้ใช้ ใช้คำสั่ง CA OFF เป็นการบอกให้ XTALK จบการรับข้อมูลเพื่อเก็บเข้า BUFFER จากนั้นอยู่ที่ผู้ใช้ว่าต้องการนำข้อมูลที่เก็บไว้ใน BUFFER ของ XTALK บันทึกลงใน DISK หรือไม่ ถ้าบันทึกให้ผู้ใช้ตั้งชื่อไฟล์ โดยมีนามสกุลเป็น .HEX เช่น TEST.HEX เป็นต้น

BUP [1-6]

สำหรับการ UPLOAD โปรแกรมภาษาเบสิค ที่อยู่ในหน่วยความจำขึ้นบนเครื่องคอมพิวเตอร์ PC ในลักษณะของ INTEL HEX FILE โดยมีหลักการเช่นเดียวกับคำสั่ง XUP เพียงแต่จะทำการคำนวนแอดเดรสเริ่มต้น และสุดท้ายให้โดยอัตโนมัติ แต่ถ้าขณะที่ใช้คำสั่ง BUP โดยไม่มีโปรแกรมภาษาเบสิค อยู่ในหน่วยความจำ คำสั่งนี้จะถูกยกเลิกโดยอัตโนมัติเช่นกัน

```
>BUP                          ( กดคีย์ Enter )
Wait 10 second for UPLOAD Intel Hex file...
```

ANT-32 จะรอเป็นเวลา 10 วินาที เช่นเดียวกับคำสั่ง XUP จากนั้นให้ผู้ใช้ดำเนินการตามวิธีเดียวกันกับการใช้คำสั่ง XUP แตกต่างกันที่การตั้งนามสกุลไฟล์ ควรเป็น .BHX เช่น TEST.BHX เป็นต้น

โปรแกรมเบสิคที่ถูก UPLOAD ขึ้นบนเครื่องคอมพิวเตอร์ PC ด้วยคำสั่ง BUP นี้สามารถนำกลับมาใช้งานบน ANT-32 โดยใช้คำสั่ง XDOWN โดยไม่ต้องระบุค่า OFFSET

สำหรับผู้ใช้ที่ต้องการนำโปรแกรมเบสิค ที่อยู่ในหน่วยความจำ นำไปโปรแกรมลงบน EPROM เพื่อการเก็บข้อมูลแบบถาวร และทำการ AUTO START มีวิธีการดังต่อไปนี้

1.UPLOAD โปรแกรมเบสิค ขึ้นบนเครื่องคอมพิวเตอร์ PC ก่อน โดยใช้คำสั่ง BUP และระบุหมายเลข 1 ถึง 6 ด้วย ซึ่งคำสั่ง BUP1-6 จะกำหนดแอดเดรสเริ่มต้น ของ INTEL HEX FILE เป็น 0000H เสมอ เพื่อสะดวกในการที่ผู้ใช้จะนำ HEX FILE นี้ไปโปรแกรมลงบน EPROM โดย BUP1-6 จะมีหน้าที่แตกต่างกันดังนี้

BUP1 = UPLOAD FILE + PROG1 COMMAND

BUP2 = UPLOAD FILE + PROG2 COMMAND

BUP3 = UPLOAD FILE + PROG3 COMMAND

BUP4 = UPLOAD FILE + PROG4 COMMAND

BUP5 = UPLOAD FILE + PROG5 COMMAND

BUP6 = UPLOAD FILE + PROG6 COMMAND

โดยที่ PROG1-PROG6 เป็นคำสั่งของ BASIC เพื่อทำการ AUTO START โปรแกรมเบสิค ดูรายละเอียดของคำสั่งนี้ได้ในคู่มือ BASIC-52 VSER'S MANVAL ใน CHAPTER11

2. นำ INTEL HEX FILE ที่ได้จากการ UPLOAD โดยใช้คำสั่ง BUP 1-6 ไปแปลเป็นอ๊อบเจ็คไฟล์ (.OBJ) แล้วนำไปโปรแกรมลงบน EPROM โดยเครื่องโปรแกรม EPROM ทั่วไป

3.นำ EPROM มาใส่ที่ตำแหน่ง U4 ของ ANT-32 แล้วทำการ RESET ถ้าทุกอย่างเรียบร้อย ANT-32 จะ AUTO START ตามคำสั่ง PROG1-6 ที่เลือกทันที่

ตัวอย่างการใช้งาน

```
>BUP2                    ( กดคีย์ Enter )
Basic upload with PROG 2 command.
Wait 10 second for UPLOAD intel Hex file...
```

DUMP [Start]

สำหรับการดูข้อมูลในหน่วยความจำข้อมูล (DATA MEMORY) โดยมีแอดเดรสที่เรียกดูได้ระหว่าง 0000H - FFFFH

>DUMP 1000

จะเป็นการเรียกดูข้อมูลในหน่วยความจำข้อมูล ตั้งแต่แอดเดรส 1000H เป็นต้นไป

EPROG

สำหรับการนำไฟล์โปรแกรมภาษาเบสิคที่อยู่ในหน่วยความจำ โปรแกรมลงบน EEPROM เบอร์ 2864 โดยมีการทำงานเสมือนคำสั่ง PROG ของ BASIC - 52

EPROG1

โปรแกรม EEPROM #2864 ด้วยข้อมูลเสมือนคำสั่ง PROG 1 ของ BASIC-52

EPROG2

โปรแกรม EEPROM #2864 ด้วยข้อมูลเสมือนคำสั่ง PROG 2 ของ BASIC-52

EPROG3

โปรแกรม EEPROM #2864 ด้วยข้อมูลเสมือนคำสั่ง PROG 3 ของ BASIC-52

EPROG4

โปรแกรม EEPROM #2864 ด้วยข้อมูลเสมือนคำสั่ง PROG 4 ของ BASIC-52

EPROG5

โปรแกรม EEPROM #2864 ด้วยข้อมูลเสมือนคำสั่ง PROG 5 ของ BASIC-52

EPROG6

โปรแกรม EEPROM #2864 ด้วยข้อมูลเสมือนคำสั่ง PROG 6 ของ BASIC-52

EBLANK

สำหรับล้างข้อมูล EEPROM #2864

DIR

สำหรับการเรียกดูชื่อและจำนวนไฟล์ที่อยู่ใน EPROM

```
>DIR
< ROM NO.>     < *** FILE NAME *** >
01             USERPORT
02             SILA
03             MM58167
04             - NO NAME -
05             TESTPORT
>READY
```

ชื่อไฟล์ที่คำสั่ง DIR อ่านได้นั้น ทำได้โดยการ ระบุชื่อไฟล์ไว้ที่บรรทัดแรกของโปรแกรม ด้วยคำสั่ง REM เช่น

```
10 REM SILA
20 FOR I = 1 TO 10
30 PRINT "SILA RESEARCH CO.,LTD."
40 NEXT I
50 END
```

ในที่นี้ บรรทัดที่ 10 จะเป็นการตั้งชื่อไฟล์ เมื่อนำไฟล์นี้ไปโปรแกรมลงบน EPROM หรือ EEPROM แล้วนำมาใส่ที่ซ็อคเก็ต U4 จะสามารถใช้คำสั่ง DIR เรียกดูชื่อไฟล์ได้

สรุปคำสั่งพิเศษ

```
XDOWN [OFFSET]     - Download Intel Hex file
XUP   START,END    - Upload   Intel Hex file
BUP                - Upload   current basic file
BUP1               - Upload   current basic file with PROG1
                     and prepare file for use to program EPROM
BUP2               - Upload   current basic file with PROG2
                     and prepare file for use to program EPROM
BUP3               - Upload   current basic file with PROG3
                     and prepare file for use to program EPROM
BUP4               - Upload   current basic file with PROG4
                     and prepare file for use to program EPROM
BUP5               - Upload   current basic file with PROG5
                     and prepare file for use to program EPROM
BUP6               - Upload   current basic file with PROG6
                     and prepare file for use to program EPROM
DUMP  [START]      - Display data memory
EPROG              - Program EEPROM #2864 with current basic file
EPROG1             - Program EEPROM #2864 with PROG1 command
EPROG2             - Program EEPROM #2864 with PROG2 command
EPROG3             - Program EEPROM #2864 with PROG3 command
EPROG4             - Program EEPROM #2864 with PROG4 command
EPROG5             - Program EEPROM #2864 with PROG5 command
EPROG6             - Program EEPROM #2864 with PROG6 command
EBLANK             - Erase    EEPROM #2864
DIR                - Directory files in EPROM at U4 socket
```

การใช้งานโปรแกรม XTALK

โปรแกรม XTALK คือโปรแกรมสำหรับการสื่อสารทั่วไป โดยในที่นี้จะใช้สื่อสารในแบบ LOCAL ซึ่งเป็นการสื่อสารโดยผ่านสาย RS232 โปรแกรมนี้มีรายละเอียดค่อนข้างมาก แต่ในที่นี้จะสรุปการใช้งานพอสังเขป และจะเน้นที่การใช้งานกับ NIC-52 เป็นแนวทางหลัก โดยพอจะสรุปเป็นข้อ ๆ ได้ดังนี้

1. โปรแกรมในแผ่น DISK ที่ให้มาจะ AUTOEXEC ให้เรียบร้อยแล้ว โดยเป็นการใช้งานตามการกำหนดคือ BAUD RATE = 9600, STOP BIT = 1, PARITY = NONE DATA = 8, PORT = COM1, CWAIT=DELAY 2, LOCAL COMMUNICATION

2. การใช้งานมีโหมดต่าง ๆ คือ
   - โหมดการสื่อสาร โดยจะอยู่ในขบวนการรับและส่งข้อมูลทางสาย และที่บรรทัดล่างสุดจะแสดงสถานะบางอย่าง พร้อมทั้งแสดงคำสั่ง(^A)ในการเปลี่ยนโหมดด้วย
   - โหมด COMMAND จะเป็นการรับคำสั่งต่าง ๆ ของ XTALK ในจุดนี้ผู้ใช้จะขอดูตัวแปรต่าง ๆ ของการสื่อสารได้ด้วยคำสั่ง ^F คำสั่งทั้งหมดของ XTALK จะดูได้ด้วยคำสั่ง HE (HELP) ในโหมดนี้สิ่งต่าง ๆ ที่ปรากฏบนจอจะเกิดขึ้นจาก XTALK เอง ไม่ใช่สิ่งที่มาจากการสื่อสาร ผู้ใช้จะกลับไปโหมดการสื่อสารได้ด้วยการกด ENTER หรือคำสั่ง GO LOCAL

3. การใช้คำสั่ง CA (CATURE) เพื่อการเก็บสิ่งต่าง ๆ ที่ปรากฏบนจอลงใน FILE จะทำได้ โดยใช้ CA และตามด้วยชื่อ FILE หลังจากนั้นสิ่งที่เกิดขึ้นบนจอทุกอย่าง (ที่มาจากการสื่อสาร) จะถูกเก็บลงใน FILE และผู้ใช้จะสิ้นสุดขบวนการได้ด้วยคำสั่ง CA OFF กรณีนี้จะใช้สำหรับการเก็บโปรแกรมภาษา BASIC-52 ในแบบ TEXT FILE ได้ หรือใช้ร่วมกับคำสั่งของ NIC-52MON เพื่อการ UP LOAD ข้อมูลต่าง ๆ จาก NIC-52 ขึ้นเครื่อง PC

4. คำสั่ง SE (SEND) ใช้สำหรับการส่งข้อมูลจาก FILE ออกทางสาย ซึ่งเป็นขบวนการ DOWN LOAD จากเครื่อง PC ไปยัง NIC-52 โดยใช้งานร่วมกับคำสั่งใน NIC-52MON การใช้คำสั่ง SE และ CA จะช่วยให้การใช้งานของผู้ใช้สะดวกมากยิ่งขึ้น ซึ่งสามารถจะทำการ UP และ DOWN LOAD ข้อมูลไปมา ทำให้การเก็บโปรแกรมหรือข้อมูลต่าง ๆ กระทำอยู่บนเครื่อง PC ได้

ANT-32 กับภาษาเบสิค

การพัฒนาโปรแกรมด้วยภาษาเบสิค มีขั้นต้อนต่างๆ สรุปได้ดังนี้

1. ต่อสาย RS232 ระหว่างเครื่องคอมพิวเตอร์ PC กับ ANT-32 ให้เรียบร้อย

2. เรียกใช้โปรแกรม ANT32.BAT ซึ่งอยู่ในแผ่นดิสต์ " ANT-32 UTILITY "

```
A>ANT32                    ( กดคีย์ Enter )
```

เครื่องจะเข้าโปรแกรม CROSSTALK และอยู่ในสภาวะที่พร้อมจะทำการสื่อสาร โดยเครื่องคอมพิวเตอร์ PC จะทำหน้าที่เสมือนเป็นจอภาพ และคีย์บอร์ด ของ ANT-32

3. เปิดเครื่อง ANT-32 และกด SPACE ที่เครื่องคอมพิวเตอร์ PC , ANT-32 จะทำการคำนวน ค่าอัตราการรับส่งข้อมูล ให้โดยอัตโนมัติ และถ้าระบบเรียบร้อยดี จะมีข้อความต่อไปนี้ ปรากฏบนจอภาพ

```
* MCS-51 (tm) BASIC V1.1 *
READY
>
```

4. จากนั้นท่านสามารถเขียนคำสั่ง หรือโปรแกรมภาษาเบสิค เพื่อสั่งงาน ANT-32

```
>10 FOR I=1 TO 5           ( กดคีย์ Enter )
>20 PRINT I                ( กดคีย์ Enter )
>30 NEXT I                 ( กดคีย์ Enter )
```

บริษัท ดิลาริเสิร์ช จำกัด

ทดลองใช้คำสั่ง LIST เพื่อเรียกดูโปรแกรม

```
>LIST                                    ( กดคีย์ Enter )
10  FOR I = 1 TO 5
20  PRINT I
30  NEXT I
READY
>
```

ทดลองใช้คำสั่ง RUN เพื่อให้โปรแกรมที่เขียนไว้ข้างต้นทำงาน

```
>RUN                                    ( กดคีย์ Enter )
1
2
3
4
5
READY
>
```

5. ท่านสามารถเก็บโปรแกรมภาษาเบสิคไว้ในดิสค์ของเครื่องคอมพิวเตอร์ PC ได้ โดยใช้คำสั่ง BUP และสามารถโหลดโปรแกรมเบสิคคืนมาได้ โดยใช้คำสั่ง XDOWN รายละเอียดการใช้งานสองคำสั่งนี้ อ่านได้ในหัวข้อ ANT-32MON BASIC MONITOR

6. เมื่อพัฒนาโปรแกรมเรียบร้อยแล้ว ANT-32 สามารถทำการ RUN โปรแกรมที่อยู่ใน ROM (U4 socket) ตำแหน่งไฟล์ ROM1 แบบ AUTO START โดยไม่ต้องอาศัยการติดต่อกับเครื่องคอมพิวเตอร์ PC ด้วยการใช้คำสั่ง BUP1-6 หรือ EPROG1-6 รายละเอียดการใช้งานสองคำสั่งนี้ อ่านได้ในหัวข้อ ANT-32MON BASIC MONITOR

ANT-32 กับ ภาษาเบสิค + แอสเซมบลี้

การพัฒนาโปรแกรมด้วยภาษาเบสิค ร่วมกับ ภาษาแอสเซมบลี้ สรุปหลักการได้ดังนี้

1. ให้โปรแกรมเบสิคเป็นโปรแกรมหลัก และโปรแกรมแอสเซมบลี้ให้เป็นโปรแกรมย่อย โดยโปรแกรมเบสิคสามารถเรียกให้โปรแกรมแอสเซมบลี้ทำงาน ด้วยคำสั่ง CALL ของภาษาเบสิค เช่น CALL 9000H โปรแกรมเบสิคจะเรียกโปรแกรมแอสเซมบลี้ที่ตำแหน่งแอดเดรส 9000H ให้ทำงาน

2. การเขียนโปรแกรมแอสเซมบลี้ให้ใช้โปรแกรมเอดิเตอร์ทั่วไป หรือใช้โปรแกรม ED.COM ที่มีอยู่ในดิสค์ ANT-32 UTILITY และจุดเริ่มต้นของโปรแกรม (origin) ควรอยู่ที่แอดเดรส 9000H ขึ้นไป

3. โปรแกรมแอสเซมบลี้ที่เขียนเสร็จแล้ว นำมาแปลเป็นอ็อบเจ็คโค้ด โดยใช้โปรแกรมตัวแปลภาษาแอสเซมบลี้ 8051 ทั่วไป หรือใช้โปรแกรม SXA51.EXE ที่มีอยู่ในดิสค์ ANT-32 UTILITY โดยไฟล์ที่แปลได้จะอยู่ในลักษณะของ INTEL HEX FILE

4. ทำการ DOWN LOAD ไฟล์โปรแกรมแอสเซมบลี้ที่แปลแล้ว เข้ามาในหน่วยความจำข้อมูลของ ANT-32 ด้วยคำสั่ง XDOWN และโปรแกรมแอสเซมบลี้นี้จะทำงาน เมื่อมีการใช้คำสั่ง CALL ของโปรแกรมภาษาเบสิค

5. การเขียนโปรแกรมแอสเซมบลี้ เพื่อใช้ร่วมกับโปรแกรมภาษาเบสิค มีข้อควรรู้และควรระวัง ดังนี้

- เบสิค ใช้รีจิสเตอร์แบงค์ 0,1,2 ผู้ใช้ห้ามเปลี่ยนแปลงค่าในรีจิสเตอร์เหล่านี้ นอกจากในขณะที่เรียกใช้ฟังก์ชั่นเบสิค จะใช้รีจิสเตอร์แบงค์ 0 เป็นตัวส่งผ่านค่าได้

- รีจิสเตอร์แบงค์ 3 ผู้ใช้สามารถใช้ได้ เบสิคจะใช้รีจิสเตอร์แบงค์ 3 นี้ขณะที่ทำงานตามคำสั่ง PGM เท่านั้น

- Acc ,DPTR ผู้ใช้สามารถใช้ได้ แต่ต้องนึกเสมอว่า โปรแกรมเบสิคสามารถเปลี่ยนแปลงค่าได้

6. ผู้ใช้สามารถส่งผ่านค่าตัวแปร ระหว่างโปรแกรมเบสิคกับแอสเซมบลี้ โดยผ่านทาง ARGUMENT STACK

7. โปรแกรมแอสเซมบลี้สามารถเรียกใช้ฟังก์ชั่นเบสิคได้ โดยมีหลักการดังนี้

- ฟังก์ชั่นเบสิค จะมีหมายเลขประจำฟังก์ชั่นนั้นๆ เรียกว่า OPBYTE

- เมื่อต้องการเรียกใช้ฟังก์ชั่นเบสิค ให้เลือกใช้รีจิสเตอร์แบงค์ 0 ก่อน จากนั้นให้ค่า OPBYTE ของฟังก์ชั่นเบสิคที่ต้องการไว้ที่ Acc แล้ว CALL 30H

- ผลลัพธ์จากการเรียกใช้ ฟังก์ชั่นเบสิคจะเก็บใน รีจิสเตอร์แบงค์ 0 เสมอ และในบางฟังก์ชั่น จะเก็บผลลัพธ์ไว้ที่ ARGUMENT STACK

8. เพื่อความเข้าใจยิ่งขึ้น สามารถอ่านเพิ่มเติมได้ใน คู่มือ BASIC-52 USER'S MANUAL ใน CHAPTER 9 (ASSEMBLY LANGUAGE LINKAGE)

ANT-32 กับ ภาษาแอสเซมบลี้

การพัฒนาโปรแกรมด้วยภาษาแอสเซมบลี้ มีหลักการดังนี้

1. การเขียนโปรแกรมแอสเซมบลี้ให้ใช้โปรแกรมเอดิเตอร์ทั่วไป หรือใช้โปรแกรม ED.COM ที่มีอยู่ในดิสค์ ANT-32 UTILITY และจุดเริ่มต้นของโปรแกรม (origin) ต้องอยู่ที่แอดเดรส 0000H

2. โปรแกรมแอสเซมบลี้ที่เขียนเสร็จแล้ว นำมาแปลเป็นอ็อบเจ็คโค็ด โดยใช้โปรแกรมตัวแปลภาษาแอสเซมบลี้ 8051 ทั่วไป หรือใช้โปรแกรม SXA51.EXE ที่มีอยู่ในดิสค์ ANT-32 UTILITY โดยไฟล์ที่แปลได้จะอยู่ในลักษณะของ INTEL HEX FILE

3. ไฟล์โปรแกรมที่แปลเรียบร้อยแล้ว สามารถส่งไปยังบอร์ด ANT-32 ได้ โดยใช้บอร์ด EE-232 (EPROM EMULATOR) โดยซ็อคเก็ตของ EE-232 เสียบอยู่ที่บอร์ด ANT-32 แทนที่ ROM ANT-32MON แล้วทำการส่งไฟล์ จากเครื่องคอมพิวเตอร์ PC ไปที่ EE-232 โดยผ่านทางพอร์ต RS232

4. สำหรับท่านที่ไม่มี EE-232 (EPROM EMULATOR) ให้นำโปรแกรมที่แปลได้ ซึ่งอยู่ในลักษณะของ INTEL HEX นำไปแปลเป็นอ็อบเจ็คไฟล์ก่อน แล้วนำไปโปรแกรมลงบน EPROM โดยเครื่องโปรแกรม EPROM ทั่วไป จากนั้นนำ EPROM ที่โปรแกรมเรียบร้อย ใส่ลงบนบอร์ด ANT-32 แทนที่ ROM ANT-32 MON (socket U2)

5. กดสวิทช์รีเซ็ทบนบอร์ด ANT-32 ถ้าโปรแกรมแอสเซมบลี้ ที่ท่านเขียนถูกต้อง ANT-32 จะทำงานตามที่ท่านโปรแกรมไว้ทันที

นอกจากวิธีที่กล่าวมาแล้วนั้น ท่านยังสามารถใช้ ANT-32MON BASIC MONITOR ช่วยในการพัฒนา โดยมีหลักการดังนี้

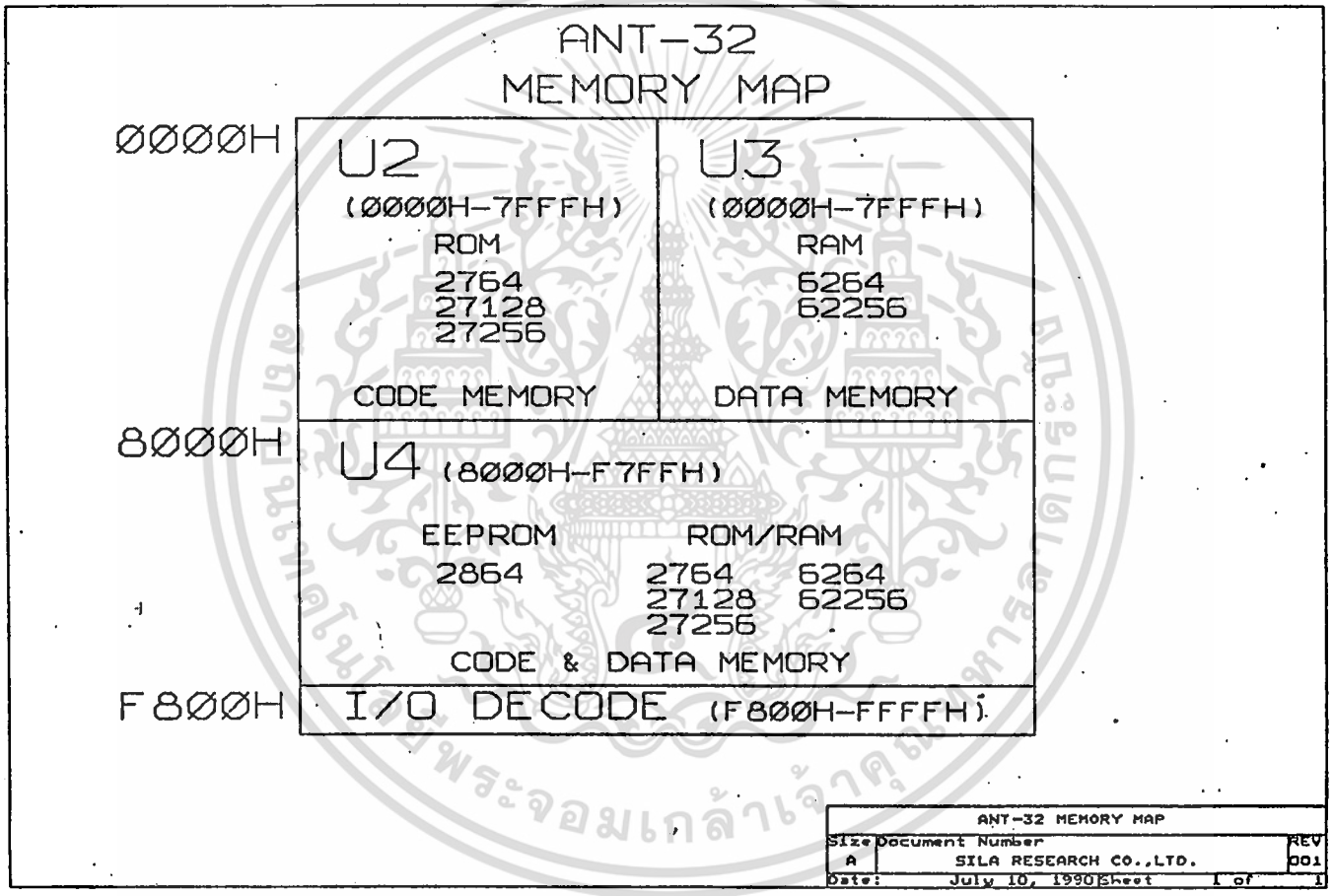
1. โปรแกรมแอสเซมบลี้ที่เขียนให้มีจุดเริ่มต้น (origin) ตั้งแต่แอดเดรส 8000H ขึ้นไป

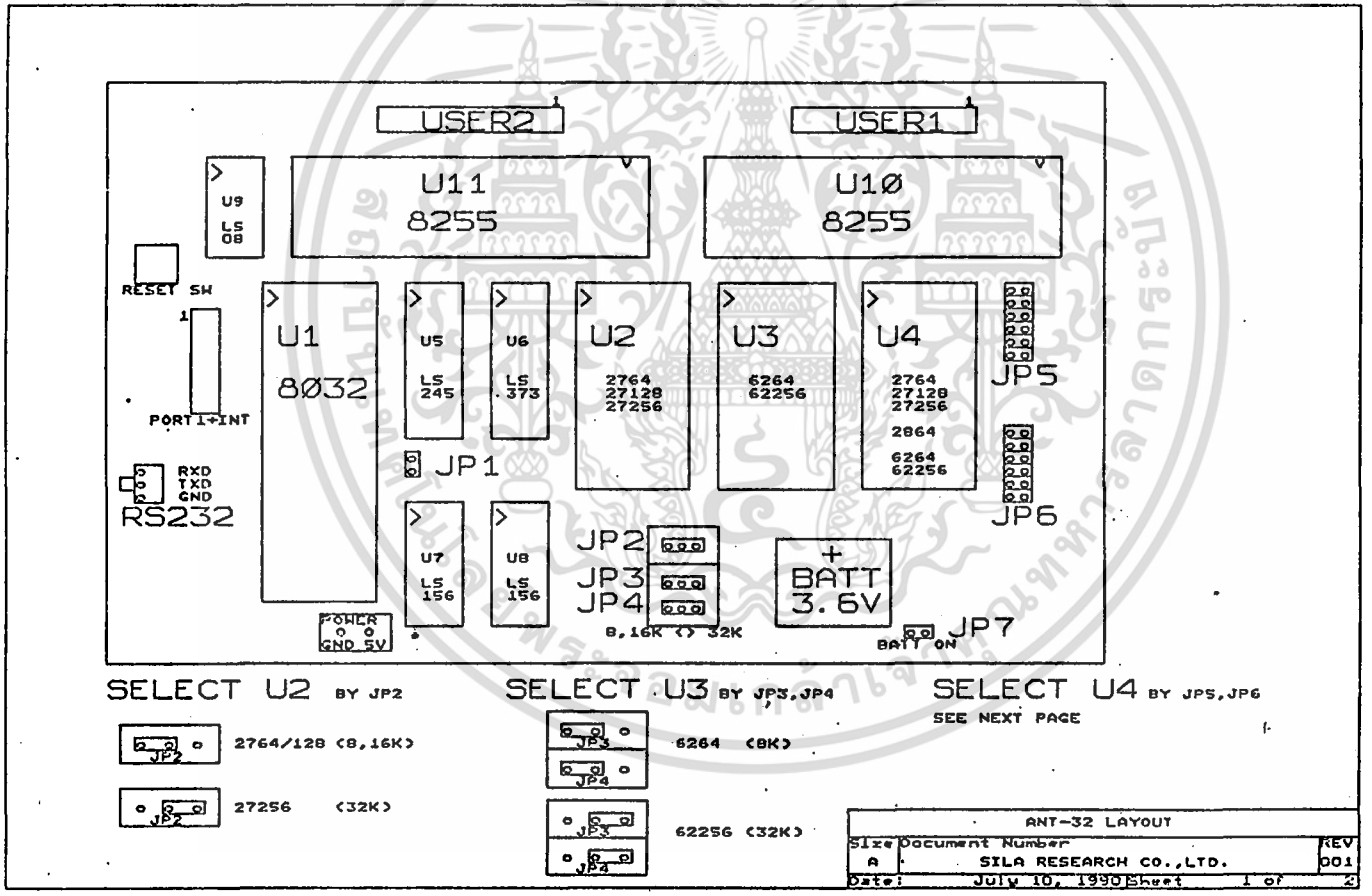
2. ใช้คำสั่ง XDOWN ทำการ DOWN LOAD ไฟล์โปรแกรมแอสเซมบลี้ที่แปลแล้วที่อยู่ในลักษณะ INTEL HEX FILE ลงมาที่หน่วยความจำของ ANT-32

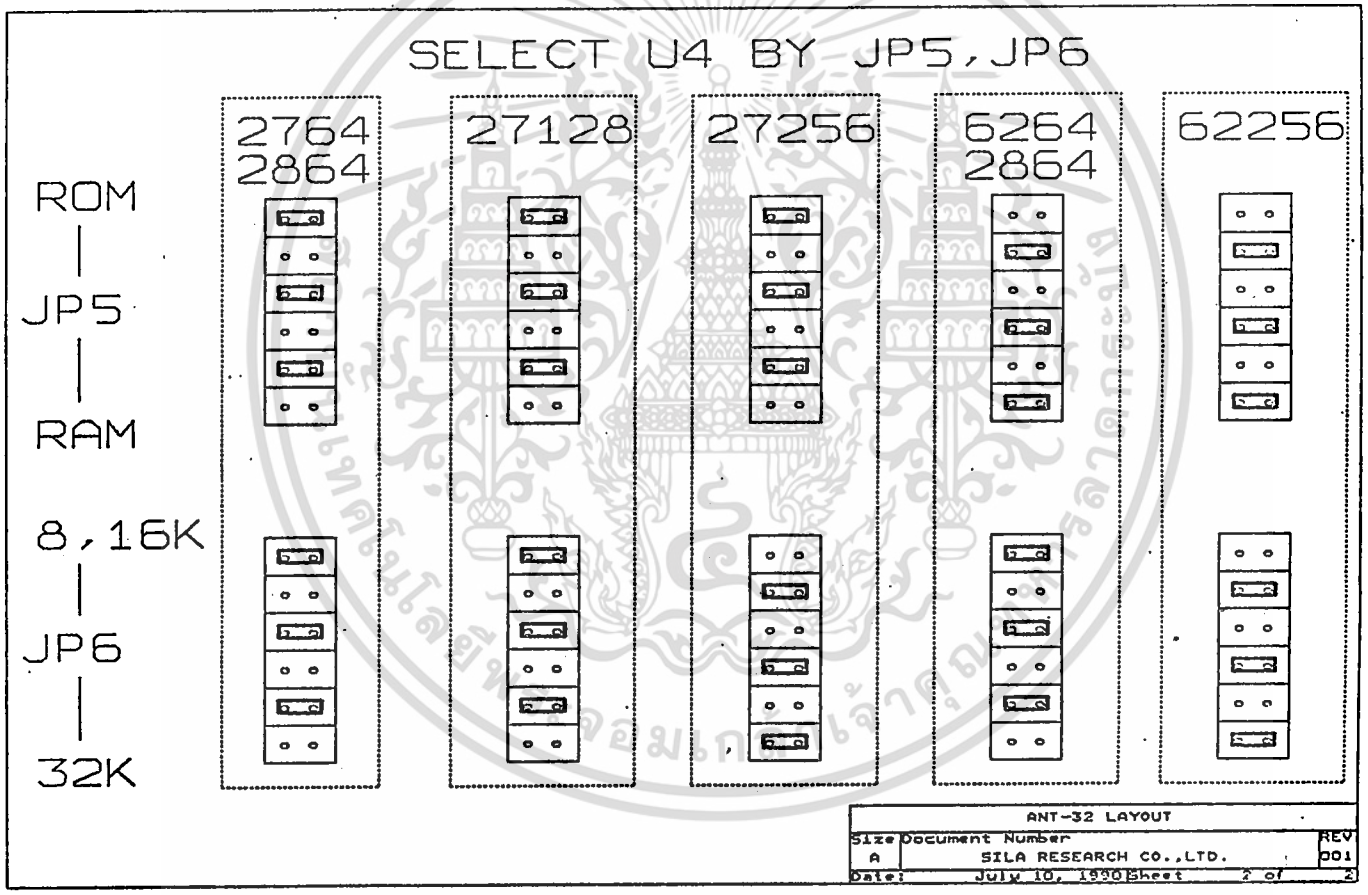
3. เมื่อต้องการทดสอบโปรแกรม ให้ใช้คำสั่ง CALL ของเบสิค เรียกโปรแกรมแอสเซมบลี้ให้ทำงาน

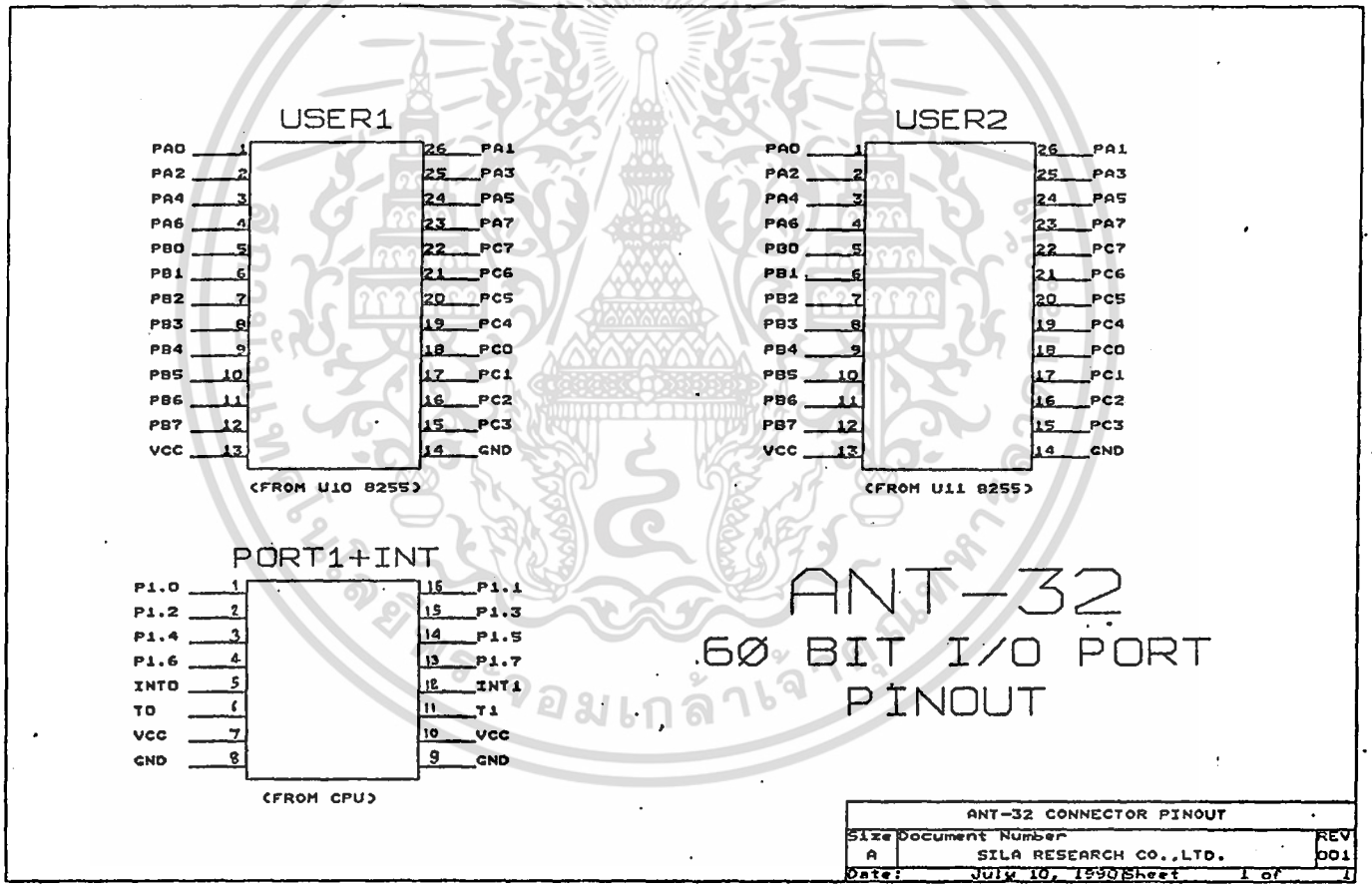
ANT-32 SPECIFICATION

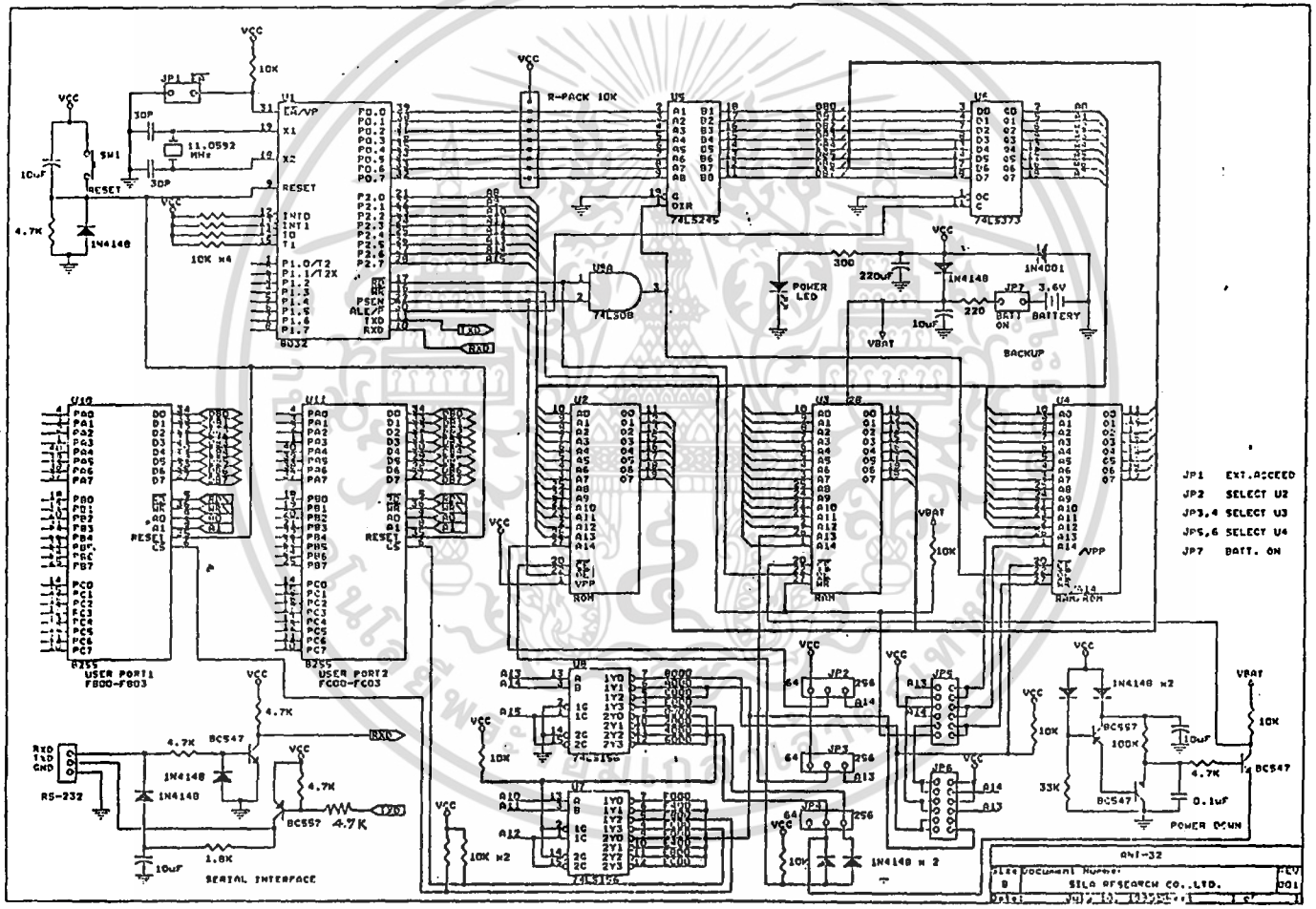
CPU:	8032
CPU CLOCK:	11.0592 MHz
INTERNAL MEMORY:	256 BYTE (RAM)
EXT.CODE MEMORY:	(U2) 8-32K SELECT 2764,27128,27256 (ROM)
EXT.DATA MEMORY:	(U3) 8-32K SELECT 6264,62256 (RAM)
EXT.CODE & DATA MEMORY:	(U4) 8-30K SELECT 2764,27128,27256 (ROM)
	2864 (EEPROM)
	6264,62256 (RAM)
INTERNAL PORT:	12 BIT I/O
EXTERNAL PORT:	USER1 8255 PORT I/O 24 BIT
	USER2 8255 PORT I/O 24 BIT
BACKUP:	DATA MEMORY (U3) 52 HOUR
CHARGE TIME:	48 HOUR
LANGUAGE:	MCS BASIC-52
	ASSEMBLY (BY DOWNLOAD HEX FILE)
CONNECTOR:	16P INTERNAL PORT
	26P USER1 PORT
	26P USER2 PORT
	2P POWER SUPPLY
	3P SERIAL INTERFACE (RS232)
SERIAL INTERFACE:	RS232C
POWER:	5V DC 290mA (U4 NOT INCLUDE)
SIZE:	5.25" x 3.9"
OPTION:	BATTERY NI-CAD 3.6V 60mA


ANT-32
MEMORY MAP
ØØØØH
U2
(ØØØØH-7FFFH)
ROM
2764
27128
27256
CODE MEMORY
U3
(ØØØØH-7FFFH)
RAM
6264
62256
DATA MEMORY
8ØØØH
U4 (8ØØØH-F7FFH)
EEPROM
2864
ROM/RAM
2764
27128
27256
6264
62256
CODE & DATA MEMORY
F8ØØH
I/O DECODE (F8ØØH-FFFFH)
ANT-32 MEMORY MAP
Size
A
Document Number
SILA RESEARCH CO.,LTD.
REV
001
Date: July 10, 1990 Sheet 1 of 1


USER2
USER1
U9
LS
08
U11
8255
U10
8255
RESET SW
U1
8032
U5
LS
245
U6
LS
373
U2
2764
27128
27256
U3
6264
62256
U4
2764
27128
27256
2864
6264
62256
JP5
JP6
PORT1+INT
JP1
RXD
TXD
GND
RS232
U7
LS
156
U8
LS
156
JP2
JP3
JP4
8,16K <> 32K
+
BATT
3.6V
JP7
BATT ON
POWER
GND 5V
SELECT U2 BY JP2
2764/128 (8,16K)
27256 (32K)
SELECT U3 BY JP3,JP4
6264 (8K)
62256 (32K)
SELECT U4 BY JP5,JP6
SEE NEXT PAGE
ANT-32 LAYOUT
Size A
Document Number
SILA RESEARCH CO.,LTD.
REV 001
Date: July 10, 1990 Sheet 1 of 2


SELECT U4 BY JP5,JP6
2764
2864
27128
27256
6264
2864
62256
ROM
JP5
RAM
8,16K
JP6
32K
ANT-32 LAYOUT
Size
A
Document Number
SILA RESEARCH CO.,LTD.
REV
001
Date:
July 10, 1990
Sheet
2 of
2


USER1
PA0 1
PA2 2
PA4 3
PA6 4
PB0 5
PB1 6
PB2 7
PB3 8
PB4 9
PB5 10
PB6 11
PB7 12
VCC 13
26 PA1
25 PA3
24 PA5
23 PA7
22 PC7
21 PC6
20 PC5
19 PC4
18 PC0
17 PC1
16 PC2
15 PC3
14 GND
(FROM U10 8255)
USER2
PA0 1
PA2 2
PA4 3
PA6 4
PB0 5
PB1 6
PB2 7
PB3 8
PB4 9
PB5 10
PB6 11
PB7 12
VCC 13
26 PA1
25 PA3
24 PA5
23 PA7
22 PC7
21 PC6
20 PC5
19 PC4
18 PC0
17 PC1
16 PC2
15 PC3
14 GND
(FROM U11 8255)
PORT1+INT
P1.0 1
P1.2 2
P1.4 3
P1.6 4
INT0 5
T0 6
VCC 7
GND 8
16 P1.1
15 P1.3
14 P1.5
13 P1.7
12 INT1
11 T1
10 VCC
9 GND
(FROM CPU)
ANT-32
60 BIT I/O PORT
PINOUT
ANT-32 CONNECTOR PINOUT
Size A
Document Number
SILA RESEARCH CO.,LTD.
REV 001
Date: July 10, 1990 Sheet 1 of 1


JP1 EXT.ACCEED
JP2 SELECT U2
JP3,4 SELECT U3
JP5,6 SELECT U4
JP7 BATT. ON
8051
74LS245
74LS373
74LS08
POWER LED
BACKUP
BATTERY
USER PORT1
F800-F803
USER PORT2
FC00-FC03
SERIAL INTERFACE
RS-232
POWER DOWN
ANT-32
SILA RESEARCH CO.,LTD.

MCS®-51 INSTRUCTION SET

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag C	OV	AC	Instruction	Flag C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn — Register R7–R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].

@Ri — 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)				
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Mnemonic		Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

All mnemonics copyrighted ©Intel Corporation 1980

Table 11. Instruction Opcodes in Hexadecimal Order

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A, # data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr, code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A, # data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr, # data
44	2	ORL	A, # data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr, # data
54	2	ANL	A, # data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr, # data
64	2	XRL	A, # data
65	2	XRL	A,data addr

Table 11. Instruction Opcodes in Hexadecimal Order (Continued)

Hex Code	Number of Bytes	Mnemonic	Operands
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A+PC
84	1	DIV	AB
85	3	MOV	data addr, data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A+DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3

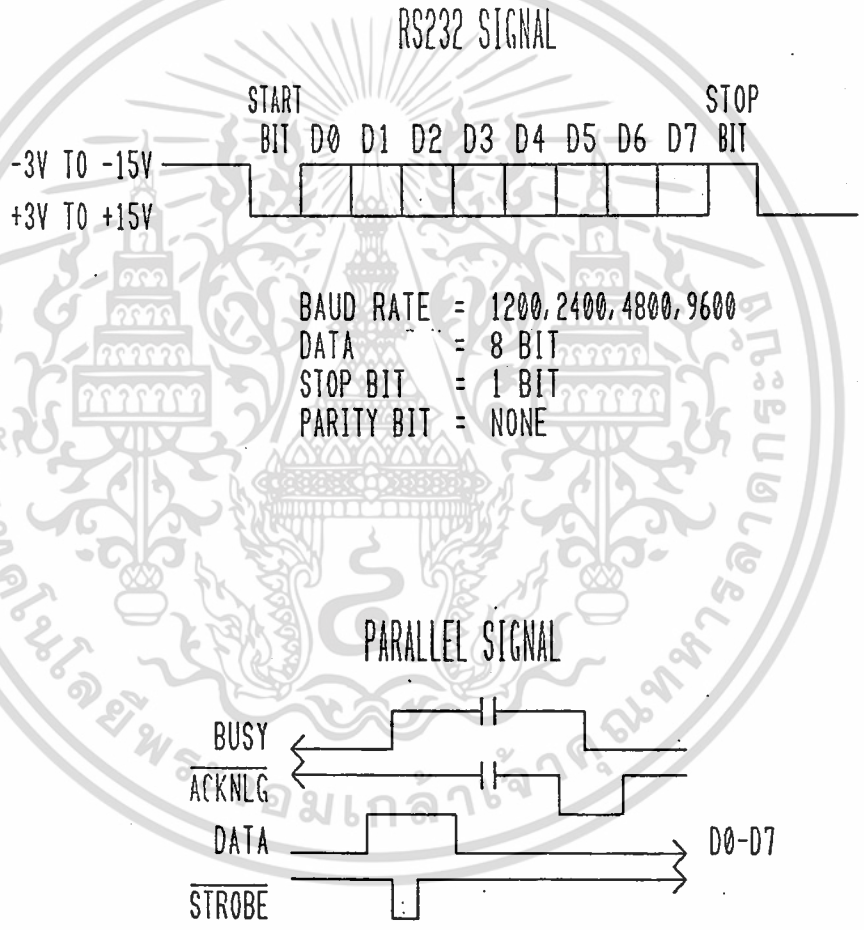
Table 11. Instruction Opcodes in Hexadecimal Order (Continued)

Hex Code	Number of Bytes	Mnemonic	Operands
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

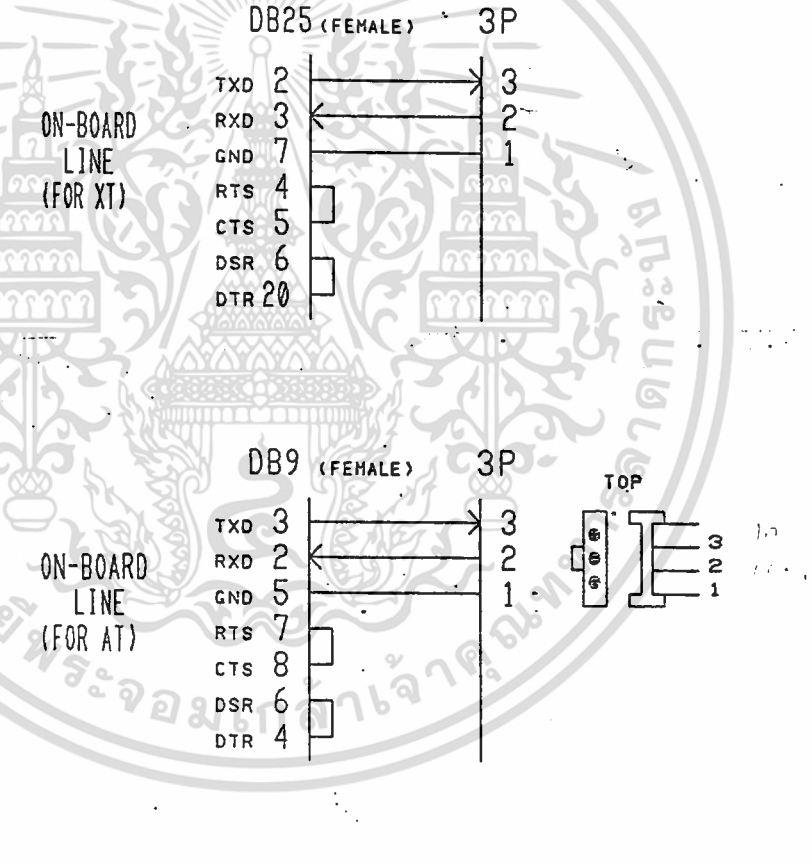
8255 MODE Ø SUMMARY

CONTROL CODE IN MODE Ø.

PORT A	PORT B	PORT CØ-C3	PORT C4-C7	CODE(HEX)
O	O	O	O	8Ø
O	O	O	I	88
O	O	I	O	81
O	O	I	I	89
O	I	O	O	82
O	I	O	I	8A
O	I	I	O	83
O	I	I	I	8B
I	O	O	O	9Ø
I	O	O	I	98
I	O	I	O	91
I	O	I	I	99
I	I	O	O	92
I	I	O	I	9A
I	I	I	O	93
I	I	I	I	9B

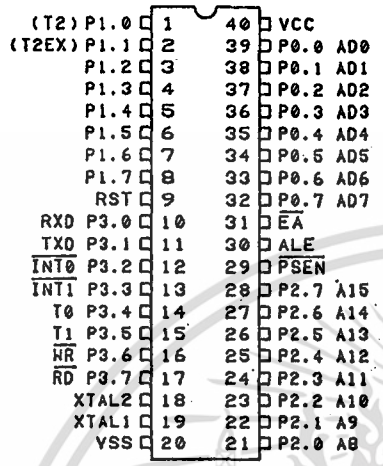


การต่อสาย RS232

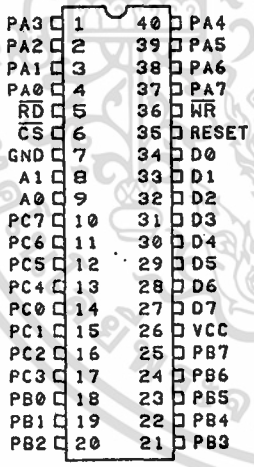


PIN CONFIGURATION

8031(32)



8255



SXA51 8051/52 Cross Assembler

Introduction

Congratulations. You are now the owner of a fast, reliable cross-assembler for the Intel 8051/52 series of microprocessors.

Many cross-assembler manufacturers base their assemblers on a "table-driven", cross-assembler shell. They simply plug the mnemonics for a new target processor into the shell and introduce another cross-assembler. Binary Technology has been specializing in the 8051/52 since 1982 and the SXA51 was developed exclusively for this family. As a result, SXA51 is faster, more flexible, and has more specific error codes than most competitors' products.

SXA51 is an absolute assembler meaning that it does not produce relocatable modules. Relocating assemblers require the use of a linker in order to generate absolute code. SXA51 generates absolute code in only one step, thus considerably shortening the development process. We provide for modular program development by including a utility which creates one large object file from several smaller object files. The HEX utility sorts and merges Intel hex format files and reports on start address, finish address, gaps, overlaps, checksum, and entry address. It can also compare and report on the differences between two object files.

Another advantage of SXA51 is its flexibility. Labels and symbols can be any length. Underscores are valid alpha characters. Dollar signs can be embedded in symbols for readability and labels do not have to be against the left margin. All this provides the freedom to use any programming style you wish; even an indented modular style.

This manual assumes familiarity with the 8051/52 family and with their assembly language mnemonics. If you feel unfamiliar in this area, we recommend the *'EMBEDDED CONTROLLER HANDBOOK'* (Intel order number 210918-005). Chapter 7, *'MCS-51 Programmers Guide and Instruction Set'* contains most of what you will need.

Syntax Notation

The syntax notation used in this manual is as follows: items in square brackets are optional, ellipsis (...) are used to indicate that 'more of the same' is permissible, and italics mean substitute your own names. A typical source file line might be:

[*LABEL*:] OPCODE [*operand1*[, *operand2*]] [;*comment...*]

This means an optional label, followed by "OPCODE" followed (optionally) by one or two operands, followed by an optional comment.

Invoking SXA51

The cross-assembler is invoked as: SXA51 *filename*. This will assemble the source file *filename*.asm while outputting the object (machine) code in Intel hex format into *filename*.hex. Error messages and source lines containing errors will be written to the screen.

The above behaviour can be modified with the following flags: -n will suppress generation of the object file. -l will generate a file *filename*.lst which will contain a complete listing file, and a symbol table. -c will replace the symbol table with a symbol cross-reference. -d will make the assembler more verbose while it is running. These flags can be combined in any way desired, for example:

```
SXA51 -lcd foobar
```

will assemble 'foobar.asm'. A listing file with a cross reference (foobar.lst) will be generated and SXA51 will report on its progress as it runs.

In summary:

-l = Make a listing file (*filename*.lst)
-d = Be verbose
-n = Do not make an object (*filename*.hex) file
-c = Include a symbol cross-reference listing

Symbols and Labels

Symbols and labels can be any length and may contain underscores and dollar signs. All characters A-Z and 0-9 may be used but the first character must be alphabetic. SXA51 does not differentiate between upper case and lower case (i.e., LaBeL: and LABEL: are the same). Only the first 16 characters are significant. This means that SXA51 will treat SYMBOL_SEVENTY_FOUR and SYMBOL_SEVENTY_FIVE as the same. Dollar signs inside symbols or labels are ignored by the assembler. Labels must be terminated with a colon and do not have to be against the left margin.

SXA51 stores all symbols and labels as 16 bit values.

Comments

Comments begin with a semicolon. SXA51 will ignore any text to the right of a semicolon.

The operation of the assembler can also be controlled from inside the source through the use of controls. A leading dollar sign '$' is required. Some controls cannot be changed once the assembler has begun generating code. These are the *primary* controls and are identified as such in the descriptions below. Primary controls should only be used at the beginning of the source.

$[no]date (Defaults: MS-DOS $nodate; VMS $date)

This causes the system date and time to be included on the top right of all listing pages. The format is `Mon 04-Apr-1987 12:08`. It is in 24 hour format.

$eject

This forces the listing to continue at the beginning of the next page. It will do so in the manner dictated by the setting of '$formfeed' below.

$[no]formfeed (Default: $noformfeed)

$formfeed causes multiple linefeeds (based on the value of '$length') to be used instead of a formfeed. $noformfeed is for older printers that don't respond to a formfeed.

$[no]list (Default: $list)

$nolist turns off the listing. This is useful if you have a large file and only want to examine the listing of a small part of it. List and nolist do not affect the symbol table or cross-reference. Therefore, a $nolist at the beginning of the source will cause only the symbol table or cross-reference to be generated.

$if *expression*
$else
$endif

Conditional assembly. Causes the assembler to jump over blocks of code based on the evaluation of the expression following '$if'. For example:

```
TEST_ONE  EQU    0
TEST_TWO  EQU    1

START:
          do this
          do this
          do this

$if TEST_ONE
          don't do this
          don't do this
          don't do this
$else
          do this
          do this
          do this
$endif

$if NOT(TEST_TWO)
          don't do this
          don't do this
          don't do this
$endif
          do this
          do this
          etc.
```

$length *n* (Default: MS-DOS $length 66; VMS $length 0)

(Primary) Sets the length (in lines) of the listing page.

$include "*filename*"

Include the text from another file at this point as if it were written into this file. The lines from an include file are marked in the listing file with a '+'. Includes cannot be nested which means that a file 'included' in another file cannot have an include in it. The name of the file to be included must be enclosed in double quotes.

For example:

```
$include "C:\subdir\header.inc" (MS-DOS)
                - or -
$include "DUA0:[subdir]header.inc" (VAX/VMS)
```

$[no]nullfill (Default: $nonullfill)

'DS *n*' (define space) causes some assemblers to generate an entry in the object file which will cause *n* nulls (zeros) to be loaded into program memory. Other assemblers simply jump over *n* locations and do not generate any program code for that space. With SXA51 you have a choice. $nullfill generates zeros while $nonullfill does not. The latter is mandatory when 'DS'ing over a space that is to be overlaid later with another object file. (See the description of the HEX utility.)

$title "*string*"

Causes *string* to be added to the top of each page of the program listing. The title must be enclosed in double quotes. It will default to the program source name if the quotes are omitted. The title can be no more than 48 characters in length.

$width *n* (Default: $width 132)

(Primary) Sets the width (in characters) of the program listing. This actually sets the width of the listing page header and the symbol table or cross reference. It will not truncate source lines.

$xref (Default: $noxref)

(Primary) Generates a symbol cross-reference. This is the same as using the '-c' invocation flag.

Assembler Directives

DB *expression*[, *expression*...]

DB (define byte) will place a byte or sequence of bytes into program memory. It can also insert an ascii string into sequential memory locations if the string is enclosed in single quotes. The following will place a 'null terminated' string into sequential memory locations beginning at the label 'POINTER:'

```
POINTER:   DB 'This is a string',0
```

DS *expression*

DS (define space) will set aside space in program memory based on the evaluation of *expression* (also see the description of $nullfill).

DW *expression*[, *expression*...]

DW (define word) will write a 16 bit value or sequence of values into program memory high byte first, then low byte. *Note that this is reversed from earlier Intel standards.*

ORG *expression*

Resets the program memory pointer to a new value. Subsequent code will be placed in memory beginning at *expression.*

END [*expression*]

End terminates the program and (optionally) indicates the code entry address.

SYMBOL_NAME EQU *expression*
SYMBOL_NAME BIT *expression*

Assigns the value of *expression* to 'symbol_name'. The 'BIT' operator is retained for compatability with Intel assemblers. SXA51 makes no distinction between the two types.

SYMBOL_NAME SET *expression*

SET is the same as EQU but does not complain if the same symbol is assigned a new value later on.

Expressions

Expressions can be made up of numbers and/or symbols. Numbers are assumed to be decimal unless indicated otherwise. Hex numbers must begin with a digit and end with an "H". 30H and 0F4H are valid hex numbers; F3H is not. Octal numbers end with an "O" or "Q" and binary numbers end with a "B". The special symbol, "$", refers to the program location "at the beginning of the present line". For example, the line, "`JMP $`", will create an endless loop.

The order of precedence for operators is:

Highest: unary + unary - HIGH LOW
^ (exponentation)
* /
+ - MOD SHR SHL AND OR XOR
Lowest: EQ NE GE LE GT LT

The expression parser will evaluate left-to-right if the operators are of equal precedence. The best rule of thumb is to use parentheses when in doubt.

Bit Operations

SXA51 adheres to the Intel convention of bit notation. That is, *n.m* where *n* is the bit-addressable byte address and *m* is the bit inside that byte. For example, 22H.3 refers to the fourth bit in bit-addressable byte 3. (Bit-addressable memory begins at location 20H). This would translate to bit address 13H. Notations such as ACC.7 are also permitted.

There is a small price to pay for the added speed of working with an absolute assembler. Specifically, you will have to keep your code module origins and routine addresses organized. This is not as bad as it might initially sound, and the difference between a 30 second assemble and a 15 minute assemble and link is worth it.

Suppose that you have finished one module and are now working on a second that contains a routine which will be called by the first. The location of this routine will probably move relative to the second module's origin as the module evolves. The best way to deal with this is to put an LJMP instruction at the beginning of the second module causing a jump into the actual routine. The first module will call the routine at the location of the jump. In this way, the first module will not require changes or reassembly as work proceeds on the second module. In fact, the first module could be burned into a PROM to speed up the downloading process during development and debugging. Several routines in one module can be done in this manner by using a sequence of LJMPs at the beginning of the module. Once the code is stable, the extra LJMPs can be removed without danger of changing the operation of the program.

Another modular development technique used with SXA51 is to create an $include file which contains the EQUates for origins and EQUates for the locations of "public" routines. Public routines are any which must be called from a different module. This file can then be included in each of the modules so there will be no differences of opinion regarding the location of any given routine.

One example of this method is demonstrated by the use of the Binary Technology M/DP monitor-debugger. The M/DP contains several useful routines such as CHROUT (send a character out the serial port). The location given in the M/DP documentation for CHROUT is 0008H. (This location actually contains an LJMP to the real CHROUT routine inside the M/DP.) The next step is to create a file ('mdp.inc', for example) which contains the line 'CHROUT EQU 0008H'. Then, in your program, insert the line '$include "mdp.inc". Thereafter, a 'CALL CHROUT' will cause the program to find its way to the routine in the MD/P ROM.

HEX utility

The HEX utility sorts, combines, checks, and/or compares single or multiple Intel hex format files.

It is invoked as:

HEX [-dco] *filename1* [*filename2...*][, *outfile*]

This will sort and merge filename1.hex, filename2.hex, etc. and compare the result with outfile.hex. It will also report on the starting address, the ending address, gaps, overlaps, checksum, and the entry address.

The flags are as follows:

-d = report should be in decimal (rather than hex)
-c = compare with (rather than write into) outfile.hex
-o = override 'output file exists' message

When merging files, overlaps should be considered fatal. The down-loader or PROM programmer is not going to know which entry for a given address is valid.

Version 1.0 5/25/1987

SXA51 Error Messages

Byte:

B	Value out-of-bounds: Must be between 0 and 255

Control:

C1	A primary control must be used before any code generation
C2	Unrecognized control

Expression:

E1	Unbalanced parentheses
E2	Cannot translate variable name
E3	Divide by zero
E4	Incorrect bit address
E5	General expression error
E6	Radix error
E7	Bit.n: n must be between 0 and 7

Include

I	Include nesting too deep

Multiply defined.

M1	Label multiply defined
M2	EQU or SET multiply defined

Offset error:

O1	ACALL or AJMP must be inside 4096 byte 'page'
O2	SJMP must be within +127 or -128

Syntax:

S	General.
S1	First char. on line must be alphabetic
S2	Label error: probably a reserved word
S3	First char. (after label) must be alphabetic
S4	Reserved word in EQU or SET
S5	Incorrect use of opcode
S6	Cannot evaluate ORG expression
S7	Incorrect register or data type
S8	Cannot DEC DPTR
S9	Expected 'A'
S10	Expected comma

S11	Expected 'AB'
S12	Expected '@A+DPTR'
S13	Expected '@A+PC' or '@A+DPTR'
S14	Expected 'A, @DPTR', 'A, @R0' or 'A, @R1'
S15	Expected '@R0' or '@R1'
S16	SETB operand must be Bit or C
S17	String must be enclosed in quotes or is too long

Unrecognized:

U	General
U0	Opcode

Pass 1 errors:

"Incorrect forward reference"

This error is caused if a symbol is referenced that has not yet been defined yet and the symbol is one which can affect the location of the code being generated. An error is flagged during the first pass of the assembler and it is aborted. Such uses would be: "DS *undefined_symbol*" or "JMP *undefined_label*" (This is the generic version of the jump instruction which will be converted by the assembler into either a one or a two byte jump instruction.) A "CALL *undefined_label*" will fail for the same reason as the JMP.

"No END statement."

This is a non-fatal error. It is good practice to include an END statement.

Built-in 8051/52 Register Names

Bit addresses

TF1	8F	SM0	9F	EA	AF
TR1	8E	SM1	9E		
TF0	8D	SM2	9D	ET2	AD
TR0	8C	REN	9C	ES	AC
IE1	8B	TB8	9B	ET1	AB
IT1	8A	RB8	9A	EX1	AA
IE0	89	TI	99	ET0	A9
IT0	88	RI	98	EX0	A8
RD	B7			CY	D7
WR	B6			AC	D6
T1	B5	PT2	BD	F0	D5
T0	B4	PS	BC	RS1	D4
INT1	B3	PT1	BB	RS0	D3
INT0	B2	PX1	BA	OV	D2
TXD	B1	PT0	B9		
RXD	B0	PX0	B8	P	D0

Direct addresses

ACC	E0	P3	B0	T2CON	C8
B	F0	PCON	87	TH0	8C
DPH	83	PSW	D0	TL0	8A
DPL	82	RCAP2H	CB	TH1	8D
IE	A8	RCAP2L	CA	TL1	8B
IP	B8	SBUF	99	TH2	CD
P0	80	SCON	98	TL2	CC
P1	90	SP	81	TMOD	89
P2	A0	TCON	88		

Predefined Origins

RESET	0000
EXTI0	0003
TIMER0	000B
EXTI1	0013
TIMER1	001B
SINT	0023

1.4 QUICK REFERENCE

COMMANDS:

COMMAND	FUNCTION	EXAMPLE(S)
RUN	Execute a program	RUN
CONT	CONTinue after a STOP or control-C	CONT
LIST	LIST program to the console device	LIST LIST 10-50
LIST#	LIST program to serial printer	LIST# LIST# 50
LIST@	LIST program to user driver (version 1.1 only)	LIST@ LIST@ 50
NEW	erase the program stored in RAM	NEW
NULL	set NULL count after carriage return-line feed	NULL NULL 4
RAM	evoke RAM mode, current program in READ/WRITE memory	RAM
ROM	evoke ROM mode, current program in ROM/EPROM memory	ROM ROM 3
XFER	transfer a program from ROM/EPROM to RAM	XFER
PROG	save the current program in EPROM	PROG
PROG1	save baud rate information in EPROM	PROG1
PROG2	save baud rate information in EPROM and execute program after RESET	PROG2
PROG3	save baud rate and MTOP information in EPROM (version 1.1 only)	PROG3
PROG4	save baud rate and MTOP information in EPROM and execute program after RESET (version 1.1 only)	PROG4

1.4 QUICK REFERENCE

COMMANDS:

COMMAND	FUNCTION	EXAMPLE(S)
PROG5	same as PROG4 except that external RAM is not cleared on RESET or power up if external RAM contains a 0A5H in location 5EH (version 1.1 only)	PROG5
PROG6	same as PROG6 except that external code location 4039H is CALLED after RESET (version 1.1 only)	PROG6
FPROG	save the current program in EPROM using the INTELligent algorithm	FPROG
FPROG1	save baud rate information in EPROM using the INTELligent algorithm	FPROG1
FPROG2	save baud rate information in EPROM and execute program after RESET, use INTELligent algorithm	FPROG2
FPROG3	same as PROG3, except INTELligent programming algorithm is used (version 1.1 only)	FPROG3
FPROG4	same as PROG4, except INTELligent programming algorithm is used (version 1.1 only)	FPROG4
FPROG5	same as PROG5, except INTELligent programming algorithm is used (version 1.1 only)	FPROG5
FPROG6	same as PROG6, except INTELligent programming algorithm is used (version 1.1 only)	FPROG6

1.4 QUICK REFERENCE

STATEMENTS:

STATEMENT	FUNCTION	EXAMPLE(S)
BAUD	set baud rate for line printer port	BAUD 1200
CALL	CALL assembly language program	CALL 9000H
CLEAR	CLEAR variables, interrupts and Strings	CLEAR
CLEARS	CLEAR Stacks	CLEARS
CLEARI	CLEAR Interrupts	CLEARI
CLOCK1	enable REAL TIME CLOCK	CLOCK1
CLOCK0	disable REAL TIME CLOCK	CLOCK0
DATA	DATA to be read by READ statement	DATA 100
READ	READ data in DATA statement	READ A
RESTORE	RESTORE READ pointer	RESTORE
DIM	allocate memory for arrayed variables	DIM A(20)
DO	set up loop for WHILE or UNTIL	DO
UNTIL	test DO loop condition (loop if false)	UNTIL A = 10
WHILE	test DO loop condition (loop if true)	WHILE A = B
END	terminate program execution	END
FOR-TO-{STEP}	set up FOR-NEXT loop	FOR A = 1 TO 5
NEXT	test FOR-NEXT loop condition	NEXT A

1.4 QUICK REFERENCE

STATEMENTS:

STATEMENT	FUNCTION	EXAMPLE(S)
GOSUB	execute subroutine	GOSUB 1000
RETURN	RETURN from subroutine	RETURN
GOTO	GOTO program line number	GOTO 500
ON GOTO	conditional GOTO	ON A GOTO 5, 20
ON GOSUB	conditional GOSUB	ON A GOSUB 2, 6
IF-THEN-{ELSE}	conditional test	IF A<B THEN A = 0
INPUT	INPUT a string or variable	INPUT A
LET	assign a variable or string a value (LET is optional)	LET A 10
ONERR	ONERRor GOTO line number	ONERR 1000
ONTIME	generate an interrupt when TIME is equal to or greater than ONTIME argument-line number is after comma	ONTIME 10, 1000
ONEX1	GOSUB to line number following ONEX1 when INT1 pin is pulled low	ONEX1 1000
PRINT	PRINT variables, strings or literals P. is shorthand for PRINT	PRINT A
PRINT#	PRINT to software serial port	PRINT# A
PH0.	PRINT HEX mode with zero suppression	PH0. A
PH1.	PRINT HEX mode with no zero suppression	PH1. A
PH0.#	PH0. to line printer	PH0.# A
PH1.#	PH1.# to line printer	PH1.# A

1.4 QUICK REFERENCE

STATEMENTS:

STATEMENT	FUNCTION	EXAMPLE(S)
PRINT@	PRINT to user defined driver (version 1.1 only)	PRINT@ 5*5
PH0.@	PH0. to user defined driver (version 1.1 only)	PH0. @ XBY(5EH)
PH1.@	PH1. to user defined driver (version 1.1 only)	PH1.@ A
PGM	Program an EPROM (version 1.1 only)	PGM
PUSH	PUSH expressions on argument stack	PUSH 10, A
POP	POP argument stack to variables	POP A, B, C
PWM	PULSE WIDTH MODULATION	PWM 50, 50, 100
REM	REMark	REM DONE
RETI	RETurn from Interrupt	RETI
STOP	break program execution	STOP
STRING	allocate memory for STRINGs	STRING 50, 10
UI1	evoke User console Input routine	UI1
UI0	evoke BASIC console Input routine	UI0
UO1	evoke User console Output routine	UO1
UO0	evoke BASIC console Output routine	UO0
ST@	store top of stack at user specified location (version 1.1 only)	ST@ 1000H ST@ A
LD@	load top of stack from user specified location (version 1.1 only)	LD@ 1000H LD@ A
IDLE	wait for interrupt (version 1.1 only)	IDLE
RROM	run a program in EP(ROM) (version 1.1 only)	RROM 3

1.4 QUICK REFERENCE

OPERATORS — DUAL OPERAND:

OPERATOR	FUNCTION	EXAMPLE(S)
+	ADDITION	1+1
/	DIVISION	10/2
**	EXPONENTATION	2**4
*	MULTIPLICATION	4*4
–	SUBTRACTION	8–4
.AND.	LOGICAL AND	10.AND.5
.OR.	LOGICAL OR	2.OR.1
.XOR.	LOGICAL EXCLUSIVE OR	3.XOR.2

OPERATORS — SINGLE OPERAND:

ABS( )	ABSOLUTE VALUE	ABS(–3)
NOT( )	ONES COMPLEMENT	NOT(0)
INT( )	INTEGER	INT(3.2)
SGN( )	SIGN	SGN(–5)
SQR( )	SQUARE ROOT	SQR(100)
RND	RANDOM NUMBER	RND
LOG( )	NATURAL LOG	LOG(10)
EXP( )	"e" (2.7182818) TO THE X	EXP(10)
SIN( )	RETURNS THE SINE OF ARGUMENT	SIN(3.14)
COS( )	RETURNS THE COSINE OF ARGUMENT	COS(0)
TAN( )	RETURNS THE TANGENT OF ARGUMENT	TAN(.707)
ATN( )	RETURNS ARCTANGENT OF ARGUMENT	ATN(1)

1.4 QUICK REFERENCE

OPERATORS — SPECIAL FUNCTION:

CBY( )	READ PROGRAM MEMORY	P. CBY(4000)
DBY( )	READ/ASSIGN INTERNAL DATA MEMORY	DBY(99) = 10
XBY( )	READ/ASSIGN EXTERNAL DATA MEMORY	P. XBY(10)
GET	READ CONSOLE	P. GET
IE	READ/ASSIGN IE REGISTER	IE = 82H
IP	READ/ASSIGN IP REGISTER	IP = 0
PORT1	READ/ASSIGN I/O PORT 1 (P1)	PORT1 = 0FFH
PCON	READ/ASSIGN PCON REGISTER	PCON = 0
RCAP2	READ/ASSIGN RCAP2 (RCAP2H:RCAP2L)	RCAP2 = 100
T2CON	READ/ASSIGN T2CON REGISTER	P. T2CON
TCON	READ/ASSIGN TCON REGISTER	TCON = 10H
TMOD	READ/ASSIGN TMOD REGISTER	P. TMOD
TIME	READ/ASSIGN THE REAL TIME CLOCK	P. TIME
TIMER0	READ/ASSIGN TIMER0 (TH0: TL0)	TIMER0 = 0
TIMER1	READ/ASSIGN TIMER1 (TH1: TL1)	P. TIMER1
TIMER2	READ/ASSIGN TIMER2 (TH2: TL2)	TIMER2 = 0FFH

STORED CONSTANT:

PI	Pi -- 3.1415926	PI

1.5 INSTRUCTION SET SUMMARY

COMMANDS	STATEMENTS	OPERATORS
RUN	BAUD	ADD (+)
CONT	CALL	DIVIDE (/)
LIST	CLEAR	EXPONENTIATION (**)
LIST#	CLEAR(S&I)	MULTIPLY (*)
LIST@ (V1.1)	CLOCK(1&0)	SUBTRACT (-)
NEW	DATA	LOGICAL AND (AND.)
NULL	READ	LOGICAL OR (.OR.)
RAM	RESTORE	LOGICAL X-OR (.XOR.)
ROM	DIM	LOGICAL NOT (.OR)
XFER	DO-WHILE	ABS( )
PROG	DO-UNTIL	INT( )
PROG1	END	SGN( )
PROG2	FOR-TO-STEP	SQR( )
PROG3 (V1.1)	NEXT	RND
PROG4 (V1.1)	GOSUB	LOG( )
PROG5 (V1.1)	RETURN	EXP( )
PROG6 (V1.1)	GOTO	SIN( )
FPROG	ON-GOTO	COS( )
FPROG1	ON-GOSUB	TAN( )
FPROG2	IF-THEN-ELSE	ATN( )
FPROG3 (V1.1)	INPUT	=, >, > =, <, < =, <>
FPROG4 (V1.1)	LET	ASC( )
FPROG5 (V1.1)	ONERR	CHR( )
FPROG6 (V1.1)	ONEX1	CBY( )
	ONTIME	DBY( )
	PRINT	XBY( )
	PRINT#	GET
	PRINT@ (V1.1)	IE
	PH0.	IP
	PH0.#	PORT1
	PH0.@ (V1.1)	PCON
	PH1.	RCAP2
	PH1.#	T2CON
	PH1.@ (V1.1)	TCON
	PGM (V1.1)	TMOD
	PUSH	TIME
	POP	TIMER0
	PWM	TIMER1
	REM	TIMER2
	RETI	XTAL
	STOP	MTOP
	STRING	LEN
	UI(1&0)	FREE
	U0(1&0)	PI
	LD@ (V1.1)	
	ST@ (V1.1)	
	IDLE (V1.1)	
	RROM (V1.1)	

ภาคผนวก ง

DATA SHEET

ICL7109
12-Bit μP-Compatible A/D Converter

INTERSIL

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.

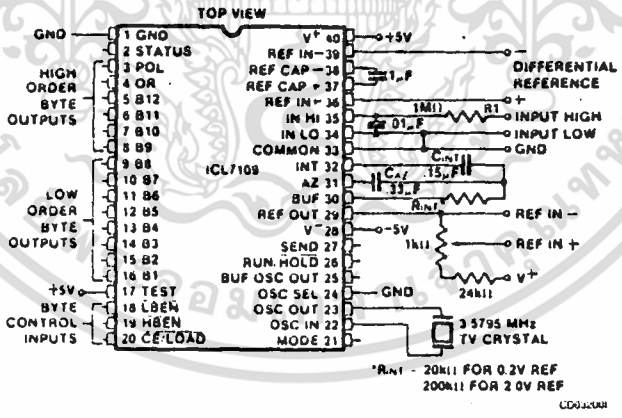
The ICL7109 provides the user with the high accuracy, low noise, low drift, versatility and economy of the dual-slope integrating A/D converter. Features like true differential input and reference, drift of less than 1μV/°C, maximum input bias current of 10pA, and typical power consumption of 20mW make 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

Note: All typical values have been guaranteed by characterization and 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 ..... 1W @ +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

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the devices. This is a stress rating only and functional operation of the devices 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, $T_A = 25°C$, 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} = 0.0V Full Scale = 409.6mV	-0000_8	$\pm0000_8$	$+0000_8$	Octal Reading
	Ratiometric Reading	$V_{IN} = V_{REF}$ V_{REF} = 204.8mV	3777_8	3777_8 4000_8	4000_8	Octal Reading
	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

DIGITAL SECTION

SYMBOL	PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	Output High Voltage	I_{OUT} = 100µA 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			50	pF
V_{IH}	Input High Voltage	Pins 18-21, 26, 27 referred to GND	2.5			V
V_{IL}	Input Low Voltage	Pins 18-21, 26, 27 referred to GND			1	V

Note: All typical values have been guaranteed by characterization and are not tested.

ICL7109

INTERSIL

ELECTRICAL CHARACTERISTICS (CONT.)

SYMBOL	PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
	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}	Octillator 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.

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.
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
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)

(Pins 3–16: HI = true; All three state output data bits)

PIN	SYMBOL	DESCRIPTION
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.
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

Note: All typical values have been guaranteed by characterization and are not tested.

PIN	SYMBOL	DESCRIPTION
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

PIN	SYMBOL	DESCRIPTION
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.

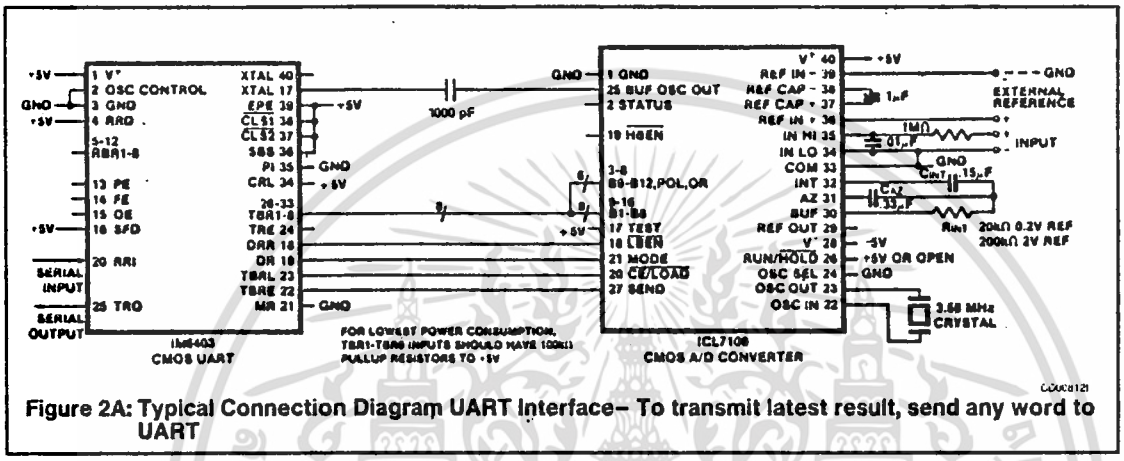


Figure 2A: Typical Connection Diagram UART Interface– To transmit latest result, send any word to UART

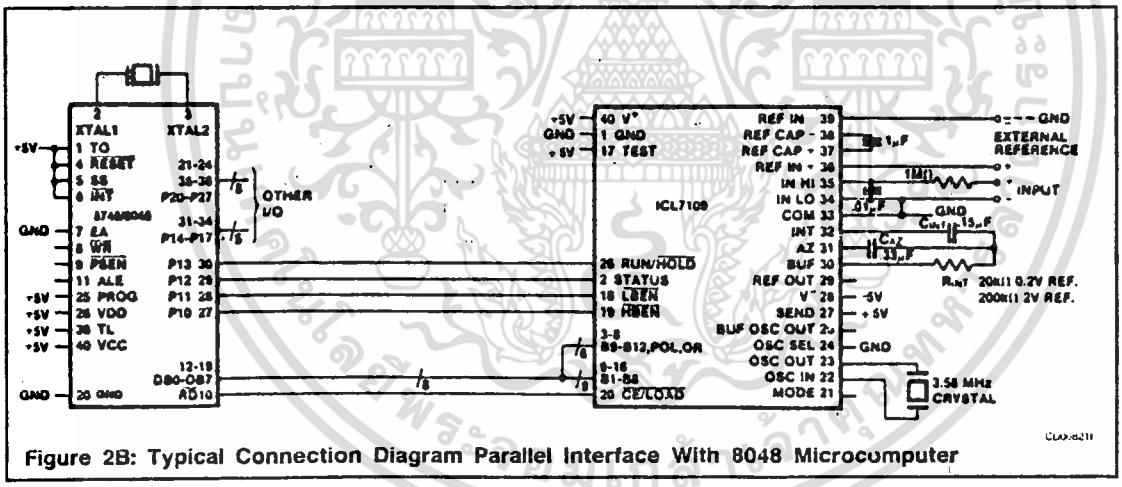


Figure 2B: Typical Connection Diagram Parallel Interface With 8048 Microcomputer

DETAILED DESCRIPTION

Analog Section

Figure 3 shows the equivalent circuit of the Analog Section of the ICL7109. When the RUN/$\overline{HOLD}$ 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 guaranteed by characterization and are not tested.

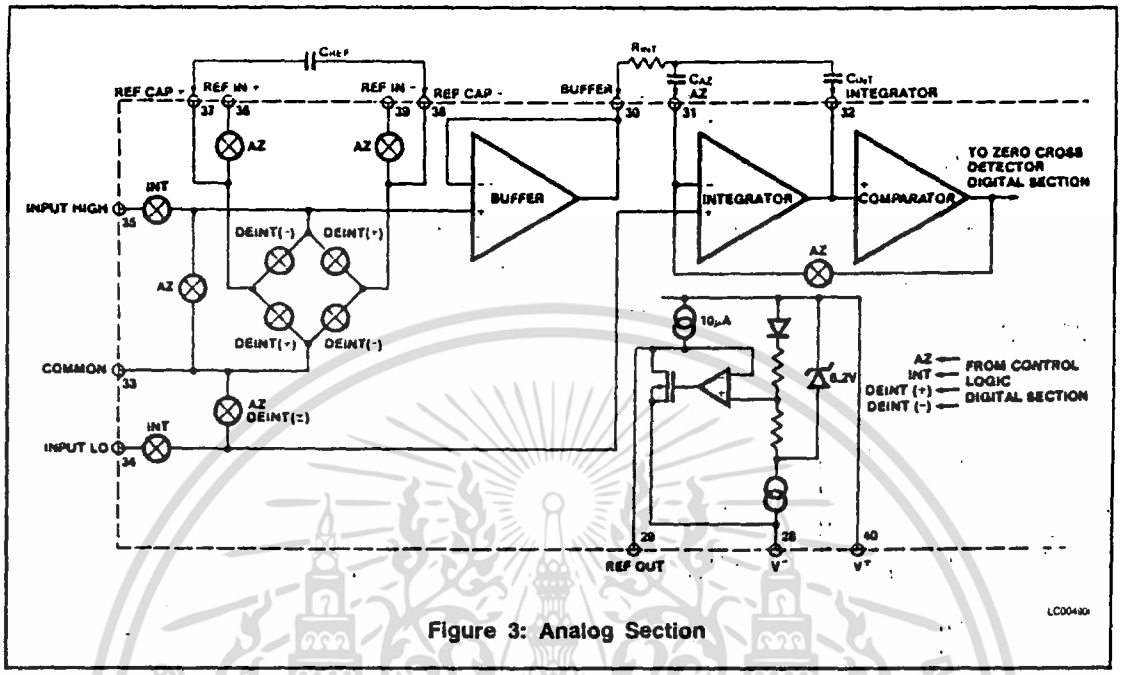


Figure 3: Analog Section

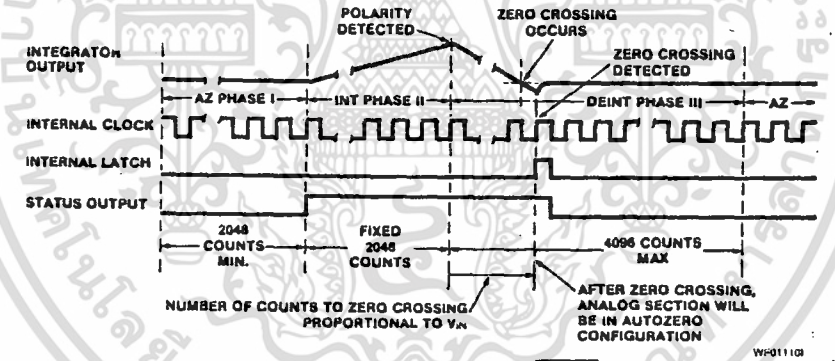


Figure 4: Conversion Timing (RUN/HOLD Pin High)

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.

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

Note: All typical values have been guaranteed by characterization and are not tested.

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 7-1/2 conversions per second (61.72kHz clock frequency) as provided by the crystal oscillator, nominal values for C_{INT} and C_{AZ} are 0.15μF 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

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

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: the smaller the capacitor the 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.

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.

Note: All typical values have been guaranteed by characterization and are not tested.

Reference Voltage

The analog input required to generate a full scale output of 4096 counts is $V_{IN} = 2V_{REF}$. Thus for a normalized scale, a reference of 2.048V should be used for a 4.096V full scale, and 204.8mV should be used for a 0.4096V full scale. However, in many applications where the A/D is sensing the output of a transducer, there will exist a scale factor other than unity between the absolute output voltage to be measured and a desired digital output. For instance, in a weighing system, the designer might like to have a full scale reading when the voltage from the transducer is 0.682V. Instead of dividing the input down to 409.6mV, the input voltage should be measured directly and a reference voltage of 0.341V should be used. Suitable values for integrating resistor and capacitor are 34kΩ 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 Selection

The digital section includes the clock oscillator and scaling circuit, a 12-bit binary counter with output latches and TTL-compatible three-state output drivers, polarity, over-range and control logic, and UART handshake logic, as shown in 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/$\overline{\text{HOLD}}$ 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 guaranteed by characterization and are not tested.

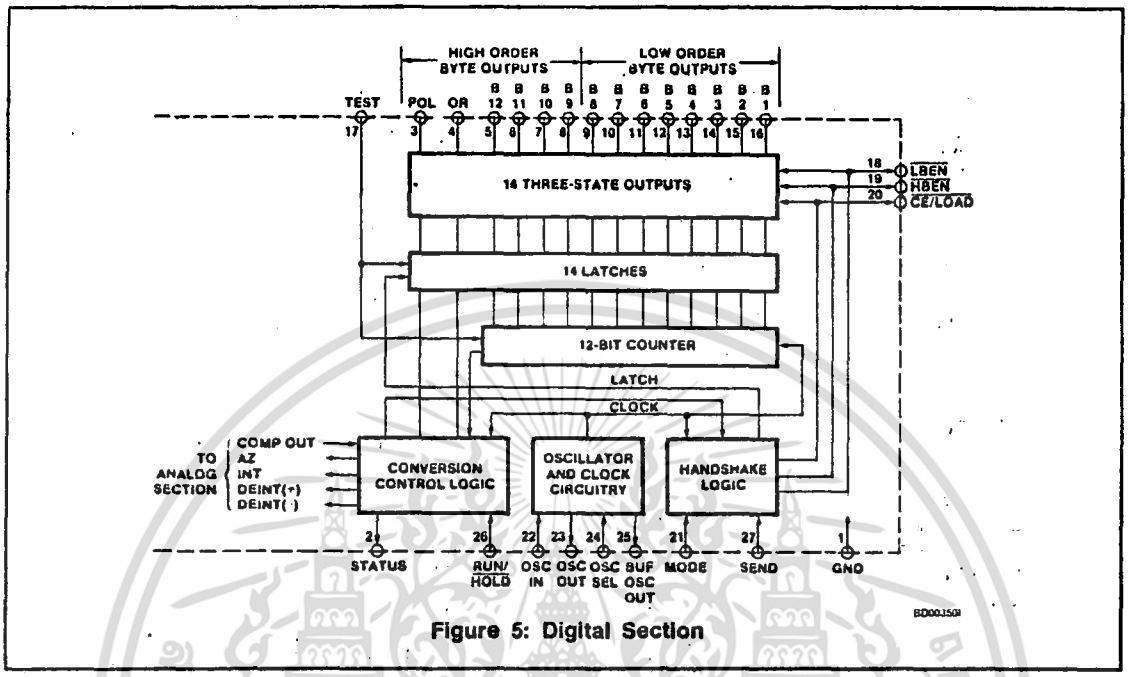


Figure 5: Digital Section

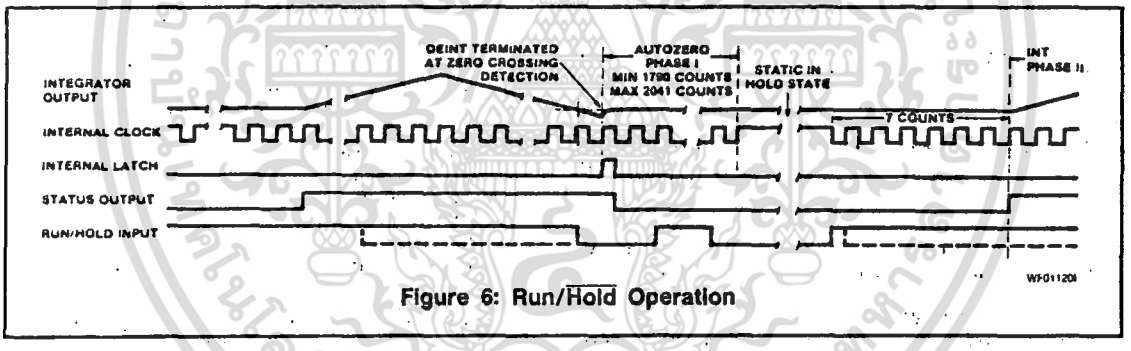


Figure 6: Run/Hold Operation

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

Alternately, RUN/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/HOLD input can be provided by connecting it to the Buffered Oscillator Output. In this mode the conversion time is dependent on the input value measured. Also refer to Intersil Application Bulletin A032 for a discussion of the effects this will have on Auto-Zero performance.

If the RUN/HOLD input goes low and stays low during Auto-Zero (Phase I), the converter will simply stop at the end of Auto-Zero and wait for RUN/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 guaranteed by characterization and 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

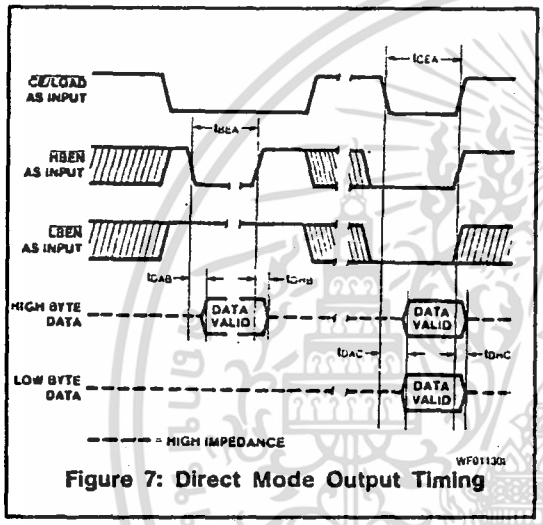


Figure 7: Direct Mode Output Timing

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 Intersil 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 sorial 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 1-1/2 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.

Note: All typical values have been guaranteed by characterization and are not tested.

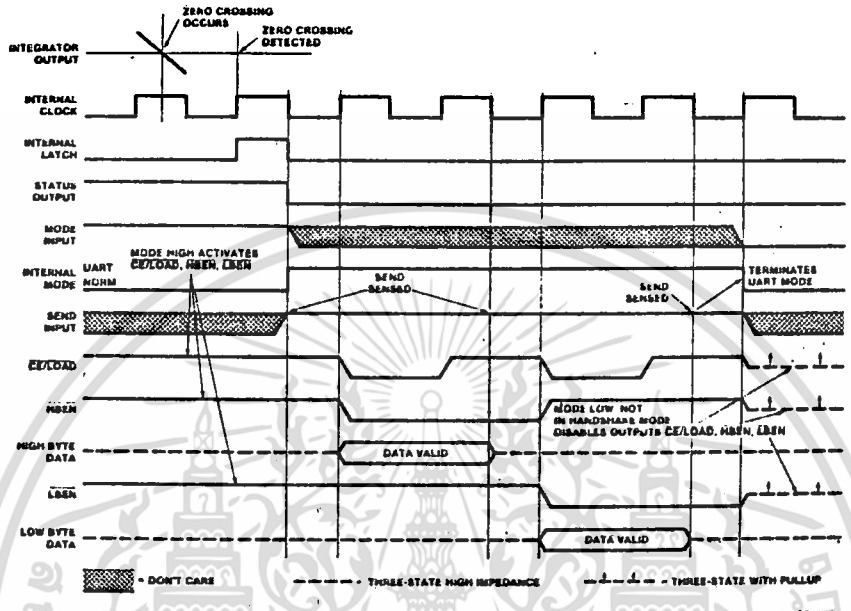


Figure 8: Handshake With Send Held Positive

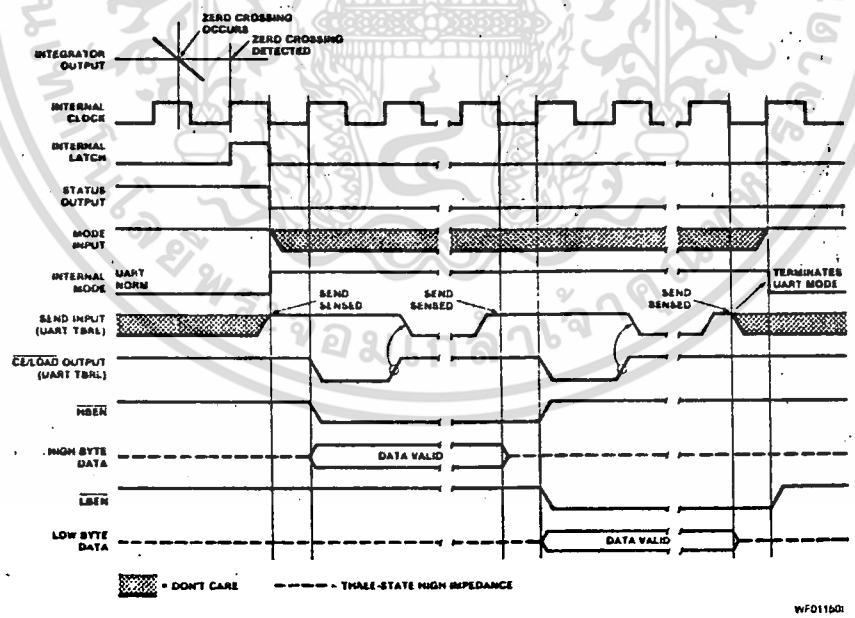


Figure 9: Handshake — Typical UART Interface Timing

Note. All typical values have been guaranteed by characterization and are not tested.

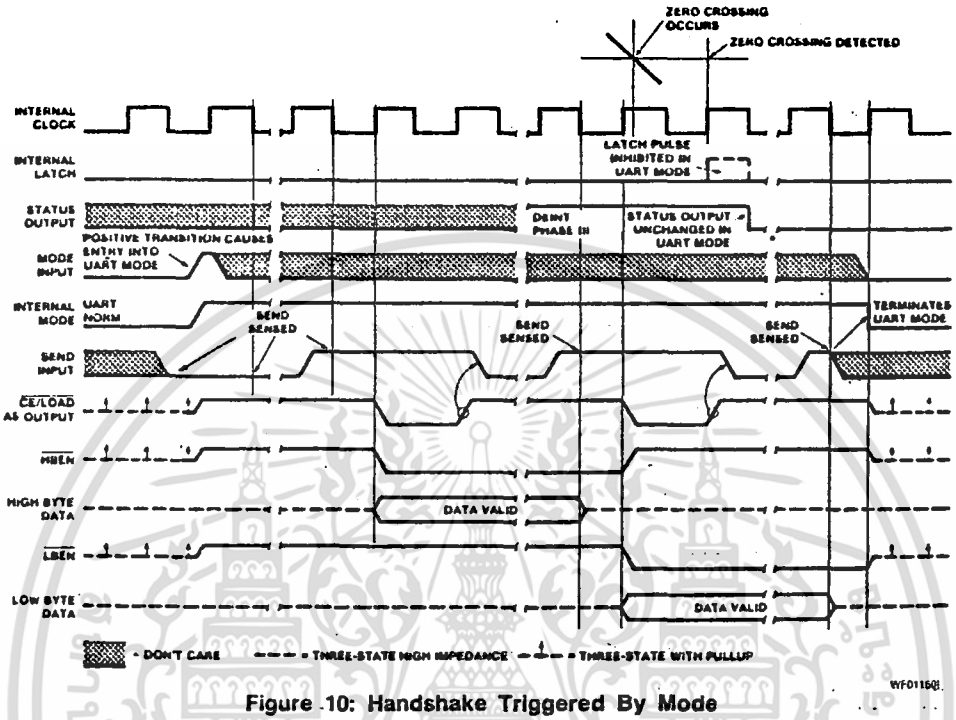


Figure 10: Handshake Triggered By Mode

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 9 shows a handshake output sequence triggered by such an edge. In addition, the SEND input is shown as being low when the converter enters handshake mode. In this case, the whole output sequence is controlled by the SEND input, and the sequence for the first (high order) byte is similar to the sequence for the second byte. This diagram also shows the output sequence taking longer than a conversion cycle. Note that the converter still makes conversions, with the STATUS output and RUN/$\overline{HOLD}$ 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.

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 $f \approx 0.45/RC$. A 100kΩ resistor is recommended for useful ranges of

Note: All typical values have been guaranteed by characterization and are not tested.

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).

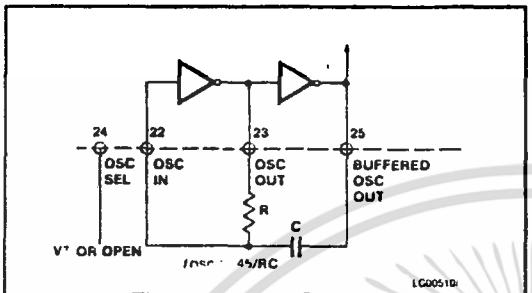


Figure 11: RC Oscillator

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 11, 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:

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

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

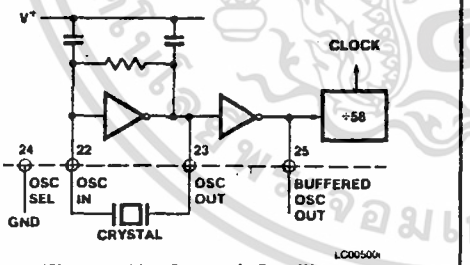


Figure 12: Crystal Oscillator

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 IM6403 UART, it is possible to use one 3.58MHz crystal for both devices. The BUFFERED OSCILLATOR OUTPUT of the ICL7109 may be used to drive the OSCILLATOR INPUT of the UART, saving the need for a second crystal. However, the BUFFERED OSCILLATOR OUTPUT does not have a great deal of drive 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 1/2 (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 $\overline{CE}$/$\overline{LOAD}$ 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 $\overline{CE}$/$\overline{LOAD}$ serves as a chip enable, and the $\overline{HBEN}$ and $\overline{LBEN}$ 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 $\overline{HBEN}$ and $\overline{LBEN}$ as flag inputs, and $\overline{CE}$/$\overline{LOAD}$ as a master enable, which could be the READ strobe available from most microprocessors.

Note: All typical values have been guaranteed by characterization and are not tested.

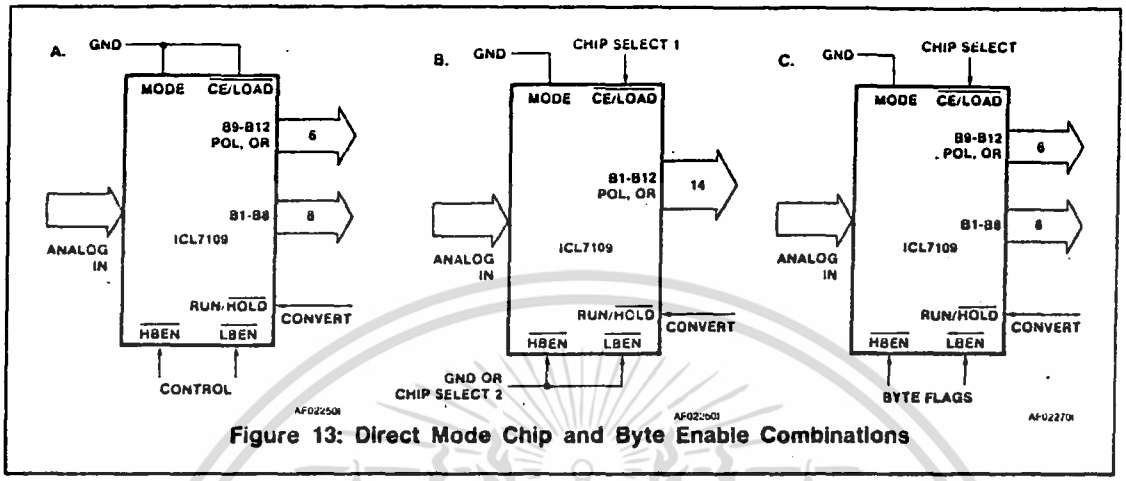


Figure 13: Direct Mode Chip and Byte Enable Combinations

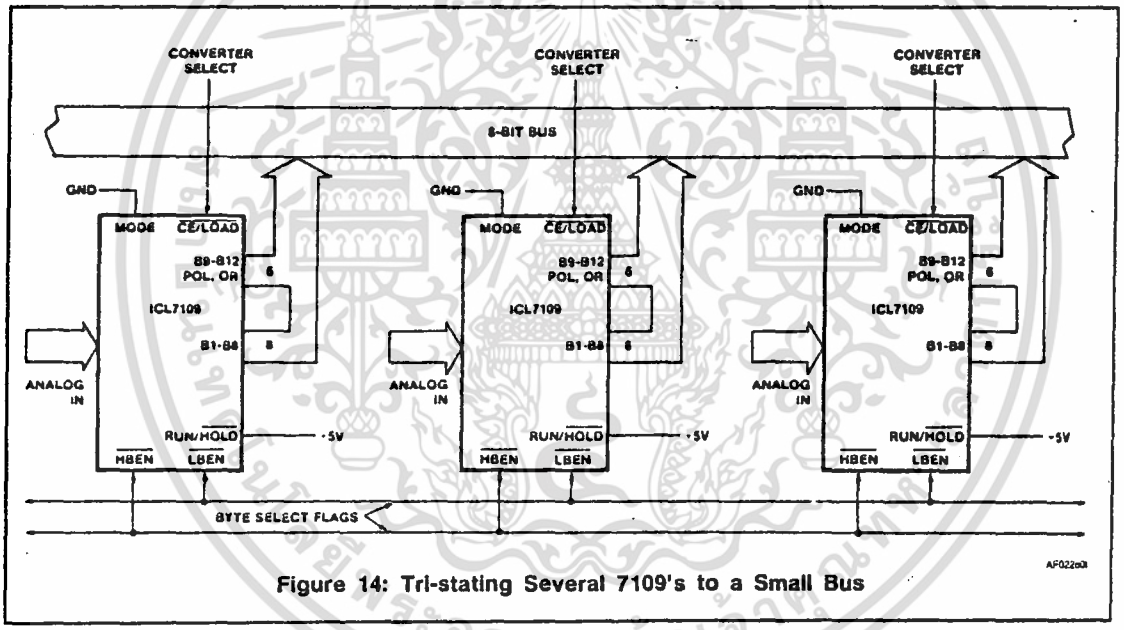


Figure 14: Tri-stating Several 7109's to a Small Bus

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

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 1/2 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/$\overline{HOLD}$ input being used to initiate conversions under software control.

Note: All typical values have been guaranteed by characterization and are not tested.

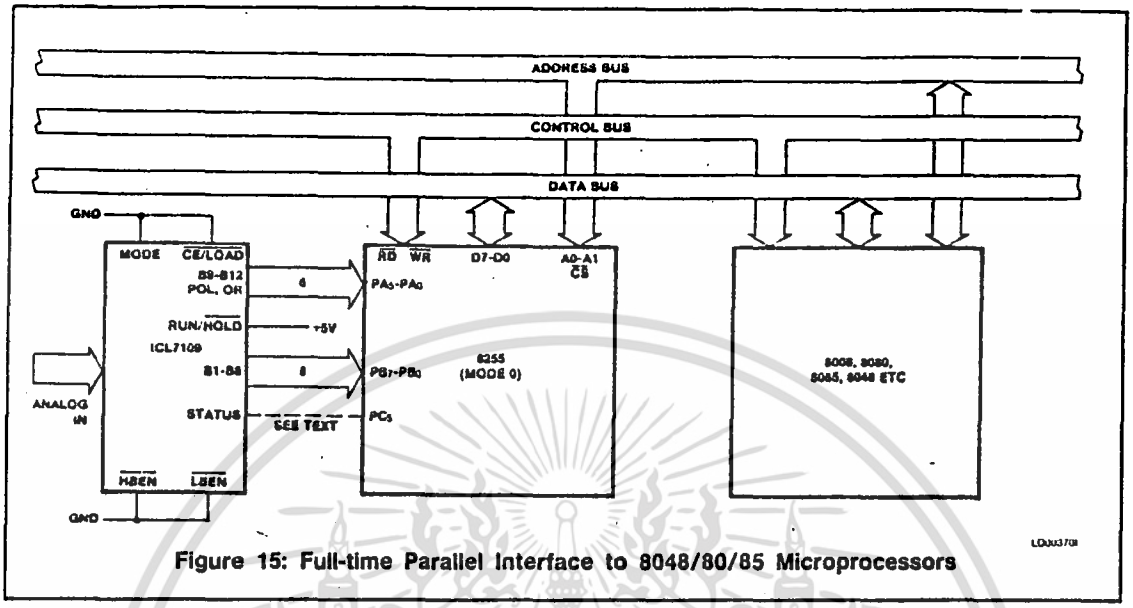


Figure 15: Full-time Parallel Interface to 8048/80/85 Microprocessors

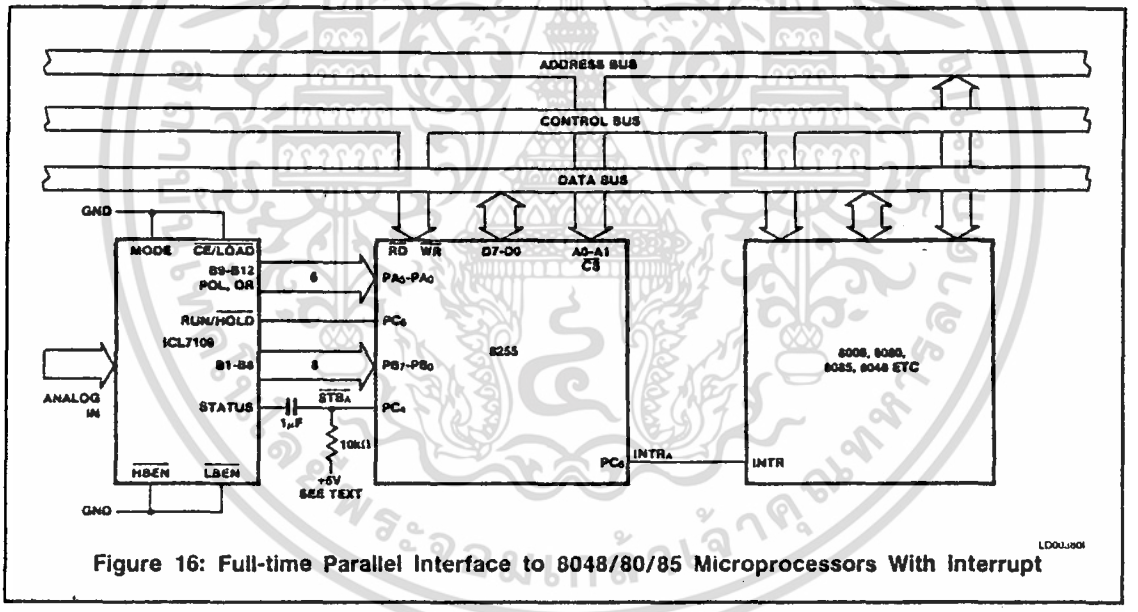


Figure 16: Full-time Parallel Interface to 8048/80/85 Microprocessors With Interrupt

A similar interface to Motorola MC6800 or MOS Technology MCS650X systems is shown in Figure 17. The high to low transition of the STATUS output generates an interrupt via the 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 guaranteed by characterization and are not tested.

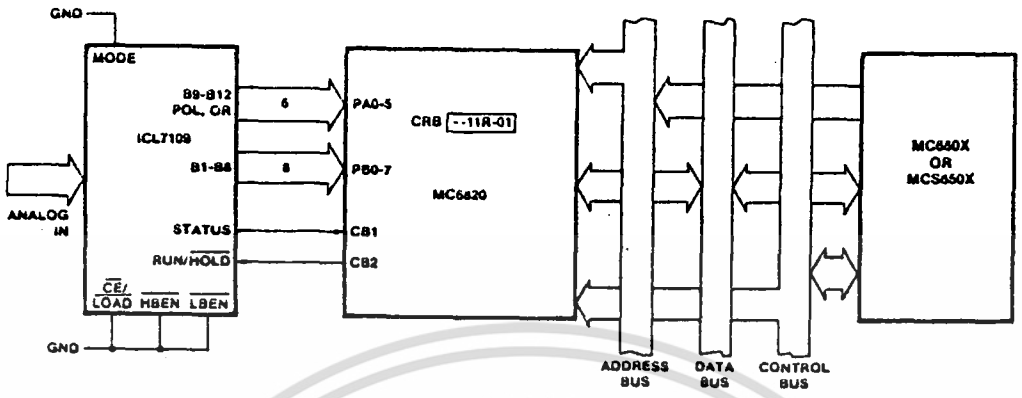


Figure 17: Full-time Parallel Interface to MC680X or MCS650X Microprocessors

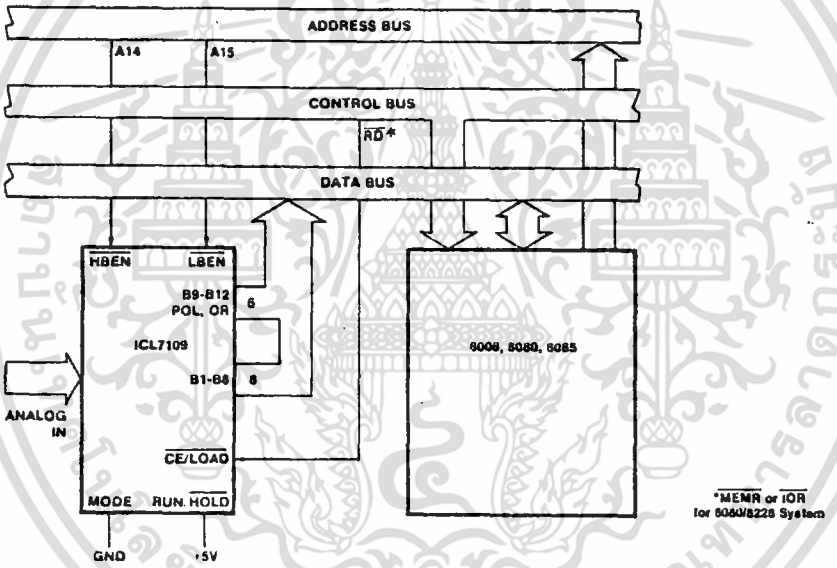


Figure 18: Direct Interface — ICL7109 to 8080/8085

Note: All typical values have been guaranteed by characterization and are not tested.

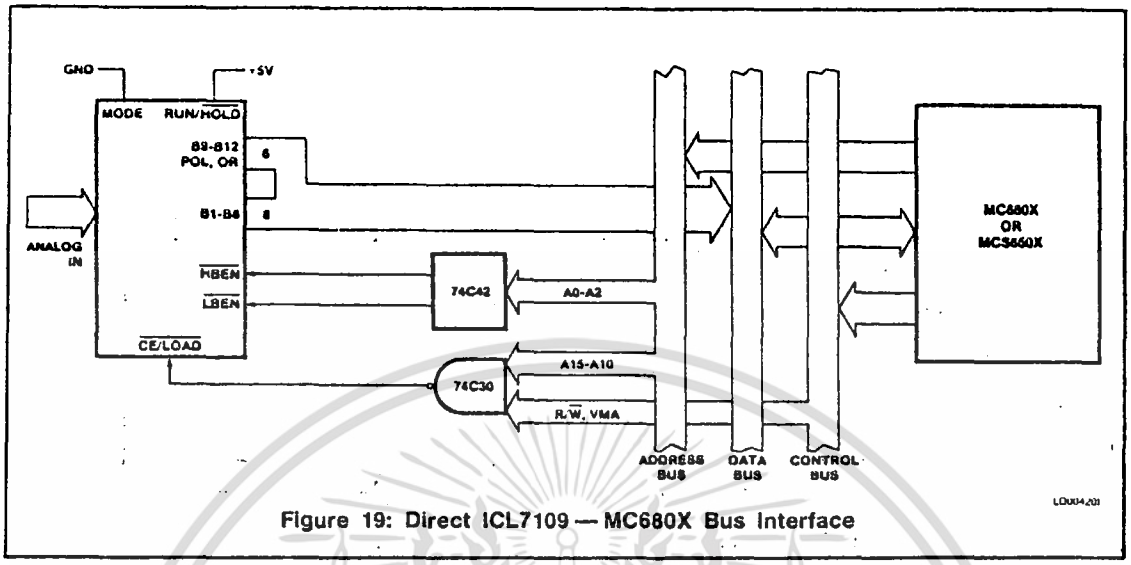


Figure 19: Direct ICL7109 — MC680X Bus Interface

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 $\overline{CE/LOAD}$, 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 $\overline{SEND}$ input to the ICL7109, and using the $\overline{CE/LOAD}$ 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 ($\overline{SEND}$ 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/$\overline{HOLD}$ 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/$\overline{HOLD}$ are tied high to save port outputs.

The handshake mode is particularly convenient for directly interfacing to industry standard UARTs (such as the Intersil IM6402/6403 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, $\overline{LBEN}$ 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 guaranteed by characterization and are not tested.

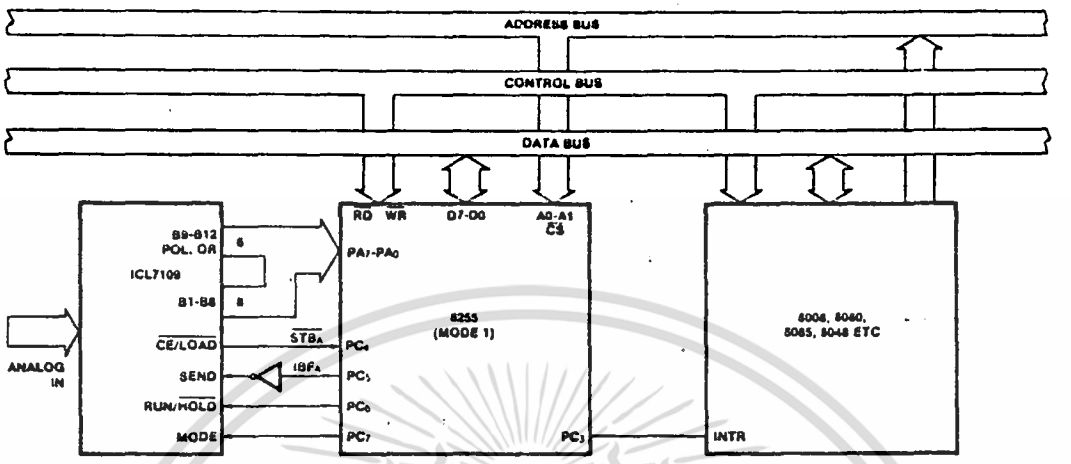


Figure 20: Handshake Interface — ICL7109 to 8048, 80/85

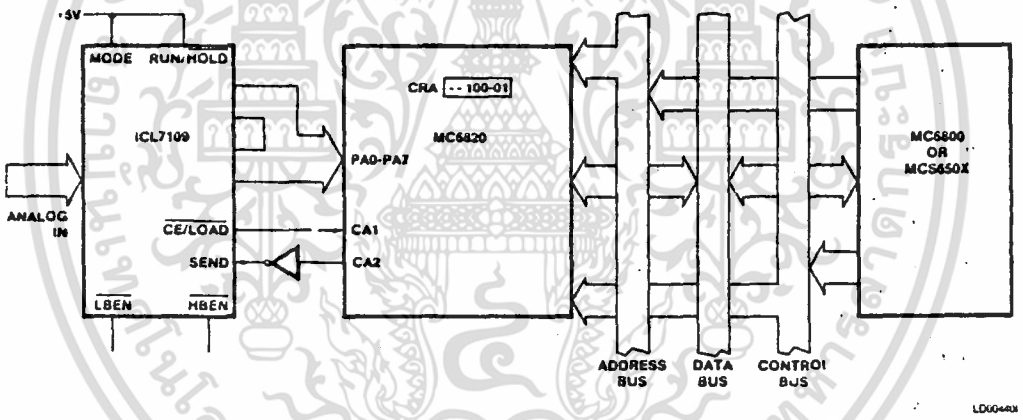


Figure 21: Handshake Interface — ICL7109 to MC6800, MCS650X

Note: All typical values have been guaranteed by characterization and are not tested.

AO49
Applying the 7109 A/D Converter

This article examines the operation and applications of the ICL7109 monolithic, CMOS, 12 bit, integrating analogue to digital converter which was introduced to the market early in 1979.

Figure 1 shows the equivalent circuit of the ICL 7109. When the RUN/$\overline{\text{HOLD}}$ 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 2. They are Auto-Zero (AZ), Signal Integrate (INT) and Deintegrate (DE).

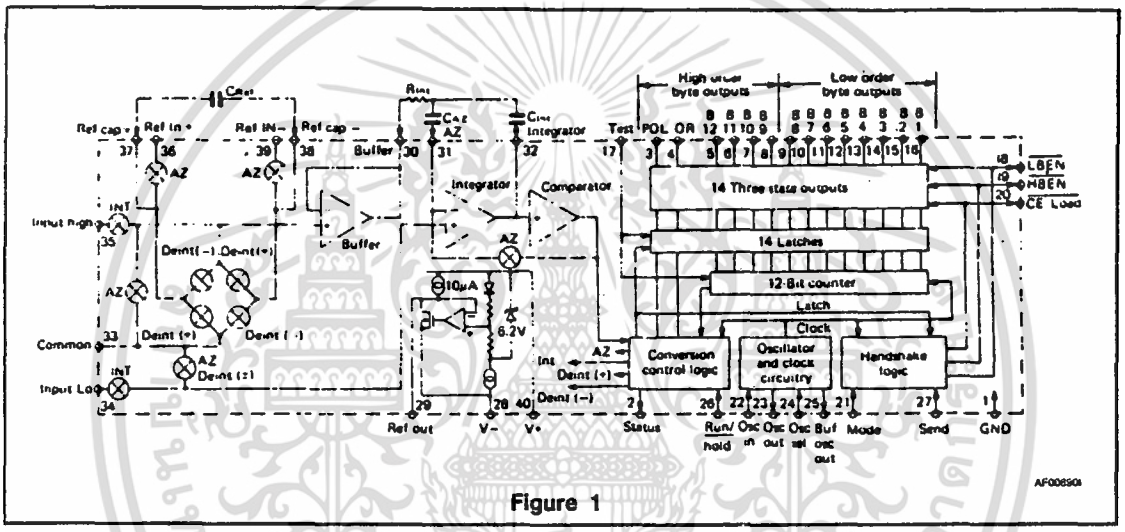


Figure 1

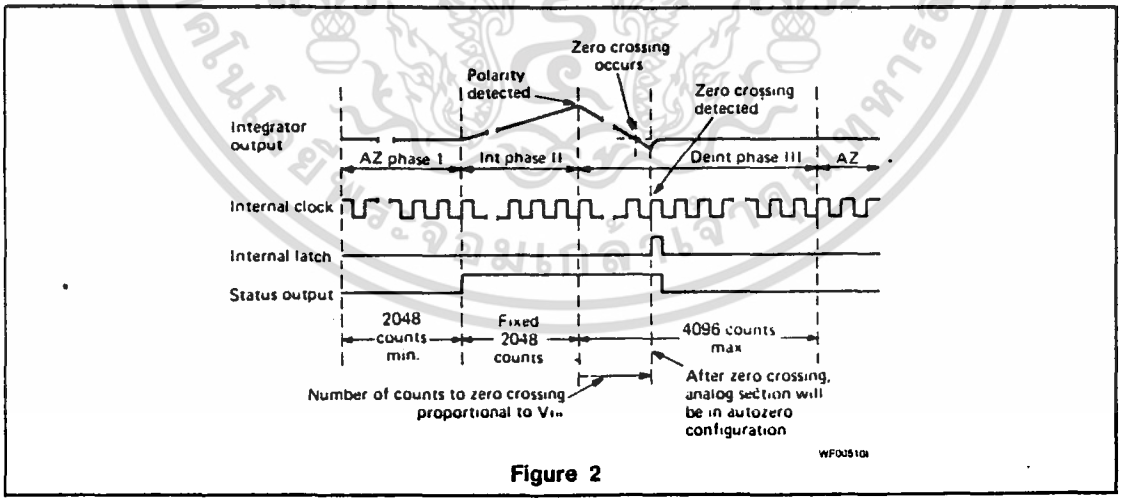


Figure 2

Auto-zero phase. During auto-zero three things happen. First, input high and low are disconnected from their pins and internally shorted to analogue 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.

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

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

DIFFERENTIAL INPUT

The input can accept differential voltages anywhere within the common mode range of the input amplifier, or specifically from 0.5V below the positive supply to 1V above the negative supply. In this range the system has a c.m.r.r. of 86dB typical. However, since the integrator also swings with the common mode voltage, care must be exercised to ensure that integrator output does not saturate. A worst case condition would be a large positive common mode voltage with a near fullscale 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 0.3V of either supply without loss of linearity.

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

DIIFERENTIAL REFERENCE

The reference voltage can be generated anywhere within the power supply voltage of the converter. The ICL7109 provides a reference output (pin 29) which may be used with a resistive divider to generate a 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. 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, an external high-quality reference should be used where the ambient temperature is not controlled or where high-accuracy absolute measurements are being made. The internal reference may then be used as a pre-regulator for an external reference, such as the ICL8069 bandgap reference diode.

DIGITAL SECTION

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

The MODE is used to control the output mode of the converter.

Direct mode. When the MODE pin is left at a low level, the data outputs (bits 1 to 8 low order byte, bits 9 to 12, polarity and overrange 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, and enables the converter to be interfaced directly, either as I/O or by memory mapping, to any microprocessor system with an 8-bit, 12-bit or 16-bit word length.

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 designed to allow a direct interface between the ICL7109 and industry-standard UARTs (such as the Intersil CMOS UARTs, IM 6402/3) with no external logic required. When triggered into the handshake mode, the ICL7109 provides all the control and flag signals necessary to sequence the two bytes of data into the UART and initiate their transmission in serial form. This greatly eases the task and reduces the cost of designing remote data acquisition stations using serial data transmission to minimise the number of lines to the central controlling processor.

Entry into the handshake mode is controlled by the MODE input. When the MODE terminal is held high, the ICL7109 will enter the handshake mode after new data has been stored in the output latches at the end of every conversion performed. The MODE terminal may also be used to trigger entry into the handshake mode on demand.

In this mode, the SEND input is connected to the UART TBRE output so that the ICL7109 can detect when the UART is ready for more data. The $\overline{CE}/\overline{LOAD}$ pin becomes an output strobe and is connected to the UART TBRL input to clock data into the UART $\overline{HBEN}$ and $\overline{LBEN}$ also become outputs which identify the high and low bytes respectively. Figure 3 shows the output sequence.

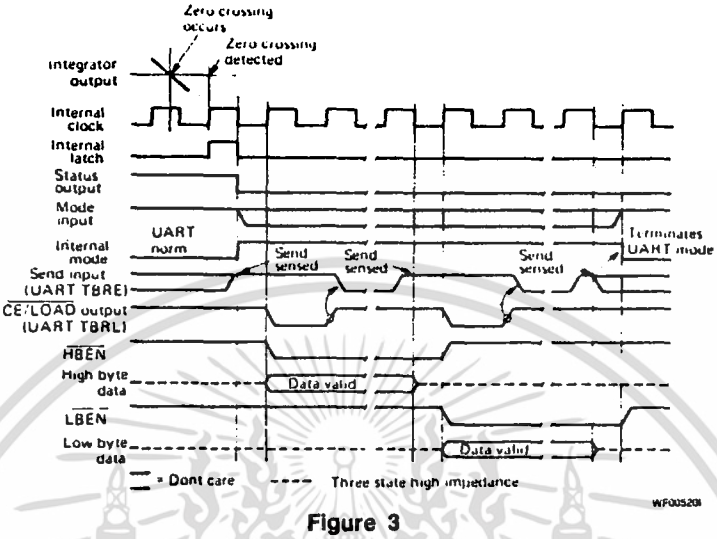


Figure 3

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

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 2 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.

When RUN/HOLD is held high or left open (it has an internal pull-up resistor) the 7109 converts continuously, taking 8192 clock cycles for each conversion. If RUN/HOLD is taken low, the current conversion is completed and the 7109 then remains in Auto Zero state until RUN/HOLD is taken high. A single, positive pulse on RUN/HOLD will cause one conversion only to take place and is a simple way of providing a "convert on command".

From the foregoing, it will be seen that the ICL7109 is a converter offering unusual flexibility in both input and output interfacing. Let us first see how flexible input interface leads to a number of applications.

BRIDGE MEASUREMENTS

The differential input of an ICL7109 lends itself to measurement of error voltages in resistance bridges, as the common mode voltage is immaterial. Moreover, because the 7109 is ratiometric, if the reference input to the chip is derived from the bridge supply, changes in the supply will not upset the output reading. A typical example is a bridge connected load cell.

Some strain gauge bridges have very low output voltages. The input noise of the ICL7109 is about 15μV peak to peak, so that if a bridge with say 5μV per count were used the output reading would jitter. This problem can be solved in two ways. The first is to use a low drift, low offset amplifier, such as the ICL7650 chopper stabilized operational amplifier, before the converter. An alternative is to have the microprocessor take the average of several conversions. (Usually 2, 4, 8 or 16 so that the division sum is a simple right shift of the binary word).

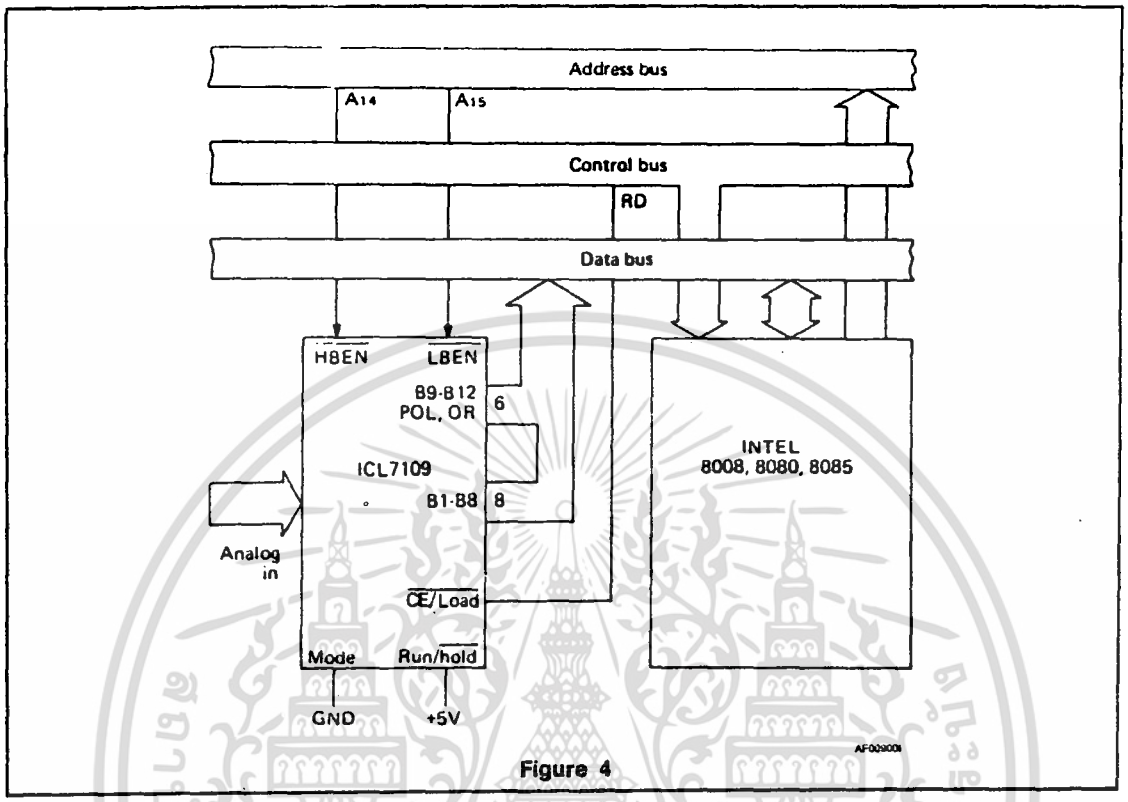


Figure 4

OFFSET ZERO MEASUREMENTS

By hooking Input Low of the 7109 to some voltage other than signal ground, measurements with an offset zero are easily possible. Consider a temperature measurement scheme using the −2mV/°C temperature coefficient of the V_{be} of a silicon transistor. By connecting one input pin to the transistor and the other to a variable voltage source the 7109 can be made to read zero at 0°C, even though the transistor V_{be} is still about 0.65V. Because the temperature coefficient of the transistor is negative, IN HI and IN LO have been interchanged so that the output goes positive for increasing temperature.

These are just a few examples of how the fully differential, fully ratiometric inputs of the ICL7109 can simplify the design of input circuitry. Now let us pass on to output configurations.

DIRECT PARALLEL PROCESSOR INTERFACES

The separately tri-state byte wide outputs of the ICL7109 make it ideal for interfacing to 8-bit microprocessor busses. Figure 4 shows a direct memory mapped interface to an 8-bit Intel 8085 processor bus. In this case linear addressing is used to select the two data bytes.

SERIAL INTERFACE

The ICL7109 is easily combined with a UART, and if necessary an analogue multiplexer, to provide a complete remote serial data acquisition station (Figure 5). The UART at the processor end of the serial link sends a digital word to the remote UART This word specifies the multiplexer address, and at the same time the remote UART Data Ready flag takes the RUN/HOLD line of the 7109 high to initiate a conversion. The appropriate input is therefore selected and converted and when conversion is complete the 7109 enters its handshake sequence, transmitting the converted data over the serial link. The HBEN output from the 7109 also clears the UART Data Ready flag so that RUN/HOLD returns low and no further conversions take place.

Optoisolators can optionally be put in the serial lines to allow for large common mode voltage differences between processor and remote station. Because all components in the remote station are low power CMOS, the generation of isolated supplies is simplified.

REPLACING V TO F CONVERTERS

A popular method of making a low cost serial interface, particularly where opto-isolation is required, is to use a V to F converter. Such an approach has its problems, as V to Fs need accurate and stable analogue components to operate correctly. They also do not generally have zero offset, scale factor drift and linearity commensurate with 12-bit accuracy applications.

The ICL7109 has a much more flexible input interface and better performance at a cost comprable with V to F converters. The STATUS output remains high for a period of

N + 2049½ clock cycles, where N is the digital reading. (See Figure 2). Therefore if the BUF OSC OUT from the 7109 is gated with STATUS a pulse train results which can be passed through an opto-isolator and counted by a processor to determine the digital reading, in much the same way as it would with a V to F converter. The processor only needs a simple software timeout loop to determine when the pulse train has ended. There are no critical timing requirements such as are encountered when using a V to F converter.

Alternatively, the STATUS and BUF OSC OUT lines can be separately opto-isolated and fed to a hardware counter system, if display of data is required. A suitable counter/display system uses the ICM7217 4 decade counter.

The 2049½ pulses are subtracted by presetting the counter to 7950, the tens complement of 2050. This technique has been used to make d.v.m.s with high voltage input isolation.

CONVERSION SPEED

The ICL7109 is an integrating (dual slope) converter and as such is not intended for very fast applications. The data sheet specifications are quoted at 7.5 conversions per second (corresponding to an internal clock frequency of about 61.5kHz, or a clock period of 16.3μs).

The speed of a dual slope converter is principally limited by the response time of the comparator, bearing in mind that the input to the comparator is a shallow ramp rather than a nice clean voltage step. In the ICL7109 the comparator gain/bandwidth product is about 200MHz, giving a delay of about 4μs. For this delay to represent a ½ count error the clock period would be 8μs, giving 15 conversions per second.

The logic of the ICL7109 is capable of operating at up to 500kHz, however, which would give about 60 conversions per second. At this speed the comparator delay will result in a zero offset. Linearity and resolution are, however, unaffected. (Remember that as the clock frequency is increased, the integrator and auto zero capacitors should be reduced in proportion). In many systems it is a simple matter for the microprocessor to subtract the zero offset.

An alternative technique is to compensate the zero offset by placing a small resistor in series with the integrator capacitor. Because the current in the integrator capacitor is constant during deintegrate, this resistor produces a "lead" on the ramp which can be calculated as follows.

Slope of the ramp, $\frac{dv}{dt} = \frac{I}{C}$

Offset voltage of ramp, $\Delta V = IR$

Time lead of ramp,

$$\Delta t = \Delta V, \frac{dt}{dV} = IR, \frac{C}{I} = RC$$

Therefore:- $R = \frac{\Delta t}{C}$, *where* $\Delta t = 4\mu s$.

By this means, the zero offset is cancelled (the system works for both input polarities). One error is being offset with another, and the two may not track with temperature, so this method is not to be relied on for wide temperature range applications.

The ICL7109 has shown itself to be one of the most versatile and cost effective A/D converters on the market, replacing existing 12-bit converters, as well as creating new applications which previously had been the domain of V to F converters and other devices.

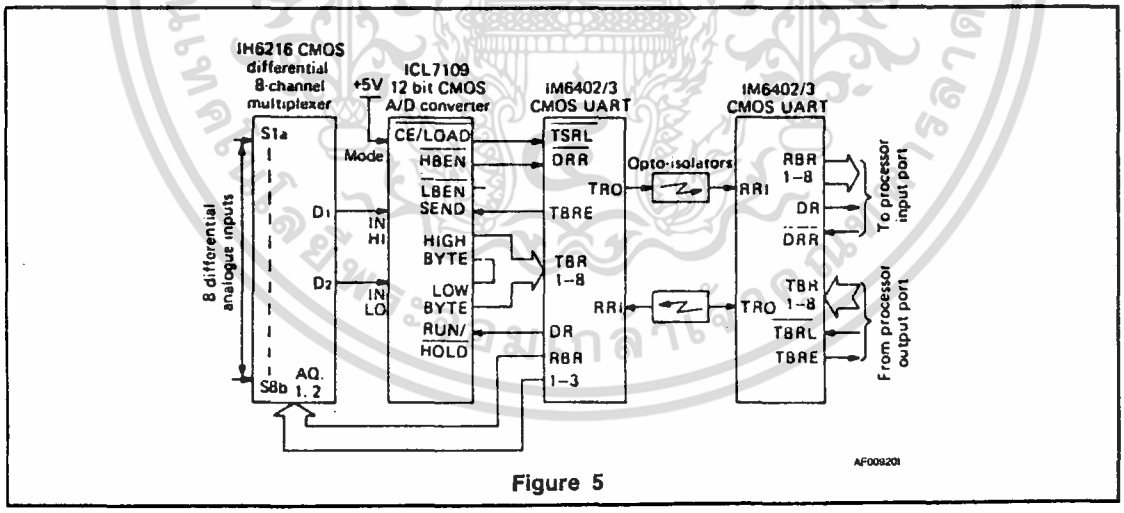


Figure 5

AO32
Understanding the Auto-Zero and Common Mode Performance of the ICL7106/7107/7109 Family

INTRODUCTION

Most of Intersil's one chip A/D converters offer differential input, differential reference and separable analog and digital ground references. The price of all this freedom, of course, is technical vigilance, and this note is intended as a defense manual against the potholes and landmines it makes accessible. The discussion is based on the ICL7106/7, but applies in large part to the ICL7116/7, the ICL7126, the ICL7109, and to a lesser extent to the ICL7135.

GENERAL DESCRIPTION

Figure 1 shows the Block Diagram of the Analog Section for the ICL7106 and 7107. Each measurement cycle is divided into three phases. They are (1) auto-zero (A-Z), (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 the 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 A-Z accuracy is limited only by the noise of the system. In any case, the offset referred to the input is less than 10μV.

Signal Integrate Phase

During signal INTegrate, the auto-zero loop is opened, the internal short is removed, and the internal input high and low are connected to the external pins. The converter then integrates the differential voltage between INHI and INLO for a fixed time. This differential voltage can be within a wide common mode range — within one volt of either supply. If, on the other hand, the input signal has no return with respect to the converter power supply, INLO can be tied to analog COMMON to establish the correct common-mode voltage. 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 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. The time required for the output to return to zero is proportional to the input signal. Specifically the digital reading displayed is

$$1000\left(\frac{V_{in}}{V_{ref}}\right).$$

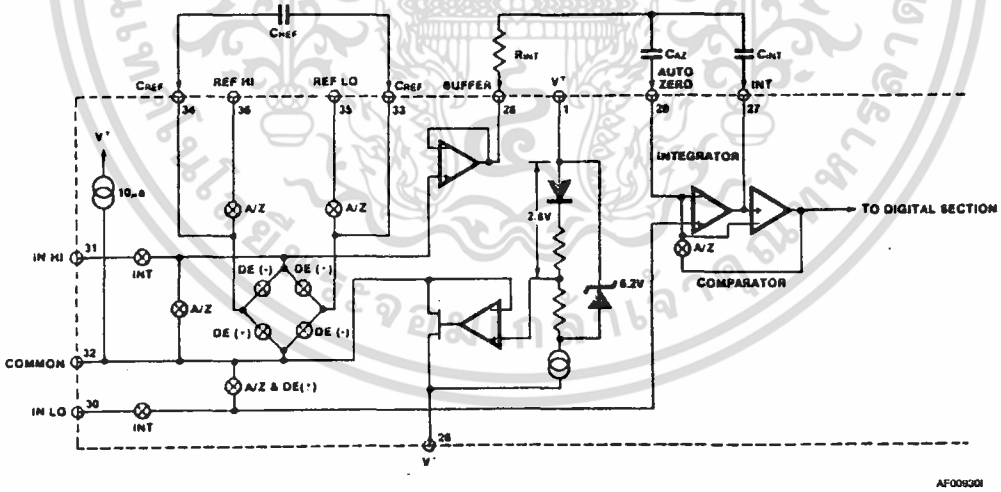


Figure 1: Analog Section of 7106/7107

407025-001

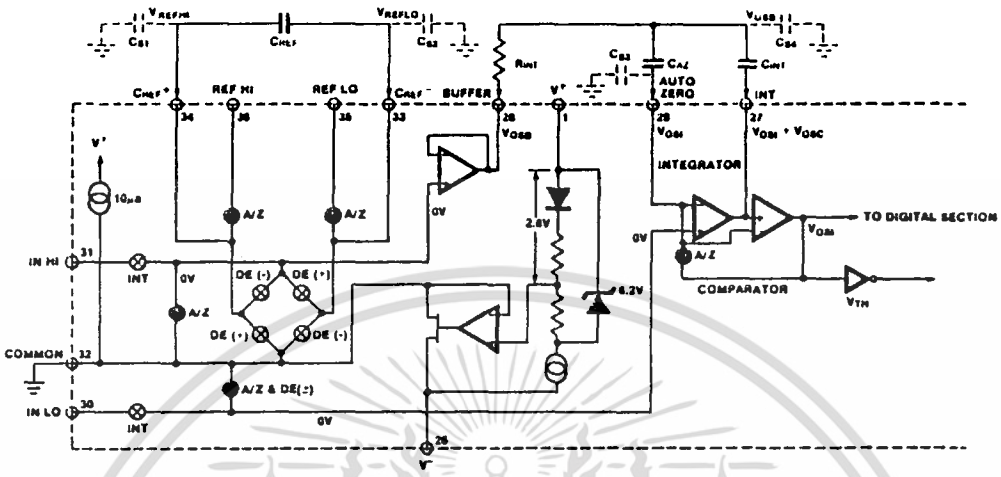


Figure 2: Auto-Zero Phase, with Offset Voltages and Stray Capacitances

CMRR AND COMMON MODE VOLTAGE EFFECTS

There are three basic voltages applied to the ICL7106/7, etc. which can give "common mode voltage" consequences. These are indicated in Figures 2, 3, and 4 which show the analog section in the phases described above. The choices are 1 of reference voltage source to COMMON, 2) of input voltage source to COMMON, and 3) of COMMON to (digital) supply voltage.

During Auto-Zero, the outputs of the buffer, integrator, and comparator are all within various offset voltages of analog COMMON. These are marked on Figure 2, which shows the Auto-Zero phase. For the remainder of the discussion, these offset voltages will be ignored, since they are merely added to other voltage changes described. The non-inverting inputs of the buffer and integrator are also tied to analog COMMON, so it is convenient to describe all these voltages with respect to COMMON.

Reference Common Mode Voltage to COMMON

The reference capacitor is recharged during the Auto-Zero time; the stray capacitance shown in Figure 2 as CS1 and CS2 will also be charged. During DE-integrate (Figure 4) the reference capacitor is switched so that one or the other of its terminals is at analog COMMON. This will cause charge-sharing with the stray capacitances on the other terminal. In particular, a common mode voltage on the reference input (with respect to COMMON) will give a roll-over error, since the effective DE-integrate reference will be higher in one polarity than the other. The ideal here is for $(V_{REFHI} + V_{REFLO}) = 2\ V_{ANCOM}$, at least for equal stray capacitances, but this is inconvenient in most applications. The roll-over error contribution at full scale (ignoring a second order term) is

$$2000\ \frac{(V_{REFLO}\ C_{S1} + V_{REFHI}\ C_{S2})}{V_{REF}\ C_{REF}} \simeq$$

$$2000\ \frac{V_{CM}\ (C_{S1} + C_{S2})}{V_{REF}\ C_{REF}}\ \text{(counts)}$$

For $C_{REF} = 0.1\ \mu F$, $C_S = 15pF$, $V_{CM}/V_{REF} = 10$, this can give two counts of error, but if $V_{REFLO} = 0$, and C_{S2} is 5pF, the error is 0.1 counts, lost in the noise level. In the latter case (a very common application condition) C_{S1} does not contribute any errors, so putting the "outside foil" of the reference capacitor to this side will minimize roll-over. Also increasing C_{REF} (without corresponding increases in C_S) will reduce rollover. Note that stray capacitance to the buffer output is also unimportant if either REFHI or REFLO is at COMMON.

Input Voltage to COMMON

First, the direct CMRR of the buffer and integrator op amps will themselves lead to a scale factor error and an offset if INLO is not at analog COMMON. Higher order CMRR terms are generally negligible, and this first order term is very small for most devices. It can be adjusted out in most applications with a reference voltage adjustment. More serious is the effect of stray capacitance to ground of the integrating and auto-zero capacitors, and the AZ pin, C_{S4} and C_{S3} in Figure 2. The AZ pin will swing from COMMON to INLO (Figure 3) and C_{S3} will have to be charged through C_{AZ}, giving an error voltage on C_{AZ}, during the integrate phase, of:

$$\Delta\ V_{AZ} = V_{INLO}\ \frac{C_{S3}}{C_{AZ}}$$

This acts as an offset voltage referred to the input, and is most serious for small ratios of full-scale input voltage to common mode voltage: For $C_{AZ} = 0.47\ \mu F$, $C_{S3} = 10pF$, $V_{INLO} = 2V$, the offset will be $40\mu V$, or 0.4 counts for 200mV full scale input. This charge is recovered in the transition back to COMMON for DE-integrate (Figure 4), so the offset is not continued in this phase. The same charge, together with that due to C_{S4}, also flows through the integrating capacitor. Ignoring second-order terms, the error

voltage on the integrator output during the integrate phase will be:

$$\Delta V_{INT} = V_{INLO}\left(\frac{C_{S3}}{C_{AZ}} + \frac{C_{S3}+C_{S4}}{C_{INT}}\right)$$

However, the charge transferred through the integrating capacitor is also recovered for DE-integrate and does not cause any errors, except for inputs near zero. The decision as to the polarity of the input signal, and the required DE-integrate reference polarity is made precisely at the end of the integrate phase, and for small input signals the error charge on the integrating capacitor due to charging strays can exceed the true signal charge, leading to an incorrect choice of polarity. Thus, the return of the error charge at the beginning of DE-integrate will lead at once to a zero crossing and a result of zero with the wrong polarity. This shows up as a series of readings such as (in a bad case) −4, −3, +0, +0, +0, +0, +1, +2, +3, +4 for INLO negative (Figure 5). The magnitude of this effect is dependent on the full scale integrator swing, and is given by:

$$\Delta \text{ pol.} = 2000\left(\frac{V_{INLO}}{V_{INTFS}}\right)\left(\frac{C_{S3}}{C_{AZ}} + \frac{C_{S3}+C_{S4}}{C_{INT}}\right)\text{(in counts)}$$

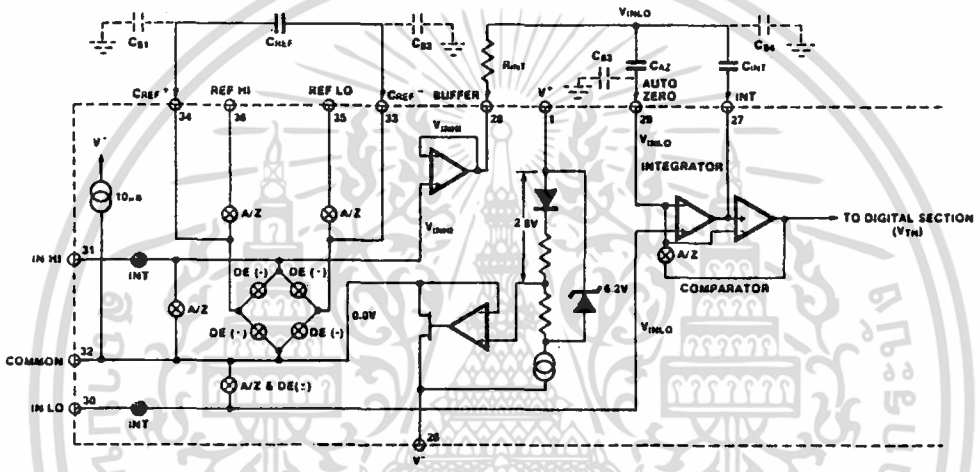


Figure 3: Integrate Phase. All voltages shown with respect to COMMON

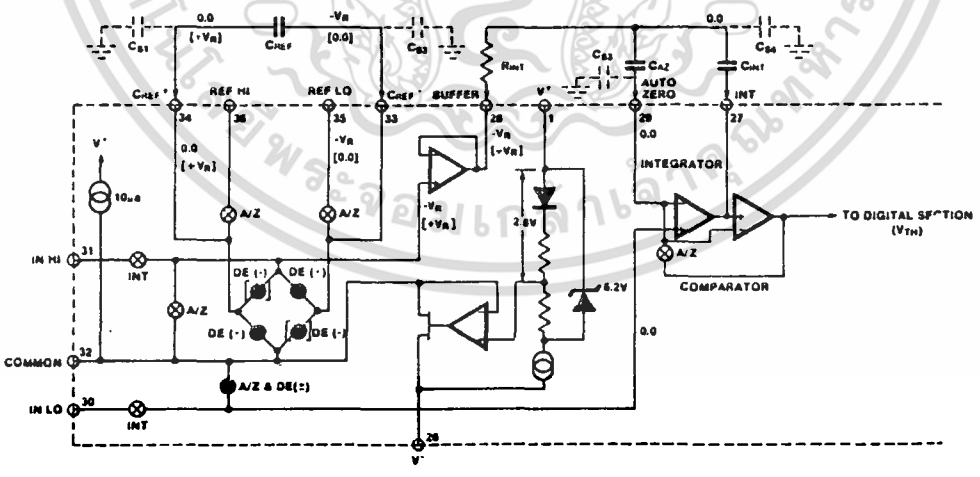


Figure 4: DE-integrate Phase. Voltages shown with respect to COMMON, for positive input. ( ) shows values for negative input

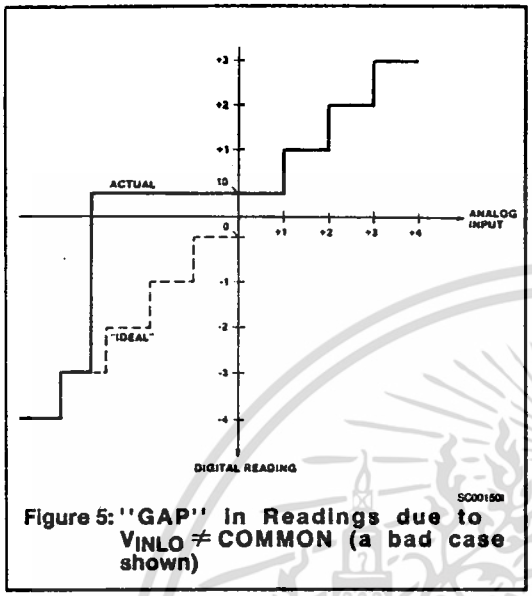


Figure 5: "GAP" in Readings due to $V_{INLO} \neq$ COMMON (a bad case shown)

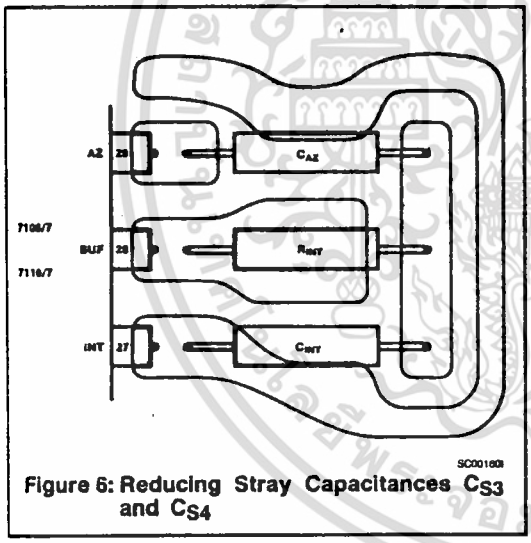


Figure 6: Reducing Stray Capacitances C_{S3} and C_{S4}

For $C_{INT} = .22\mu F$, $C_{S4} = 10pF$, $V_{INTFS} = +2V$, and other values as before, this amounts to about 0.23 counts, but for $C_{AZ} = 0.047\mu F$ (recommended for 2.0V F.S.) Δ pol is 0.6 counts. A small increase in stray capacitance or reduction of integrator swing will give a significant "gap" in the readings, as shown in Figure 5. This effect, the only one causing significant nonlinearity, can be reduced by guarding the integrating and auto-zero capacitors and resistor with either BUFFer out or INTegrator out pins in so far as possible. This can readily be done on a PC board by simple extension of the traces leading from those pins to the three components, as suggested in Figure 6. Note that excessive capacitance across R_{INT} will increase the width of the zero reading (see A017 for a discussion of a similar effect), while capacitance across C_{AZ} and C_{INT} has no effect on the circuit.

Analog COMMON to Digital Supply Voltages

The COMMON line on the ICL7106/7 family of devices provides a convenient ground-return point in many applications; particularly with floating (battery) supplies. However, in a fixed supply environment, improved integrator swing (improving many system parameters) can be achieved if COMMON is pulled more negative, and the circuit has been set up to allow this. The effects described above are all independent of the actual level of COMMON, but the next one is not!

The DE-integrate phase should ideally terminate when the output of the comparator returns to the value it had during Auto-Zero (analog COMMON), but will actually terminate when the output passes through the logic threshold of the zero-crossing gate and flip-flop combination. The "free" analog COMMON voltage is very close to the logic threshold, giving a negligible error, but if analog COMMON is pulled negative, zero crossing will be detected late for positive inputs, (reading high) and early for negative inputs (reading low), this leads to a (positive) offset. The polarity detection sees the same influence, so no nonlinearity results. The magnitude of this offset depends on comparator gain (typically 7-8K, but as low as 3K in some devices) and F.S. integrator swing;

$$\text{OFFSET (COMMON)} = 2000 \left| \frac{V_{COMMON} - V_{TH}}{V_{INTFS}\ A_{AVCOMP}} \right|$$

With a 2V swing and 3K gain, this will contribute 1/3 count per COMMON negative volt. This can be used to measure comparator gain, at least with moderate accuracy, which is otherwise hard to do. Obviously, the offset can be minimized by maximizing the integrator swing. The comparator gain varies from device to device, and is limited also by the need to keep the comparator fast. Various improvements in this gain have been made, and will probably continue to be made in the future, but this offset should be considered carefully if COMMON is to be moved away from its "free" location, or if the logic supplies are altered.

The Auto-Zero Loop Residual

During the Auto-Zero phase, the converter self-corrects for all the offset voltages in the buffer, integrator, and comparator.

This section covers a normally undetectable, but under some circumstances significant, error generated in the auto-zero system. A similar effect which occurs in the 2-chip systems has been discussed previously (see A030, Appendix A), but the details and remedies are sufficiently different to warrant a separate discussion.

The relevant circuit to be discussed is shown in Figure 7 and the major cycle waveforms in Figure 8. Let us first assume that the prior auto-zero cycle has been indefinitely long, or is otherwise ideal, so that the conversion starts with no residual error on the auto-zero capacitor. The integrate and DE-integrate cycles will be classically perfect to the point at which a zero-crossing actually occurs (at the output of the integrator). However, from this point two delays occur; first the comparator output is delayed (due to comparator delay) and secondly the zero-crossing is not registered until the next appropriate clock edge. (For further

discussion of this, see Application Note A017). At this point, the circuit is returned to the auto-zero connection (logic and switch delays may be absorbed in comparator delay as far as our discussion is concerned). The net result is that the integrator output voltage will have passed the zero-crossing point by an amount given by

$$V_{Ires} = \pm V_{IFS}\left(\frac{c_D + c_X}{c_{FS}}\right)$$

where $0 \le c_X \le 1$ is the variable delay, c_D is the fixed delay, c_{FS} is the full scale count in units of clock pulse periods, and V_{IFS} is the full scale integrator swing in volts.

Note: In all subsequent discussions, "C" indicates a capacitor, while "c" denotes a number (not necessarily an integer) of counts.

The range of this residual voltage corresponds to the integrator swing per count, and is independent of input value, except for polarity.

Note, however, that we have assumed a zero-crossing actually occurred. If the input is overloaded (past full scale), DE-integrate will terminate with a substantial residual voltage remaining on the integrator capacitor. The maximum value of this residual depends on the total possible swings of buffer and integrator, as compared to the "full scale" values used. In general, we may treat this case as corresponding to a large negative value of c_X.

The immediate effect of closing the auto-zero loop may be seen by examining Figure 7. We may consider the comparator as acting as an op-amp. Under these conditions: the voltage across the auto-zero impedance is high, and the (nonlinear) impedance is low; on the other hand, the initial voltage across the integrating resistor is zero.

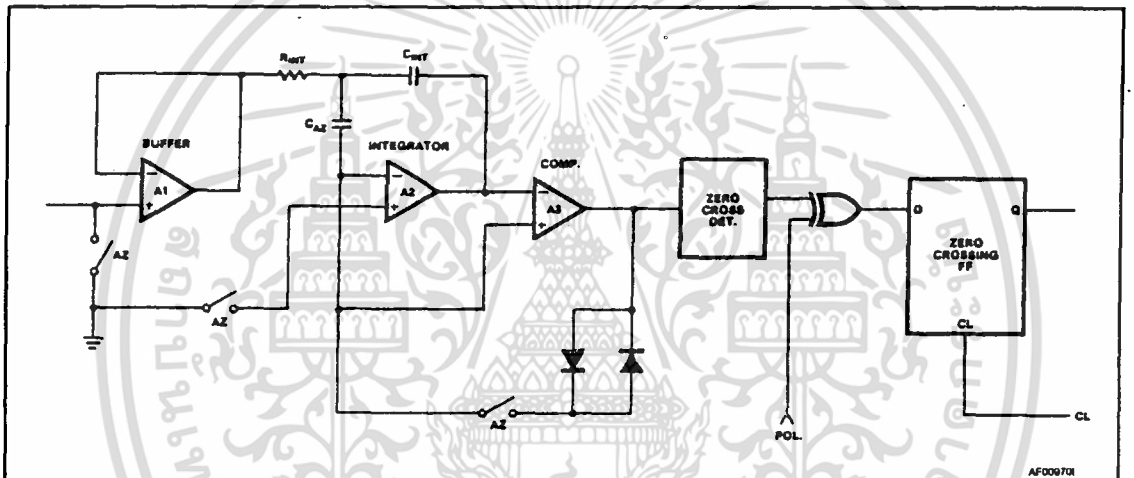


Figure 7: The Analog System (simplified to show only Auto-Zero connections)

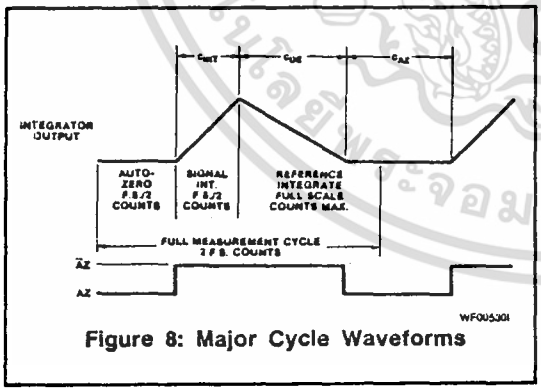


Figure 8: Major Cycle Waveforms

Thus, the auto-zero capacitor will be charged rapidly to exactly cancel the residual voltage, as shown in Figure 9. The output of the integrator is now at the correct position, but the auto-zero and integrator capacitors have shared the original error. The junction point of the two capacitors and resistor has been moved by a portion of the original residual voltage, given by:

$$V_{AZI} = V_{Ires}\left(\frac{C_{INT}}{C_{AZ} + C_{INT}}\right) \qquad (4.1)$$

This voltage will decay with a time constant controlled by the integrating resistor and the two capacitors, while the auto-zero capacitor is easily kept in step owing to the high comparator gain. Thus, at the end of the auto-zero time, $t_{AZ} = c_{AZ}\ t_{cp}$., the residual will be reduced to:

$$V_{AZres} = V_{AZI}\ \exp\left(\frac{(-c_{AZ})\ (t_{cp})}{R_{INT}(C_{INT} + C_{AZ})}\right)$$

$$= V_{Ires}\left(\frac{C_{INT}}{C_{AZ} + C_{INT}}\right)\exp\left(\frac{(-c_{AZ})\ (t_{cp})}{R_{INT}\ (C_{INT} + C_{AZ})}\right)$$

For the residual left after a zero-crossing, we may further refine this to:

$$V_{AZres} = V_{IFS}\left(\frac{c_X + c_D}{CFS}\right)\frac{C_{INT}}{(C_{AZ} + C_{INT})} \qquad (4.2)$$

$$\exp\left(\frac{-c_{AZ}\, t_{cp}}{R_{INT}\,(C_{INT} + C_{AZ})}\right)$$

For the overrange case, we may again assume a large negative c_X value.

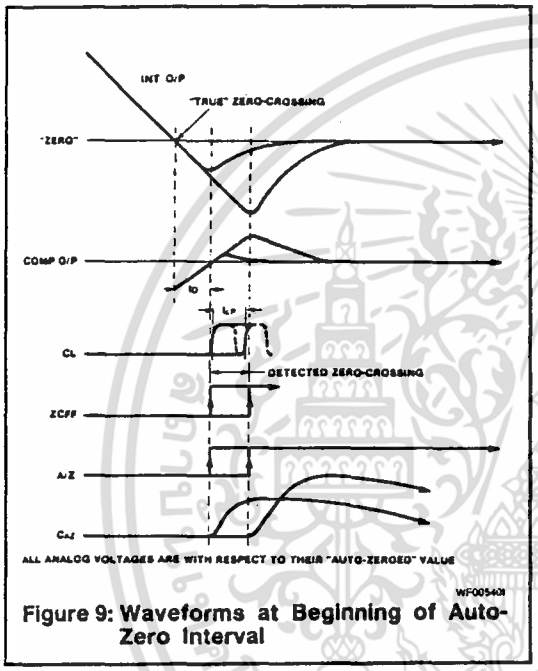


Figure 9: Waveforms at Beginning of Auto-Zero Interval

Now $R_{INT}\, C_{INT}$ is controlled by the buffer swing, V_{BFS}, the integrator swing, V_{IFS}, and the integration time $t_{INT} = c_{INT}\, t_{cp}$, so that

$$V_{BFS}/R_{INT} \cdot t_{INT} = C_{INT}\, V_{IFS},$$

$$\text{or } R_{INT}\, C_{INT} = c_{INT}\frac{V_{BFS}}{V_{IFS}}\, t_{cp}$$

Also we may write $C_{AZ} = a\, C_{INT}$, for convenience, and will then obtain

$$V_{AZres} = V_{Ires}\left(\frac{1}{1 + a}\right)\exp\left(\frac{-c_{AZ}\, V_{IFS}}{c_{INT}\, V_{BFS}\,(1 + a)}\right) \qquad (4.3)$$

This residual voltage on the auto-zero capacitor effectively increases the magnitude of the input voltage as seen on the output of the buffer. Thus, converting this voltage to count-equivalents.

$$c_{AZres} = \frac{V_{AZres}}{V_{BFS}} \cdot c_{FS} = \frac{V_{IFS}}{V_{BFS}}\frac{(c_X + c_D)}{(1 + a)} \qquad (4.4)$$

$$\exp\left(-\frac{c_{AZ}}{c_{INT}} \cdot \frac{V_{IFS}}{V_{BFS}} \cdot \frac{1}{1 + a}\right)$$

Since this voltage also subtracts from the reference, its effect at the input is magnified in the ratio

$$c_{INres} = c_{AZres}\left(\frac{c_{INT} + c_{DE}}{c_{INT}}\right) \text{ so that}$$

$$c_{INres} = \left(1 + \frac{c_{DE}}{c_{INT}}\right)\left(\frac{V_{IFS}}{V_{BFS}}\right)\frac{(c_X + c_D)}{(1 + a)} \qquad (4.5)$$

$$\exp\left(-\frac{c_{AZ}}{c_{INT}} \cdot \frac{V_{IFS}}{V_{BFS}} \cdot \frac{1}{1 + a}\right)$$

Note that c_{DE} is equal to the displayed result, except for overrange conditions, when it is equal to c_{FS} and the first bracket becomes 3. Also, $c_{AZ} = c_{INT}$; and this expression, so substituted, determines the overrange residual performance.

For the normal in-range condition, two things should be noted here. First, this residual acts to increase the input voltage magnitude, and secondly, a small increase in input voltage tends to decrease the magnitude of the residual until the result count changes. These effects lead to "stickyness" in the readings; suppose, in a noise-free system, that the input voltage is at a level where the residual's a minimum, the detected zero-crossing follows the true one as closely as possible. A minute increase in input voltage will cause the zero-crossing to be detected one pulse later and the residual to jump to its maximum value. The effect of this is a small increase in the apparent input voltage; thus if we now remove the minute increase, the residual voltage effect will maintain the new higher reading; in fact we will have to reduce the input voltage by an amount commensurate with the effective residual voltage to force the reading to drop back to the lower value. In more detail, we should consider the equilibrium conditions on the auto-zero capacitor. Clearly, the voltage added at the end of reference integrate must just balance that which decays away during the auto-zero interval. So far the relationships we have developed have assumed a zero residual before the conversion, but in the equilibrium condition the residual given by equation (4.4) remains, and at the end of conversion, the new amount, given by equation (4.1), is added to this, so we start the "auto-zero decay" interval with

$$c_{AZI} = c_{AZres} + \frac{V_{IFS}}{V_{BFS}}\frac{(c_X + c_D)}{(1 + a)} \qquad (4.6)$$

By combining equations (4.4) and (4.6) we find, for the equilibrium condition,

$$c_{AZres} = \pm\frac{V_{IFS}}{V_{BFS}}\frac{(c_X + c_D)}{(1 + a)}$$

$$\left[\exp\left(+\frac{c_{AZ} V_{IFS}}{c_{INT} V_{BFS}\,(1 + a)}\right) - 1\right]^{-1}.$$

Once again, the effect of this at the input is multiplied by the ratio of total input integrate times, so that, under equilibrium conditions,

$$C_{INres} = \pm\frac{V_{IFS}}{V_{BFS}}\left(1 + \frac{c_{DE}}{c_{INT}}\right)\frac{(c_X + c_D)}{(1 + a)} \qquad (4.7)$$

$$\left[\exp\left(\frac{C_{AZ}V_{IFS}}{C_{INT}\ V_{BFS}(1+\alpha)}\right)-1\right]^{-1}$$

Those expert at skipping to the end of the difficult bit will recognize that as the final equation, in terms of complexity. So let us now see what it means. Clearly, the error term is greater, the larger C_{DE}/C_{INT}, and the smaller C_{AZ}/C_{INT}. For the devices considered here (except some applications of the ICL7109) these are both worst case near full scale input, where $C_{DE}/C_{INT} \approx 2$ and $C_{AZ}/C_{INT} \approx 1$.

Substituting these, we find the worst case

$$C_{INres} \simeq \pm\frac{V_{IFS}}{V_{BFS}}(3)\frac{(C_X+C_D)}{(1+\alpha)}\left[\exp\left(\frac{V_{IFS}}{V_{BFS}(1+\alpha)}\right)-1\right]^{-1} \quad (4.8)$$

Recall that C_D is fixed; and C_X must be between 0 and 1. The expression is now a function purely of the ratio of integrator and buffer full scale swings, and the ratio of auto-zero and integrator capacitors, α. The effect of the latter ratio is mixed; a larger value reduces the initial error, but increases the time — constant for its decay. The relationship is plotted in Figure 10 and shows the desirability of keeping the integrator swing higher than the buffer swing. Note also that a lower capacitance ratio α always improves the residual. However, both noise and the common-mode effects discussed in Section 3 above require a large auto-zero capacitor, and a compromise must be reached. In general, if the full scale input is small, a large C_{AZ} is needed, but for larger full scale inputs, a smaller value is best. Note also that the comparator delay (C_D in equation (4.8)) is also effectively enhanced. This has the effect of shrinking the zero somewhat more than normally occurs. Since this term changes sign with polarity, the converter will have a tendency to keep the current sign at zero input.

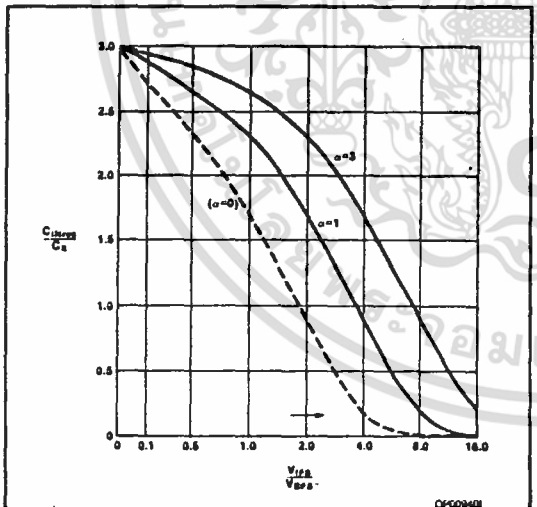


Figure 10: Auto-Zero Loop Residual vs. Integrator/Buffer Swing and Capacitor Ratios (worst case)

The effects of noise should be mentioned here. The worst case value of residual shown in Figure 10 assumes a very gradual approach to equilibrium, and any noise spike causing the reading to flash to the next value will destroy this carefully established residual value! Thus, for any system with noise of 1/3 count or more, the effect is greatly reduced, and even 1/10 count of noise will restrict the actual hysteresis value found in practice. The detailed analysis of the auto-zero residual problem in the presence of appreciable noise is left as an exercise for the masochist.

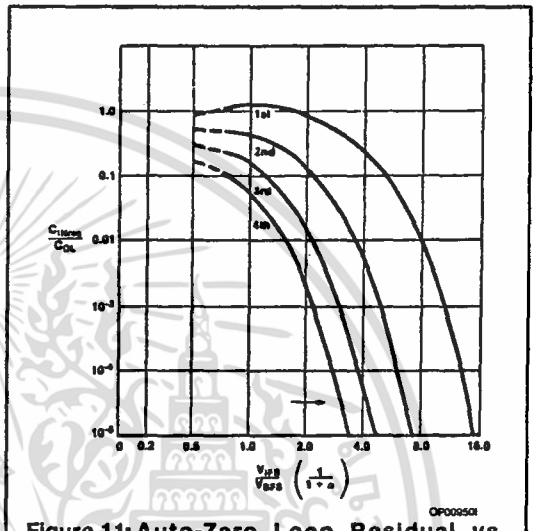


Figure 11: Auto-Zero Loop Residual vs. Integrator/Buffer Swing for Overload and Capacitor Ratio

For overrange conditions, the controlling equation is (4.5) with the appropriate substitutions for the count ratios. Specifically, putting C_{OR} for the count-equivalent value of the overrange above full scale, and ignoring C_D, we obtain:

$$C_{INres} = 3\left(\frac{V_{IFS}}{V_{BFS}}\right)\left(\frac{C_{OR}}{1+\alpha}\right)\exp\left[-\frac{V_{IFS}}{V_{BFS}}\cdot\frac{1}{1+\alpha}\right] \quad (4.9)$$

This is plotted in Figure 11, and shows the very strong dependence on the integrator to buffer swing ratio. The direction is the opposite of that for the post-zero-crossing residual, as well as being normally much larger. A positive overrange on one reading will tend to make the next reading(s) too negative, and vice versa. The influence on second and even subsequent readings after an overrange can also be appreciable in some cases. The miss-charge trapped on the auto-zero capacitor during the first conversion after an overrange will still be there at the end. If this conversion is in-range, we may ignore C_X and C_D and just consider the continuation of the exponential decay during the following Auto-Zero phase: Thus, at the beginning of the second conversion, the residual will have been reduced to:

$$C_{AZres\ 2} = C_{AZres}\ \exp\left[-\frac{C_{AZ}}{C_{INT}}\cdot\frac{V_{IFS}}{V_{BFS}}\cdot\frac{1}{1+\alpha}\right]$$

where C_{AZres} is given by equation (4.4). This will be similar for subsequent in-range conversions. The effect at the input is again increased by the time ratio of DE-integrate and INTegrate, and so we may write, for the effective error at the input on the nth conversion after the overrange:

$$C_{INresn} = \left(1 + \frac{C_{DE}}{C_{INT}}\right)\left(\frac{V_{IFS}}{V_{BFS}}\right)\left(\frac{C_{OR}}{1+\alpha}\right)\left[\exp\left(-\frac{C_{AZ}}{C_{INT}} \cdot \frac{V_{IFS}}{V_{BFS}} \cdot \frac{1}{1+\alpha}\right)\right]^n \quad (4.10)$$

This also is plotted in Figure 11, for various values of n against $\frac{V_{IFS}}{V_{BFS}}\frac{1}{1+\alpha}$, for worst case conditions (an overload followed by several full scale conversions).

The residual can be reduced for devices, such as the ICL7109, which provide indications of overrange conversions and auto-zero phase (OR and STATUS in the ICL7109) by reducing the integrator time constant during all or part of the Auto-Zero phase after an overrange conversion. This can be done by shorting out all or part of the integrating resistor R_{INT} by a suitable analog switch. A circuit to do this is shown in Figure 12 for the ICL7109. Care should be taken to ensure that the switch does not cause errors due to charge injection into the capacitors when going OFF. Alternatively the clock can be slowed down or stopped, or Run/$\overline{\text{Hold}}$ used to extend the Auto-Zero phase under the same conditions. These techniques are much harder to apply to devices such as the ICL7106/7 which do not provide the necessary signals. Generally, however, these devices are not used in multiplexing applications.

SUMMARY

This note has described the most common behavior patterns that cause concern and/or confusion among users of the ICL7106/7 and similar products, and their origins. Hopefully, it will help alleviate or eliminate any consequent applications problems with this family of devices. Naturally, some parts will not show all of the effects; for instance, the ICL7116/7 and ICL7135 cannot suffer from large common-mode voltages between reference and COMMON, because no such voltage can be applied, and the ICL7135 has a modified auto-zero sequence that alters the residual effects in the Auto-Zero Loop Residual Section.

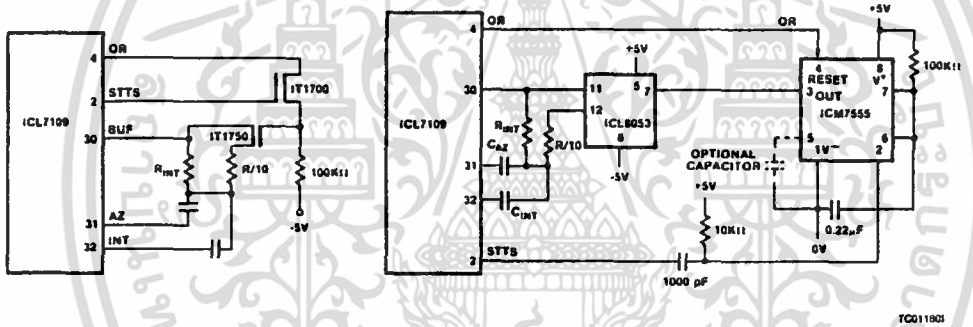


Figure 12: Circuits to Reduce Overrange Residual on ICL7109

ICL7660
CMOS Voltage Converter

INTERSIL

GENERAL DESCRIPTION

The Intersil ICL7660 is a monolithic CMOS power supply circuit which offers unique performance advantages over previously available devices. The ICL7660 performs supply voltage conversion from positive to negative for an input range of +1.5V to +10.0V, resulting in complementary output voltages of −1.5V to −10.0V. Only 2 non-critical external capacitors are needed for the charge pump and charge reservoir functions. The ICL7660 can also be connected to function as a voltage doubler and will generate output voltages up to +18.6V with a +10V input. Note that an additional diode is required for V_{SUPPLY} > 6.5V.

Contained on chip are a series DC power supply regulator, RC oscillator, voltage level translator, and four output power MOS switches. A unique logic element senses the most negative voltage in the device and ensures that the output N-channel switch source-substrate junctions are not forward biased. This assures latchup free operation.

The oscillator, when unloaded, oscillates at a nominal frequency of 10kHz for an input supply voltage of 5.0 volts. This frequency can be lowered by the addition of an external capacitor to the "OSC" terminal, or the oscillator may be overdriven by an external clock.

The "LV" terminal may be tied to GROUND to bypass the internal series regulator and improve low voltage (LV) operation. At medium to high voltages (+3.5 to +10.0 volts), the LV pin is left floating to prevent device latchup.

FEATURES

- **Simple Conversion of +5V Logic Supply to ±5V Supplies**
- **Simple Voltage Multiplication (V_{OUT} = (−) nV_{IN}).**
- **99.9% Typical Open Circuit Voltage Conversion Efficiency**
- **98% Typical Power Efficiency**
- **Wide Operating Voltage Range 1.5V to 10.0V**
- **Easy to Use — Requires Only 2 External Non-Critical Passive Components**

APPLICATIONS

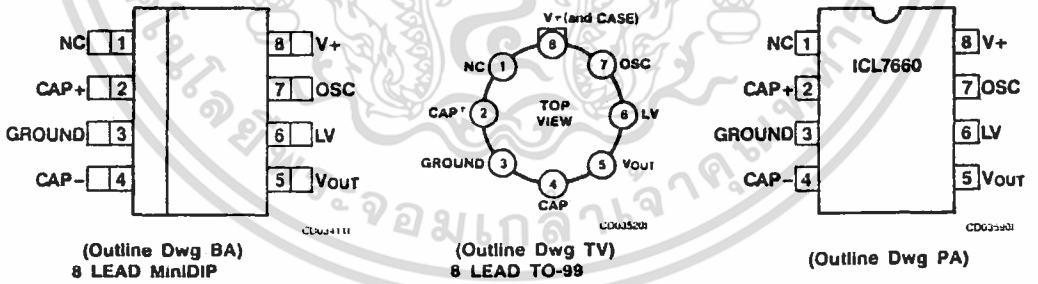
- **On Board Negative Supply for Dynamic RAMs**
- **Localized μ-Processor (8080 Type) Negative Supplies**
- **Inexpensive Negative Supplies**
- **Data Acquisition Systems**

ORDERING INFORMATION

PART NUMBER	TEMP. RANGE	PACKAGE
ICL7660CTV	0° to +70°C	TO-99
ICL7660CBA	0°C to +70°C	8 PIN SOIC
ICL7660CPA	0° to +70°C	8 PIN MINI DIP
ICL7660MTV*	−55° to +125°C	TO-99
ICL7660/D	–	DICE**

*Add /883B to part number if 883B processing is required.

**Parameter min/max limits guaranteed at 25°C only for DICE orders.



(Outline Dwg BA)
8 LEAD MiniDIP

(Outline Dwg TV)
8 LEAD TO-99

(Outline Dwg PA)

Figure 1: Pin Configurations

302063-003

Note. All typical values have been guaranteed by characterization and are not tested.

ICL7660

INTERSIL

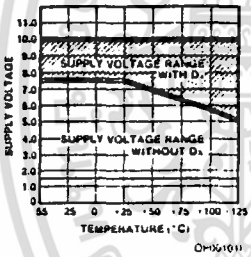
OPERATING CHARACTERISTICS (CONT.)

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYP	MAX	
R_{OUT}	Output Source Resistance	I_{OUT} = 20mA, T_A = 25°C		55	100	Ω
		I_{OUT} = 20mA, 0°C ≤ T_A ≤ +70°C			120	Ω
		I_{OUT} = 20mA, −55°C ≤ T_A ≤ +125°C (Note 3)			150	Ω
		V^+ = 2V, I_{OUT} = 3mA, LV to GROUND 0°C ≤ T_A ≤ +70°C			300	Ω
		V^+ = 2V, I_{OUT} = 3mA, LV to GROUND, −55°C ≤ T_A ≤ +125°C, D_X in circuit (Note 3)			400	Ω
f_{OSC}	Oscillator Frequency			10		kHz
P_{Ef}	Power Efficiency	R_L = 5kΩ	95	98		%
$V_{OUT\ Ef}$	Voltage Conversion Efficiency	R_L = ∞	97	99.9		%
Z_{OSC}	Oscillator Impedance	V^+ = 2 Volts		1.0		MΩ
		V = 5 Volts		100		kΩ

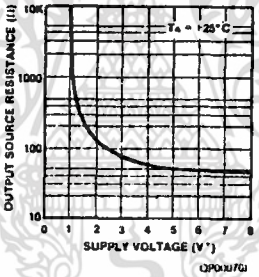
Notes: 1. Connecting any input terminal to voltages greater than V+ or less than GROUND may cause destructive latchup. It is recommended that no inputs from sources operating from external supplies be applied prior to "power up" of the ICL7660.
2. Derate linearly above 50°C by 5.5mW/°C.
3. ICL7660M only.

TYPICAL PERFORMANCE CHARACTERISTICS (Circuit of Figure 3)

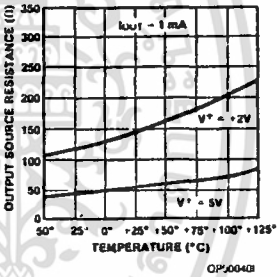
OPERATING VOLTAGE AS A FUNCTION OF TEMPERATURE



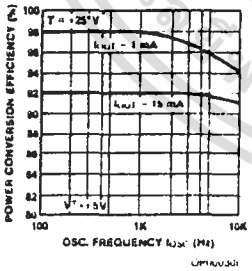
OUTPUT SOURCE RESISTANCE AS A FUNCTION OF SUPPLY VOLTAGE



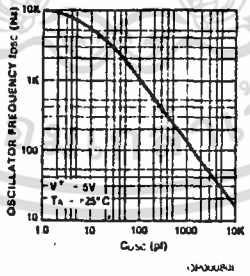
OUTPUT SOURCE RESISTANCE AS A FUNCTION OF TEMPERATURE



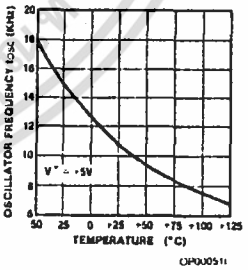
POWER CONVERSION EFFICIENCY AS A FUNCTION OF OSC. FREQUENCY



FREQUENCY OF OSCILLATION AS A FUNCTION OF EXTERNAL OSC. CAPACITANCE



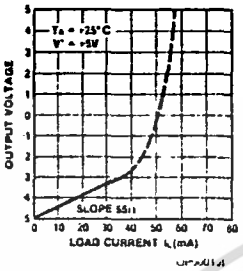
UNLOADED OSCILLATOR FREQUENCY AS A FUNCTION OF TEMPERATURE



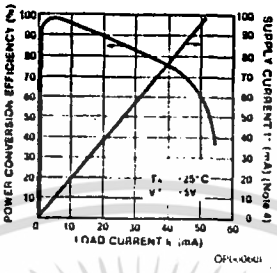
Note: All typical values have been guaranteed by characterization and are not tested.

TYPICAL PERFORMANCE CHARACTERISTICS (CONT.)

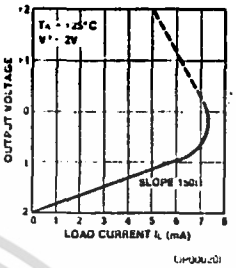
OUTPUT VOLTAGE AS A FUNCTION OF OUTPUT CURRENT



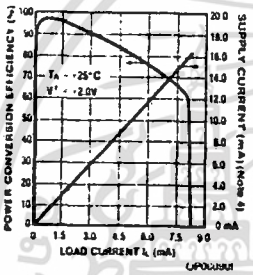
SUPPLY CURRENT & POWER CONVERSION EFFICIENCY AS A FUNCTION OF LOAD CURRENT



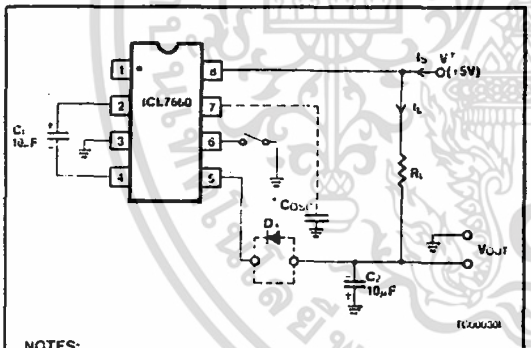
OUTPUT VOLTAGE AS A FUNCTION OF OUTPUT CURRENT



SUPPLY CURRENT & POWER CONVERSION EFFICIENCY AS A FUNCTION OF LOAD CURRENT



NOTE 4. These curves include in the supply current that current fed directly into the load R_L from V+ (see Figure 3). Thus, approximately half the supply current goes directly to the positive side of the load, and the other half, through the ICL7660, to the negative side of the load. Ideally, $V_{OUT} \cong 2\ V_{IN}$, $I_S \cong 2\ I_L$, so $V_{IN} \cdot I_S \cong V_{OUT} \cdot I_L$.



NOTES:

1. For large values of C_{OSC} (> 1000pF) the values of C_1 and C_2 should be increased to 100µF.
2. D_X is required for supply voltages greater than 6.5V @ -55°C ≤ T_A ≤ +70°C; refer to performance curves for additional information.

Figure 3: ICL7660 Test Circuit

DETAILED DESCRIPTION

The ICL7660 contains all the necessary circuitry to complete a negative voltage converter, with the exception of 2 external capacitors which may be inexpensive 10µF polarized electrolytic types. The mode of operation of the device may be best understood by considering Figure 4, which shows an idealized negative voltage converter. Capacitor C_1 is charged to a voltage, V^+, for the half cycle when switches S_1 and S_3 are closed. (Note: Switches S_2 and S_4 are open during this half cycle.) During the second half cycle of operation, switches S_2 and S_4 are closed, with S_1 and S_3 open, thereby shifting capacitor C_1 negatively by V^+ volts. Charge is then transferred from C_1 to C_2 such that the voltage on C_2 is exactly V^+, assuming ideal switches and no load on C_2. The ICL7660 approaches this ideal situation more closely than existing non-mechanical circuits.

In the ICL7660, the 4 switches of Figure 4 are MOS power switches; S_1 is a P-channel device and S_2, S_3 & S_4 are N-channel devices. The main difficulty with this approach is that in integrating the switches, the substrates of S_3 & S_4 must always remain reverse biased with respect to their sources, but not so much as to degrade their "ON" resistances. In addition, at circuit startup, and under output short circuit conditions ($V_{OUT} = V^+$), the output voltage must be sensed and the substrate bias adjusted accordingly. Failure to accomplish this would result in high power losses and probable device latchup.

This problem is eliminated in the ICL7660 by a logic network which senses the output voltage (V_{OUT}) together with the level translators, and switches the substrates of S_3 & S_4 to the correct level to maintain necessary reverse bias.

The voltage regulator portion of the ICL7660 is an integral part of the anti-latchup circuitry, however its inherent voltage drop can degrade operation at low voltages.

Note: All typical values have been guaranteed by characterization and are not tested.

ICL7660

Therefore, to improve low voltage operation the "LV" pin should be connected to GROUND, disabling the regulator. For supply voltages greater than 3.5 volts the LV terminal must be left open to insure latchup proof operation, and prevent device damage.

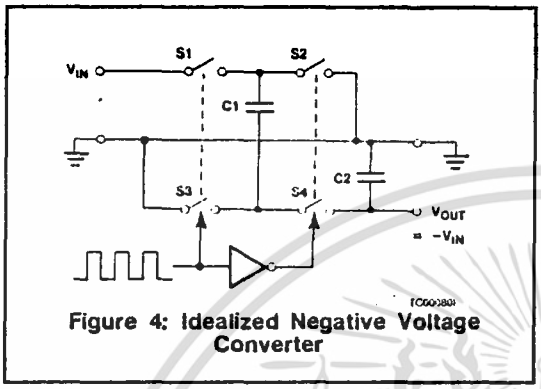


Figure 4: Idealized Negative Voltage Converter

THEORETICAL POWER EFFICIENCY CONSIDERATIONS

In theory a voltage converter can approach 100% efficiency if certain conditions are met:

A The drive circuitry consumes minimal power.

B The output switches have extremely low ON resistance and virtually no offset.

C The impedances of the pump and reservoir capacitors are negligible at the pump frequency.

The ICL7660 approaches these conditions for negative voltage conversion if large values of C_1 and C_2 are used.

ENERGY IS LOST ONLY IN THE TRANSFER OF CHARGE BETWEEN CAPACITORS IF A CHANGE IN VOLTAGE OCCURS. The energy lost is defined by:

$$E = 1/2\ C_1\ (V_1^2 - V_2^2)$$

where V_1 and V_2 are the voltages on C_1 during the pump and transfer cycles. If the impedances of C_1 and C_2 are relatively high at the pump frequency (refer to Figure 4) compared to the value of R_L, there will be a substantial difference in the voltages V_1 and V_2. Therefore it is not only desirable to make C_2 as large as possible to eliminate output voltage ripple, but also to employ a correspondingly large value for C_1 in order to achieve maximum efficiency of operation.

DO'S AND DON'TS

1. Do not exceed maximum supply voltages.
2. Do not connect LV terminal to GROUND for supply voltages greater than 3.5 volts.
3. Do not short circuit the output to V^+ supply for supply voltages above 5.5 volts for extended periods, however, transient conditions including startup are okay.
4. When using polarized capacitors, the + terminal of C_1 must be connected to pin 2 of the ICL7660 and the + terminal of C_2 must be connected to GROUND.
5. Add diode D_X as shown in Figure 3 for high-voltage, elevated temperature applications.
6. Add capacitor (~0.1μF, disc) from pin 8 to ground to limit rate of rise of input voltage to approximately 2V/μs.

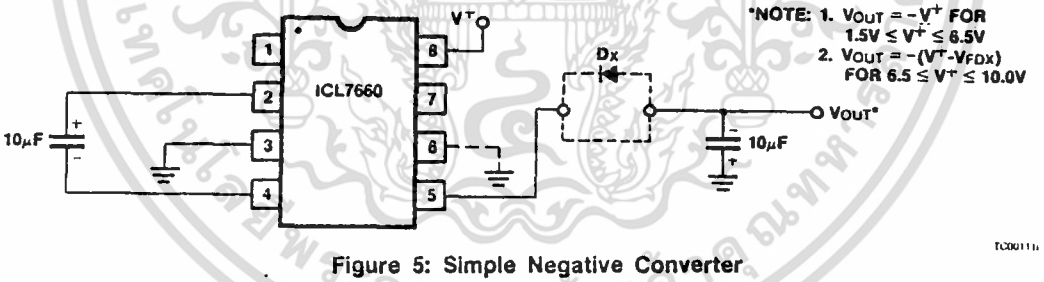


Figure 5: Simple Negative Converter

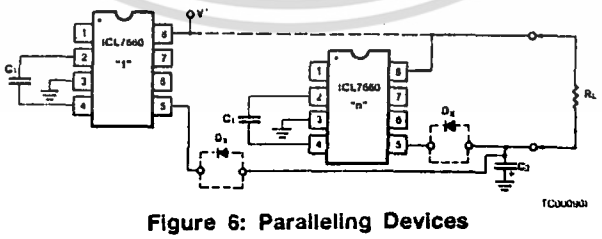


Figure 6: Paralleling Devices

Note: All typical values have been guaranteed by characterization and are not tested.

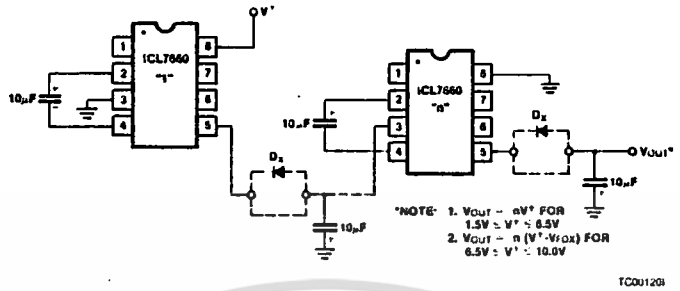


Figure 7: Cascading Devices for Increased Output Voltage

CONSIDERATIONS FOR HIGH VOLTAGE & ELEVATED TEMPERATURE

The ICL7660 will operate efficiently over its specified temperature range with only 2 external passive components (storage & pump capacitors), provided the operating supply voltage does not exceed 6.5 volts at +70°C and 5.0 volts at +125°C. Exceeding these maximums at the temperatures indicated may result in destructive latchup of the ICL7660. (Ref: Graph "Operating Voltage Vs. Temperature")

Operation at supply voltages of up to 10.0 volts over the full temperature range without danger of latchup can be achieved by adding a general purpose diode in series with the ICL7660 output, as shown by "D_X" in the circuit diagrams. The effect of this diode on overall circuit performance is the reduction of output voltage by one diode drop (approximately 0.6 volts).

TYPICAL APPLICATIONS

Simple Negative Voltage Converter

The majority of applications will undoubtedly utilize the ICL7660 for generation of negative supply voltages. Figure 5 shows typical connections to provide a negative supply where a positive supply of +1.5V to +10.0 volts is available. Keep in mind that pin 6 (LV) is tied to the supply negative (GND) for supply voltages below 3.5 volts, and that diode D_X must be included for proper operation at higher voltages and/or elevated temperatures.

The output characteristics of the circuit in Figure 5 are those of a nearly ideal voltage source in series with 55 ohms. Thus for a load current of −10mA and a supply voltage of +5 volts, the output voltage will be −4.3 volts. The dynamic output impedance due to the capacitor impedances is approximately $1/\omega C$, where:

$$C = C_1 = C_2$$

$$\text{which gives } \frac{1}{\omega C} = \frac{1}{2\pi\, f_{PUMP} \times 10^{-5}} \cong 3 \text{ ohms}$$

for $C = 10\mu F$ and $f_{PUMP} = 5kHz$ (1/2 of oscillator frequency)

Paralleling Devices

Any number of ICL7660 voltage convertors may be paralleled to reduce output resistance. The reservoir capacitor, C_2, serves all devices while each device requires its own pump capacitor, C_1. The resultant output resistance would be approximately:

$$R_{OUT} = \frac{R_{OUT} \text{ (of ICL7660)}}{n \text{ (number of devices)}}$$

Cascading Devices

The ICL7660 may be cascaded as shown to produce larger negative multiplication of the initial supply voltage. However, due to the finite efficiency of each device, the practical limit is 10 devices for light loads. The output voltage is defined by:

$$V_{OUT} = -n\,(V_{IN}),$$

where n is an integer representing the number of devices cascaded. The resulting output resistance would be approximately the weighted sum of the individual ICL7660 R_{OUT} values.

Changing the ICL7660 Oscillator Frequency

It may be desirable in some applications, due to noise or other considerations, to increase the oscillator frequency. This is achieved by overdriving the oscillator from an external clock, as shown in Figure 8. In order to prevent possible device latchup, a 1kΩ resistor must be used in series with the clock output. In a situation where the designer has generated the external clock frequency using TTL logic, the addition of a 10kΩ pullup resistor to V^+ supply is required. Note that the pump frequency with external clocking, as with internal clocking, will be 1/2 of the clock frequency. Output transitions occur on the positive-going edge of the clock.

Note: All typical values have been guaranteed by characterization and are not tested.

ICL7660

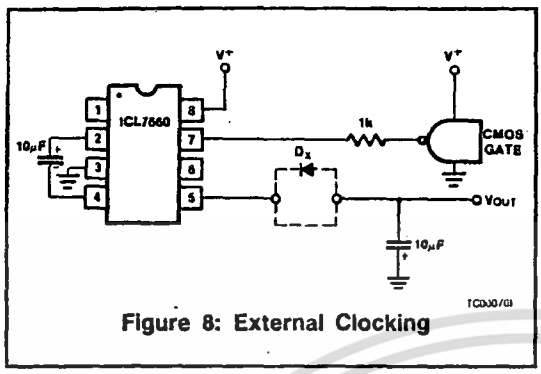


Figure 8: External Clocking

It is also possible to increase the conversion efficiency of the ICL7660 at low load levels by lowering the oscillator frequency. This reduces the switching losses, and is shown in Figure 9. However, lowering the oscillator frequency will cause an undesirable increase in the impedance of the pump (C_1) and reservoir (C_2) capacitors; this is overcome by increasing the values of C_1 and C_2 by the same factor that the frequency has been reduced. For example, the addition of a 100pF capacitor between pin 7 (Osc) and V^+ will lower the oscillator frequency to 1kHz from its nominal frequency of 10kHz (a multiple of 10), and thereby necessitate a corresponding increase in the value of C_1 and C_2 (from 10µF to 100µF).

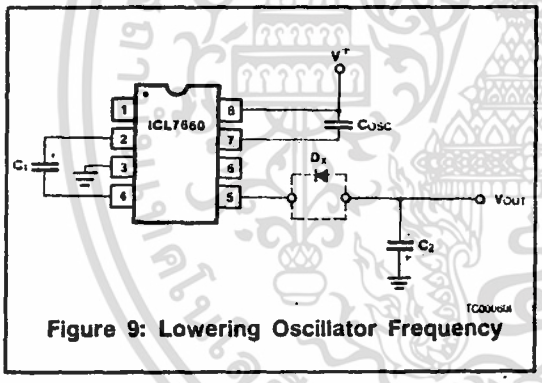


Figure 9: Lowering Oscillator Frequency

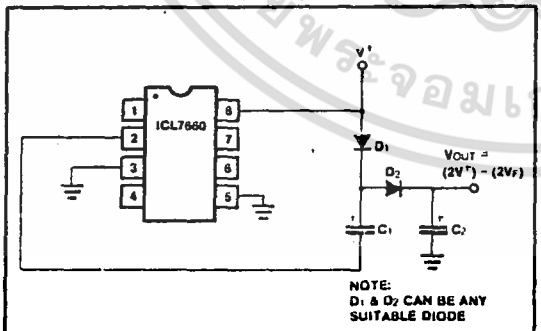


Figure 10: Positive Voltage Doubler

Positive Voltage Doubling

The ICL7660 may be employed to achieve positive voltage doubling using the circuit shown in Figure 10. In this application, the pump inverter switches of the ICL7660 are used to charge C_1 to a voltage level of $V^+ - V_F$ (where V^+ is the supply voltage and V_F is the forward voltage drop of diode D_1). On the transfer cycle, the voltage on C_1 plus the supply voltage (V^+) is applied through diode D_2 to capacitor C_2. The voltage thus created on C_2 becomes $(2V^+) - (2V_F)$ or twice the supply voltage minus the combined forward voltage drops of diodes D_1 and D_2.

The source impedance of the output (V_{OUT}) will depend on the output current, but for $V^+ = 5$ volts and an output current of 10mA it will be approximately 60 ohms.

Combined Negative Voltage Conversion and Positive Supply Doubling

Figure 11 combines the functions shown in Figures 5 and 10 to provide negative voltage conversion and positive voltage doubling simultaneously. This approach would be, for example, suitable for generating +9 volts and −5 volts from an existing +5 volt supply. In this instance capacitors C_1 and C_3 perform the pump and reservoir functions respectively for the generation of the negative voltage, while capacitors C_2 and C_4 are pump and reservoir respectively for the doubled positive voltage. There is a penalty in this configuration which combines both functions, however, in that the source impedances of the generated supplies will be somewhat higher due to the finite impedance of the common charge pump driver at pin 2 of the device.

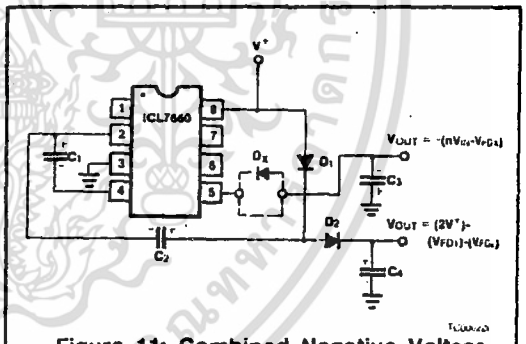


Figure 11: Combined Negative Voltage Converter and Positive Doubler

Voltage Splitting

The bidirectional characteristics can also be used to split a higher supply in half, as shown in Figure 12. The combined load will be evenly shared between the two sides and a high value resistor to the LV pin ensures start-up. Because the switches share the load in parallel, the output impedance is much lower than in the standard circuits, and higher currents can be drawn from the device. By using this circuit, and then the circuit of Figure 7, +15V can be converted (via +7.5, and −7.5) to a nominal −15, although with rather high series output resistance (~250Ω).

Note: All typical values have been guaranteed by characterization and are not tested.

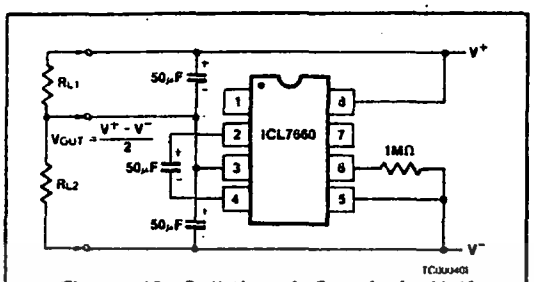


Figure 12: Splitting A Supply in Half

Regulated Negative Voltage Supply

In some cases, the output impedance of the ICL7660 can be a problem, particularly if the load current varies substantially. The circuit of Figure 13 can be used to overcome this by controlling the input voltage, via an ICL7611 low-power CMOS op amp, in such a way as to maintain a nearly constant output voltage. Direct feedback is inadvisable, since the ICL7660's output does not respond instantaneously to change in input, but only after the switching delay. The circuit shown supplies enough delay to accommodate the 7660, while maintaining adequate feedback. An increase in pump and storage capacitors is desirable, and the values shown provides an output impedance of less than 5Ω to a load of 10mA.

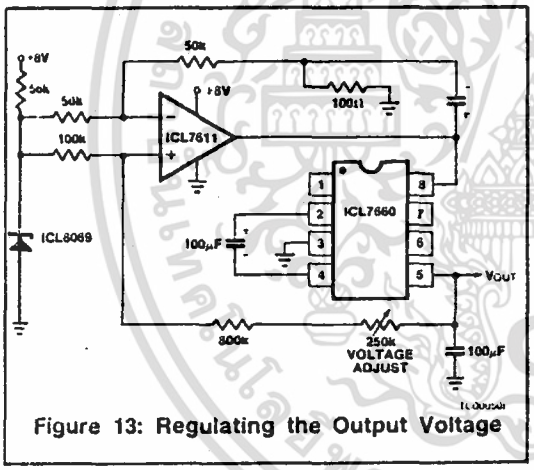


Figure 13: Regulating the Output Voltage

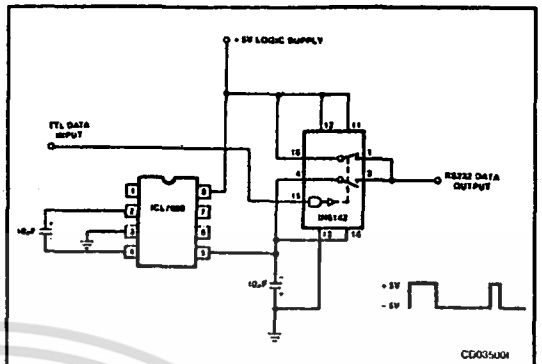


Figure 14: RS232 Levels From A Single 5V Supply

OTHER APPLICATIONS

Further information on the operation and use of the ICL7660 may be found in A051 "Principals and Applications of the ICL7660 CMOS Voltage Converter" by Peter Bradshaw and Dave Bingham.

Note: All typical values have been guaranteed by characterization and are not tested.

AO51
Principles and Applications of the ICL7660 CMOS Voltage Converter

by Peter Bradshaw & Dave Bingham

INTRODUCTION

This application note describes a device originally designed to solve the specific problem of needing a negative supply when only a positive supply is available. This is very common, and occurs, for example, in systems using dynamic RAMs where the three-supply devices require a low current body bias supply of around −5V. Negative supply voltage is also desired in systems with a lot of digital logic (at +5V) but containing a small analog section using A/D converters, such as the ICL7107 or ICL7109 and/or op amps and comparators, operating on ground referenced signals. In all these cases, the current requirement and regulation are not very demanding, but nevertheless, generating such a −5-V supply is usually expensive and inefficient. Typically, a large number of discrete and integrated-circuit components are needed to convert the common +5-V line into a negative one, or to add an extra output to the main supply, the backplane wiring, etc.

This problem is solved by the ICL7660, a monolithic CMOS power supply circuit offering unique performance advantages over previously available devices. With the addition of only two non-critical capacitors (for charge pump and storage), it performs the complete supply voltage conversion from positive to negative for any input voltage between +1.5V and +10V, and provides the complementary output voltage of −1.5V to −10V. (An additional diode is needed for voltages above 6.5V.) The device operates by charging a pump capacitor to the input supply voltage and then applying the capacitor across the output supply, transferring the necessary charge to an open-circuit storage capacitor.

The ICL7660 delivers an open-circuit output equal to the negative of the input voltage to within 0.1%. Capable of producing 20mA, the device has a power-conversion efficiency of about 98% for load currents of 2 to 5mA. The use of two or more 7660s extends the device's capability, as will be shown later.

PRINCIPLES OF OPERATION

Since the 7660 multiplies either positive or negative voltages by a factor of two, it can be considered a simple voltage doubler. This basic voltage doubling operation is shown in Figure 1, where S1 and S3 are the switches used to charge C1, and S2 and S4 transfer the charge to C2. It differs from most voltage doublers in that the usual blocking diodes are replaced by on-chip active MOS transistor switches.

For a negligible load, clearly the voltage inversion will be nearly perfect, with only a tiny charge being lost to stray capacitance. With a significant load, the behavior is more complex.

The amount of charge transferred from C1 to C2 depends upon the amount lost from C2 to the load, and this charge must be made up by C1 from the basic power supply. The switches themselves also have series resistance, leading to further theoretical complications, but the net result is a typical overall output impedance of around 55Ω (100Ω max), provided that the capacitors are sufficiently large. For the natural oscillation frequency of the built-in oscillator (approx. 10kHz) values of 10μF are adequate.

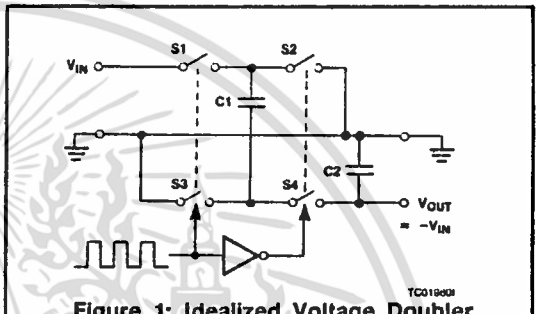


Figure 1: Idealized Voltage Doubler

The complete implementation of this function is achieved on a single CMOS chip, as shown in Figure 2.

The 7660 contains all the necessary conversion functions on-chip, except for the external pump and output reservoir capacitors and is made with a low-threshold CMOS technology using p- and n-channel transistors that turn on at 0.6V. The low power dissipation, simplicity, and small chip size of CMOS make it a near-ideal technology for this application.

The 7660 contains an RC oscillator, a series voltage regulator, a voltage-level translator, and a logic network (Figure 2). The logic network senses the voltage on the sources and drains of the two output n-channel transistors Q_3 and Q_4 and ensures that their substrates are always correctly biased.

POWER EFFICIENCY

In the case where a capacitor is charged and discharged between two voltages, V_1 and V_2, the energy lost is defined by

$$E = \frac{C(V_1 2 - V_2 2)}{2}$$

where C is value of the capacitor in farads and E is the lost energy. If $V_1 - V_2$ is very small compared with V_1, the percentage energy loss is also small, given as:

$$\frac{100(V_1 2 - V_2 2)}{2(V_1 2)}$$

At the limit, when $V_2 = V_1$, no energy is lost. If the values of C_1 and C_2 in Figure 1 are made very large and their impedances at the switching frequency are very low compared with the load resistance, energy-conversion efficiencies approaching 100% can be obtained. Energy is lost only

by a change of voltage during the transfer of charge into and out of a capacitor.

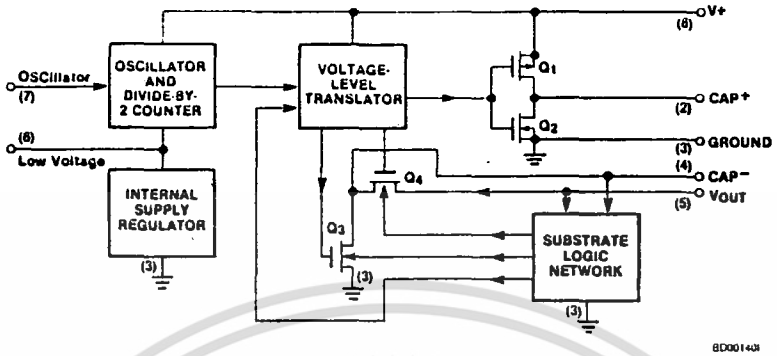


Figure 2: Functional Diagram of Complete Chip

DETAILED DESCRIPTION

Oscillator — Divider — Regulator

The 7660's oscillator (Figure 3) drives a conventional divide-by-2 counter whose principal function is to supply a 50% duty cycle output (at half the input frequency) to the voltage-level translator circuit. The conventional static counter requires a two-phase clock, and supplies an output signal and its complement.

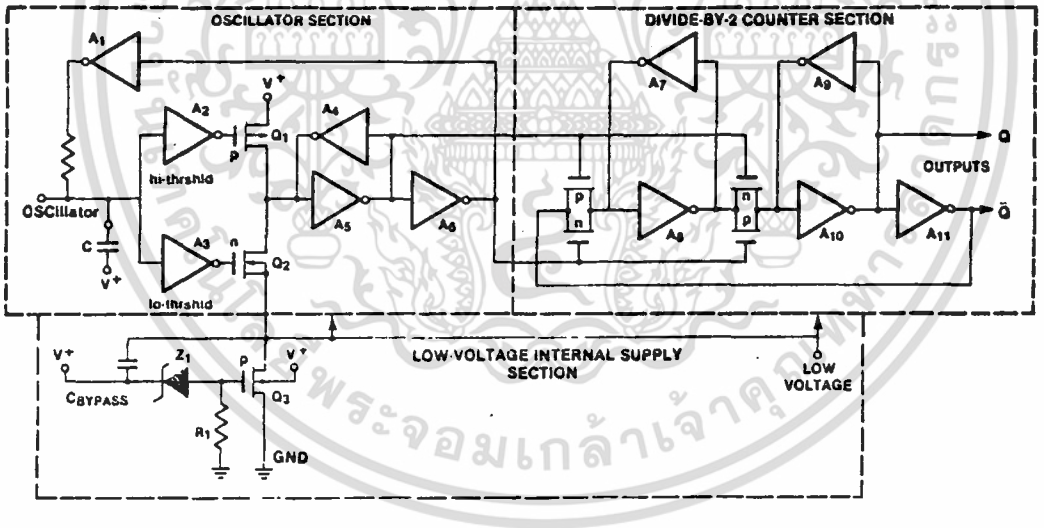


Figure 3: The 7660's Oscillator, Divider and Internal Regulator

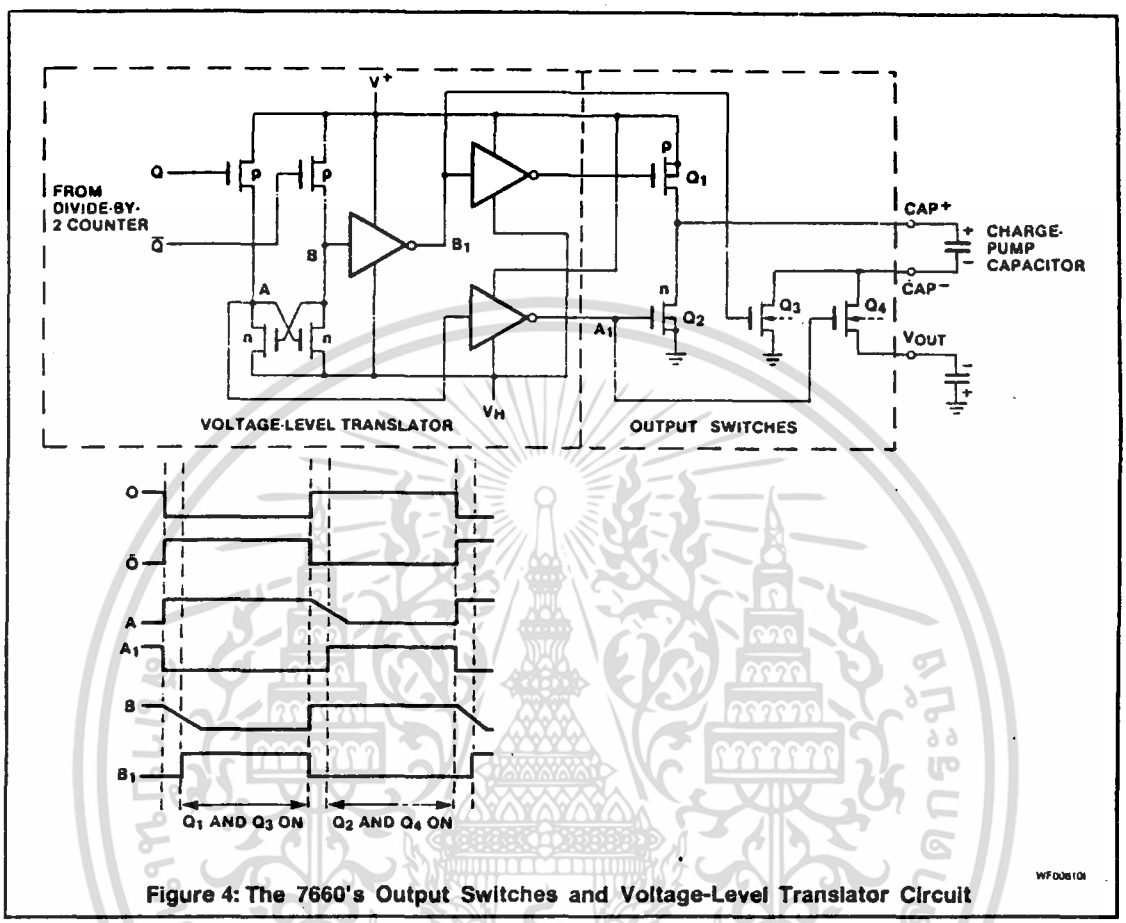


Figure 4: The 7660's Output Switches and Voltage-Level Translator Circuit

When the output of inverter A_1 is switched high, capacitor C charges positively until inverter A_2 (which has a high input-voltage trip point) switches its output low, to turn on transistor Q_1. Q_1 in turn forces the ratioed-inverter latch $A_4 - A_5$ to switch its output low. C then discharges negatively until inverter A_3 (which has a low input-voltage trip point) switches its output high, turning on transistor Q_2. The output of Q_2 resets $A_4 - A_5$ and restarts the cycle.

Since the oscillator has a high input impedance of about 1MΩ, it may be driven from an external source such as a TTL gate or equivalent, or its frequency may be lowered by the addition of an external capacitor. At room temperature with a +5-V supply and no external capacitor, the oscillator frequency will be 10kHz. The internal capacitance is about 10pF.

A series voltage regulator consisting of zener reference diode Z_1, resistor R_1, and source-follower p-channel transistor Q_3 provides a partially regulated supply for all the low-voltage circuitry on the chip. The regulator can supply up to −5V (with respect to the positive power supply) for input supply voltages of about 6V and higher. Because of the modest size of Q_3, the voltage regulator not only reduces power consumption at high supply voltages, but also limits the maximum current taken by the oscillator and the divide-by-2 counter.

The LV terminal can be used to short out the on-chip series regulator for better operation at low supply voltages. With the Low-Voltage terminal connected to ground, operation with an input supply voltage as low as 1V is possible. At higher voltages, however, it is mandatory that this terminal be open, in order to allow the internal voltage regulator to stop device latchup and avoid internal damage.

The Level-Translator and Output Switches

The level translators (Figure 4) provide switching signals to the gates of the four output transistors, Q_1 through Q_4, with amplitudes equal to the sum of the output and supply voltages. They also ensure that a break-before-make sequence takes place as switching alternates between charge and pump configurations.

The Substrate Logic Network

The substrate logic network (Figure 5) is the most critical part of the converter chip. Its two main functions are to make sure that the substrates of Q_2 and Q_4 (Figure 4) are never forward-biased with respect to their sources and drains, and to establish the most negative voltage of any

part of the circuit in either the charge or the pump cycles. This internal negative supply, V_H, is used to power the level translators. It drives the gate of either Q_3 or Q_4 to a voltage similar to that of the sources to ensure transistor turn off.

Transistors Q_3 and Q_4 require special drive considerations, since the sources and drains are inverted on each device during pump and charge phases. Consider Q_3's operation, for example. During the charge phase, the most positive source/drain terminal is connected to the external charge-pump capacitor. This terminal is then, by definition, the drain, whereas the source which is more negative is connected to ground. To minimize Q_3's resistance, it is also desirable to connect its substrate to ground and not to the output voltage or to V_H, since reverse-biasing the substrate of an MOS transistor with respect to its source increases its threshold voltage, and therefore the ON resistance.

During the pumping phase, the external capacitor's negative terminal is shifted negatively by a voltage approximately equal to the supply voltage. In this case, the most negative source/drain terminal is connected to the negative side of the external capacitor (and thus becomes the source of Q_3), and its drain is connected to ground.

Similar source-drain reversals occur for Q_4 except that here conditions are different for output short-circuit operation than during normal operation. Sensing circuitry monitors the voltages on the external capacitor's negative side and V_{OUT}, and compares them with ground. The substrate of Q_4 is then connected to the most negative of them. Figure 5 shows the substrate steering transistors for Q_3 and Q_4. The steering transistors (Q_{S1-5} are relatively small n-channel devices, and share Q_3 and Q_4's substrates.

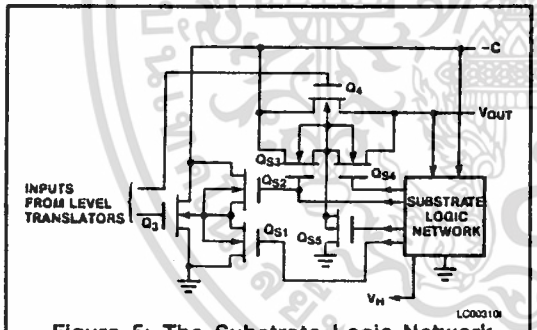


Figure 5: The Substrate Logic Network

SCR Latch Up

A CMOS device is inherently a four-layer, or silicon-controlled-rectifier (SCR), structure. This structure can be turned on through the forward biasing of the inherent pn junctions, and unless external current-limiting circuitry is used, latchup and resultant failure can occur.

The n-channel transistor source acts as the cathode of the SCR, and the p+ source of the p-channel transistor acts as the anode. Either n- or p-channel drains can act as the SCR gate. With about 2 V or more across the anode and cathode, the SCR can have either a low-impedance (ON) or high-impedance (OFF) state. For the ON state to occur, three things must happen: the product of the transistors' current gains, or betas, must be at least unity, a current greater than the holding current must be present, and a trigger pulse must be applied to either gate of the SCR. Trigger signals may be caused by static discharge on the gates or by connecting either gate to the power supplies before connecting power-supply lines to other terminals of the SCR. Even extremely high rates of voltage change across any two or more SCR pn junctions can produce latchup.

Triggering a CMOS SCR causes it to present an extremely low impedance (1 to 100ohms) across the power supply. Unless the power supply is current-limited, the device latches up and is often destroyed, usually by the vaporization of one of the bonding wires.

Although 7660 output-section switching transients are mainly capacitive, they inject currents into the substrate. At high input supply voltages, these transients can forward-bias junctions associated with the p- well or the Q_4 substrate. This in turn may trigger the inherent SCR in Q_4 and the adjacent on-chip circuitry. The result is to rapidly discharge the reservoir capacitor.

After the reservoir capacitor is almost totally discharged and the current in the SCR has fallen below the holding value, the device again operates correctly, until the output voltage (reservoir capacitance voltage) reaches the same critical value, and the latchup phenomenon starts again. Since this effect occurs only during the start of the charge cycle, and not during the pump cycle, isolating the reservoir capacitor with an external diode at the V_{OUT} terminal prevents capacitor discharge. This is recommended when using the device at higher voltage and temperatures. Otherwise the substrate logic network prevents SCR triggering, which is therefore not a problem for most operating conditions.

BASIC APPLICATION

The applications of the ICL7660 are remarkably varied, especially considering the rather narrow nature of the basic device function.

The basic circuit is shown in Figure 6, and the output characteristics for 5V inversion in Figure 7. For light loads, the output voltage follows the input very precisely, while for heavier loads, the output can be viewed as having perfect inversion, plus an output resistance of about 55Ω.

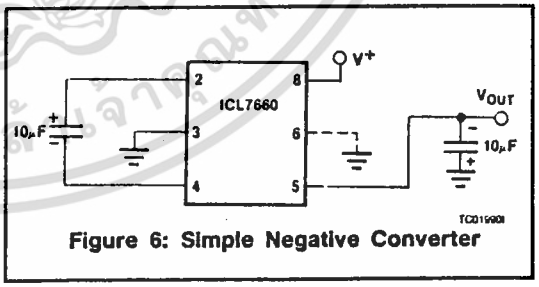


Figure 6: Simple Negative Converter

Thus at 18mA load, the output voltage drops about 1V below the input. Beyond around 40mA, the voltage drop becomes very non-linear, and the circuit self-limits, thereby protecting itself against excessive power dissipation. The output ripple is dependant primarily upon the output capacitor, since this must hold up the load during half the cycle time (or one oscillator period). In the steady-state case, this ripple is made up during the other half cycle time, and

enough pump capacitance should be used to ensure that this is done monotonically. The recommended values ensure this for the internal oscillator frequency.

For operation at low voltages, the output impedance begins to rise rather rapidly, as a result of reduced turn-on voltage on the MOSFET switches (Figure 8). This effect can be reduced by bypassing the internal regulator, tying LV to Ground, as shown in Figure 9. **This must not be done, however, if the incoming supply can exceed 6V under any circumstances, as the internal logic oscillator and divider stages will be damaged.** Note also the use of a series diode (Dx) at higher voltage and temperature, to protect the device against SCR action.

Figure 9 also shows an external oscillator capacitor. This can be used to reduce the oscillator frequency, giving a slight improvement in efficiency; see Figure 10.

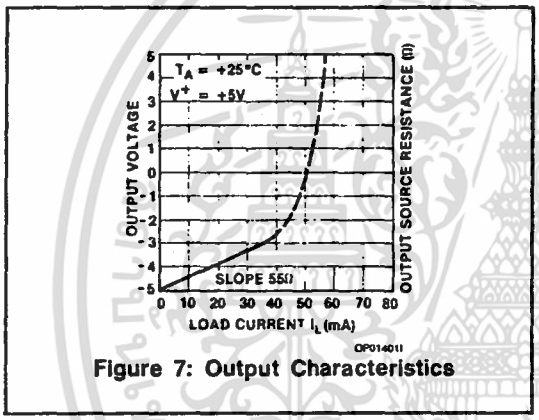


Figure 7: Output Characteristics

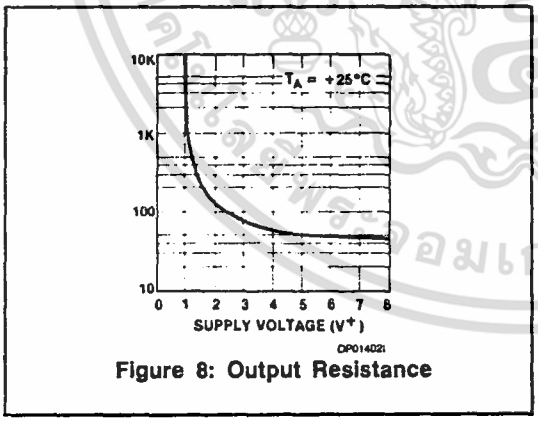


Figure 8: Output Resistance

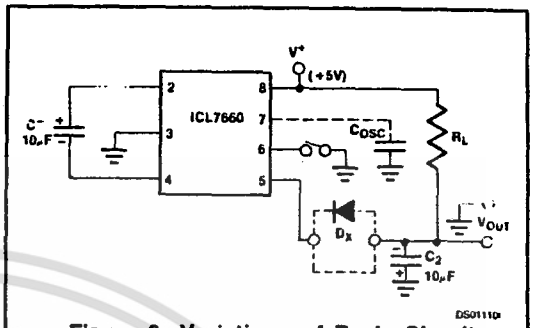


Figure 9: Variations of Basic Circuit

The dependence of the frequency on this external capacitance is shown in Figure 11. This can also be done to move the frequency away from a band of undue sensitivity to EMI in a system. However the output ripple will be increased, and the output impedance also unless the pump and storage capacitors are correspondingly increased.

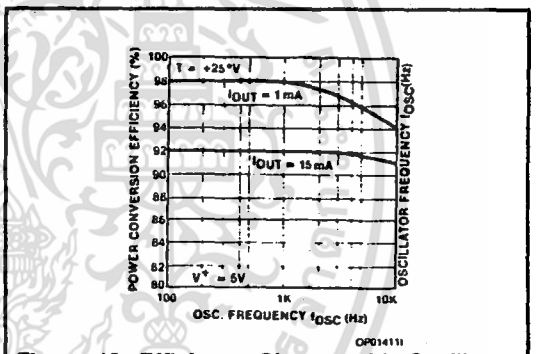


Figure 10: Efficiency Change with Oscillator Frequency

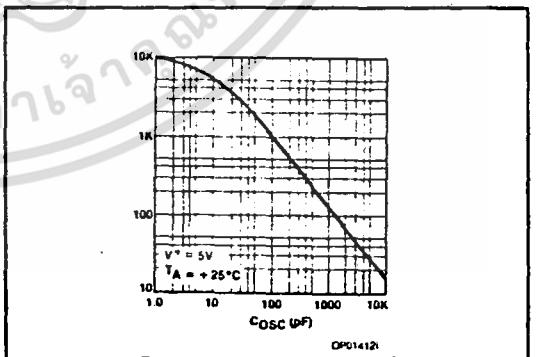


Figure 11: Frequency Variation with Oscillator Capacitance

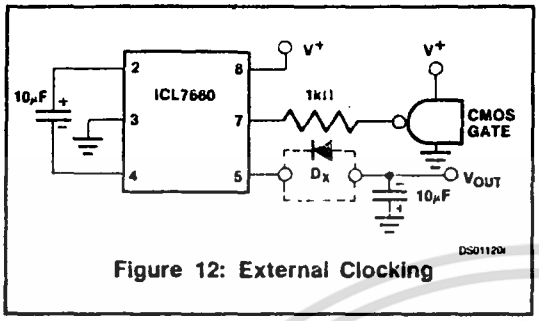


Figure 12: External Clocking

Synchronization to an external clock can be readily achieved, as shown in Figure 12. A TTL device can be used with the addition of a pull-up resistor (10kΩ to V^+ is suitable), as can any input swinging rail-to-rail on the positive supply. The series resistor prevents problems with overdrive on the internal logic. Output transitions occur on the positive edge of the external input.

WIDER (Parallel Connections)

For applications where the voltage drop due to load current is excessive, several ICL7660s can be paralleled. Normally this cannot be done efficiently with power supply circuits, since each one has a different idea of where the "ideal" output voltage would be and they usually end up fighting each other. However, here they see equal input voltages, and the virtually perfect inversion assures that each one does have the same idea of where the output should be so load-sharing is assured. Each device must have a separate pump capacitor, since the oscillators cannot be synchronized except with an external drive, and even then the −2 will be in a random condition. The connections are shown in Figure 13. Naturally the output capacitor is common to each device. Running independently, the ripple content will include components at the difference frequency as well as the individual pumping frequencies. If this is undesirable, a single exclusive NOR gate can be used to put two ICL7660s into antiphase by comparing the outputs on pin 2, and clocking one to maintain near synchronization with the basic oscillator of the other, as shown in Figure 14.

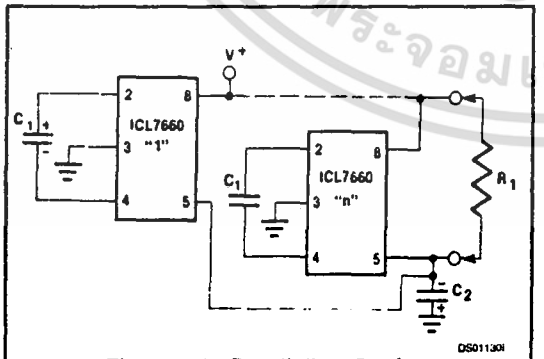


Figure 13: Paralleling Devices

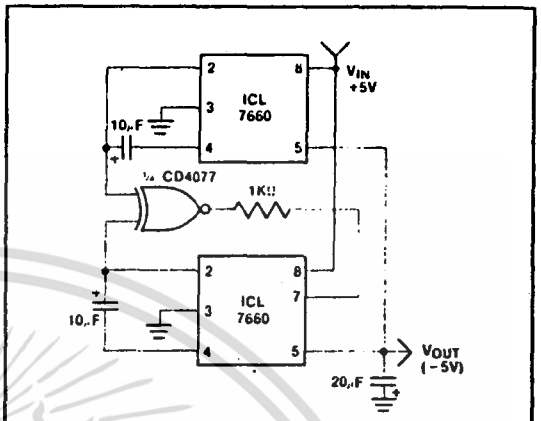


Figure 14: Synchronizing Two ICL7660s

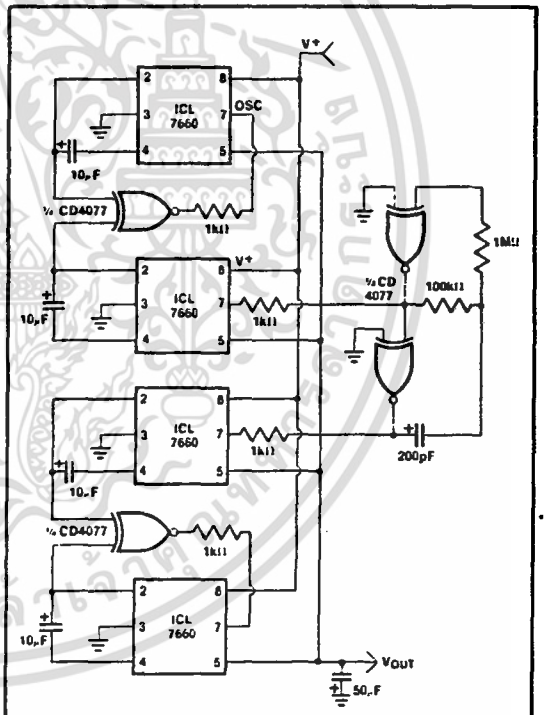


Figure 15: Synchronizing a Quad

The concept can be extended to drive four devices in four separate phases, using a single extra logic-gate package, as shown in Figure 15. The duty cycle of the oscillator is reasonably close to 50%, so driving two pairs, each in the configuration of Figure 14, from opposite phases of the oscillator gives four separately-timed pumps per cycle. This circuit will give about 75mA output before the voltage drops by 1V, or an output impedance of under 14Ω. The four-

phase operation minimizes the ripple, while ensuring very even load sharing. For even more parallel synchronous device, a Johnson counter using Q and $\overline{Q}$ outputs should be considered.

DEEPER (Series Connection)

It is also possible to connect ICL7660s in series, cascading them to generate higher negative voltages. The basic connections are shown in Figure 16.

This technique can be extended to several multiplication levels. However, the basic limitations of this technique must be recognized. In line with the Laws of Thermodynamics, the input current required for each stage is twice the load current on that stage, plus the quiescent current required to operate that stage.

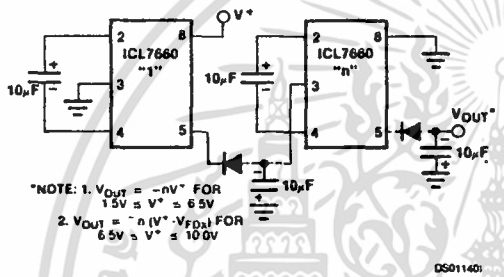


Figure 16: Cascading Devices for Increased Output Voltage

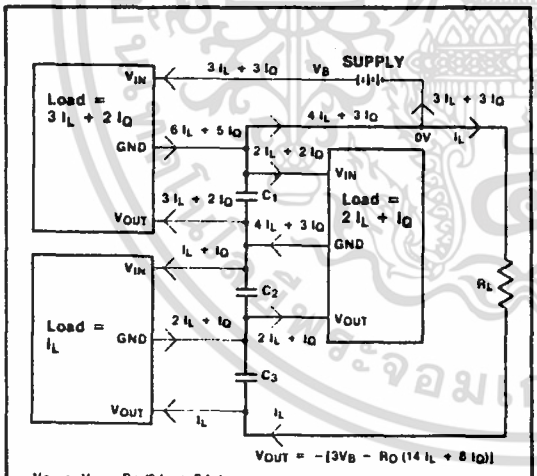


Figure 17: Current Flow for Cascaded Devices

Thus the load current is rapidly multiplied down the chain, as shown in Figure 17. Note also that the quiescent current increases the load current on each stage, though not as fast as the ultimate load itself.

Furthermore, the loss in voltage in early stages due to series resistance is multiplied through all subsequent stages. Thus the effective output impedance mounts rapidly with the number of stages. (See Table 1) This effect can be reduced by paralleling devices in the lowest stages (see above.) If the weighting corresponds to the square of the position, the effective resistance to load current goes up only linearly with the number of stages, but the cost quickly becomes prohibitive. Nevertheless, for light loads and moderate multiplication, useful performance can be achieved.

TABLE 1

#STAGES	RESISTANCE	MULTIPLIERS
n	$R_O(L)$	$R_O(Q)$
1	1	0
2	5	2
3	14	8
4	30	20
5	55	40

A variation of this circuit, another form of series circuit, is shown in Figure 18. This circuit can be used effectively to generate −15V from +5V in light load applications using only two devices. The output impedance corresponds roughly to n = 2 in Table 1, much better than if the previous circuit were used with n = 3. In general, geometric increases, as in Figure 18, are better until the voltage limit is reached, at which time arithmetic cascading as in Figure 16 must be utilized.

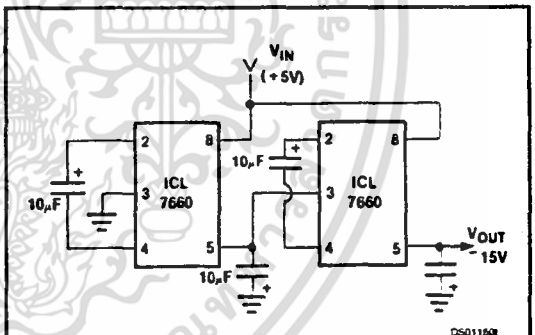


Figure 18: Getting −15V from +5V

UPSIDE DOWN (Positive Multiplication)

The ICL7660 may be employed to achieve positive voltage multiplication using the circuit shown in Figure 9. In this application, the pump inverter switches of the ICL7660 are used to charge C_1 to a voltage level of $V^+ - V_F$ (where V^+ is the supply voltage and V_F is the forward voltage drop of diode D_1.) On the transfer cycle, the voltage on C_1 plus the supply voltage (V^+) is applied through diode D_2 to capacitor C_2. The voltage thus created on C_2 becomes $(2V^+) - (2V_F)$ or twice the supply voltage minus the combined forward voltage drops of diodes D_1 and D_2.

The source impedance of the output (V_{OUT}) will depend on the output current, but for $V^+ = 5$ volts and an output current of 10mA it will be approximately 60 ohms.

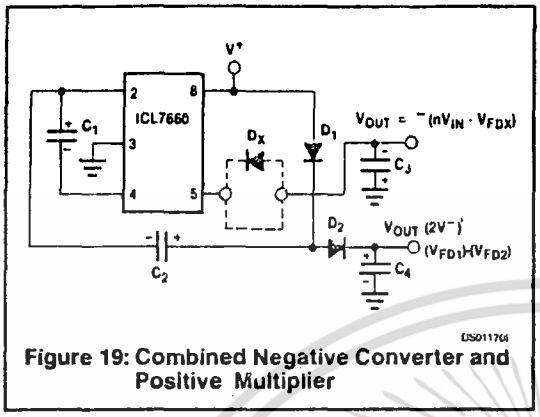


Figure 19: Combined Negative Converter and Positive Multiplier

DIVIDE AND CONQUER

The ICL7660 can be used to split a supply in half, as shown in Figure 20.

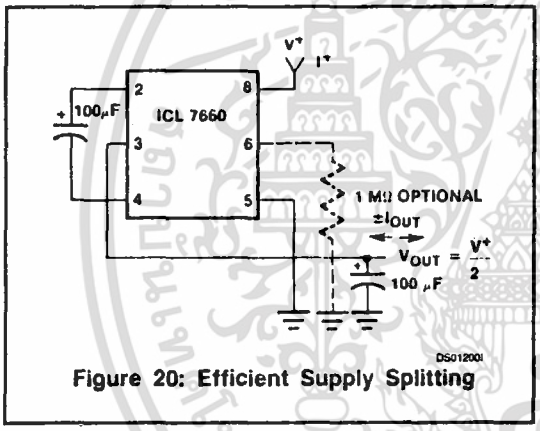


Figure 20: Efficient Supply Splitting

Here the "basic" output connection and the "basic" negative supply input are exchanged and the output voltage thus becomes the midpoint. Start-up can be a problem, and although careful capacitance and load balancing may frequently be adequate, a simple resistor to LV will always work. The circuit is useful for series-fed line systems, where a heavy local load at low voltage can be converted to a lighter current, at high voltage. Other useful applications are in driving low voltage (eg. ±7.5V) circuits from ±15V supplies, or low voltage logic from 9V or 12V batteries. The output impedance is extremely low; all parts of the circuit cooperate in sharing the current, and so act in parallel.

For other division ratios, the series configurations of Figure 16 can be driven backwards, to generate V_{in}/n, or even $m/n(V_{in})$, for small values of m and n. Again, care must be taken to ensure start up for each device.

One interesting combination of several preceding circuits is shown in Figure 21, where a +15V supply is converted, via +7.5V and −7.5V, to −15V using three ICL7660s. The output impedance of this circuit is about 250Ω.

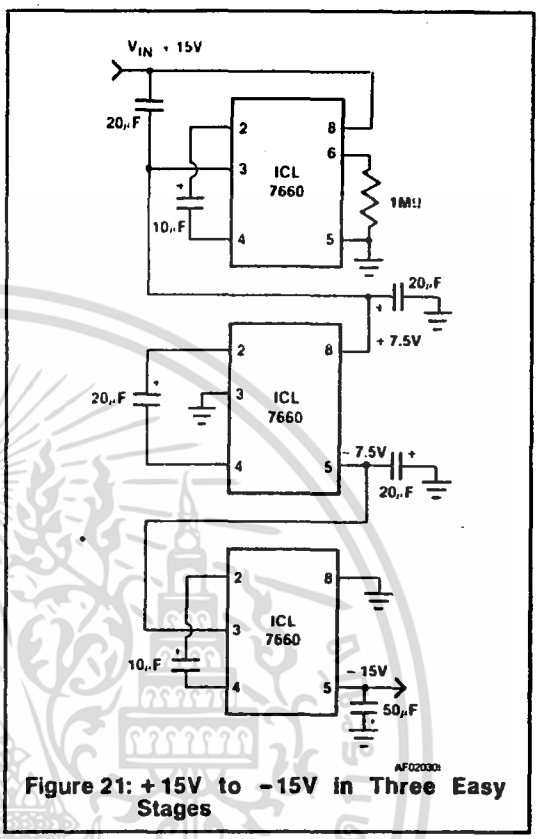


Figure 21: +15V to −15V in Three Easy Stages

For cases where the output impedance of an ICL7660 circuit is too high, obviously some form of output regulation can be used. However in most cases adequate regulation can be achieved at high efficiency by pre-regulating the input. A suitable circuit is shown in Figure 22, using the ICL7611 low power CMOS op amp. Because of the large source-current capability of this op amp, even on its lowest bias current setting, very efficient operation is possible. An ICL8069 band-gap device is used as the reference generator for the regulator. The output impedance can be reduced to 4Ω, while maintaining a current capability of well over 10mA. In designing circuits of this type, it is important to remember that there is a switching delay averaging one oscillator cycle between the output of the op amp and the actual output voltage. This can have substantial repercussions on the transient response if the time-constants in the circuit are not adequate. If multiple voltage converters are used, synchronization schemes such as those of Figures 14 and 15 are probably advisable.

MESSING ABOUT

The applications shown so far have corresponded to the use of the ICL7660 as a sort of equivalent of single turns on a power transformer, with paralleled turns to get more current, series turns for more voltage, etc. However, there are some other possibilities. By looking again at the block diagram (Figure 2), it is evident that the device could be used as a 50% duty cycle high power clock driver, using

either the internal oscillator or an external signal, as in Figure 23. An antiphase clock can also be derived from the circuit, as shown, but the pull-up on this output, being an N-channel switch only, does not have as good a voltage swing. It is adequate for TTL level operation, but for CMOS clocking may require an external pull-up resistor or transistor.

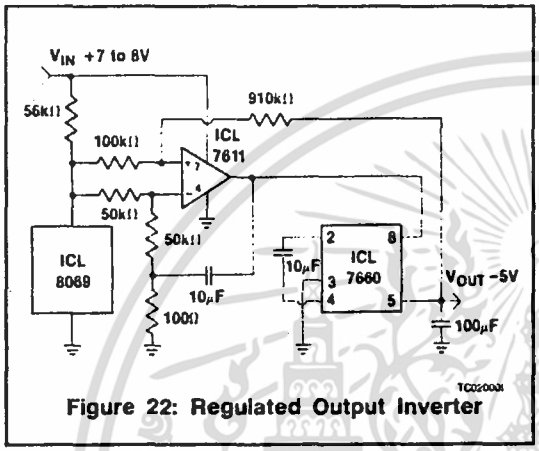


Figure 22: Regulated Output Inverter

Another interesting class of applications comes from the capability to synchronously detect the output of an AC driven transducer, as shown in Figure 24. (This could be viewed as a signal transformer application.) Although the circuit shown utilizes a linear transformer type of transducer, any similar device may be used. The output voltage, which is correctly phased and of either polarity, may be fed into an A/D converter such as an ICL106/7 or ICL7109, for display or microprocessor interface as desired.

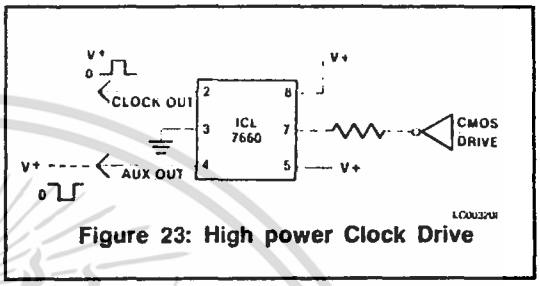


Figure 23: High power Clock Drive

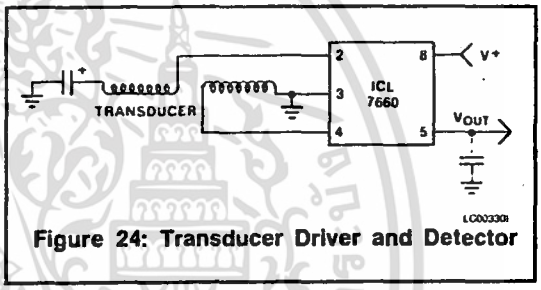


Figure 24: Transducer Driver and Detector

เอกสารอ้างอิง

1. น.ต.ดร.ไพศาล สงวนหมู่,รศ.ยืน ภู่วรวรรณ., **การสื่อสารข้อมูลและไมโครคอมมพิวเตอร์เนตเวอร์ค.** กรุงเทพฯ: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2532

2. William J.Beyda.,**Basic Data Communications:A Comprehensive Overview.** Englewood Cliffs, N.J.: Prentice-Hall, INC.,1989

3. William Sinnema and Tom Mcgovern.,**Digital,Analog,and Data Communication(2nd ed.).** Englewood Cliffs, N.J.: Prentice-Hall,INC

4. D.E.Pippenger and E.J. Tobaben., **Linear and Interface Circuits applications(2nd ed.).** United States of America. Texas Instruments Incorporated., 1988

5. Joseph C. Nichols,Elizabeth A. Nichols and Keith R. Musson Mcgraw-Hill, Inc., 1982

6. ดร.ประสิทธิ์ ประพิณมงคลการ., **หลักการระบบสื่อสาร.** กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2521

7. Uyless D. Black, **Data Communications and Distributed Networks (2nd.ed.).** Englewood Cliffs, N.J.: Prentice-Hall,INC,1987

8. Richard Templer and George Attard., " **The World of Liquid Crystals** " New Scientist (May 4, 1991)

9. ดร.พูลพงศ์ บุญพราหมณ์., " **ผลึกเหลวเพื่องานอุตสาหกรรม** " วิทยาศาสตร์ (39-11) (1985)

10. Burr-Brown Corporation., **Burr-Brown Integrated Circuits Data Book Supplement.** Burr-Brown Corporation, United States of America,1987

11. Paul Horowitz and Winfield Hill., **The Art of Electronic (2nd ed.)**. Press Syndicate of The University of Cambridge, 1989

12.Kenneth J. Ayala., **The 8051 Microcontroller Architecture, Programing and Applications.** West Publishing Company,United states of America,1991

13. Intel Corporation., **Basic-52 User's Manual,** 1981

14.Intel Corporation., **Microprocessor Data Book MCS-51 Microcontrollers,** 1981

15. ปรอง กองทรัพย์โต,ไพบูลย์ ศิระพัฒน์ และสุรเมธ สัจจอิสริยวุฒิ **ระบบพัฒนาไมโครคอมพิวเตอร์ชิปเดี่ยว 8052 AH-BASIC,** วิทยานิพนธ์วิทยาศาสตรบัณฑิต, ภาควิชาฟิสิกส์ประยุกต์, คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า ลาดกระบัง 2532

16. สมภพ ภูริวิกรัยพงศ์,สราวุธ วังบุญคง และชาวดี **นิชัยรัตน์,เครื่องควบคุมอุณหภูมิโปรแกรมได้** วิทยานิพนธ์วิทยาศาสตรบัณฑิต ,ภาควิชาฟิสิกส์ประยุกต์,คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง 2532

17. Daniel H. Sheingold.,**Analog-Digital Conversion Handbook,** 3rd.ed.,Prentice-Hall,Englewood Cliffs.,1986.

18. สมยศ ภู่งามแปลง. **"MCS-51 ไมโครคอนโทรลเลอร์"** วารสารเซมิคอนดักเตอร์ อิเลคทรอนิคส์ 93 (2532):264-269.

19. สมิธ วรภัทราทร. **"8052 เบสิกไมโครคอนโทรลเลอร์"** วารสารเซมิคอนดักเตอร์ อิเลคทรอนิคส์ 96 (2532):218-225.

20. เปรมจิตร วิสุทธิศิริ. **"พื้นฐานวงจรเอทูดี ดีทูเอ ตอน 2 วงจรแปลงอะนาลอกเป็นดิจิตอล"** วารสารเซมิคอนดักเตอร์ อิเลคทรอนิคส์ 103 (2533):302-309

ประวัติผู้เขียน

นาย ปฐมสรรค ศิริหาญยากร

เกิดเมื่อวันที่ 23 มิถุนายน พุทธศักราช 2512 ที่กรุงเทพมหานคร จบการศึกษาระดับมัธยมต้นจากโรงเรียน ศรีบุญยานนท์ จังหวัดนนทบุรี และจบการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียน บวรนิเวศ กรุงเทพฯ จากนั้นได้เข้าศึกษาต่อในระดับปริญญาตรี ในภาควิชา ฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง กรุงเทพฯ ในปีการศึกษา 2531 และจบการศึกษาในปี 2534

นาย ณรงค์ แสงแก้ว

เกิดเมื่อวันที่ 27 เมษายน พุทธศักราช 2513 ที่โรงพยาบาล อำเภอดำเนินสะดวก จังหวัดราชบุรี เริ่มการศึกษาตอนต้นที่ โรงเรียนวัดปราสาทสิทธ์ ตำบลประสาทสิทธ์ เมื่อจบการศึกษาก็เข้ามาเรียนต่อที่ โรงเรียนปทุมคงคา กรุงเทพฯ จากนั้นได้เข้าศึกษาต่อในระดับปริญญาตรี ที่ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง กรุงเทพฯ ในปีการศึกษา 2531 และจบการศึกษาในปี 2534