การหาพารามิเตอร์ของระบบ

(SYSTEM IDENTIFICATION)

โดย

นายกิตติศักดิ์ โควินท์ทวีวัฒน์

นายชัชวาลย์ ทวีศักดิ์ศิริผล

นายศุภโชค สุทธาพานิช

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

การหาพารามิเตอร์ของระบบ
(SYSTEM IDENTIFICATION)

โดย

นายกิตติศักดิ์	โควินท์ทวีวัฒน์	38014028
นายชัชวาลย์	ทวีศักดิ์ศิริผล	38014110
นายศุภโชค	สุทธาพานิช	38014511

อาจารย์ที่ปรึกษา

อาจารย์พรสุข	รติโรจน์อนันต์
รศ.วิพันธ์	ปรีชาพานิช
รศ.สุเธียร	เกียรติสุนทร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การหาพารามิเตอร์ระบบ

(SYSTEM IDENTIFICATION)

ผู้จัดทำ

1.นายกิตติศักดิ์ โควินท์ทวีวัฒน์ 38014028

2.นายชัชวาลย์ ทวีศักดิ์ศิริผล 38014110

3.นายศุภโชค สุทธาพานิช 38014511

.....อาจารย์ที่ปรึกษา

(อาจารย์ พรสุข รติโรจน์อนันต์)

.....อาจารย์ที่ปรึกษา

(รศ. วิพันธ์ ปรีชาพานิช)

.....อาจารย์ที่ปรึกษา

(รศ. สุเธียร เกียรติสุนทร)

การหาพารามิเตอร์ของระบบ

System Identification

โดย นายกิตติศักดิ์ โควินท์ทวีวัฒน์ 38014028
นายชัชวาลย์ ทวีศักดิ์ศิริผล 38014110
นายศุภโชค สุทธาพานิช 38014511

อาจารย์ที่ปรึกษา อาจารย์พรสุข รติโรจน์อนันต์
รศ. วิพันธ์ ปรีชาพานิช
รศ. สุเชียร เกียรติสุนทร

บทคัดย่อ

การหาค่าพารามิเตอร์ระบบเป็นเครื่องมือในการหาค่าพารามิเตอร์ของระบบที่ไม่ทราบค่าเพื่อทำการออกออกแบบตัวควบคุมตามต้องการ

ปริญญานิพนธ์ฉบับนี้จะกล่าวถึงหลักการหาค่าพารามิเตอร์ระบบแบบรีเคอร์ซีพลีสต์สแควร์ โดยจะทำการศึกษาและเขียนโปรแกรมเพื่อทำการหาพารามิเตอร์โดยใช้ภาษา C++

ได้ทดสอบโปรแกรมที่สร้างขึ้นโดยการหาค่าพารามิเตอร์ของระบบที่จำลองไว้และระบบจริง ผลลัพธ์ที่ได้ใช้ได้ดีในระดับหนึ่ง

ABSTRACT

System Identification is a useful tool for identifying the unknown plant parameter which allows the designer to design the appropriate controller.

In this project the principle of System Identification used Recursive Least Square Method is studied and implemented into a software developed using the C++ programming langaue.

This simulation program is tested by using it to identify parameters of both the simulation and real plants, with satisfactory results obtained.

สารบัญ

เอกสารอ้างอิง

กิตติกรรมประกาศ

สารบัญรูปภาพ

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

ในอุตสาหกรรมต่างๆโดยทั่วไป จะประกอบด้วยกระบวนการต่างๆและในแต่ละกระบวนการนั้นจะมีลักษณะเฉพาะของตัวเอง ซึ่งเป็นปัจจัยสำคัญที่มีผลต่อการเลือกขั้นตอนการผลิต ประสิทธิภาพของการผลิต รวมทั้งต้นทุนการผลิต ดังนั้นปัญหาที่เกิดขึ้นเสมอคือเราควรจะทำการกำหนดคุณลักษณะระบบอย่างไร มีขั้นตอนการทำงานอย่างไร ใช้เครื่องมืออะไร โดยมีเป้าหมายเพื่อที่จะให้ได้ผลลัพธ์ที่ได้จากกระบวนการถูกต้องที่สุด

ในปริญญานิพนธ์ฉบับนี้จะนำเสนอเกี่ยวกับรายละเอียดของขั้นตอนในการหาพารามิเตอร์ของระบบควบคุมที่ไม่ทราบค่า เพื่อมาออกแบบตัวควบคุม การหาค่าพารามิเตอร์ของระบบเป็นวิชาที่มีเนื้อหาเกี่ยวข้องการสร้างแบบจำลองทางคณิตศาสตร์โดยอาศัยข้อมูลจากการทดลอง การหาค่าพารามิเตอร์ระบบนี้ใช้สำหรับหารูปแบบที่เหมาะสมสำหรับการควบคุม ออกแบบขั้นตอนในการหารูปแบบของโมเดล โดยวิธีในการหาค่าพารามิเตอร์นี้จะใช้วิธีของ Recursive Least Square โดยอาศัยโปรแกรมที่เขียนด้วยภาษา C++ เพราะเป็นวิธีที่สามารถนำมาประยุกต์ใช้กับ real plantได้และเป็นวิธีที่ทำให้ค่าพารามิเตอร์ลู่เข้าสู่ค่าที่ถูกต้องได้อย่างรวดเร็ว และสุดท้ายก็ทำการตรวจสอบโมเดลที่จำลองขึ้นด้วย

การหาค่าพารามิเตอร์ระบบนี้สามารถนำมาประยุกต์ใช้กับงานต่างๆได้มากมาย เช่น ในการประยุกต์ใช้กับระบบสัญญาณ การสื่อสาร และในงานวิศวกรรมสาขาต่างๆ

ในปริญญานิพนธ์ฉบับนี้จะมีรายละเอียดของบทต่างๆโดยสังเขปดังนี้

บทที่ 1 จะกล่าวถึงเนื้อหาโดยย่อและการประยุกต์ใช้งานของการหาพารามิเตอร์ระบบ

บทที่2 จะกล่าวถึงทฤษฎีที่เกี่ยวกับการหาพารามิเตอร์ระบบ ซึ่งก็ได้แก่ การเลือกสัญญาณอินพุท การเลือกโครงสร้างโมเดล วิธีการหาค่าพารามิเตอร์แบบรีเคอร์ซีพ รวมทั้งการตรวจสอบโมเดล

บทที่3 จะกล่าวถึงการออกแบบวงจรเพื่อทำการติดต่อกับระบบจริงในการหาค่าพารามิเตอร์ระบบ เช่น วงจรฟิลเตอร์ วงจรเอทูดี วงจรไมโครคอนโทรลเลอร์ เป็นต้น

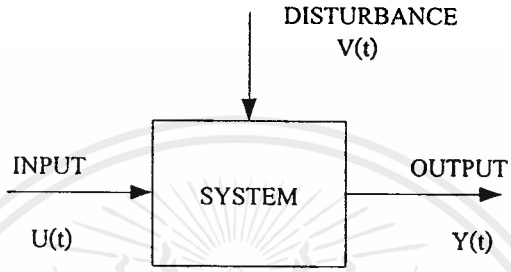
บทที่4จะกล่าวถึงวิธีการวิเคราะห์และผลจากการวิเคราะห์ในการหาค่าพารามิเตอร์ของระบบควบคุมที่ทราบค่า (แพลนท์ที่ซิมูเลตขึ้น) และการหาค่าพารามิเตอร์ระบบควบคุมที่จำลองขึ้นมา

บทที่ 5 จะเกี่ยวข้องกับการสรุปและวิจารณ์ผลการวิเคราะห์การหาพารามิเตอร์ของระบบ

บทที่ 2
ทฤษฎีและหลักการของการหาพารามิเตอร์ของระบบ

2.1การหาค่าพารามิเตอร์ระบบ

การหาค่าพารามิเตอร์ระบบเป็นการออกแบบ dynamic system จากข้อมูลที่ทำการทดลองซึ่งจะมีลักษณะดังนี้



บล็อคไดอะแกรมจะแสดงความสัมพันธ์ระหว่างสัญญาณที่ได้จากการวัด เป็นการสะดวกที่ทำการแยกระหว่างสัญญาณอินพุทกับเอาท์พุท ซึ่งเอาท์พุทนั้นส่วนหนึ่งจะมาจากอินพุท แต่พบว่าบางครั้งเอาท์พุทจะได้รับผลกระทบจากสัญญาณอื่นที่ไม่ใช่สัญญาณอินพุท ซึ่งค่าอินพุทที่เราไม่ต้องการนี้เราจะเรียกว่าสัญญาณ Disturbance หรือ noise ซึ่งเราจะแทนอินพุท เอาท์พุท และ Disturbance ด้วย U(t) , Y (t) และ V(t) โดยเราสามารถควบคุมอินพุท U(t) ได้แต่ไม่สามารถควบคุมสัญญาณ Disturbance พบว่าในบางระบบจะไม่มีอินพุทจะเป็นไปตามข้อมูลระบบ

จะเห็นว่าสัญญาณทุกชนิดเป็นฟังก์ชันของเวลา ในการหาค่าพารามิเตอร์ระบบจะสัญญาณเป็นแบบดีสครีต ดังนั้นสัญญาณในการวัดจึงถูกบันทึกไว้ในรูปสัญญาณดีสครีต ซึ่งมีค่าคาบเวลาการชักตัวอย่าง T

ในกระบวนการอุตสาหกรรมต้องควบคุมคำสั่งการทำงานอย่างปลอดภัยและมีประสิทธิภาพ ในโมเดลบางชนิดต้องการออกแบบโมเดลของตัวควบคุม สามารถเปลี่ยนแปลงชนิดและระดับที่ซับซ้อนบางครั้งมันก็เพียงพอต่อการหาค่า Gain Margin, Crossover Frequency และ Phase Margin ใน Bode Plot ในกรณีอื่นตัว Optimal controller ผู้ออกแบบต้องการข้อมูลของโมเดลเพื่ออธิบายผลของ Disturbance ต่อโมเดล

การประยุกต์ signal processing ในกรณีนี้ การสื่อสารข้อมูล Radar Sonar และอื่นๆเป็น Filter ในการออกแบบฟิลเตอร์ที่ดีต้องรู้คุณสมบัติของสัญญาณ

2.2 ชนิดของโมเดล

1. Mental Intuitive หรือ Verbal Model (เป็นแบบที่เกิดจากจินตนาการ โดยสัญชาติญาณ)
2. กราฟแสดงตาราง Bode Plot เป็นตัวอย่างของกราฟ
3. Mathematical Model ได้แก่สมการผลต่างและสมการเชิงอนุพันธ์ ในแบบนี้จะใช้ในการหาค่าและออกแบบโมเดลได้ดี

แบบจำลองทางคณิตศาสตร์ (Mathematical Modeling) เป็นการวิเคราะห์สมบัติทางกายภาพเพื่ออธิบายพฤติกรรมของระบบ การหาค่าพารามิเตอร์ระบบเป็นการทดลอง ซึ่งทั้ง 2 รูปแบบสามารถเทียบกันได้ ถ้าระบบมีความซับซ้อนมากเราก็จะใช้ Identification Technique

โมเดลที่ใช้ในการหาค่าพารามิเตอร์ของระบบต้องมีคุณสมบัติดังนี้

- อยู่ในข้อจำกัดที่ถูกต้อง
- ให้รายละเอียดของโมเดลที่ถูกต้อง
- มีความสัมพันธ์ง่ายๆต่อการนำไปสร้างและใช้

การ Identification ไม่ใช่วิธีที่จะไม่มีข้อผิดพลาดเลย แต่จะขึ้นอยู่กับผู้ใช้ว่าสามารถหาโครงสร้างโมเดลที่เหมาะสมได้ซึ่งเป็นปัญหาสำคัญ ถ้าระบบมีความสัมพันธ์ไม่เป็นเชิงเส้น

- พบว่าไม่มีข้อมูลใดที่จะสมบูรณ์โดยไม่ถูกรบกวนจาก noise ซึ่งนำมาพิจารณา
- ระบบจะแปรไปตามเวลา จะเป็นปัญหาถ้าโมเดลที่สร้างขึ้นไม่แปรตามเวลา
- เป็นเรื่องยากในการที่จะวัดค่าสัญญาณที่เปลี่ยนไป

2.3 ศัพท์เทคนิคที่ใช้ในการหาค่าพารามิเตอร์

- System Identification เป็นการหาแบบจำลองทางคณิตศาสตร์จากข้อมูลจากกาทดลอง
- System ระบบเป็นลักษณะทางกายภาพที่ได้มาจากข้อมูลที่ทำการทดลองโดยอ้างถึงกระบวนการ โดยจะทำการพิจารณาตามทฤษฎี Identification ซึ่งจำเป็นตอการนำไปสู่ข้อมูลที่ตั้งสมมติฐาน ระบบเป็นสัญลักษณ์แทนกระบวนการทางคณิตศาสตร์ของกระบวนการ ในทางปฏิบัติจะใช้ข้อมูลที่ระบบไม่รู้จักและสามารถนำมันมาใช้ได้ แม้ว่าจะไม่รู้จักมันแต่สามารถใช้โดยตรงสำหรับข้อมูลที่ใช้กับคอมพิวเตอร์ ในการที่จะประยุกต์ใช้เทคนิค Identification ไม่จำเป็นต้องรู้จักระบบ แต่จะใช้แนวคิดของระบบในกาทำสมการผลต่างภายใต้สภาวะที่เปลี่ยนแปลง
- Model Structure โครงสร้างโมเดล บางครั้งเราจะใช้ Nonparametric model ในกรณีที่ต้องการอธิบายโมเดลโดย Curve, Function หรือตาราง ผลตอบสนองของความถี่จะเป็น Curve ที่บรรจุข้อมูลเกี่ยวกับระบบ ในหลายๆกรณีที่เกี่ยวข้องกับ Parametric model โดยตรงก็จะบ่งลักษณะโดยพาราเมตริกซ์เวกเตอร์ที่แทนด้วยสัญลักษณ์ ϕ โมเดลจะแทนด้วย $\mu(0)$ เมื่อ ϕ เปลี่ยนไปตามค่าที่เป็นไปได้เราจะได้เซตของโมเดลและโครงสร้างโมเดล

- Identification Method เป็นวิธีการต่างๆในการใช้หาค่าพารามิเตอร์ระบบ
- Experimental Condition เป็นเงื่อนไขการทดลอง จะเป็นตัวบอกว่าจะทำการ Identification อย่างไร และจะรวมถึงการเลือกค่าสัญญาณต่างๆในกระบวนการชักตัวอย่าง การคัดข้อมูลนำมาประมาณค่าพารามิเตอร์
- Dynamic System เป็นระบบที่มีการเปลี่ยนแปลง ซึ่งคุณสมบัติของระบบจะแปรค่าตามเวลา
- Estimation data เป็นเซตข้อมูลที่กำหนดมาให้เหมาะสมกับโมเดล
- Validation data เป็นเซตข้อมูลที่ใช้สำหรับตรวจคุณภาพโมเดลในที่นี้จะ รวมถึงการ simulate โมเดล และคำนวณหาค่าความผิดพลาดขอองโมเดล
- Model set หรือ Model structure เป็นโมเดลที่ปรับค่าพารามิเตอร์
- Parameter estimation เป็นการหาค่าพารามิเตอร์ที่ดีที่สุด ปัญหาการหาพารามิเตอร์ระบบคือต้องการหาโครงสร้างโมเดลและค่าพารามิเตอร์ที่ถูกต้อง
- Parametric Identification Method เป็นเทคนิคที่ใช้ในการหาพารามิเตอร์ของโครงสร้างโมเดลที่กำหนดขึ้นมา และค่าที่ได้มาต้องสัมพันธ์กับค่าเอาท์พุทที่วัดได้
- Model validation เป็นกระบวนการตรวจสอบความถูกต้องโมเดล โดยใช้หลักการ twisting and turning ทำการปรับค่าจนได้ค่าที่ถูกต้อง คุณสมบัติที่สำคัญคือโมเดลต้องสามารถให้ค่าเซตข้อมูลที่ถูกต้อง

2.4 ขั้นตอนพื้นฐานในการหาค่าพารามิเตอร์ระบบ

ปัญหาการหาค่าพารามิเตอร์ระบบคือหาโมเดลของระบบจกาข้อมูลอินพุทเอาท์พุทที่ได้จากการวัด

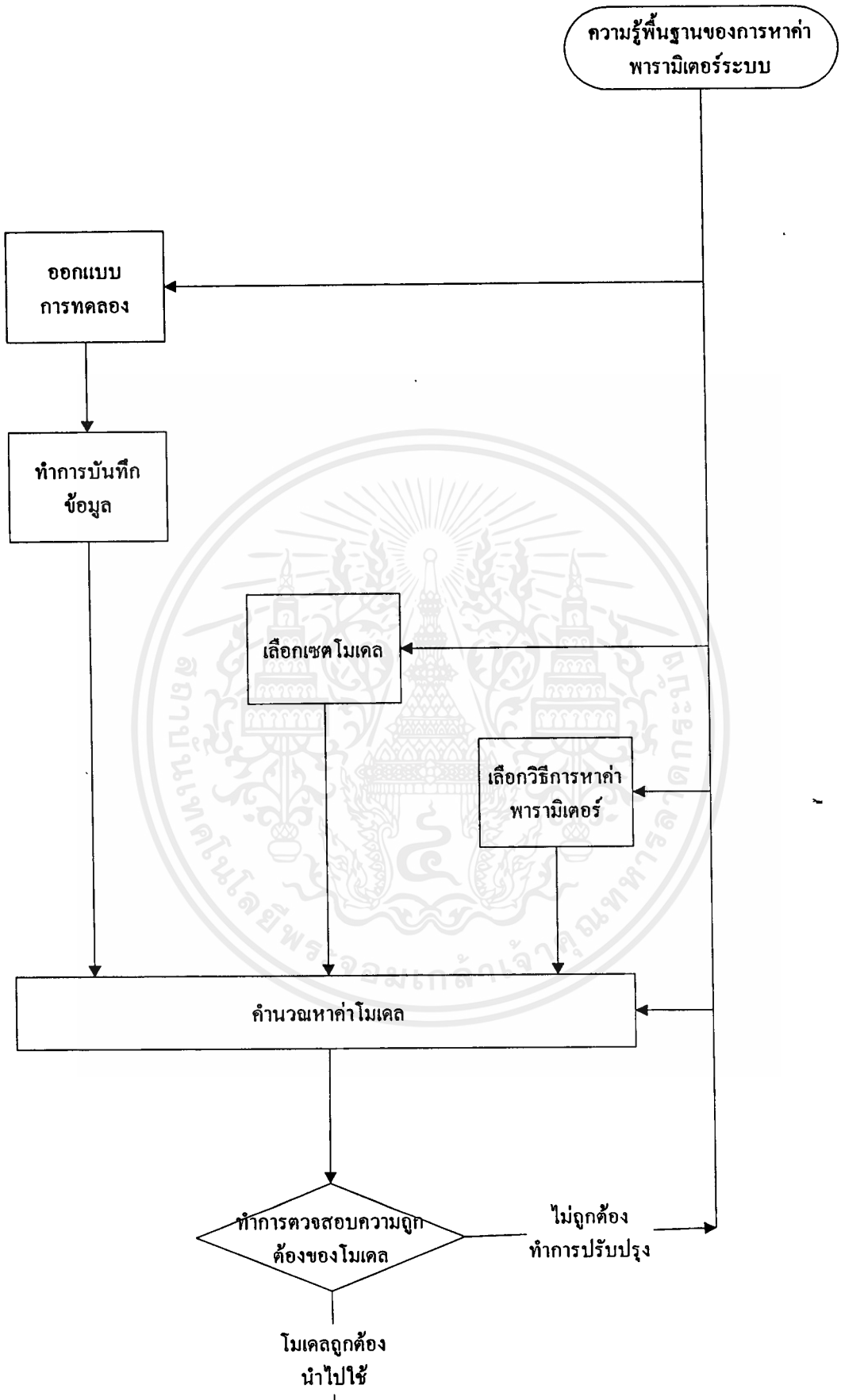
กระบวนการในการหาโมเดลของ Dynamic System จากข้อมูลอินพุทเอาท์พุทที่ได้จากการวัดจะประกอบด้วย

1. ข้อมูลอินพุทเอาท์พุท
2. โครงสร้างแบบจำลอง
3. เกณฑ์ในการเลือกโมเดล และวิธีการหาค่าพารามิเตอร์ระบบ

กระบวนการหาค่าพารามิเตอร์เพื่อจะเลือกโครงสร้างแบบจำลอง การคำนวณหาค่าแบบจำลองที่ดีที่สุด และการหาโมเดลที่มีคุณสมบัติที่ต้องการสามารถหาได้โดย

1. ออกแบบการทดลองและเก็บข้อมูลอินพุทเอาท์พุทของกระบวนการ เพื่อทำการหาค่าพารามิเตอร์
2. พิจารณาข้อมูลทั้งหมดเพื่อดูแนวทางของข้อมูล
3. เลือกและอธิบายโครงสร้างโมเดล

4. คำนวณโมเดลที่ดีที่สุดในโครงสร้างโมเดลที่ขึ้นอยู่กับอินพุทเอาท์พุทและเกณฑ์ในการเลือกโมเดล วิธีการทำนายค่าผิดพลาด Maximum Likelihood หรือ Instrumental Variable ก็จะถูกอ้างถึง สำหรับ parametric model
5. หาคุณสมบัติของแบบจำลองที่เลือกมา คำนวณหาฟังก์ชันต่างๆของโมเดล พิจารณาค่า ซีโรโพล ทำการ simulate โมเดล รวมทั้งการแปลงระบบต่อเนื่องไปเป็นระบบไม่ต่อเนื่อง
6. ถ้าโมเดลถูกต้องก็หยุดการทำงานเพียงเท่านี้ แต่ถ้าไม่ก็กลับไปทำงานในขั้นตอนต่างๆที่ผ่านมา เช่นถ้าต้องการเลือกโมเดลใหม่ก็กลับไปทำงานในขั้นตอนที่ 3 ถ้าต้องการเปลี่ยนวิธีการหาค่าพารามิเตอร์ก็กลับไปขั้นตอนที่ 4 หรือบางทีอาจต้องย้อนไปยังขั้นตอนที่ 1 และ 2 เพื่อพิจารณาอินพุทเอาท์พุทใหม่



โฟลวชาร์ตแสดงขั้นตอนการหาค่าพารามิเตอร์ระบบ

2.5 สาเหตุการเกิดข้อผิดพลาดในการหาค่าพารามิเตอร์

- สัญญาณทดสอบมีขนาดใหญ่เกิน Plant จริงอาจรับไม่ได้วิธีการหาค่าพารามิเตอร์ไม่ค่อยแน่นอน
- สัญญาณ Disturbance ทำให้เกิดความเสียหาย
- โมเดลไม่ได้เตรียมไว้สำหรับสัญญาณ Disturbance
- เกี่ยวข้องกับขนาดของ procedure
- ไม่ได้มีการตรวจสอบความถูกต้องของโมเดล

2.6 สัญญาณอินพุท (Input Signal)

สัญญาณอินพุทที่ใช้ในการทดลองหาค่าพารามิเตอร์จะมีผลกระทบต่อค่าพารามิเตอร์ที่หาได้ เราสามารถแบ่งชนิดของสัญญาณอินพุทได้เป็น

1. **Step Function** จะมีลักษณะคือจะมีค่าเป็นศูนย์เมื่อเวลาน้อยกว่าศูนย์ และมีค่าเป็นแอมปลิจูดที่ต้องการเมื่อเวลามากกว่าศูนย์

$$u(t) = \begin{cases} 0 & t < 0 \\ u_0 & t \geq 0 \end{cases} \qquad (2.1)$$

อินพุทที่เป็นสเตปจะให้ข้อมูลเกี่ยวกับผลตอบสนองของระบบ

2. **Psudorandom binary sequence (PSRB)** เป็นสัญญาณที่เปลี่ยนแปลงระหว่างสองระดับ มีลักษณะเป็นพัลส์สี่เหลี่ยม พัลส์แต่ละลูกจะมีความกว้างไม่เท่ากันเปลี่ยนไปตามสัญญาณแรนดอม PRSB จะมีลักษณะเช่นเดียวกับ White noise
3. **Autoregressive moving average sequence** จะได้ข้อมูลจากคอมพิวเตอร์
4. **Sum of sinusoid** เป็นสัญญาณอินพุทที่สามารถเขียนได้ในรูป

$$u(t) = \sum_{k=1}^{n} a_k(\sin\omega_k t + \varphi_j) \qquad 0 \leq \omega_1 < \omega_2 < \mathrm{K} < \omega_n \leq \pi \qquad (2.2)$$

โดยสามารถเลือกแอมปลิจูด a_i ความถี่ ω_i และค่าเฟส φ_i

2.7 การเลือกสัญญาณอินพุทสำหรับการหาค่าพารามิเตอร์ระบบ

การที่ค่า Prediction Error มีค่าเข้าใกล้ศูนย์ไม่ได้หมายความว่าค่าพารามิเตอร์ของโมเดลจะถูกต้อง

ตัวอย่างเช่น ในระบบที่เป็นดีสครีตแสดงด้วยสมการ

$$y(t+1) \quad = \quad -a_1 y(t) + b_i u(t) \tag{2.3}$$

และพิจารณาโมเดลที่จำลองขึ้น มีลักษณะดังนี้

$$\hat{y}(t+1) \quad = \quad -\hat{a}_1\, y(t) + \hat{b}_l\, u(t) \tag{2.4}$$

โดย $\hat{y}(t+1)$ เป็นเอาท์พุทของโมเดลที่จำลองขึ้นโดยค่าผิดพลาดจะเป็นไปตามสมการ

$$\in(t+1) \quad = \quad y(t+1) - \hat{y}(t+1) \quad = \quad \left[\hat{a}_1 - a_1\right] y(t) + \left[b_1 - \hat{b}_1\right] u(t) \quad = \quad 0 \tag{2.5}$$

และค่า $y(t)$ สามารถเขียนในรูป $u(t)$ เป็น

$$y(t) \quad = \quad \frac{b_1 q^{-1}}{1 + a_1 q^{-1}} u(t) \tag{2.6}$$

จากสมการข้างบนจะได้

$$\begin{aligned} (\alpha_0 + \alpha_1 q^{-1}) u(t) \quad &= \quad 0 \\ b_1 - \hat{b} = \alpha_0 \quad ; \quad & b_1 \hat{a}_1 - a_1 \hat{b}_1 \end{aligned} \tag{2.7}$$

จากสมการเชิงอนุพันธ์จะได้

$$u(t) \quad = \quad z^t \quad = \quad e^{ST_S t} \tag{2.8}$$

โดยค่า T_S เป็นคาบการสุ่มตัวอย่าง และได้

$$(z\alpha_0 + \alpha_1) z^{t-1} \quad = \quad 0 \tag{2.9}$$

หาค่า z จากสมการคุณลักษณะได้เป็น

$$(z\alpha_0 + \alpha_1) = 0 \tag{2.10}$$

จะได้ค่า z เป็น

$$z = -\frac{\alpha_1}{\alpha_0} = e^{\sigma T_s} \tag{2.11}$$

และ

$$u(t) = e^{\sigma T_s t} \tag{2.12}$$

ในกรณีที่อินพุทเป็นสเตป $u(t)$ เป็นค่าคงที่ หรือค่า $\sigma = 0$ จะได้ความสัมพันธ์

$$-\alpha_0 = \alpha_0 \Rightarrow \frac{b_1}{1+a_1} = \frac{\hat{b}_1}{1+\hat{a}_1} \tag{2.13}$$

ซึ่งค่า a_1 กับ $\hat{a}_1$ และ b_1 กับ $\hat{b}_1$ ไม่จำเป็นต้องเท่ากัน แต่จะมีค่า steady- state gain เท่ากัน ทำให้การหาค่าพารามิเตอร์ไม่ถูกต้อง

ในกรณีที่อินพุทเป็นสัญญาณไซน์

$$u(t) = e^{j\omega T_s t} \tag{2.14}$$

จะได้

$$[e^{j\omega T_s t}\alpha_0 + \alpha_1]e^{j\omega T_s (t-1)} = 0 \tag{2.15}$$

จะได้ค่า

$$\alpha_1 = \alpha_0 = 0 \Rightarrow \hat{b}_1 = b_1 \quad , \quad \hat{a}_1 = a_1$$

ซึ่งจะเห็นว่าสัญญาณอินพุทแบบไซน์จะสามารถให้ค่าพารามิเตอร์ที่ถูกต้องได้ จึงสามารถสรุปได้ว่าการหาค่าพารามิเตอร์ที่ดีจะต้องการสัญญาณอินพุทที่ประกอบด้วยค่าองค์ประกอบทางความถี่จำ

นวมาก ส่วนใหญ่จะนิยมใช้สัญญาณ PSRB เพราะเป็นสัญญาณที่ประกอบด้วยองค์ประกอบทางความถี่จำนวนมาก

2.8 การหาค่าพารามิเตอร์แบบรีเคอร์ซีพ (Recursive Identification Method)

โดยทั่วๆไปแล้วเป็นเรื่องยากในการที่จะทำการหาค่าพารามิเตอร์ของโมเดล จึงจำเป็นต้องใช้วิธีการหาค่าพารามิเตอร์ของไดนามิคโมเดลโดยตรงจากการทดลอง

ไดนามิคโมเดล จะสามารถแบ่งได้เป็น 2 ประเภท

1. นอนพาราเมตริคซ์โมเดล (Nonparametric Model) ตัวอย่างเช่น ผลตอบสนองความถี่ ผลตอบสนองเสตป

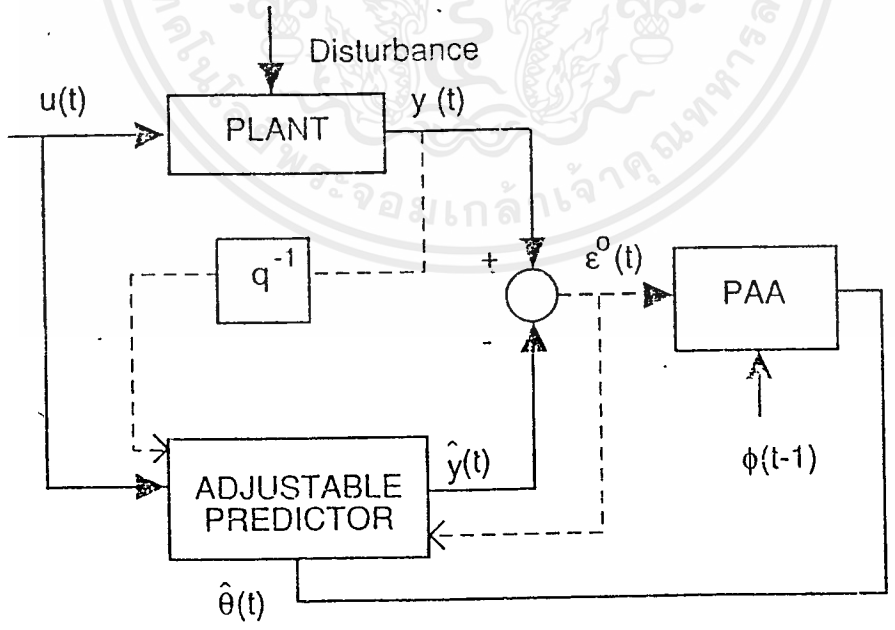
2. พาราเมตริคซ์โมเดล (Parametric Model) ตัวอย่างเช่น ทรานสเฟอร์ฟังก์ชัน หรือ สมการเชิงอนุพันธ์

พาราเมตริคซ์ไดนามิคโมเดล จะถูกนำมาใช้ในการหาค่าพารามิเตอร์ เพราะเป็นวิธีที่เหมาะสมที่สุดสำหรับการออกแบบและควบคุม

การหาค่าพารามิเตอร์ระบบมีอยู่ด้วยกันหลายวิธี แต่ควรจะเลือกวิธีที่สะดวกรวดเร็ว และได้ค่าที่ถูกต้องที่สุด การหาค่าพารามิเตอร์แบบรีเคอร์ซีพเป็นวิธีที่มีคุณสมบัติตามที่กล่าวมา

2.9 หลักการทำงานของการหาค่าพารามิเตอร์แบบรีเคอร์ซีพ

หลักการหาค่าพารามิเตอร์แบบรีเคอร์ซีพจะเป็นดังรูป



รูปที่ 2.1 โครงสร้างการทำงานของการหาพารามิเตอร์แบบรีเคอร์ซีพ

จากรูปที่ 2.1 เมื่อทำการป้อนค่าอินพุทเข้าไปก็จะเกิดเอาท์พุทขึ้นมา 2 ตัว เอาท์พุทของแพลนท์จริงกับเอาท์พุทของโมเดลที่ทำการชิมูเลตขึ้น ค่าความผิดพลาด (Prediction Error) $\varepsilon(k)$ ระหว่างค่าเอาท์พุทของแพลนท์จริงกับค่าเอาท์พุทที่ได้จากโมเดลที่ทำการชิมูเลตขึ้นจะถูกป้อนให้กับตัวพารามิเตอร์อะแดปเทชั่นอัลกอริทึม (Parameter Adaptation Algorithm) PAA ซึ่งแต่ละคาบที่ทำการชักตัวอย่าง PAA จะนำค่าผิดพลาดมาพิจารณาแล้วนำผลที่ได้ไปทำการปรับค่าพารามิเตอร์ของโมเดลที่ชิมูเลตขึ้นมา ซึ่งจะทำงานเช่นนี้ไปเรื่อยๆจนกว่าค่าผิดพลาดจะเป็นศูนย์ ส่วนที่นำมาปรับค่าพารามิเตอร์นอกเหนือจาก PAA แล้วก็ยังมีค่าเอาท์พุทของแพลนท์จริงก่อนหน้าอีกด้วย

ค่าพารามิเตอร์เวกเตอร์ใหม่จะพิจารณาจาก

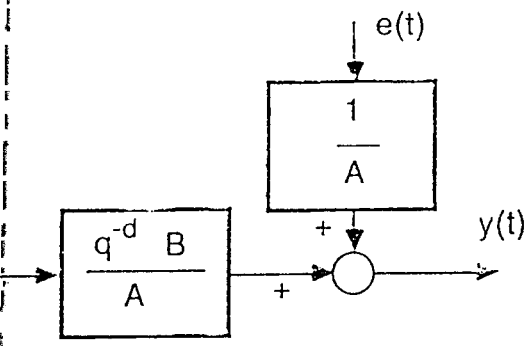
[พารามิเตอร์ใหม่] = [พารามิเตอร์ใหม่] + [ค่าอะแดปเทชันเกน]*[ค่า สัญญาณที่ได้จากการวัด]*[ค่าผิดพลาด]

การหาค่าพารามิเตอร์แบบรีเคอร์ซีพมีหลายวิธี ซึ่งทุกวิธีนั้นจะมีโครงสร้างของ PAA เหมือนกัน แต่จะต่างกันตรงการเลือกค่าอะแดปเทชันเกน ซึ่งมีองค์ประกอบดังนี้

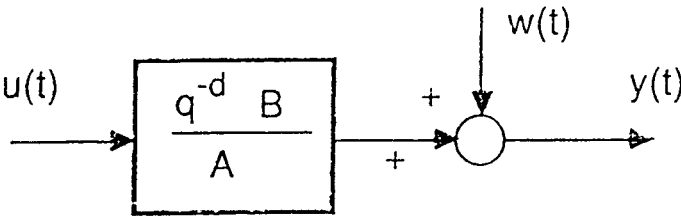
- Predictor Structure
- ส่วนประกอบต่างๆของ Observation Vector
- ขนาดของ Adjustable Parameter Vector และ Observation Vector
- แนวทางการเกิดค่า Prediction Error และ Adaptation Error

พบว่าการหาค่าพารามิเตอร์แบบรีเคอร์ซีพและเงื่อนไขต่างๆสามารถทำให้เกิดโครงสร้างโมเดลได้ 4 แบบ

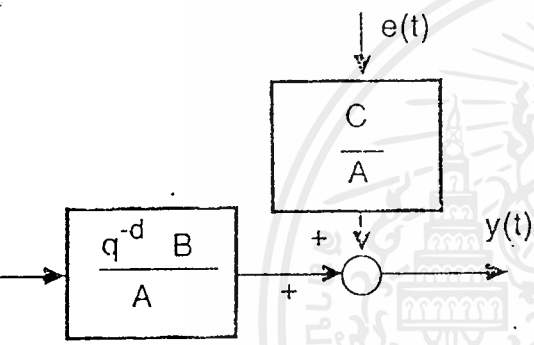
$A(q\text{-}1)y(t) = q^{-d}B(q^{-1})u(t) + e(t)$



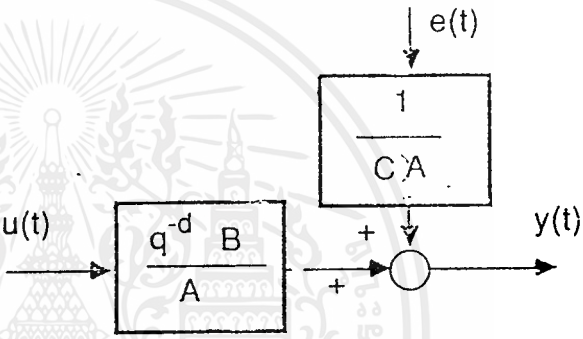
S2: $A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + A(q^{-1})w(t)$



$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + C(q^{-1})e(t)$



S4: $A(q\text{-}1)y(t) = q^{-d}B(q^{-1})u(t) + [1/C(q^{-1})]e(t)$



รูปที่ 2.2 รูปแบบของโครงสร้างโมเดลชนิดต่างๆ

พบว่าโครงสร้างโมเดลแบบ ARX (รูปบนขวา) สามารถใช้กับวิธี Recursive Least Square Method ได้เพราะว่าเป็นวิธีที่ใช้กับสัญญาณรบกวนที่เป็น white noise เท่านั้น

ARX Model สามารถแสดงได้ดัง

$$S: \quad A(q^{-1})y(k) = q^{-d}B(q^{-1})u(k) + e(k) \tag{2.16}$$

โดย $e(k)$ เป็น white noise ดังนั้นจึงสามารถใช้ Recursive Least Square Method กับ ARX Model ได้โดยไม่มีปัญหา

เหตุผลที่ ARX Model เป็นโมเดลแบบเดียวที่สามารถใช้วิธี Recursive Least Square เพราะ ARX มีลิเนียร์รีเกรสชัน (Linear Regression) ซึ่งเป็นลักษณะเฉพาะของวิธี Least Square โดยจะมีลักษณะเป็นไปตามสมการต่อไปนี้

$$A(q) = 1 + a_1 q^{-1} + \ldots + a_n q^{-n} \tag{2.17}$$

$$B(q) = b_0 + b_1 q^{-1} + \mathrm{K} + b_n q^{-n} \tag{2.18}$$

ดังนั้น

$$\hat{\theta}(k) = \left[\hat{a}_1 \quad \ldots \quad \hat{a}_n \quad \hat{b}_0 \quad \ldots \quad \hat{b}_n\right]^T \tag{2.19}$$

$$\varphi(k) = \left[-y(k-1) \quad \ldots \quad -y(k-n), \quad u(k-d) \quad \ldots \quad u(k-d-n)\right]^T \tag{2.20}$$

จะได้

$$y(k) = \theta^T \varphi(k) + e(k) \tag{2.21}$$

สิ่งที่จะพิจาณาต่อไปคือค่าเอาท์พุทของโมเดลที่ทำการจำลองขึ้นซึ่งจะเขียนได้ในรูป

$$\begin{aligned} \hat{y}(k|\theta) &= \left[1 - A(q)\right] y(k) + B(q)u(k) \\ &= \theta^T \varphi(k) \end{aligned} \tag{2.22}$$

จะพบว่าความสัมพันธ์ของพารามิเตอร์เป็นเชิงเส้น

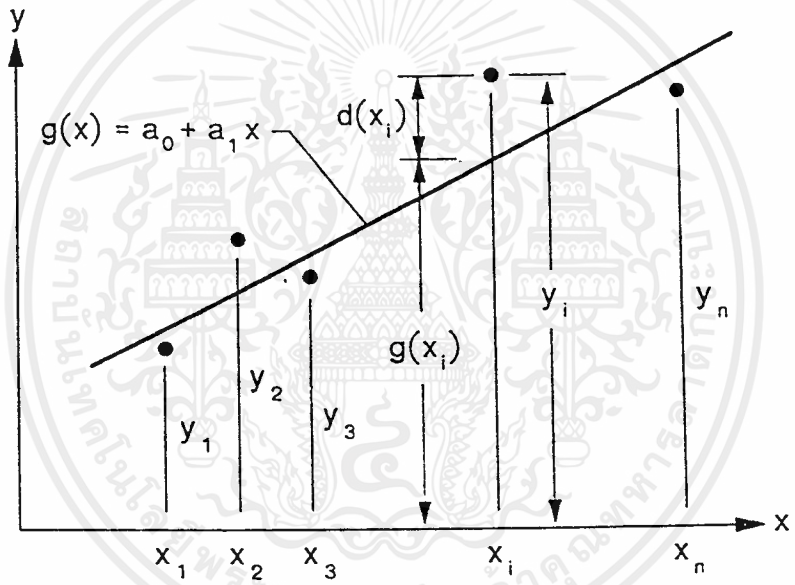
การหาค่าพารามิเตอร์แบบรีเคอร์ซีพจะทำการปรับปรุงค่าของพารามิเตอร์ โดยทำการลดค่าผลของสัญญาณรบกวน พบว่าไม่มีวิธีใดที่ใช้ได้ผลกับทุกโมเดล โดยในแต่ละโมเดลก็จะมีวิธีที่เหมาะสมที่สุดอยู่

2.10 การถดถอยแบบเชิงเส้น

การถดถอยแบบเชิงเส้น (linear regression) อาจจัดได้ว่าเป็นระเบียบวิธีที่ง่ายที่สุดที่ใช้ในการประดิษฐ์ฟังก์ชันเส้นตรงสำหรับชุดข้อมูลที่กำหนดมาให้ รูปที่ 2.3 แสดงชุดข้อมูลที่ประกอบด้วย $x_i, y_i,\ \ i = 1,2,...,n$ นั่นคือมีจำนวนข้อมูลทั้งสิ้น n ข้อมูล ในที่นี้เราจะประดิษฐ์สมการเส้นตรงในรูปแบบของฟังก์ชัน

$$g(x) \ = \ a_o + a_1 x \tag{2.23}$$

โดย a_o และ a_1 เป็นค่าคงตัวที่ไม่รู้ค่าและจำเป็นต้องคำนวณหาจากเงื่อนไขที่ว่า สมการเส้นตรงที่ประดิษฐ์ขึ้นมานี้จะก่อให้เกิดค่าความผิดพลาดโดยเฉลี่ยที่น้อยที่สุดจากข้อมูลทุกข้อมูลที่กำหนดมาให้



รูปที่ 2.3 การถดถอยเชิงเส้นโดยการประดิษฐ์ฟังก์ชันเส้นตรงจากชุดข้อมูลที่กำหนดมาให้

จากรูปนี้เราจะเห็นได้ว่า ณ ตำแหน่ง x_i ของข้อมูล i ใดๆค่าของ $g(x)$ ที่เราประดิษฐ์ขึ้นนั้นจะมีค่าที่แตกต่างไปจากค่าของข้อมูล y_i เท่ากับ $d(x_i)$ ที่ตำแหน่งนั้น นั่นหมายถึงว่า ค่าความผิดพลาด E ทั้งหมดที่เกิดขึ้นจากข้อมูลทั้งหมด n ข้อมูล อาจเขียนให้อยู่ในรูปแบบ ดังนี้

$$E = \sum_{i=1}^{n} \left[d(x_i) \right]^2 \tag{2.24}$$

ซึ่งในที่นี้เราทำการยกกำลังสมการของค่าแตกต่าง $d(x_i)$ ก็เพื่อกำจัดค่าที่มีเครื่องหมายเป็นลบ ดังนั้นในสมการจะให้ความหมายของค่าความผิดพลาดทั้งหมด สมการ (2.24) สามารถเขียนได้ว่า

$$d(x_i) \qquad E=\sum_{i=1}^{n}\left[y_i-g(x_i)\right]^2 \tag{2.25}$$

แทนสมการ (2.23) ที่ $x=x_i$ ลงในสมการ (2.25)

$$E=\sum_{i=1}^{n}\left[y_i-(a_o+a_1x_i)\right]^2 \tag{2.26}$$

จากสมการ (2.26) นี้เราสามารถคำนวณหาตัวไม่รู้ค่า a_0 และ a_1 ที่ต้องการได้โดยวิธีกำลังสองน้อยที่สุด (least – squares) ซึ่งทำจากวิธีการหาค่าต่ำสุดของค่าความผิดพลาดโดยเกี่ยวข้องกับตัวไม่รู้ค่า นั่นคือ

$$\frac{\partial E}{\partial a_o}=0 \tag{2.27a}$$

และ

$$\frac{\partial E}{\partial a_1}=0 \tag{2.27b}$$

และเงื่อนไขจากสมการ (2.27a) ให้ผลดังนี้

$$2\sum_{i=1}^{n}\left[y_i-(a_o+a_1x_i)\right](-1) = 0$$

$$\sum_{i=1}^{n}y_i-\sum_{i=1}^{n}a_o-\sum_{i=1}^{n}a_1x_i = 0$$

$$na_o+(\sum_{i=1}^{n}x_i)a_1 = \sum_{i=1}^{n}y_i \tag{2.28a}$$

และเงื่อนไขสมการ (2.27b) ให้ผลดังนี้

$$2\sum_{i=1}^{n}\left[y_i-(a_o+a_1x_i)\right](-x_i) = 0$$

$$\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} a_o x_i - \sum_{i=1}^{n} a_1 x_i^2 = 0$$

$$(\sum_{i=1}^{n} x_i) a_o + (\sum_{i=1}^{n} x_i^2) a_1 = \sum_{i=1}^{n} x_i y_i \qquad (2.28b)$$

ทั้งสองสมการนี้สามารถเขียนอยู่ในรูปแบบของเมตริกซ์ได้ดังนี้

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a_o \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix} \qquad (2.29)$$

ซึ่งเราสามารถใช้กฎของคราเมอร์ในการแก้ระบบสมการนี้เพื่อหาค่าคงตัว a_o และ a_1 ได้ดังนี้

$$a_o = \frac{(\sum_{i=1}^{n} y_i)(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i y_i)}{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2} \qquad (2.30a)$$

$$a_1 = \frac{(\sum_{i=1}^{n} x_i y_i) - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2} \qquad (2.30b)$$

ค่าคงตัว a_o และ a_1 ที่คำนวณได้นี้ เมื่อแทนกลับในสมการ (2.23) ก็จะได้สมการเส้นตรงที่แสดงการถดถอยเชิงเส้นแบบที่ต้องการ

2.11 Recursive Least Square

Recursive Least Square เป็นวิธีหนึ่งในการหาค่าพารามิเตอร์ของระบบแบบรีเคอร์ซีพ ซึ่งกุญแจสำคัญในการหาค่าพารามิเตอร์คือ PAA ซึ่งลำดับขั้นตอนของ Recursive Structure ก็จะทำการหาค่าพารามิเตอร์ใหม่จากค่าพารามิเตอร์เก่า ซึ่งจะขึ้นอยู่กับค่าที่วัดได้

จากรูปโครงสร้างการหาค่าพารามิเตอร์ระบบแบบรีเคอร์ซีพ ค่า Prediction Error สามารถหาได้จากสมการ

$$\varepsilon(k) = y(k) - \varphi^T(k)\hat{\theta} \qquad (2.31)$$

โดย

$$\begin{aligned} y(k) &= \sum_{i=1}^{n} a_i y(k-1) + \sum_{i=1}^{n} b_i u(k-i) \\ &= \varphi^T(k)\hat{\theta} \end{aligned} \tag{2.32}$$

เมื่อ $\varphi(k)$ เป็นค่าต่างๆที่วัดได้ (Observation Vector) สามารถเขียนได้ในรูป

$$\varphi^T(k) = \left[-y(k-1) \quad \ldots \quad -y(k-n), \quad u(k-1) \quad \ldots \quad u(k-n)\right] \tag{2.33}$$

และ $\hat{\theta}(k)$ เป็นค่าเวกเตอร์พารามิเตอร์ที่ได้จากโมเดลที่จำลองขึ้น

$$\hat{\theta}^T(k) = \left[\hat{a}_1 \quad \ldots \quad \hat{a}_n \quad \hat{b}_1 \quad \ldots \quad \hat{b}_n\right] \tag{2.34}$$

เราสามารถหาค่ากำลังสองต่ำสุดของค่าผิดพลาดจาก

$$J(\hat{\theta}) = \frac{1}{N}\sum_{k=1}^{N}\left[y(k) - \varphi^T(k)\hat{\theta}\right]^2 \tag{2.35}$$

ซึ่งรูปแบบสมการพัฒนามาจากการหาความสัมพันธ์เชิงเส้นของพารามิเตอร์และระบบสมการกำลังสอง นั่นก็คือฟังก์ชันกำลังสองของพารามิเตอร์ ดังนั้นเราจึงสามารถหาค่าต่ำสุดของกำลังสองของค่าผิดพลาดได้

เริ่มจากพารามิเตอร์ θ ต้องเป็นค่าที่ประมาณในการชักตัวอย่างลำดับที่ k ที่ทำให้เกิดค่ากำลังสองของค่าผิดพลาดน้อยที่สุด

ค่าผิดพลาดที่ทำให้เกิดค่ากำลังสองของค่าผิดพลาดน้อยที่สุดหาได้ จาก $\partial J/\partial\hat{\theta}$ จะได้

$$\partial J/\partial\hat{\theta} = -\frac{2}{N}\sum_{k=1}^{N}\left[y(k) - \hat{\theta}^T\varphi(k)\right]\varphi(k) = 0 \tag{2.36}$$

จากสมการข้างบนจะได้

$$\left[\theta^T\varphi(k)\right]\varphi(k) \;=\; \varphi(k)\varphi^T(k)\theta \tag{2.37}$$

ดังนั้น

$$\frac{1}{N}\left[\sum_{k=1}^{N}\varphi(k)\varphi^T\right]\hat{\theta} \;=\; \frac{1}{N}\sum_{k=1}^{N}y(k)\varphi(k) \tag{2.38}$$

และจะได้

$$\hat{\theta} \;=\; \frac{1}{N}\left[\sum_{k=1}^{N}\varphi(k)\varphi^T(k)\right]^{-1}\frac{1}{N}\sum_{k=1}^{N}\varphi(k)\,y(k) \tag{2.39}$$

ซึ่งเป็นรูปแบบของวิธี Least Square ในการหาค่าพารามิเตอร์

2.12 ลักษณะการหาค่าพารามิเตอร์แบบ Least Square

ลักษณะการประมาณค่าพารามิเตอร์แบบ Least Square จะเกี่ยวข้องกับการหาค่าพารามิเตอร์ของค่า Prediction Error ซึ่งเป็นไปตามนี้

ข้อมูลที่ได้จากการวัดสามารถแทนได้จากสมการ

$$y(k) \;=\; \varphi^T(k)\theta \;+\; v(k) \tag{2.40}$$

สำหรับลำดับ $v(k)$ อาจคิดว่า θ เป็นค่าจริงของค่าพารามิเตอร์เวกเตอร์ และจากสมการที่ผ่านมาและแทนค่า $R(N)=\frac{1}{N}\sum_{k=1}^{N}\varphi(k)\varphi^T(k)$ จะได้

$$\begin{aligned}\hat{\theta} \;&=\; [R(N)]^{-1}\frac{1}{N}\sum_{k=1}^{N}\varphi(k)[\varphi^T(k)\theta+v(k)] \\ &=\; \theta+[R(N)]^{-1}\frac{1}{N}\sum_{k=1}^{N}\varphi(k)\,y(k)\end{aligned} \tag{2.41}$$

ค่าของพารามิเตอร์ $\hat{\theta}$ จะทำการหาจากค่า θ และจะลู่เข้าสู่ θ เมื่อ N เข้าสู่อินฟินิตี้

ถ้าค่า $v(k)$ มีค่าน้อยมากเมื่อเทียบกับค่า $\varphi(k)$ ดังนั้นเทอม $[R(N)]^{-1}\frac{1}{N}\sum_{k=1}^{N}\varphi(k)v(k)$ จะมีค่าน้อย และค่าพารามิเตอร์ที่หาจากโมเดลที่จำลองขึ้นมาจะเข้าใกล้ θ เมื่อ N เข้าใกล้อินฟินิตี้และเราทำการหาพจน์ของ $v(k)$ ดังนี้ $\frac{1}{N}\sum_{k=1}^{N}\varphi(k)y(k)\rightarrow h^*$ เมื่อ N เข้าใกล้อินฟินิตี้

สำหรับการประมาณค่าแบบ Least Square ค่า $\hat{\theta}$ จะเข้าสู่ θ เมื่อค่า h^* มีค่าเป็นศูนย์จะเกิดขึ้นเมื่อ

1. $v(k)$ เป็นลำดับของตัวแปรสุ่มที่มีค่าเฉลี่ยเป็นศูนย์หรือเรียกว่า white noise ดังนั้น $v(k)$ จะไม่ขึ้นอยู่กับค่าปัจจุบันหรือค่าที่เกิดขึ้นหลังการชักตัวอย่างครั้ง $k-1$ และ $E\varphi(k)v(k)=0$
2. ลำดับของอินพุท $u(k)$ จะไม่ขึ้นอยู่กับค่า noiseที่มีค่าเฉลี่ย $v(k)$ ดังนั้น $\varphi(k)$ จะประกอบด้วยเทอมอินพุทและค่า $E\varphi(k)v(k)=0$

ถ้าเป็นไปตามลักษณะที่กล่าวมาก็สามารถหาค่าพารามิเตอร์โดยวิธี Least Square ได้

2.13 ขั้นตอนของวิธีการทำงานแบบรีเคอร์ซีพ (Recursive Algorithm)

ขั้นตอนของวิธีการทำงานแบบรีเคอร์ซีพ จะพิจารณาความสัมพันธ์ระหว่าง $\hat{\theta}(k)$ และ $\hat{\theta}(k-1)$ ด้วยวิธี Weighted Least Square ซึ่งเราสามารถเขียนแทนด้วยสมการ

$$\begin{aligned} J(\hat{\theta}) &= \frac{1}{N}(Y-Z\hat{\theta})^T W(Y-Z\hat{\theta}) \\ &= \frac{1}{N}\left[Y^TWY - Y^TWZ\hat{\theta} - \hat{\theta}^T Z^TWY + \hat{\theta}^T Z^TWZ\hat{\theta}\right] \end{aligned} \tag{2.42}$$

โดยที่

$$Y = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} \qquad Z = \begin{bmatrix} \varphi(1) \\ \vdots \\ \varphi(N) \end{bmatrix}$$

$N \quad vector \qquad\qquad N*(2n+1) \quad matrix$

โดย W เป็นไดโกนอลเมตริซ์ขนาด $N*N$

สำหรับเงื่อนไข Least Square

$$\frac{\partial J}{\partial \hat{\theta}} = \frac{1}{N}\left[-Z^T WY - Z^T WY + 2Z^T WZ\hat{\theta}\right] = 0 \qquad (2.43)$$

ดังนั้น

$$\frac{1}{N}\left[Z^T WZ\right]\hat{\theta} = \frac{1}{N}\left[Z^T WY\right] \qquad (2.44)$$

และ

$$\hat{\theta} = (Z^T WZ)^{-1} Z^T WY \qquad (2.45)$$

โดย W เป็น weighting factor

ขั้นตอนต่อไปเราจะทำการหาค่าของ

$$\hat{\theta} \leftarrow \hat{\theta}(k+1) \quad , \quad Z \leftarrow Z(k+1) \quad , \quad Y \leftarrow Y(k+1) \quad , \quad W \leftarrow W(k+1)$$

ดังนั้น

$$\begin{aligned}\hat{\theta}(k+1) &= \left[Z^T(k+1)W(k+1)Z(k+1)\right]^{-1} Z^T(k+1)W(k+1)Y(k+1) \\ &= \left[Z^T(k)W(k)Z(k) + w(k+1)\varphi(k+1)\varphi^T(k+1)\right]^{-1} * \\ &\quad \left[Z^T(k)W(k)y(k) + w(k+1)\varphi(k+1)y(k+1)\right]\end{aligned} \qquad (2.46)$$

ทำการหลีกเลี่ยงค่าอินเวอร์สโดย

$$\left[Z^T(k+1)W(k+1)Z(k+1)\right]^{-1} = F(k+1) \qquad (2.47)$$

ดังนั้น

$$F^{-1}(k+1) = Z^T(k+1)W(k+1)Z(k+1) \qquad (2.48)$$

และ

$$F^{-1}(k) = Z^T(k)W(k)Z(k) \tag{2.49}$$

จากสมการ (2.48) และ (2.49)จะได้

$$\begin{aligned} F(k+1) &= F(k) - A(k+1)\varphi^T(k+1)F(k) \\ &= \left[I - A(k+1)\varphi^T(k+1)\right]F(k) \end{aligned} \tag{2.50}$$

เมื่อ

$$A = \frac{w(k+1)F(k)\varphi(k+1)}{1+w(k+1)\varphi^T(k+1)F(k)\varphi(k+1)} \tag{2.51}$$

และจะได้

$$\hat{\theta}(k+1) = \hat{\theta}(k) + A(k+1)\left[y(k+1) - \varphi^T(k+1)\hat{\theta}(k) \right] \tag{2.52}$$

ถ้าให้

$$w(k+1) = \lambda^{-1}w(k) \quad and \quad P(k) = w(k)F(k)$$

ดังนั้น

$$P(k+1) = \frac{1}{\lambda}\left[P(k) - \frac{P(k)\varphi(k+1)\varphi'(k+1)P(k)}{\lambda+\varphi^T(k+1)P(k)\varphi(k+1)} \right] \tag{2.53}$$

และ

$$A(k+1) = \frac{P(k)\varphi(k+1)}{\lambda+\varphi^T(k)P(k)\varphi(k+1)} \quad (2.54)$$

ดังนั้นค่าพารามิเตอร์ที่ทำการปรับแต่งโดยวิธี Recursive Least Square สามารถหาได้จากสมการ

$$\hat{\theta}(k) = \hat{\theta}(k-1)+\frac{P(k-1)\varphi(k)}{\lambda+\varphi^T(k)P(k-1)\varphi(k)} \quad (2.55)$$

$$P(k) = \frac{1}{\lambda}\left[P(k-1)-\frac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{\lambda+\varphi^T(k)P(k-1)\varphi(k)}\right] \quad (2.56)$$

$$\varepsilon(k) = y(k)-\varphi^T(k)\hat{\theta}(k-1) \quad (2.57)$$

โดยที่ค่า λ มีชื่อเรียกว่า Forgetting Factor และค่า P(k) เป็นค่าอะแดปเทชันเกน

2.14 การเลือกค่า อะแดปเทชั่นเกน:

จากรูปแบบของค่าอินเวอร์สอะแดปเทชันเกน และใส่ค่า forgetting factor λ_1 และ λ_2 จะเขียนได้ในรูป

$$F(k+1)^{-1} = \lambda_1(k)F(k)^{-1}+\lambda_2(k)\varphi(k)\varphi(k-1)^T$$
$$0\le\lambda_1(k)\le 1 \quad 0\le\lambda_2(k)\le 2 \quad F(0)>0 \quad (2.58)$$

จากสมการจะพบว่า $\lambda_1<1$ จะทำให้ค่าอะแดปเทชันเกน เพิ่มขึ้น และค่า $\lambda_2>0$ จะทำที่ลดค่าอะแดปเทชันเกน ซึ่งค่าของอะแดปเทชันเกนจะมีรูปแบบดังสมการ

$$F(k+1) = \frac{1}{\lambda_1(k)}\left[F(k)-\frac{F(k)\varphi(k)\varphi^T(k)F(k)}{\frac{\lambda_1(k)}{\lambda_2(k)}+\varphi^T(k)F(k)\varphi(k)}\right] \quad (2.59)$$

โดยค่า forgetting factor ที่เหมาะสมควรจะมีค่าดังนี้ $\lambda_2=1$ และค่า $0.95\le\lambda_1\le 0.99$ เพราะว่าค่านี้จะหลีกเลี่ยงการลดลงอย่างรวดเร็วของอะแดปเทชันเกน และจะทำให้ค่าพารามิเตอร์เข้าสู่ค่าที่ถูกต้องได้เร็ว

โดยค่า forgetting factor ที่เหมาะสมควรจะมีค่าดังนี้ $\lambda_2 = 1$ และค่า $0.95 \le \lambda_1 \le 0.99$ เพราะว่าค่านี้จะหลีกเลี่ยงการลดลงอย่างรวดเร็วของอะแดปเทชันเกน และจะทำให้ค่าพารามิเตอร์เข้าสู่ค่าที่ถูกต้องได้เร็ว

โดยค่า อะแดปเทชั่นเกน มี 2 ทางเลือกที่เป็นไปได้คือ

1. $F = \alpha I \;\; ; \;\; \alpha > 0$
2. $F > 0$ (เป็น Positive definite matrix)

2.15 การเลือกค่าเกนเริ่มต้น (F_0)

ค่าเริ่มต้นของอะแดปเทชันเกนจะหาได้จากสมการ

$$F(0) = \frac{1}{\delta} I = (GI) I \quad 0 \le \delta \le 1 \tag{2.60}$$

ในตอนเริ่มต้นการทำงานยังไม่มีข้อมูลเริ่มต้นของพารามิเตอร์อยู่ ถ้าค่าอะแดปเทชันเกนมีค่าต่ำจะไม่สามารถทำงานได้ ดังนั้นต้องเลือกค่าค่าเกนเริ่มต้นให้มีค่าสูง เช่น โดยค่า อะแดปเทชั่นเกน ที่เลือกที่มีค่า $GI = 10000$

เมื่อเริ่มทำงาน และมีข้อมูลของพารามิเตอร์ในระบบแล้ว การทำงานแบบรีเคอร์ซีพจะทำการลดค่าอะแดปเทชันเกนลงไปเรื่อยๆ เมื่อค่าพารามิเตอร์เข้าใกล้ค่าที่ถูกต้อง

2.16 การตรวจสอบความถูกต้องของโมเดล (Model Validation)

สำหรับค่าความผิดพลาดเป็นลำดับของ white noise การหาค่าพารามิเตอร์ที่ถูกต้อง หมายความว่าการทำให้ค่าเอาท์พุทของโมเดลที่จำลองขึ้นมีค่ากำลังสองของค่าผิดพลาดน้อยที่สุด ค่าผิดพลาดที่เป็น white noise จะไม่เกี่ยวข้องกับตัวแปรอื่น

หลักเกณฑ์การตรวจสอบความถูกต้องของโมเดลมีดังต่อไปนี้

1. โครงสร้างของ plant และ disturbance ที่ทำการเลือกถูกต้อง
2. วิธีการหาค่าพารามิเตอร์ที่เลือกเหมาะสม
3. ค่าดีกรีของโพลิโนเมียลและค่าดีเลย์ที่เลือกถูกต้อง

ค่าผิดพลาด (prediction error) ที่เป็น white noise จะมีลักษณะดังนี้

$$\lim_{t \to \infty} E\{\in(t) \in (t-i)\} = 0 \quad i = 1,2,3,\ldots; \quad -1,-2,-3,\ldots$$

วิธีการตรวจสอบโมเดลมีขั้นตอนดังนี้

1. สร้างค่าอินพุทเอาท์พุทให้กับโมเดล
2. สร้างค่าผิดพลาดให้กับโมเดลที่จำลอง (อย่างน้อย 100 ข้อมูล)
3. ทำ Whiteness Test กับค่า Prediction Error

Whiteness Test

เป็นการทดสอบว่าค่า Prediction Error เป็น White noise หรือไม่ ซึ่งทดสอบได้โดย

$$R(0) = \frac{1}{N}\sum_{t=1}^{N}\in^2(t) \quad ; \quad RN(0) = \frac{R(0)}{R(0)} = 1$$

$$R(i) = \frac{1}{N}\sum_{t=1}^{N}\in(t)\in(t-i) \quad ; \quad RN(i) = \frac{R(i)}{R(0)} \quad ; \quad i=1,2,3,$$

(2.61)

เมื่อค่า Prediction Error เป็น White noise และจำนวนครั้งการสุ่มมากพอ $(N \to \infty)$ ดังนั้น $RN(0)=1\,;\, RN(i)=0; i \geq 1$ แต่ในความเป็นจริงไม่สามารถจะเกิดขึ้น จึงทำได้แค่เพียงกำหนดให้มันอยู่ภายใต้เงื่อนไขที่ยอมรับได้

$$RN(0) = 1; RN(i) \leq \frac{2.17}{\sqrt{N}} \quad ; \quad i \geq 1 \qquad (2.62)$$

เมื่อ N เป็นจำนวนการสุ่มตัวอย่าง

ถ้าการหาค่าพารามิเตอร์ระบบซับซ้อน(มีจำนวนค่าพารามิเตอร์มาก) ก็ต้องทำการเลือกโมเดลให้มีค่า $RN(i)$ น้อยที่สุด และโมเดลที่ดีควรมีโครงสร้างไม่ยุ่งยาก

การตรวจสอบโมเดลจะสมบูรณ์ต่อเมื่อทำการป้อนอินพุทเอาท์พุทสำหรับหาค่าพารามิเตอร์แล้วทำการตรวจสอบความถูกต้อง ในกรณีที่ค่า Prediction Error มีค่าน้อยมากเมื่อเทียบกับเอาท์พุท ก็ไม่ต้องทำ Whiteness Test เพราะว่าค่าของ noise ต่ำมากจนสามารถตัดทิ้งไปได้

บทที่ 3
ทฤษฎีและหลักการหาพารามิเตอร์ของระบบจริง

3.1 แอกทีฟฟิลเตอร์

ฟิลเตอร์คือ วงจรที่ใช้สำหรับกรองความถี่เฉพาะที่ต้องการใช้ผ่านออกมาได้เท่านั้น ส่วนความถี่อื่นๆที่ไม่ต้องการผ่านจะถูกลดทอนจนหมดไป วงจรฟิลเตอร์ที่ใช้งานมีอยู่ 2 ลักษณะคือ แบบที่เป็นวงจรพาสซีฟ (passive) และแบบแอกทีฟ (active) วงจรที่เป็นแบบพาสซีฟนั้น จะใช้เพียงอุปกรณ์ประเภทตัวต้านทาน ตัวเก็บประจุ และตัวเหนี่ยวนำเท่านั้น แต่ในวงจรประเภทแอกทีฟจะรวมถึงทรานซิสเตอร์และออปแอมป์ด้วย

- ในที่นี้จะเน้นเฉพาะวงจรที่มีออปแอมป์เป็นส่วนประกอบเท่านั้น สำหรับอุปกรณ์แอกทีฟชนิดอื่นสามารถนำมาใช้เป็นแอกทีฟฟิลเตอร์ได้เช่นกันแต่ในที่นี้จะไม่กล่าวถึง
- ปกติในการสร้างวงจรใช้งานนั้นมักพยายามหลีกเลี่ยงไม่ใช้ตัวเหนี่ยวนำ เนื่องจากค่อนข้างหายาก ราคาแพง และมีขนาดใหญ่ไม่สะดวกแก่การใช้งาน

3.1.1 วงจรกรองความถี่ต่ำแบบพื้นฐาน

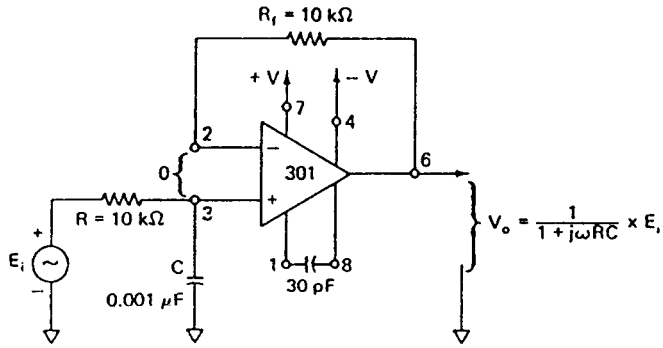
ในรูปที่ 3.1 เป็นวงจรแอกทีฟฟิลเตอร์ กรองสัญญาณความถี่ต่ำผ่านแบบที่ใช้งานกันโดยทั่วไป วงจรประกอบด้วย R, C และออปแอมป์ ซึ่งจากวงจรมีอัตราขยาย 1 เท่า กำหนดให้ R_f มีค่าเท่ากับ R และแรงดันออฟเซตมีค่าเป็น 0 โวลต์ ดังนั้นแรงดันที่ขา 2 เท่ากับแรงดันที่ขา3 สำหรับแรงดันที่ขา 2 ซึ่งมีค่าเท่ากับ V_O นั้น จะทำให้แรงดันคร่อม C มีค่าเป็น V_O ด้วยเช่นกัน ถ้าเราพิจารณาในส่วน และ C ที่ต่อกับ E_i เราจะได้สมการเป็น

$$V_O = \frac{1/j\omega C}{R + 1/j\omega C} * E_i \tag{3.1}$$

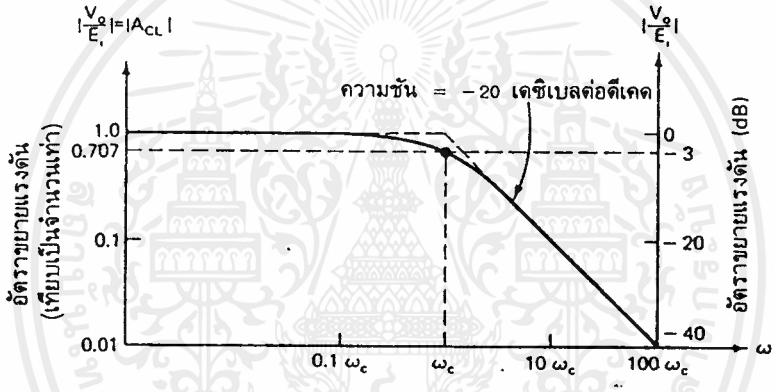
เมื่อ ω คือ ความถี่เชิงมุมของ Ei มีหน่วยเป็น เรเดียน/วินาทีและ j มีค่าเป็นค่ารีแอกแตนของ C มีค่าเป็น $1/\mathrm{j}\omega\mathbf{C}$

จากสมการนำมาเขียนใหม่เป็นอัตราขยายแบบลูบปิดได้คือ

$$A_{CL} = \frac{V_O}{E_i} = \frac{1}{1 + j\omega C} \tag{3.2}$$



(ก) ฟิลเตอร์ชนิดกรองความถี่ต่ำผ่านที่มีค่า –20 เดซิเบลต่อดีเคด



(ข) กราฟแสดงการตอบสนองความถี่ของวงจรนี้

รูปที่ 3.1 กราฟแสดงการตอบสนองความถี่ของวงจรฟิลเตอร์ชนิดกรองความถี่ต่ำผ่านที่มีค่า –20 เดซิเบลต่อดีเคด

ในรูปที่ 3.1 จากที่ได้กล่าวมาแล้วว่าเป็นวงจรประเภทกรองความถี่ต่ำผ่าน ต่อไปเราจะทำการวิเคราะห์สมการนี้ จากสมการที่ 2 ที่ได้แสดงถึงค่าของอัตราขยายแบบปิดลูป A_{CL} ซึ่งมีค่าเปลี่ยนแปลงไปตามความถี่ ถ้าพิจารณาที่ความถี่ต่ำ ω มีค่าเข้าใกล้ศูนย์ นำไปแทนในสมการ จะได้ขนาด $A_{CL} = 1$ และที่ความถี่สูง ω เข้าใกล้อินฟินิตี้ ขนาดของ $A_{CL} = 0$

ในรูปที่ 3.1 ข เป็นกราฟที่แสดงขนาดของ A_{CL} เปรียบเทียบกับ ω แสดงให้เห็นว่าที่ความถี่สูงกว่าความถี่คัตออฟ ω_C ขนาดของ A_{CL} จะมีค่าลดลงด้วยอัตรา 20 เดซิเบลต่อดีเคด ดังนั้นสรุปได้ว่าเฉพาะในช่วงที่มีความถี่สูงกว่า ω_C อัตราการขยายแรงดันจะมีค่าลดลง 10 เท่า เมื่อความถี่มีค่าเพิ่ม 10 เท่าเช่นกัน

ขั้นตอนในการออกแบบ

1. เลือกค่าความถี่คัตออฟที่จะใช้งาน ω_C หรือ f_C

2. เลือกค่าตัวเก็บประจุ C_3 ของวงจร โดยควรมีค่าอยู่ระหว่าง0.001 ถึง 0.1 ไมโครฟารัด

3. จากค่า C_3 ที่เลือกไว้ จะทำให้ได้ค่า C_1 และ C_2 ที่จะใช้งาน คือ

$$C_1 = \frac{1}{2}C_3$$

4. คำนวณค่าความต้านทาน R ที่ต้องใช้ได้จากสมการ

$$R = \frac{1}{\omega_C C_3} \tag{3.6}$$

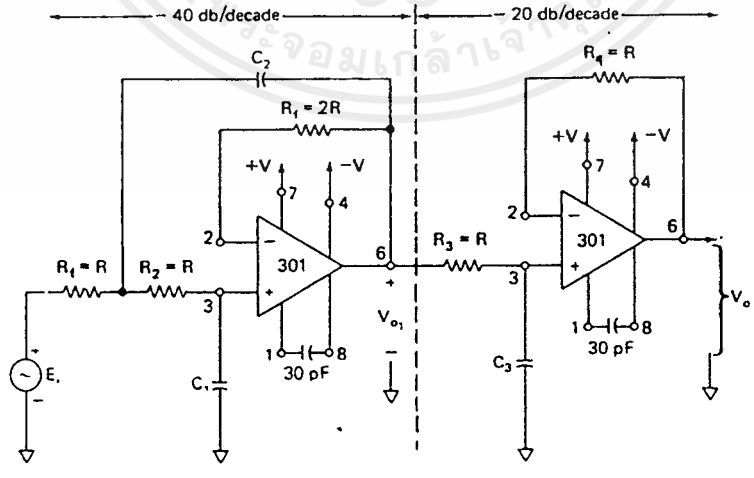
ในการเลือกค่าความต้านทาน R ที่จะใช้งานควรอยู่ในช่วง 10 ถึง 100 กิโลโอห์ม ถ้าคำนวณได้ค่าที่นอกเหนือจากค่าในช่วงนี้ออกไป ควรปรับเปลี่ยนค่า C_3 เพื่อให้ได้ค่า R ที่เหมาะสม

5. กำหนดให้ค่าความต้านทาน R_1, R_2, R_3 ทั้ง 3 ตัวมีค่าเท่ากับ R

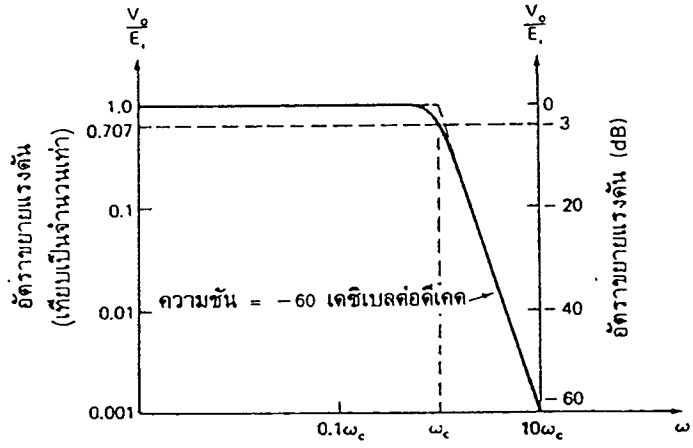
6. กำหนดให้ $R_{f1} = 2R$ และ $R_{f2} = R$

สาเหตุที่เราต้องมีวงจรความถี่ต่ำผ่านในการหาพารามิเตอร์ของระบบจริงนั้นก็เพราะว่าเราต้องการที่กำจัดสัญญาณรบกวนที่ความถี่สูงๆออกไปและป้องกันการเกิดความผิดพลาดเมื่อป้อนสัญญาณที่มีความถี่ต่างกันเข้าไปเมื่อทำการชักตัวอย่างผลของสัญญาณที่ได้ออกมาจะเหมือนกัน โดยในโครงงานได้เลือกความถี่คัตออฟที่ 10 KHz จึงคำนวณค่าความต้านทานและค่าตัวเก็บประจุได้ดังนี้

R = 15Kohm C1 = 0.5nf C2 = 2nf C3 = 1nf



(ก) วงจรบัตเตอร์เวิร์ทที่ให้ค่า –60 เดซิเบลต่อดีเคด



(ข) กราฟแสดงการตอบสนองความถี่ของวงจรบัตเตอร์เวิร์ทนี้

รูปที่ 3.2 การออกแบบวงจรบัตเตอร์เวิร์ทที่ให้ค่า –60 เดซิเบลต่อดีเคด และกราฟแสดงการตอบสนองความถี่

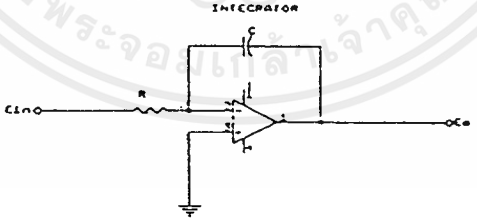
3.2 พื้นฐานวงจร A/D (วงจรเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอล)

วงจร A/D ที่ใช้กันอยู่ทั่วๆไปมีหลายแบบคือ

3.2.1 แบบที่ใช้วงจรเปรียบเทียบแบบขนานหรือแบบ " แฟลช " (Parallel Comparator Simultaneous " Flash" A/D Converter)

วงจรเอทูดีนี้ใช้หลักการง่ายๆอีกทั้งเป็นวิธีที่รวดเร็วที่สุด โดยใช้เวลาในการแปลงระดับนาโนวินาทีทีเดียว แต่วงจรเอทูดีแบบนี้ก็ยังมีข้อเสียคือถ้าเราต้องการวงจรที่มีความละเอียดสูงก็ต้องใช้วงจรเปรียบเทียบสูงขึ้น เช่นถ้าต้องการความละเอียดขนาด 8 บิท ก็ต้องใช้วงจรเปรียบเทียบมากถึง 255 ตัว

3.2.2 วงจรเอทูดีที่ใช้การอินทริเกรต



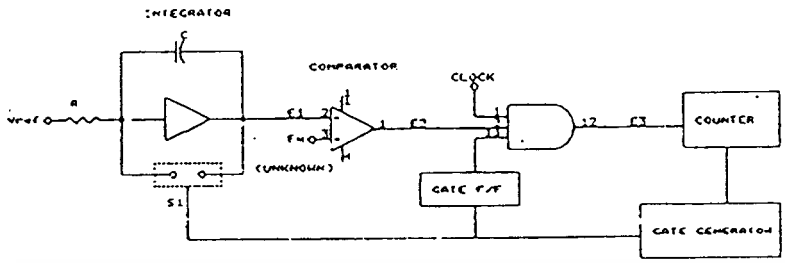
รูปที่ 3.3 วงจรอินทิเกรทเตอร์

วงจรแปลงสัญญาณเอทูดีที่ใช้เทคนิคการอินทริเกรตมี 2 แบบคือ

3.2.2.1 แบบสโลปเดี่ยวหรือแบบแรมป์ (Single Ramp หรือ Single Slope A/D Converter)

วงจรเอทูดีแบบนี้ประกอบด้วยวงจรกำเนิดสัญญาณแรมป์ วงจรเปรียบเทียบ วงจรนับ BCD หรือนับเลขฐานสองซึ่งวงจรลักษณะนี้มักนำไปใช้ในการเปลี่ยนเวลาเป็นขนาดของสัญญาณ (Time to

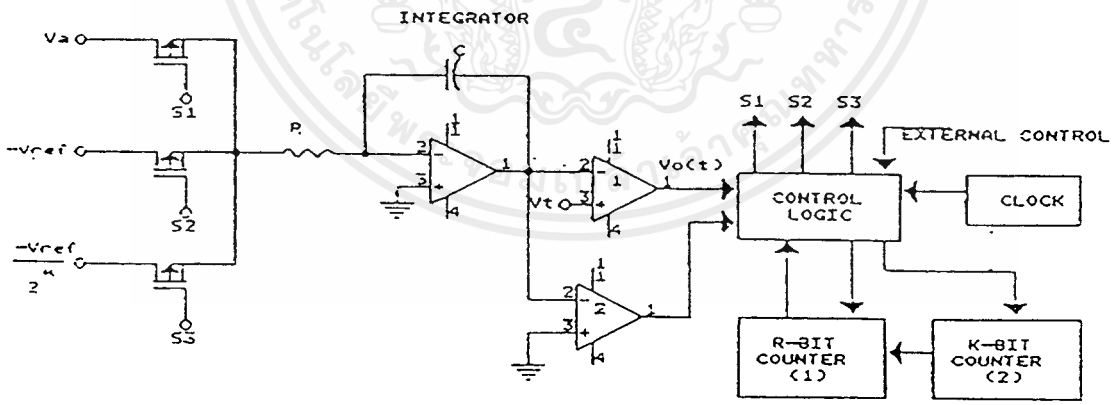
Amplitude Conversion) หรืออาจใช้ในดิจิตอลโวลต์มิเตอร์แต่ไม่ใช้กับงานที่ต้องการความถูกต้องสูง เนื่องจากการเปลี่ยนแปลงในแหล่งกำเนิดสัญญาณแรมป์ขึ้นกับอุณหภูมิและผลตอบสนองต่ออินพุต ทำให้ไม่มีความคงที่ ดังนั้นจึงมีการเปลี่ยนแปลงให้ดีขึ้นกลายเป็นแบบสโลปคู่ (Dual - Slope)



รูปที่ 3.4 วงจรแปลง A/D แบบสโลปเดียว

3.2.2.2 แบบสโลปคู่ (Dual – Slope A/D Converter)

วงจรส่วนใหญ่คล้ายกับแบบสโลปเดี่ยว แต่มีสวิตซ์อินพุตเพิ่มขึ้นเพื่อทำการเลือกระหว่างแรงดันอินพุตกับอินพุตอ้างอิง(วงจรเปรียบเทียบต่อขาสัญญาณอินพุตกลับกันแบบสโลปเดี่ยว) ข้อดีของวงจรเปลี่ยนสัญญาณแบบสโลปคู่นี้คือ ความถูกต้อง ราคาถูก เสถียรภาพทางอุณหภูมิดี ข้อเสียคือความเร็วต่ำ ในการเปลี่ยนสัญญาณ 1 ครั้งอาจใช้เวลาถึง 100 ms (ในขณะที่แฟลชใช้เวลาเพียง 30 ns)



รูปที่ 3.5 วงจรแปลง A/D แบบสโลปคู่

3.2.3 วงจรเปลี่ยนสัญญาณเอทูดีที่ใช้วงจรนับและดีทูเอประกอบกัน

3.2.3.1 แบบวงจรนับเดี่ยว (Single Counter)

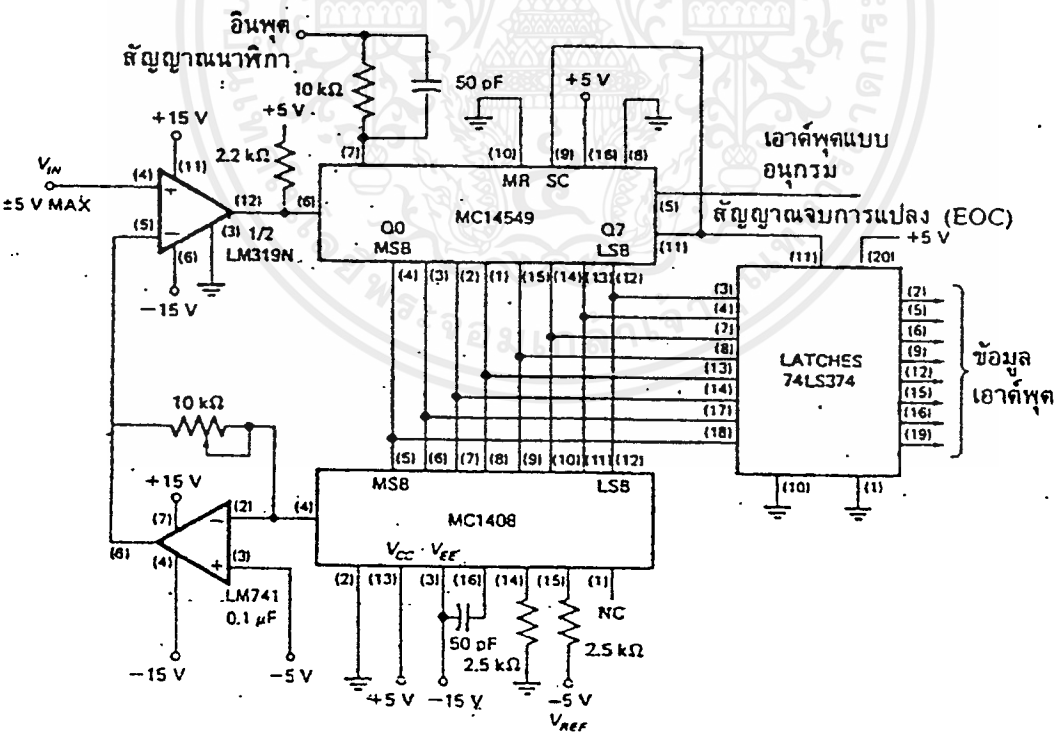
เป็นวงจรเปลี่ยนสัญญาณเอทูดีที่ใช้วงจรนับและดีทูเอประกอบกัน โดยปกติแล้วสัญญาณแรมป์เชิงเส้นจะประกอบด้วยสัญญาณขั้นบันไดเล็กๆจำนวนมากที่เกิดจากการต่อเอาท์พุตของวงจรนับเข้ากับวงจรแปลงสัญญาณแปลงดีทูเอ โดยขนาดของขั้นบันไดแต่ละขั้นขึ้นอยู่กับจำนวนบิทหรือความละเอียดของวงจรเอทูดีนั้นๆ

3.2.3.2 แบบแทร็กกิ้ง (Tracking A/D Converter)

การทำงานจะคล้ายกับแบบใช้วงจรนับเดี่ยว แต่การนับจะไม่ได้เริ่มจากศูนย์แต่จะทำการนับขึ้นหรือลงจากค่าล่าสุดไปยังค่าใหม่แล้วแต่ว่าแรงดันอินพุตในรอบใหม่มีค่าสูงกว่าหรือต่ำกว่าค่าที่แล้ว ข้อดีของเอทูดีแบบนี้คือทำงานได้เร็วขึ้น

3.2.4 แบบใช้การประมาณค่า(Successive Approximation A/D Converter)

วงจรเอทูดีแบบนี้มีข้อได้เปรียบทางด้านความละเอียด เพราะความละเอียด n บิต สามารถกำหนดได้จากสัญญาณนาฬิกา n ลูก ถ้าเราต้องการความละเอียด 8 บิตจะต้องการพัลส์ของสัญญาณนาฬิกา 8 ลูกในขณะที่ใช้แบบวงจรนับต้องใช้พัลส์ถึง 256 ลูก วงจร SA (Successive Approximation) นี้แสดงไว้ในรูปที่ หัวใจของวงจรคือ Successive Approximation Register (SAR) เช่นเบอร์ MC14549 ที่มีการทำงานดังต่อไปนี้



รูปที่ 3.6 วงจรเปลี่ยนสัญญาณเอทูดีแบบ Successive Approximation

เมื่อเริ่มทำการเปลี่ยนสัญญาณ พัลส์ลูกแรกจะทำการส่งบิตที่มีนัยสำคัญสูงสุดไปยังดีทูเอเบอร์ MC1408 โดย SAR จะรอสัญญาณจากวงจรเปรียบเทียบ LM319 ซึ่งทำการตรวจสอบว่าเอาท์พุตของวง

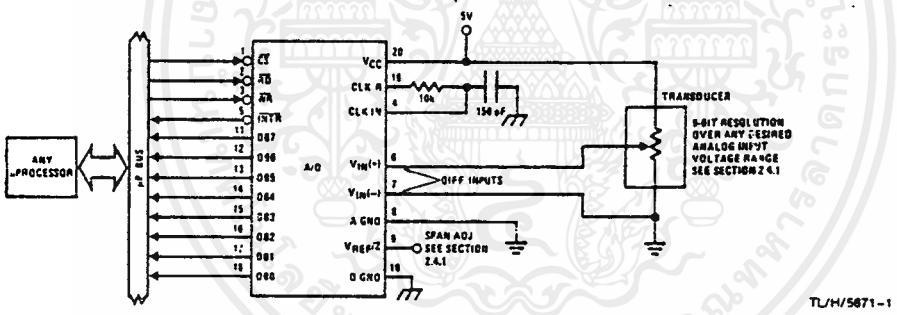
จรดีทูเอมากกว่าหรือน้อยกว่าแรงดันอินพุต Vin ถ้าเอาท์พุตของวงจรเปรียบเทียบมีระดับ "high" เอาท์พุตของดีทูเอจึงต่ำกว่า Vin_SAR จะทำการเก็บบิตที่มีนัยสำคัญสูงสุดไว้ ถ้าเอาท์พุตของวงจรเปรียบเทียบเป็นระดับ "low" เอาท์พุตของวงจรเปรียบเทียบจึงมากกว่า Vin_SAR จะทำการรีเซตบิตที่มีนัยสูงสุดนั้น

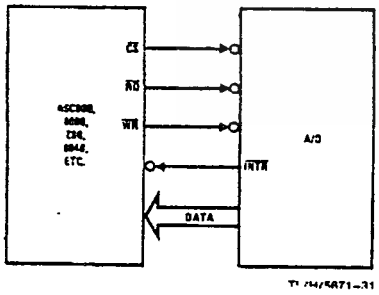
พัลส์ลูกต่อมาก็ทำเช่นเดียวกัน โดยบิตที่ได้คือ บิตที่มีนัยสำคัญรองลงมา SAR ทำงานแบบนี้ไปจนถึงบิตที่มีนัยสำคัญต่ำสุด แต่ละบิตใช้สัญญาณนาฬิกาลูกเดียวครบทุกบิต แล้ว SAR ทำการส่งสัญญาณ EOC (End of Conversion) ออกไปสัญญาณ EOC เป็นตัวบอกว่าสายสัญญาณเอาท์พุตที่ขนานกันมาทุกเส้นมีข้อมูลดิจิตอลของสัญญาณอินพุตครบถ้วนแล้ว ถ้าสัญญาณ EOC ถูกต่อไปยังอินพุตที่เป็นจุดเริ่มการเปลี่ยนสัญญาณ การเปลี่ยนสัญญาณก็จะเกิดขึ้นอย่างต่อเนื่อง

วงจรแปลงเอทูดีชนิดนี้มีความเร็วและความละเอียดสูง จึงเป็นวงจรที่นิยมนำมาใช้กันอย่างแพร่หลาย

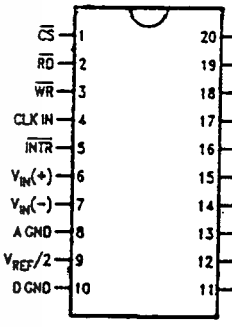
การเลือกไอซี A/D มาใช้งานในวงจรการหาค่าพารามิเตอร์ของระบบ

ในวงจรการหาค่าพารามิเตอร์ของระบบได้เลือกใช้ไอซีเบอร์ ADC0805 ซึ่งเป็น A/D 8 บิตแบบ Successive Approximation โดยมีวงจรแลตช์ภายในที่เหมาะสำหรับต่อเข้ากับ data bus ของไมโครโปรเซสเซอร์ โดยจะมี conversion time = 100 μs





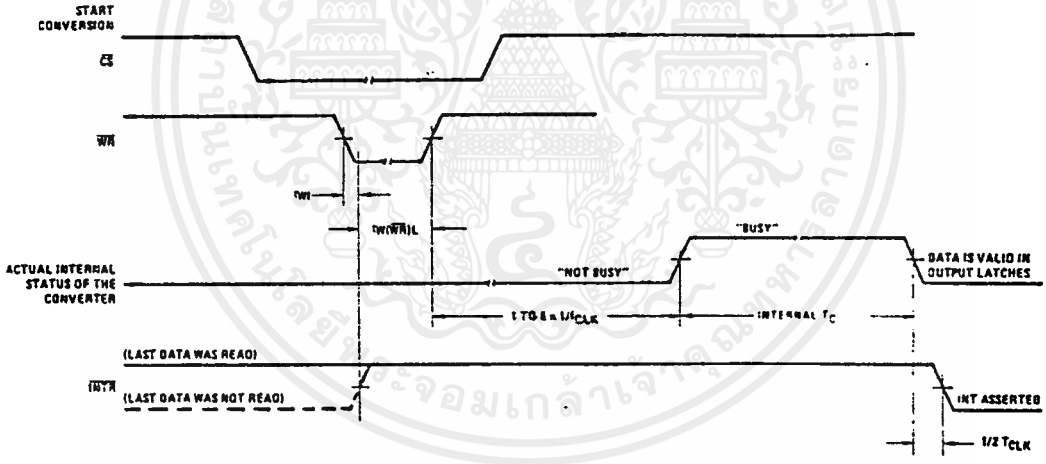
Error Specification (Includes Full-Scale, Zero Error, and Non-Linearity)			
Part Number	Full-Scale Adjusted	$V_{REF}/2 = 2.500\ V_{DC}$ (No Adjustments)	$V_{REF}/2$ = No Connection (No Adjustments)
ADC0801	±¼ LSB		
ADC0802		±½ LSB	
ADC0803	±½ LSB		
ADC0804		±1 LSB	
ADC0805			±1 LSB



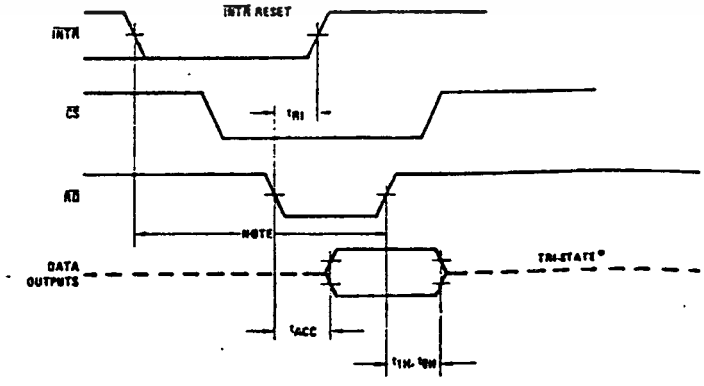
รูปที่ 3.7 แสดงลักษณะการต่อ ADC0805 เข้ากับไมโครโปรเซสเซอร์

ความหมายของขาต่างๆของ ADC0805

- $\overline{CS}$ Chip Select (active low) ไมโครโปรเซสเซอร์จะต้องให้ค่า "0" กับขานี้เมื่อต้องการติดต่อกับ ADC0805
- $\overline{WR}$ WRITE(active low) ถ้าสัญญาณเป็น "0" ADC0805 จะทำการ sampling ค่ามาเก็บไว้
- $\overline{RD}$ READ (active low) ถ้าสัญญาณเป็น "0" ADC0805 จะส่งค่าที่ทำการแปลงเป็น digital แล้วให้กับไมโครโปรเซสเซอร์
- VCC Digital Supply Voltage เป็นขาที่ต่อกับแหล่งจ่ายไฟให้กับ A/D ซึ่งมีขนาด 5V
- $V_{REF/2}$ ต่อกับค่า $V_{ref/2}$ ซึ่ง A/D จะใช้เพื่อนำไปเปรียบเทียบกับ input แต่สำหรับไอซี ADC0805 ไม่จำเป็นต้องต่อขา $V_{ref/2}$ นี้
- DB0-DB7 เป็นเอาท์พุตพอร์ทของข้อมูลต่อเข้ากับ DATA BUS ของไมโครโปรเซสเซอร์ โดย DB0 เป็น LSB (Least Significant Bit) และ DB7 เป็น MSB (Most Significant Bit)
- A GND และ D GND เป็น Ground ของวงจร
- $V_{in}(+)$ และ $V_{in}(-)$ เป็นขาอินพุต
- CLK R และ CLK IN ต่อกับวงจรให้กำเนิดสัญญาณ clock ของ A/D



Output Enable and Reset $\overline{INTR}$



รูปที่ 3.8 แสดง Timing Diagram ของ ADC0805

3.3 การใช้งาน Timer

Timer/Counter ของ 89C51 ประกอบด้วยรีจีสเตอร์ที่สามารถเลือกการทำงานเป็น Timer หรือ Counter อย่างใดอย่างหนึ่งจำนวน 2 ตัว แต่ละตัวมีขนาด 16 บิต คือ Timer 0 และ Timer 1 ตามลำดับรีจีสเตอร์ที่ใช้เป็น Timer 0 ประกอบด้วย TL0,TH0 ขนาด 8 บิต ส่วน Timer 1 ประกอบด้วย TL1,TH1 ขนาด 8 บิตเช่นกัน ในการทำงานเป็น Timer หรือ Counter จะมีรายละเอียดที่แตกต่างกันออกไปดังนี้

Timer : ใช้รีจีสเตอร์ Timer1 หรือ 2 เป็นตัวนับจำนวน Machine Cycle และเนื่องจากใน 1 Machine Cycle ใดๆของ MCS51 ประกอบไปด้วย 12 คาบสัญญาณออสซิลเลเตอร์ (oscillator period) ดังนั้นอัตราเร็วในการนับจึงมีค่าเป็น 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้

Counter : ค่าในรีจีสเตอร์ที่ใช้เป็น Counter ที่ถูกเลือกใช้งานจะถูกเพิ่มค่าทีละหนึ่งเมื่อมีการเปลี่ยนสถานะจาก Logic 1 เป็น 0 ซึ่งตรวจจับได้จากขา T0 และ T1 ในขณะนั้น

นอกจากจะสามารถเลือกการทำงานของ Register ให้เป็น Timer หรือ Counter ได้แล้วในแต่ละการทำงานเป็น Timer หรือ Counter ของ Timer 0 หรือ Timer 1 ก็ยังมีการทำงานที่แยกย่อยลงไปอีกถึง 4 แบบ(โหมด 0, 1, 2 และ 3)ตามความเหมาะสมของการใช้งาน การทำงานย่อยทั้ง 4 ประเภทของ Timer 0 และ Timer 1 มีรายละเอียดดังนี้

ผู้ใช้สามารถเลือกการทำงานให้เป็น Timer หรือ Counter อย่างใดอย่างหนึ่งโดยการกำหนดค่าบิต C/T ใน Register ใช้งานเฉพาะ TMOD หากบิตนี้มีค่าเป็น 0 หมายถึงเลือกให้ Register ทำงานเป็น Timer แต่ถ้าบิตนี้ เป็น 1 หมายถึง เลือกให้ทำงานเป็น Counter

Register TMOD

TMOD7	TMOD6	TMOD5	TMOD4	TMOD3	TMOD2	TMOD1	TMOD0
GATE	C/T	M1	M0	GATE	C/T	M1	M0

บิต TMOD7 ถึง TMOD4 สำหรับ Timer 1

บิต TMOD3 ถึง TMOD0 สำหรับ Timer 0

1. GATE บิตเลือกการควบคุมให้ Register สำหรับใช้เป็น Timer หรือ Counter ทำงานโดยควบคุมจากฮาร์ดแวร์หรือซอฟท์แวร์ดังนี้

- เมื่อบิต TRx (TR0,TR1) และ GATE ถูกเซต Timer หรือ Counter จะทำงานต่อเมื่อสถานะที่ขา INTx (INT0,INT1) มีค่าเป็น 1 (ควบคุมจากฮาร์ดแวร์)
- เมื่อบิต GATE ถูกเคลียร์ Timer หรือ Counter จะทำงานก็ต่อเมื่อบิต TRx ถูกเซตโดยไม่ขึ้นกับสถานะสัญญาณที่ขา INTx (ควบคุมจากซอฟท์แวร์)

2. C/T บิตเลือกการทำงานของ Register สำหรับใช้เป็น Timer หรือ Counter ดังนี้

- 0 หมายถึงทำงานเป็น Timer

- 1 หมายถึงทำงานเป็น Counter (นับจำนวนพัลส์ภายนอกที่ขา Tx)

3. M1 บิตสำหรับเลือกโหมดการทำงานของ Timer 0 หรือ Timer 1

M0 บิตสำหรับเลือกโหมดการทำงานของ Timer 0 หรือ Timer 1

ถ้าบิต M1 และ M0 เป็น

0 0 Mode 0 : Timer หรือ Counter ขนาด 13 บิต

0 1 Mode 1 : Timer หรือ Counter ขนาด 16 บิต

1 0 Mode 2 : Timer หรือ Counter ขนาด 8 บิต

1 1 Mode 3 : Timer 0

Register TL0 ใช้เป็น Timer หรือ Counter ขนาด 8 บิตที่ควบคุมการทำงานได้จากบิตของ Timer 0 เอง

Register TH0 ใช้เป็น Timer ขนาด 8 บิต ที่ควบคุมการทำงานได้จากบิตของ Timer 1

Timer 1 หยุดทำงาน (หยุดนับ)

โหมดการทำงานของ Timer / Counter

โหมด 0 ใช้ Register ขนาด 8 บิต เป็นตัวนับ โดยมีการเพิ่มค่าครั้งละ 1 ทุกครั้งที่นับสัญญาณได้ครบ 32 ครั้ง

ในการทำงานโหมดนี้ Register ที่ใช้นับไม่ว่าจะถูกกำหนดการทำงานเป็น Timer หรือ Counter จะถูกใช้เพียง 13 บิตเท่านั้น (8 บิตใน Register THx รวมกับ 5 บิตใน Register TLx) โดยในขณะที่ค่าใน Register ถูกเปลี่ยนจากเดิมที่เป็น 1 ทั้งหมดเป็น 0 ทั้งหมด (เกิด Overflow) จะทำให้บิต TF0 หรือ TF1 ถูกเซต สัญญาณที่ใช้ในการนับจะผ่านเข้ามายัง Register ที่ทำการนับได้ก็ต่อเมื่อ บิต TRx = 1 และบิต GATE = 0

การเซตให้บิต GATE เป็น 1 จะเป็นการกำหนดให้ Register ที่ใช้เป็น Timer หรือ Counter ถูกควบคุมการทำงานโดยสัญญาณที่ขา INTx (ควบคุมการนับโดยฮาร์ดแวร์) MCS51ไปใช้วัด (บิต TRx ต้องเป็น 1) ส่วนการเคลียร์ให้บิต GATE เป็น 0 จะมีผลให้การควบคุมการทำงานของ Register ที่ใช้เป็น Timer หรือ Counter ได้โดยซอฟท์แวร์ ซึ่งทำได้โดยการเซต หรือเคลียร์บิต TRx ด้วยคำสั่งในโปรแกรม

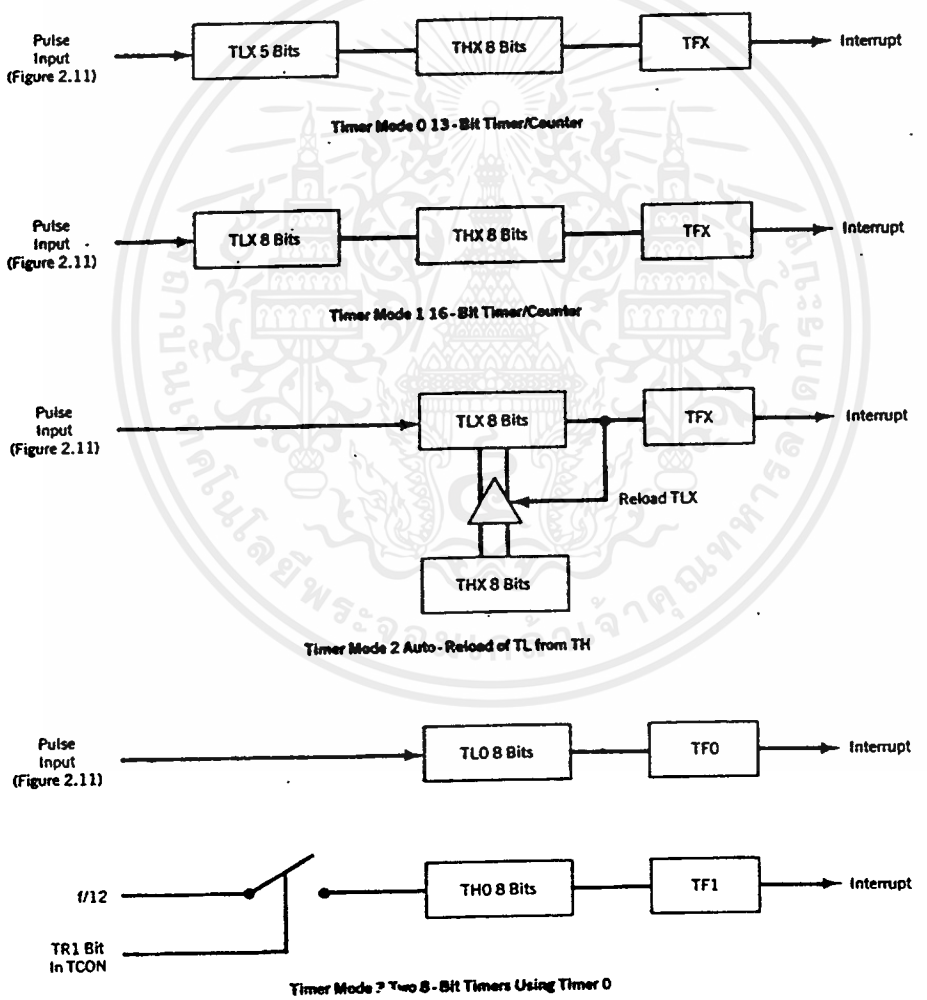
โหมด 1 การทำงานจะเหมือนในโหมด 0 ทุกประการ แต่ Register Timer หรือ Counter ถูกใช้งานครบทั้ง 16 บิตแทนที่จะเป็น 13บิต นั่นคือ Timer หรือ Counter ในโหมดนี้มีขนาด 16 บิต

โหมด 2 จะใช้ Register เพียง 8 บิต ซึ่งก็คือ Register TLx และมีการโหลดค่าเองด้วยค่าใน THx เมื่อเกิด Overflow ใน Register TLx โดยที่ค่าในริจิสเตอร์ THx นี้ สามารถกำหนดได้ล่วงหน้าในprogram และจะไม่เปลี่ยนแปลงเมื่อถูกโหลดไปไว้ใน Register TLx

การทำงานในโหมดนี้ มีไว้เพื่อใช้สร้างสัญญาณ Interrupt ที่มีคาบเวลาคงที่ หรือใช้สร้างฐานเวลาให้แก่ MCS51

โหมด 3 Register Timer 1จะไม่มีการนับ ซึ่งมีผลเหมือนกับให้ค่าบิต TR1=0 แต่สำหรับ Timer 0 จะมีการทำงานดังต่อไปนี้

Timer 0 ในโหมด 3 จะบังคับให้ Register ใช้งานเฉพาะ TL0 ของ Timer 0 ถูกใช้เป็น Timer หรือ Counter ขนาด 8 บิต โดยสามารถควบคุม TL0 ได้ จากบิต C/T ,GATE ,TR0,INT0 และการเกิด Oerflow ของ Register TL0 จะมีผลไปเซตบิต TF0 ส่วน Register ใช้งานเฉพาะ TH0 ของ Timer 0 จะถูกบังคับให้ใช้งานเป็น Timer เป็นอย่างเดียว โดยสามารถควบคุมการทำงานได้จากบิต TR1 และ TF1 ของ Timer 1 และจะเซตบิต TF1 เมื่อเกิด Overflow นั่นคือ ขณะนี้ Register TH0 จะควบคุมการเกิด Interrupt ของ Timer 1



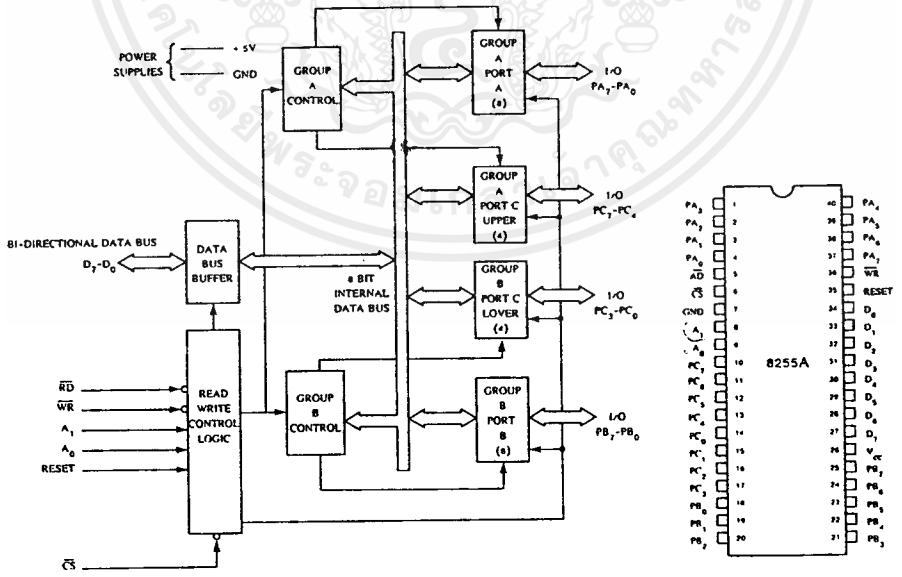
รูปที่ 3.9 การทำงานของ Timer ในโหมดต่างๆ

3.4 ทฤษฎีและการใช้งาน 8255 เบื้องต้น

ไมโครโปรเซสเซอร์นั้นนอกจากติดต่อกับหน่วยความจำโดยการนำข้อมูลไปเก็บหรืออ่านข้อมูลใดๆออกจากหน่วยความจำแล้วตัว CPU เองอาจจะติดต่อกับส่วนประกอบภายนอกอื่น ๆ อีกด้วย เช่น การรับคีย์บอร์ด การแสดงผล หรือแม้แต่การนำ CPU ไปควบคุมอุปกรณ์ต่าง ๆ นั้น CPU ต้องติดต่อ (รับหรือส่งข้อมูล) โดยผ่านทางอินพุตหรือเอาท์พุตโดยใช้ TTL บางเบอร์มาใช้เป็นพอร์ทสำหรับ CPU ได้ แต่ทั้งนี้การใช้ไอซี TTL มีข้อจำกัดหลายอย่าง เช่น ในกรณีที่จำเป็นต้องใช้พอร์ทหลายๆพอร์ท เพราะต้องติดต่อกับอุปกรณ์ภายนอกหลายจุด จึงต้องใช้ไอซีเหล่านี้หลายตัวและอาจทำให้ยากในการออกแบบวงจรอีกทั้งไม่สามารถจะเปลี่ยนแปลงการทำงานให้แตกต่างไปจากเดิมที่ได้ออกแบบไว้แล้ว ดังนั้นผู้ผลิต CPU ในตระกูลต่าง ๆ จึงมักจะผลิตไอซีประเภท LSI ที่ทำหน้าที่เป็น พอร์ทมาเพื่อใช้ CPU เบอร์นั้นๆได้สะดวกซึ่งจะทำให้การรับข้อมูลมีความน่าเชื่อถือและยังสามารถเปลี่ยนแปลงชนิดของพอร์ท (จากอินพุตเป็นเอาท์พุตหรือจากเอาท์พุตเป็นอินพุต) ได้ง่ายโดยการควบคุมจาก CPU เองซึ่งไอซีที่จะกล่าวถึงนี้ก็คือไอซีเบอร์ 8255 ของบริษัทอินเทลเป็นไอซีที่ทำหน้าที่เป็นอินพุตและเอาท์พุตพอร์ท นิยมใช้งานกันมากที่สุดทั้งยังมีราคาถูกและหาซื้อง่าย

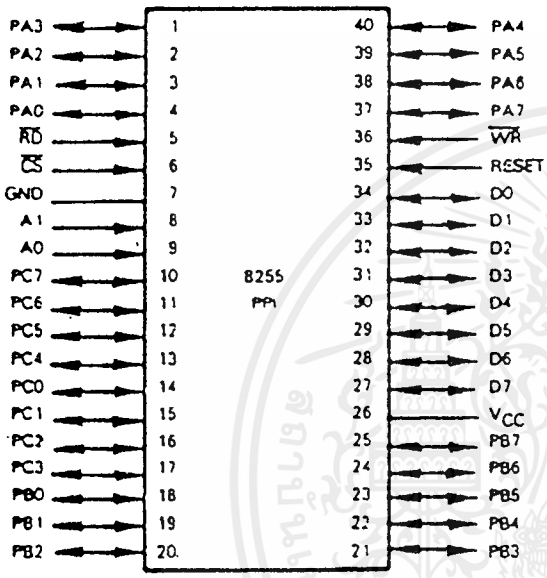
3.4.1 ลักษณะเบื้องต้น

8255 เป็นไอซีที่มี 40 ขา ได้รับการออกแบบมาให้มีสัญญาณเพื่อเชื่อมต่อกับ 8088 8255เป็นไอซีที่ต่อเป็นพอร์ทให้ไมโครโปรเซสเซอร์ได้ 3 พอร์ท โดยมีแผนผังและการจัดขาของ 8255A-5 ดังรูปที่



รูปที่ 3.10 แสดงแผนผังและการจัดขาของ 8255A-5

การเรียกพอร์ทของ 8255 จะเรียกพอร์ทต่างๆว่า พอร์ท A พอร์ท B และพอร์ท C โดยพอร์ท C แยกเป็น 2 ส่วนคือ พอร์ท C ล่างหรือตั้งแต่ PC_0 - PC_3 มีจำนวน 4 บิต และพอร์ท C บนหรือตั้งแต่ PC_4 - PC_7 ที่พิเศษคือ พอร์ททุกพอร์ทเป็นได้ทั้งพอร์ทอินพุทและพอร์ทเอาท์พุท นอกจากนี้ยังมีพอร์ทอีกพอร์ทหนึ่งซึ่งทำหน้าที่ควบคุมการทำงานของพอร์ท A , B , C โดยการรับคำสั่งมาจาก CPU พอร์ทนี้เราเรียกว่าพอร์ทควบคุม (Control Port) พอร์ทนี้จะใช้งานได้ก็ต่อเมื่อ CPU ต้องการกำหนดลักษณะการทำงานของพอร์ท A , B , C หรือต้องการเปลี่ยนแปลงหลังจากที่กำหนดไว้เดิม CPU จะส่งรหัสควบคุมมาทางดาต้าบัส (Data Bus) ให้แก่พอร์ทควบคุมนี้ รูปที่...แสดงรูปของไอซี 8255A



PIN NAME	DESCRIPTION	TYPE
D0 - D7	Bidirectional Data Bus	Bidirectional
PA0 - PA7	Eight I/O pins designated as Port A	Bidirectional
PB0 - PB7	Eight I/O pins designated as Port B	Bidirectional
PC0 - PC7	Eight I/O pins designated as Port C upper and Port C lower	Bidirectional
$\overline{RD}$	Read from device control	Input
$\overline{WR}$	Write to device control	Input
RESET	System reset	Input
$\overline{CS}$	Device select	Input
A0,A1	I/O port select	Input
V_{CC},GND	Power and Ground	

รูปที่ 3.11 ตำแหน่งขาต่างๆของ 8255

3.4.2 ขาสัญญาณต่างๆของ 8255

D_0 - D_7 เป็นขาที่ข้อมูลอินพุทเอาท์พุทจะต้องผ่านเข้าออกจากส่วนนี้ D_0 - D_7 จึงต่อเข้ากับระบบบัสของไมโครโปรเซสเซอร์ เพื่อให้ไมโครโปรเซสเซอร์สามารถอ่านหรือเขียนข้อมูลจากพอร์ทผ่านทางบัสนี้

CS (สัญญาณเลือกชิป) ขานี้เป็นขาอินพุทที่จะรับสัญญาณจากภายนอกเพื่อเลือกชิป 8255 โดยเมื่อขานี้เป็น 0 จะทำให้ 8255 ต่อเข้ากับระบบบัสของไมโครโปรเซสเซอร์ เพื่อให้ไมโครโปรเซสเซอร์เขียนหรืออ่านข้อมูลจากพอร์ทได้ แต่ถ้าเป็นลอจิก 1 มันจะปลดตัวเองออกจากระบบบัสของ CPU (โดยการทำให้เป็น HI-Z)

RD (สัญญาณการอ่าน) เป็นสัญญาณอินพุทที่ต้องส่งมาจากซีพียู เมื่อสัญญาณที่ขานี้เป็น 0 และสัญญาณ CS เป็น 0 ด้วย ไอซี 8255 จะทำตัวให้ซีพียูอ่านข้อมูลจากบัสในขณะที่เป็นพอร์ทอินพุท

WR เป็นสัญญาณการเขียน จะแอคทีฟเมื่อสัญญาณ WR และสัญญาณ CS เป็น 0 สัญญาณนี้จะมาจากซีพียูเมื่อต้องการเขียนข้อมูลลงบนพอร์ทที่กำหนด

A_0 - A_1 (สัญญาณแอดเดรส) ลอจิกของสัญญาณทั้งสองจะถอดรหัสออกเป็น 4 รหัส เพื่อกำหนดรีจิสเตอร์ภายในที่เชื่อมต่อกับพอร์ทอินพุทเอาท์พุทของ 8255

RESET (สัญญาณรีเซต) เป็นสัญญาณที่ส่งจากภายนอกเข้ามาทำการรีเซต 8255 เพื่อเคลียร์สถานะต่างๆของ 8255 เมื่อ 8255 ได้รับการรีเซต ก็จะกลับเข้าสู่โหมดอินพุทหรือทุกพอร์ทที่เป็นพอร์ทอินพุท

PA_0 - PA_7 เป็นสายสัญญาณที่เป็นพอร์ทของ 8255 ที่ชื่อพอร์ท A การเลือกพอร์ทจะเลือกโดยสัญญาณแอดเดรส A_0 - A_1

PB_0 - PB_7 เป็นสายสัญญาณที่เป็นพอร์ท B ของ 8255 ถูกเลือกโดยสัญญาณแอดเดรส A_0 - A_1

PC_0 - PC_7 เป็นสายสัญญาณที่เป็นพอร์ท C ของ 8255 การกำหนดพอร์ทนี้จะได้รับการกำหนดโดยสัญญาณแอดเดรส A_0 - A_1 พอร์ท C นี้แบ่งเป็น 2 กลุ่มคือ กลุ่ม PC_0 - PC_3 และกลุ่ม PC_4 - PC_7

3.4.3 รีจิสเตอร์ภายในของ 8255

เมื่อต่อ 8255 เข้ากับไมโครโปรเซสเซอร์แล้ว สิ่งที่ผู้ใช้จะต้องทำคือ การโปรแกรมให้ 8255 ทำงานตามที่ต้องการ จากการที่ 8255 มีพอร์ทที่ไมโครโปรเซสเซอร์มองเห็น 4 พอร์ท แต่ละพอร์ทจะเสมือนเป็นรีจิสเตอร์ที่สามารถเขียนและอ่านได้ รีจิสเตอร์แต่ละตัวนี้จึงถูกกำหนดด้วยแอดเดรสตามที่ตั้งไว้ สัญญาณของขาควบคุมที่ประกอบกันจะแสดงความหมายดังตารางที่

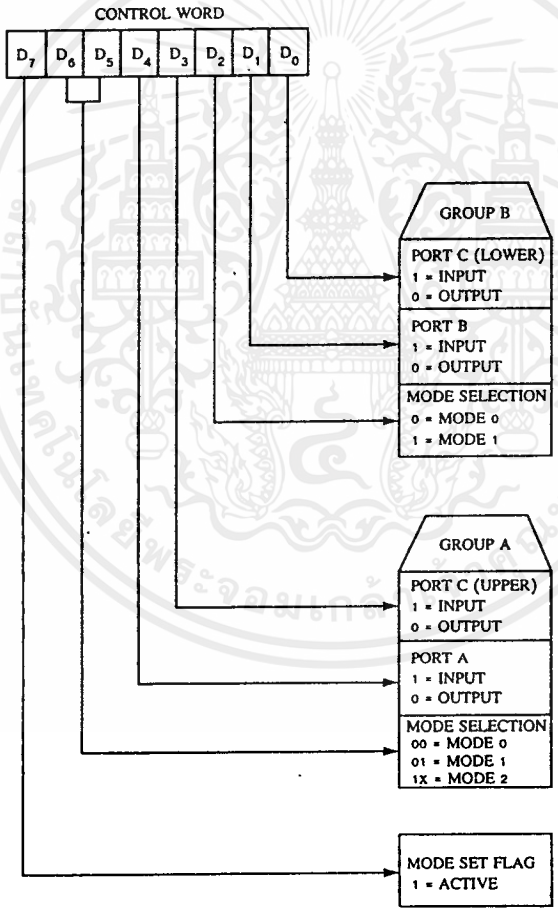
$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0	การทำงาน
0	1	0	0	0	ส่งข้อมูลไปที่พอร์ท A
0	0	1	0	0	รับข้อมูลจากพอร์ท A
0	1	0	0	1	ส่งข้อมูลไปที่พอร์ท B
0	0	1	0	1	รับข้อมูลจากพอร์ท B
0	1	0	1	0	ส่งข้อมูลไปที่พอร์ท C
0	0	1	1	0	รับข้อมูลจากพอร์ท C
0	1	0	1	1	ส่งรหัสควบคุมไปที่พอร์ทควบคุม
0	0	1	1	1	ไม่มีความหมาย
1	X	X	X	X	8255 ไม่ทำงานปลดสายอินพุทออกจากระบบ (Hi-Z)

ตารางที่ 3.1 ตารางแสดงผลของสัญญาณควบคุมต่างๆที่มีผลต่อ 8255

การใช้งาน 8255 จะต้องส่งรหัสควบคุม (control code) เข้าไปยังพอร์ทข้อมูลควบคุม เพื่อควบคุมการทำงานของ 8255 โดยใช้สัญญาณควบคุมพอร์ทหมายเลข 03H การควบคุมการทำงานของ 8255 มีหลายโหมดแต่ละโหมดจะแตกต่างกันออกไป การโปรแกรมให้ 8255 ทำงานจะทำได้ 3 โหมดคือ โหมด 0 โหมด 1 และโหมด 2 ซึ่งในการทำโครงงานครั้งนี้ได้ใช้ 8255 ทำงานในโหมด 0 ซึ่งจะได้อธิบายหลักการทำงานของ 8255 ในโหมด 0 ต่อไป

โหมด 0 หรืออินพุทเอาท์พุทแบบพื้นฐาน (Simple I/O Port)

การกำหนดโหมดการทำงาน จะต้องส่งข้อมูลคำสั่งเข้าไปโปรแกรมในพอร์ทควบคุมของ 8255 ซึ่งในที่นี้ใช้พอร์ทหมายเลข 03H แต่ละบิทของข้อมูลที่ส่งไปจะมีความหมายในตัวเอง ลักษณะความหมายของแต่ละบิทในรหัสควบคุมแสดงได้ดังรูปที่



รูปที่ 3.12 แสดงความหมายของแต่ละบิตในระหัสควบคุม

การโปรแกรม 8255 คือ การให้ค่ารหัสบิทต่างๆเข้าไปในรหัสควบคุมแล้วส่งไปยังรีจิสเตอร์ของพอร์ทควบคุม ความหมายของบิทต่างๆมีดังนี้

บิท D_7 เป็นบิทที่แสดงรหัสคำสั่งควบคุม ถ้าบิทนี้เป็น 1 หมายถึงรหัสควบคุมนี้จะมีผลต่อการเปลี่ยนแปลงการเซตโหมดต่างๆของ 8255

บิท D_6 และ D_5 เป็นการเลือกโหมดของพอร์ท A ซึ่งมี 3 โหมด 0 โหมด 1 และโหมด 2

บิท D_4 ถ้ามีค่าเป็น 0 หมายถึงการกำหนดพอร์ท A เป็นเอาท์พุท ถ้ามีค่าเป็น 1 จะหมายถึงการกำหนดให้พอร์ท A เป็นอินพุท

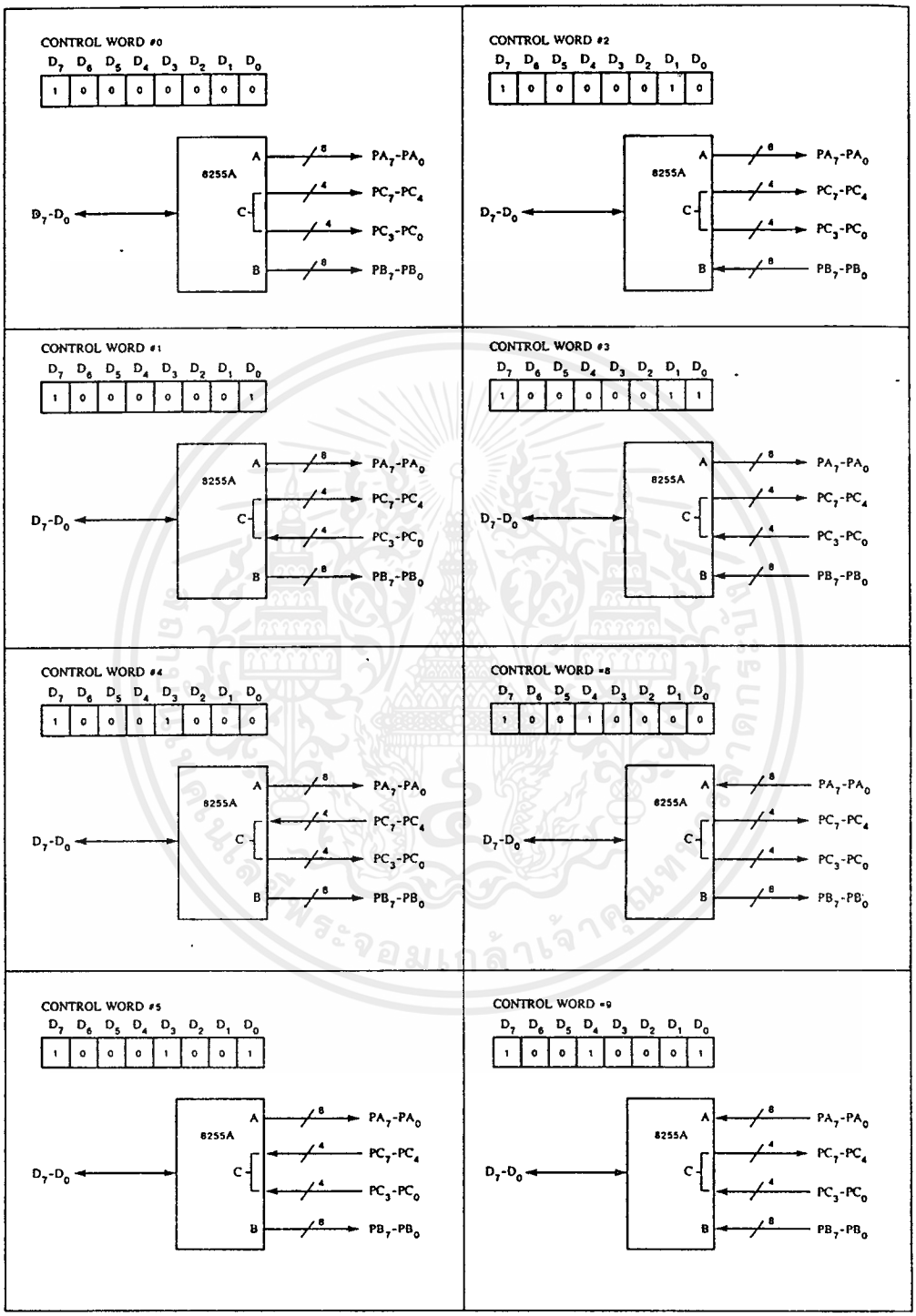
บิท D_3 เป็นบิทที่บอกถึงการเซ็ตของพอร์ท C บน ถ้าเป็น 0 จะทำให้พอร์ท C บนเป็นเอาท์พุท

บิท D_2 เป็นบิทที่บอกถึงการเซ็ตของพอร์ท B ถ้าเป็น 0 หมายถึง การเลือกพอร์ท B เป็นโหมด 0 และถ้าเป็น 1 หมายถึงการเลือกโหมด 1

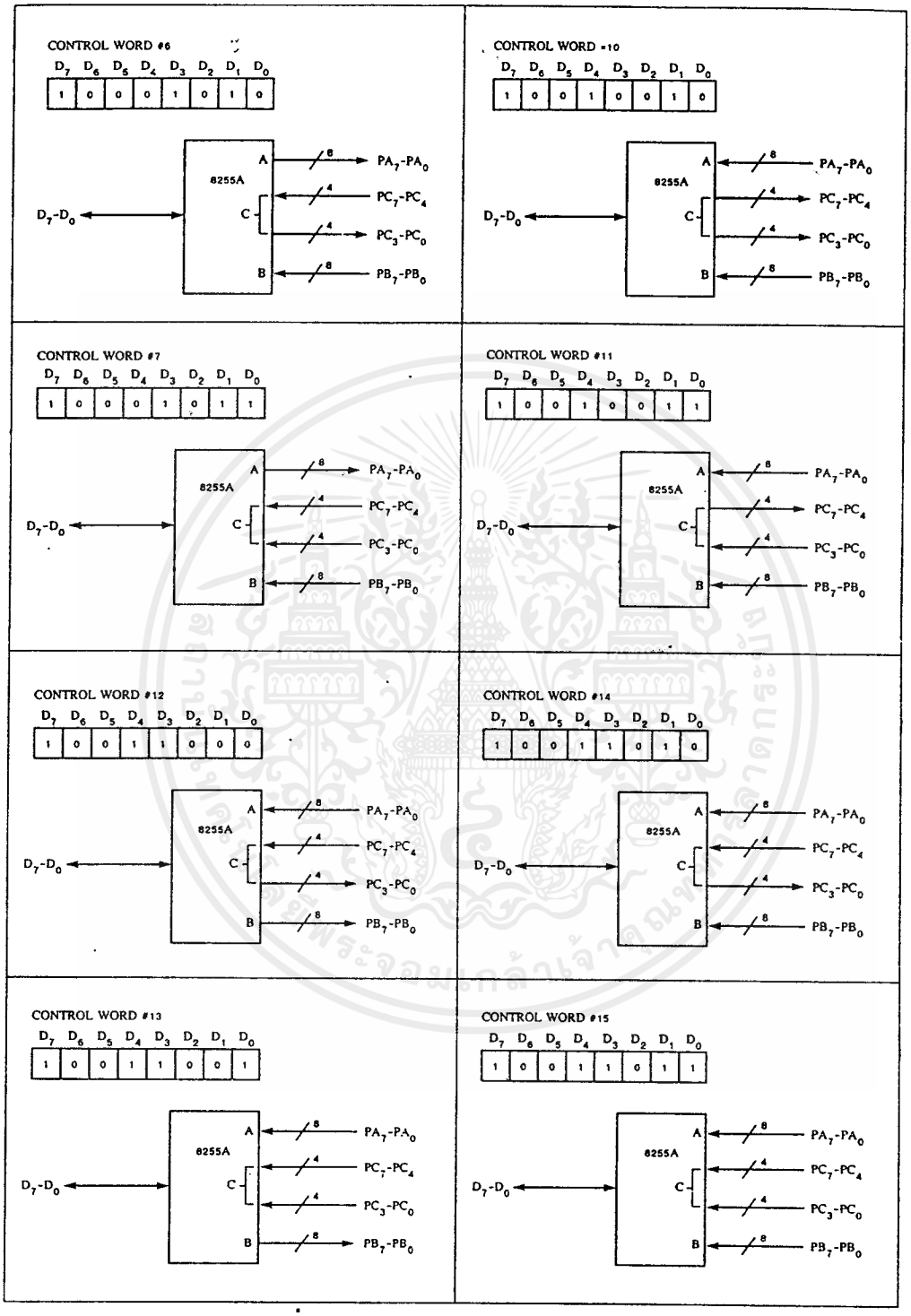
บิท D_1 เป็นการกำหนดอินพุทเอาท์พุทของพอร์ท B ถ้าเป็น 0 หมายถึงเอาท์พุท ถ้าเป็น 1 หมายถึงอินพุท

บิท D_0 เป็นการกำหนดอินพุทเอาท์พุทของพอร์ท C ล่าง ถ้าเป็น 0 หมายถึงเอาท์พุท ถ้าเป็น 1 หมายถึงอินพุท

การโปรแกรม 8255 จะเริ่มจากการเซ็ตค่าที่ต้องการแล้วเอาท์พุทไปยังพอร์ทควบคุม เช่น ถ้าต้องการโปรแกรมให้ทั้งพอร์ท A,B และ C เป็นพอร์ทเอาท์พุททั้งหมด เราจะเลือกให้ 8255 อยู่ในโหมด 0 โดยมีรหัสควบคุมเป็น 10000000 หรือ 80H เนื่องจากมีพอร์ทที่รับส่งข้อมูล 3 พอร์ทคือ พอร์ท A พอร์ท B และพอร์ท C ซึ่งพอร์ท C จะแยกออกเป็น 2 ส่วนคือ พอร์ท C ล่าง และพอร์ท C บน เราสามารถโปรแกรมให้ทั้ง 4 พอร์ทนี้เป็นอินพุทหรือเอาท์พุทก็ได้ เช่น ถ้าให้รหัสควบคุมเป็น 82H จะทำให้พอร์ท B เป็นอินพุท พอร์ท A และพอร์ท C เป็นเอาท์พุท โดยมีรูปแบบความเป็นไปได้ในการโปรแกรมให้พอร์ท A, B, C เป็นอินพุตหรือเอาท์พุตพอร์ทได้ทั้งหมด 16 รูปแบบดังรูปที่



รูปที่ 3.13 แสดงรหัสควบคุมในโหมด 0



รูปที่ 3.13 (ต่อ) แสดงรหัสควบคุมในโหมด

3.5 การอินเทอร์เฟส IBM/PC โดยผ่านทางสล็อตเสริม

ข้อดีประการหนึ่งของพีซีก็คือภายในเครื่องคอมพิวเตอร์จะมีสล็อตเสริม(expansion slots) ซึ่งเป็นช่องสำหรับเสียบแผ่นวงจรอิเล็กทรอนิกส์ที่ต้องการเพิ่มเติมเข้าไป ทำให้แผ่นวงจรเหล่านั้นสามารถติดต่อกับเครื่องคอมพิวเตอร์ได้โดยผ่านทางชุดของสายนำสัญญาณที่เราเรียกว่าบัส(bus)

สล็อตเสริมนี้จะช่วยให้เราสามารถต่อเติมเสริมแต่งความสามารถพิเศษหลายๆอย่างให้กับเครื่องคอมพิวเตอร์ของเราได้ เช่น การเสียบแผ่นวงจรหรือที่เราเรียกว่าอะแดปเตอร์การ์ด(adapter card) ก็จะทำให้สามารถเพิ่มความละเอียดและสีสันให้กับจอภาพได้มากยิ่งขึ้นหรืออาจใช้ต่อเติมเพื่อทำให้คอมพิวเตอร์สามารถบันทึกเสียงและเล่นดนตรีได้ หรืออาจทำให้เขียนหรืออ่านข้อมูลจากเทปได้ ซึ่งในการทำปริญญานิพนธ์เรื่อง System Identification นี้เราได้ใช้เครื่องคอมพิวเตอร์เพื่อทำการควบคุมการทำงานของวงจรรับสัญญาณจากอินพุตและเอาท์พุตของระบบจริงโดยผ่านทางสล็อตเสริม ซึ่งอุปกรณ์เหล่านี้ไม่ว่าจะเป็นอุปกรณ์ที่ต่อผ่านสล็อตเสริมหรืออุปกรณ์ที่ต่อเข้ากับเครื่องคอมพิวเตอร์โดยตรง เช่น คีย์บอร์ดล้วนต้องอาศัยการรับและส่งข้อมูลผ่านระบบบัสเสมอ

3.5.1 บัสแบบต่างๆ

ในพีซีโดยทั่วไประบบบัสจะแบ่งออกได้ 3 ชนิดคือ บัสแอดเดรส(address bus) , บัสข้อมูล(data bus) และบัสควบคุม(control bus) ซึ่งแบ่งออกเป็นระบบบัสแบบต่างๆได้ดังนี้

บัส 8 บิต ข้อมูลถูกส่งไปยังอะแดปเตอร์การ์ดและอุปกรณ์ต่างๆทางบัสผ่านชุดของสายนำสัญญาณชนิด 8 เส้น

บัส 16 บิตหรือบัส ISA ข้อมูลถูกส่งไปบนชุดของสายนำสัญญาณ 8 เส้นหรือ 16 เส้นขึ้นอยู่กับชนิดของอะแดปเตอร์การ์ดที่เสียบอยู่บนสล็อตเสริม

บัส EISA หรือบัส MCA ข้อมูลถูกส่งไปบนชุดของสายนำสัญญาณ 32 เส้นเพื่อติดต่อกับอะแดปเตอร์การ์ดที่ออกแบบเฉพาะสำหรับบัสชนิด 32 บิต ข้อแตกต่างระหว่างบัสชนิดทั้งสองก็คือสล็อตเสริมแบบ MCA ไม่สามารถรองรับอะแดปเตอร์การ์ดแบบ 8 หรือ 16 บิตได้ขณะที่สล็อตเสริมแบบ EISA รองรับได้ สล็อตบัสชนิด EISA ได้ถูกออกแบบมาอย่างชาญฉลาดโดยอนุญาตให้การ์ดรุ่นเดิมที่เป็นชนิด 8 และ 16 บิตสามารถเสียบลงไปได้ด้วยและการ์ดที่เป็นชนิด EISA ก็สามารถเสียบลงไปได้ด้วย โดยการ์ด EISA สามารถเสียบลงไปได้ลึกกว่า ทั้งนี้เพื่อสัมผัสกับลายวงจรไฟฟ้าชนิด 32 บิตแบบ EISA ซึ่งมีลักษณะช่องห่างระหว่างลายวงจรที่ซอยถี่มากกว่าปกติ

ในการทำปริญญานิพนธ์เรื่อง System Identification ครั้งนี้ได้ใช้การ์ดแบบ ISA (ISA card) แต่การติดต่อกับเครื่องคอมพิวเตอร์จะใช้จำนวนบิตข้อมูลเพียงแค่ 8 บิตเท่านั้น (ISA บัสสามารถติดต่อส่งผ่านข้อมูลได้จำนวน 16 บิต) คือใช้งานแค่ส่วนบนของ ISA card ซึ่งมีจำนวน 62 pin เท่านั้น โดยแต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขา แบ่งออกเป็น 2 ข้างๆละ 31 ขา ส่วนการเรียกตำแหน่งของขาของสล็อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้ายหรือขวา) ของสล็อต โดยขาที่อยู่ข้างซ้ายของสล็อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งของขา เช่น ขา B16 คือขาทางด้านซ้ายของสล็อตขาที่ 16

(นับจากทางด้านท้ายของเครื่อง) ส่วนขาที่อยู่ทางด้านขวาของสล็อตจะเรียกโดยใช้ตัวอักษร "A" นำหน้าเลขตำแหน่งของขา เช่น ขา A24 ก็คือขาทางด้านขวาของขาสล็อตที่ 24 (นับจากทางด้านท้ายของเครื่อง) ส่วนอีก 36 pin ซึ่งเป็นส่วนล่างของ ISA card ยังไม่ได้ใช้งานซึ่งอาจจะได้ใช้งานถ้ามีความจำเป็นที่จะต้องใช้เพื่อเพิ่มประสิทธิภาพการทำงานของอุปกรณ์

ISA(Industry Standard Architecture) บัส คือ IBM PC ATบัส บางทีอาจเรียกว่า ATบัส ในบริษัทผู้ผลิตคอมพิวเตอร์ทั่วๆไปจะเรียกระบบบัสนี้ว่า ISAบัสเนื่องจาก AT เป็นเครื่องหมายการค้าของบริษัท IBM จุดเริ่มต้นของ ISAบัสมาจาก 8088-base IBM PC ซึ่งเป็นที่รู้จักกันในปีค.ศ. 1981 PC/XTบัสมี 62 pin แล้วในปีค.ศ. 1984 ก็มี IBM PC AT เกิดขึ้นมาพร้อมกับเพิ่มเติมอีก 36 pin จาก PC/XT เพื่อความสะดวกในการใช้งานกับไมโครโปรเซสเซอร์ 80286 ซึ่งมีสัญญาณข้อมูล 16 บิตและสัญญาณแอดเดรส 24 บิต

3.5.2 รายละเอียดเกี่ยวกับสัญญาณต่างๆ

OSC (Oscillator; ขา B30)

ขานี้เป็นเอาท์พุทที่เชื่อมต่อกับสัญญาณคล็อกที่มีค่าความถี่สูงสุดบนเมนบอร์ด คือ 14.31818 MHz ซึ่งมีคาบเวลาประมาณ 70 ns และมี Duty Cycle ประมาณ50%

CLK (Clock; ขา B20)

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งต่อกับสัญญาณคล็อกที่ถูกสร้างขึ้นโดยการหารสัญญาณ OSC ด้วย 3 ทำให้ได้ความถี่ประมาณ 4.77 MHz หรือมีช่วงเวลาใน 1 คาบเท่ากับ 210 ns สัญญาณนี้ถูกใช้เป็นคล็อกของระบบ

RESET DRV (ขา B2)

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งจะแอคทีฟ (ลอจิก "1") ในช่วงที่เราเริ่มจ่ายไฟให้กับระบบ และจะยังคงแอคทีฟไปจนกว่าระบบต่างๆภายใน IBM/PC จะพร้อมที่จะทำงานได้ จากนั้นสัญญาณนี้ก็จะเปลี่ยนกลับเป็นลอจิก "0" นอกจากนี้ในระหว่างการทำงานของ IBM/PC ถ้าระดับของแรงดันแหล่งจ่ายไฟตกลง สัญญาณนี้ก็จะถูกทำให้แอคทีฟเช่นกัน

A0-A19 (Address Bus: ขา A31-A12):

ขาสัญญาณทั้ง 20 ขานี้เป็นเอาท์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำหรืออุปกรณ์ I/O ที่ 8088 ต้องการติดต่อด้วย โดยที่สัญญาณ A0 จะมีนัยสำคัญต่ำสุด (Least Significant Bit) และ A19 จะมีนัยสำคัญสูงสุด(Most Significant Bit) สำหรับค่าแอดเดรสบนบัสแอดเดรส A0-A19 นี้จะถูกกำหนดโดย 8088 ในระหว่างขบวนการอ่าน/เขียนข้อมูลลงในหน่วยความจำหรืออุปกรณ์ I/O

จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้จะมีอยู่ 20 เส้น ซึ่งสามารถที่จะอ้างแอดเดรสของหน่วยความจำได้ถึง 1 Mbyte แต่อย่างไรก็ตามจะมีแอดเดรสบางแอดเดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้วคือแอดเดรสของหน่วยความจำ RAM บนเมนบอร์ดที่ถูกใช้โดยระบบจำนวน 64 Kbyte (สำหรับ

IBM PC/XT จะเป็นจำนวน 256 Kbyte) และแอดเดรสสำหรับหน่วยความจำ ROM อีก 48 Kbytr ซึ่งจะถูกจัดอยู่ในช่วงของแอดเดรสบนสุดใน 1 Mbyte คือ 0FC00H จนถึง 0FFFFFH (สำหรับ IBM PC/XT จะเป็น 64 Kbyte)

สำหรับการอ้างแอดเดรสของพอร์ท I/O นั้นจะใช้แอดเดรสเพียง 16 เส้นคือ A0-A15 ซึ่งจะทำให้อ้างแอดเดรสของพอร์ทได้ 64K พอร์ทโดยผ่านทางชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสที่เหลือคือ A16-A19 นั้นจะไม่ถูกใช้งาน อย่างไรก็ตามภายใน IBM/PC จะใช้เส้นแอดเดรสในการอ้างแอดเดรสของพอร์ทเพียง 10 เส้น คือจาก A0-A9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ในช่วง 0200H จนถึง 03FFH เท่านั้น

D0-D7 (Data Bus : ขา A9-A2) :

ขาสัญญาณนี้จะเป็นแบบ Bi-Directional ซึ่งจะต่อกับบัสข้อมูลของระบบ เพื่อทำหน้าที่ในการส่งผ่านข้อมูลระหว่างพอร์ท I/O กับ IBM/PC โดยบิท D0 จะมีนัยสำคัญต่ำสุดและบิท D7 จะมีนัยสำคัญสูงสุด ซึ่งโดยทั่วไปขอบขาขึ้นของสัญญาณ $\overline{IOW}$ หรือ $\overline{MEMW}$ นี้จะถูกใช้เพื่อสั่งให้พอร์ท I/O หรือหน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลไปเก็บไว้

ALE (Address Latch Enable; ขา B28)

ขาสัญญาณนี้เป็นสัญญาณเอาท์พุทที่ 8288 Bus Controller สร้างขึ้นเพื่อใช้สำหรับแสดงการเริ่มต้นของบัสไซเคิล และแสดงให้อุปกรณ์ภายนอกทราบว่าแอดเดรสที่ 8088 ต้องการจะติดต่อด้วยนั้นถูกส่งออกมาบนบัสแอดเดรสแล้ว โดยที่สัญญาณ ALE นี้จะเปลี่ยนจากลอจิก "1" เป็น "0" เมื่อค่าแอดเดรสที่ถูกต้องถูกส่งออกมาบนบัสข้อมูลเรียบร้อยแล้ว

I/O CHCK (I/O Channel Check; ขา A1)

ขาสัญญาณนี้เป็นอินพุทที่ใช้ในการแสดงความผิดพลาดเกี่ยวกับพาริตี้ ที่เกิดขึ้นในการทำงานของวงจรอินเทอร์เฟสหรืออุปกรณ์ I/O เมื่อขาสัญญาณนี้ได้รับลอจิก "0" จะทำให้ 8088 ถูกอินเทอร์รัพท์แบบ Non-Maskable (NMI) อย่างไรก็ตามเราสามารถที่จะกำหนดให้วงจรภายในของ IBM/PC ทำการขออินเทอร์รัพท์ (เมื่อได้รับสัญญาณ I/O CHCK) หรือไม่ก็ได้

I/O CHRDY (I/O Channel Ready; ขา A10)

ขาสัญญาณนี้เป็นอินพุทที่ใช้เพิ่มช่วงเวลาในบัสไซเคิลในกรณีที่อุปกรณ์ I/O หรือหน่วยความจำที่เกี่ยวข้องกับขบวนการในบัสไซเคิลที่เกิดขึ้นนั้นไม่สามารถทำงานทันตามช่วงเวลาปกติของบัสไซเคิลนั้นๆได้

IRQ2 – IRQ7 (Interrupt Request 2 – 7; ขา B4 และ B25 – B21)

ขาสัญญาณทั้ง 6 นี้เป็นขาอินพุทที่ใช้สำหรับการขออินเทอร์รัพท์จาก 8088 โดยสัญญาณเหล่านี้จะต่อเข้ากับ 8259A บนเมนบอร์ดโดยตรง โปรแกรมในส่วน BIOS ของ IBM/PC จะทำการโปรแกรม 8259A ให้ IRQ2 มีลำดับความสำคัญสูงสุด (Highest Priority) และ IRQ7 มีลำดับความสำคัญที่ต่ำสุด ในกรณีที่มีการขออินเทอร์รัพท์เกิดขึ้น คือระดับลอจิกที่ขา IRQ ขาใดขาหนึ่งถูกเปลี่ยน

จากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น) 8259A ก็จะทำการส่งสัญญาณ INT ให้กับ 8088 เพื่อทำการขออินเทอร์รัพท์

IOR (I/O Read : ขา B14) :

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ที่สร้างขึ้นโดย 8288 Bus Controller เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O เพื่อใช้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสข้อมูล

IOW (I/O Write : ขา B13) :

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ที่สร้างขึ้นโดย 8288 Bus Controller เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลจากพอร์ท I/O เพื่อใช้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นรับข้อมูลไปเก็บไว้

บัสของแหล่งจ่ายไฟของระบบ

+5 Vdc (ขา B3 และ B29):

ขาทั้งสองนี้ต่อกับแหล่งจ่ายไฟ DC + 5V ของระบบโดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$ 5% คืออยู่ในช่วง +4.75 ถึง +5.25 Vdc

+12 Vdc (ขา B9):

ขานี้ต่อกับแหล่งจ่ายไฟ DC + 12V ของระบบโดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$ 5% คืออยู่ในช่วง +11.4 ถึง +12.6 Vdc

-5 Vdc (ขา B5):

ขานี้ต่อกับแหล่งจ่ายไฟ DC - 5V ของระบบโดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$ 10% คืออยู่ในช่วง –5.5 ถึง –4.5 Vdc

-12 Vdc (ขา B7):

ขานี้ต่อกับแหล่งจ่ายไฟ DC -12V ของระบบโดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$ 10% คืออยู่ในช่วง –13.2 ถึง –10.8 Vdc

GND (ขา B1,B10 และ B31) :

ขาทั้งสามนี้จะต่อเข้ากับกราวด์ของระบบ

3.5.3 การจัดแอดเดรสสำหรับพอร์ท I/O ใน IBM/PC

ในหัวข้อนี้จะกล่าวถึงวิธีการอ้างและใช้งานแอดเดรสต่างๆของพอร์ท I/O ที่ใช้งานอยู่ใน IBM/PC ในการควบคุมและตรวจสอบสภาวะการทำงานรวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นชิพซัพพอร์ทหรือการ์ดต่างๆที่ใช้ในระบบของ IBM/PC นั้นจะกระทำผ่านทางพอร์ท I/O ของระบบดังนั้นในการที่จะควบคุมการทำงานของอุปกรณ์เหล่านี้ จึงจำเป็นต้องศึกษาวิธีการควบคุมพอร์ท I/O ต่างๆของระบบด้วยและเนื่องจากการควบคุมและติดต่อกับพอร์ทเหล่านี้ต้องกระทำโดยการอ้างถึงแอดเดรสของพอร์ท I/O เหล่านั้นโดยตรง จึงจำเป็นต้องศึกษาถึงการอ้างแอดเดรสของ 8088 ใน IBM/PC ด้วย

สำหรับแอดเดรสของพอร์ท I/O ต่างๆนั้นจะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088 ซึ่งแอดเดรสเหล่านี้เป็นแอดเดรสที่จัดไว้สำหรับพอร์ท I/O โดยเฉพาะคือแยกจากแอดเดรสของหน่วยความจำโดยเด็ดขาด ภายในไมโครโปรเซสเซอร์เบอร์ 8088 นี้จะมีแอดเดรสสำหรับใช้กับพอร์ทI/O อยู่ทั้งสิ้น 65,536 หรือ 64K แอดเดรส (ในขณะที่มีแอดเดรสสำหรับหน่วยความจำอยู่ 1 Mbyte) ซึ่งทำให้การอ้างแอดเดรสของพอร์ท I/O ที่ทำงานร่าวกับ 8088 นั้นต้องใช้จำนวนเส้นแอดเดรสในบัสแอดเดรสทั้งสิ้น 16 เส้น คือ A0-A15 แต่สำหรับใน IBM/PC นี้ถูกออกแบบมาให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นล่าง คือ A0-A9 เท่านั้นดังนั้นในการอ้างถึงแอดเดรสของพอร์ทของอุปกรณ์หรือชิพซัพพอร์ทใดๆที่ใช้ร่วมกับ IBM/PC จึงใช้จำนวนเส้นแอดเดรสเพียง 10 เส้นด้วย โดยเส้นแอดเดรสที่เหลือคือ A10-A15 นั้นจะไม่ถูกนำมาใช้งาน อย่างไรก็ตามถึงแม้ว่าเส้นแอดเดรส A10-A15 นี้จะไม่ถูกนำมาใช้งานแต่ค่าแอดเดรสบนเส้นแอดเดรสเหล่านี้ยังคงเปลี่ยนแปลงตามค่าแอดเดรสของพอร์ทที่กำหนดไว้ในคำสั่ง OUT หรือ IN อยู่ด้วยเพียงแต่ไม่ถูกนำมาดีโค้ดร่วมกับแอดเดรส A0-A9 เท่านั้น

เนื่องจากใน IBM/PC ได้ใช้งานเส้นแอดเดรสเพียง 10 เส้นคือ A0-A9 ดังนั้นจึงสามารถที่จะอ้างแอดเดรสของพอร์ทได้สูงสุดเพียง 1024 พอร์ทเท่านั้น นอกจากนี้ในกรณีที่เป็นการอ่านข้อมูลจากพอร์ทของ IBM/PC ข้อมูลในบิท A9 จะถูกจัดให้มีหน้าที่ในการแบ่งพอร์ททั้ง 1024 พอร์ทออกเป็น 2 ส่วนอีกด้วย กล่าวคือถ้าข้อมูลในบิท A9 เป็น "0" แล้ว เราจะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทของอุปกรณ์ที่อยู่บนเมนบอร์ดของ IBM/PC เช่น 8253-5, 8237-5 หรือ 8259A เท่านั้น แต่ถ้าข้อมูลในบิท A9 เป็น "1" ก็จะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทที่อยู่บนการ์ดต่างๆเท่านั้น ดังที่จะแสดงพอร์ททั้ง 1024 ได้ดังตารางที่3.2

Hex range	Usage	
000-00F	DMA chip 8237A-5	Assigned to system board Components
020-021	Interrupt 8259A	
040-043	Timer 8253-5	
060-063	PPI 8255A-5	
080-083	DMA page registers	
0Ax	NMI mark register	
0Cx	Reserved	
0Ex	Reserved	
100-1FF	Not usable	
200-20F	Game control	Assigned to feature card Ports
210-217	Expansion unit	
220-24F	Reserved	
278-27F	Reserved	
2F0-2F7	Reserved	
2F8-2FF	Asynchronous communications (2)	
300-31F	Prototype card *	
320-32F	Fixed disk	
378-37F	Printer	
380-38C	SDLC communications	
380-389	Binary synchronous communications (2)	
3A0-3A9	Binary synchronous communications (1)	
3B0-3BF	IBM monochrome display/printer	
3C0-3CF	Reserved	
3D0-3DF	Color/graphics	
3E0-3EF	Reserved	
3F0-3F7	Diskette	
3F8-3FF	Asynchronous communications (1)	

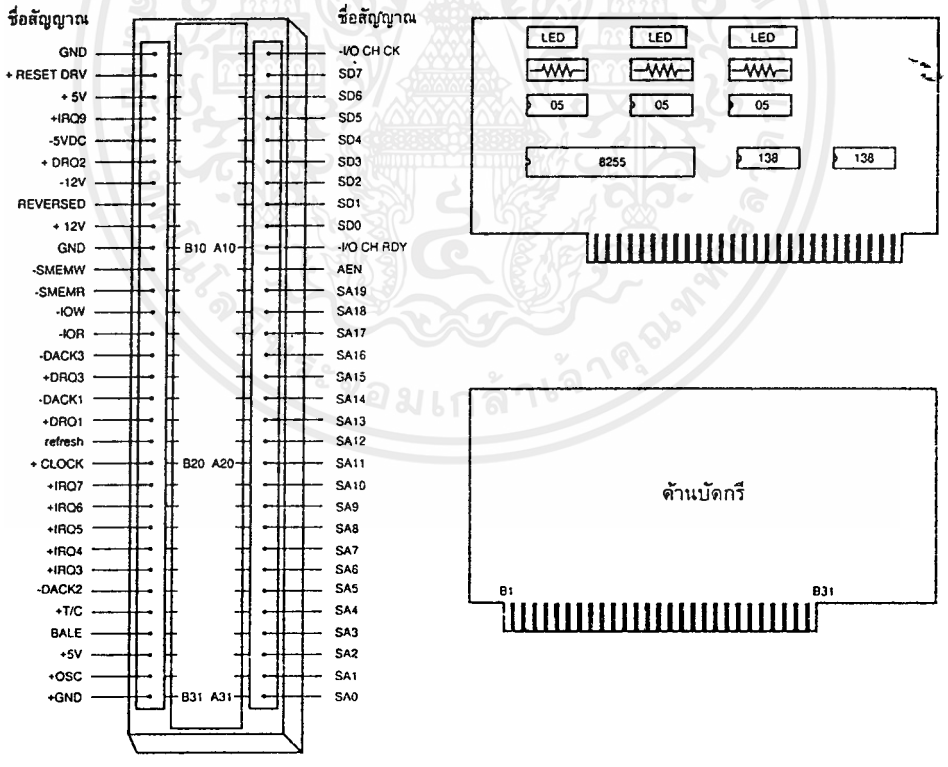
ตารางที่3.2 ตารางแสดงแอดเดรสพอร์ท I/O ของ IBM/PC

จากที่กล่าวมานั้นจะสรุปได้ว่าพอร์ทบน IBM/PC ทั้ง 1024 พอร์ทถูกแบ่งออกเป็น 2 กลุ่ม โดยกลุ่มแรกเป็นกลุ่มของพอร์ทที่อยู่บนเมนบอร์ดและกลุ่มที่สองเป็นกลุ่มที่จัดเตรียมไว้สำหรับพอร์ทที่อยู่บนการ์ดต่างๆ อย่างไรก็ตามสิ่งหนึ่งที่ต้องคำนึงถึงก็คือถ้าแอดเดรสที่เราเลือกให้กับพอร์ทนี้ตรงกับค่าแอดเดรสเดิมที่มีอยู่บนเมนบอร์ดแล้ว เมื่อเราทำการส่งข้อมูลให้กับพอร์ทที่อยู่ในตำแหน่งของแอดเดรสนี้ก็เท่ากับเป็นการส่งข้อมูลให้กับทั้งพอร์ทที่อยู่บนเมนบอร์ดและพอร์ทที่อยู่บนการ์ดด้วย ซึ่งในกรณีนี้อาจก่อให้เกิดความผิดพลาดได้เช่นกัน ดังนั้นในการกำหนดค่าแอดเดรสให้กับพอร์ทที่ถูกสร้างขึ้นบนการ์ดต่างๆจึงควรจะใช้ค่าแอดเดรสที่แอดเดรสบิท A9 มีค่าเป็น “1” คือแอดเดรส 0FE00H จนถึง 0FFFFH เท่านั้น (แอดเดรสบิท A10-A15 ไม่ถูกใช้ในการดีโค้ด แต่เพื่อความสะดวกจึงกำหนดให้มีค่าเป็น “1” ในฐานสองทั้งหมด แต่ในการใช้งานจริงอาจเปลี่ยนให้แอดเดรส A10-A15 แต่ละบิทมีค่าเป็น “0” หรือ “1” ก็ได้)

3.5.4 การใช้งานแอดเดรสสำหรับพอร์ท I/O ใน IBM/PC

1.ในกลุ่มแรกนี้เป็นกลุ่มของพอร์ท I/O ที่อยู่บนเมนบอร์ดของ IBM/PC ซึ่งจะมีแอดเดรสอยู่ในตำแหน่ง 0000H จนถึง 01FFH สำหรับแอดเดรสของพอร์ท I/O ในกลุ่มนี้จะถูกใช้ในการอ้างแอดเดรสของชิพซัพพอร์ทและอุปกรณ์ที่เป็น I/O ต่างๆบนเมนบอร์ดของ IBM/PC

2.ในกลุ่มที่สองนี้จะเป็นกลุ่มของพอร์ท I/O ที่ถูกใช้งานอยู่บนการ์ดที่ใช่เสียบบนสล็อตเสริมของ IBM/PC สำหรับแอดเดรสของพอร์ทเหล่านี้จะเริ่มต้นจากแอดเดรส 0200H จนถึง 03FFH ซึ่งก็คือแอดเดรสที่มีบิท A9 เป็น "1" นั่นเอง ดังนั้นก่อนที่จะทำการออกแบบวงจรอินเทอร์เฟสที่จำเป็นต้องใช้ค่าแอดเดรสสำหรับพอร์ท I/O จึงควรจะตรวจสอบดูก่อนว่าการ์ดต่างๆที่ใช้อยู่ในระบบมีการ์ดใดบ้างและการ์ดเหล่านั้นใช้งานแอดเดรสใดบ้างจากนั้นจึงทำการออกแบบวงจรอินเทอร์เฟสโดยเลือกใช้เฉพาะแอดเดรสที่ยังไม่ถูกใช้งานซึ่งในโครงงานนี้ได้ใช้แอดเดรสช่วง 300H – 31FH ซึ่งเมื่อดูจากตารางที่ 3.2 จะเห็นได้ว่าเป็นช่วงของ Prototype card มาใช้เป็นแอดเดรสสำหรับการอ้างถึง Interface Card และส่วนของสล็อตที่ใช้สำหรับต่อกับ Card ต่างๆนั้นสามารถแสดงดังรูปที่3.14



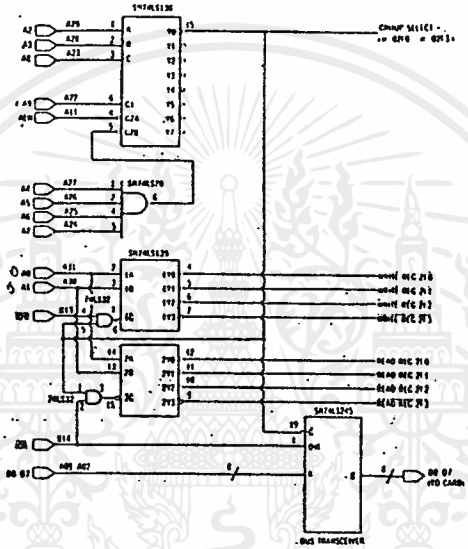
รูปที่ 3.14 แสดงขาสัญญาณบนสล็อต และการ์ดอินเทอร์เฟส

3.5.5 เทคนิคในการดีโค้ดแอดเดรสสำหรับพอร์ท I/O

ในหัวข้อต่างๆที่ผ่านมาข้างต้นได้กล่าวถึงการอ้างแอดเดรสและการใช้งานแอดเดรสต่างๆของพอร์ท I/O ใน IBM/PC สำหรับในหัวข้อนี้จะกล่าวถึงวิธีการต่างๆที่ใช้ในการดีโค้ดแอดเดรสต่างๆให้เป็นไปตามที่เราต้องการ

3.5.5.1 การดีโค้ดแบบ Fixed

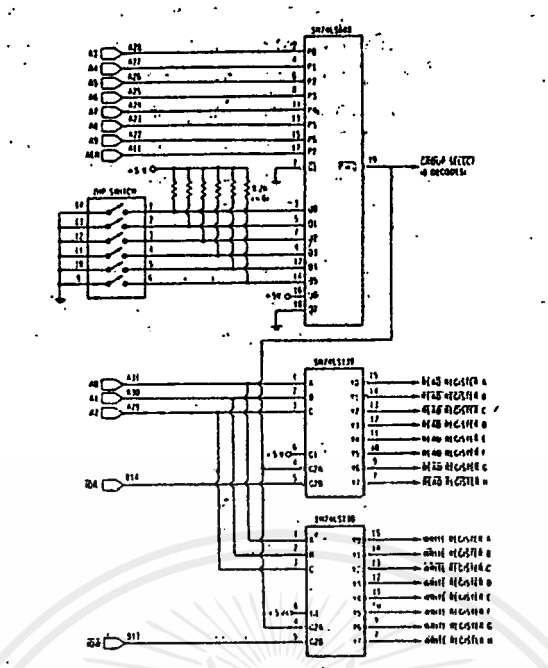
การดีโค้ดแบบนี้เป็นวิธีที่ง่ายและสะดวกในการดีโค้ดแอดเดรสหรือกลุ่มของแอดเดรส ซึ่งจะเป็นการกำหนดจำนวนของแอดเดรสที่ต้องการใช้ จากนั้นจึงทำการเลือกบล็อกของแอดเดรสที่ยังไม่ถูกใช้งานโดยการ์ดหรือวงจรอินเทอร์เฟสอื่นๆแล้วจึงออกแบบวงจรที่ทำการดีโค้ดแอดเดรสที่เราต้องการ



รูปที่ 3.15 ตัวอย่างวงจรดีโค้ดแอดเดรสแบบ Fixed

3.5.5.2 การดีโค้ดโดยใช้สวิทช์เลือก (DIP SWitch)

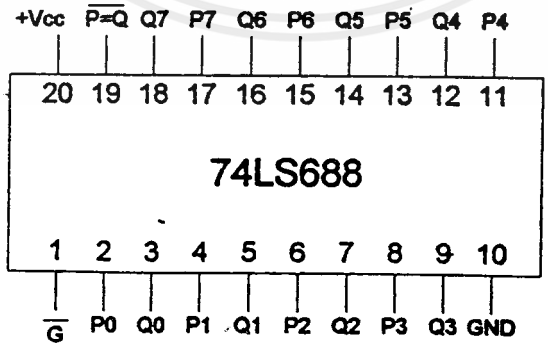
การดีโค้ดแบบ Fixed ที่ได้กล่าวมาแล้วมีข้อเสียอยู่บางประการคือ แอดเดรสที่เราเลือกใช้งานไว้นั้นอาจซ้ำกันกับแอดเดรสของการ์ดอื่นที่เรานำมาเพิ่มเข้าไปในระบบภายหลังได้ ซึ่งในกรณีเช่นนี้เราต้องแก้ไขวงจรเพื่อหลีกเลี่ยงไปใช้แอดเดรสอื่นที่ยังว่างอยู่และไม่ถูกใช้งานโดยการ์ดที่จะเพิ่มเข้าไปใหม่ซึ่งยุ่งยากและเสียเวลามากขึ้น ปัญหาเช่นนี้เราสามารถแก้ไขได้โดยใช้วงจรดีโค้ดที่สามารถเปลี่ยนแปลงค่าแอดเดรสได้ โดยเพียงแต่เปลี่ยนตำแหน่งของสวิทช์เลือก ที่เซทไว้ในวงจรเท่านั้นดังรูปที่ 3.16



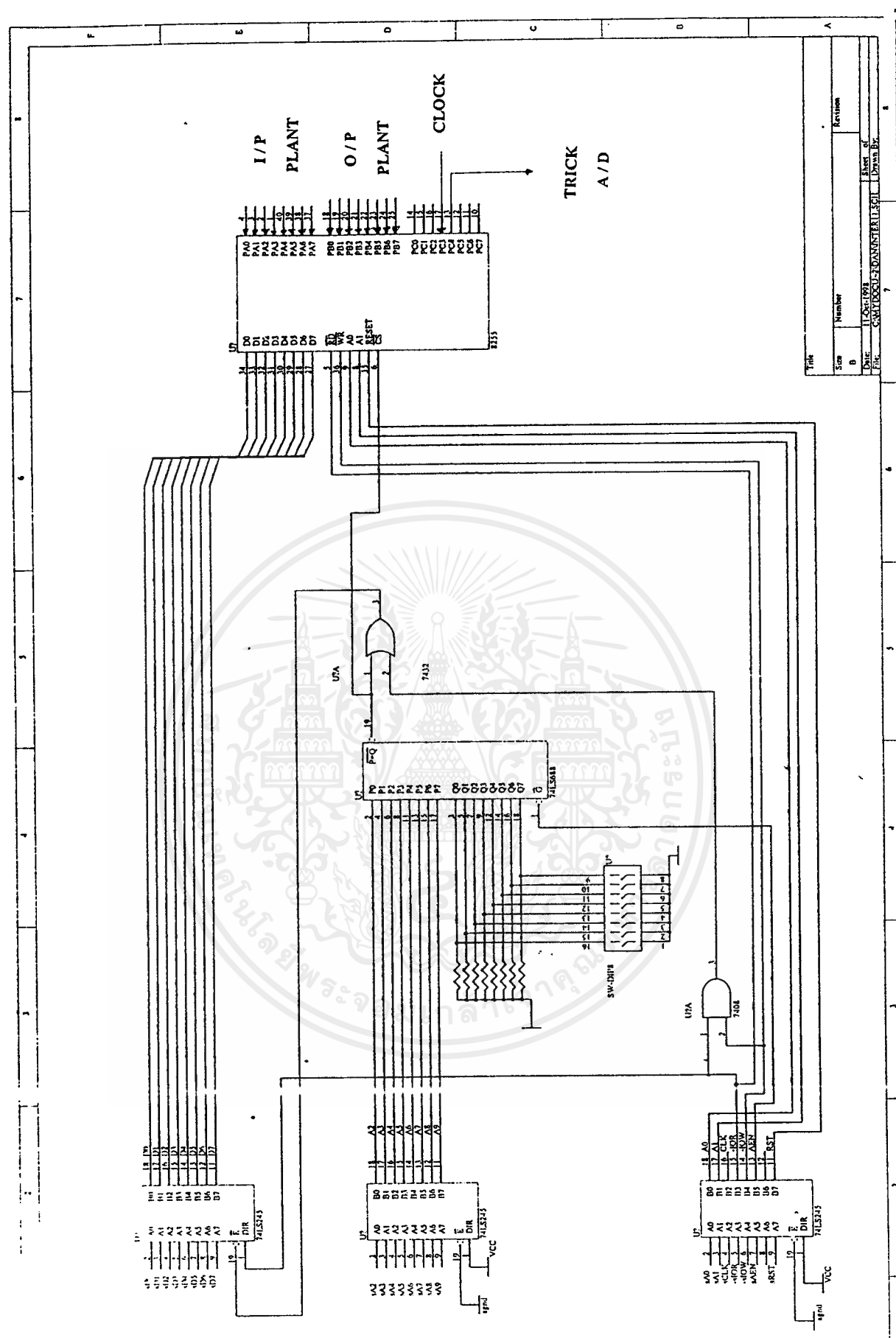
รูปที่ 3.16 ตัวอย่างวงจรดีโค้ดโดยใช้สวิทช์เลือก

3.5.6วงจรการทำงานของ Interface Card

เราใช้ไอซีเบอร์74LS688(ดังรูปที่3.17)จะทำการดีโค้ดสัญญาณแอดเดรสจากสล็อตของคอมพิวเตอร์มาเพื่อใช้แอดเดรสให้กับ 8255 สัญญาณที่จะนำมาใช้ในการดีโค้ดมี A0-A9 แต่8255 ต้องการสัญญาณจาก A0 และ A1 ไปควบคุมรีจีสเตอร์ภายในตัวมันเพื่อทำการเลือกพอร์ททั้ง 4 พอร์ท ดังนั้นสัญญาณที่นำมาใช้ได้จึงมีแค่ A2-A9 เท่านั้นส่วน A0-A1 นั้นจะถูกต่อเข้ากับขา A0 และ A1 ของ 8255 โดยตรง



รูปที่ 3.17 แสดงตำแหน่งขาของไอซี



รูปที่ 3.18 แสดงการต่อวงจรรวมทั้งหมดของอินเทอร์เฟสการ์ด

จากรูปวงจรรูปที่ 3.18 จะสังเกตเห็นว่าจะนำสัญญาณ A0-A9, AEN, $\overline{IOR}$, $\overline{IOW}$, RESET จากสล็อตเสริมมาใช้เท่านั้น ส่วนสัญญาณอื่นๆไม่นำมาใช้

การทำงานของ74LS688 ไอซีนี้จะทำการเปรียบเทียบสัญญาณอินพุต 2 ส่วน คือ P0-P7 และ Q0-Q7 จะให้เอาท์พุทออกมาเป็น Logic " 0 "เมื่อ P=Q สัญญาณ Q ต่ออยู่กับ DIP SW ในการเลือกแอดเดรสของการ์ดเลือกโดยการตั้งสวิทซ์ซึ่งในโครงงานนี้จะใช้แอดเดรสช่วง 300H – 31FH เมื่อมีแอดเดรสค่า 300H มาเข้าที่อินพุตของ 74LS688 ที่ขาเอาท์พุทจะให้เอาท์พุทออก
เป็น " 0 " ทำให้ CS ของ 8255 ทำงาน 8255จะต่อ Data Bus ของตัวเองเข้ากับ Data Bus ของคอมพิวเตอร์โดยผ่านทางสล็อตเสริม สำหรับแอดเดรสที่ใช้มีอยู่ 4 ตำแหน่ง คือ

300H ควบคุมการทำงานของพอร์ท A

301H ควบคุมการทำงานของพอร์ท B

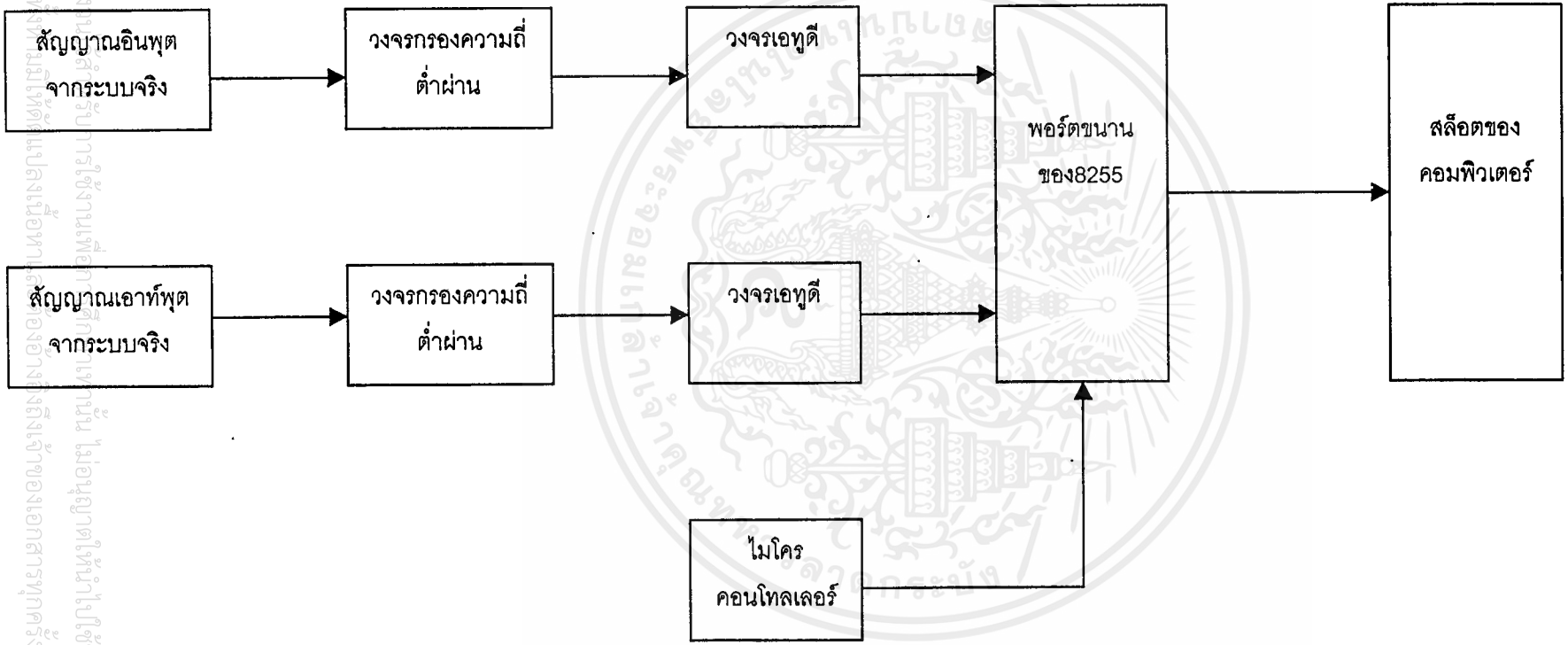
302H ควบคุมการทำงานของพอร์ท C

303H ส่ง Control Word ให้แก่ 8255

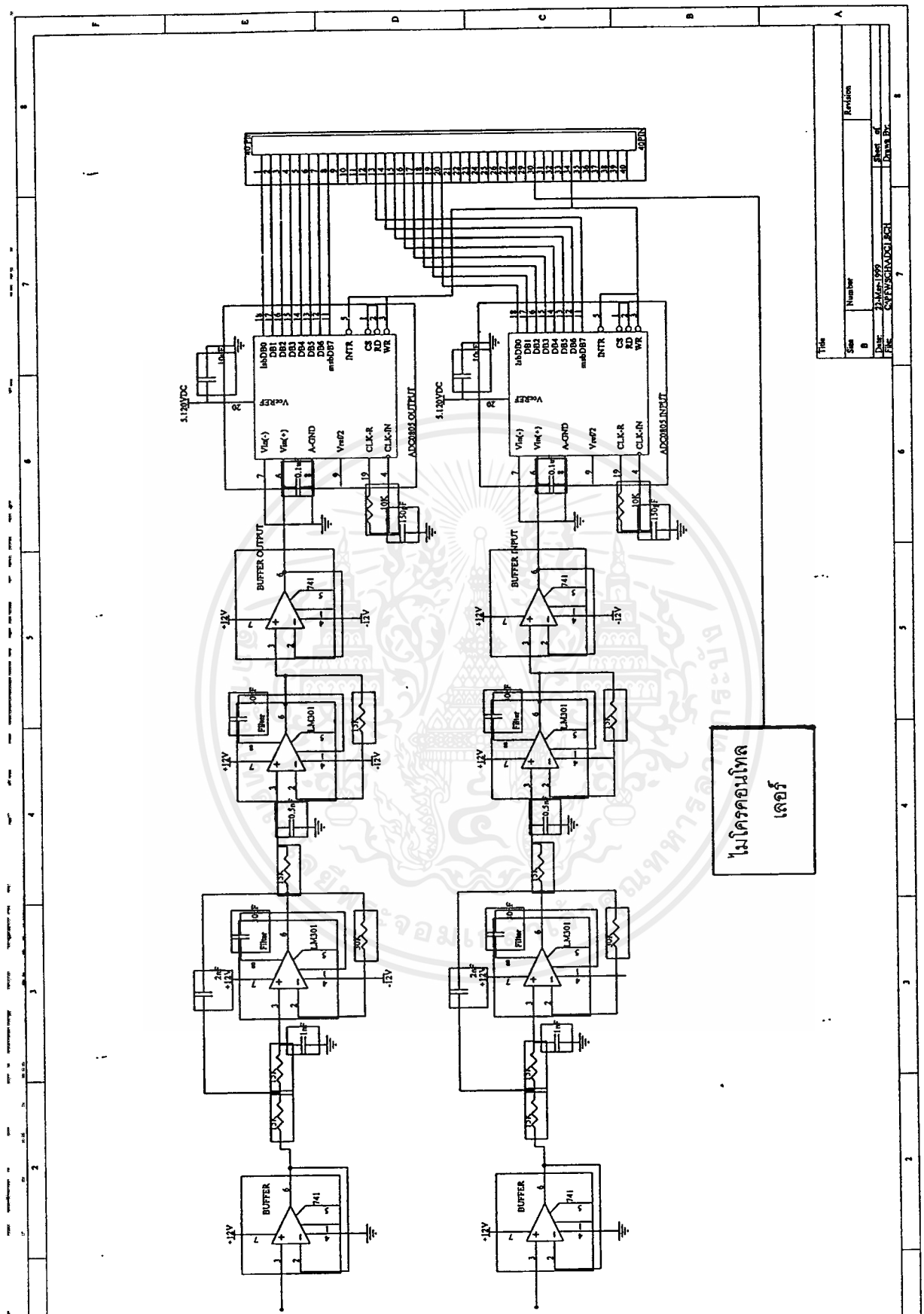
การใช้งาน 8255 Interface Card ขั้นแรกต้องทำการตั้งแอดเดรสของ Card ก่อนโดยที่จะทำการตั้ง DIP SW ซึ่งการตั้งแอดเดรสนั้นต้องตั้งบิท A8 และ A9 เป็น " 1 " เพื่อให้ค่าเป็น 300H lesiy [A0, A1 จะนำไปใช้ในการเซ็ตการทำงานแต่ละพอร์ทของ 8255 ว่าจะให้ทำงานเป็นอินพุตหรือเอาท์พุตต่อไป

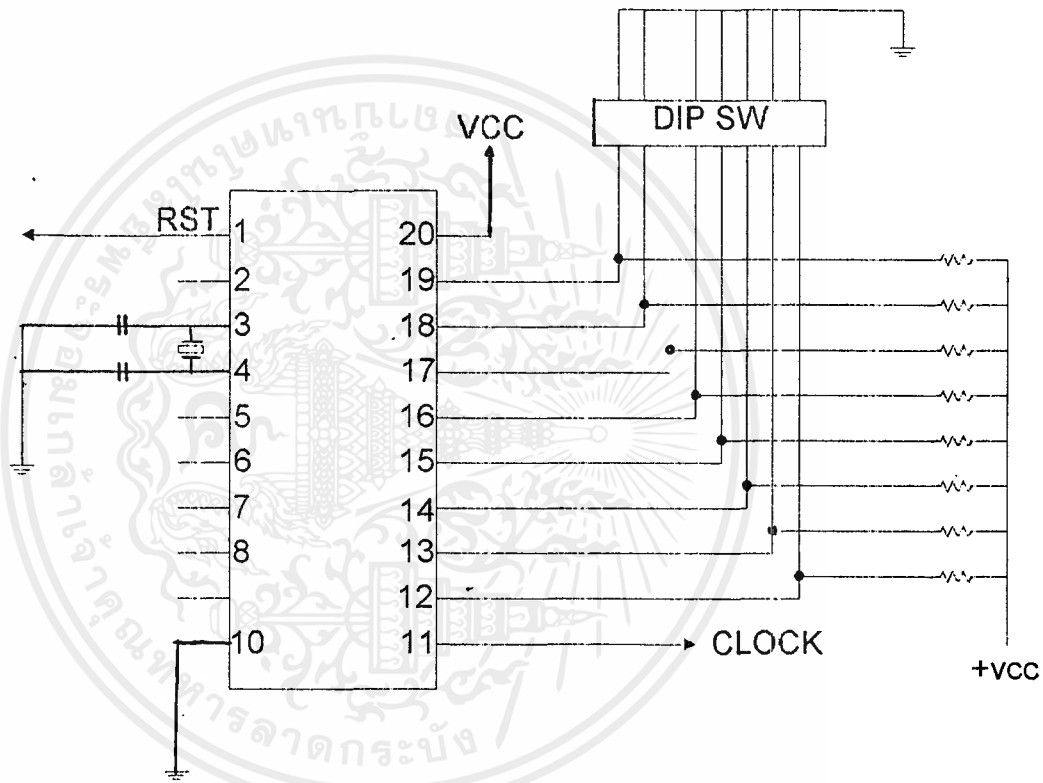
3.6 การทำงานของวงจรทั้งหมดที่ใช้ในการหาพารามิเตอร์ของระบบ

เริ่มต้นวงจรของเราจะทำการรับสัญญาณทั้งสัญญาณอินพุตและเอาท์พุตจากระบบจริงโดยผ่านเข้าบัพเฟอร์แล้วส่งต่อไปยังวงจรกรองความถี่ต่ำผ่านจากนั้นจะส่งต่อไปยังวงจรเอทูดีซึ่งสัญญาณอินพุตและเอาท์พุตจะแยกเข้าวงจรเอทูดีคนละตัวกันหลังจากนั้นสัญญาณดิจิตอลที่ได้จากวงจรเอทูดีทั้ง 2 ตัวจะถูกส่งต่อไปยังการ์ดอินเทอร์เฟสโดยอินพุตของระบบจริงจะเข้าสู่พอร์ต A ของ 8255 ส่วนเอาท์พุตของระบบจริงจะเข้าสู่พอร์ต B ของ 8255 โดยเราใช้ Control Word (93H) ควบคุมให้พอร์ต A,B เป็นอินพุตพอร์ทส่วนพอร์ต C บนและ C ล่างเป็นอินพุตและเอาท์พุตตามลำดับ โดยพอร์ต C ล่าง(C3) จะทำการรับสัญญาณClock จากไมโครคอนโทลเลอร์เพื่อสร้างฐานเวลาให้กับคอมพิวเตอร์ซึ่งนำมาใช้กำหนดคาบเวลาในการชักตัวอย่างเพื่อให้วงจรเอทูดีทำงานได้ตามคาบเวลาในการชักตัวอย่างที่กำหนด ซึ่งสามารถแสดงวงจรการทำงานของระบบทั้งหมดได้ดังรูปที่ 3.19



รูปที่ 3.19 Block Diagram ของวงจรทั้งหมดในการหาพารามิเตอร์ของระบบจริง





รูปที่ 3.21 วงจรการสร้างฐานเวลาจากไมโครคอนโทลเลอร์

บทที่ 4

การทดลองและผลการทดลองหาพารามิเตอร์ของระบบ

ปัจจัยสำคัญที่ควรจะพิจารณา ได้แก่

4.1 ชนิดของสัญญาณอินพุท

วิธีการหาค่าพารามิเตอร์บางวิธีจะต้องการสัญญาณอินพุทเฉพาะแบบ เช่น การวิเคราะห์โดเมนความถี่ต้องการสัญญาณอินพุท เป็นสัญญาณ Sinusoid หรือ สำหรับการวิเคราะห์ช่วง Transient สัญญาณอินพุท ที่ต้องการอาจจะเป็น Step หรือ Impulse

สิ่งที่จะพิจารณาถัดมาก็คือ Amplitude ของสัญญาณ การเลือก Amplitude ควรพิจารณาจากผลดังต่อไปนี้

- ในทางปฏิบัติ ระบบส่วนใหญ่จะเป็นระบบที่เป็น Non-Linear และเราสามารถประมาณค่าของระบบโดยใช้ Linear Model ระบบที่เป็น Non-Linear จะมีความเป็น Linear เฉพาะขอบเขตใดขอบหนึ่งของสัญญาณอินพุทเท่านั้น ดังนั้นในการประมาณค่าพารามิเตอร์จึงไม่ควรเลือก Amplitude ของสัญญาณอินพุทที่สูงเกินไป แต่ในทางกลับกันเราสามารถทดลองค่อยๆเพิ่มขนาดของ Amplitude ของสัญญาณอินพุท เพื่อทดลองหาขอบเขตที่ระบบมีความเป็น Linear
- แม้ว่าไม่ควรเลือกขนาดของ Amplitude ของสัญญาณอินพุทที่สูงเกินไป แต่การเลือกขนาดของ Amplitude ของสัญญาณอินพุทมีค่าสูงๆ จะทำให้อัตราส่วนระหว่างสัญญาณ ต่อ สัญญาณรบกวนลดลง ส่งผลให้การประมาณค่าพารามิเตอร์ของระบบมีความถูกต้องมากขึ้น

สิ่งที่สำคัญในการเลือกสัญญาณอินพุท ก็คือ ต้องสนใจว่า สัญญาณอินพุทมีพลังงานส่วนใหญ่อยู่ในแถบความถี่ที่เราสนใจหรือไม่ เช่น มีสัญญาณอินพุท อยู่ 2 สัญญาณ ได้แก่ White noise และ Step จากวิธีการหาค่าพารามิเตอร์แบบ Least Square และจากผลการคำนวณเชิงตัวเลข(System Identification, Petre Stoica, Chapter 12,Page 470) สามารถสรุปได้ว่า

- สัญญาณ Step จะให้การประมาณค่าที่ถูกต้องสำหรับ Sampling time ที่ค่อนข้างมาก และการประมาณค่าของ Static Gain จะมีความถูกต้องมากกว่า White noise
- สัญญาณ White noise จะให้การประมาณค่าที่ถูกต้องกว่าสำหรับช่วงเวลา Sampling Time ที่น้อยๆ

และจากการวิเคราะห์โดเมนความถี่ สามารถสรุปได้ว่าสำหรับสัญญาณอินพุทที่มีความถี่ต่ำ(เช่น Step)จะให้ความถูกต้องในการประมาณค่า Parameter ที่ดีสำหรับ Model ที่มีลักษณะของ Low Frequency Behavior สำหรับโมเดลที่มีลักษณะของ High and Medium frequency

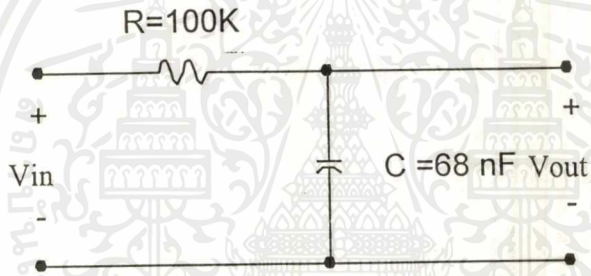
Behavior จะเหมาะสมสำหรับสัญญาณที่มีองค์ประกอบทางความถี่ที่สูงกว่า(White Noise)

4.2 ขนาดของ Sampling interval

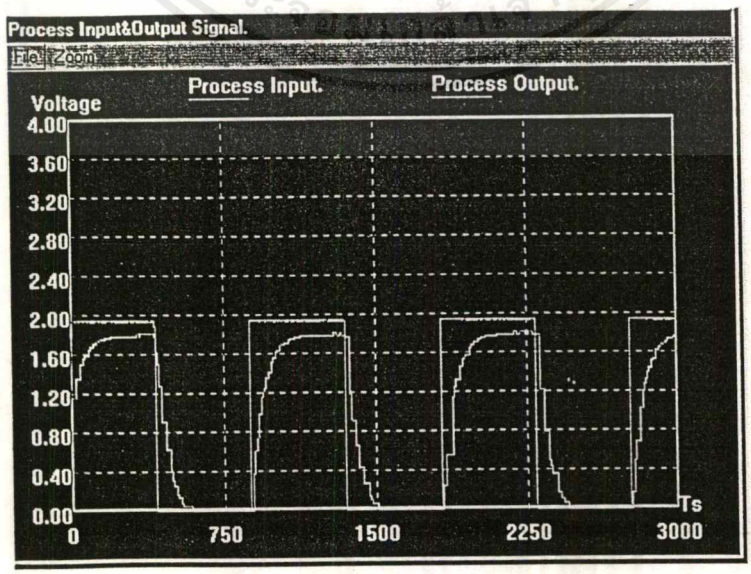
ในการเลือกขนาดของ Sampling interval มีสิ่งที่จะต้องพิจารณาดังนี้

- ขึ้นอยู่กับความถี่ Cut-off ของ Pre-Filter ที่จะนำมาใช้ระหว่างกรองสัญญาณรบกวน และป้องการการเกิด Alias ซึ่งตามหลักแล้ว Sampling frequency ต้องมากกว่า 2 เท่าของความถี่สูงสุดของสัญญาณ ในที่นี้คือค่า Cut-off frequency ของตัว Filter
- ในการใช้ Sampling interval ที่มาก จะทำให้ได้รายละเอียดเกี่ยวกับ High frequency dynamic ค่อนข้างน้อย แต่ถ้า Sampling interval ที่น้อยเกินไปจะทำให้ได้รับรายละเอียดเกี่ยวกับ Low frequency dynamic น้อยเกินไป และยังทำให้สัญญาณรบกวนมีอิทธิพลในการทำการหาพารามิเตอร์ของระบบมากขึ้นด้วย

4.2 ตัวอย่างการทดลอง

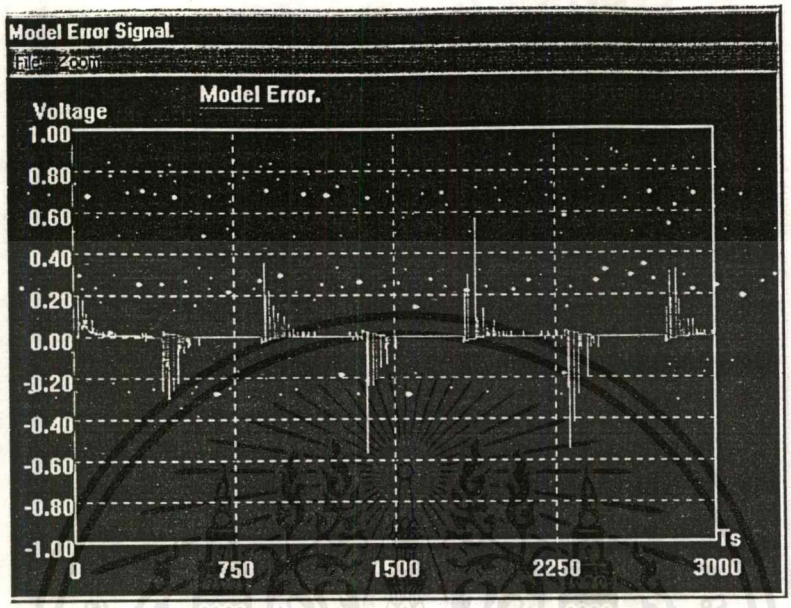


ทำการทดลองหาค่าพารามิเตอร์ของวงจร R-C โดยที่มีค่า R = 100k, C = 68 nF, Sampling time = 0.3 mS, สัญญาณอินพุทเป็น สัญญาณ Step มี Amplitude เท่ากับ 1.9 ความถี่ 4 Hzได้ผลดังนี้



รูป 4.1 กราฟของอินพุท ,เอาท์พุท ของวงจร R-C

กราฟต่อไปนี้เป็นกราฟของ Prediction Error ซึ่งจะสามารถบ่งบอกถึงความถูกต้องของ Model ซึ่ง Prediction Error = Process Output – Model Output



รูป 4.2 กราฟแสดงค่า Prediction Error

4.3 ตัวอย่างการทำ Simulation

4.4.1 ระบบที่มี Order เท่ากับ 2

พิจารณาระบบที่มีสมการผลต่างดังนี้

$y(k) - 0.46y(k-1) - 0.0217y(k-2) = 0.2449x(k-1) + 0.1348x(k-2)$

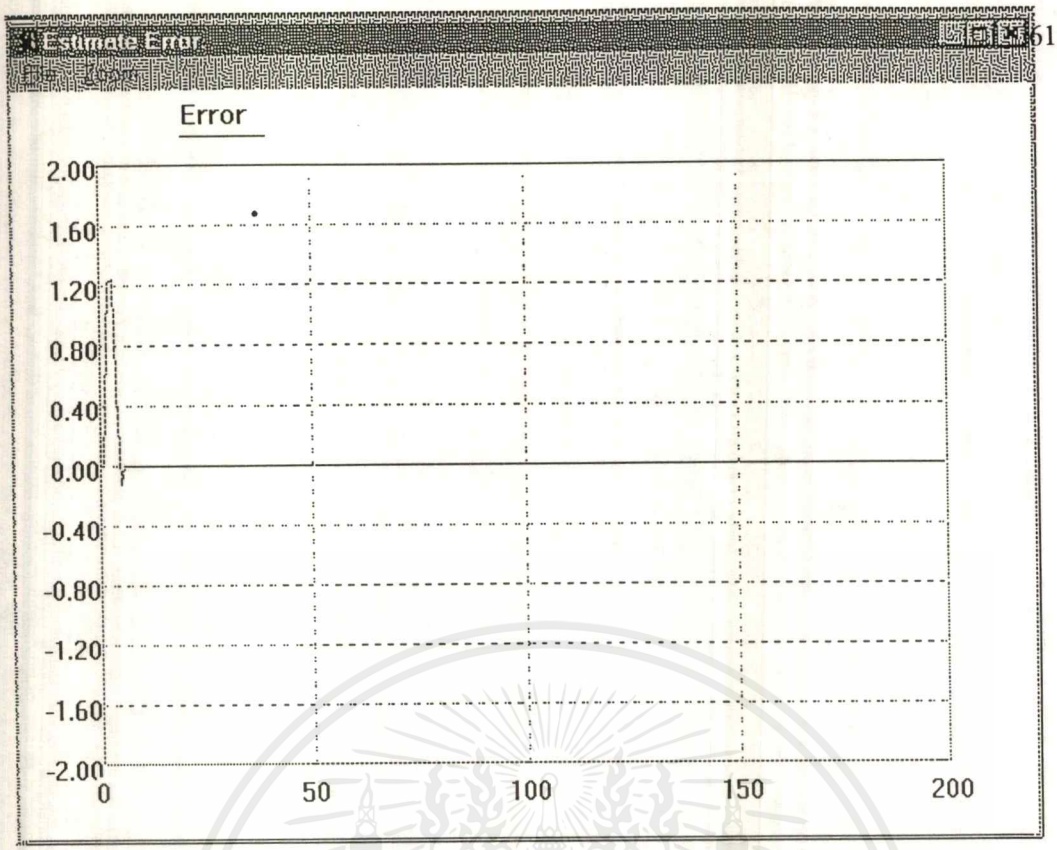
ในการทำ Simulation โดยให้ Order ของโมเดลเท่ากับ 2 ,Sampling frequency เท่ากับ1000 Hz, ใช้ข้อมูลจำนวน 1000 ข้อมูล, ค่า Forgetting factor เท่ากับ 1.0 และค่าของ Adaptive gain เท่ากับ 10000

ให้สัญญาณอินพุทเป็น Square wave มี Amplitude เท่ากับ 5, มีความถี่ 10 Hz

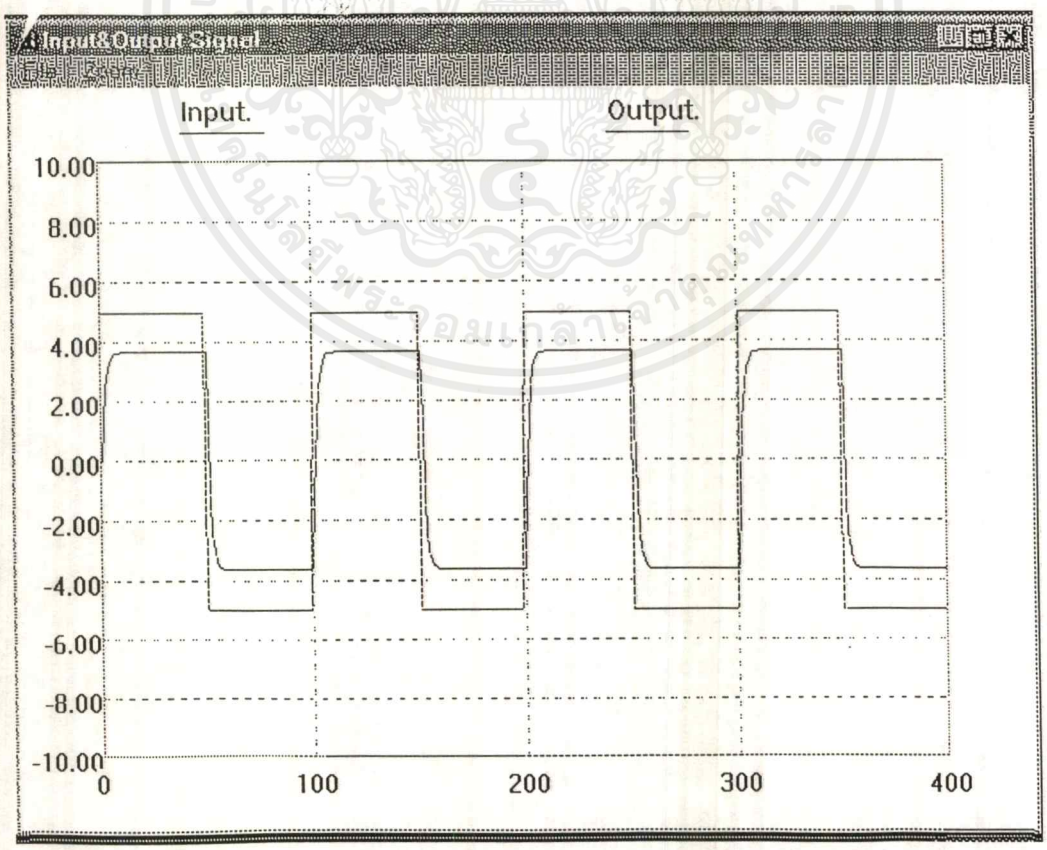
ผลการทดลอง จะได้สมการผลต่างที่คำนวณได้ดังนี้

$y(k) - 0.459996y(k-1) - 0.021702y(k-2) = 0.2449x(k-1) + 0.134801x(k-2)$

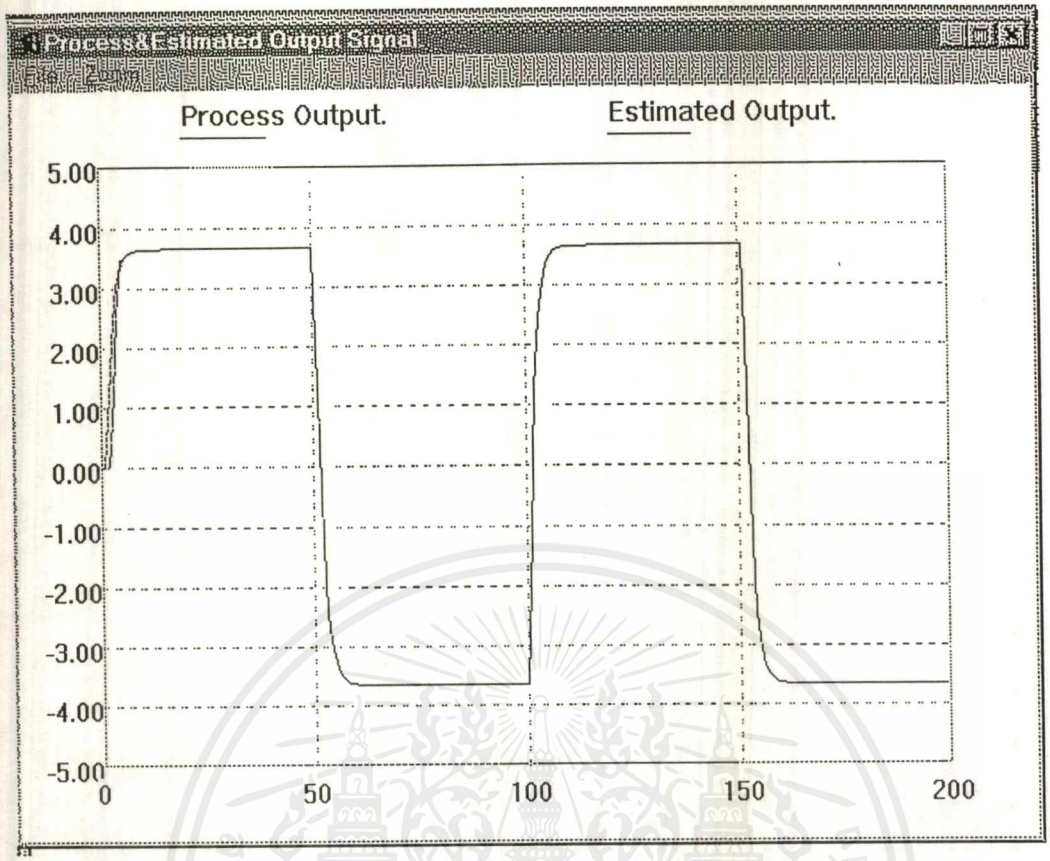
รูปกราฟที่จะแสดงต่อไปนี้เป็นรูปกราฟที่ได้มาจากการทำ Simulation ของระบบ order 2



รูป 4.3 กราฟของ Error ระหว่าง ระบบที่ Simulate กับ โมเดล



รูป 4.4 กราฟเปรียบเทียบระหว่างสัญญาณอินพุทและสัญญาณเอาท์พุท



รูป 4.5 กราฟเปรียบเทียบระหว่างค่าเอาท์พุทที่ได้จากระบบที่ Simulate และจากโมเดล

4.4.1 ระบบที่มี Order เท่ากับ 3

พิจารณาระบบที่มีสมการผลต่างดังนี้

$$y(k) - 1.1558y(k-1) + 1.0088y(k-2) - 0.26448y(k-3) =$$
$$0.137900x(k-1) + 0.3791x(k-2) + 0.071494y(k-3)$$

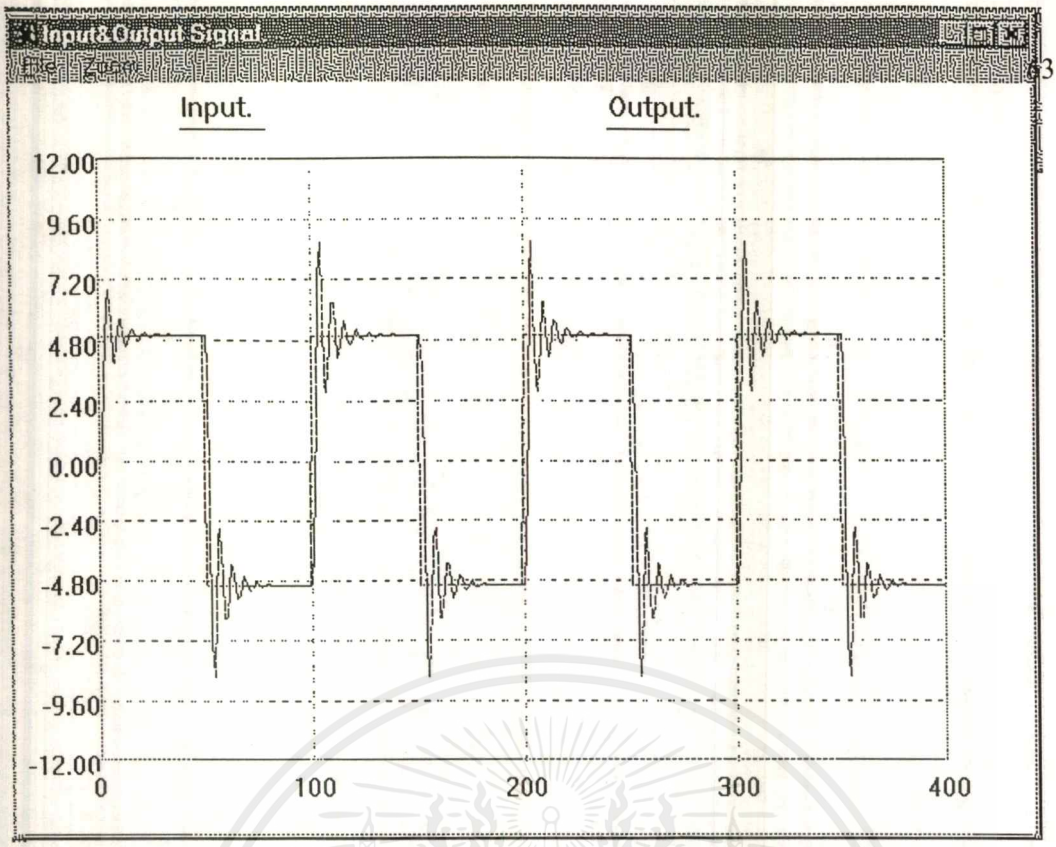
ในการทำ Simulation โดยให้ Order ของโมเดลเท่ากับ 3 ,Sampling frequency เท่ากับ 1000 Hz, ใช้ข้อมูลจำนวน 1000 ข้อมูล, ค่า Forgetting factor เท่ากับ 1.0, และ Adaptive gain เท่ากับ 10000

ให้สัญญาณอินพุทเป็น Square wave มี Amplitude เท่ากับ 5, มีความถี่ 10 Hz

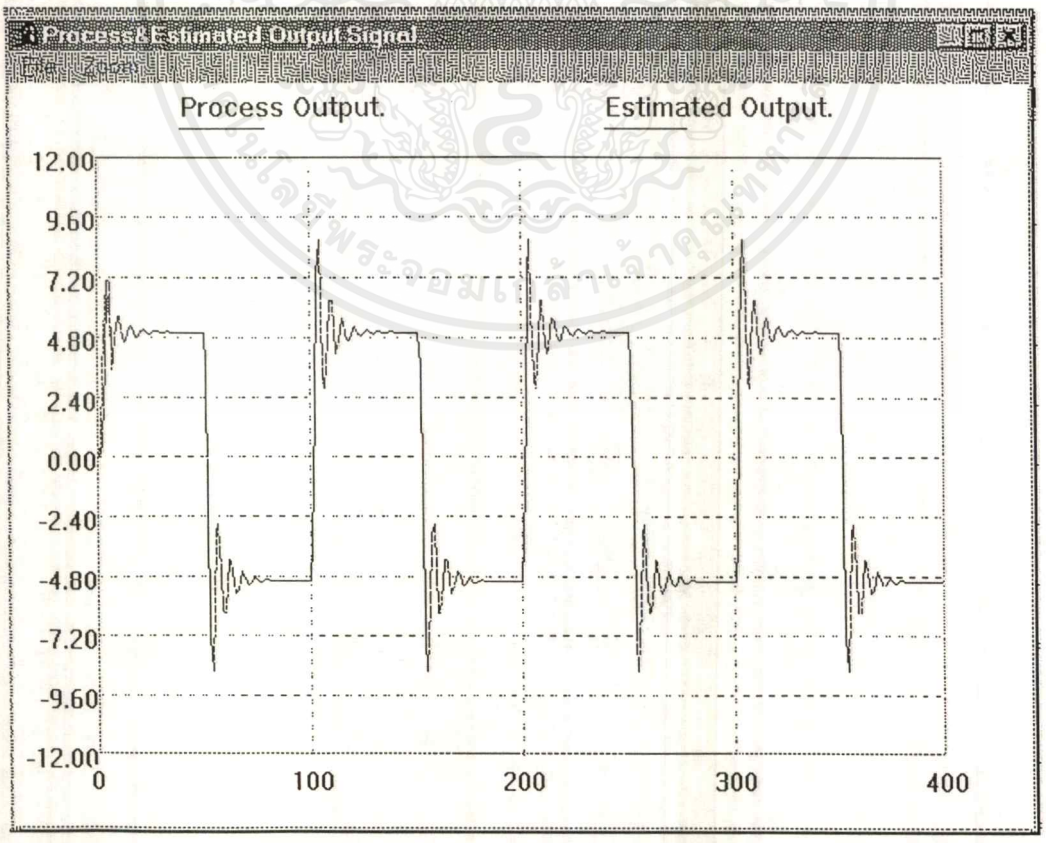
ผลการทดลอง จะได้สมการผลต่างที่คำนวณได้ดังนี้

$$y(k) - 1.155797y(k-1) + 1.008797y(k-2) - 0.264478y(k-3) =$$
$$0.137900x(k-1) + 0.3791x(k-2) + 0.071496y(k-3)$$

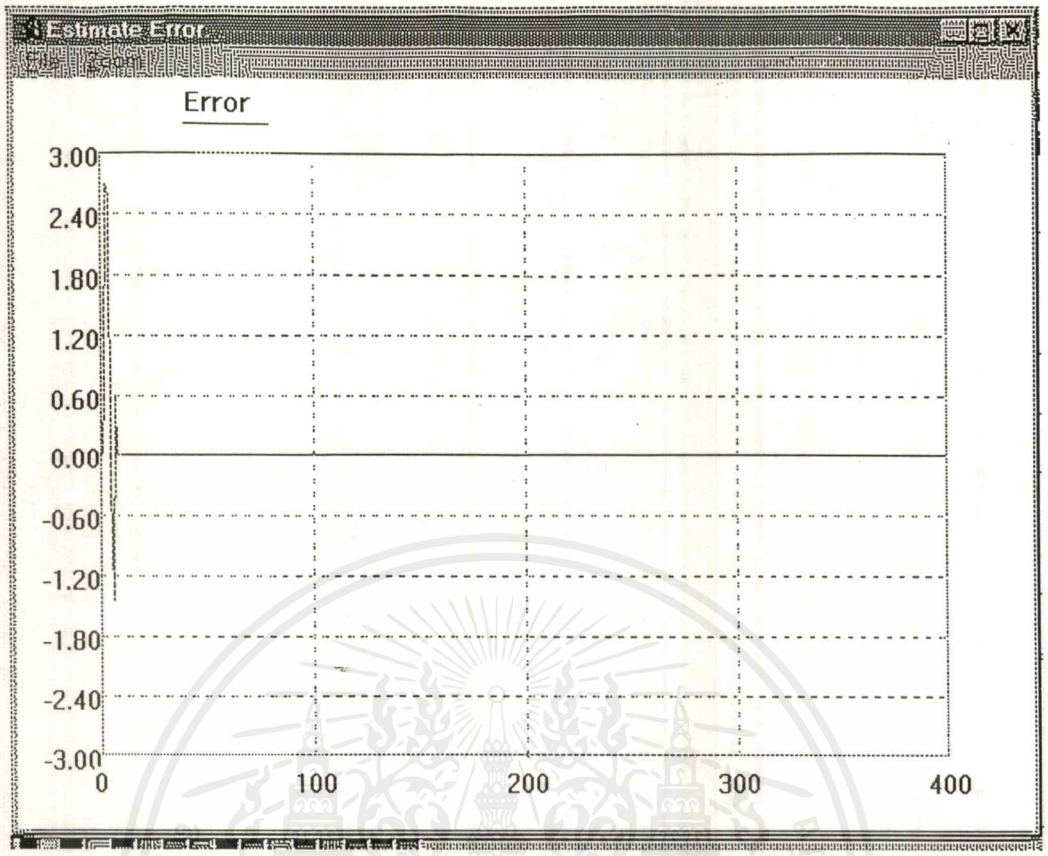
รูปกราฟที่จะแสดงต่อไปนี้เป็นรูปกราฟที่ได้มาจากการทำ Simulation ของระบบ order 3



รูป 4.6 กราฟเปรียบเทียบระหว่างสัญญาณอินพุท และสัญญาณเอาท์พุท



รูป 4.7 กราฟเปรียบเทียบระหว่าง ค่าเอาท์พุทที่ได้จากระบบที่ Simulate และจากโมเดล



รูป 4.8กราฟของ Error ระหว่าง ระบบที่ Simulate กับ โมเดล

บทที่ 5
บทสรุปและวิจารณ์

1. สรุป

การหาพารามิเตอร์ของระบบมีความจำเป็นอย่างยิ่งสำหรับการที่จะเริ่มทำการควบคุมระบบใดๆด้วยวิธีการ Adaptive ซึ่งจะมีข้อดี คือ สามารถควบคุมระบบที่มีสภาพแวดล้อมที่มีการเปลี่ยนแปลงอันเป็นผลให้พารามิเตอร์ของระบบเปลี่ยนไป ด้วยวิธีการ Adaptive นี้เองทำให้เราไม่ต้องไปปรับตัวควบคุมใหม่ โดยตัวควบคุมจะปรับตัวเองตามค่าพารามิเตอร์ที่เปลี่ยนไปส่งผลให้ระบบยังคงมีผลตอบสนองที่ต้องการ

2. ปัญหาที่พบ

2.1 ความถี่ของการ Sampling ของ A/D มีค่าน้อยเกินไปสำหรับระบบที่มีค่า Time Constant น้อยๆ เช่น วงจร R-C ซึ่งมี Time Constant เท่ากับ 6.8 ms

2.2 ความถูกต้องของโมเดล โมเดลจะมีความถูกต้องเฉพาะเวลาหนึ่งๆเท่านั้น จึงมีความจำเป็นที่จะต้องทำการหาพารามิเตอร์ของระบบตลอดเวลา

2.3 หนังสือที่เกี่ยวกับการเขียนโปรแกรมด้วย Borland C++ 5.0 บน Windows มีน้อยมาก จึงเป็นการยากที่จะค้นคว้าพัฒนาโปรแกรมการหาค่าพารามิเตอร์ของระบบ

2.4 เนื่องจากต้องเสียเวลาพัฒนาวงจร A/D จึงทำให้มีเวลาในการศึกษาเกี่ยวกับการหาค่าพารามิเตอร์ของระบบมีน้อยลง

2.5 เนื่องจากความซับซ้อนทางคณิตศาสตร์ จึงเป็นการยากที่จะศึกษาเกี่ยวกับการหาค่าพารามิเตอร์ของระบบ

3. ข้อเสนอแนะ

แม้ว่าการทำการหาพารามิเตอร์ของระบบโดยใช้วิธี Least Square จะมีผลที่น่าพอใจในระดับหนึ่ง แต่ก็ต้องการการพัฒนาต่อไป งานที่ควรพัฒนาต่อไปมีดังนี้

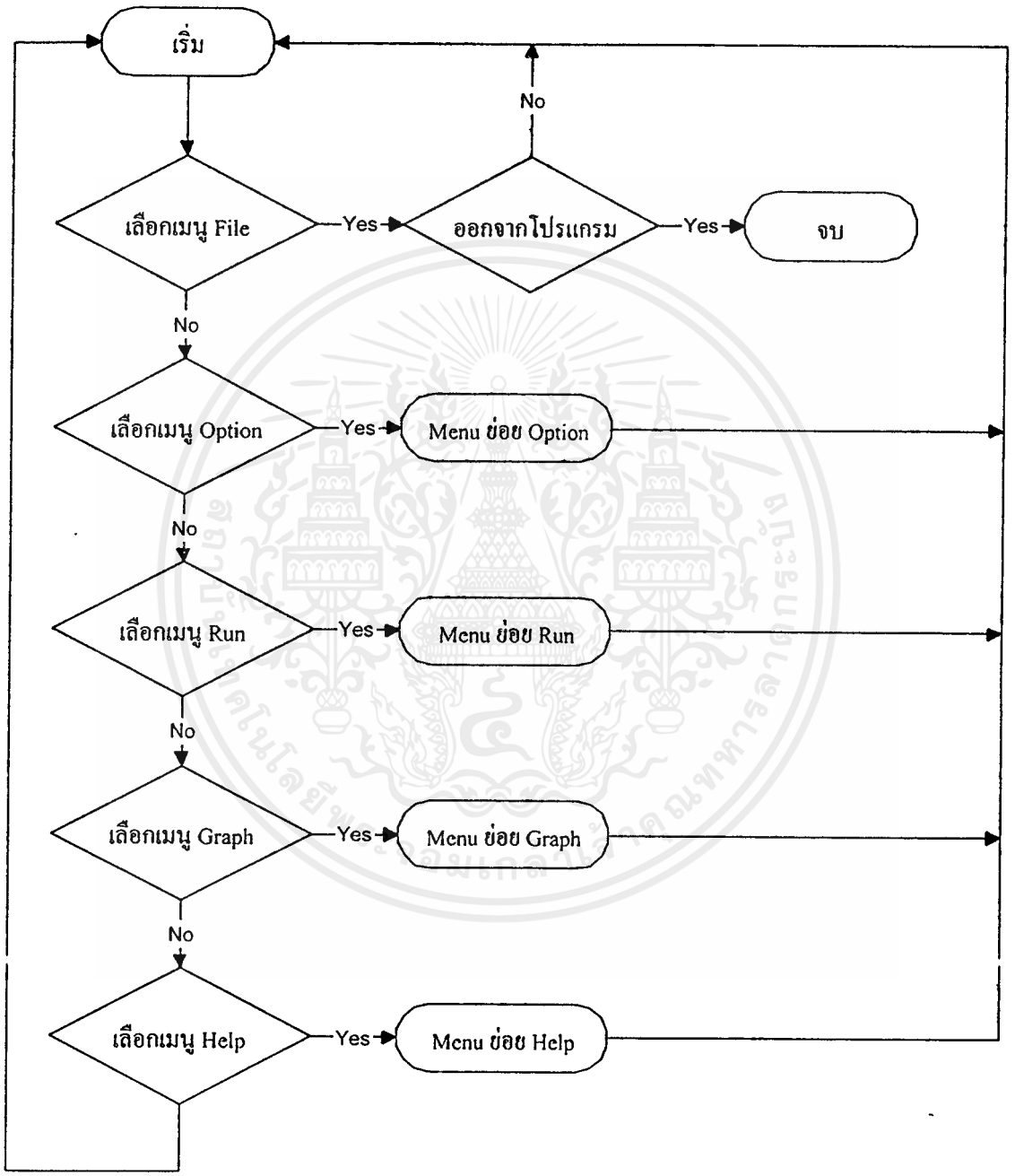
2.1 พัฒนาเพื่อใช้ในการทำการหาค่าพารามิเตอร์แบบ Real - Time โดยอาจพัฒนา Software และ Hardware ขึ้นมาใหม่ให้ดีขึ้น โดยใช้ต้นแบบที่มีอยู่แล้ว

2.2 ศึกษาถึงทฤษฎี Digital Signal Process เพื่อที่จะศึกษาถึงระบบ Digital และนำความรู้ที่ได้มาพัฒนาต่อไป

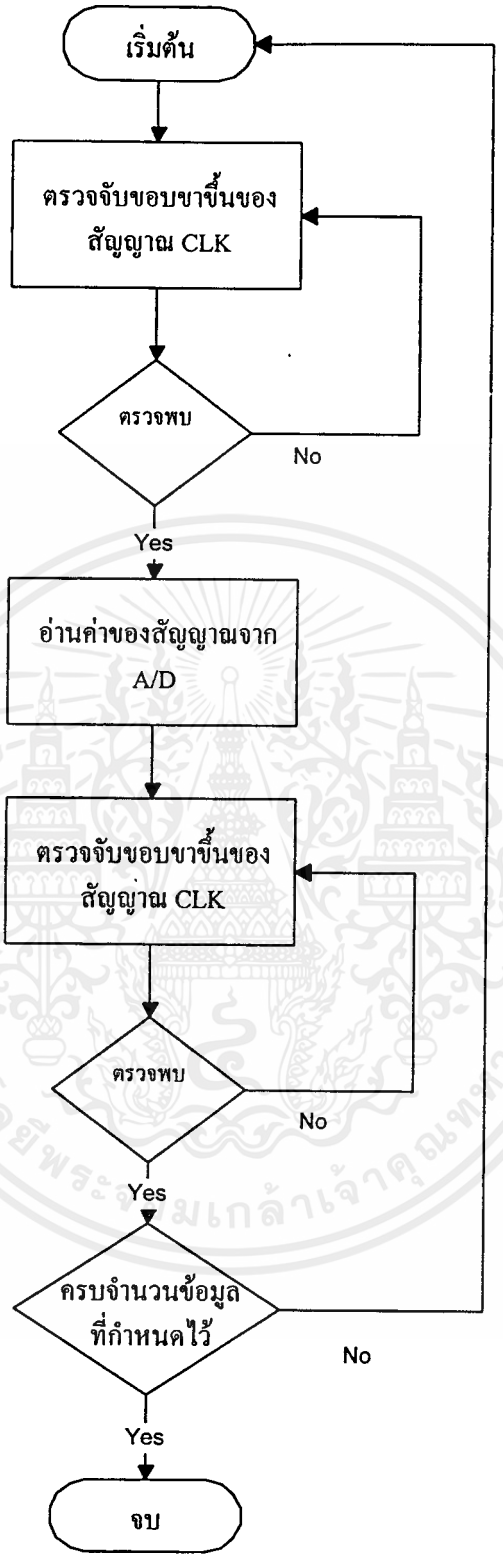
2.3 ศึกษาถึงวิธีการหาค่าพารามิเตอร์วิธีอื่น เช่น วิธี IV เนื่องจากแต่ละวิธีการจะมีข้อดี-ข้อเสียแตกต่างกันไป

2.4 จาก (2.3) ในการหาค่าพารามิเตอร์อาจจะใช้มากกว่าหนึ่งวิธีได้ เช่น อาจจะใช้วิธี Least Square ในช่วงแรกๆ เพื่อให้ได้ค่าพารามิเตอร์ที่มีค่าที่ใกล้เคียง และใช้วิธี IV ต่อไป เพื่อที่จะใช้กับระบบที่มี Noise ที่ไม่ใช่ White noise

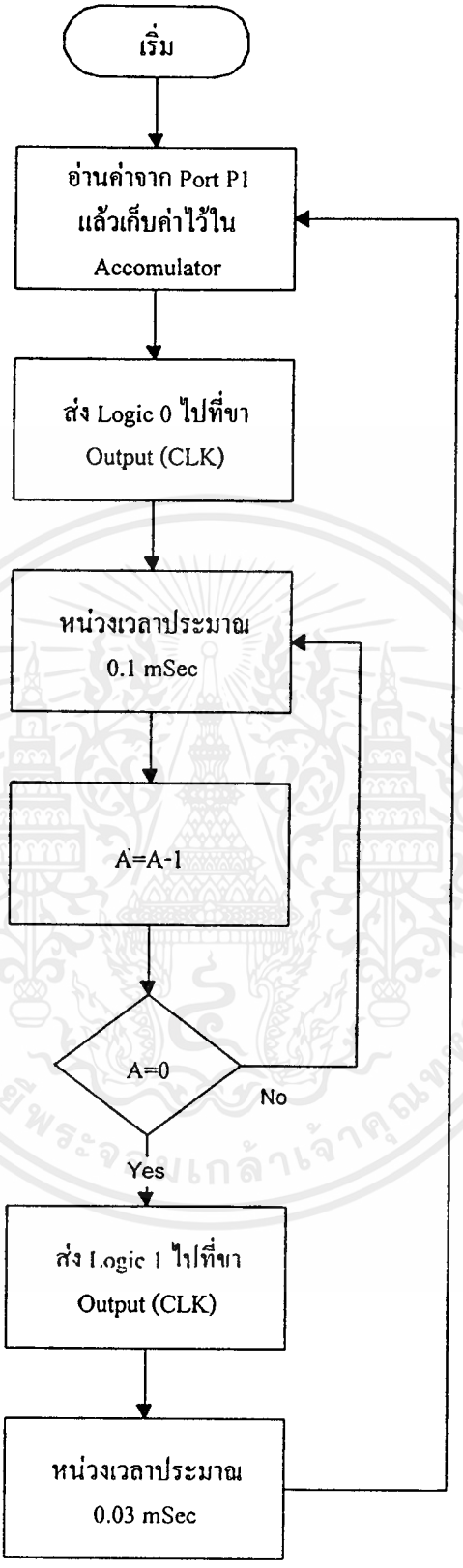
ภาคผนวก



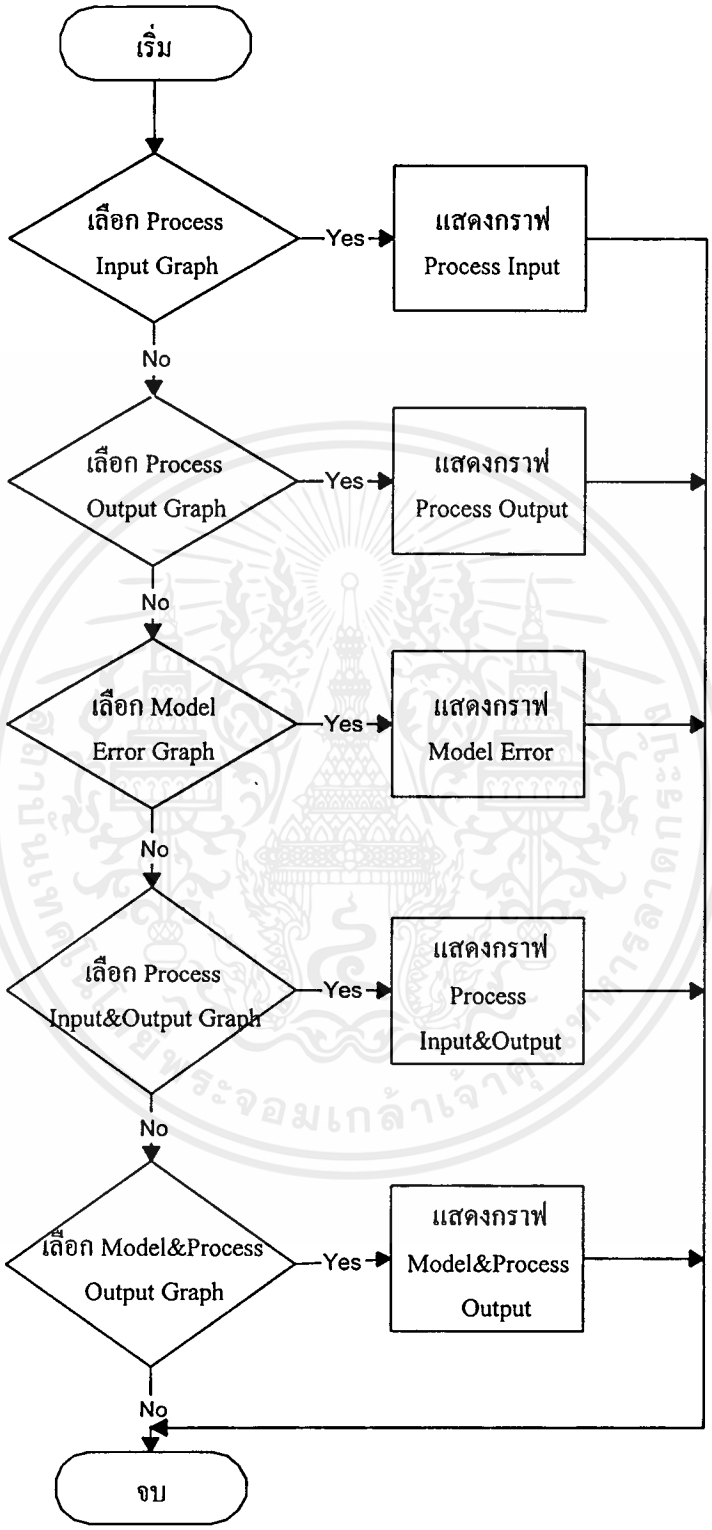
โฟลวชาร์ตการทำงานของเมนูหลัก



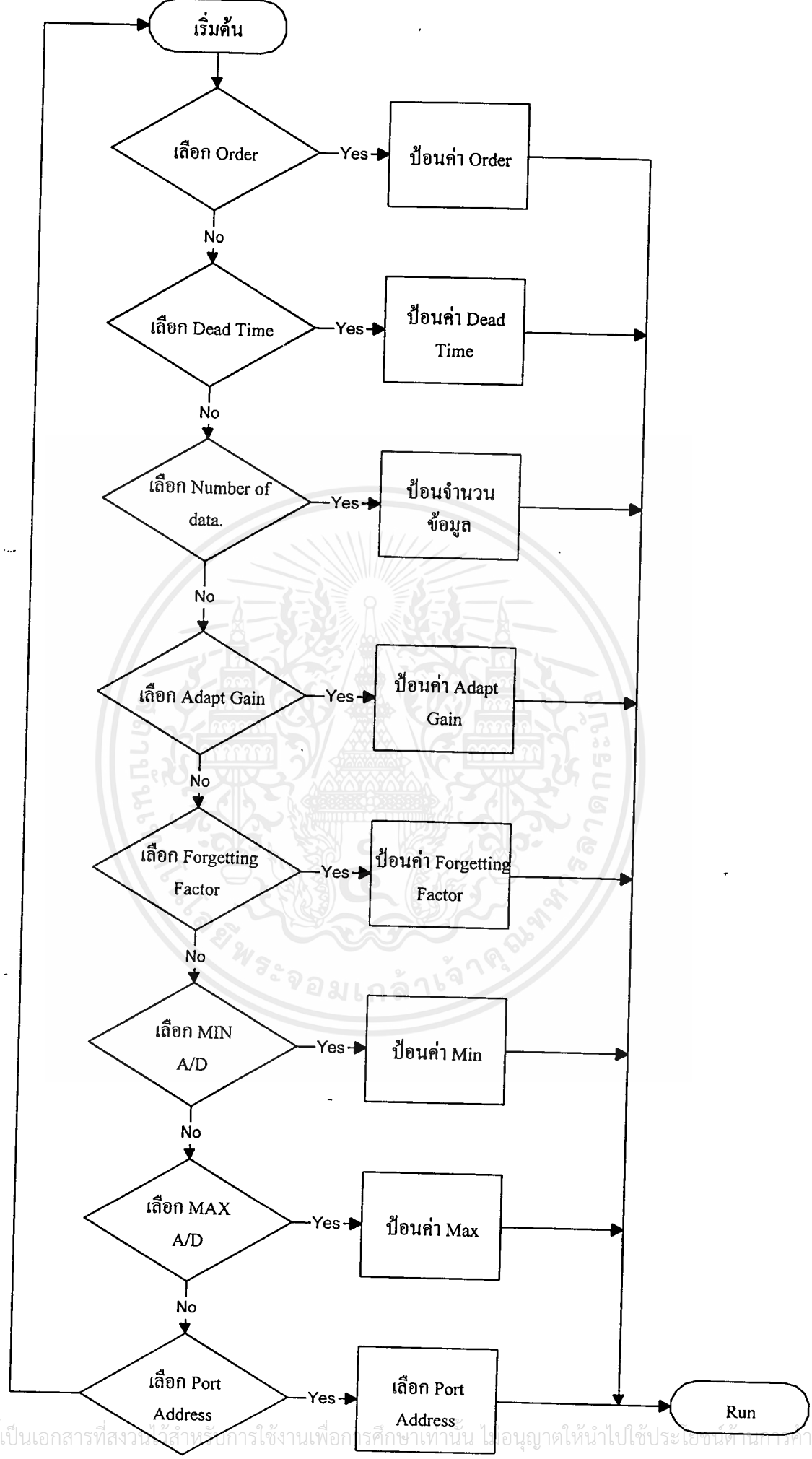
โฟวลชาร์ตการรับค่า ข้อมูลจาก A/D



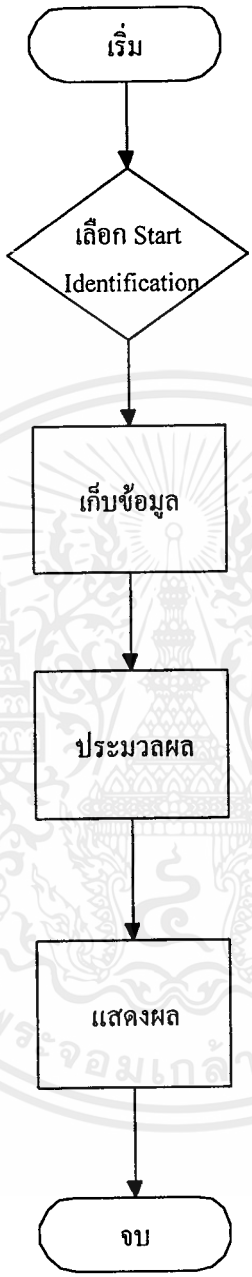
โฟลวชาร์ตของ ซอฟท์แวร์การสร้าง CLOCK ของ MCS - 51



โฟลวชาร์ตแสดงเมนูย่อยกราฟ



โฟลวชาร์ตแสดงการทำงานแบบย่อยของโปรแกรม



โฟลวชาร์ตของเมนูย่อย Run

National Semiconductor

ADC0801/ADC0802/ADC0803/ADC0804/ADC0805
8-Bit μP Compatible A/D Converters

General Description

The ADC0801, ADC0802, ADC0803, ADC0804 and ADC0805 are CMOS 8-bit successive approximation A/D converters that use a differential potentiometric ladder—similar to the 256R products. These converters are designed to allow operation with the NSC800 and INS8080A derivative control bus with TRI-STATE® output latches directly driving the data bus. These A/Ds appear like memory locations or I/O ports to the microprocessor and no interfacing logic is needed.

Differential analog voltage inputs allow increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

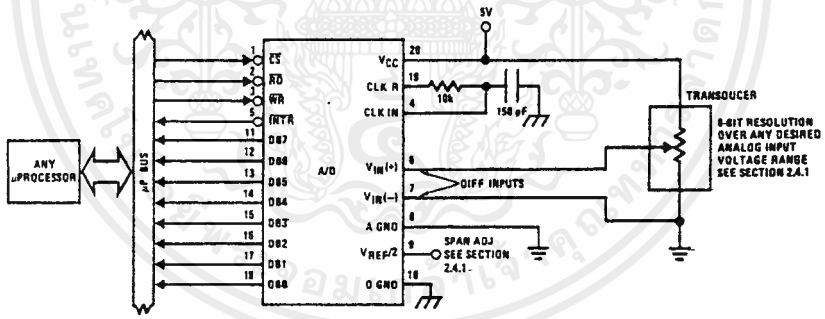
Features

- Compatible with 8080 μP derivatives—no interfacing logic needed - access time - 135 ns
- Easy interface to all microprocessors, or operates "stand alone"
- Differential analog voltage inputs
- Logic inputs and outputs meet both MOS and TTL voltage level specifications
- Works with 2.5V (LM336) voltage reference
- On-chip clock generator
- 0V to 5V analog input voltage range with single 5V supply
- No zero adjust required
- 0.3" standard width 20-pin DIP package
- 20-pin molded chip carrier or small outline package
- Operates ratiometrically or with 5 V_{DC}, 2.5 V_{DC}, or analog span adjusted voltage reference

Key Specifications

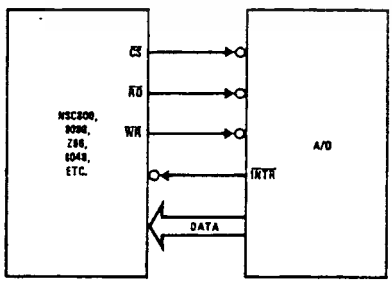
- Resolution 8 bits
- Total error ±¼ LSB, ±½ LSB and ±1 LSB
- Conversion time 100 μs

Typical Applications



TL/H/5671–1

8080 Interface



TL/H/5671–31

Error Specification (Includes Full-Scale, Zero Error, and Non-Linearity)			
Part Number	Full-Scale Adjusted	$V_{REF}/2 = 2.500\ V_{DC}$ (No Adjustments)	$V_{REF}/2$ = No Connection (No Adjustments)
ADC0801	±¼ LSB		
ADC0802		±½ LSB	
ADC0803	±½ LSB		
ADC0804		±1 LSB	
ADC0805			±1 LSB

Absolute Maximum Ratings (Notes 1 & 2)

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

Supply Voltage (V_{CC}) (Note 3)	6.5V
Voltage	
Logic Control Inputs	−0.3V to +18V
At Other Input and Outputs	−0.3V to (V_{CC} + 0.3V)
Lead Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (plastic)	260°C
Dual-In-Line Package (ceramic)	300°C
Surface Mount Package	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
Storage Temperature Range	−65°C to +150°C
Package Dissipation at T_A = 25°C	875 mW
ESD Susceptibility (Note 10)	800V

Operating Ratings (Notes 1 & 2)

Temperature Range	$T_{MIN} \le T_A \le T_{MAX}$
ADC0801/02LJ, ADC0802LJ/883	−55°C ≤ T_A ≤ +125°C
ADC0801/02/03/04LCJ	−40°C ≤ T_A ≤ +85°C
ADC0801/02/03/05LCN	−40°C ≤ T_A ≤ +85°C
ADC0804LCN	0°C ≤ T_A ≤ +70°C
ADC0802/03/04LCV	0°C ≤ T_A ≤ +70°C
ADC0802/03/04LCWM	0°C ≤ T_A ≤ +70°C
Range of V_{CC}	4.5 V_{DC} to 6.3 V_{DC}

Electrical Characteristics

The following specifications apply for V_{CC} = 5 V_{DC}, $T_{MIN} \le T_A \le T_{MAX}$ and f_{CLK} = 640 kHz unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
ADC0801: Total Adjusted Error (Note 8)	With Full-Scale Adj. (See Section 2.5.2)			±¼	LSB
ADC0802: Total Unadjusted Error (Note 8)	$V_{REF}/2$ = 2.500 V_{DC}			±½	LSB
ADC0803: Total Adjusted Error (Note 8)	With Full-Scale Adj.. (See Section 2.5.2)			±½	LSB
ADC0804: Total Unadjusted Error (Note 8)	$V_{REF}/2$ = 2.500 V_{DC}			±1	LSB
ADC0805: Total Unadjusted Error (Note 8)	$V_{REF}/2$-No Connection			±1	LSB
$V_{REF}/2$ Input Resistance (Pin 9)	ADC0801/02/03/05 ADC0804 (Note 9)	2.5 0.75	8.0 1.1		kΩ kΩ
Analog Input Voltage Range	(Note 4) V(+) or V(−)	Gnd−0.05		V_{CC} + 0.05	V_{DC}
DC Common-Mode Error	Over Analog Input Voltage Range		±1/16	±⅛	LSB
Power Supply Sensitivity	V_{CC} = 5 V_{DC} ±10% Over Allowed V_{IN}(+) and V_{IN}(−) Voltage Range (Note 4)		±1/16	±⅛	LSB

AC Electrical Characteristics

The following specifications apply for V_{CC} = 5 V_{DC} and T_A = 25°C unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_C	Conversion Time	f_{CLK} = 640 kHz (Note 6)	103		114	μs
T_C	Conversion Time	(Note 5, 6)	66		73	$1/f_{CLK}$
f_{CLK}	Clock Frequency Clock Duty Cycle	V_{CC} = 5V, (Note 5) (Note 5)	100 40	640	1460 60	kHz %
CR	Conversion Rate in Free-Running Mode	$\overline{INTR}$ tied to $\overline{WR}$ with $\overline{CS}$ = 0 V_{DC}, f_{CLK} = 640 kHz	8770		9708	conv/s
$t_{W(\overline{WR})L}$	Width of $\overline{WR}$ Input (Start Pulse Width)	$\overline{CS}$ = 0 V_{DC} (Note 7)	100			ns
t_{ACC}	Access Time (Delay from Falling Edge of $\overline{RD}$ to Output Data Valid)	C_L = 100 pF		135	200	ns
t_{1H}, t_{0H}	TRI-STATE Control (Delay from Rising Edge of $\overline{RD}$ to Hi-Z State)	C_L = 10 pF, R_L = 10k (See TRI-STATE Test Circuits)		125	200	ns
t_{WI}, t_{RI}	Delay from Falling Edge of $\overline{WR}$ or $\overline{RD}$ to Reset of $\overline{INTR}$			300	450	ns
C_{IN}	Input Capacitance of Logic Control Inputs			5	7.5	pF
C_{OUT}	TRI-STATE Output Capacitance (Data Buffers)			5	7.5	pF
CONTROL INPUTS [Note: CLK IN (Pin 4) is the input of a Schmitt trigger circuit and is therefore specified separately]						
V_{IN} (1)	Logical "1" Input Voltage (Except Pin 4 CLK IN)	V_{CC} = 5.25 V_{DC}	2.0		15	V_{DC}

AC Electrical Characteristics (Continued)

The following specifications apply for $V_{CC} = 5V_{DC}$ and $T_{MIN} \leq T_A \leq T_{MAX}$, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
CONTROL INPUTS [Note: CLK IN (Pin 4) is the input of a Schmitt trigger circuit and is therefore specified separately]						
$V_{IN}(0)$	Logical "0" Input Voltage (Except Pin 4 CLK IN)	$V_{CC} = 4.75\ V_{DC}$			0.8	V_{DC}
$I_{IN}(1)$	Logical "1" Input Current (All Inputs)	$V_{IN} = 5\ V_{DC}$		0.005	1	μA_{DC}
$I_{IN}(0)$	Logical "0" Input Current (All Inputs)	$V_{IN} = 0\ V_{DC}$	−1	−0.005		μA_{DC}
CLOCK IN AND CLOCK R						
V_T+	CLK IN (Pin 4) Positive Going Threshold Voltage		2.7	3.1	3.5	V_{DC}
V_T-	CLK IN (Pin 4) Negative Going Threshold Voltage		1.5	1.8	2.1	V_{DC}
V_H	CLK IN (Pin 4) Hysteresis $(V_T+)-(V_T-)$		0.6	1.3	2.0	V_{DC}
$V_{OUT}(0)$	Logical "0" CLK R Output Voltage	$I_O = 360\ \mu A$ $V_{CC} = 4.75\ V_{DC}$			0.4	V_{DC}
$V_{OUT}(1)$	Logical "1" CLK R Output Voltage	$I_O = -360\ \mu A$ $V_{CC} = 4.75\ V_{DC}$	2.4			V_{DC}
DATA OUTPUTS AND $\overline{INTR}$						
$V_{OUT}(0)$	Logical "0" Output Voltage Data Outputs $\overline{INTR}$ Output	 $I_{OUT} = 1.6$ mA, $V_{CC} = 4.75\ V_{DC}$ $I_{OUT} = 1.0$ mA, $V_{CC} = 4.75\ V_{DC}$			 0.4 0.4	 V_{DC} V_{DC}
$V_{OUT}(1)$	Logical "1" Output Voltage	$I_O = -360\ \mu A$, $V_{CC} = 4.75\ V_{DC}$	2.4			V_{DC}
$V_{OUT}(1)$	Logical "1" Output Voltage	$I_O = -10\ \mu A$, $V_{CC} = 4.75\ V_{DC}$	4.5			V_{DC}
I_{OUT}	TRI-STATE Disabled Output Leakage (All Data Buffers)	$V_{OUT} = 0\ V_{DC}$ $V_{OUT} = 5\ V_{DC}$	−3		 3	μA_{DC} μA_{DC}
I_{SOURCE}		V_{OUT} Short to Gnd, $T_A = 25°C$	4.5	6		mA_{DC}
I_{SINK}		V_{OUT} Short to V_{CC}, $T_A = 25°C$	9.0	16		mA_{DC}
POWER SUPPLY						
I_{CC}	Supply Current (Includes Ladder Current) ADC0801/02/03/04LCJ/05 ADC0804LCN/LCV/LCWM	$f_{CLK} = 640$ kHz, $V_{REF}/2 =$ NC, $T_A = 25°C$ and $\overline{CS} = 5V$		 1.1 1.9	 1.8 2.5	 mA mA

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

Note 2: All voltages are measured with respect to Gnd, unless otherwise specified. The separate A Gnd point should always be wired to the D Gnd.

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

Note 4: For $V_{IN}(-) \geq V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input (see block diagram) which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

Note 5: Accuracy is guaranteed at $f_{CLK} = 640$ kHz. At higher clock frequencies accuracy can degrade. For lower clock frequencies, the duty cycle limits can be extended so long as the minimum clock high time interval or minimum clock low time interval is no less than 275 ns.

Note 6: With an asynchronous start pulse, up to 8 clock periods may be required before the internal clock phases are proper to start the conversion process. The start request is internally latched, see *Figure 2* and section 2.0.

Note 7: The $\overline{CS}$ input is assumed to bracket the $\overline{WR}$ strobe input and therefore timing is dependent on the $\overline{WR}$ pulse width. An arbitrarily wide pulse width will hold the converter in a reset mode and the start of conversion is initiated by the low to high transition of the $\overline{WR}$ pulse (see timing diagrams).

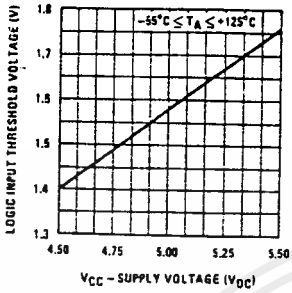
Note 8: None of these A/Ds requires a zero adjust (see section 2.5.1). To obtain zero code at other analog input voltages see section 2.5 and *Figure 5*.

Note 9: The $V_{REF}/2$ pin is the center point of a two-resistor divider connected from V_{CC} to ground. In all versions of the ADC0801, ADC0802, ADC0803, and ADC0805, and in the ADC0804LCJ, each resistor is typically 16 kΩ. In all versions of the ADC0804 except the ADC0804LCJ, each resistor is typically 2.2 kΩ.

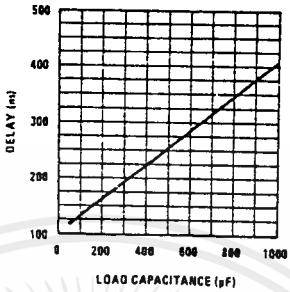
Note 10: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Typical Performance Characteristics

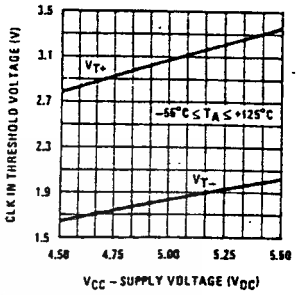
Logic Input Threshold Voltage vs. Supply Voltage



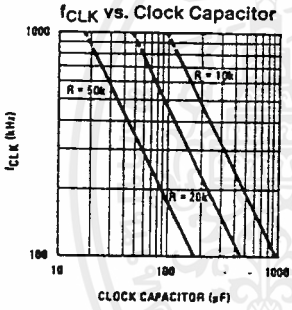
Delay From Falling Edge of $\overline{RD}$ to Output Data Valid vs. Load Capacitance



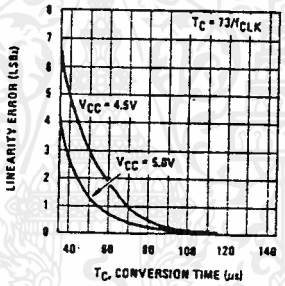
CLK IN Schmitt Trip Levels vs. Supply Voltage



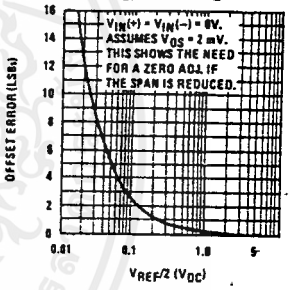
f_{CLK} vs. Clock Capacitor



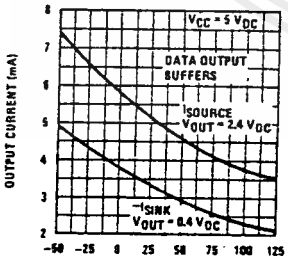
Full-Scale Error vs Conversion Time



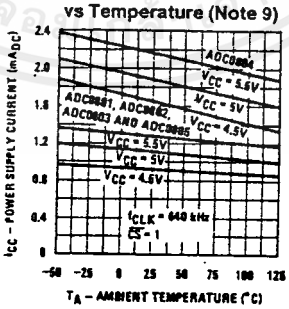
Effect of Unadjusted Offset Error vs. $V_{REF}/2$ Voltage



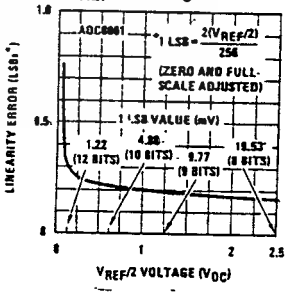
Output Current vs Temperature



Power Supply Current vs Temperature (Note 9)

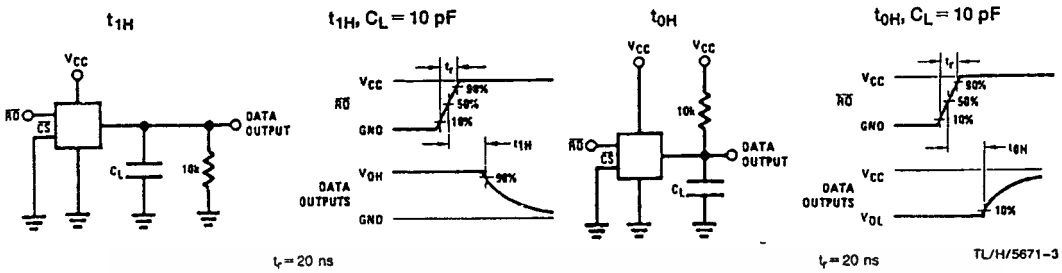


Linearity Error at Low $V_{REF}/2$ Voltages



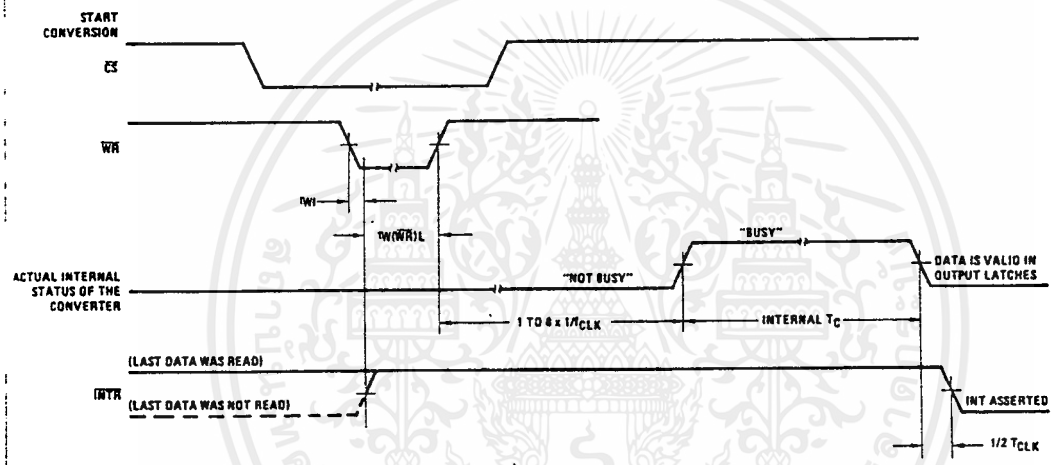
TL/H/5671–2

TRI-STATE Test Circuits and Waveforms

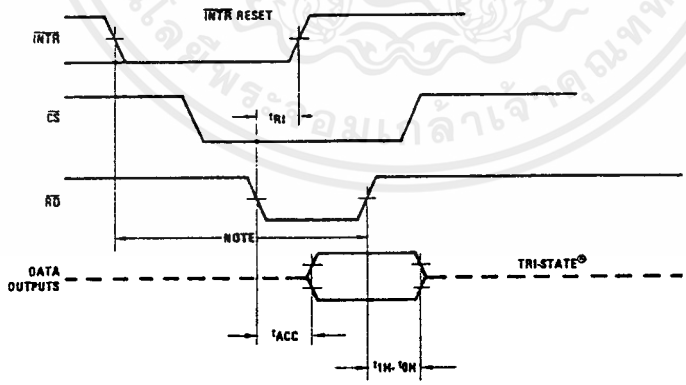


TL/H/5671–3

Timing Diagrams (All timing is measured from the 50% voltage points)



Output Enable and Reset $\overline{INTR}$

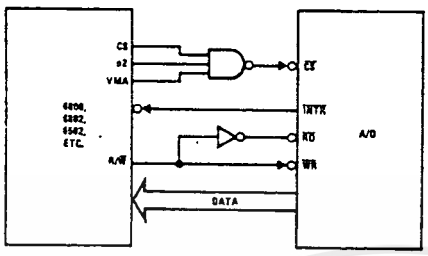


TL/H/5671–4

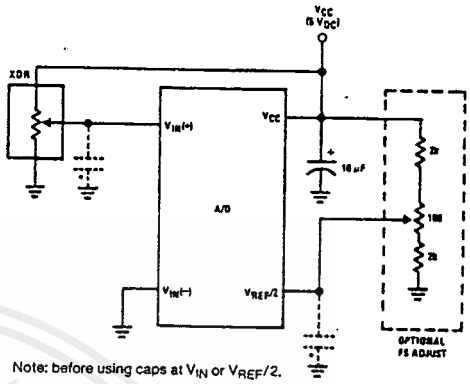
Note: Read strobe must occur 8 clock periods ($8/f_{CLK}$) after assertion of interrupt to guarantee reset of $\overline{INTR}$.

Typical Applications (Continued)

6800 Interface

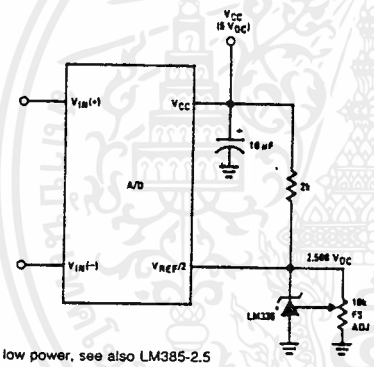


Ratiometric with Full-Scale Adjust



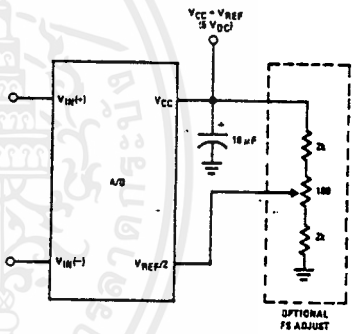
Note: before using caps at V_{IN} or $V_{REF}/2$, see section 2.3.2 Input Bypass Capacitors.

Absolute with a 2.500V Reference

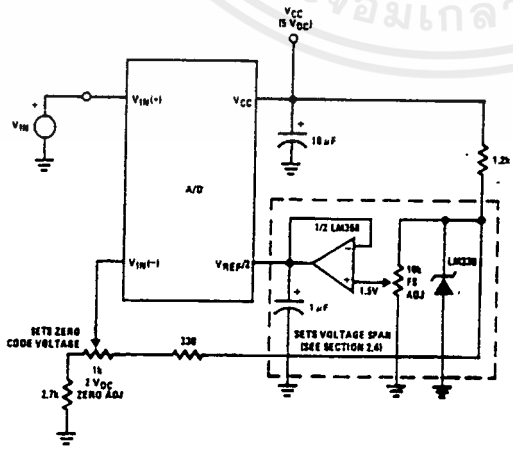


*For low power, see also LM385-2.5

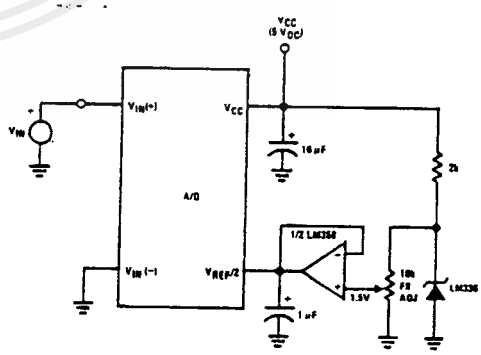
Absolute with a 5V Reference



Zero-Shift and Span Adjust: $2V \leq V_{IN} \leq 5V$



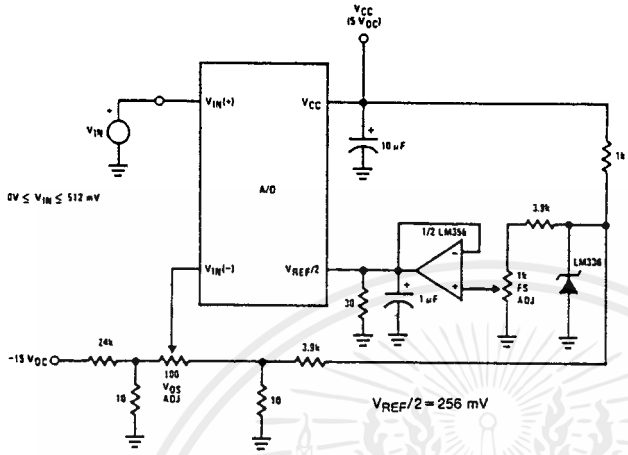
Span Adjust: $0V \leq V_{IN} \leq 3V$



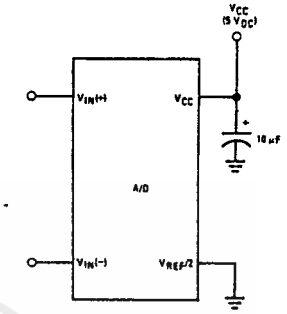
TL/H/5671–5

Typical Applications (Continued)

Directly Converting a Low-Level Signal



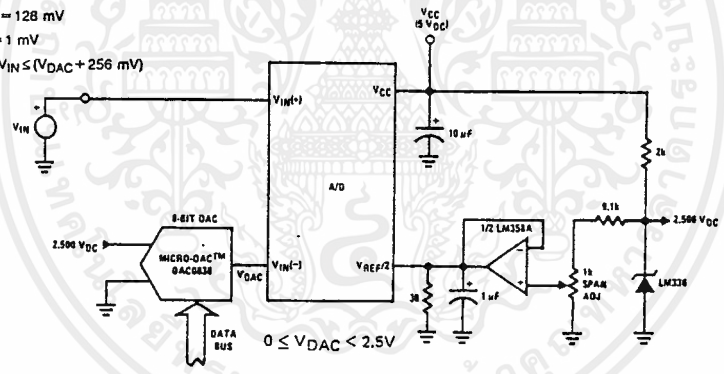
A µP Interfaced Comparator



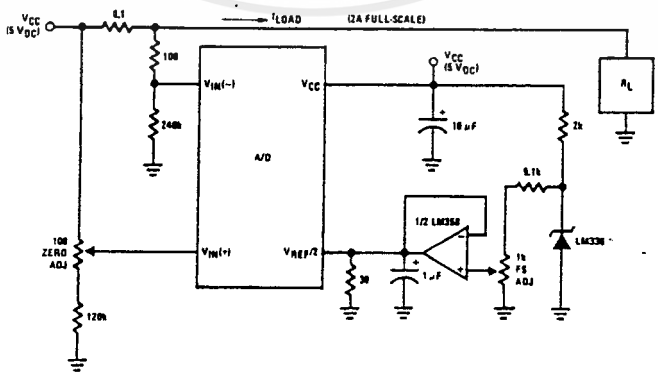
For: $V_{IN}(+) > V_{IN}(-)$
Output = FF_{HEX}
For: $V_{IN}(+) < V_{IN}(-)$
Output = 00_{HEX}

1 mV Resolution with µP Controlled Range

$V_{REF}/2$ = 128 mV
1 LSB = 1 mV
$V_{DAC} \leq V_{IN} \leq (V_{DAC} + 256\ mV)$



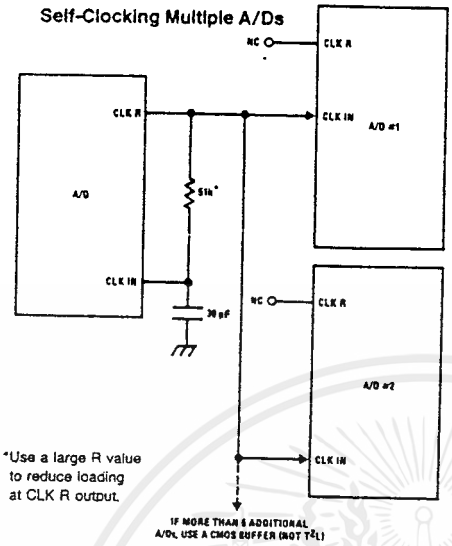
Digitizing a Current Flow



TL/H/5671–6

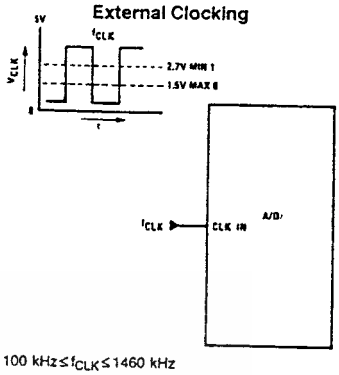
Typical Applications (Continued)

Self-Clocking Multiple A/Ds



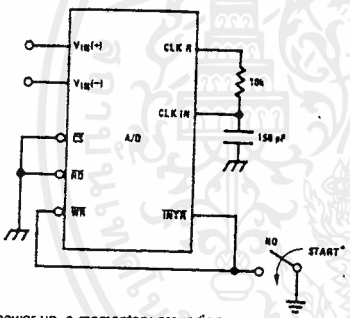
*Use a large R value to reduce loading at CLK R output.

External Clocking



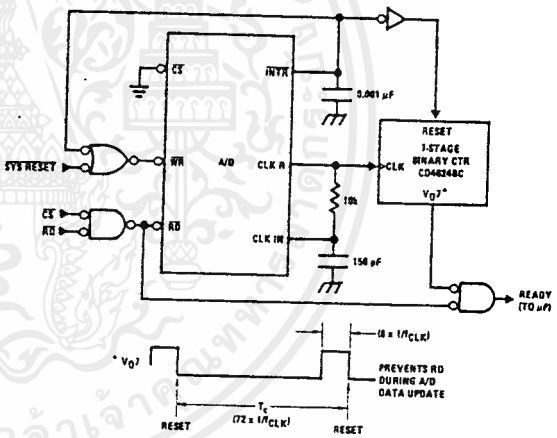
100 kHz ≤ f_{CLK} ≤ 1460 kHz

Self-Clocking in Free-Running Mode

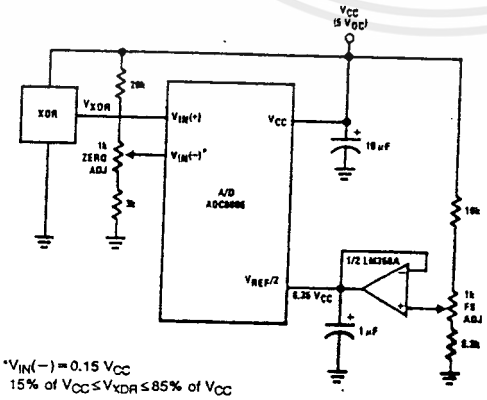


*After power-up, a momentary grounding of the $\overline{WR}$ input is needed to guarantee operation.

μP Interface for Free-Running A/D

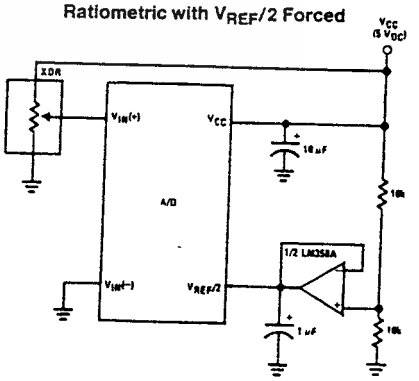


Operating with "Automotive" Ratiometric Transducers



*$V_{IN(-)} = 0.15\ V_{CC}$
15% of $V_{CC} \leq V_{XDR} \leq 85\%$ of V_{CC}

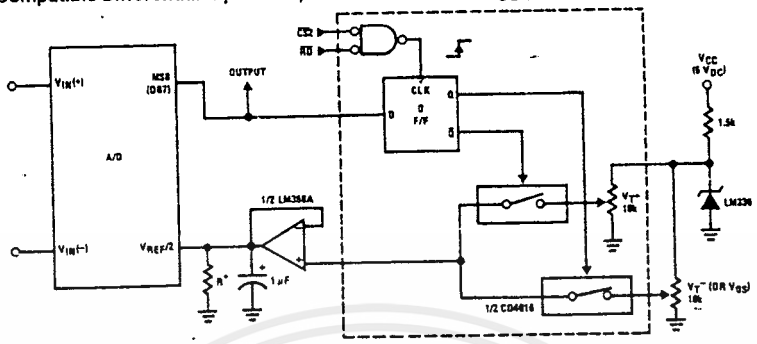
Ratiometric with $V_{REF}/2$ Forced



TL/H/5671–7

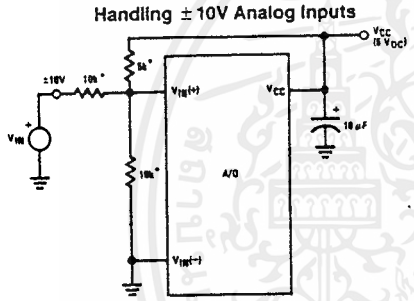
Typical Applications (Continued)

μP Compatible Differential-Input Comparator with Pre-Set V_{OS} (with or without Hysteresis)



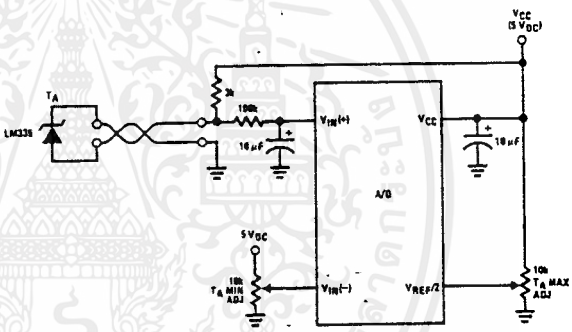
*See *Figure 5* to select R value
D87 = "1" for $V_{IN}(+) > V_{IN}(-) + (V_{REF}/2)$
Omit circuitry within the dotted area if hysteresis is not needed

Handling ±10V Analog Inputs

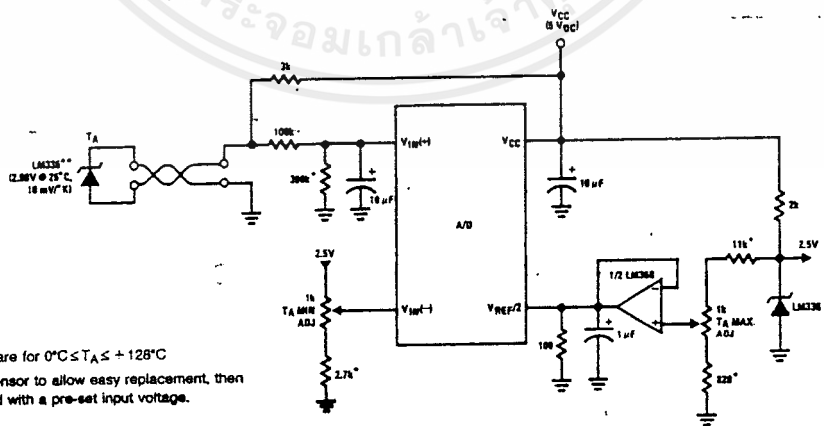


*Beckman Instruments #694-3-R10K resistor array

Low-Cost, μP Interfaced, Temperature-to-Digital Converter



μP Interfaced Temperature-to-Digital Converter



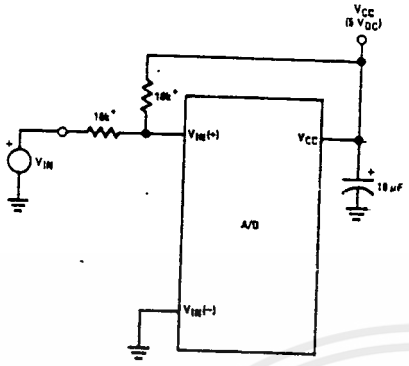
*Circuit values shown are for $0°C \leq T_A \leq +126°C$

**Can calibrate each sensor to allow easy replacement, then A/D can be calibrated with a pre-set input voltage.

TL/H/5671–8

Typical Applications (Continued)

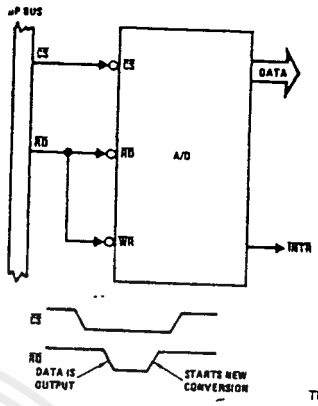
Handling ±5V Analog Inputs



TL/H/5671–33

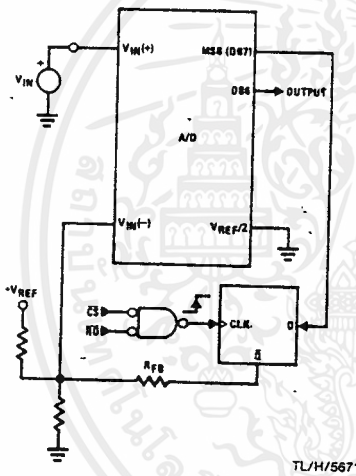
*Beckman Instruments #694-3-R10K resistor array

Read-Only Interface



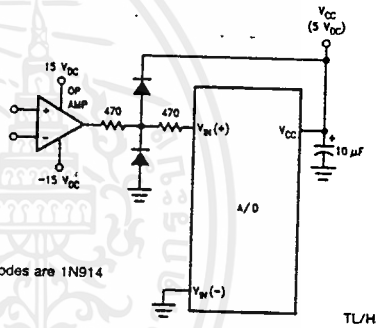
TL/H/5671–34

µP Interfaced Comparator with Hysteresis



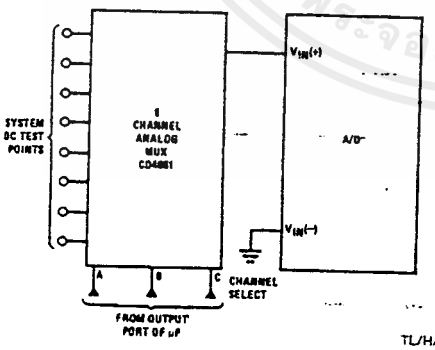
TL/H/5671–35

Protecting the Input



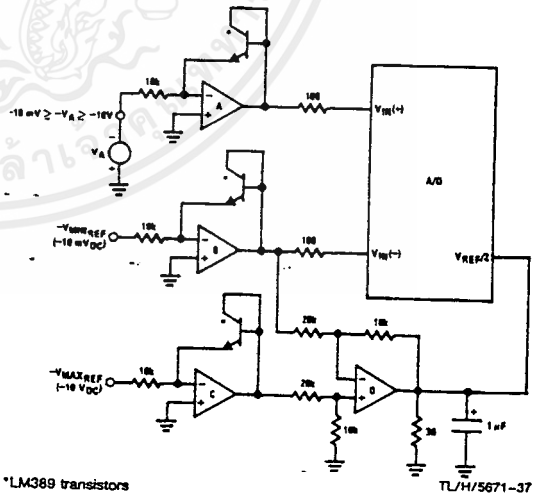
TL/H/5671–9

Analog Self-Test for a System



TL/H/5671–36

A Low-Cost, 3-Decade Logarithmic Converter



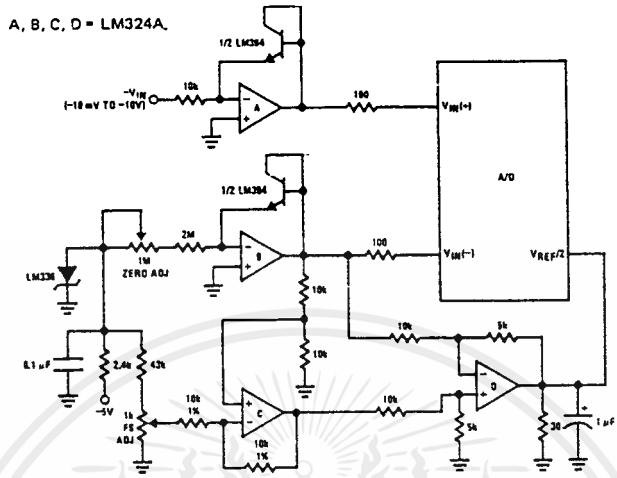
*LM389 transistors

A, B, C, D = LM324A quad op amp

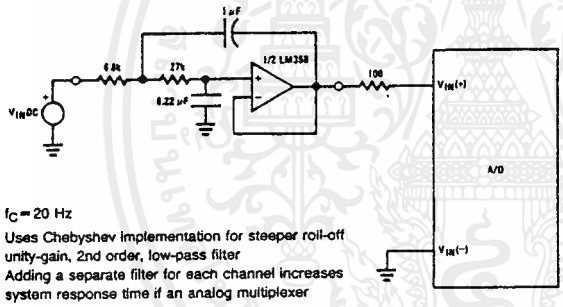
TL/H/5671–37

Typical Applications (Continued)

3-Decade Logarithmic A/D Converter



Noise Filtering the Analog Input

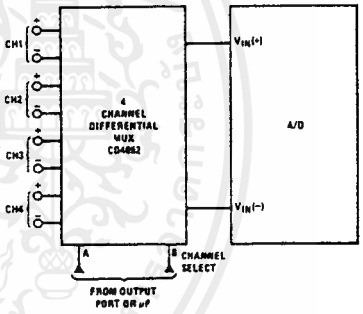


$f_C = 20$ Hz

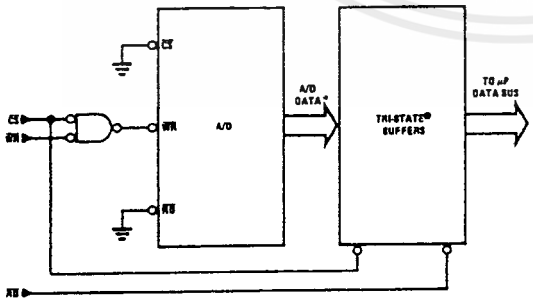
Uses Chebyshev implementation for steeper roll-off unity-gain, 2nd order, low-pass filter

Adding a separate filter for each channel increases system response time if an analog multiplexer is used

Multiplexing Differential Inputs

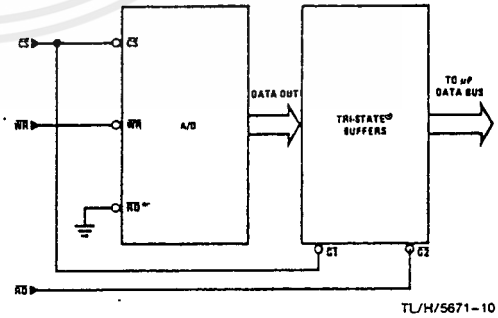


Output Buffers with A/D Data Enabled



*A/D output data is updated 1 CLK period prior to assertion of $\overline{INTR}$

Increasing Bus Drive and/or Reducing Time on Bus

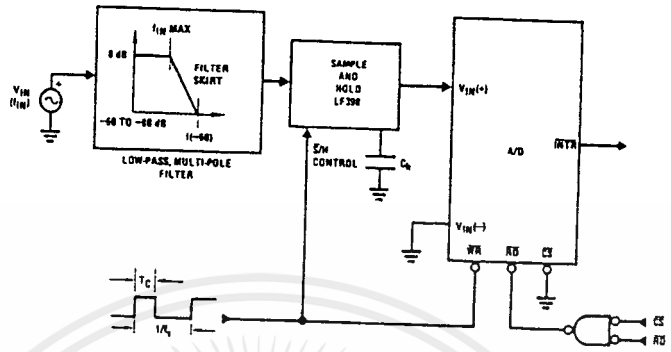


TL/H/5671–10

*Allows output data to set-up at falling edge of $\overline{CS}$

Typical Applications (Continued)

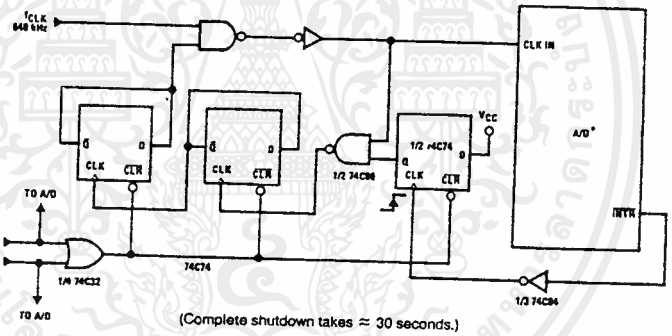
Sampling an AC Input Signal



Note 1: Oversample whenever possible [keep fs > 2f(−60)] to eliminate input frequency folding (aliasing) and to allow for the skirt response of the filter.

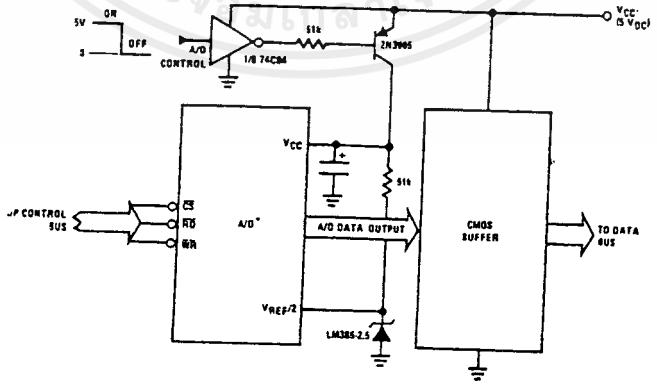
Note 2: Consider the amplitude errors which are introduced within the passband of the filter.

70% Power Savings by Clock Gating



(Complete shutdown takes ≈ 30 seconds.)

Power Savings by A/D and V_{REF} Shutdown



TL/H/5671–11

*Use ADC0801, 02, 03 or 05 for lowest power consumption.

Note: Logic inputs can be driven to V_{CC} with A/D supply at zero volts.

Buffer prevents data bus from overdriving output of A/D when in shutdown mode.

Functional Description

1.0 UNDERSTANDING A/D ERROR SPECS

A perfect A/D transfer characteristic (staircase waveform) is shown in *Figure 1a*. The horizontal scale is analog input voltage and the particular points labeled are in steps of 1 LSB (19.53 mV with 2.5V tied to the $V_{REF}/2$ pin). The digital output codes that correspond to these inputs are shown as D−1, D, and D+1. For the perfect A/D, not only will center-value (A−1, A, A+1, ...) analog inputs produce the correct output ditigal codes, but also each riser (the transitions between adjacent output codes) will be located ± ½ LSB away from each center-value. As shown, the risers are ideal and have no width. Correct digital output codes will be provided for a range of analog input voltages that extend ± ½ LSB from the ideal center-values. Each tread (the range of analog input voltage that provides the same digital output code) is therefore 1 LSB wide.

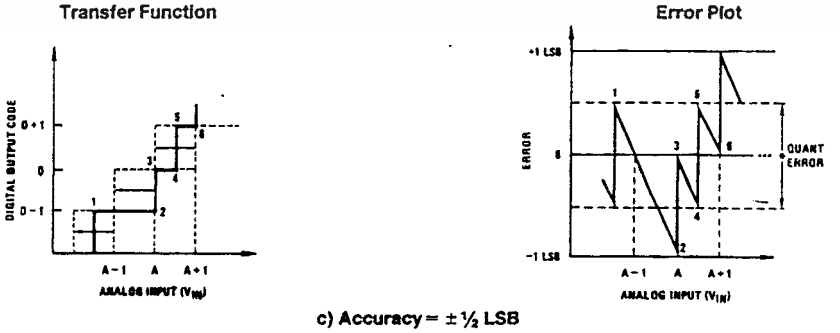
Figure 1b shows a worst case error plot for the ADC0801. All center-valued inputs are guaranteed to produce the correct output codes and the adjacent risers are guaranteed to be no closer to the center-value points than ± ¼ LSB. In other words, if we apply an analog input equal to the center-value ± ¼ LSB, *we guarantee* that the A/D will produce the correct digital code. The maximum range of the position of the code transition is indicated by the horizontal arrow and it is guaranteed to be no more than ½ LSB.

The error curve of *Figure 1c* shows a worst case error plot for the ADC0802. Here we guarantee that if we apply an analog input equal to the LSB analog voltage center-value the A/D will produce the correct digital code.

Next to each transfer function is shown the corresponding error plot. Many people may be more familiar with error plots than transfer functions. The analog input voltage to the A/D is provided by either a linear ramp or by the discrete output steps of a high resolution DAC. Notice that the error is continuously displayed and includes the quantization uncertainty of the A/D. For example the error at point 1 of *Figure 1a* is + ½ LSB because the digital code appeared ½ LSB in advance of the center-value of the tread. The error plots always have a constant negative slope and the abrupt upside steps are always 1 LSB in magnitude.

a) Accuracy = ±0 LSB: A Perfect A/D

b) Accuracy = ± ¼ LSB



c) Accuracy = ± ½ LSB

TL/H/5671–12

FIGURE 1. Clarifying the Error Specs of an A/D Converter

Functional Description (Continued)

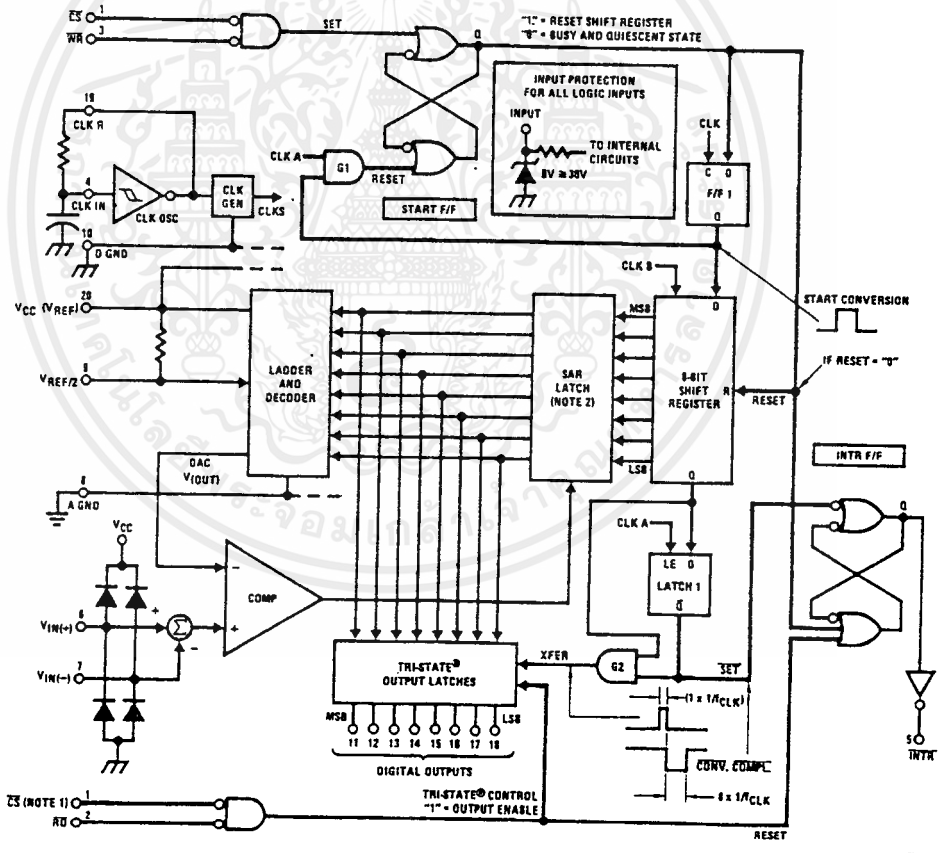
2.0 FUNCTIONAL DESCRIPTION

The ADC0801 series contains a circuit equivalent of the 256R network. Analog switches are sequenced by successive approximation logic to match the analog difference input voltage [$V_{IN}(+) - V_{IN}(-)$] to a corresponding tap on the R network. The most significant bit is tested first and after 8 comparisons (64 clock cycles) a digital 8-bit binary code (1111 1111 = full-scale) is transferred to an output latch and then an interrupt is asserted ($\overline{INTR}$ makes a high-to-low transition). A conversion in process can be interrupted by issuing a second start command. The device may be operated in the free-running mode by connecting $\overline{INTR}$ to the $\overline{WR}$ input with $\overline{CS}=0$. To ensure start-up under all possible conditions, an external $\overline{WR}$ pulse is required during the first power-up cycle.

On the high-to-low transition of the $\overline{WR}$ input the internal SAR latches and the shift register stages are reset. As long as the $\overline{CS}$ input and $\overline{WR}$ input remain low, the A/D will remain in a reset state. *Conversion will start from 1 to 8 clock periods after at least one of these inputs makes a low-to-high transition.*

A functional diagram of the A/D converter is shown in *Figure 2.* All of the package pinouts are shown and the major logic control paths are drawn in heavier weight lines.

The converter is started by having $\overline{CS}$ and $\overline{WR}$ simultaneously low. This sets the start flip-flop (F/F) and the resulting "1" level resets the 8-bit shift register, resets the Interrupt (INTR) F/F and inputs a "1" to the D flop, F/F1, which is at the input end of the 8-bit shift register. Internal clock signals then transfer this "1" to the Q output of F/F1. The AND gate, G1, combines this "1" output with a clock signal to provide a reset signal to the start F/F. If the set signal is no longer present (either $\overline{WR}$ or $\overline{CS}$ is a "1") the start F/F is reset and the 8-bit shift register then can have the "1" clocked in, which starts the conversion process. If the set signal were to still be present, this reset pulse would have no effect (both outputs of the start F/F would momentarily be at a "1" level) and the 8-bit shift register would continue to be held in the reset mode. This logic therefore allows for wide $\overline{CS}$ and $\overline{WR}$ signals and the converter will start after at least one of these signals returns high and the internal clocks again provide a reset signal for the start F/F.



TL/H/5671–13

Note 1: $\overline{CS}$ shown twice for clarity.

Note 2: SAR = Successive Approximation Register.

FIGURE 2. Block Diagram

Functional Description (Continued)

After the "1" is clocked through the 8-bit shift register (which completes the SAR search) it appears as the input to the D-type latch, LATCH 1. As soon as this "1" is output from the shift register, the AND gate, G2, causes the new digital word to transfer to the TRI-STATE output latches. When LATCH 1 is subsequently enabled, the Q output makes a high-to-low transition which causes the INTR F/F to set. An inverting buffer then supplies the $\overline{INTR}$ input signal.

Note that this $\overline{SET}$ control of the INTR F/F remains low for 8 of the external clock periods (as the internal clocks run at 1/8 of the frequency of the external clock). If the data output is continuously enabled ($\overline{CS}$ and $\overline{RD}$ both held low), the $\overline{INTR}$ output will still signal the end of conversion (by a high-to-low transition), because the $\overline{SET}$ input can control the Q output of the INTR F/F even though the RESET input is constantly at a "1" level in this operating mode. This $\overline{INTR}$ output will therefore stay low for the duration of the $\overline{SET}$ signal, which is 8 periods of the external clock frequency (assuming the A/D is not started during this interval).

When operating in the free-running or continuous conversion mode ($\overline{INTR}$ pin tied to $\overline{WR}$ and $\overline{CS}$ wired low—see also section 2.8), the START F/F is SET by the high-to-low transition of the $\overline{INTR}$ signal. This resets the SHIFT REGISTER which causes the input to the D-type latch, LATCH 1, to go low. As the latch enable input is still present, the $\overline{Q}$ output will go high, which then allows the INTR F/F to be RESET. This reduces the width of the resulting $\overline{INTR}$ output pulse to only a few propagation delays (approximately 300 ns).

When data is to be read, the combination of both $\overline{CS}$ and $\overline{RD}$ being low will cause the INTR F/F to be reset and the TRI-STATE output latches will be enabled to provide the 8-bit digital outputs.

2.1 Digital Control Inputs

The digital control inputs ($\overline{CS}$, $\overline{RD}$, and $\overline{WR}$) meet standard T²L logic voltage levels. These signals have been renamed when compared to the standard A/D Start and Output Enable labels. In addition, these inputs are active low to allow an easy interface to microprocessor control busses. For non-microprocessor based applications, the $\overline{CS}$ input (pin 1) can be grounded and the standard A/D Start function is obtained by an active low pulse applied at the $\overline{WR}$ input (pin 3) and the Output Enable function is caused by an active low pulse at the $\overline{RD}$ input (pin 2).

2.2 Analog Differential Voltage Inputs and Common-Mode Rejection

This A/D has additional applications flexibility due to the analog differential voltage input. The $V_{IN}(-)$ input (pin 7) can be used to automatically subtract a fixed voltage value from the input reading (tare correction). This is also useful in 4 mA–20 mA current loop conversion. In addition, common-mode noise can be reduced by use of the differential input.

The time interval between sampling $V_{IN}(+)$ and $V_{IN}(-)$ is 4-½ clock periods. The maximum error voltage due to this slight time difference between the input voltage samples is given by:

$$\Delta V_e(MAX) = (V_P)(2\pi f_{cm})\left(\frac{4.5}{f_{CLK}}\right),$$

where:

ΔV_e is the error voltage due to sampling delay

V_P is the peak value of the common-mode voltage

f_{cm} is the common-mode frequency

As an example, to keep this error to ¼ LSB (~5 mV) when operating with a 60 Hz common-mode frequency, f_{cm}, and using a 640 kHz A/D clock, f_{CLK}, would allow a peak value of the common-mode voltage, V_P, which is given by:

$$V_P = \frac{[\Delta V_{e(MAX)}(f_{CLK})]}{(2\pi f_{cm})(4.5)}$$

or

$$V_P = \frac{(5 \times 10^{-3})(640 \times 10^3)}{(6.28)(60)(4.5)}$$

which gives

$V_P \cong 1.9V$.

The allowed range of analog input voltages usually places more severe restrictions on input common-mode noise levels.

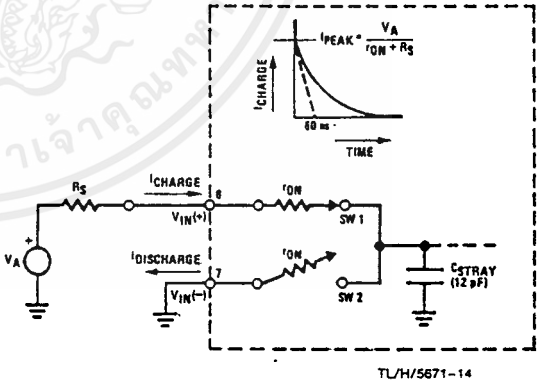
An analog input voltage with a reduced span and a relatively large zero offset can be handled easily by making use of the differential input (see section 2.4 Reference Voltage).

2.3 Analog Inputs

2.3.1 Input Current

Normal Mode

Due to the internal switching action, displacement currents will flow at the analog inputs. This is due to on-chip stray capacitance to ground as shown in *Figure 3*.



r_{ON} of SW 1 and SW 2 ≅ 5 kΩ

$\tau = r_{ON} C_{STRAY} \cong 5\ k\Omega \times 12\ pF = 60\ ns$

FIGURE 3. Analog Input Impedance

Functional Description (Continued)

The voltage on this capacitance is switched and will result in currents entering the $V_{IN}(+)$ input pin and leaving the $V_{IN}(-)$ input which will depend on the analog differential input voltage levels. These current transients occur at the leading edge of the internal clocks. They rapidly decay and *do not cause errors* as the on-chip comparator is strobed at the end of the clock period.

Fault Mode

If the voltage source applied to the $V_{IN}(+)$ or $V_{IN}(-)$ pin exceeds the allowed operating range of $V_{CC}+50$ mV, large input currents can flow through a parasitic diode to the V_{CC} pin. If these currents can exceed the 1 mA max allowed spec, an external diode (1N914) should be added to bypass this current to the V_{CC} pin (with the current bypassed with this diode, the voltage at the $V_{IN}(+)$ pin can exceed the V_{CC} voltage by the forward voltage of this diode).

2.3.2 Input Bypass Capacitors

Bypass capacitors at the inputs will average these charges and cause a DC current to flow through the output resistances of the analog signal sources. This charge pumping action is worse for continuous conversions with the $V_{IN}(+)$ input voltage at full-scale. For continuous conversions with a 640 kHz clock frequency with the $V_{IN}(+)$ input at 5V, this DC current is at a maximum of approximately 5 μA. Therefore, *bypass capacitors should not be used at the analog inputs or the $V_{REF}/2$ pin* for high resistance sources (> 1 kΩ). If input bypass capacitors are necessary for noise filtering and high source resistance is desirable to minimize capacitor size, the detrimental effects of the voltage drop across this input resistance, which is due to the average value of the input current, can be eliminated with a full-scale adjustment while the given source resistor and input bypass capacitor are both in place. This is possible because the average value of the input current is a precise linear function of the differential input voltage.

2.3.3 Input Source Resistance

Large values of source resistance where an input bypass capacitor is not used, *will not cause errors* as the input currents settle out prior to the comparison time. If a low pass filter is required in the system, use a low valued series resistor (≤ 1 kΩ) for a passive RC section or add an op amp RC active low pass filter. For low source resistance applications, (≤ 1 kΩ), a 0.1 μF bypass capacitor at the inputs will prevent noise pickup due to series lead inductance of a long wire. A 100Ω series resistor can be used to isolate this capacitor—both the R and C are placed outside the feedback loop—from the output of an op amp, if used.

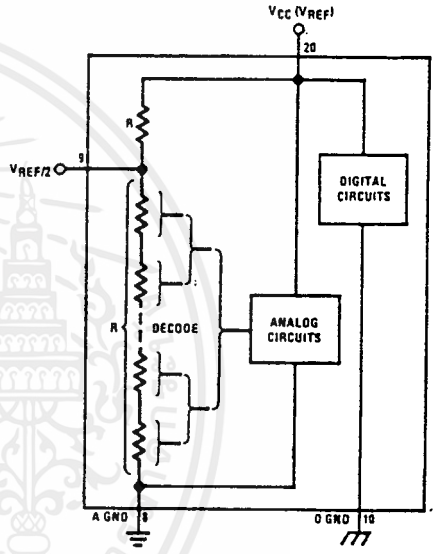
2.3.4 Noise

The leads to the analog inputs (pin 6 and 7) should be kept as short as possible to minimize input noise coupling. Both noise and undesired digital clock coupling to these inputs can cause system errors. The source resistance for these inputs should, in general, be kept below 5 kΩ. Larger values of source resistance can cause undesired system noise pickup. Input bypass capacitors, placed from the analog inputs to ground, will eliminate system noise pickup but can create analog scale errors as these capacitors will average the transient input switching currents of the A/D (see section 2.3.1.). This scale error depends on both a large source resistance and the use of an input bypass capacitor. This error can be eliminated by doing a full-scale adjustment of the A/D (adjust $V_{REF}/2$ for a proper full-scale reading—see section 2.5.2 on Full-Scale Adjustment) with the source resistance and input bypass capacitor in place.

2.4 Reference Voltage

2.4.1 Span Adjust

For maximum applications flexibility, these A/Ds have been designed to accommodate a 5 V_{DC}, 2.5 V_{DC} or an adjusted voltage reference. This has been achieved in the design of the IC as shown in *Figure 4*.



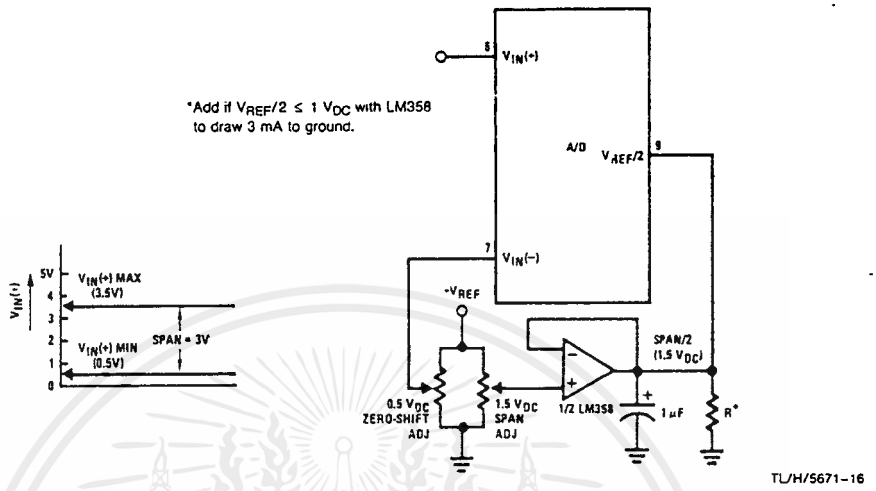
TL/H/5671–15

FIGURE 4. The $V_{REFERENCE}$ Design on the IC

Notice that the reference voltage for the IC is either ½ of the voltage applied to the V_{CC} supply pin, or is equal to the voltage that is externally forced at the $V_{REF}/2$ pin. This allows for a ratiometric voltage reference using the V_{CC} supply, a 5 V_{DC} reference voltage can be used for the V_{CC} supply or a voltage less than 2.5 V_{DC} can be applied to the $V_{REF}/2$ input for increased application flexibility. The internal gain to the $V_{REF}/2$ input is 2, making the full-scale differential input voltage twice the voltage at pin 9.

An example of the use of an adjusted reference voltage is to accommodate a reduced span—or dynamic voltage range of the analog input voltage. If the analog input voltage were to range from 0.5 V_{DC} to 3.5 V_{DC}, instead of 0V to 5 V_{DC}, the span would be 3V as shown in *Figure 5*. With 0.5 V_{DC} applied to the $V_{IN}(-)$ pin to absorb the offset, the reference voltage can be made equal to ½ of the 3V span or 1.5 V_{DC}. The A/D now will encode the $V_{IN}(+)$ signal from 0.5V to 3.5 V with the 0.5V input corresponding to zero and the 3.5 V_{DC} input corresponding to full-scale. The full 8 bits of resolution are therefore applied over this reduced analog input voltage range.

Functional Description (Continued)



a) Analog Input Signal Example

b) Accommodating an Analog Input from 0.5V (Digital Out = = 00_{HEX}) to 3.5V (Digital Out = FF_{HEX})

FIGURE 5. Adapting the A/D Analog Input Voltages to Match an Arbitrary Input Signal Range

2.4.2 Reference Accuracy Requirements

The converter can be operated in a ratiometric mode or an absolute mode. In ratiometric converter applications, the magnitude of the reference voltage is a factor in both the output of the source transducer and the output of the A/D converter and therefore cancels out in the final digital output code. The ADC0805 is specified particularly for use in ratiometric applications with no adjustments required. In absolute conversion applications, both the initial value and the temperature stability of the reference voltage are important factors in the accuracy of the A/D converter. For $V_{REF}/2$ voltages of 2.4 V_{DC} nominal value, initial errors of ±10 mV_{DC} will cause conversion errors of ±1 LSB due to the gain of 2 of the $V_{REF}/2$ input. In reduced span applications, the initial value and the stability of the $V_{REF}/2$ input voltage become even more important. For example, if the span is reduced to 2.5V, the analog input LSB voltage value is correspondingly reduced from 20 mV (5V span) to 10 mV and 1 LSB at the $V_{REF}/2$ input becomes 5 mV. As can be seen, this reduces the allowed initial tolerance of the reference voltage and requires correspondingly less absolute change with temperature variations. Note that spans smaller than 2.5V place even tighter requirements on the initial accuracy and stability of the reference source.

In general, the magnitude of the reference voltage will require an initial adjustment. Errors due to an improper value of reference voltage appear as full-scale errors in the A/D transfer function. IC voltage regulators may be used for references if the ambient temperature changes are not excessive. The LM336B 2.5V IC reference diode (from National Semiconductor) has a temperature stability of 1.8 mV typ (6 mV max) over $0°C \leq T_A \leq +70°C$. Other temperature range parts are also available.

2.5 Errors and Reference Voltage Adjustments

2.5.1 Zero Error

The zero of the A/D does not require adjustment. If the minimum analog input voltage value, $V_{IN(MIN)}$, is not ground, a zero offset can be done. The converter can be made to output 0000 0000 digital code for this minimum input voltage by biasing the A/D $V_{IN}(-)$ input at this $V_{IN(MIN)}$ value (see Applications section). This utilizes the differential mode operation of the A/D.

The zero error of the A/D converter relates to the location of the first riser of the transfer function and can be measured by grounding the V_{IN} (−) input and applying a small magnitude positive voltage to the V_{IN} (+) input. Zero error is the difference between the actual DC input voltage that is necessary to just cause an output digital code transition from 0000 0000 to 0000 0001 and the ideal ½ LSB value (½ LSB = 9.8 mV for $V_{REF}/2 = 2.500\ V_{DC}$).

2.5.2 Full-Scale

The full-scale adjustment can be made by applying a differential input voltage that is 1½ LSB less than the desired analog full-scale voltage range and then adjusting the magnitude of the $V_{REF}/2$ input (pin 9 or the V_{CC} supply if pin 9 is not used) for a digital output code that is just changing from 1111 1110 to 1111 1111.

Functional Description (Continued)

2.5.3 Adjusting for an Arbitrary Analog Input Voltage Range

If the analog zero voltage of the A/D is shifted away from ground (for example, to accommodate an analog input signal that does not go to ground) this new zero reference should be properly adjusted first. A $V_{IN}(+)$ voltage that equals this desired zero reference plus ½ LSB (where the LSB is calculated for the desired analog span, 1 LSB = analog span/256) is applied to pin 6 and the zero reference voltage at pin 7 should then be adjusted to just obtain the 00_{HEX} to 01_{HEX} code transition.

The full-scale adjustment should then be made (with the proper $V_{IN}(-)$ voltage applied) by forcing a voltage to the $V_{IN}(+)$ input which is given by:

$$V_{IN}(+) \text{ fs adj} = V_{MAX} - 1.5\left[\frac{(V_{MAX} - V_{MIN})}{256}\right],$$

where:

V_{MAX} = The high end of the analog input range

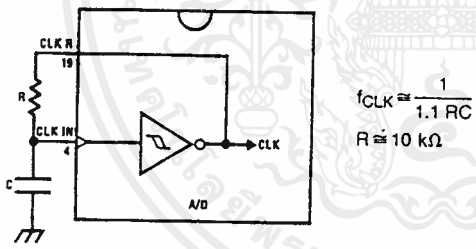
and

V_{MIN} = the low end (the offset zero) of the analog range. (Both are ground referenced.)

The $V_{REF}/2$ (or V_{CC}) voltage is then adjusted to provide a code change from FE_{HEX} to FF_{HEX}. This completes the adjustment procedure.

2.6 Clocking Option

The clock for the A/D can be derived from the CPU clock or an external RC can be added to provide self-clocking. The CLK IN (pin 4) makes use of a Schmitt trigger as shown in *Figure 6*.



TL/H/5671–17

FIGURE 6. Self-Clocking the A/D

Heavy capacitive or DC loading of the clock R pin should be avoided as this will disturb normal converter operation. Loads less than 50 pF, such as driving up to 7 A/D converter clock inputs from a single clock R pin of 1 converter, are allowed. For larger clock line loading, a CMOS or low power TTL buffer or PNP input logic should be used to minimize the loading on the clock R pin (do not use a standard TTL buffer).

2.7 Restart During a Conversion

If the A/D is restarted ($\overline{CS}$ and $\overline{WR}$ go low and return high) during a conversion, the converter is reset and a new conversion is started. The output data latch is not updated if the conversion in process is not allowed to be completed, therefore the data of the previous conversion remains in this latch. The $\overline{INTR}$ output simply remains at the "1" level.

2.8 Continuous Conversions

For operation in the free-running mode an initializing pulse should be used, following power-up, to ensure circuit operation. In this application, the $\overline{CS}$ input is grounded and the $\overline{WR}$ input is tied to the $\overline{INTR}$ output. This $\overline{WR}$ and $\overline{INTR}$ node should be momentarily forced to logic low following a power-up cycle to guarantee operation.

2.9 Driving the Data Bus

This MOS A/D, like MOS microprocessors and memories, will require a bus driver when the total capacitance of the data bus gets large. Other circuitry, which is tied to the data bus, will add to the total capacitive loading, even in TRI-STATE (high impedance mode). Backplane bussing also greatly adds to the stray capacitance of the data bus.

There are some alternatives available to the designer to handle this problem. Basically, the capacitive loading of the data bus slows down the response time, even though DC specifications are still met. For systems operating with a relatively slow CPU clock frequency, more time is available in which to establish proper logic levels on the bus and therefore higher capacitive loads can be driven (see typical characteristics curves).

At higher CPU clock frequencies time can be extended for I/O reads (and/or writes) by inserting wait states (8080) or using clock extending circuits (6800).

Finally, if time is short and capacitive loading is high, external bus drivers must be used. These can be TRI-STATE buffers (low power Schottky such as the DM74LS240 series is recommended) or special higher drive current products which are designed as bus drivers. High current bipolar bus drivers with PNP inputs are recommended.

2.10 Power Supplies

Noise spikes on the V_{CC} supply line can cause conversion errors as the comparator will respond to this noise. A low inductance tantalum filter capacitor should be used close to the converter V_{CC} pin and values of 1 μF or greater are recommended. If an unregulated voltage is available in the system, a separate LM340LAZ-5.0, TO-92, 5V voltage regulator for the converter (and other analog circuitry) will greatly reduce digital noise on the V_{CC} supply.

2.11 Wiring and Hook-Up Precautions

Standard digital wire wrap sockets are not satisfactory for breadboarding this A/D converter. Sockets on PC boards can be used and all logic signal wires and leads should be grouped and kept as far away as possible from the analog signal leads. Exposed leads to the analog inputs can cause undesired digital noise and hum pickup, therefore shielded leads may be necessary in many applications.

Functional Description (Continued)

A single point analog ground that is separate from the logic ground points should be used. The power supply bypass capacitor and the self-clocking capacitor (if used) should both be returned to digital ground. Any $V_{REF}/2$ bypass capacitors, analog input filter capacitors, or input signal shielding should be returned to the analog ground point. A test for proper grounding is to measure the zero error of the A/D converter. Zero errors in excess of ¼ LSB can usually be traced to improper board layout and wiring (see section 2.5.1 for measuring the zero error).

3.0 TESTING THE A/D CONVERTER

There are many degrees of complexity associated with testing an A/D converter. One of the simplest tests is to apply a known analog input voltage to the converter and use LEDs to display the resulting digital output code as shown in *Figure 7*.

For ease of testing, the $V_{REF}/2$ (pin 9) should be supplied with 2.560 V_{DC} and a V_{CC} supply voltage of 5.12 V_{DC} should be used. This provides an LSB value of 20 mV.

If a full-scale adjustment is to be made, an analog input voltage of 5.090 V_{DC} (5.120−1½ LSB) should be applied to the $V_{IN}(+)$ pin with the $V_{IN}(-)$ pin grounded. The value of the $V_{REF}/2$ input voltage should then be adjusted until the digital output code is just changing from 1111 1110 to 1111 1111. This value of $V_{REF}/2$ should then be used for all the tests.

The digital output LED display can be decoded by dividing the 8 bits into 2 hex characters, the 4 most significant (MS) and the 4 least significant (LS). Table I shows the fractional binary equivalent of these two 4-bit groups. By adding the voltages obtained from the "VMS" and "VLS" columns in Table I, the nominal value of the digital display (when $V_{REF}/2$ = 2.560V) can be determined. For example, for an output LED display of 1011 0110 or B6 (in hex), the voltage values from the table are 3.520 + 0.120 or 3.640 V_{DC}. These voltage values represent the center-values of a perfect A/D converter. The effects of quantization error have to be accounted for in the interpretation of the test results.

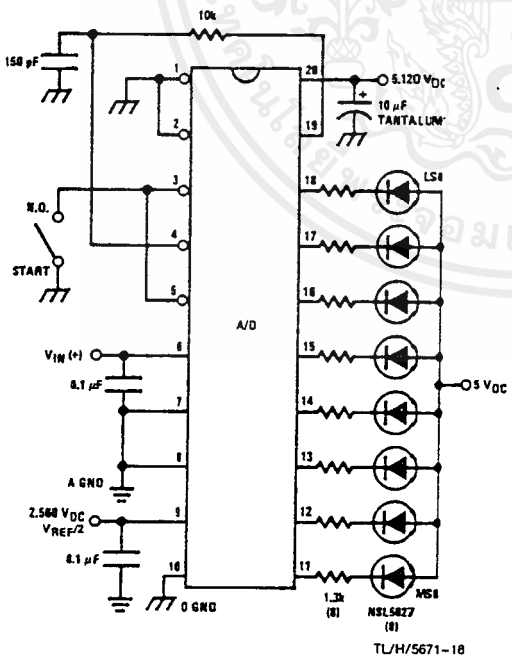


FIGURE 7. Basic A/D Tester

For a higher speed test system, or to obtain plotted data, a digital-to-analog converter is needed for the test set-up. An accurate 10-bit DAC can serve as the precision voltage source for the A/D. Errors of the A/D under test can be expressed as either analog voltages or differences in 2 digital words.

A basic A/D tester that uses a DAC and provides the error as an analog output voltage is shown in *Figure 8*. The 2 op amps can be eliminated if a lab DVM with a numerical subtraction feature is available to read the difference voltage, "A−C", directly. The analog input voltage can be supplied by a low frequency ramp generator and an X-Y plotter can be used to provide analog error (Y axis) versus analog input (X axis).

For operation with a microprocessor or a computer-based test system, it is more convenient to present the errors digitally. This can be done with the circuit of *Figure 9*, where the output code transitions can be detected as the 10-bit DAC is incremented. This provides ¼ LSB steps for the 8-bit A/D under test. If the results of this test are automatically plotted with the analog input on the X axis and the error (in LSB's) as the Y axis, a useful transfer function of the A/D under test results. For acceptance testing, the plot is not necessary and the testing speed can be increased by establishing internal limits on the allowed error for each code.

4.0 MICROPROCESSOR INTERFACING

To dicuss the interface with 8080A and 6800 microprocessors, a common sample subroutine structure is used. The microprocessor starts the A/D, reads and stores the results of 16 successive conversions, then returns to the user's program. The 16 data bytes are stored in 16 successive memory locations. All Data and Addresses will be given in hexadecimal form. Software and hardware details are provided separately for each type of microprocessor.

4.1 Interfacing 8080 Microprocessor Derivatives (8048, 8085)

This converter has been designed to directly interface with derivatives of the 8080 microprocessor. The A/D can be mapped into memory space (using standard memory address decoding for $\overline{CS}$ and the $\overline{MEMR}$ and $\overline{MEMW}$ strobes) or it can be controlled as an I/O device by using the $\overline{I/O\ R}$ and $\overline{I/O\ W}$ strobes and decoding the address bits A0 → A7 (or address bits A8 → A15 as they will contain the same 8-bit address information) to obtain the $\overline{CS}$ input. Using the I/O space provides 256 additional addresses and may allow a simpler 8-bit address decoder but the data can only be input to the accumulator. To make use of the additional memory reference instructions, the A/D should be mapped into memory space. An example of an A/D in I/O space is shown in *Figure 10*.

Functional Description (Continued)

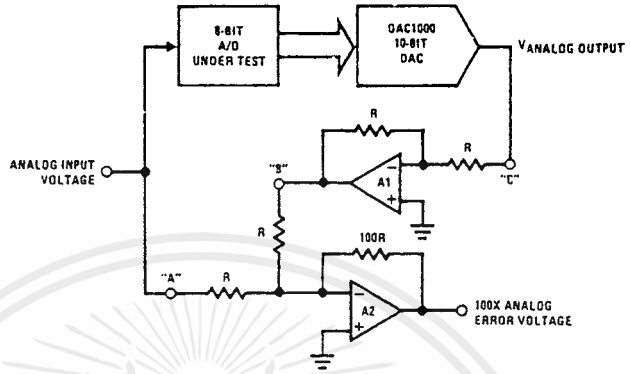
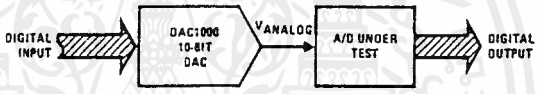


FIGURE 8. A/D Tester with Analog Error Output



TL/H/5671–19

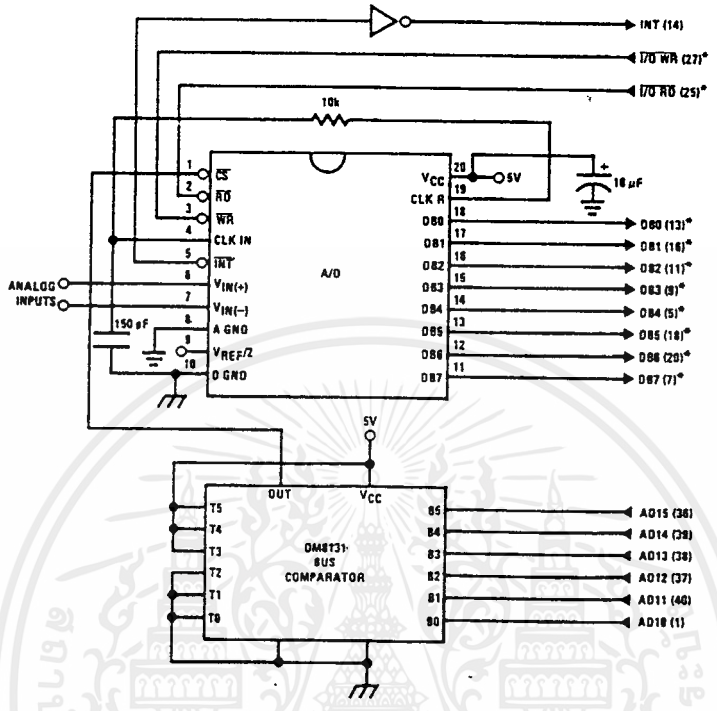
FIGURE 9. Basic "Digital" A/D Tester

TABLE I. DECODING THE DIGITAL OUTPUT LEDs

HEX	BINARY	FRACTIONAL BINARY VALUE FOR		OUTPUT VOLTAGE CENTER VALUES WITH $V_{REF}/2 = 2.560\ V_{DC}$	
		MS GROUP	LS GROUP	VMS GROUP*	VLS GROUP
F	1 1 1 1	15/16	15/256	4.800	0.300
E	1 1 1 0	7/8	7/128	4.480	0.280
D	1 1 0 1	13/16	13/256	4.160	0.260
C	1 1 0 0	3/4	3/64	3.840	0.240
B	1 0 1 1	11/16	11/256	3.520	0.220
A	1 0 1 0	5/8	5/128	3.200	0.200
9	1 0 0 1	9/16	9/256	2/880	0.180
8	1 0 0 0	1/2	1/32	2/560	0.160
7	0 1 1 1	7/16	7/256	2.240	0.140
6	0 1 1 0	3/8	3/128	1.920	0.120
5	0 1 0 1	5/16	2/256	1.600	0.100
4	0 1 0 0	1/4	1/64	1/280	0.080
3	0 0 1 1	3/16	3/256	0.960	0.060
2	0 0 1 0	1/8	1/128	0.640	0.040
1	0 0 0 1	1/16	1/256	0.320	0.020
0	0 0 0 0			0	0

*Display Output = VMS Group + VLS Group

Functional Description (Continued)



TL/H/5671–20

Note 1: *Pin numbers for the DP8228 system controller, others are INS8080A.

Note 2: Pin 23 of the INS8228 must be tied to +12V through a 1 kΩ resistor to generate the RST 7 instruction when an interrupt is acknowledged as required by the accompanying sample program.

FIGURE 10. ADC0801–INS8080A CPU Interface

SAMPLE PROGRAM FOR *FIGURE 10* ADC0801–INS8080A CPU INTERFACE

```
0038    C3 00 03    RST 7:          JMP     LD DATA
  •       •           •               •
  •       •           •               •
0100    21 00 02    START:          LXI H 0200H         ; HL pair will point to
                                                        ; data storage locations
0103    31 00 04    RETURN:         LXI SP 0400H        ; Initialize stack pointer (Note 1)
0106    7D                          MOV A, L            ; Test # of bytes entered
0107    FE 0F                       CPI 0F H            ; If # = 16, JMP to
0109    CA 13 01                    JZ CONT             ; user program
010C    D3 E0                       OUT E0 H            ; Start A/D
010E    FB                          EI                  ; Enable interrupt
010F    00          LOOP:           NOP                 ; Loop until end of
0110    C3 0F 01                    JMP LOOP            ; conversion
0113      •         CONT:             •
  •       •           •               •
  •       •         (User program to  •
  •       •         process data)     •
  •       •           •               •
  •       •           •               •
0300    DB E0       LD DATA:        IN E0 H             ; Load data into accumulator
0302    77                          MOV M, A            ; Store data
0303    23                          INX H               ; Increment storage pointer
0304    C3 03 01                    JMP RETURN
```

Note 1: The stack pointer must be dimensioned because a RST 7 instruction pushes the PC onto the stack.

Note 2: All address used were arbitrarily chosen.

Functional Description (Continued)

The standard control bus signals of the 8080 $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$) can be directly wired to the digital control inputs of the A/D and the bus timing requirements are met to allow both starting the converter and outputting the data onto the data bus. A bus driver should be used for larger microprocessor systems where the data bus leaves the PC board and/or must drive capacitive loads larger than 100 pF.

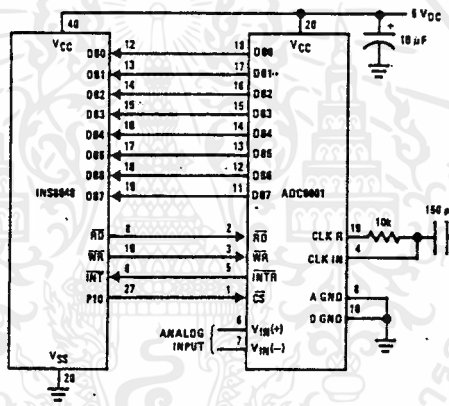
4.1.1 Sample 8080A CPU Interfacing Circuitry and Program

The following sample program and associated hardware shown in *Figure 10* may be used to input data from the converter to the INS8080A CPU chip set (comprised of the INS8080A microprocessor, the INS8228 system controller and the INS8224 clock generator). For simplicity, the A/D is controlled as an I/O device, specifically an 8-bit bi-directional port located at an arbitrarily chosen port address, E0. The TRI-STATE output capability of the A/D eliminates the need for a peripheral interface device, however address decoding is still required to generate the appropriate $\overline{CS}$ for the converter.

It is important to note that in systems where the A/D converter is 1-of-8 or less I/O mapped devices, no address decoding circuitry is necessary. Each of the 8 address bits (A0 to A7) can be directly used as $\overline{CS}$ inputs—one for each I/O device.

4.1.2 INS8048 Interface

The INS8048 interface technique with the ADC0801 series (see *Figure 11*) is simpler than the 8080A CPU interface. There are 24 I/O lines and three test input lines in the 8048. With these extra I/O lines available, one of the I/O lines (bit 0 of port 1) is used as the chip select signal to the A/D, thus eliminating the use of an external address decoder. Bus control signals $\overline{RD}$, $\overline{WR}$ and $\overline{INT}$ of the 8048 are tied directly to the A/D. The 16 converted data words are stored at on-chip RAM locations from 20 to 2F (Hex). The $\overline{RD}$ and $\overline{WR}$ signals are generated by reading from and writing into a dummy address, respectively. A sample interface program is shown below.



TL/H/5671–21

FIGURE 11. INS8048 Interface

SAMPLE PROGRAM FOR *FIGURE 11* INS8048 INTERFACE

```
04 10                   JMP     10H             ;Program starts at addr 10
                        ORG     3H
04 50                   JMP     50H             ;Interrupt jump vector
                        ORG     10H             ;Main program
99 FE                   ANL     P1, #0FEH       ;Chip select
81                      MOVX    A, @R1          ;Read in the 1st data
                                                ;to reset the intr
89 01           START:  ORL     P1, #1          ;Set port pin high
B8 20                   MOV     R0, #20H        ;Data address
B9 FF                   MOV     R1, #0FFH       ;Dummy address
BA 10                   MOV     R2, #10H        ;Counter for 16 bytes
23 FF           AGAIN:  MOV     A, #0FFH        ;Set ACC for intr loop
99 FE                   ANL     P1, #0FEH       ;Send CS (bit 0 of P1)
91                      MOVX    @R1, A          ;Send WR out
05                      EN      I               ;Enable interrupt
96 21           LOOP:   JNZ     LOOP            ;Wait for interrupt
EA 1B                   DJNZ    R2, AGAIN       ;If 16 bytes are read
00                      NOP                     ;go to user's program
00                      NOP
                        ORG     50H
81              INDATA: MOVX    A, @R1          ;Input data, CS still low
A0                      MOV     @R0, A          ;Store in memory
18                      INC     R0              ;Increment storage counter
89 01                   ORL     P1, #1          ;Reset CS signal
27                      CLR     A               ;Clear ACC to get out of
93                      RETR                    ;the interrupt loop
```

Functional Description (Continued)

4.2 Interfacing the Z-80

The Z-80 control bus is slightly different from that of the 8080. General $\overline{RD}$ and $\overline{WR}$ strobes are provided and separate memory request, $\overline{MREQ}$, and I/O request, $\overline{IORQ}$, signals are used which have to be combined with the generalized strobes to provide the equivalent 8080 signals. An advantage of operating the A/D in I/O space with the Z-80 is that the CPU will automatically insert one wait state (the $\overline{RD}$ and $\overline{WR}$ strobes are extended one clock period) to allow more time for the I/O devices to respond. Logic to map the A/D in I/O space is shown in *Figure 13*.

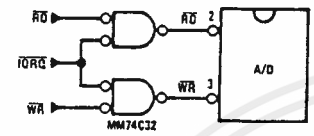


FIGURE 13. Mapping the A/D as an I/O Device for Use with the Z-80 CPU

Additional I/O advantages exist as software DMA routines are available and use can be made of the output data transfer which exists on the upper 8 address lines (A8 to A15) during I/O input instructions. For example, MUX channel selection for the A/D can be accomplished with this operating mode.

4.3 Interfacing 6800 Microprocessor Derivatives (6502, etc.)

The control bus for the 6800 microprocessor derivatives does not use the $\overline{RD}$ and $\overline{WR}$ strobe signals. Instead it employs a single R/$\overline{W}$ line and additional timing, if needed, can be derived fom the $\phi 2$ clock. All I/O devices are memory mapped in the 6800 system, and a special signal, VMA, indicates that the current address is valid. *Figure 14* shows an interface schematic where the A/D is memory mapped in the 6800 system. For simplicity, the $\overline{CS}$ decoding is shown using ½ DM8092. Note that in many 6800 systems, an already decoded $\overline{475}$ line is brought out to the common bus at pin 21. This can be tied directly to the $\overline{CS}$ pin of the A/D, provided that no other devices are addressed at HX ADDR: 4XXX or 5XXX.

The following subroutine performs essentially the same function as in the case of the 8080A interface and it can be called from anywhere in the user's program.

In *Figure 15* the ADC0801 series is interfaced to the M6800 microprocessor through (the arbitrarily chosen) Port B of the MC6820 or MC6821 Peripheral Interface Adapter, (PIA). Here the $\overline{CS}$ pin of the A/D is grounded since the PIA is already memory mapped in the M6800 system and no $\overline{CS}$ decoding is necessary. Also notice that the A/D output data lines are connected to the microprocessor bus under program control through the PIA and therefore the A/D $\overline{RD}$ pin can be grounded.

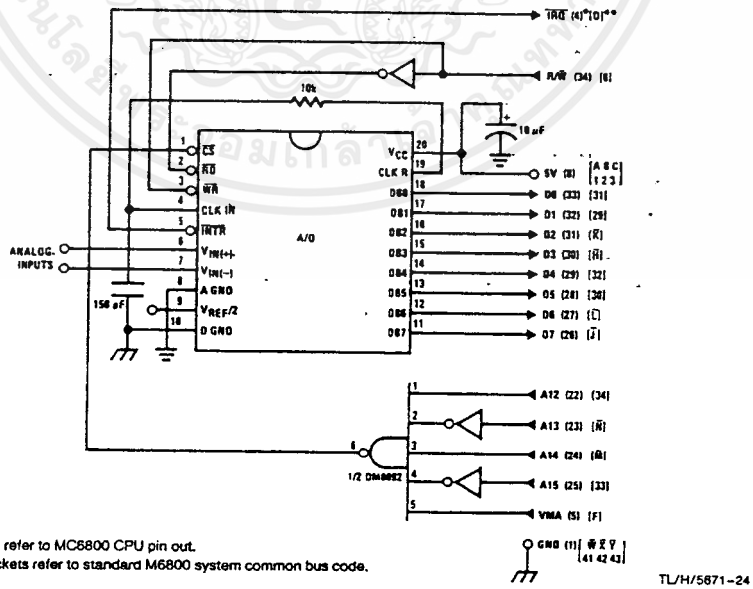
A sample interface program equivalent to the previous one is shown below *Figure 15*. The PIA Data and Control Registers of Port B are located at HEX addresses 8006 and 8007, respectively.

5.0 GENERAL APPLICATIONS

The following applications show some interesting uses for the A/D. The fact that one particular microprocessor is used is not meant to be restrictive. Each of these application circuits would have its counterpart using any microprocessor that is desired.

5.1 Multiple ADC0801 Series to MC6800 CPU Interface

To transfer analog data from several channels to a single microprocessor system, a multiple converter scheme presents several advantages over the conventional multiplexer single-converter approach. With the ADC0801 series, the differential inputs allow individual span adjustment for each channel. Furthermore, all analog input channels are sensed simultaneously, which essentially divides the microprocessor's total system servicing time by the number of channels, since all conversions occur simultaneously. This scheme is shown in *Figure 16*.



Note 1: Numbers in parentheses refer to MC6800 CPU pin out.
Note 2: Number or letters in brackets refer to standard M6800 system common bus code.

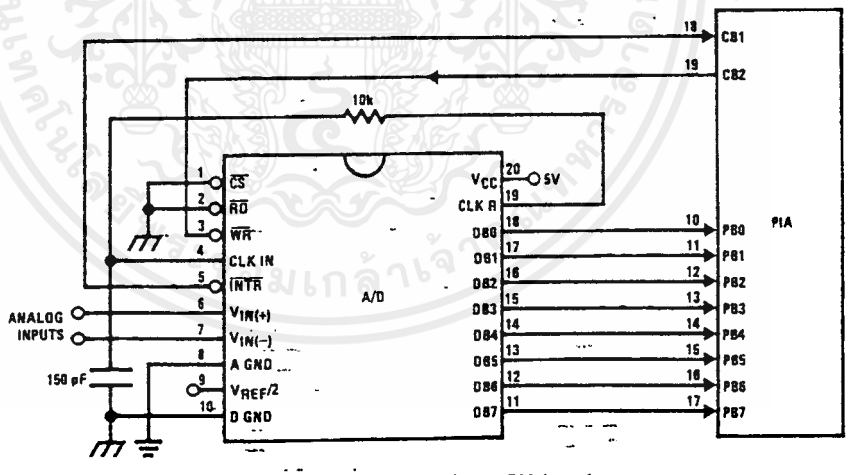
FIGURE 14. ADC0801-MC6800 CPU Interface

Functional Description (Continued)

SAMPLE PROGRAM FOR *FIGURE 14* ADC0801-MC6800 CPU INTERFACE

```
0010   DF 36        DATAIN   STX    TEMP2     ; Save contents of X
0012   CE 00 2C              LDX    #$002C    ; Upon IRQ low CPU
0015   FF FF F8              STX    $FFF8     ; jumps to 002C
0018   B7 50 00              STAA   $5000     ; Start ADC0801.
001B   0E                    CLI
001C   3E           CONVRT   WAI               ; Wait for interrupt
001D   DE 34                 LDX    TEMP1
001F   8C 02 0F              CPX    #$020F    ; Is final data stored?
0022   27 14                 BEQ    ENDP
0024   B7 50 00              STAA   $5000     ; Restarts ADC0801
0027   08                    INX
0028   DF 34                 STX    TEMP1
002A   20 F0                 BRA    CONVRT
002C   DE 34        INTRPT   LDX    TEMP1
002E   B6 50 00              LDAA   $5000     ; Read data
0031   A7 00                 STAA   X          ; Store it at X
0033   3B                    RTI
0034   02 00        TEMP1    FDB    $0200     ; Starting address for
                                               ; data storage
0036   00 00        TEMP2    FDB    $0000
0038   CE 02 00     ENDP     LDX    #$0200    ; Reinitialize TEMP1
003B   DF 34                 STX    TEMP1
003D   DE 36                 LDX    TEMP2
003F   39                    RTS               ; Return from subroutine
                                               ; To user's program
```

Note 1: In order for the microprocessor to service subroutines and interrupts, the stack pointer must be dimensioned in the user's program.



TL/H/5671-2

FIGURE 15. ADC0801-MC6820 PIA Interface

Functional Description (Continued)

SAMPLE PROGRAM FOR *FIGURE 15* ADC0801–MC6820 PIA INTERFACE

```
0010  CE 00 38   DATAIN   LDX   #$0038    ;Upon IRQ low CPU
0013  FF FF F8           STX   $FFF8      ;jumps to 0038
0016  B6 80 06           LDAA  PIAORB     ;Clear possible IRQ flags
0019  4F                 CLRA
001A  B7 80 07           STAA  PIACRB
001D  B7 80 06           STAA  PIAORB     ;Set Port B as input
0020  0E                 CLI
0021  C6 34              LDAB  #$34
0023  86 3D              LDAA  #$3D
0025  F7 80 07   CONVRT  STAB  PIACRB     ;Starts ADC0801
0028  B7 80 07           STAA  PIACRB
002B  3E                 WAI              ;Wait for interrupt
002C  DE 40              LDX   TEMP1
002E  8C 02 0F           CPX   #$020F     ;Is final data stored?
0031  27 0F              BEQ   ENDP
0033  08                 INX
0034  DF 40              STX   TEMP1
0036  20 ED              BRA   CONVRT
0038  DE 40   INTRPT     LDX   TEMP1
003A  B6 80 06           LDAA  PIAORB     ;Read data in
003D  A7 00              STAA  X          ;Store it at X
003F  3B                 RTI
0040  02 00   TEMP1      FDB   $0200      ;Starting address for
                                          ;data storage
0042  CE 02 00 ENDP      LDX   #$0200     ;Reinitialize TEMP1
0045  DF 40              STX   TEMP1
0047  39                 RTS              ;Return from subroutine
      PIAORB   EQU   $8006     ;To user's program
      PIACRB   EQU   $8007
```

The following schematic and sample subroutine (DATA IN) may be used to interface (up to) 8 ADC0801's directly to the MC6800 CPU. This scheme can easily be extended to allow the interface of more converters. In this configuration the converters are (arbitrarily) located at HEX address 5000 in the MC6800 memory space. To save components, the clock signal is derived from just one RC pair on the first converter. This output drives the other A/Ds.

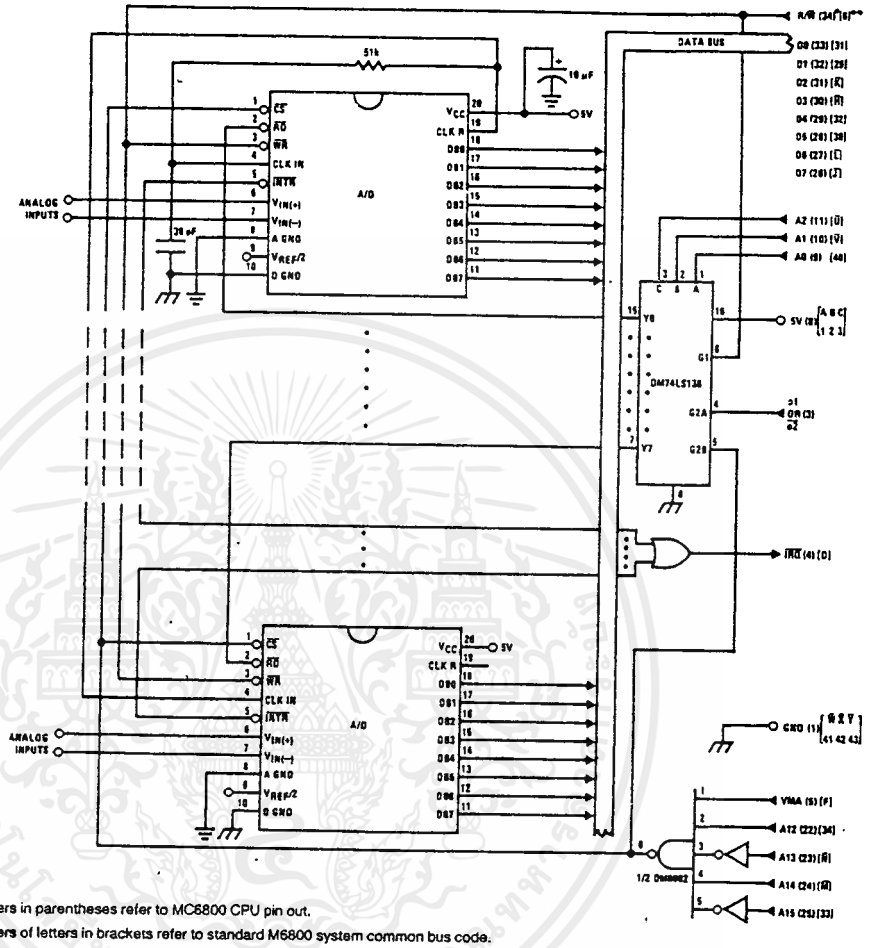
All the converters are started simultaneously with a STORE instruction at HEX address 5000. Note that any other HEX address of the form 5XXX will be decoded by the circuit, pulling all the $\overline{CS}$ inputs low. This can easily be avoided by using a more definitive address decoding scheme. All the interrupts are ORed together to insure that all A/Ds have completed their conversion before the microprocessor is interrupted.

The subroutine, DATA IN, may be called from anywhere in the user's program. Once called, this routine initializes the CPU, starts all the converters simultaneously and waits for the interrupt signal. Upon receiving the interrupt, it reads the converters (from HEX addresses 5000 through 5007) and stores the data successively at (arbitrarily chosen) HEX addresses 0200 to 0207, before returning to the user's program. All CPU registers then recover the original data they had before servicing DATA IN.

5.2 Auto-Zeroed Differential Transducer Amplifier and A/D Converter

The differential inputs of the ADC0801 series eliminate the need to perform a differential to single ended conversion for a differential transducer. Thus, one op amp can be eliminated since the differential to single ended conversion is provided by the differential input of the ADC0801 series. In general, a transducer preamp is required to take advantage of the full A/D converter input dynamic range.

Functional Description (Continued)



Note 1: Numbers in parentheses refer to MC6800 CPU pin out.

Note 2: Numbers of letters in brackets refer to standard M6800 system common bus code.

TL/H/5

FIGURE 16. Interfacing Multiple A/Ds in an MC6800 System

SAMPLE PROGRAM FOR *FIGURE 16* INTERFACING MULTIPLE A/Ds IN AN MC6800 SYSTEM

ADDRESS	HEX CODE	MNEMONICS			COMMENTS
0010	DF 44	DATAIN	STX	TEMP	; Save Contents of X
0012	CE 00 2A		LDX	#$002A	; Upon $\overline{IRQ}$ LOW CPU
0015	FF FF F8		STX	$FFF8	; Jumps to 002A
0018	B7 50 00		STAA	$5000	; Starts all A/D's
001B	0E		CLI		
001C	3E		WAI		; Wait for interrupt
001D	CE 50 00		LDX	#$5000	
0020	DF 40		STX	INDEX1	; Reset both INDEX
0022	CE 02 00		LDX	#$0200	; 1 and 2 to starting
0025	DF 42		STX	INDEX2	; addresses
0027	DE 44		LDX	TEMP	
0029	39		RTS		; Return from subroutine
002A	DE 40	INTRPT	LDX	INDEX1	; INDEX1 → X
002C	A6 00		LDAA	X	; Read data in from A/D at X
002E	08		INX		; Increment X by one
002F	DF 40		STX	INDEX1	; X → INDEX1
0031	DE 42		LDX	INDEX2	; INDEX2 → X

Functional Description (Continued)

SAMPLE PROGRAM FOR *FIGURE 16* INTERFACING MULTIPLE A/Ds IN AN MC6800 SYSTEM

ADDRESS	HEX CODE		MNEMONICS		COMMENTS
0033	A7 00		STAA	X	; Store data at X
0035	8C 02 07		CPX	#$0207	; Have all A/D's been read?
0038	27 05		BEQ	RETURN	; Yes: branch to RETURN
003A	08		INX		; No: increment X by one
003B	DF 42		STX	INDEX2	; X → INDEX2
003D	20 EB		BRA	INTRPT	; Branch to 002A
003F	3B	RETURN	RTI		
0040	50 00	INDEX1	FDB	$5000	; Starting address for A/D
0042	02 00	INDEX2	FDB	$0200	; Starting address for data storage
0044	00 00	TEMP	FDB	$0000	

Note 1: In order for the microprocessor to service subroutines and interrupts, the stack pointer must be dimensioned in the user's program.

For amplification of DC input signals, a major system error is the input offset voltage of the amplifiers used for the preamp. *Figure 17* is a gain of 100 differential preamp whose offset voltage errors will be cancelled by a zeroing subroutine which is performed by the INS8080A microprocessor system. The total allowable input offset voltage error for this preamp is only 50 μV for ¼ LSB error. This would obviously require very precise amplifiers. The expression for the differential output voltage of the preamp is:

$$V_O = \underbrace{[V_{IN}(+) - V_{IN}(-)]}_{\text{SIGNAL}} \underbrace{\left[1 + \frac{2R2}{R1}\right]}_{\text{GAIN}} +$$

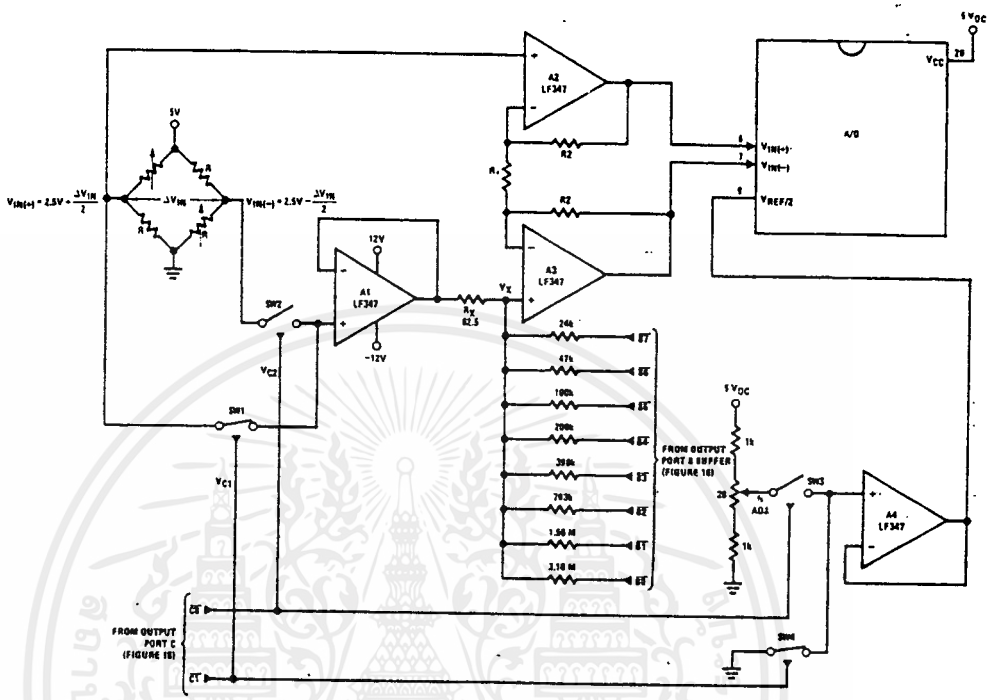
$$\underbrace{(V_{OS_2} - V_{OS_1} - V_{OS_3} \pm I_X R_X)}_{\text{DC ERROR TERM}} \underbrace{\left(1 + \frac{2R2}{R1}\right)}_{\text{GAIN}}$$

where I_X is the current through resistor R_X. All of the offset error terms can be cancelled by making $\pm I_X R_X = V_{OS1} + V_{OS3} - V_{OS2}$. This is the principle of this auto-zeroing scheme.

The INS8080A uses the 3 I/O ports of an INS8255 Programable Peripheral Interface (PPI) to control the auto zeroing and input data from the ADC0801 as shown in *Figure 18*. The PPI is programmed for basic I/O operation (mode 0) with Port A being an input port and Ports B and C being output ports. Two bits of Port C are used to alternately open or close the 2 switches at the input of the preamp. Switch SW1 is closed to force the preamp's differential input to be zero during the zeroing subroutine and then opened and SW2 is then closed for conversion of the actual differential input signal. Using 2 switches in this manner eliminates concern for the ON resistance of the switches as they must conduct only the input bias current of the input amplifiers.

Output Port B is used as a successive approximation register by the 8080 and the binary scaled resistors in series with each output bit create a D/A converter. During the zeroing subroutine, the voltage at V_X increases or decreases as required to make the differential output voltage equal to zero. This is accomplished by ensuring that the voltage at the output of A1 is approximately 2.5V so that a logic "1" (5V) on any output of Port B will source current into node V_X thus raising the voltage at V_X and making the output differential more negative. Conversely, a logic "0" (0V) will pull current out of node V_X and decrease the voltage, causing the differential output to become more positive. For the resistor values shown, V_X can move ±12 mV with a resolution of 50 μV, which will null the offset error term to ¼ LSB of full-scale for the ADC0801. It is important that the voltage levels that drive the auto-zero resistors be constant. Also, for symmetry, a logic swing of 0V to 5V is convenient. To achieve this, a CMOS buffer is used for the logic output signals of Port B and this CMOS package is powered with a stable 5V source. Buffer amplifier A1 is necessary so that it can source or sink the D/A output current.

Functional Description (Continued)

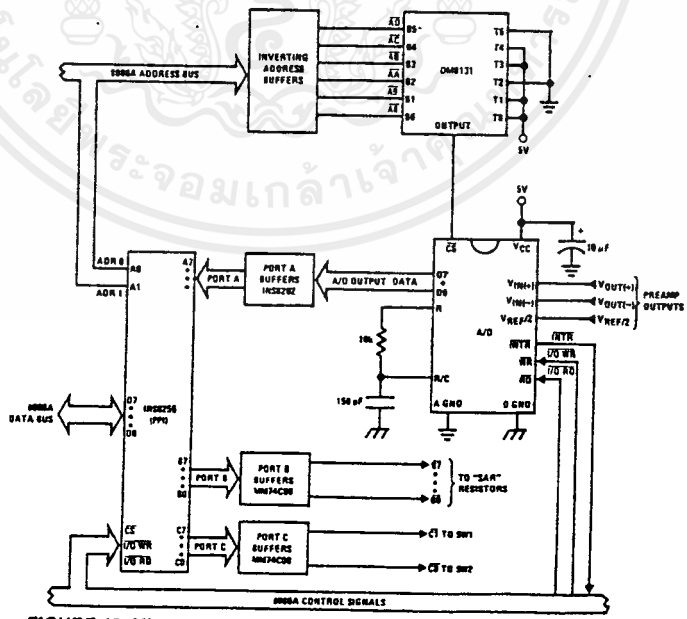


Note 1: R2 = 49.5 R1

Note 2: Switches are LMC13334 CMOS analog switches.

Note 3: The 9 resistors used in the auto-zero section can be ±5% tolerance.

FIGURE 17. Gain of 100 Differential Transducer Preamp



TL/H/5671–27

FIGURE 18. Microprocessor Interface Circuitry for Differential Preamp

A flow chart for the zeroing subroutine is shown in *Figure 19*. It must be noted that the ADC0801 series will output an all zero code when it converts a negative input [$V_{IN}(-) \geq V_{IN}(+)$]. Also, a logic inversion exists as all of the I/O ports are buffered with inverting gates.

Basically, if the data read is zero, the differential output voltage is negative, so a bit in Port B is cleared to pull V_X more negative which will make the output more positive for the next conversion. If the data read is not zero, the output voltage is positive so a bit in Port B is set to make V_X more positive and the output more negative. This continues for 8 approximations and the differential output eventually converges to within 5 mV of zero.

The actual program is given in *Figure 20*. All addresses used are compatible with the BLC 80/10 microcomputer system. In particular:

Port A and the ADC0801 are at port address E4

Port B is at port address E5

Port C is at port address E6

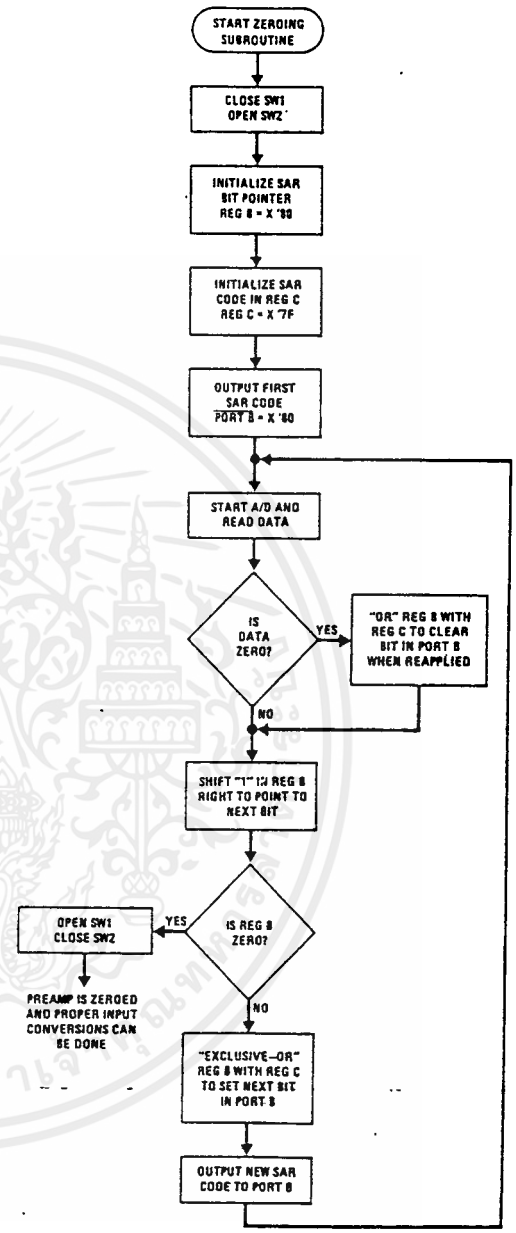
PPI control word port is at port address E7

Program Counter automatically goes to ADDR:3C3D upon acknowledgement of an interrupt from the ADC0801

5.3 Multiple A/D Converters in a Z-80 Interrupt Driven Mode

In data acquisition systems where more than one A/D converter (or other peripheral device) will be interrupting program execution of a microprocessor, there is obviously a need for the CPU to determine which device requires servicing. *Figure 21* and the accompanying software is a method of determining which of 7 ADC0801 converters has completed a conversion ($\overline{INTR}$ asserted) and is requesting an interrupt. This circuit allows starting the A/D converters in any sequence, but will input and store valid data from the converters with a priority sequence of A/D 1 being read first, A/D 2 second, etc., through A/D 7 which would have the lowest priority for data being read. Only the converters whose INT is asserted will be read.

The key to decoding circuitry is the DM74LS373, 8-bit D type flip-flop. When the Z-80 acknowledges the interrupt, the program is vectored to a data input Z-80 subroutine. This subroutine will read a peripheral status word from the DM74LS373 which contains the logic state of the $\overline{INTR}$ outputs of all the converters. Each converter which initiates an interrupt will place a logic "0" in a unique bit position in the status word and the subroutine will determine the identity of the converter and execute a data read. An identifier word (which indicates which A/D the data came from) is stored in the next sequential memory location above the location of the data so the program can keep track of the identity of the data entered.



TL/H/5671–28

FIGURE 19. Flow Chart for Auto-Zero Routine

```
3D00   3E90     MVI 90
3D02   D3E7     Out Control Port
3D04   2601     MVI H 01          Auto-Zero Subroutine    ; Program PPI
3D06   7C       MOV A,H
3D07   D3E6     OUT C                                     ; Close SW1 open SW2
3D09   0680     MVI B 80                                  ; Initialize SAR bit pointer
3D0B   3E7F     MVI A 7F                                  ; Initialize SAR code
3D0D   4F       MOV C,A           Return
3D0E   D3E5     OUT B                                     ; Port B = SAR code
3D10   31AA3D   LXI SP 3DAA       Start                   ; Dimension stack pointer
3D13   D3E4     OUT A                                     ; Start A/D
3D15   FB       IE
3D16   00       NOP               Loop                    ; Loop until INT asserted
3D17   C3163D   JMP Loop
3D1A   7A       MOV A,D           Auto-Zero
3D1B   C600     ADI 00
3D1D   CA2D3D   JZ Set C                                  ; Test A/D output data for zero
3D20   78       MOV A,B           Shift B
3D21   F600     ORI 00                                    ; Clear carry
3D23   1F       RAR                                       ; Shift "1" in B right one place
3D24   FE00     CPI 00                                    ; Is B zero? If yes last
3D26   CA373D   JZ Done                                   ; approximation has been made
3D29   47       MOV B,A
3D2A   C3333D   JMP New C
3D2D   79       MOV A,C           Set C
3D2E   B0       ORA B                                     ; Set bit in C that is in same
3D2F   4F       MOV C,A                                   ; position as "1" in B
3D30   C3203D   JMP Shift B
3D33   A9       XRA C             New C                   ; Clear bit in C that is in
3D34   C30D3D   JMP Return                                ; same position as "1" in B
3D37   47       MOV B,A           Done                    ; then output new SAR code.
3D38   7C       MOV A,H                                   ; Open SW1, close SW2 then
3D39   EE03     XRI 03                                    ; proceed with program. Preamp
3D3B   D3E6     OUT C                                     ; is now zeroed.
3D3D            •                 Normal
                •
                •
                Program for processing
                proper data values
3C3D   DBE4     IN A              Read A/D Subroutine     ; Read A/D data
3C3F   EEFF     XRI FF                                    ; Invert data
3C41   57       MOV D,A
3C42   78       MOV A,B                                   ; Is B Reg = 0? If not stay
3C43   E6FF     ANI FF                                    ; in auto zero subroutine.
3C45   C21A3D   JNZ Auto-Zero
3C48   C33D3D   JMP Normal
```

Note: All numerical values are hexadecimal representations.

FIGURE 20. Software for Auto-Zeroed Differential A/D

5.3 Multiple A/D Converters in a Z-80® Interrupt Driven Mode (Continued)

The following notes apply:

1) It is assumed that the CPU automatically performs a RST 7 instruction when a valid interrupt is acknowledged (CPU is in interrupt mode 1). Hence, the subroutine starting address of X0038.

2) The address bus from the Z-80 and the data bus to the Z-80 are assumed to be inverted by bus drivers.

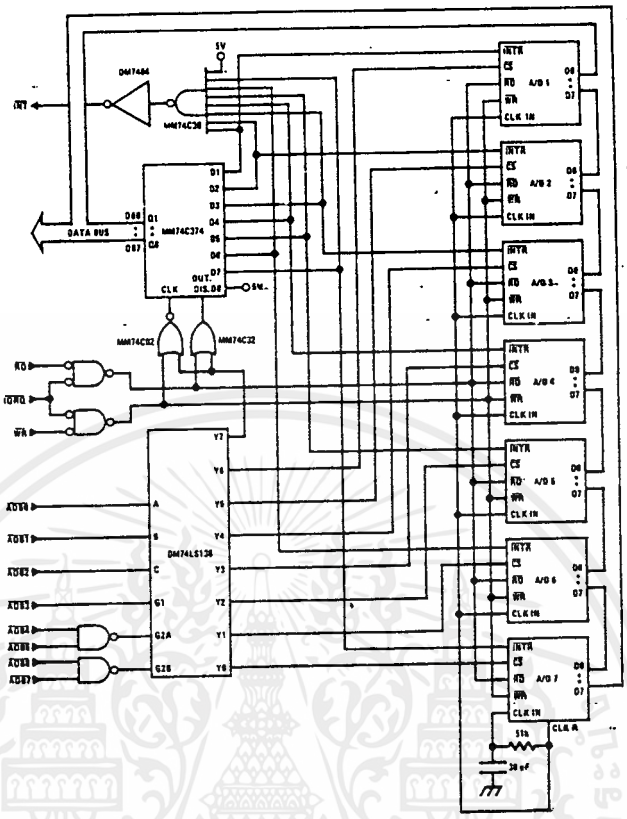
3) A/D data and identifying words will be stored in sequential memory locations starting at the arbitrarily chosen address X 3E00.

4) The stack pointer must be dimensioned in the main program as the RST 7 instruction automatically pushes the PC onto the stack and the subroutine uses an additional 6 stack addresses.

5) The peripherals of concern are mapped into I/O space with the following port assignments:

HEX PORT ADDRESS	PERIPHERAL
00	MM74C374 8-bit flip-flop
01	A/D 1
02	A/D 2
03	A/D 3
04	A/D 4
05	A/D 5
06	A/D 6
07	A/D 7

This port address also serves as the A/D identifying word in the program.



TL/H/5671–29

FIGURE 21. Multiple A/Ds with Z-80 Type Microprocessor

INTERRUPT SERVICING SUBROUTINE

LOC	OBJ CODE	SOURCE STATEMENT		COMMENT
0038	E5		PUSH HL	; Save contents of all registers affected by.
0039	C5		PUSH BC	; this subroutine.
003A	F5		PUSH AF	; Assumed INT mode 1 earlier set.
003B	21 00 3E		LD (HL), X3E00	; Initialize memory pointer where data will be stored.
003E	0E 01		LD C, X01	; C register will be port ADDR of A/D converters.
0040	D300		OUT X00, A	; Load peripheral status word into 8-bit latch.
0042	DB00		IN A, X00	; Load status word into accumulator.
0044	47		LD B, A	; Save the status word.
0045	79	TEST	LD A, C	; Test to see if the status of all A/D's have
0046	FE 08		CP, X08	; been checked. If so, exit subroutine
0048	CA 60 00		JPZ, DONE	
004B	78		LD A, B	; Test a single bit in status word by looking for
004C	1F		RRA	; a "1" to be rotated into the CARRY (an $\overline{INT}$
004D	47		LD B, A	; is loaded as a "1"). If CARRY is set then load
004E	DA 5500		JPC, LOAD	; contents of A/D at port ADDR in C register.
0051	0C	NEXT	INC C	; If CARRY is not set, increment C register to point
0052	C3 4500		JP, TEST	; to next A/D, then test next bit in status word.
0055	ED 78	LOAD	IN A, (C)	; Read data from interrupting A/D and invert
0057	EE FF		XOR FF	; the data.
0059	77		LD (HL), A	; Store the data
005A	2C		INC L	
005B	71		LD (HL), C	; Store A/D identifier (A/D port ADDR).
005C	2C		INC L	
005D	C3 51 00		JP, NEXT	; Test next bit in status word.
0060	F1	DONE	POP AF	; Re-establish all registers as they were
0061	C1		POP BC	; before the interrupt.
0062	E1		POP HL	
0063	C9		RET	; Return to original program

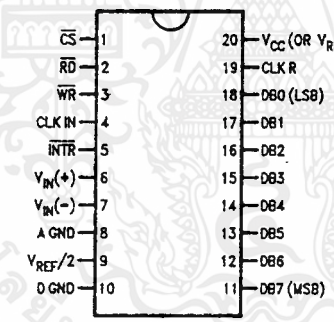
Ordering Information

TEMP RANGE		0°C TO 70°C	0°C TO 70°C	0°C TO 70°C	−40°C TO +85°
ERROR	± ¼ Bit Adjusted				ADC0801LCN
	± ½ Bit Unadjusted	ADC0802LCWM	ADC0802LCV		ADC0802LCN
	± ½ Bit Adjusted	ADC0803LCWM	ADC0803LCV		ADC0803LCN
	± 1Bit Unadjusted	ADC0804LCWM	ADC0804LCV	ADC0804LCN	ADC0805LCN
PACKAGE OUTLINE		M20B—Small Outline	V20A—Chip Carrier	N20A—Molded DIP	

TEMP RANGE		−40°C TO +85°C	−55°C TO +125°C
ERROR	± ¼ Bit Adjusted	ADC0801LCJ	ADC0801LJ
	± ½ Bit Unadjusted	ADC0802LCJ	ADC0802LJ,
	± ½ Bit Adjusted	ADC0803LCJ	ADC0802LJ/883
	± 1Bit Unadjusted	ADC0804LCJ	
PACKAGE OUTLINE		J20A—Cavity DIP	J20A—Cavity DIP

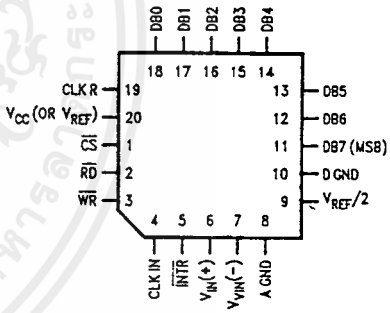
Connection Diagrams

ADC080X
Dual-In-Line and Small Outline (SO) Packages



TL/H/5671–30

ADC080X
Molded Chip Carrier (PCC) Package



TL/H/5671–3

See Ordering Information

Features

- Compatible with MCS-51™ Products
- 2K Bytes of Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- 2.7V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Two-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 15 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial UART Channel
- Direct LED Drive Outputs
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes

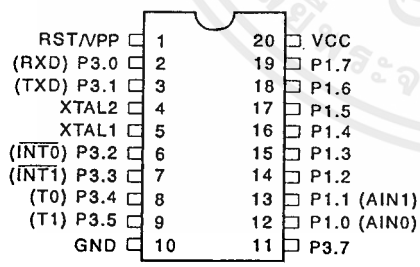
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2K Bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2K Bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

PDIP/SOIC

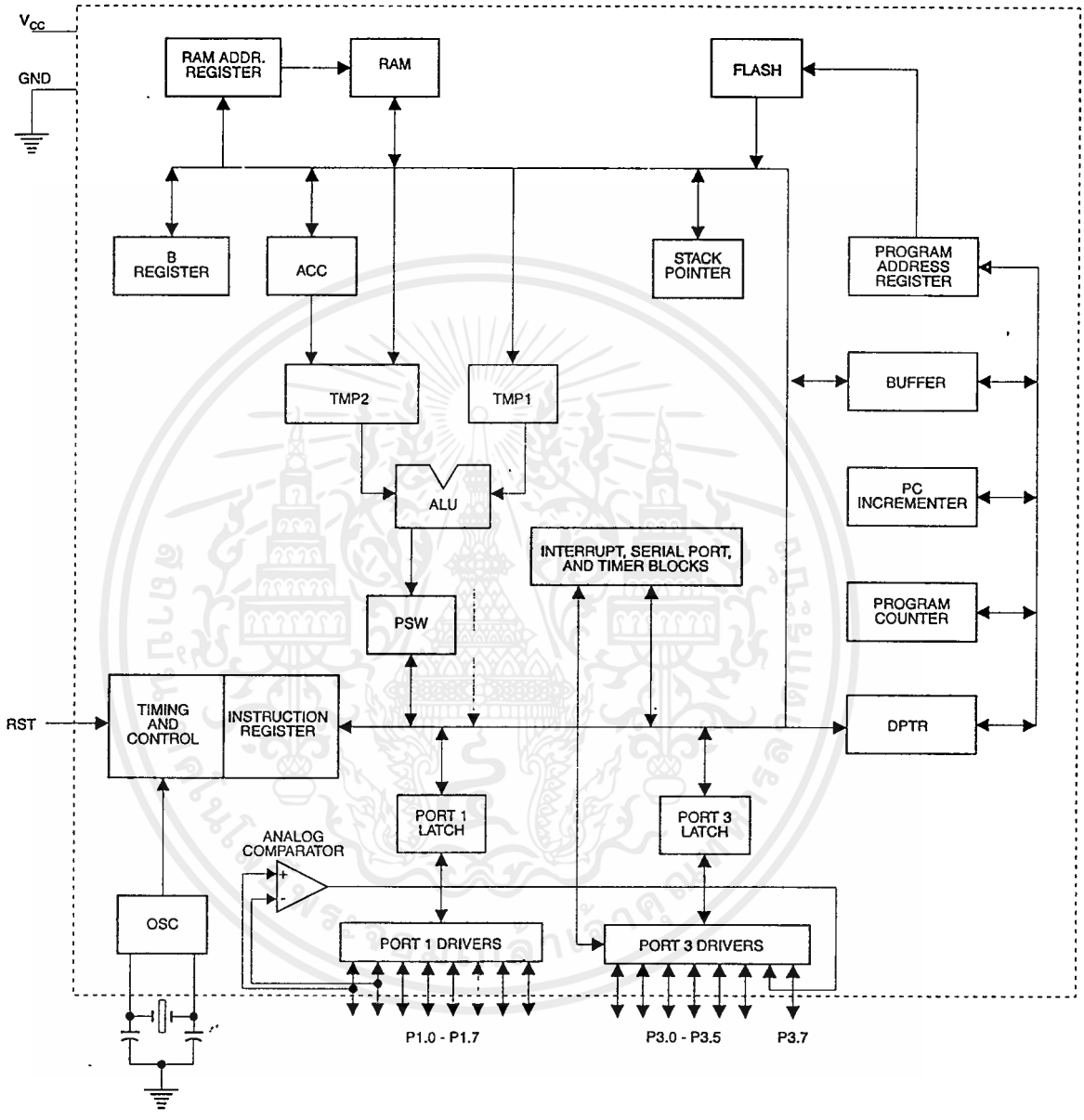


8-Bit Microcontroller with 2K Bytes Flash

AT89C2051

0368D-B-12/97

Block Diagram



Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 1
Port 1 is an 8-bit bidirectional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (I_{IL}) because of the internal pullups.

Port 1 also receives code data during Flash programming and verification.

Port 3
Port 3 pins P3.0 to P3.5, P3.7 are seven bidirectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C2051 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)

Port 3 also receives some control signals for Flash programming and verification.

RST
Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the oscillator is running resets the device.

Each machine cycle takes 12 oscillator or clock cycles.

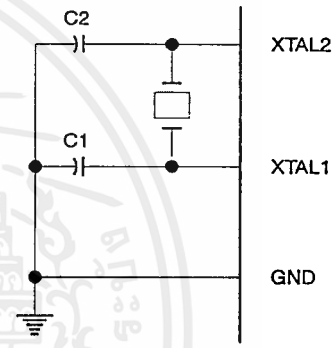
XTAL1
Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2
Output from the inverting oscillator amplifier.

Oscillator Characteristics

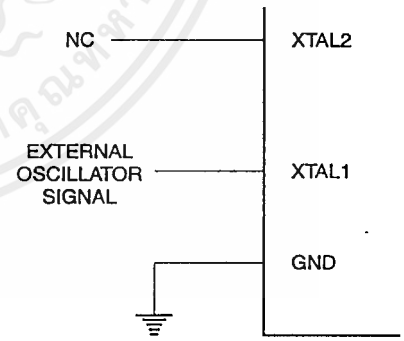
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in the table below.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Table 1. AT89C2051 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H									0CFH
0C0H									0C7H
0B8H	IP XXX00000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0XX00000								0AFH
0A0H									0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H		SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000	87H

Restrictions on Certain Instructions

The AT89C2051 and is an economical and cost-effective member of Atmel's growing family of microcontrollers. It contains 2K bytes of flash program memory. It is fully compatible with the MCS-51 architecture, and can be programmed using the MCS-51 instruction set. However, there are a few considerations one must keep in mind when utilizing certain instructions to program this device.

All the instructions related to jumping or branching should be restricted such that the destination address falls within the physical program memory space of the device, which is 2K for the AT89C2051. This should be the responsibility of the software programmer. For example, LJMP 7E0H would be a valid instruction for the AT89C2051 (with 2K of memory), whereas LJMP 900H would not.

1. Branching instructions:
LCALL, LJMP, ACALL, AJMP, SJMP, JMP @A+DPTR

These unconditional branching instructions will execute correctly as long as the programmer keeps in mind that the destination branching address must fall within the physical boundaries of the program memory size (locations 00H to 7FFH for the 89C2051). Violating the physical space limits may cause unknown program behavior.

CJNE [...], DJNZ [...], JB, JNB, JC, JNC, JBC, JZ, JNZ With these conditional branching instructions the same rule above applies. Again, violating the memory boundaries may cause erratic execution.

For applications involving interrupts the normal interrupt service routine address locations of the 80C51 family architecture have been preserved.

2. MOVX-related instructions, Data Memory:
The AT89C2051 contains 128 bytes of internal data memory. Thus, in the AT89C2051 the stack depth is limited to 128 bytes, the amount of available RAM. External DATA memory access is not supported in this device, nor is external PROGRAM memory execution. Therefore, no MOVX [...] instructions should be included in the program.

A typical 80C51 assembler will still assemble instructions, even if they are written in violation of the restrictions mentioned above. It is the responsibility of the controller user to know the physical features and limitations of the device being used and adjust the instructions used correspondingly.

Program Memory Lock Bits

On the chip are two lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

Lock Bit Protection Modes[1]

Program Lock Bits			
	LB1	LB2	Protection Type
1	U	U	No program lock features.
2	P	U	Further programming of the Flash is disabled.
3	P	P	Same as mode 2, also verify is disabled.

Note: 1. The Lock Bits can only be erased with the Chip Erase operation.

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Power Down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

Programming The Flash

The AT89C2051 is shipped with the 2K bytes of on-chip PEROM code memory array in the erased state (i.e., contents = FFH) and ready to be programmed. The code memory array is programmed one byte at a time. *Once the array is programmed, to re-program any non-blank byte, the entire memory array needs to be erased electrically.*

Internal Address Counter: The AT89C2051 contains an internal PEROM address counter which is always reset to 000H on the rising edge of RST and is advanced by applying a positive going pulse to pin XTAL1.

Programming Algorithm: To program the AT89C2051, the following sequence is recommended.

1. Power-up sequence:
   Apply power between V_{CC} and GND pins
   Set RST and XTAL1 to GND
2. Set pin RST to 'H'
   Set pin P3.2 to 'H'
3. Apply the appropriate combination of 'H' or 'L' logic levels to pins P3.3, P3.4, P3.5, P3.7 to select one of the programming operations shown in the PEROM Programming Modes table.

To Program and Verify the Array:

4. Apply data for Code byte at location 000H to P1.0 to P1.7.
5. Raise RST to 12V to enable programming.
6. Pulse P3.2 once to program a byte in the PEROM array or the lock bits. The byte-write cycle is self-timed and typically takes 1.2 ms.
7. To verify the programmed data, lower RST from 12V to logic 'H' level and set pins P3.3 to P3.7 to the appropiate levels. Output data can be read at the port P1 pins.
8. To program a byte at the next address location, pulse XTAL1 pin once to advance the internal address counter. Apply new data to the port P1 pins.
9. Repeat steps 5 through 8, changing data and advancing the address counter for the entire 2K bytes array or until the end of the object file is reached.
10. Power-off sequence:
    set XTAL1 to 'L'
    set RST to 'L'
    Turn V_{CC} power off

Data Polling: The AT89C2051 features Data Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P1.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. Data Polling may begin any time after a write cycle has been initiated.

Ready/Busy: The Progress of byte programming can also be monitored by the RDY/BSY output signal. Pin P3.1 is pulled low after P3.2 goes High during programming to indicate BUSY. P3.1 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed code data can be read back via the data lines for verification:

1. Reset the internal address counter to 000H by bringing RST from 'L' to 'H'.
2. Apply the appropriate control signals for Read Code data and read the output data at the port P1 pins.
3. Pulse pin XTAL1 once to advance the internal address counter.
4. Read the next code data byte at the port P1 pins.
5. Repeat steps 3 and 4 until the entire array is read.

The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire PEROM array (2K bytes) and the two Lock Bits are erased electrically by using the proper combination of control signals and by holding P3.2 low for 10 ms. The code array is written with all "1"s in the Chip Erase operation and must be executed before any non-blank memory byte can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 001H, and 002H, except that P3.5 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(001H) = 21H indicates 89C2051

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Flash Programming Modes

Mode		RST/VPP	P3.2/$\overline{PROG}$	P3.3	P3.4	P3.5	P3.7
Write Code Data[1][3]		12V		L	H	H	H
Read Code Data[1]		H	H	L	L	H	H
Write Lock	Bit - 1	12V		H	H	H	H
	Bit - 2	12V		H	H	L	L
Chip Erase		12V	(2)	H	L	L	L
Read Signature Byte		H	H	L	L	L	L

Notes:
1. The internal PEROM address counter is reset to 000H on the rising edge of RST and is advanced by a positive pulse at XTAL 1 pin.
2. Chip Erase requires a 10-ms $\overline{PROG}$ pulse.
3. P3.1 is pulled Low during programming to indicate RDY/$\overline{BSY}$.

Figure 3. Programming the Flash Memory

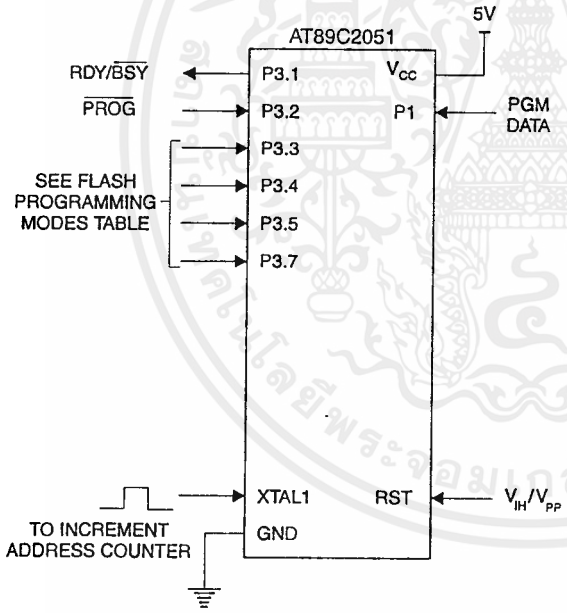
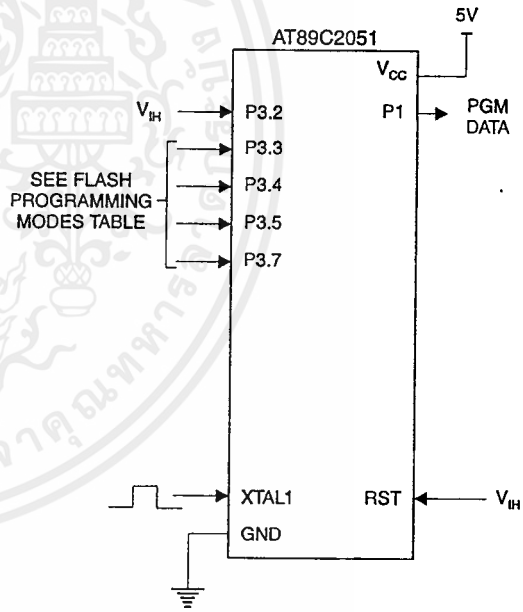


Figure 4. Verifying the Flash Memory



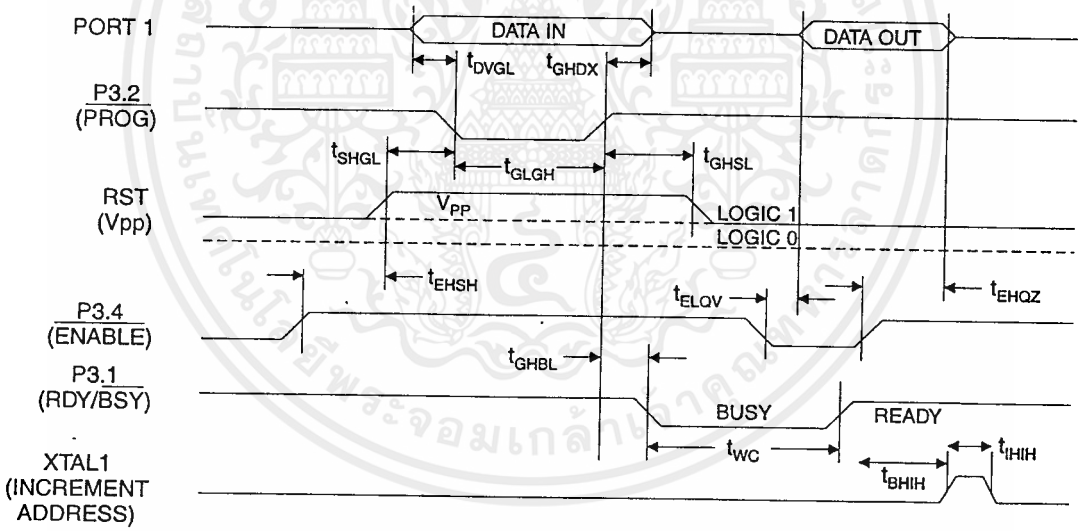
Flash Programming and Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ± 10%

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Enable Voltage	11.5	12.5	V
I_{PP}	Programming Enable Current		250	µA
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	1.0		µs
t_{GHDX}	Data Hold After $\overline{PROG}$	1.0		µs
t_{EHSH}	P3.4 ($\overline{ENABLE}$) High to V_{PP}	1.0		µs
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		1.0	µs
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	1.0	µs
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		50	ns
t_{WC}	Byte Write Cycle Time		2.0	ms
t_{BHIH}	RDY/$\overline{BSY}$\ to Increment Clock Delay	1.0		µs
t_{IHIL}	Increment Clock High	200		ns

Note: 1. Only used in 12-volt programming mode.

Flash Programming and Verification Waveforms



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	25.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

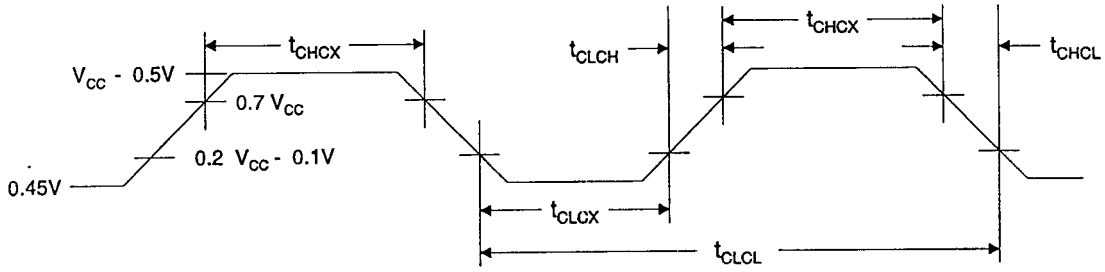
T_A = -40°C to 85°C, V_{CC} = 2.0V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage		-0.5	0.2 V_{CC} - 0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage[1] (Ports 1, 3)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 2.7V		0.5	V
V_{OH}	Output High Voltage (Ports 1, 3)	I_{OH} = -80 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -30 µA	0.75 V_{CC}		V
		I_{OH} = -12 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1, 3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-750	µA
I_{LI}	Input Leakage Current (Port P1.0, P1.1)	0 < V_{IN} < V_{CC}		±10	µA
V_{OS}	Comparator Input Offset Voltage	V_{CC} = 5V		20	mV
V_{CM}	Comparator Input Common Mode Voltage		0	V_{CC}	V
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz, V_{CC} = 6V/3V		15/5.5	mA
		Idle Mode, 12 MHz, V_{CC} = 6V/3V P1.0 & P1.1 = 0V or V_{CC}		5/1	mA
	Power Down Mode[2]	V_{CC} = 6V P1.0 & P1.1 = 0V or V_{CC}		100	µA
		V_{CC} = 3V P1.0 & P1.1 = 0V or V_{CC}		20	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 20 mA
Maximum total I_{OL} for all output pins: 80 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum V_{CC} for Power Down is 2V.

External Clock Drive Waveforms



External Clock Drive

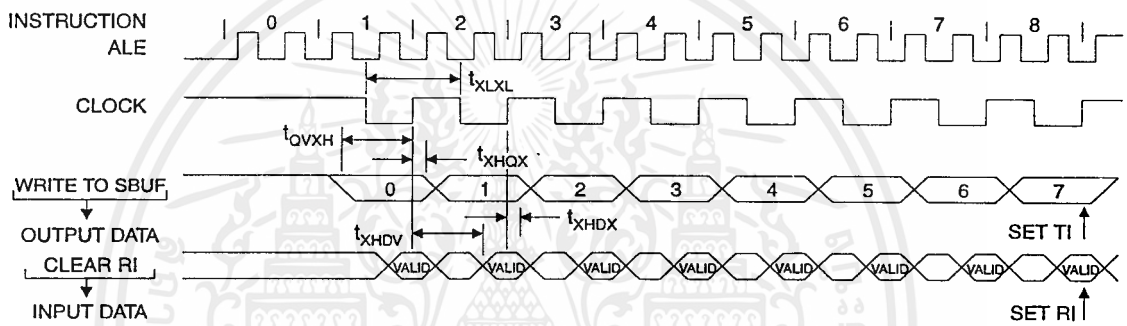
Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	12	0	24	MHz
t_{CLCL}	Clock Period	83.3		41.6		ns
t_{CHCX}	High Time	30		15		ns
t_{CLCX}	Low Time	30		15		ns
t_{CLCH}	Rise Time		20		20	ns
t_{CHCL}	Fall Time		20		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

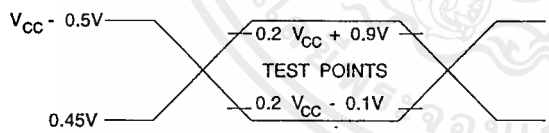
(V_{CC} = 5.0V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-117		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}$-133	ns

Shift Register Mode Timing Waveforms

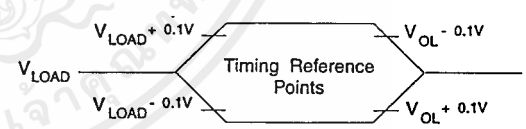


AC Testing Input/Output Waveforms(1)



Note: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

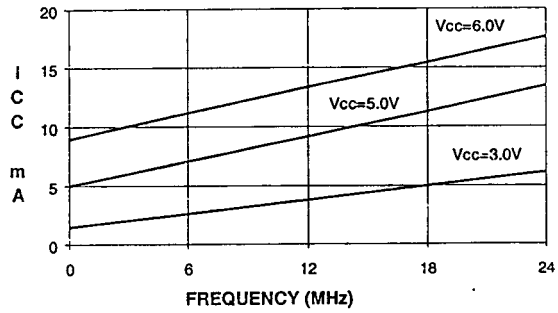
Float Waveforms(1)



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change frothe loaded V_{OH}/V_{OL} level occurs.

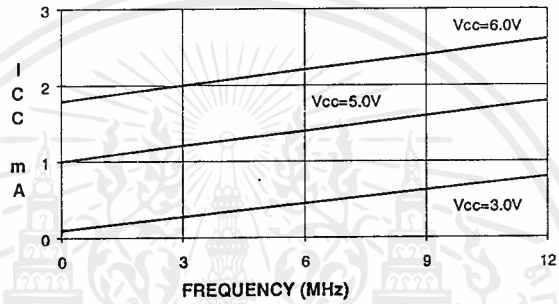
AT89C2051

TYPICAL ICC - ACTIVE (85°C)



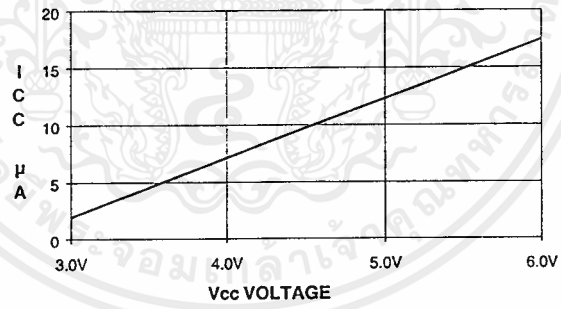
AT89C2051

TYPICAL ICC - IDLE (85°C)



AT89C2051

TYPICAL ICC vs. VOLTAGE- POWER DOWN (85°C)



Notes:
1. XTAL1 tied to GND for I_{CC} (power down)
2. P1.0 and P1.1 = V_{CC} or GND
3. Lock bits programmed

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	2.7V to 6.0V	AT89C2051-12PC AT89C2051-12SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-12PI AT89C2051-12SI	20P3 20S	Industrial (-40°C to 85°C)
		AT89C2051-12PA AT89C2051-12SA	20P3 20S	Automotive (-40°C to 105°C)
24	4.0V to 6.0V	AT89C2051-24PC AT89C2051-24SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-24PI AT89C2051-24SI	20P3 20S	Industrial (-40°C to 85°C)

Package Type	
20P3	20 Lead, 0.300" Wide, Plastic Dual In-line Package (PDIP)
20S	20 Lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)

เอกสารอ้างอิง

[1] Ioan D. Landan , "System Identification and Control Design," Prentice Hall , 1990.

[2] L.Ljung , "System Identification ," Prentice Hall , 1987.

[3] Rolf Johansson , "System Modeling and Identification ," Prentice Hall , 1993.

[4] Torsten Soderstrom , Petre Stoica , "System Identification ," Prentice Hall , 1988.

[5] วิโรจน์ อัศวรังสี, " การใช้งานออปแอมป์และลิเนียร์ไอซี " สำนักพิมพ์ซีเอ็ดยูเคชัน จำกัด ,1987.

[6] บัณฑิต จามรภูติ, " เข้าใจการทำงานดอสอินเทอร์เฟส " สำนักพิมพ์ซีเอ็ดยูเคชั่น จำกัด , 2539.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดีเนื่องจากได้รับคำแนะนำและได้รับความช่วยเหลือจากหลายท่านดังนี้

อาจารย์ พรสุข รติโรจน์อนันต์ อาจารย์ที่ปรึกษา

รศ. สุเธียร เกียรติสุนทร อาจารย์ที่ปรึกษา

รศ. วิพันธ์ ปรีชาพานิช อาจารย์ที่ปรึกษา

และเพื่อนๆ(เอ๋,แดน,อู๋,ตี๋,โอ)จากภาคอีเล็คทรอนิกส์ และ(ป่าน,และตั๊มที่คอยก่อกวน) จากภาคคอนโทรล และกลุ่มโปรเจ็คที่คอยให้คำปรึกษาและช่วยเหลือตลอดเวลาจึงขอขอบคุณมา ณ ที่นี้