สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

เครื่องวัดค่าความถ่วงจำเพาะ

MEASURING SPECIFIC GRAVITY

นายอัสนัย อาจจำนงค์

นายอุทัย กิ้มเส้ง

เลขหมู่.....
เลขทะเบียน.....61904
วัน,เดือน,ปี.....24 ก.ค. 2549

b.....
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

MEASURING SPECIFIC GRAVITY

ASSANAI AJJUMNONK

UTHAI KIMSENG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2004

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์	เครื่องวัดค่าความถ่วงจำเพาะ MEASURING SPECIFIC GRAVITY			
นักศึกษาผู้จัดทำ	นายอัสนัย	อาจจำนงค์	**รหัสประจำตัว**	45015630
	นายอุทัย	กิ้มเส้ง	**รหัสประจำตัว**	45015632
ปริญญา	วิศวกรรมศาสตรบัณฑิต			
สาขาวิชา	วิศวกรรมการวัดคุม			
ปีการศึกษา	2547			

อาจารย์ผู้ควบคุมปริญญานิพนธ์	**ลายมือชื่อ**
รศ. ประภาษ อุคคกิมาพันธุ์ รศ. ประสิทธิ์ จุลเสรีวงศ์ ผศ. อัมพวัน ใจกล้า	

วัน/เดือน/ปี ที่สอบ วันพฤหัสบดีที่ 24 มีนาคม พ.ศ. 2548

สถานที่สอบ ณ ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารองรับแล้ว

(รศ. ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	เครื่องวัดค่าความถ่วงจำเพาะ
	MEASURING SPECIFIC GRAVITY
นักศึกษาผู้จัดทำ	นายอัสนัย อาจจำนงค์
	นายอุทัย กิ้มเส้ง
อาจารย์ที่ปรึกษา	รศ. ประภาษ อุดคกิมาพันธุ์
	รศ. ประสิทธิ์ จุลเสรีวงศ์
	ผศ. อัมพวัน ใจกล้า
ปีการศึกษา	2547

บทคัดย่อ

ในการหาค่าความถ่วงจำเพาะของของเหลวโดยมากเราได้จากตาราง ค่าความถ่วงจำเพาะของของเหลว แต่ในบางครั้ง เมื่อเราต้องการอยากจะรู้ค่าความถ่วงจำเพาะของของเหลวบางชนิดที่ไม่สามารถพบได้จากตารางค่าความถ่วงจำเพาะ หรือไม่สามารถรู้ว่าของเหลวนั้นเป็นของเหลวชนิดใด ซึ่งเป็นการยากในการหาค่าความถ่วงจำเพาะของของเหลวนั้นๆ จากปัญหานี้เราจึงได้มีความคิดในการสร้างเครื่องวัดค่าความถ่วงจำเพาะขึ้นมา โดยอาศัยหลักการจากแนวคิดของอาร์คิมิดิสมาทำการสร้างเครื่องมือสำหรับวัดค่าความถ่วงจำเพาะ

เครื่องวัดค่าความถ่วงจำเพาะนี้จะเป็นการนำเอาไมโครคอนโทรลเลอร์มาใช้ประโยชน์โดยการนำข้อมูลการเปลี่ยนแปลงค่าทางไฟฟ้าของโหลดเซลล์ มาแสดงผลค่าในรูปแบบของค่าน้ำหนักต่อปริมาตร และเก็บข้อมูลนั้นได้รวมทั้งนำข้อมูลมาประมวลผลได้อีกด้วย

Thesis Title	Measuring Specific Gravity	
Authors	Mr. Assanai	Ajjumnonk
	Mr. Uthai	Kimseng
Thesis Advisor	Assoc.Prof. Prapart	Aukkagimaphan
	Assoc.Prof. Prasit	Julsereewong
	Asst.Prof. Ampawan	Jaikla
Year	2004	

ABSTRACT

Due to measuring specific gravity, the most common method is to get from the substances' specific gravity table. But sometime we can't do that because of any reason. According to Archimedes 's theory, we can apply it to invent the equipment for measuring specific gravity.

The design's criteria is to transfer the electrical changing value of load cell into the weight per volume. We then calculate previous outcome by microcontroller to specific gravity value.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดีก็เพราะได้รับความเมตตากรุณาจาก รองศาสตราจารย์ ประภาษ อุคคกิมาพันธุ์ รองศาสตราจารย์ ประสิทธิ์ จุลเสรีวงศ์ และผู้ช่วยศาสตราจารย์ อัมพวัน ใจกล้า ที่ได้ให้คำแนะนำแก่ผู้วิจัยตลอดเรื่อยมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์ และเครื่องมือต่างๆ ที่ใช้ในการจัดทำปริญญานิพนธ์นี้ ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณอาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่าน คุณกฤษณ์ เสมอพิทักษ์ และคุณเจตติยะ ศรีพิทักษ์ ที่ให้คำแนะนำตักเตือนอันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VI

สารบัญรูป VII

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและเหตุจูงใจ 1

1.2 วัตถุประสงค์ 1

1.3 ผลที่คาดว่าจะได้รับ 1

1.4 ขอบเขตปริญญานิพนธ์ 2

1.5 รายละเอียดของปริญญานิพนธ์ 2

บทที่ 2 ทฤษฎีเบื้องต้นที่เกี่ยวข้อง 3

2.1 ความหมายของของเหลว (Fluid) 3

2.1.1 ความหนาแน่น (Density) 4

2.1.2 น้ำหนักจำเพาะ (Specific Weight) 5

2.1.3 ความถ่วงจำเพาะ (Specific Gravity) 5

2.2 เกจวัดความเครียด (Strain Gauge) 6

2.2.1 วงจรสเตรนเกจ (Strain Gauge Circuit) 14

2.2.2 วงจรแบ่งแรงดัน (Voltage Divider) 14

2.2.3 วงจรวีทสโตนบริดจ์ (Wheatstone Bridge) 18

2.2.4 ชนิดของวงจรบริดจ์ที่นำมาใช้งาน 19

2.3 สเต็ปปิ้งมอเตอร์ (Stepping Motor) 22

2.3.1 ชนิดและโครงสร้างของสเต็ปปิ้งมอเตอร์ 23

2.3.2 หลักการทำงานของสเต็ปปิ้งมอเตอร์ 24

2.3.3 โครงสร้างและการทำงานของ Stepping Motor 25

สารบัญ (ต่อ)

หน้า

2.3.4 การกระตุ้นเฟสของลวดสเตเตอร์ 26

2.4 การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล 29

2.4.1 หลักการทำงานของ A/D Converter 30

2.4.2 ความผิดพลาดของ A/D Converter 33

2.4.3 ซัสเซสซีพแอบพล็อกซิเมชั่น A/D Converter 35

2.5 การรับส่งข้อมูลแบบอนุกรม 36

2.5.1 หลักการเบื้องต้นของการรับส่งข้อมูลแบบอนุกรม 41

2.6 Microsoft Visual Basic 41

2.6.1 ฮาร์ดแวร์หรืออุปกรณ์ที่เหมาะสมกับการใช้โปรแกรม Visual Basic 42

2.6.2 เริ่มต้นกับโปรแกรมวิชวลเบสิก 42

2.6.3 ไฟล์ประเภทต่างๆ ที่มีในโปรเจ็กต์ของ VB6 44

2.6.4 คอนโทรลที่สำคัญที่ใช้ในโครงงาน 45

บทที่ 3 หลักการและทฤษฎีที่ใช้ในการออกแบบ 54

3.1 หลักการการทำงาน 54

3.2 ทฤษฎีและหลักการอ้างอิง 57

บทที่ 4 ผลการทดลอง 59

4.1 กล่าวนำ 59

4.2 การทดลองตอนที่ 1 59

4.3 ผลการทดลองตอนที่ 2 63

บทที่ 5 สรุปผลการทดลองและข้อเสนอแนะ 69

5.1 สรุปผลการทดลอง 69

5.2 ปัญหาที่เกิดขึ้นและแนวทางแก้ไข 69

5.3 ข้อเสนอแนะ 69

บรรณานุกรม 70

สารบัญ (ต่อ)

หน้า

ภาคผนวก ก. โปรแกรมการทำงาน..... 71

ภาคผนวก ข. เอกสารประกอบ..... 85

สารบัญตาราง

ตารางที่ **หน้า**

2.1 ตารางหน่วยของระบบอังกฤษ และระบบ SI. 4
2.2 ตารางสำหรับแอพพลิเคชั่นต่างๆ 43
2.3 แสดงกลุ่มไฟล์โปรเจ็กต์ 45
2.4 แสดงการตรวจเหตุการณ์ (Event) 51
4.1 ตารางแสดงผลการวัดค่าความถ่วงจำเพาะของของเหลวแต่ละชนิด 68

สารบัญรูป

รูปที่ หน้า

2.1 ความสัมพันธ์ระหว่างความเค้นกับความเครียด 8

2.2 Strain Gauge แบบยึดติด 9

2.3 Strain Gauge แบบไม่ยึดติด 10

2.4 วงจรวีทสโตนบริดจ์อย่างง่าย ๆ 12

2.5 Static Strain และ Dynamic Strain ในอุดมคติ 14

2.6 ลักษณะของ Semiconductor ที่ใช้เป็น Strain Gauge 14

2.7 วงจร Potentiometer 15

2.8 วงจรวีทสโตนบริดจ์แบบ 4 แขน 18

2.9 การติดตั้ง Strain Gauge ตามเงื่อนไข Poisson แบบ Half Bridge 20

2.10 ลักษณะการติดตั้ง Strain Gauge ในแบบต่างๆ 22

2.11 สเต็ปมอเตอร์แบบ 5 สาย 23

2.12 สเต็ปมอเตอร์แบบ 6 สาย 23

2.13 สเต็ปมอเตอร์แบบไบโพลาร์ 23

2.14 แสดงภาพถ่ายโครงสร้างของมอเตอร์ 24

2.15 เส้นแรงแม่เหล็กที่ทำให้เกิดแรงบิด 25

2.16 แสดงการเคลื่อนที่ทีละสเต็ป เมื่อกระตุ้นเฟส 1 เฟส 2 26

2.17 แสดงการกระตุ้น แบบเดินหน้าและถอยหลัง 26

2.18 แสดงการทำงานของสเต็ปปิ้งมอเตอร์ 27

2.19 แสดงการกระตุ้นเฟสของสเตเตอร์ 27

2.20 แสดงการป้อนพัลส์ ให้กับสเต็ปปิ้งมอเตอร์ และการหมุน 28

2.21 เส้นแรงแม่เหล็กเมื่อขับแบบ 2 เฟส 28

2.22 ลักษณะเส้นแรงแม่เหล็กเมื่อขับแบบ 2 เฟส 29

2.23 แสดงการต่อขดลวดของสเต็ปปิ้งมอเตอร์ 29

2.24 แสดงสัญญาณของ A/D Converter 30

2.25 คุณสมบัติของ A/D Converter แบบ 4 บิต 30

2.26 แสดงการตอบสนองของเวลาแปลงสัญญาณของ A/D Converter 31

2.27 ขบวนการแปลงสัญญาณดิจิตอลด้วย A/D Converter 32

2.28 ข้อผิดพลาดของ A/D Converter 32

สารบัญรูป (ต่อ)

รูปที่ หน้า

2.29 A/D Converter 34

2.30 กราฟเอาต์พุตของส่วน D/A ของ A/D Converter 36

2.31 คอนเวอร์เตอร์ซึ่งใช้เทคนิค SAR 35

2.32 กราฟของส่วน D/A ระหว่างการแปลงสัญญาณ 36

2.33 แสดงบิตของข้อมูลที่ทำการส่งแบบอนุกรม 37

2.34 แสดงรูปสัญญาณของข้อมูลที่ถูกส่งไปตามสายส่งแบบอนุกรม 37

2.35 การเพิ่ม START BIT เข้าไปก่อนหน้าบิต D0 39

2.36 การเพิ่ม PARITY BIT ลงไปในข้อมูลแต่ละไบต์ 40

2.37 รูปแบบของข้อมูลแต่ละไบต์ในการรับส่งข้อมูลแบบอนุกรม 40

2.38 ไดอะล็อกซ์บ๊อกซ์ชนิดของแอพพลิเคชันสำหรับที่ให้โปรแกรมเมอร์เลือกพัฒนา 43

2.39 คอนโทรลคอมมานด์บัสทอนบนทูลบ๊อกซ์ และปรากฏอยู่บนฟอร์ม 46

2.40 คอนโทรลเท็กซ์บ๊อกซ์บนทูลบ๊อกซ์ และที่ปรากฏอยู่บนฟอร์ม 46

2.41 คอนโทรลลาเบลบนทูลบ๊อร์ก และที่ปรากฏอยู่บนฟอร์ม 47

2.42 คอนโทรลออฟชั่นบัสทอนบนทูลบ๊อกซ์ และที่ปรากฏลอยู่บนฟอร์ม 48

2.43 คอนโทรลคอมโบบ๊อกซ์เมื่อปรากฏบนทูลบ๊อกซ์ และอยู่บนฟอร์ม 48

2.44 แสดงคอนโทรลไทล์เมอร์บนทูลบ๊อกซ์ และปรากฏอยู่บนฟอร์ม 49

2.45 แสดงการส่งเปลี่ยนของสายข้อมูล 52

3.1 แสดงโครงสร้างการทำงานของเครื่องวัด 54

3.2 แสดง Flow Chart การทำงานของเครื่อง 55

3.3 แสดงสัญญาณพัลล์ที่จ่ายให้มอเตอร์สเต็ปปิ้ง กับระยะการเคลื่อนที่ของลูกตุ้ม 56

3.4 แสดงความสัมพันธ์ของระยะการเคลื่อนที่ของลูกตุ้มกับน้ำหนักที่ได้จากลูกตุ้ม 56

3.5 แสดงวัตถุที่จมอยู่ในภาชนะที่มีของเหลวอยู่ 57

3.6 ลูกตุ้มทรงกระบอกที่จุ่มลงไปในภาชนะที่บรรจุของเหลว 58

4.1 ชุดการทดลองเบื้องต้น 59

4.2 ของเหลวที่ทดลองเป็นน้ำ 60

4.3 ของเหลวที่ทดลองเป็นน้ำมันก๊าด 61

4.4 ของเหลวที่ทดลองเป็นน้ำมันปาล์ม 61

4.5 ของเหลวที่ทดลองเป็นน้ำมันปาล์ม กับ น้ำ 62

สารบัญรูป (ต่อ)

รูปที่ หน้า

4.6 โครงสร้างของเครื่องวัดค่าความถ่วงจำเพาะที่สร้างขึ้น 63
4.7 แสดงปุ่มการทำงานของเครื่อง 63
4.8 แสดงวิธีการทำงานของเครื่องวัดค่าความถ่วงจำเพาะ 64
4.9 แสดงการวัดของเหลวที่ต้องการจะรู้ค่าความถ่วงจำเพาะ 64
4.10 แสดงหน้าจอการแสดงผลก่อนเริ่มการทดลอง 65
4.11 ของเหลวที่ทดลองเป็นน้ำ 65
4.12 ของเหลวที่ทดลองเป็นน้ำมันปาล์ม 66
4.13 ของเหลวที่ทดลองเป็นน้ำมันก๊าด 66
4.14 ของเหลวที่ทดลองเป็นน้ำมันเบนซิน 67
4.15 ของเหลวที่ทดลองเป็นน้ำ กับ น้ำมันปาล์ม 67
4.16 ของเหลวที่ทดลองเป็นน้ำ กับ น้ำมันเบนซิน 68

บทที่ 1
บทนำ

1.1 กล่าวนำ

ในปัจจุบันอุตสาหกรรมและการควบคุมต่างๆ ได้พัฒนาไปมากจากเดิมที่ส่วนใหญ่จะเป็นแบบธรรมดามาเป็นแบบกึ่งอัตโนมัติ และเป็นอัตโนมัติ ดังนั้นอุปกรณ์เครื่องมือวัดต่างๆ ที่ใช้ในการประกอบการเหล่านี้ จะต้องได้รับการพัฒนาให้ดีขึ้นตามไปด้วย จะเห็นได้ว่าในปัจจุบันเราได้มีการนำเครื่องคอมพิวเตอร์เข้ามามีบทบาทอย่างมาก ไม่ว่าจะเป็นงานด้านเอกสาร หรืองานด้านการควบคุมอุตสาหกรรมที่เรามักจะพบได้เสมอ ดังนั้นแนวทางการพัฒนาอุปกรณ์ส่วนใหญ่จึงมีทิศทางเดียวกัน นั่นคือการทำทุกอย่างให้ใช้งานร่วมกันได้ โดยการใช้คอมพิวเตอร์เป็นตัวเชื่อมเข้าด้วยกัน

เครื่องวัดค่าความถ่วงจำเพาะนี้มีหลักการในทำงาน โดยอาศัยการเปลี่ยนแปลงน้ำหนักของวัตถุที่อยู่ในของเหลว โดยการเปลี่ยนแปลงขึ้นอยู่กับแรงพยุงของของเหลวนั้นๆ โดยการรับค่าการเปลี่ยนแปลงของน้ำหนักเข้ามาและทำการแสดงผลออกมา ในส่วนที่รับค่าน้ำหนักเข้ามาประกอบไปด้วย LOAD CELL วงจร Pre-Amplifier ทำหน้าที่ในการขยายสัญญาณและกำหนดค่าสัญญาณที่เหมาะสมให้กับวงจร ADC เพื่อทำการแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอล แล้วส่งไปยังส่วนแสดงผลด้วยการส่งข้อมูลแบบอนุกรม แล้วทำการแสดงผลด้วย คอมพิวเตอร์ หรือ จอ LCD การแสดงผลจะใช้ MCS-51 และคอมพิวเตอร์เป็นตัวทำการประมวลผลที่ได้ออกมาเป็นตัวเลข หรือกราฟแสดงความสัมพันธ์ของการเปลี่ยนแปลงของค่าความถ่วงจำเพาะของของเหลวนั้นๆ ออกมา

1.2 วัตถุประสงค์

1. เพื่อนำความรู้มาประยุกต์ได้
2. เพื่อเพิ่มทักษะ และความเข้าใจเกี่ยวกับเครื่องมือประเภท Displacement
3. เพื่อเรียนรู้ และรู้จักประยุกต์การใช้เครื่องมือ และอุปกรณ์ให้ถูกกับงาน
4. เพื่อการศึกษา และใช้งานโปรแกรมควบคุมด้วยไมโครคอนโทรลเลอร์

1.3 ผลที่คาดว่าจะได้รับ

1. ได้นำความรู้ทางทฤษฎีทักษะจากการทดลองและประยุกต์ใช้ได้จริง
2. ได้เห็นถึงปัญหาที่เกิดขึ้นจากการปฏิบัติงานจริง และสามารถแก้ไขปัญหาที่เกิดขึ้นได้
3. สามารถที่จะเลือกใช้อุปกรณ์ได้เหมาะสมกับงาน
4. ได้สร้างเครื่องมือวัดที่สามารถใช้ประโยชน์ได้จริง
5. ได้เรียนรู้วิธีการทำงานเป็นกลุ่ม

1.4 ขอบเขตของปริญญานิพนธ์

1. สามารถวัดค่าความถ่วงจำเพาะของของเหลวสองชนิดได้ โดยของเหลวที่วัดนั้นต้องมีการแยกชั้นกันอย่างชัดเจน
2. สามารถแสดงผลการวัดค่าความถ่วงจำเพาะเป็นตัวเลข และกราฟการเปลี่ยนแปลงของค่าความถ่วงจำเพาะต่อระยะทางได้บนจอคอมพิวเตอร์
3. สามารถรองรับการทำงานของโปรแกรมของไมโครคอนโทรลเลอร์ได้

1.5 รายละเอียดของปริญญานิพนธ์

ในปริญญานิพนธ์นี้ได้แบ่งเนื้อหาออกเป็น 5 บท และ 2 ภาคผนวก โดยแต่ละบทจะแบ่งเนื้อหาออกเป็น

- บทที่ 1 บทนำ.....จะเป็นการกล่าวถึงเนื้อหาโดยย่อของปริญญานิพนธ์ฉบับนี้
- บทที่ 2 ทฤษฎีที่เกี่ยวข้อง.....จะกล่าวถึงรายละเอียดเบื้องต้น ในส่วนของความหมายของความถ่วงจำเพาะ โปรแกรมและอุปกรณ์ต่างๆ ที่นำมาใช้
- บทที่ 3 หลักการและทฤษฎีที่ใช้ในการออกแบบ.....เป็นเนื้อหาในส่วนของการนำทฤษฎีและหลักการมาทำการออกแบบโครงสร้างขึ้นมา
- บทที่ 4 ผลการทดลอง.....เนื้อหาส่วนใหญ่ เป็นผลการทดลองของเครื่องวัดค่าของความถ่วงจำเพาะที่ทดลองวัดของเหลวชนิดต่างๆ
- บทที่ 5 สรุปผลการทดลอง
- ภาคผนวกที่ 1 โปรแกรมการทำงาน
- ภาคผนวกที่ 2 เอกสารประกอบ

บทที่ 2
ทฤษฎีเบื้องต้นที่เกี่ยวข้อง

2.1 ความหมายของของเหลว (Fluid)

เราเรียกสารที่มีสถานะเป็นของเหลว โดยอาศัยคุณสมบัติต่างๆ เป็นสารที่สามารถไหลได้ หรือเป็นสารที่ต่อต้านความเค้นเฉือนได้น้อย ของไหลจึงผิดรูปได้ง่าย และมีรูปร่างเปลี่ยนไปตามภาชนะบรรจุ อย่างไรก็ตามคุณสมบัติของของเหลวเราพอสามารถสรุปได้ว่า ของเหลวมีระยะห่างระหว่างโมเลกุลมากกว่าของแข็งแต่น้อยกว่าก๊าซ ของเหลวจึงไหลได้เช่นเดียวกับก๊าซ และโมเลกุลแต่ละตัวจะไม่อยู่คงที่ จะเคลื่อนที่ไปมาได้ มีแรงยึดเกาะระหว่างโมเลกุลน้อย และยังไม่ใกล้พอจะสร้างแรงดึงดูดให้โมเลกุลจัดเรียงตัวเป็นโครงผลึกเหมือนของแข็งได้ ดังนั้นในการรวมกลุ่มของโมเลกุลของของเหลวไม่สามารถต้านทานต่อการถูกทำให้ผิดรูป ของเหลวจึงมีรูปร่างตามลักษณะภาชนะที่บรรจุ ของเหลวจัดว่าเป็นของไหลชนิดที่อัดตัวไม่ได้ หรืออัดตัวได้ยาก (Incompressible Fluid) เมื่อมีความดันเปลี่ยนแปลง ปริมาตรจะเกิดการเปลี่ยนไปน้อยมากจนถือได้ว่ามีปริมาตรและความหนาแน่นคงที่ เมื่อมีแรงภายนอกมากระทำในลักษณะอีลาสติกดีฟอร์เมชั่นเท่านั้น แต่ถ้าเกิดมีแรงเฉือนมากระทำเพียงเล็กน้อยต่อของไหลใดๆ ไม่ว่าจะมีความข้นใสเพียงใดมันจะเปลี่ยนรูปไปทันที และจะไม่คืนสู่รูปร่างเดิมถึงแม้ว่าความเค้นเฉือนนั้นจะหมดไปแล้ว

นอกจากนี้ของเหลวยังสามารถแบ่งแยกออกตามแนววิเคราะห์ได้อีก 2 อย่างคือ ของเหลวสมมติหรือที่เรียกว่าของเหลวในจินตนาการ (ideal fluids) และของเหลวจริง (real fluids or practicalfluids)

ของเหลวสมมติ เป็นของเหลวที่ไม่มีความหนืด ไม่มีความตึงผิว และอัดตัวไม่ได้เลย จึงไม่มีแรงเสียดทานเกิดขึ้นในขณะที่เคลื่อนที่ ดังนั้น ความดันของของเหลวจะกระทำในทิศทางที่ตั้งฉากกับภาชนะเสมอ แม้ว่าของเหลวจะเคลื่อนที่อยู่ก็ตาม อย่างไรก็ตาม ในความเป็นจริงจะไม่เป็นเช่นนั้น เพราะของเหลวสมมติเป็นของเหลวที่จินตนาการขึ้นเท่านั้นโดยนักคณิตศาสตร์ เพื่อเป็นการวิเคราะห์ของเหลวในขณะที่เคลื่อนที่ได้ง่ายๆ ด้วยวิธีทางคณิตศาสตร์ เช่น ในการวิเคราะห์ของเหลวที่มีความหนืดต่ำๆ คือ น้ำ หรืออากาศ เป็นต้น ซึ่งค่าที่ได้ก็อาจเกิดการคลาดเคลื่อนไปได้มาก

ของเหลวจริง เป็นของเหลวที่มีคุณสมบัติต่างๆตามธรรมชาติที่มีอยู่ในตัวของของเหลวนั้นและเกิดขึ้นจริงขณะนำไปใช้งาน เช่น คุณสมบัติที่เกี่ยวกับความหนืด ความตึงผิว และสามารถอัดตัวได้ ไม่ว่าจะเป็นของเหลวหรือก๊าซและถ้าของเหลวจริงมีการเคลื่อนที่เกิดขึ้นแล้วก็จะมีแรงเฉือนเกิดขึ้นด้วยแรง เหล่านี้จะมีทิศทางสวนกันกับการเคลื่อนที่ จึงทำให้เกิดแรงต้านทานหรือแรงเสียดทานเกิดขึ้นขณะที่ของเหลวกำลังไหล จึงมีการวิเคราะห์แรงต่างๆ ที่เกิดขึ้น โดยไม่ใช้การวิเคราะห์

เฉพาะแรงที่เกิดจากความดันเพียงอย่างเดียว เหมือนกับของเหลวสมมติ ทั้งๆที่ของเหลวนั้นกำลังมีการเคลื่อนที่อยู่

2.1.1 ความหนาแน่น (Density)

ความหนาแน่นของของเหลว ρ (Rho) เป็นคุณสมบัติอย่างหนึ่งของของไหล ซึ่งหมายถึงปริมาณของของเหลวต่อหนึ่งหน่วยปริมาตรของของเหลวนั้น เขียนเป็นสมการได้ดังนี้

$$\rho = \frac{m}{V} \text{ หรือ } \frac{W}{gV} \tag{2.1}$$

จากคำที่กล่าวว่า ตะกั่วหนักกว่าอะลูมิเนียม เราไม่ได้หมายความว่าตะกั่วทุกชิ้นต้องหนักกว่าอลูนิเนียมเสมอไป แต่จะหมายถึง น้ำหนักเมื่อเปรียบเทียบที่ปริมาตรของโลหะทั้งสองเท่ากัน นั่นคือเมื่อปริมาตรเท่ากัน มวลของตะกั่วย่อมมากกว่ามวลของอะลูมิเนียมเสมอ มวลและปริมาตรของสารจึงเป็นที่มาของปริมาณค่าคงตัวปริมาณหนึ่งเราเรียกว่า *ความหนาแน่น (Density)* ซึ่งเป็นสมบัติเฉพาะตัวของแต่ละสาร

ตารางที่ 2.1 ตารางหน่วยของระบบอังกฤษ และระบบ SI

	ระบบอังกฤษ	ระบบ SI
มวล... m	slug	kg
ปริมาตร... V	ft^3	m^3
ความหนาแน่น... ρ	$slug/ft^3$	kg/m^3

หมายเหตุ

เนื่องจากแต่ละโมเลกุลของสารจะมีมวลที่คงที่ ไม่ว่าจะอยู่ในสถานะของแข็ง ของเหลว หรือว่าก๊าซ ดังนั้น ความหนาแน่นจึงเป็นสัดส่วนกันกับจำนวนโมเลกุลของสารนั้น ในหนึ่งหน่วยปริมาตร เมื่ออุณหภูมิเพิ่มขึ้นจำนวนโมเลกุลในของเหลวจะกระจายออกไป ทำให้มีจำนวนโมเลกุลต่อหน่วยปริมาตรน้อยลง ดังนั้น ความหนาแน่นของของเหลวจะลดลงเมื่ออุณหภูมิเพิ่มขึ้น แต่เมื่อเราใช้ความดันอัดลงไปบนของเหลว จะทำให้มีโมเลกุลจำนวนมากถูกอัดไว้ในหนึ่งหน่วยปริมาตร ดังนั้นความหนาแน่นของของไหลจะเพิ่มมากขึ้น เมื่อความดันของของเหลวเพิ่มขึ้น

2.1.2 น้ำหนักจำเพาะ (Specific Weight)

น้ำหนักจำเพาะของของเหลว γ (Gramma) เป็นคุณสมบัติที่หมายถึงน้ำหนักของของไหลต่อหนึ่งหน่วยปริมาตร หรือหมายถึงแรงอันเกิดจากแรงดึงดูดของโลกที่กระทำกับของไหลในหนึ่งหน่วยปริมาตร เขียนเป็นสมการได้ดังนี้

$$\gamma = \frac{w}{V} \tag{2.2}$$

เมื่อ γ คือน้ำหนักจำเพาะของของเหลว มีหน่วยเป็น N/m^3 , kgf/m^3 , lbf/ft^3 โดยตามปกติน้ำหนักจำเพาะของน้ำ (γ_w) ที่อุณหภูมิ 4 องศาเซลเซียสมีค่าเท่ากับ

$$\gamma_w = 9810 \ N/m^3$$
$$= 1000 \ kgf/m^3$$
$$= 62.4 \ lbf/ft^3$$

น้ำหนักจำเพาะและความหนาแน่นของของเหลวจะมีความสัมพันธ์กันดังสมการต่อไปนี้

$$\gamma = \frac{w}{V} = \frac{m.g}{V} = \rho g$$

$$\gamma = \rho g$$

หรือ

$$\rho = \frac{\gamma}{g} \tag{2.3}$$

เมื่อ g คืออัตราเร่งเนื่องจากแรงดึงดูดของโลก ซึ่งมีค่าเท่ากับ 9.18 lm/s^2 หรือ 32.2 ft/s^2

จะเห็นว่าน้ำหนักจำเพาะจะแปรผันตามความหนาแน่นและอัตราเร่งเนื่องจากแรงดึงดูดของโลก แต่เนื่องจากค่าอัตราเร่งเนื่องจากแรงดึงดูดตามสถานที่ต่างๆ ไม่เท่ากัน ดังนั้นน้ำหนักจำเพาะตามสถานที่ต่างๆ ก็จะมีค่าไม่เท่ากันด้วย และจากที่ได้กล่าวมาแล้วว่าความหนาแน่นเปลี่ยนไปตามอุณหภูมิและความดันนั้น ทำให้ค่าของน้ำหนักจำเพาะขึ้นอยู่กับอุณหภูมิและความดันด้วย

ตามปกติแล้ว ความหนาแน่นของสารโดยทั่วไปขึ้นกับความดันและอุณหภูมิและเนื่องจากน้ำเป็นสารที่มีบทบาทอย่างสูงต่อชีวิตประจำวันของเรา ซึ่งเป็นจำนวนตัวเลขที่เหมาะสมสำหรับใช้เป็นค่าเปรียบเทียบกับความหนาแน่นของสารอื่น ซึ่งเราเรียกอัตราส่วนของความหนาแน่นของสารใดๆ เทียบกับความหนาแน่นของน้ำว่า *"ความถ่วงจำเพาะ (Specific Gravity)"* ของสารนั้น

2.1.3 ความถ่วงจำเพาะ (Specific Gravity)

ความถ่วงจำเพาะ (ถ.พ. หรือ SG.) ซึ่งจะหมายถึง อัตราส่วนของน้ำหนักจำเพาะหรืออัตราความหนาแน่นของของเหลวต่อน้ำหนักจำเพาะของของเหลวมาตรฐาน สำหรับค่าของเหลวจะใช้น้ำที่อุณหภูมิ 4 องศาเซลเซียส (32.9 องศาฟาเรนไฮต์) เป็นของเหลวมาตรฐาน โดยได้อัตราส่วนของมวลของสารต่อมวลของน้ำที่มีปริมาณเท่ากัน กล่าวอีกนัยหนึ่งว่าความถ่วงจำเพาะคือตัวเลข

ที่แสดงว่าสสารชนิดหนึ่งนั้นมีน้ำหนักเบากว่าหรือว่าหนักกว่าน้ำเป็นกี่เท่า เช่นถ้าความถ่วงจำเพาะของสสารมีค่าน้อยกว่าหนึ่งแสดงว่าสสารนั้นมีน้ำหนักเบากว่าน้ำเมื่อปริมาณเท่ากัน ในทางกลับกันถ้าความถ่วงจำเพาะของสสารมีค่ามากกว่าหนึ่ง แสดงว่า สสารนั้นมีน้ำหนักมากกว่าน้ำเมื่อมีปริมาณเท่ากัน

ความถ่วงจำเพาะ = มวลของสสาร ต่อ มวลของน้ำ เมื่อมีปริมาตรเท่ากัน V

เมื่อเอา V ซึ่งมีค่าเท่ากันหารตลอด

$$SG. = \frac{m/V}{m_w/V_w} \quad \text{นั้นคือ} \quad \frac{\rho}{\rho_w}$$

แต่ $$\rho = \frac{\gamma}{g}$$

จะได้ $$SG. = \frac{\gamma}{\gamma_w} = \frac{\rho}{\rho_w} \qquad (2.4)$$

เมื่อ γ_w คือน้ำหนักจำเพาะของของเหลวมาตรฐาน

ρ_w คือความหนาแน่นของของเหลวมาตรฐาน

2.2 เกจวัดความเครียด (Strain Gauge)

เกจวัดความเครียด (Strain Gauge) จัดได้ว่าเป็นทรานดิวเซอร์ชนิดที่ใช้หลักการของการที่จเปลี่ยนแปลงค่าความความต้านทานไฟฟ้าในการตรวจวัดความเครียด (Strain) ที่เกิดขึ้นจากแรงที่มากระทำ ดังนั้น เราจึงสามารถใช้ Strain Gauge ได้อย่างกว้างขวาง สำหรับการวัดน้ำหนัก แรงดัน แรงเชิงกลหรือ การแทนที่ (Displacement) ซึ่ง Strain Gauge นั้น เป็นจะทรานสดิวเซอร์แบบเฉื่อยงาน (Passive Transducer) ซึ่งจำเป็นจะต้องใช้ร่วมกับแหล่งจ่ายภายนอกด้วย โดยทั่วๆไป Strain gauge จะใช้งานร่วมกับวงจรบริดจ์

ความเครียดเป็นปรากฏการณ์ทางวิศวกรรมอย่างหนึ่ง ซึ่งเกิดขึ้นในวัตถุทุกชนิดและเกิดขึ้นทุกเวลา ความเครียดนี้อาจเกิดขึ้นจากแรงภายนอกหรือเกิดขึ้นจากน้ำหนักของตัวมันเอง โดยค่าของความเครียดนี้จะเปลี่ยนแปลงไปตามชนิดของวัตถุและแรงที่มากระทำ แต่อย่างไรก็ตามในการจะวัดความเครียดให้ถูกต้องแน่นอนนั้นทำได้ไม่ง่ายนัก

คำว่า Strain นอกจากจะหมายถึงความเครียดแล้วยังหมายถึงการยืดตัวด้วย โดยความหมายของคำว่าการยืดตัวนั้น จะคล้ายกับความหมายของคำว่า การเปลี่ยนแปลงรูปร่างอย่างเป็นเชิงเส้น (Linear Deformation) การเปลี่ยนแปลงรูปร่างนี้ โดยทั่วไปจะเกิดขึ้นกับแรงภายนอกที่มากระทำ เช่น การยืดตัวของยางเมื่อเกิดมีแรงมากระทำนั้นจะสามารถมองเห็นได้ด้วยตาเปล่า อย่างไรก็ตามความเครียดหรือการยืดตัวของสะพานรับน้ำหนัก (Bridge Strut) เมื่อรถวิ่งผ่านนั้นเราจะไม่สามารถมองเห็นได้ด้วยตาเปล่า ความเครียดหรือการยืดตัวที่กล่าวมาข้างต้นนั้นจะหมายถึงความเครียดรวม

(Total Strain) แต่ในที่เรานิยมนั้น จะแสดงค่าของความเครียดในเทอมของหน่วยความเครียดเฉลี่ย (Average Unit Strain) ซึ่งเป็นจำนวนของความเครียดต่อหน่วยความยาว สำหรับ Strain Gauge นั้นจะใช้สำหรับวัดค่าของหน่วยความเครียด หน่วยของความเครียดโดยทั่วไปจะเป็นหน่วยของ เมตรต่อเมตร หรืออาจจะกล่าวได้ว่าไม่มีหน่วย

ความสำคัญของการวัดความเครียดได้เริ่มมีมากขึ้น ตั้งแต่สมัยศตวรรษที่ 17 เมื่อ Hooke ได้ค้นพบว่า ในวัตถุหลายๆ ชนิดที่มีขนาดจำกัดนั้น จะมีอัตราส่วนคงที่ระหว่างความเค้น (Stress) และความเครียด (Strain) อยู่ โดยความเค้นนั้นอาจจะให้นิยามได้ว่า เป็นแรงภายในต่อหน่วยพื้นที่ ค่าคงที่ ซึ่งเป็นอัตราส่วนระหว่าง ความเค้นและความเครียดตามกฎของ Hooke นี้เรียกว่า ดัชนีการยืดหดตัวของวัสดุ (Modulus of Elasticity of Material) หรืออาจจะเรียกว่า Young's Modulus ดัชนีของการยืดหดตัวของวัสดุนั้นสามารถแสดงได้ด้วยสมการคณิตศาสตร์ดังนี้

$$E = \frac{\sigma_a}{\varepsilon_a} \tag{2.5}$$

เมื่อ E = ดัชนีของการยืดหดตัว (kg/cm^2)

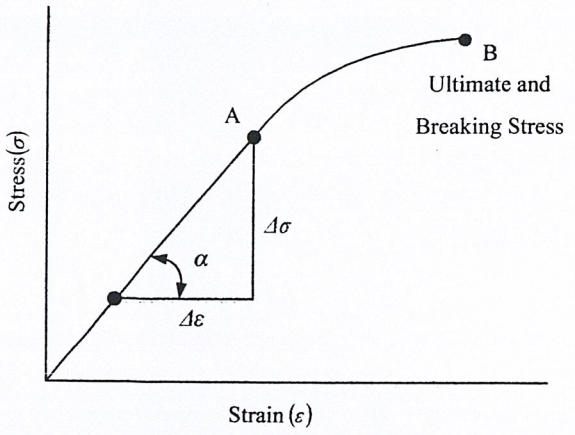
σ_a = ความเค้นที่เกิดขึ้นในแนวแกน (kg/cm^2)

ε_a = ความเครียดที่เกิดขึ้นในแนวแกน (kg/cm^2)

จากความสัมพันธ์ระหว่างความเค้นและความเครียดที่แสดงไว้ข้างต้น โดยเห็นได้ว่าค่าของความเค้นของวัสดุที่มีแรงภายนอกมากระทำนั้น สามารถคำนวณได้โดยการวัดความเครียดของวัตถุนั้น แล้วคูณด้วยดัชนีของการยืดหดตัวของวัตถุนั้น ความเค้นนี้ไม่ได้เป็นปริมาณทางกายภาพที่มีอยู่แต่เป็นปริมาณที่ต้องการหา อย่างไรก็ตาม ค่าของความเค้นนี้จะสามารถหาได้ภายในขอบเขตที่เรากำหนดเท่านั้นทั้งนี้เพราะความสัมพันธ์ระหว่างความเค้นและความเครียดของวัตถุจะเป็นเชิงเส้นในช่วงแรกๆ เท่านั้น ในรูปที่ 2.1 แสดงถึงเส้นโค้ง (Curve) ของความเค้นและความเครียดโดยทั่วๆไป ถ้าหากว่านำเอาโลหะชิ้นหนึ่งมาทำการยืดในเครื่องทดสอบและทำการวัดอย่างเที่ยงตรงแล้ว เราจะได้ความสัมพันธ์ระหว่างความเค้นและความเครียดจะคล้ายกับรูปที่ 2.1 มากจากรูปมีจุดสำคัญที่ควรพิจารณาดังนี้

1. ในช่วงแรกของ Curve นั้นค่อนข้างจะเป็นเส้นตรง นั่นคือความชันมีค่าคงที่
2. เมื่อพ้นจุด A ไปแล้ว ความเครียดจะไม่เป็นสัดส่วนเชิงเส้นกับความเค้น โดยจุด A นี้จะเรียกว่า จุด Proportional Limit จากจุด A ถึงจุด B นั้นวัตถุยังอยู่ในขอบเขตของการยืดตัวและสามารถกลับไปสู่ความยาวเดิมได้เมื่อนำแรงที่มากระทำออกไป

3. จุด B เป็นจุดล้า (yield point) ของโลหะ เมื่อถึงจุด B ความเครียดก็จะยังคงมีอยู่เรื่อยๆ ถึงแม้ว่าจะไม่มีการเพิ่มแรงกระทำ
4. เมื่อวัตถุถูกยืดจนเลยจุด B วัตถุนั้นจะไม่สามารถกลับไปมีค่าความยาวเดิมได้ ถึงแม้ว่าจะนำแรงที่มากระทำออกไปก็ตาม



รูปที่ 2.1 ความสัมพันธ์ระหว่างความเค้นกับความเครียด

Strain Gauge อาศัยหลักการที่ว่า เมื่อเส้นลวดถูกยืดออกไปค่าความต้านทานของมันก็จะมีการเปลี่ยนแปลงไป มาใช้ในการวัดน้ำหนัก,แรงดัน และอื่นๆ Strain Gauge ทั้งชนิดที่ยึดติดแน่นและชนิดที่ไม่ยึดติด (Bonded and Unbonded Type) ทำด้วยเส้นลวดเล็กๆ ขดไปมาและนำไปติดกับวัตถุที่ต้องการตรวจวัดความเครียด เมื่อ Strain Gauge ถูกยืดออก ความยาวของมันก็จะเพิ่มขึ้นในขณะที่พื้นที่หน้าตัดจะลดลงผล ก็คือความต้านทานของเส้นลวดจะเพิ่มขึ้น เนื่องจากความต้านทานของโลหะตัวนำแปรค่าโดยตรงตามความยาว และแปรค่าผกผันกับพื้นที่หน้าตัด ดังนั้นสมการทางคณิตศาสตร์แสดงความสัมพันธ์นี้จะเขียนได้ว่า

$$R = \frac{\rho L}{A} \tag{2.6}$$

เมื่อ R = ความต้านทานของ Strain Gauge

ρ = ค่าคงที่ (ขึ้นอยู่กับชนิดของเส้นลวด)

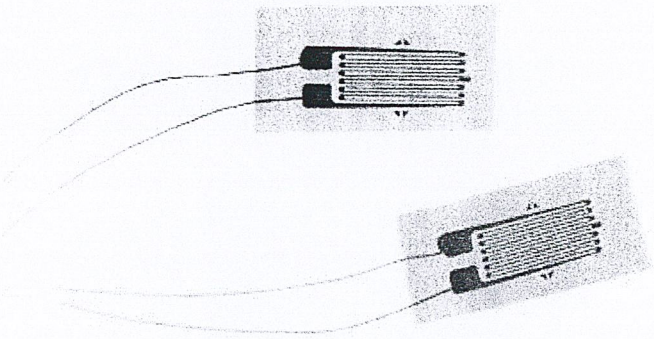
L = ความยาวของเส้นลวด

A = ความยาวของเส้นลวด

ในการใช้ Strain gauge วัดความเครียดของวัตถุนั้น จะต้องพิจารณาถึงคุณสมบัติที่สำคัญสี่ประการของ Strain gauge ดังนี้

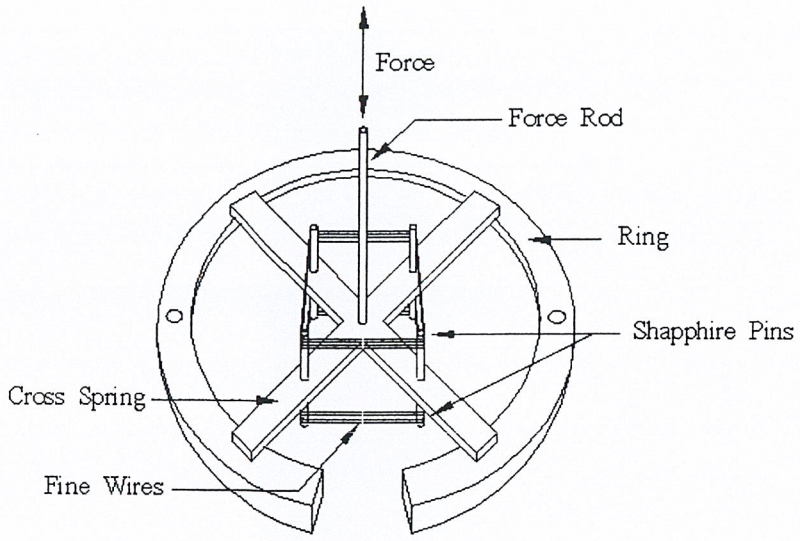
1. โครงสร้างของเกจ
2. ความไวของเกจ
3. วัสดุที่ใช้รองเกจ
4. วิธีการติดเกจ

Strain gauge สามารถจะแบ่งออกได้เป็นสองชนิด คือชนิดยึดติดแน่น (bonded Type) และชนิดยึดติด (Unbonded Type) Strain gauge ชนิดยึดติดแน่นทำจากเส้นลวดเล็กๆ ขดไปขดมาหลายครั้งและยึดติดไว้บนแผ่นกระดาษหรือพลาสติกบางๆ โดยทั่วไปนั้นเส้นลวดนี้จะมีเส้นผ่าศูนย์กลางประมาณ 0.001 นิ้ว และมีค่าความต้านทานประมาณ 120 โอห์ม เนื่องจากแรงเชิงกลที่เกิดมาจากน้ำหนักของเกจจะต้องมีค่าน้อยมากหรือเป็นศูนย์ เมื่อเทียบกับวัตถุที่ต้องการทดสอบ ดังนั้นวัตถุที่ใช้รองจะต้องอ่อนมาก แผ่นรองนี้จะถูกยึดติดกับวัตถุที่ต้องการวัด ซึ่งสามารถวัดความเครียดจากการดึงหรือจากการอัด (tensile or Compressive Strain) ได้ สำหรับ Strain Gauge ชนิดไม่ยึดติดนั้นเส้นลวดเล็กๆจะติดอยู่กับฐานที่เคลื่อนที่ได้ และฐานที่ไม่เคลื่อนที่ ดังแสดงในรูปที่ 2.2 และรูปที่ 2.3 (ก) และ (ข)

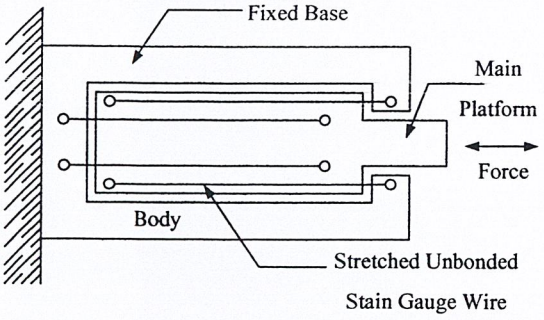


รูปที่ 2.2 Strain Gauge แบบยึดติด (Boned type)

ความไวของ Strain Gauge จะขึ้นอยู่กับการนำของวัตถุ ขนาด โครงสร้าง ความต้านทาน และการนำเอาต์พุตออกมา วัตถุตัวนำที่ใช้ทำเป็น Strain Gauge นั้นอาจจะใช้โลหะผสม หรือสารกึ่งตัวนำโลหะผสมแบบ Nickel – Chrome – Iron ที่ใช้ทำเกจจะมีความไว และอายุการใช้งานสูง โลหะผสมเหล่านี้จะใช้ดีในการวัด Dynamic Strain แต่ไม่เหมาะในการนำไปใช้วัด Static Strain ทั้งนี้เนื่องจากสัมประสิทธิ์ของอุณหภูมิสูง โดยทั่วไปโลหะผสมแบบ Copper – Nickel นั้นจะใช้เมื่ออุณหภูมิต่ำกว่า 500 °F ถึง 600 °F แต่มีความไวน้อยกว่าโลหะผสมแบบ Nickel – Chrome – Iron สำหรับโลหะผสมแบบ Nickel – Chrome นั้นจะมีประโยชน์ในการทำเป็น Strain Gauge เพื่อวัดที่อุณหภูมิสูงๆ



(ก)



(ข)

รูปที่ 2.3 (ก) และ (ข) เป็น Strain Gauge แบบไม่ยึดติด (Unbonded type)

วัสดุที่ใช้เป็นแผ่นรองของ Strain Gauge นั้นจะขึ้นอยู่กับสภาพแวดล้อมของการวัด และวิธีในการติดเกจเข้ากับวัตถุที่ต้องการวัด แผ่นรองแบบกระดาษนั้นจะนิยมใช้ ถ้าทำการวัดที่อุณหภูมิต่ำๆ ถ้าอุณหภูมิค่อนข้างสูงแผ่นรองจะเป็นพวก Epoxy และ Phenolic สำหรับกรณีที่อุณหภูมิที่สูงมากๆ นั้น จะใช้ Mica หรือ โลหะเป็นแผ่นรอง ในการวัด Static Strain เป็นเวลานานๆ นั้นค่าความทนทานของวัสดุที่ใช้เป็นแผ่นรองจะมีความสำคัญมาก

การติด Strain Gauge เข้ากับวัตถุที่ต้องการทดสอบนั้นทำได้สองวิธีก็คือ ใช้ Cement หรือใช้การเชื่อม วิธีการเชื่อมนั้นเป็นวิธีการติดเกจที่ดี ถ้าหากว่าสามารถใช้ได้กับวัตถุที่ต้องการทดสอบ สำหรับ Cement นั้นจะมีอยู่หลายชนิด ทั้งนี้ขึ้นอยู่กับวัสดุที่ใช้เป็นแผ่นรอง และสภาพสิ่งแวดล้อมในขณะทำการวัด เช่น เป็นแบบ Cellulose Adhesives, Epoxy Adhesives, Thermoplastic Resins และ Ceramics ในการใช้ Cement เหล่านี้จำเป็นจะต้องระมัดระวังเรื่องความสะอาด การดูแลรักษาและการติดเข้ากับวัตถุ

การใช้ Strain Gauge ในการวัดนั้นเราจะพิจารณาถึงปริมาณทางกายภาพสองปริมาณ คือค่าความต้านทานของเกจที่เปลี่ยนแปลงและค่าของความยาวที่เปลี่ยนแปลงไป ความสัมพันธ์ระหว่างตัวแปรสองตัวนี้เราจะเรียกว่า Gauge Factor ของ Strain Gauge และสามารถอธิบายได้ด้วยสมการทางคณิตศาสตร์ดังนี้

$$G = \frac{\Delta R/R}{\Delta L/L} \tag{2.7}$$

เมื่อ G = Gauge Factor

R = ความต้านทานเดิม

ΔR = ค่าความต้านทานที่เปลี่ยนแปลงไปจากเดิมเมื่อเกจถูกยืด

L = ความยาวเดิม

ΔL = ค่าความยาวที่เปลี่ยนแปลงไปจากเดิมเมื่อเกจถูกยืด

Gauge Factor ของ Strain Gauge นั้นก็คือ การวัดจำนวนของความต้านทานที่เปลี่ยนแปลงไปเมื่อมีความเครียดโดย Gauge Factor นี้จะเป็นดัชนีของความไวของการยืดตัว (Strain Sensitivity) ของเกจถ้าให้ตัวแปรอื่นๆ มีค่าคงที่ Gauge Factor ที่มีค่าสูงจะหมายถึงว่าเกจนั้นมีความไวสูงและมีสัญญาณเอาท์พุทมาก

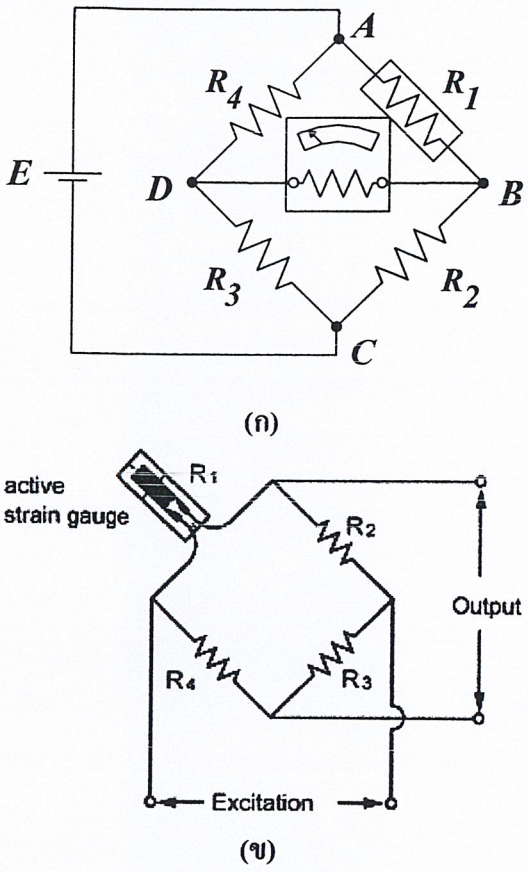
เนื่องจาก Strain Gauge แบบยึดติดแน่นนี้อาศัยหลักการที่ว่า ค่าความต้านทานไฟฟ้าของมันจะเปลี่ยนแปลงไปตามความเครียด ดังนั้น เกจจะต้องนำไปต่อเข้ากับอุปกรณ์ หรือเครื่องมือที่สามารถตรวจวัดค่าความต้านทานจำนวนน้อยๆ ที่เปลี่ยนแปลงไปได้ นั่นก็คือ ตรวจวัดหาค่าของ ΔR นั่นเอง สำหรับ Strain Gauge นั้นค่าของ Gauge Factor และค่าของความต้านทาน R ของมันจะเป็นค่าเราที่ทราบอยู่แล้วจากข้อมูลที่ผู้ผลิตได้ให้มาแล้ว ดังนั้นคงเหลือแต่เพียงค่าของหน่วยของความเครียด $\Delta L/L$ เท่านั้น สมการข้างต้นจึงสามารถเขียนใหม่ได้ว่า

$$\varepsilon = \frac{\Delta L}{L} = \frac{\Delta R/R}{G} \tag{2.8}$$

จากสมการนี้จะเห็นว่าค่าของหน่วยความเครียดจะสามารถคำนวณออกมาได้เมื่อวัดค่าของ ΔR ออกมา ทั้งนี้เพราะค่าของ R และ G นั้นเป็นค่าที่ทราบอยู่ก่อนแล้ว

ค่าของ ΔR ซึ่งเป็นค่าความต้านทานที่มีค่าไม่มากนัก จึงจำเป็นต้องใช้เครื่องมือที่มีความเที่ยงตรงแม่นยำโดยค่าของ ΔR จะหาได้มาจากการวัดค่าของความต้านทานเดิม R และวัดค่าความต้านทานของเกจเมื่อมีแรงมากระทำโดยในที่นี้สมมติให้เป็น R_1 จากนั้นนำค่าที่ได้ทั้งสองมาหักลบ

กันอย่างไรก็ตาม ค่าความแตกต่างระหว่าง R และ R_1 ซึ่งก็คือ ΔR นั้น จะต้องมีความเที่ยงตรงถึง 1/1000 ของ 1% โดยทั่วไป ΔR จะมีค่าประมาณ 0.008 โอห์ม ตามโอห์มมิเตอร์ที่ใช้กันอยู่โดยทั่วไปนั้นไม่สามารถจะนำมาใช้วัดค่าได้ เพราะมีค่าความเที่ยงตรงแม่นยำในการวัดไม่เพียงพอ แต่วีทสโตนบริดจ์(Whetstone Bridge) สามารถนำมาวัดเพื่อหาค่าของ ΔR ได้ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 (ก) และ (ข) เป็นวงจรวีทสโตนบริดจ์อย่างง่าย ๆ

จากรูปที่ 2.4 สมมติว่าให้ R_1 เป็นค่าความต้านทานของ Strain Gauge ส่วน R_3 และ R_4 เป็นค่าความต้านทานในแขนสัดส่วน (Ratio Arms) R_2 เป็นค่าความต้านทานคงที่มาตรฐาน ซึ่งถ้าวงจรบริดจ์อยู่ในสมดุลคือไม่มีกระแสไหลผ่านกัลวานอมิเตอร์ (Galvanometer) จะได้ความสัมพันธ์ระหว่างความต้านทานในวงจร ดังนี้

$$\frac{R_1}{R_2} = \frac{R_4}{R_3} \tag{2.9}$$

หรือ

$$R_1 = \frac{R_4}{R_3} R_2 \tag{2.10}$$

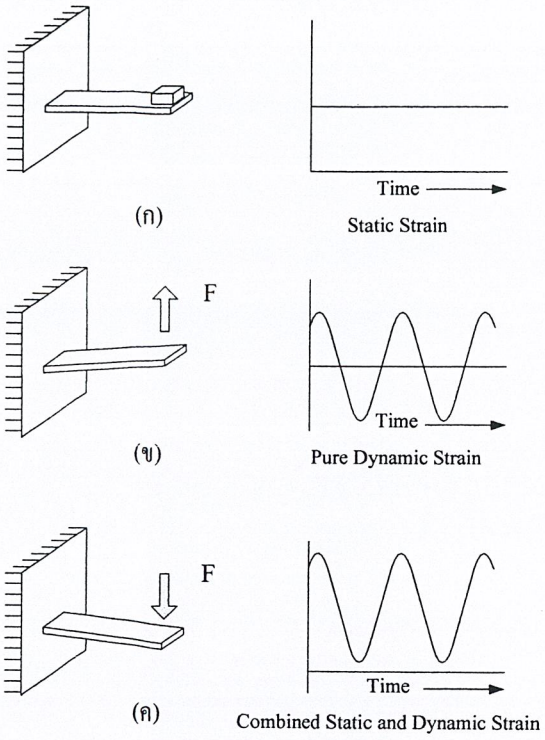
เมื่อมีแรงมากระทำต่อวัตถุที่ Strain Gauge นี้ติดอยู่ ค่าความต้านทานของ Strain Gauge จะเปลี่ยนไปเล็กน้อย ในที่นี้สมมติให้ค่าความต้านทาน R_1 เปลี่ยนไปเป็น $R_1 + \Delta R_1$ วงจรบริดจ์อยู่ในสภาพไม่สมดุลและมีกระแสไหลผ่านกัลวานอมิเตอร์ ดังนั้นเพื่อทำให้วงจรบริดจ์อยู่ในสภาพสมดุลอีกครั้งจึงต้องปรับค่า R_4 และ R_3 ใหม่ โดยเราจะสมมติให้ R_4' และ R_3' เป็นค่าความต้านทานใหม่ในแขนสัดส่วนที่ทำให้วงจรบริดจ์อยู่ในสภาพสมดุลอีกครั้งหนึ่ง จะได้สมการแสดงความสัมพันธ์จะเป็นดังนี้

$$R_1 + \Delta R_1 = \frac{R_4'}{R_3'} R_2 \tag{2.11}$$

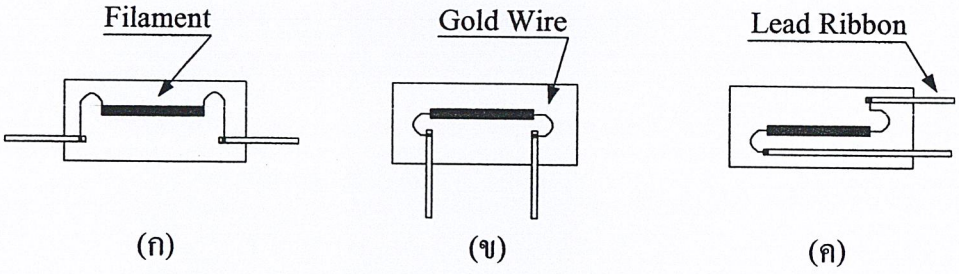
เนื่องจากเราทราบค่า R_1 อยู่แล้ว ดังนั้นจึงเป็นการง่ายในการหาค่า ΔR_1

ความเครียดส่วนมากที่ทำการวัดนั้นจะเป็นแบบ Static Strain แต่ความเครียดอีกแบบหนึ่งคือ Dynamic Strain ก็มีความสำคัญเช่นกัน Dynamic Strain นั้นเป็นความเครียดที่ขนาดของมันจะเปลี่ยนแปลงในช่วงเวลาสั้นๆ ในกรณีที่ความเครียดของวัตถุบางอย่างเปลี่ยนแปลงค่าไปเรื่อยๆและหลายๆไซเคิลต่อนาทีนั้น การวัดค่าความเครียดนี้โดยใช้วงจรบริดจ์จะทำให้ยากหรืออาจจะใช้การไม่ได้เลย การวัดค่าของ Dynamic Strain ให้ถูกต้องแน่นอนนั้นจะทำได้ยาก และอีกประการหนึ่งการทดสอบเพื่อวัดค่าการเปลี่ยนแปลงทาง Dynamic ของวัตถุสิ่งเดียวกัน จำนวนสองครั้งนั้นยากที่จะได้รับผลลัพธ์ที่เท่ากัน รูปที่ 2.5 แสดงถึง Static Strain และ Dynamic Strain ในอุดมคติ ในรูปที่ 2.5 (ก) นั้น แสดงถึง Dynamic Strain ที่ค่าของมันนั้นเปลี่ยนแปลงน้อยมากหรือไม่เปลี่ยนแปลงเลย สำหรับรูปที่ 2.5 (ข) เป็น Dynamic Strain ที่ค่า Oscillate หรืออีกนัยหนึ่งค่าความเครียดนั้นแปรค่าบวกและลบต่อเนื่องตลอดตามเวลา (แรงดึงและแรงอัด) โดยมีขนาดเท่ากันตลอด ส่วนรูปที่ 2.5 (ค) นั้นเป็นผลรวมของรูปที่ 2.5 (ก) และ 2.5 (ข)

สารกึ่งตัวนำที่ใช้ทำเป็น Strain Gauge จะมีความไวสูง เช่น Strain Gauge ที่ทำจาก Silicon ที่โค้งงอได้นั้น จะนิยมใช้ในทางปฏิบัติมาก เพราะมีความคงที่เหมือน Strain Gauge ที่ทำจากลวดโลหะ แต่มีระดับของเอาต์พุตสูงกว่าแบบโลหะ Strain Gauge แบบนี้ สามารถทำการตรวจวัดการเปลี่ยนแปลงความยาวได้ถึง 10^{-6} Inch/Inch โดยในรูปที่ 2.6 จะแสดงถึงสารกึ่งตัวนำที่ใช้ทำเป็น Strain Gauge ที่วัดได้ละเอียดมาก



รูปที่ 2.5 Static Strain และ Dynamic Strain ในอุดมคติ



รูปที่ 2.6 ลักษณะของ Semiconductor ที่ใช้เป็น Strain Gauge

2.2.1 วงจรสเตรนเกจ (Strain Gauge Circuit)

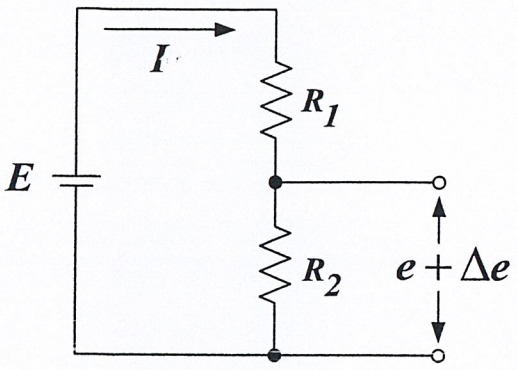
ในการหาค่าการเปลี่ยนแปลงความต้านทานในเกจ จะใช้วงจรไฟฟ้าเพื่อช่วยในการวัด เช่น วงจรแบ่งแรงดัน (Voltage Divider) หรือวงจรวีทสโตนบริดจ์ (Wheatstone Bridge) ดังรูปที่ 2.7 และรูปที่ 2.8

2.2.2 วงจรแบ่งแรงดัน (Voltage Divider)

ในรูปที่ 2.8 คือวงจรแบ่งแรงดันหรือเรียกว่า วงจร Potentiometer ซึ่งเรารู้จักกันในชื่อวงจร Ballast ซึ่งจะประกอบไปด้วยแหล่งจ่ายแรงดันไฟฟ้า DC (E) หรือในบางครั้งเราจะใช้แรงดันไฟฟ้า

AC ขึ้นอยู่กับการประยุกต์ใช้งาน แรงดันไฟฟ้านี้จะตกคร่อมอยู่ระหว่างตัวต้านทาน R_1 และ R_2 ที่มีตัวใดตัวหนึ่ง หรือทั้งสองตัวเป็น Strain Gauge โดยแรงดันไฟฟ้าทางด้านเอาต์พุต (e) ที่เกิดขึ้นตกคร่อม R_2 ซึ่งกำหนดให้เป็นอินพุทอิมพิแดนซ์สูง (High Input Impedance) ที่ใช้เพื่อวัดแรงดันจะเท่ากับ

$$e = E\frac{R_2}{R_1 + R_2} \tag{2.12}$$



รูปที่ 2.7 วงจร Potentiometer

การเปลี่ยนแปลงแรงดันเอาต์พุต Δe เกิดจากการเปลี่ยนแปลงของ ΔR_1 และ ΔR_2 ในตัวต้านทาน R_1 และ R_2 เมื่อกำหนดให้แหล่งจ่ายมีความต้านทานเป็นศูนย์และโหลดมีความต้านทานเป็นอนันต์สามารถเขียนได้ว่า

$$e + \Delta e = E\left[\frac{R_2 + \Delta R_2}{R_1 + \Delta R_1 + R_2 + \Delta R_2}\right] \tag{2.13}$$

หรือ

$$\Delta e = \left[\frac{R_2 + \Delta R_2}{R_1 + \Delta R_1 + R_2 + \Delta R_2} - \frac{R_2}{R_1 + R_2}\right]E \tag{2.14}$$

เมื่อทำให้อยู่ในรูปแบบที่ง่ายขึ้นจะได้ว่า

$$\Delta e = E\frac{R_1}{R_2}\frac{\left[\frac{\Delta R_2}{R_2}-\frac{\Delta R_1}{R_1}\right]}{\left[1+\frac{R_1}{R_2}\right]^2\left[1+\frac{\frac{\Delta R_1}{R_2}+\frac{\Delta R_2}{R_2}}{1+\frac{R_1}{R_2}}\right]} \tag{2.15}$$

จากสมการ (2.15) จัดรูปใหม่จะได้

$$1+\frac{\frac{\Delta R_1}{R_2}+\frac{\Delta R_2}{R_2}}{\left(1+\frac{R_1}{R_2}\right)} = 1+\frac{\left(\frac{R_1}{R_2}\frac{\Delta R_1}{R_1}\right)+\frac{\Delta R_2}{R_2}}{\left(1+\frac{R_1}{R_2}\right)}$$

แทนค่า $R_1/R_2 = a$ ดังนั้นสมการ (2.15) สามารถเขียนใหม่ได้

$$\Delta e = E\left[\frac{a}{(1+a)^2}\left(\frac{\Delta R_2}{R_2}-\frac{\Delta R_1}{R_1}\right)(1-N)\right] \tag{2.16}$$

เมื่อ N คือเทอมที่ไม่เป็นเชิงเส้น (Non-Linear) คือ

$$N = \left[1-\frac{1}{1+\frac{1}{(1+a)}\left(a\frac{\Delta R_1}{R_1}+\frac{\Delta R_2}{R_2}\right)}\right] \tag{2.17}$$

ถ้า R_2 คือ Strain Gauge และ R_1 คือตัวต้านทานที่ถูกกำหนดค่าไว้แล้ว ดังนั้น $\Delta R_1 = 0$ (ซึ่งจะมี Active Gauge เพียงตัวเดียวเท่านั้น) หากไม่พิจารณาถึงเทอมที่ไม่เป็นเชิงเส้นของ N ในสมการที่ (2.16) จะได้ว่า

$$\Delta e = \frac{a}{(1+a)^2}\frac{\Delta R_2}{R_2}E \tag{2.18}$$

แทนด้วยเกจแฟคเตอร์ (G) ซึ่งก็คือ การเปลี่ยนแปลงความต้านทาต่อหนึ่งหน่วยความต้านทานต่อหนึ่งหน่วยความเครียดจะได้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

$$\Delta e = \frac{a}{(1+a)^2} G\varepsilon\varepsilon \tag{2.19}$$

ค่าความไวต่อแรงดันไฟฟ้า (Voltage Sensitivity) ของวงจรหรือ Se จะเท่ากับ

$$Se = \frac{\Delta e}{\varepsilon} = \frac{a}{(1+a)^2} GE \tag{2.20}$$

เมื่อ $R_1 = R_2$

$$Se = \frac{1}{4} GE \tag{2.21}$$

จากสมการ (2.16) จะพบว่าช่วงใช้งานของวงจร Potentiometer ถูกจำกัดโดยค่าที่ยอมรับได้ของความไม่เป็นเชิงเส้นที่ถูกกำหนดโดยเทอม N ซึ่งจะขึ้นอยู่กับระดับความเครียดและอัตราส่วนความต้านทาน a ตัวประกอบที่สำคัญอีกอย่างหนึ่ง ก็คือความไวต่อแรงดันไฟฟ้าของวงจรหรือ Se ซึ่งเท่ากับอัตราส่วนความเปลี่ยนแปลงในแรงดันเอาต์พุต Δe ต่อความเครียด ε ดังสมการ (2.20) จากสมการ (2.16), (2.20) และ (2.21) ที่กล่าวมาก็จะพบว่าค่าความคาดเคลื่อนที่เกิดจากความไม่เป็นเชิงเส้นของวงจรจะมีความสัมพันธ์กับการวัดความเครียดน้อยมาก โดยทั่วไปจะอยู่ในระดับ 1000 microstrain แต่ว่าถ้า Strain Gauge ที่มีเกจเฟคเตอร์เท่ากับ 2 ได้ถูกนำมาใช้แทน R_2 ค่าของความคลาดเคลื่อนจะมีค่าประมาณ 2% ถึง 10% ในช่วงความเครียด ซึ่งจะขึ้นอยู่กับค่า a ของวงจร เมื่อค่า a มีค่าลดลงจาก 9 ถึง 2 ช่วงการวัดใช้งานจะลดลงจาก 10% ถึง 20% ของความเครียด ดังนั้นช่วงการวัดของวงจร Potentiometer จะมีค่าเพียงพอสำหรับการวัดความเครียด โดยทั่วไปในย่านของการยืดหยุ่นของวัตถุ ในการออกแบบใช้งานนี้ แรงดันไฟฟ้าเลี้ยง E จะถูกกำหนดจากค่ากระแสที่เรายอมรับได้สูงสุด ซึ่งสามารถไหลผ่านเกจได้ ซึ่งสามารถเขียนแทนด้วยสมการ (2.20) ใหม่ได้

$$\begin{aligned} Se &= \frac{E}{(1+a)} \frac{a}{(1+a)} G \\ &= IR_2 \frac{a}{(1+a)} G \end{aligned} \tag{2.22}$$

เมื่อ I = กระแสที่ไหลผ่าน R_1 และ R_2

จะเห็นว่าเมื่อกำหนดค่าของกระแสเกจและความต้านทานของเกจมาให้ ความไวจะเพิ่มขึ้นเนื่องจากค่า a ซึ่งต้องใช้แรงดันไฟฟ้ากระตุ้นที่สูงขึ้น ซึ่งในสภาวะนี้จะลดความไม่เป็นเชิงเส้นลง

61904

ข้อเสียของวงจร Potemtiometer นี้คือมักจะแสดงค่าแรงดันขณะเริ่มต้นภายใต้สภาวะที่มีค่าความเครียดเป็นศูนย์ ซึ่งโดยปกติแล้วจะต้องแสดงค่าที่เป็นศูนย์แต่ค่าแรงดันไฟฟ้าจะเปลี่ยนแปลงเมื่อเกิดความเครียดเท่านั้น แต่ข้อบกพร่องนี้สามารถแก้ไขได้โดยการใช้วงจรวีทสโตนบริดจ์

2.2.3 วงจรวีทสโตนบริดจ์ (Wheatstone Bridge)

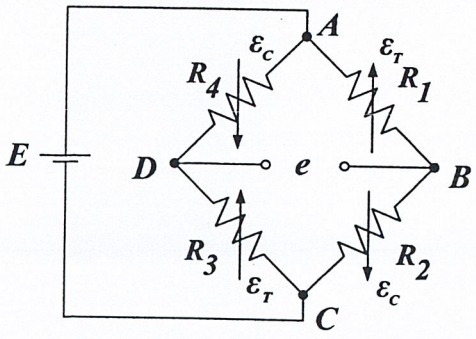
การวัดความเครียดนั้นโดยปกติแล้วจะใช้วงจรวีทสโตนบริดจ์แบบ 4 แขน โดยที่ระหว่างจุด B และ D มีความต่างศักย์เท่ากับ

$$\begin{aligned} e &= E_{BA} - E_{DA} \\ &= E\left[\frac{R_1}{R_1+R_2} - \frac{R_4}{R_3+R_4}\right] \\ &= E\left[\frac{R_1R_3 - R_2R_4}{(R_1+R_2)(R_3+R_4)}\right] \end{aligned} \tag{2.23}$$

แรงดันไฟฟ้าทางด้านเอาท์พุต (e) เท่ากับศูนย์เมื่อ $R_1R_3 = R_2R_4$ ซึ่งในสภาวะนี้จะถือว่าบริดจ์มีความสมดุล ดังนั้นแขนของบริดจ์จึงสามารถจะปรับได้ตามต้องการ เพื่อให้ได้แรงดันที่เอาต์พุตเท่ากับศูนย์เมื่อไม่มีความเครียด

เมื่อพิจารณาที่ตัวต้านทานทั้ง 4 ของบริดจ์เป็น Active Strain Gauge และความต้านทานมีการเปลี่ยนแปลงไป ΔR_1, ΔR_2, ΔR_3 และ ΔR_4 เมื่อใช้งานในสภาวะที่มีความเครียด ดังนั้น

$$e = E\left[\frac{(R_1+\Delta R_1)(R_3+\Delta R_3) - (R_2+\Delta R_2)(R_4+\Delta R_4)}{(R_1+\Delta R_1+R_2+\Delta R_2)(R_3+\Delta R_3+R_4+\Delta R_4)}\right] \tag{2.24}$$



รูปที่ 2.8 วงจรวีทสโตนบริดจ์แบบ 4 แขน

หากไม่พิจารณาเทอมของอนุพันธ์อันดับสองที่ไม่เป็นเชิงเส้น และกำหนดให้สภาวะเริ่มต้นของเกจมีความสมดุล จะได้

$$e = E\left[\frac{\Delta R_1}{R_1} - \frac{\Delta R_2}{R_2} + \frac{\Delta R_3}{R_3} - \frac{\Delta R_4}{R_4}\right]\frac{R_1 R_2}{(R_1 + R_2)^2} \tag{2.25}$$

หรือ

$$e = \frac{a}{(1+a)^2}\left[\frac{\Delta R_1}{R_1} - \frac{\Delta R_2}{R_2} + \frac{\Delta R_3}{R_3} - \frac{\Delta R_4}{R_4}\right]E \tag{2.26}$$

เมื่อ

$$a = \frac{R_1}{R_2} = \frac{R_3}{R_4}$$

ถ้ากำหนดให้เกจแฟคเตอร์ของเกจเท่ากับ G_1, G_2, G_3 และ G_4 และความเครียดที่เกิดมีค่าเท่ากับ ε_1, ε_2, ε_3 และ ε_4 ดังนั้น

$$e = \frac{a}{(1+a)^2}[G_1\varepsilon_1 - G_2\varepsilon_2 + G_3\varepsilon_3 - G_4\varepsilon_4]E \tag{2.27}$$

คุณสมบัติทางพีชคณิตของค่าความเครียดทั้ง 4 ในบริดจ์จะเพิ่มขึ้น เมื่อใช้บริดจ์ในการวัดความเครียดบ่อยครั้ง โดยเฉพาะการใช้ Strain Gauge เป็นทรานสดิวเซอร์

2.2.4 ชนิดของวงจรบริดจ์ที่นำมาใช้งาน

(ก) Quarter Bridge

บริดจ์ชนิดนี้จะมีแขนข้างหนึ่งเป็น Active Strain Gauge คือ R_1 โดยส่วนแขนอีก 3 แขนที่เหลือจะเป็นตัวต้านทานแบบค่าคงที่ ซึ่งจะมีค่าที่เท่ากันและเป็นค่าเริ่มต้นเท่ากับ R_1 ดังนั้น $R_1 = R_2 = R_3 = R_4$ และ $\Delta R_2 = \Delta R_3 = \Delta R_4 = 0$ จากสมการ (2.26)

$$e = \frac{EG\varepsilon_1}{4} = \frac{E}{4} = \frac{\Delta R_1}{R_1} \tag{2.28}$$

(ข) Half Bridge

บริดจ์ชนิดนี้มี 2 แขนที่เป็น Active Strain Gauge คือ R_1 และ R_2 ซึ่งตัวหนึ่งจะรับการดึงและอีกตัวหนึ่งรับการอัดส่วน 2 แขนที่เหลือเป็นตัวต้านทานแบบค่าคงที่ ซึ่งจะมีค่าที่เท่ากัน แรงดันไฟฟ้าเอาต์พุตที่ได้มีค่าเท่ากับ

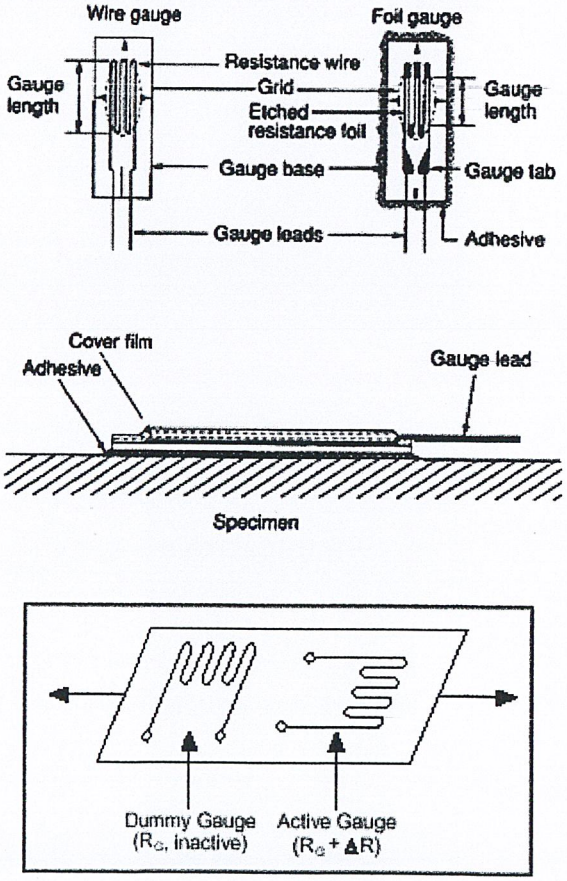
$$e = \frac{a}{(1+a)^2}[G_1\varepsilon_1 - G_2\varepsilon_2]E \tag{2.29}$$

เมื่อ $R_1 = R_2$, $G_1 = G_2$, และ $\varepsilon_1 = -\varepsilon_2 = \varepsilon$ ดังนั้น

$$\frac{\Delta R_1}{R_1} = -\frac{\Delta R_1}{R_2}$$

$$e = \frac{EG\varepsilon}{2} = \frac{E}{2}\frac{\Delta R_1}{R_1} \tag{2.30}$$

ดังนั้นความไวของ Half Bridge จึงเป็น 2 เท่าของแบบ Quarter Bridge ถ้าเกจ R_1, R_2 ถูกวางในทิศทางตั้งฉากซึ่งกันและกันในรูปแบบของพัวซอง (Poisson's Configuration) ดังรูปที่ 2.8



รูปที่ 2.9 การติดตั้ง Strain Gauge ตามเงื่อนไข Poisson แบบ Half Bridge

ดังนั้น

$$e = \frac{a}{(1+a)^2}[G_1\varepsilon_1 - G_2(-0.3\varepsilon_1)]E \tag{2.26}$$

$$= 0.325EG\varepsilon \tag{2.27}$$

(ค) Full Bridge

แขนทั้งสี่ของบริดจ์ประเภทนี้เป็น Active Gauge ซึ่ง R_1 และ R_3 เป็นแขนที่รับการดึง ส่วน R_2 และ R_4 เป็นแขนที่รับการอัด โดยเมื่อ $R_1 = R_2$, $R_3 = R_4$, $G_1 = G_2 = G_3$ และเท่ากับ G_4 และ $\varepsilon_1 = -\varepsilon_2 = \varepsilon_3 = -\varepsilon_4 = \varepsilon$ ดังนั้นเอาท์พุทที่ได้จึงเท่ากับ

$$e = EG\varepsilon = E\frac{\Delta R_1}{R_1} \tag{2.28}$$

ถ้า Full Bridge มีการจัดการแบบพัวซองดังรูปที่ 2.8 แรงดันไฟฟ้าเอาท์พุทจะเท่ากับ

$$e = \frac{E}{4}\left[G_1\varepsilon_1 - G_2\left(-0.3\varepsilon_1\right) + G_3\varepsilon_3 - G_4\left(-0.3\varepsilon_1\right)\right] \tag{2.29}$$

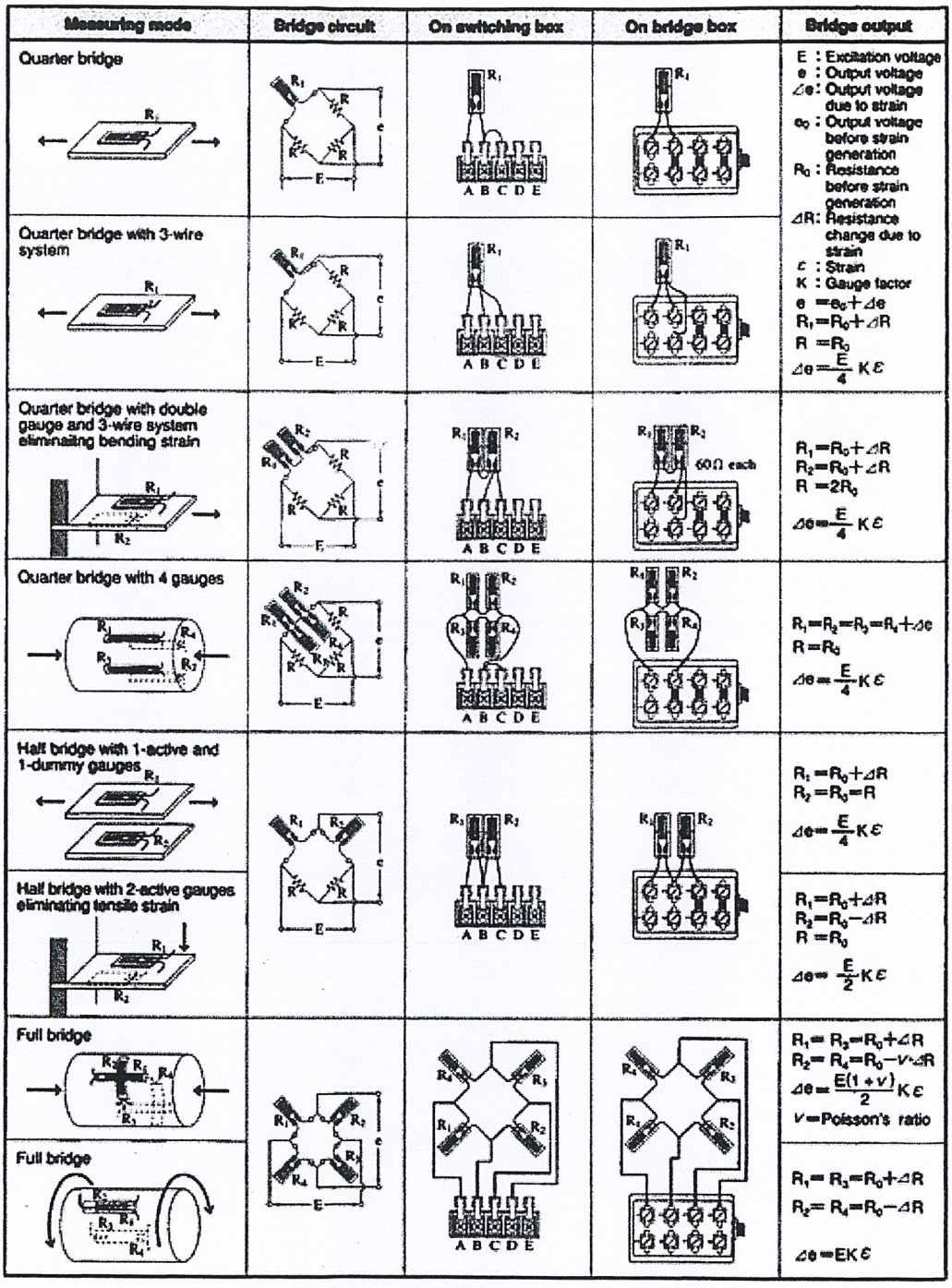
ถ้าเกจถูกกำหนดให้มีลักษณะเหมือนกันหมด $\left(G_1 = G_2 = G_3 = G_4 = G\right)$

$$e = 0.65EG\varepsilon = 0.65E\frac{\Delta R_1}{R_1} \tag{2.30}$$

ความไวต่อแรงดันไฟฟ้าของวงจรหรือ Se ภายใต้สภาวะที่แรงดันกระตุ้นมีค่าคงที่จะสัมพันธ์กับค่าเกจแฟคเตอร์ในทุกกรณี ตามความสัมพันธ์

$$Se = \frac{e}{\varepsilon} = \frac{a}{\left(1+a\right)^2}EG \tag{2.31}$$

และความไวจะมีค่าสูงสุดเมื่อ $a = 1$ เมื่อแขนทุกแขนในบริดจ์มีค่าเท่ากัน



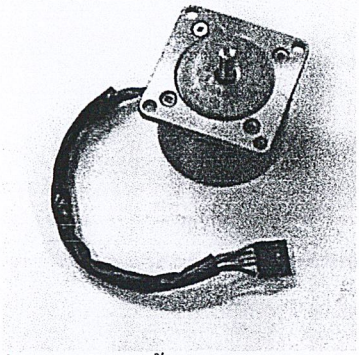
รูปที่ 2.10 ลักษณะการติดตั้ง Strain Gauge ในแบบต่างๆ

2.3. สเต็ปปิ้งมอเตอร์ (Stepping Motor)

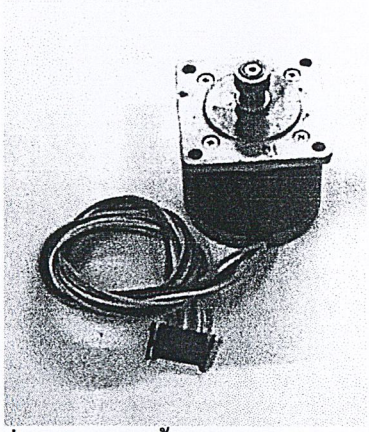
สเต็ปปิ้งมอเตอร์เป็นมอเตอร์ที่ขับเคลื่อนด้วยพัลส์ ลักษณะการขับเคลื่อนเป็นจะหมุนรอบแกนได้ 360 องศา มีลักษณะไม่ต่อเนื่องแต่ว่ามีลักษณะเป็นสเต็ป โดยแต่ละสเต็ปจะขับเคลื่อนได้ 1, 1.5, 1.8 หรือ 2 องศา ก็แล้วแต่ละโครงสร้างของมอเตอร์ และลักษณะการที่นำมอเตอร์ไปใช้จะเป็นงานที่ต้องการตำแหน่งที่แม่นยำ จึงเหมาะแก่การนำมาใช้ในการควบคุมการเคลื่อนที่

ในปัจจุบันนี้จะเห็นได้ว่าสเต็ปปิ้งมอเตอร์ได้ถูกใช้อยู่มากตัวอย่างเช่น ใน Printer, X-Y-Z Plottor Robot Arm เพราะสเต็ปปิ้งมอเตอร์จะมีข้อดีมากกว่าดีซีมอเตอร์ คือ สเต็ปปิ้งมอเตอร์มีการควบคุมแบบ Open Loop ทำให้ง่ายต่อการควบคุม และเรายังสามารถที่จะรู้ตำแหน่งของมอเตอร์ได้ตลอดเวลาอย่างแน่นอนแม่นยำ ดังนั้น งานที่ต้องการควบคุมตำแหน่งที่แน่นอน จึงมักเลือกใช้

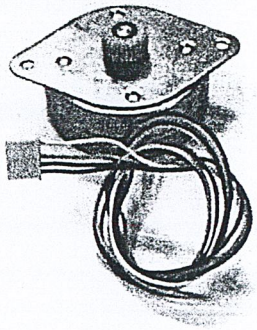
2.3.1 ชนิดและโครงสร้างของสเต็ปปิ้งมอเตอร์



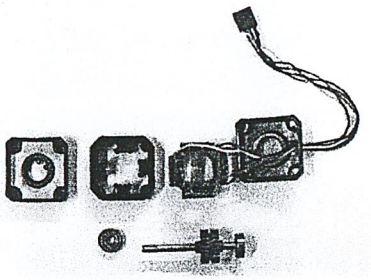
รูปที่ 2.11 สเต็ปปิ้งมอเตอร์แบบ 5 สาย



รูปที่ 2.12 สเต็ปปิ้งมอเตอร์แบบ 6 สาย



รูปที่ 2.13 สเต็ปปิ้งมอเตอร์แบบไบโพลาร์



รูปที่ 2.14 แสดงภาพถ่ายโครงสร้างของมอเตอร์

สเต็ปปิ้งมอเตอร์ที่พบในปัจจุบันมี 3 ลักษณะดังนี้

1. แบบแม่เหล็กถาวร (Permanent M..gnet_PM)

สเต็ปมอเตอร์แบบ PM จะมีสเตเตอร์ (Stator) ที่พันขดลวดไว้หลายๆโพล โดยจะมีโรเตอร์ (Rotor) เป็นรูปทรงกระบอกฟันเลื่อย และตัวโรเตอร์ทำด้วยแม่เหล็กถาวร เพื่อป้อนไฟกระแสตรงให้กับขดสเตเตอร์ จะทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์ ทำให้มอเตอร์หมุน มอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้โรเตอร์หยุดอยู่กับที่ แม้จะไม่ได้ป้อนไฟเข้าขดลวด

2. แบบแปรค่ารีลักแตนซ์ (Variable Reluctance_VR)

สเต็ปปิ้งมอเตอร์แบบ VR จะมีการหมุนโรเตอร์ได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟให้ โรเตอร์ทำมาจากสารเฟอร์โรแมกเนติก กำลังอ่อน มีลักษณะเป็นฟันเลื่อยรูปทรงกระบอก โดยจะมีความสัมพันธ์โดยตรงกับจำนวนโพลในสเตเตอร์ แรงบิดที่เกิดขึ้นจะไปหมุนโรเตอร์ ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนท์ต่ำที่สุด ตำแหน่งที่จะเกิดแน่นอน และมีเสถียรภาพแต่จะเกิดขึ้นได้หลายๆจุด ดังนั้น เมื่อป้อนไฟฟ้าเข้าขดลวดต่างๆ ในมอเตอร์แตกต่างขดกันไป ก็จะทำให้มอเตอร์หมุนไปตำแหน่งต่างๆกัน โรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM

3. แบบผสม (Hybrid_H)

สเต็ปมอเตอร์แบบ H จะเป็นลูกผสมของ VR กับ PM โดยจะมีสเตเตอร์คล้ายกับที่ใช้ใน VR โรเตอร์มีหมวกหุ้มปลาย ซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูง โดยการควบคุมขนาดรูปร่างของหมวกแม่เหล็กอย่างดี ทำให้ได้มุมการหมุน และครั้งน้อยและแม่นยำ ข้อดีก็คือ ให้แรงบิดสูงและมีขนาดกระทัดรัด และให้แรงฉุดยึดโรเตอร์นิ่งกับที่ตอนไม่จ่ายไฟ

2.3.2 หลักการทำงานของสเต็ปปิ้งมอเตอร์

ชื่อของเจ้าสเต็ปปิ้งมอเตอร์ นั่นก็ค่อนข้างที่จะบ่งบอก ถึงคุณสมบัติของตัวมันเองได้ดีอยู่แล้วว่าเป็นมอเตอร์ที่สามารถเคลื่อนที่ในลักษณะที่เป็นทีละสเต็ป ซึ่งจะต่างจากมอเตอร์ทั่วไปที่มีการเคลื่อนที่ของมันจะเป็นแบบต่อเนื่อง และสเต็ปปิ้งมอเตอร์ยังสามารถควบคุมการทำงานที่ง่าย

ด้วยการต่อร่วมกับอุปกรณ์ภายนอกเพียงเล็กน้อย ดังนั้นเราจะเห็นว่าสเต็ปปิ้งมอเตอร์ถูกนำไปใช้งานอย่างกว้างขวาง ไม่ว่าจะเป็นงานทางด้าน Robot, แขนกล, Plotter, หรือว่างานใดๆ ที่ต้องใช้การเคลื่อนที่ในลักษณะแบบสเต็ป ในการเรียนรู้หลักการทำงานของสเต็ปปิ้งมอเตอร์ ก็ย่อมก่อให้เกิดประโยชน์แก่เราให้สามารถนำเอาสเต็ปปิ้งมอเตอร์ไปประยุกต์ใช้งานได้

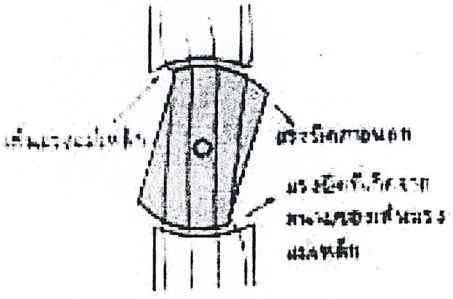
การควบคุม สเต็ปปิ้งมอเตอร์ใช้ 2 วิธีในการ Implement

- ใช้ Computer หรือ บอร์ด Control ควบคุม จะต้องมีการ Interface กับ บอร์ด ซึ่งจะต้องมี Software Control ซึ่งสามารถควบคุมความแม่นยำได้สูง และสามารถประยุกต์ได้กับทุกแกนและทุกสเต็ป แต่ผู้ใช้ต้องสามารถเขียนโปรแกรมติดต่อกับมอเตอร์ได้

- ใช้วงจรดิจิตอลควบคุมค่อนข้างจะ Fix ตำแหน่ง ความเร็ว และบางส่วนโดยอย่างมาก แต่สามารถสร้างได้ง่าย และสะดวกสำหรับผู้เริ่มต้น

2.3.3 โครงสร้างและการทำงานของสเต็ปปิ้งมอเตอร์

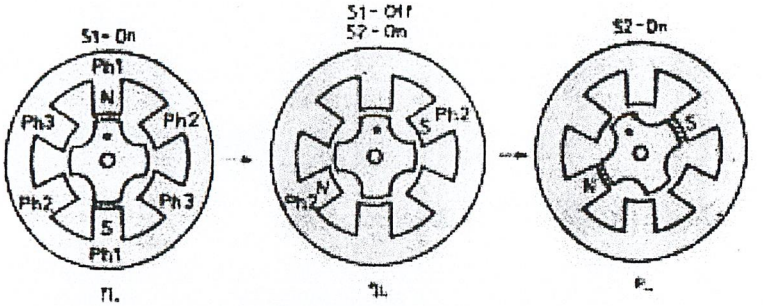
เนื่องจากสเต็ปปิ้งมอเตอร์นี้ มีโรเตอร์เป็นเหล็กอ่อน ซึ่งมีคุณสมบัติพยายามปรับตัวเองให้อยู่ในแนวที่เส้นแรงแม่เหล็กผ่านมากที่สุด ดังในรูปที่ 2.15 เมื่อเกิดเส้นแรงแม่เหล็กขึ้นที่สเตเตอร์ตัดผ่านโรเตอร์ ตัวโรเตอร์ก็จะพยายามปรับตัวเองเพื่อให้เส้นแรงแม่เหล็กตัดผ่านตัวเองมากที่สุด โดยการหมุนตัวเองทำให้เกิดมุมของการหมุนขึ้น และมอเตอร์จะหยุดหมุน ก็เมื่อเส้นแรงแม่เหล็กที่ตัดผ่านตัวมันถึงจุดที่มากที่สุด



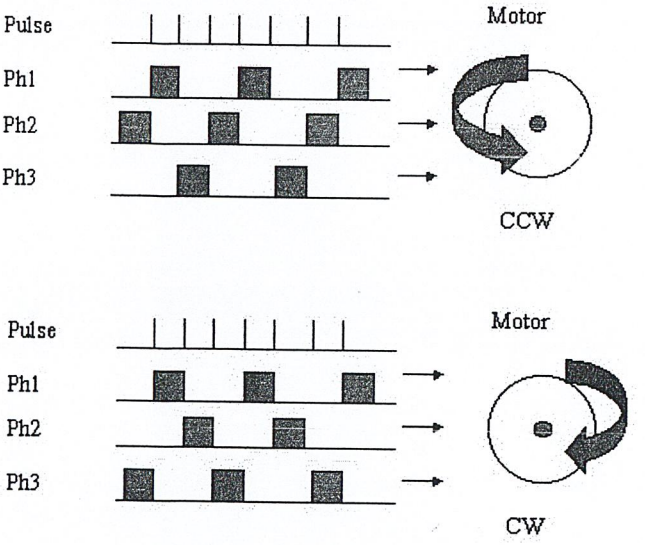
รูปที่ 2.15 เส้นแรงแม่เหล็กที่ทำให้เกิดแรงบิด

การทำให้สเต็ปปิ้งมอเตอร์หมุน ทำได้โดยอาศัยหลักการนี้ แต่ต้องทำให้เส้นแรงแม่เหล็กเกิดขึ้น โดยรับช่วงต่อกันไปเรื่อยๆ ดังรูปที่ 2.16 (ก), (ข), และ (ค) ซึ่งจะแสดงถึงการหมุนของมอเตอร์ โดยทิศทางขึ้นอยู่กับการขับกระแสเข้าขดลวด ว่าจะให้ไปทางไหน และเมื่อต้องการจะให้มอเตอร์หยุดก็หยุดการขับโรเตอร์ มอเตอร์ก็จะหยุด ณ ตำแหน่งสุดท้ายที่มีการขับที่สเตเตอร์ ดังนั้นเราจึงสามารถ รู้ตำแหน่งของมอเตอร์ได้ โดยการนับจำนวนพัลส์ที่ป้อนให้มอเตอร์โดยใช้สูตร

มุมที่เปลี่ยนไป = ค่ามุมต่อสเต็ป * จำนวนพัลส์ที่ป้อนให้



รูปที่ 2.16 แสดงการเคลื่อนที่ทีละสเต็ป เมื่อกระตุ้นเฟส 1 เฟส 2

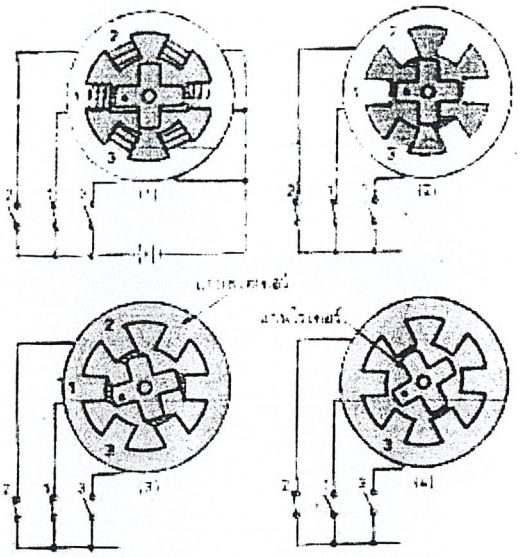


รูปที่ 2.17 แสดงการกระตุ้น แบบเดินหน้าและถอยหลัง

2.3.4 การกระตุ้นเฟสขดลวดสเตเตอร์

การทำให้สเต็ปปิ้งมอเตอร์หมุนได้นั้น จะต้องกระตุ้นเฟสของขดลวดสเตเตอร์ ให้เรียงกันไปเรื่อยๆ ทางใดทางหนึ่ง ถ้าต้องการให้หมุนกลับก็กระตุ้นเฟสในทิศทางกลับกัน ซึ่งการกระตุ้นเฟสของสเตเตอร์ มีอยู่ 3 แบบ คือ

- Single Phase Excitation
- Two Phase Excitation
- Half Step Operation



รูปที่ 2.18 แสดงการทำงานของสเต็ปปิ้งมอเตอร์

Single Phase

สัญญาณนาฬิกาที่	R	1	2	3	4	5	6	7	8
เฟสที่ 1									
เฟสที่ 2									
เฟสที่ 3									

Two Phase

สัญญาณนาฬิกาที่	R	1	2	3	4	5	6	7	8
เฟสที่ 1									
เฟสที่ 2									
เฟสที่ 3									

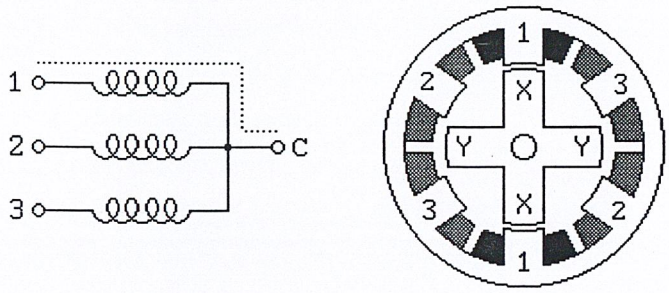
Half Step

สัญญาณนาฬิกาที่	R	1	2	3	4	5	6	7	8
เฟสที่ 1									
เฟสที่ 2									
เฟสที่ 3									

รูปที่ 2.19 แสดงการกระตุ้นเฟสของสเตเตอร์

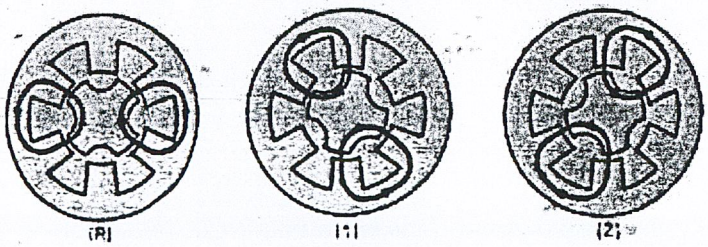
ในการขับ Two Phase Excitation เส้นแรงแม่เหล็ก จะไม่ผ่านแกนเหล็กที่เป็นเส้นตรงเลยทีเดียวเหมือนแบบ Single Phase Excitation แต่จะวกกลับมาเข้าสู่แกนทางด้านข้างๆ ดังรูปที่ 2.18 และเส้นแรงแม่เหล็กส่วนหนึ่ง จะมาจากแกนตรงข้าม ดังรูปที่ 2.20

การขับสเต็ปปิ้งมอเตอร์แบบ Two Phase Excitation จะมีลักษณะเดียวกันกับการขับแบบกระตุ้นเฟสเดียว แต่ในการกระตุ้นแต่ละครั้งนั้นจะกระตุ้นทีเดียวพร้อมกันทั้ง 2 เฟส จะต่างกันก็ตรงที่การขับ Two Phase เข้าตำแหน่งแต่ละสเต็ปได้เร็วกว่าแบบเฟสเดียว และแรงบิดมากกว่าการขับแบบ Single Phase ด้วย

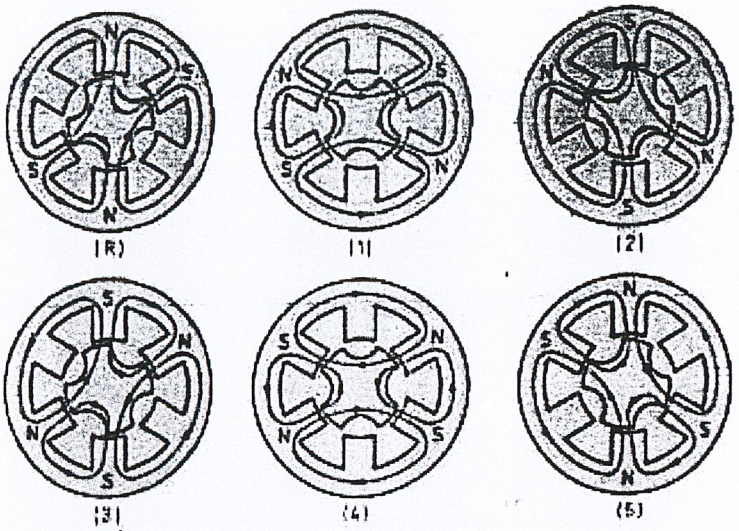


รูปที่ 2.20 แสดงการป้อนพัลส์ ให้กับสเต็ปปิ้งมอเตอร์ และการหมุน

1. เมื่อ เฟส 1 ถูกป้อนพัลส์เข้ามาทำให้เกิดสนามแม่เหล็กที่สเตเตอร์ที่ขดลวด 1 และจะไปผลักกับโรเตอร์ให้เคลื่อนที่ไปคนละทาง
2. เมื่อ เฟส 2 ถูกป้อนพัลส์ ก็จะทำให้เกิดสนามแม่เหล็กขึ้นที่ขด 2 และก็จะไปผลักโรเตอร์ให้เกิดการเคลื่อนที่ไปอีก
3. เมื่อ เฟส 3 ถูกป้อนพัลส์ ก็จะทำให้เกิดสนามแม่เหล็กขึ้นที่ขด 3 และก็จะผลักให้โรเตอร์เกิดการเคลื่อนที่ไปอีก



รูปที่ 2.21 เส้นแรงแม่เหล็กเมื่อขับแบบ 2 เฟส

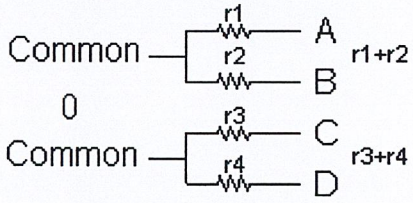


รูปที่ 2.22 ลักษณะเส้นแรงแม่เหล็กเมื่อขับแบบ 2 เฟส

รูปร่างของสเต็ปปิ้งมอเตอร์

คล้ายกับดีซีมอเตอร์ แต่มีสายไฟมากกว่า 2 เส้น และที่นิยมใช้คือ 6 เส้น สำหรับสเต็ปปิ้งมอเตอร์ 2 เฟส คือ

- A-B-Common
- C-D-Common



รูปที่ 2.23 แสดงการต่อขดลวดของสเต็ปปิ้งมอเตอร์

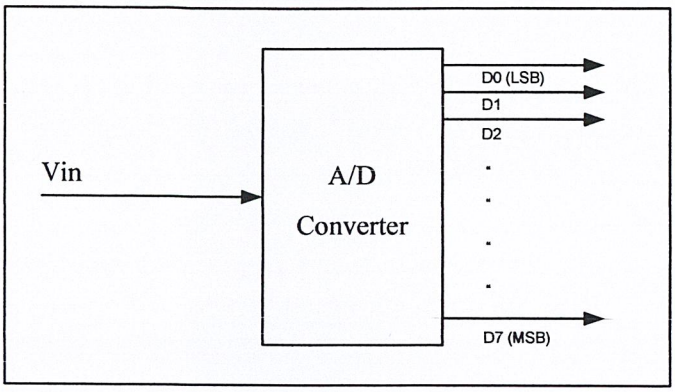
ในแต่ละเฟสจะเชื่อมถึงกัน วิธีการ Check หาเฟส โดยการใช้โอมห์มิเตอร์ ตรวจว่า 3 เส้นใดที่ช็อตถึงกัน แบ่งเป็น 2 ชุดแล้วแยก AB-Comman โดยใช้หลักว่าถ้าขดลวดมีค่าความต้านทานภายในเป็น r เมื่อเราวัดโดยจับที่คู่ A-Common และคู่ B-Common จะอ่านได้ r1 และ r2 แต่ถ้าเป็นคู่ A-B จะอ่านได้ r1+r2 ในทำนองเดียวกัน C-D Common (แต่ต้องระวังแยกเฟสให้ถูกต้อง มิฉะนั้นมอเตอร์จะไม่หมุน)

2.4 การแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล

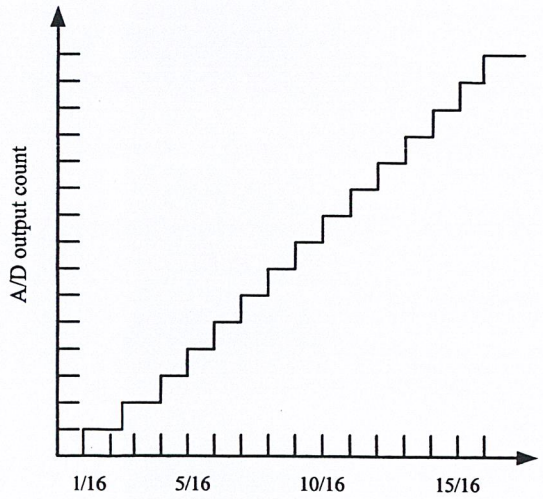
ในการวัดสัญญาณต่างๆ ด้วยคอมพิวเตอร์พีซีนั้น ซึ่งโดยปกติแล้วจะทำการประมวลผลข้อมูลซอฟแวร์จากพีซี ซึ่งการรับข้อมูลของพีซีเป็นแบบดิจิตอล ดังนั้นในการติดต่อกับสัญญาณ

อนาล็อกที่ได้จากอุปกรณ์ภายนอกหรือจากตัวแปลงสัญญาณ ดังนั้นจำเป็นต้องมีการแปลงสัญญาณอนาล็อกที่ส่วนดังกล่าวให้เป็นสัญญาณดิจิตอลเสียก่อนแล้วจึงนำมาประมวลผล

A/D Converter นั้นก็คืออุปกรณ์แปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอล รูปที่ 2.23 แสดงบล็อกไดอะแกรมของ A/D Converter โดยวงจรนี้จะผลิตเอาต์พุตแบบไบนารี่ n บิต ซึ่งเป็นสัดส่วนกับแรงดันอินพุต รูปที่ 4.2 แสดงถึงคุณสมบัติของ A/D Converter



รูปที่ 2.24 แสดงสัญญาณของ A/D Converter

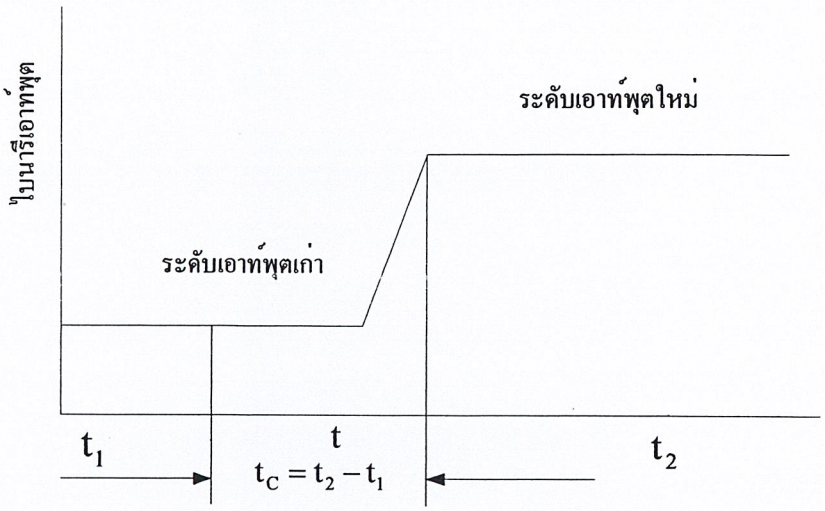


รูปที่ 2.25 คุณสมบัติของ A/D Converter แบบ 4 บิต

2.4.1 หลักการทำงานของ A/D Converter

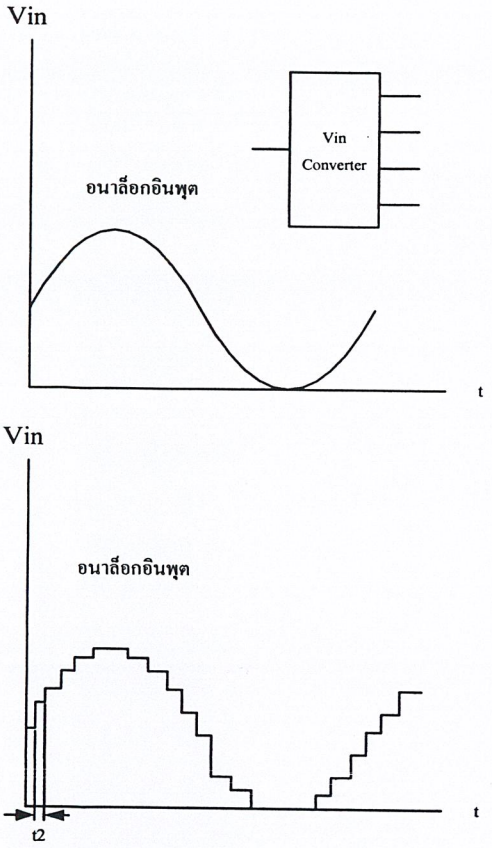
A/D Converter อินพุตจะเป็นสัญญาณอนาล็อก และส่วนเอาต์พุตจะเป็นสัญญาณดิจิตอลจำนวน n บิต อินพุตของ A/D Converter เป็นสัญญาณอนาล็อกที่ผ่านออกมาจากวงจรแซมเปิล และโฮลด์ (Sample And Hold) ซึ่งส่วนของวงจรนี้ทำหน้าที่รับสัญญาณอนาล็อกที่ต้องการแปลงเป็นสัญญาณดิจิตอลเข้ามาในวงจร แล้วจึงนำเอาต์พุตไปต่อเชื่อมกับอินพุตของ A/D Converter โดยการ

แปลงสัญญาณของ A/D Converter แต่ละครั้ง วงจรจะรับสัญญาณอนาล็อกแล้วส่งเข้าที่ขาอินพุต ของ A/D Converter ในช่วงเวลาที่เท่ากับเวลาแซมเปิล (Sample Time) ที่กำหนดจากความถี่แซม เปิล (Sample Frequency) ที่กำหนดในวงจร ดังนั้นการควบคุมการทำงานของ A/D Converter ใน การแปลงสัญญาณจะมีค่าจำนวนข้อมูลเท่าใด พิจารณาได้จากค่าความถี่แซมเปิลภายในของชุดวงจร แซมเปิล และโฮลนี้ อย่างไรก็ตามการพิจารณาเวลาการทำงานของ A/D Converter ยังขึ้นอยู่กับผล ของเวลาในการแปลงสัญญาณ (Conversion Time,) ภายใน A/D Converter คือ เวลาที่ใช้ไปใน ระหว่างที่อินพุตเข้ามาจนถึงการแสดงค่าระดับเอาท์พุตใหม่ ตามรูปที่ 2.26 เกิดผลการหน่วงเวลา (Time Delay)ขึ้น



รูปที่ 2.26 แสดงการตอบสนองของเวลาแปลงสัญญาณของ A/D Converter

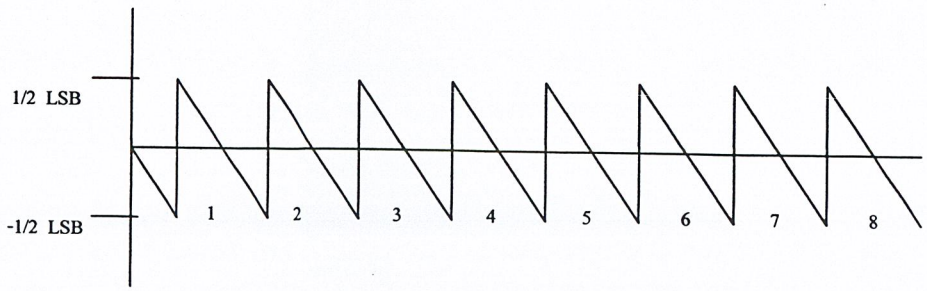
พิจารณาตามรูปที่ 2.25 อินพุตจากสัญญาณอนาล็อกที่เข้าใน A/D Converter จะอยู่ ณ เวลา t_1 และสัญญาณตอบสนองของอินพุตอนาล็อกนั้นเกิดขึ้น โดยผลต่างของเวลาทั้งสอง คือเวลาใน การแปลงสัญญาณ t_C ซึ่งเวลาดังกล่าวนี้จะเป็นเวลาที่ใช้จริงของ A/D Converter ที่ใช้ในการแปลง สัญญาณอนาล็อกเป็นสัญญาณดิจิตอล ดังนั้นการกำหนดเวลาแซมเปิล ควรจะมีเวลามากกว่าเวลา ในการแปลงสัญญาณจาก A/D Converter เพราะว่าสัญญาณอนาล็อกที่เป็นอินพุตจะต้องมีคงค่าอยู่ ตลอดเวลาในการแปลงสัญญาณ จึงจะได้ค่าดิจิตอลเอาต์พุตที่ถูกต้อง และเพื่อจะให้ทราบถึงผลของ การแซมเปิลจนสัญญาณอินพุตอนาล็อกไปเป็นสัญญาณเอาต์พุต พิจารณาจากกราฟไซน์ ดังรูปที่ 2.26 ไบนารีเอาต์พุต ถ้าให้เวลาแปลงสัญญาณน้อยมากๆ จนตัดทิ้งได้ เวลาในการแซมปลิ้ง 1/10 ของสัญญาณอินพุตจะได้กราฟเป็นลักษณะตามรูปที่ 2.26 ถ้าลดเวลาแซมเปิลให้น้อยลง ก็จะเพิ่ม รีโซลูชั่น (เอาต์พุตบิต) ให้มากขึ้นก็ได้ผลของสัญญาณอนาล็อกอินพุตมากขึ้น



รูปที่ 2.27 ขบวนการแปลงสัญญาณดิจิตอลด้วย A/D Converter

2.4.2 ความผิดพลาดของ A/D Converter

หลักสำคัญที่ใช้ในการพิจารณาถึงคุณสมบัติ ในการเปลี่ยนแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลนั้น สามารถจะสังเกตได้จากรูปที่ 2.27 จะเห็นจำนวนขั้นบันไดทั้งหมดมี 15 ขั้น เอาต์พุตของ A/D Converter ประมาณว่าเป็นสัญญาณอินพุตแบบดิจิตอล กราฟแสดงข้อผิดพลาดในเอาต์พุตที่จุดต่างๆ ดูได้จากรูปที่ 2.27



รูปที่ 2.28 ข้อผิดพลาดของ A/D Converter

ในกรณีของ A/D Converter ที่มีรีโซลูชั่นสูงๆ (เอาต์พุตบิตมากๆ) โดยความกว้างสูงสุดของข้อผิดพลาด (-1/2 LSB ถึง 1/2 LSB) จะลดลง ซึ่งโดยตามทฤษฎีแล้ว ถ้าเป็น A/D Converter ใน

อุดมคติจะต้องมีจำนวนเอาต์พุตมากนับไม่ได้ (ถึงอนันต์) ดังนั้นจะมีค่ารีโซลูชั่นถึงอนันต์จึงทำให้กราฟรูปที่ 2.27 ขนาดแกนตั้งเท่ากับ 0 หรือ ความแตกต่างระหว่าง – 1/2 LSB ถึง1/2 LSB มีค่าค่าประมาณ 0 การสร้าง A/D Converter ซึ่งเรียกข้อผิดพลาดนี้ว่า ควอนไตร์ตันเป็นข้อผิดพลาดที่ไม่สามารถกำจัดทิ้ง

ข้อผิดพลาดอีกอย่างหนึ่งคือ ผลของออฟเซตจะที่เกิดขึ้นในการแปลงสัญญาณของตัว A/D Converter ทำให้ผลการแปลงสัญญาณผิดพลาดไปได้ แต่กรณีออฟเซตนี้สามารถแก้ไขได้โดยการปรับค่าความต้านทานของอินพุตเข้า A/D Converter เอาต์พุตของ A/D Converter เกิดอไลอาสซิ่ง (Aliasing) ซึ่งในการใช้งานที่มีการเปลี่ยนแปลงสัญญาณอินพุตเร็วๆ ควรที่จะใช้การแซมเปิลแบบความเร็วสูง เพื่อจะให้ได้ความถูกต้องมากขึ้น ซึ่งเราทราบกันว่าในปัจจุบันการควบคุมระบบจะใช้การแซมเปิลที่มีอัตราสูง ในอัตราการแซมเปิลมากที่สุดจะถูกจำกัดโดยความเร็วที่ใช้ในการแปลงสัญญาณของ A/D Converter เช่น ถ้าแซมเปิลทุก 10 nS ก็จะไม่ดี เมื่อความเร็วการแปลงสัญญาณเป็น 500 nS จะทำให้เอาต์พุตออกมานำไปใช้งานไม่ได้

วงจรของ A/D Converter

ในปัจจุบันมีวิธีการและเทคนิคมากมายเกี่ยวกับการแปลงสัญญาณ แต่วิธีการหลักก็ยังคงเป็นจะพวกดังต่อไปนี้ เช่น แรมป์ A/D Converter (RampA/D Converter) , ซัสเซสซีพแอบพล็อก-ซิเมชั่น A/D Converter (Successive Aproximation A/D Converter , SAR)

แรมป์เอาต์พุตของ A/D Converter

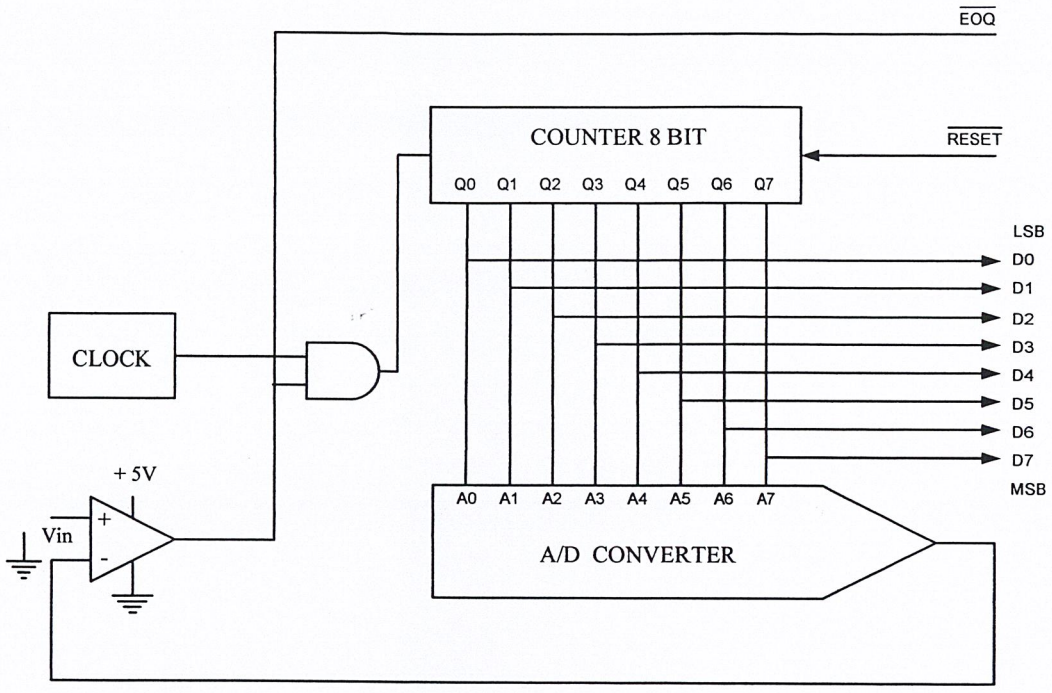
เป็น A/D Converter ที่สามารถเข้าใจการทำงานได้ง่ายที่สุดโดยแสดงโครงสร้างการทำงานดังรูปที่ 2.28 ซึ่งเมื่อดูจากรูปสามารถอธิบายการทำงานตามหลักการได้ดังนี้

1. คอมพิวเตอร์ที่ส่งสัญญาณรีเซตไปที่เคาต์เตอร์ เพื่อให้สามารถรับสัญญาณของอินพุตได้ โดยขณะนี้ผลจากเอาต์พุตของ A/D Converter (Vo) มีค่าอยู่ที่ระดับต่ำสุด

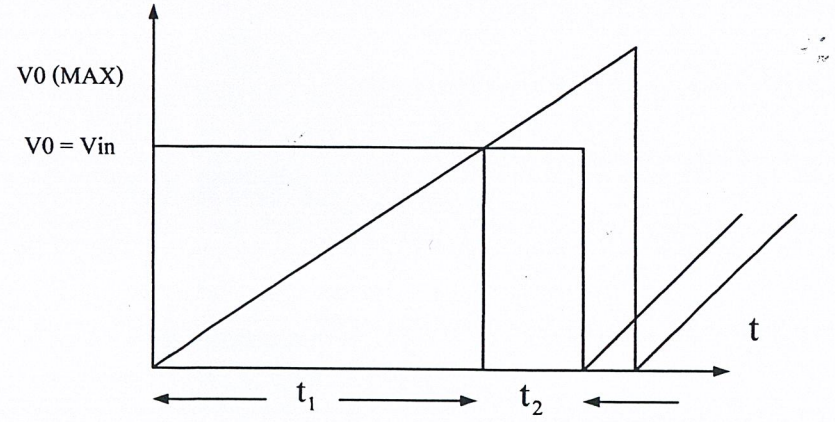
2. การทำงานของวงจรมีการเปลี่ยนแปลงความถี่ของสัญญาณนาฬิกา ซึ่งสัญญาณเอาต์พุตถูกนำมา AND กับสัญญาณนาฬิกานี้แล้ว จึงส่งไปยังเคาต์เตอร์เป็นผลให้ Vo แสดงระดับแรงดันที่สูงขึ้นทีละ 1 LSB

3. ณ. จุดเดียวกันนั้นตัวเคาต์เตอร์จะนับเพิ่มขึ้นจนกระทั่งเอาต์พุต A/D Converter สูงกว่าค่าแรงดันอินพุต Vin เมื่อถึงจุดนี้ของคอมพาราเตอร์ก็จะแสดงค่าไปจนกระทั่งเป็น 0 โวลต์ เมื่อถึง 0 โวลต์ที่จะหยุดสัญญาณนาฬิกาและหยุดการนับที่จุด ซึ่งค่า Vo ก็จะเริ่มมากกว่า Vin ขา EOC จะลดระดับเป็นระดับต่ำและส่งสัญญาณไปให้คอมพิวเตอร์ ว่าข้อมูลขณะนี้พร้อมที่จะอ่านได้แล้ว ซึ่งข้อมูลที่ส่งไปยังคอมพิวเตอร์อ่านข้อมูลโดยผ่านไปทางอินพุตพอร์ตหลังจากคอมพิวเตอร์อ่าน

4. ข้อมูลเสร็จ คอมพิวเตอร์จะส่งสัญญาณรีเซ็ตมาที่ A/D Converter หลังจากนั้นก็เริ่มทำกระบวนการแบบเดิมอีกครั้ง



รูปที่ 2.29 A/D Converter



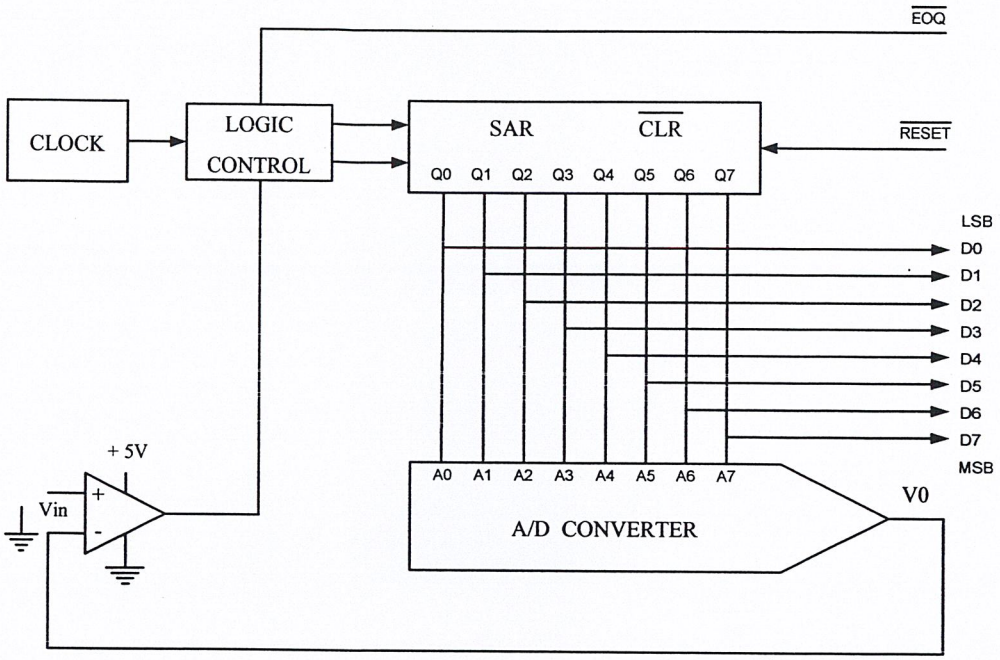
รูปที่ 2.30 กราฟเอาต์พุตของส่วน D/A ของ A/D Converter

กราฟรูปนี้แสดงถึงการทำงานของ แรมป์ โดย t_1 แทนเวลาที่ใช้นับแรมป์ เอาต์พุตของที่มี D/A Converter เริ่มจะเลย Vin และ t_2 แทนเวลาระหว่างที่คอมพิวเตอร์อ่านข้อมูลและส่งสัญญาณรีเซตกลับมาให้ A/D Converter เริ่มต้นกระบวนการเดิมอีกครั้งหนึ่ง ดังนั้นในการเก็บของข้อมูล 1 ค่า จะต้องใช้เวลาในการแปลงสัญญาณ 2 ช่วงด้วยกัน คือ เวลา t_1 ถูกกำหนดโดยค่าของ Vin และ

ความถี่ของสัญญาณนาฬิกา ส่วนเวลา t_2 ถูกกำหนดโดยเวลาของคอมพิวเตอร์อ่านข้อมูลและส่งสัญญาณรีเซตกลับ

2.4.3 ซัสเซสซีพแอบพล็อกซิเมชั่น A/D Converter

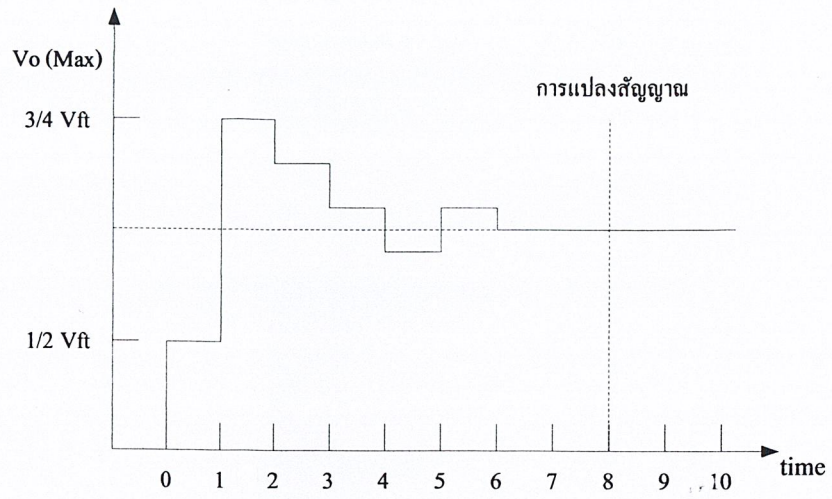
SAR เป็นตัวหลักของ A/D Converter ซึ่งจะมีลักษณะที่ใกล้เคียงกับแบบแรมป์ โดยจะมีโครงสร้างการทำงานแสดงดังรูปที่ 2.30 ซึ่งเป็นบล็อกไดอะแกรม A/D Converter แบบ SAR



รูปที่ 2.31 คอนเวอร์เตอร์ซึ่งใช้เทคนิค SAR

SAR Converter ต้องใช้ D/A Converter และมีคอมพาราเตอร์ในการทำงานที่เหมือนกัน ส่วนแตกต่างจะเห็น ได้แก่ ส่วนเคาต์เตอร์ วิธีที่แรมป์ทำงาน โดยอาศัยการนับขึ้นในลำดับไบนารี่จนกระทั่งเกิน Vin ทำงานของ SAR Converter จะอธิบายต่อไป และใช้รูปที่ 2.30 และรูปที่ 2.31 ในการบรรยายดังต่อไปนี้

1. การเริ่มต้นของการเปลี่ยนแปลงจากสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล พิจารณาจากสัญญาณนาฬิกาที่ส่งไปยังระบบควบคุมลอจิก โดยเอาต์พุตของระบบควบคุมลอจิกจะส่งพัลส์หนึ่งลูกให้กับ SAR เพื่อเซตค่าของ MSB และลบล้างค่าเอาต์พุตที่ยังเหลืออยู่ ซึ่งจะทำให้ค่าของ Vo เท่ากับค่า Vft/2 ในรูปที่ 2.31 ซึ่งจะน้อยกว่าค่า Vin และเอาต์พุตของคอมพาราเตอร์จะยังคงอยู่ในระดับ “1” ระบบควบคุมลอจิกจะตรวจสอบด้วยว่าคอมพาราเตอร์ยังคงเป็น “1” อยู่ที่จะส่งสัญญาณไปที่ SAR ซึ่งจะค้างค่าของ MSB เป็นลอจิก “1” ที่ Q7



รูปที่ 2.32 กราฟของส่วน D/A ระหว่างการแปลงสัญญาณ

2. เมื่อสัญญาณนาฬิกาถูกตัดไป MSB ตัวใหม่ ซึ่งอยู่ที่ Q6 ของ SAR จะถูกเซตค่าเป็น "1" ดังรูปที่ 2.31 เอาต์พุตของ D/A นั้นจะเกินค่าของ Vin ซึ่งทำให้เอาต์พุตคอมพาราเตอร์เปลี่ยนระดับ ระบบควบคุมลอจิกตรวจสอบทราบก็จะไม่ทำการค้างค่า Q6 นั้นคือค่าที่ค้างนั้นที่อยู่ที่ตำแหน่ง Q6 เท่ากับ "0"

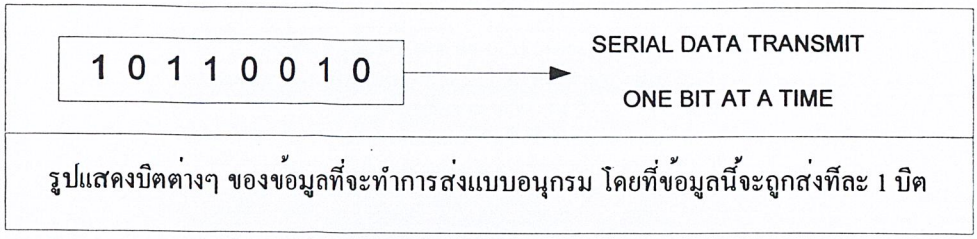
3. สัญญาณนาฬิกาลูกถัดมา Q6 จะถูกเคลียร์ และ Q5 ถูกเซตจากการทำการเปรียบเทียบได้เอาต์พุตของคอมพาราเตอร์เป็น "1" ส่วนควบคุมลอจิกก็จะไม่ทำการค้างค่า Q5 ไว้เพราะยังสูงกว่าค่าของ Vin

โดยปกติแล้ววงจรจะทดสอบเอาต์พุต จะเริ่มต้นที่ MSB ของ SAR ถ้าเอาต์พุตของ D/A Converter เกินค่า Vin แล้ว บิตนั้นก็จะค้างค่าลอจิก "1" ไว้ที่หลังการตรวจสอบหมดแล้ว ไบนารี่เอาต์พุตเป็นสัดส่วนโดยตรงกับจำนวนบิตของเคาต์เตอร์ SAR

SAR A/D Converter ก็มีข้อผิดพลาดแบบเดียวกับแบบแรมป์ คือข้อผิดพลาดทางอัตราในการขยายออฟเซต และความไม่เป็นเชิงเส้นซึ่งเกิดขึ้นในส่วน D/A

2.5 การรับส่งข้อมูลแบบอนุกรม

การรับส่งข้อมูลแบบอนุกรม เป็นการรับส่งข้อมูลทีละบิต แทนที่จะทำการส่งข้อมูลพร้อมกันทุกบิตในเวลาเดียวกัน ในการรับส่งข้อมูลแบบนี้มีชื่อว่า "การรับส่งข้อมูลแบบอนุกรม (Serial Communication)"



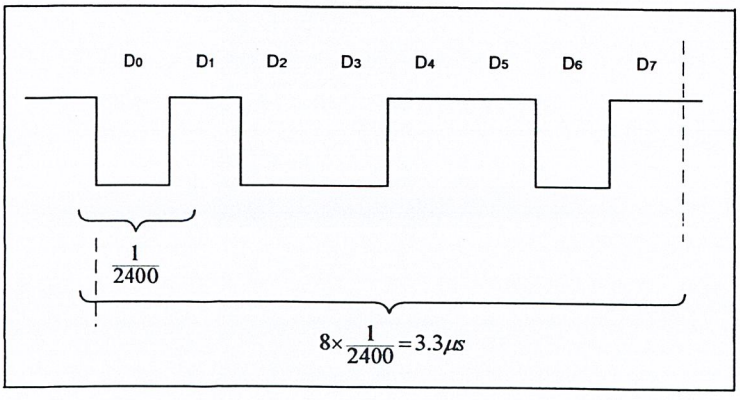
รูปแสดงบิตต่างๆ ของข้อมูลที่จะทำการส่งแบบอนุกรม โดยที่ข้อมูลนี้จะถูกส่งทีละ 1 บิต

รูปที่ 2.33 แสดงบิตของข้อมูลที่ทำการส่งแบบอนุกรม

สำหรับการรับส่งข้อมูลแบบขนานนั้นแม้ว่าจะมีความเร็วสูงกว่าการส่งแบบอนุกรมอยู่มากก็ตาม แต่ต้องใช้จำนวนสายในการส่งผ่านข้อมูลเป็นจำนวนมากกว่าแบบอนุกรม ทำให้สิ้นเปลืองค่าใช้จ่ายในการวางสายโดยไม่จำเป็น และมีการลดทอนของสัญญาณมากกว่าแบบอนุกรมอีกด้วย ทำให้เกิดความผิดพลาดในการส่งผ่านข้อมูลขึ้นได้ง่าย ดังนั้น การส่งผ่านข้อมูลในระยะทางไกลๆ เรามักจะเลือกใช้การรับส่งข้อมูลแบบอนุกรม เพื่อลดจำนวนสายส่ง ซึ่งจะช่วยในการลดค่าใช้จ่ายในการวางสายได้อย่างมาก ถึงแม้ว่าการรับส่งข้อมูลแบบนี้จะมีความยุ่งยาก และช้ากว่าการรับส่งข้อมูลแบบขนานอยู่บ้างก็ตาม

BAUD RATE สิ่งที่สำคัญมากสิ่งหนึ่งในการรับส่งข้อมูลแบบอนุกรมนี้ก็คือ ความถี่ที่ใช้ในการส่งข้อมูล ซึ่งจะต้องสัมพันธ์ระหว่างอุปกรณ์ที่ทำการรับส่งข้อมูล และความถี่ที่ใช้นี้เรียกว่า "BAUD RATE" ซึ่งจะหมายถึง "อัตราการรับส่งข้อมูลเป็นจำนวนกี่บิตในเวลา 1 วินาที" ถ้าหากว่าเครื่องส่งใช้ BAUD RATE ที่ไม่สัมพันธ์กับเครื่องรับแล้ว ก็จะทำให้การรับส่งข้อมูลเกิดผิดพลาดขึ้นได้

โดยทั่วไปค่าของ BAUD RATE นั้นจะใช้ค่าต่างๆ ดังต่อไปนี้คือ 110, 150, 3000, 1200, 2400, 4800 และ 9600 สำหรับในที่นี้จะสมมติว่า เราต้องการที่จะส่งข้อมูลแบบอนุกรมด้วยอัตรา 2400 BAUD (2400 บิต/วินาที) และข้อมูลที่ต้องการจะส่งก็คือ 0B2H 10110010B ซึ่งเราสามารถที่จะแสดงได้ในรูปของสัญญาณดังรูปที่ 2.33



รูปที่ 2.34 แสดงรูปสัญญาณของข้อมูลที่ถูกส่งไปตามสายส่งแบบอนุกรม

จากรูปที่ 2.33 จะเห็นว่าความกว้างของสัญญาณแต่ละบิตจะเท่ากับ 1/ BAUD RATE วินาที ซึ่งจาก BAUD RATE ที่ต้องการที่จะใช้คือ 2400 BAUD นั้นจะทำให้ความกว้างของแต่ละบิตมีค่าเท่ากับ 1/2400 วินาที หรือ เท่ากับ 416 microseconds ซึ่งจากความกว้างของแต่ละบิตที่จะส่งไปตามสายส่งนี้ ทำให้เราสามารถจะคำนวณเวลาที่ต้องใช้ในการรับส่งข้อมูลแต่ละไบท์ (8 บิต) ได้ดังนี้ คือเท่ากับ 8416 microseconds หรือ 3328 microseconds อย่างไรก็ตามเพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นได้ จึงมีการเพิ่มบิตต่างๆ ลงไปในแต่ละไบต์ของข้อมูล เพื่อที่จะช่วยในการตรวจสอบความถูกต้องของข้อมูลที่เครื่องรับได้รับเข้ามา สำหรับบิตต่างๆที่เพิ่มเข้ามานี้ ก็คือ START, STOP และ PARITY BIT ซึ่งจะทำให้ข้อมูลในแต่ละไบต์นี้มีมากกว่า 8 บิต และเวลาที่ใช้ในการรับส่งข้อมูลก็จะมากขึ้นตามไปด้วย

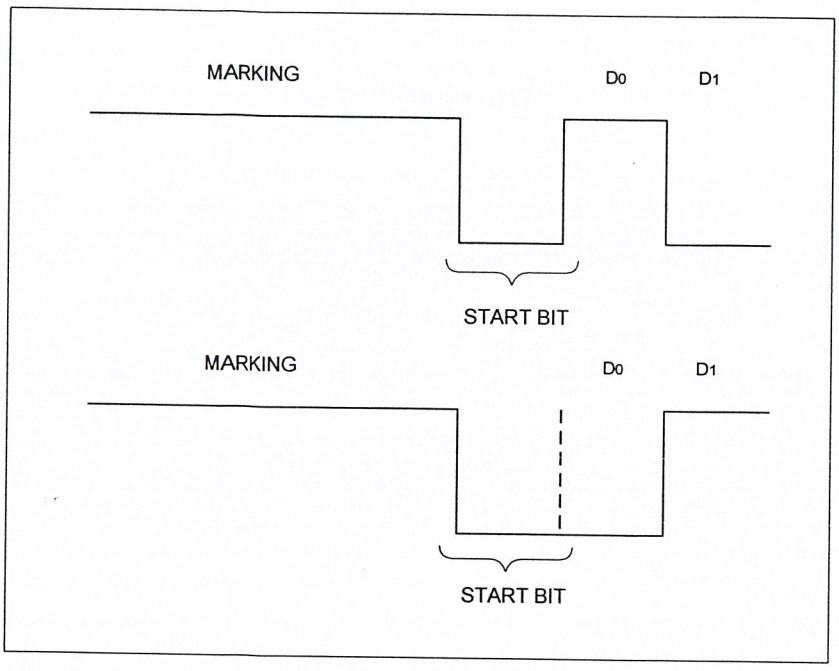
START BIT ในการส่งผ่านข้อมูลแบบอนุกรมนี้ เราจำเป็นที่จะต้องทำให้อุปกรณ์ที่จะรับข้อมูลทราบว่า ข้อมูลที่ส่งมานั้นเริ่มต้นที่จุดใด ดังนั้นเราจึงจำเป็นที่จะต้องเพิ่มข้อมูล 1 บิต ลงไปก่อนหน้าข้อมูลจริง (ACTUAL DATA) ที่จะทำการรับส่ง (การส่งอนุกรมจะส่งบิต D0 เป็น บิตแรก และ D7 เป็นบิตสุดท้าย) คือ ทำการเพิ่มบิตนี้ลงไปหน้าบิต D0 นั่นเอง และเรียกบิตนี้ว่า "START BIT"

หน้าที่ของ START BIT นี้นอกจากจะใช้ในการบอกว่าข้อมูลนั้นเริ่มต้นที่ใดแล้ว ยังทำงานร่วมกับ STOP BIT เพื่อช่วยในการแยกข้อมูลแต่ละชุดออกจากกัน และความกว้างของบิตนี้จะมีค่าเท่ากับความกว้างของบิตอื่นๆ ในข้อมูลที่จะส่ง

เมื่ออุปกรณ์ที่จะส่งข้อมูลยังไม่ได้ทำการส่งข้อมูลใดๆ ออกมานั้น สายส่งจะอยู่ในสภาวะที่เรียกว่า "MARKING" ซึ่งเป็นสภาวะที่ไม่มีการรับส่งข้อมูลใดๆ เกิดขึ้น ในที่นี้เราจะสมมติให้ค่า MARKING ของสายส่งเป็นลอจิก "1" START BIT ที่จะเพิ่มเข้าไปนี้จะมีลอจิกที่ตรงข้ามกับลอจิกของ MARKING ดังนั้นในกรณีนี้ START BIT จะมีลอจิก "0"

สำหรับ START BIT นี้จะมีความกว้างเท่ากับ 1 บิตของข้อมูล เช่น ใน 1 บิตของข้อมูลที่มีความยาวเท่ากับ 416 microseconds START BIT ก็จะมีความกว้างของสัญญาณเท่ากับ 416 micro-seconds ด้วย ในรูปที่ 2.34 จะแสดงให้เห็นถึง START BIT ที่เพิ่มเข้าไปก่อนหน้าข้อมูล

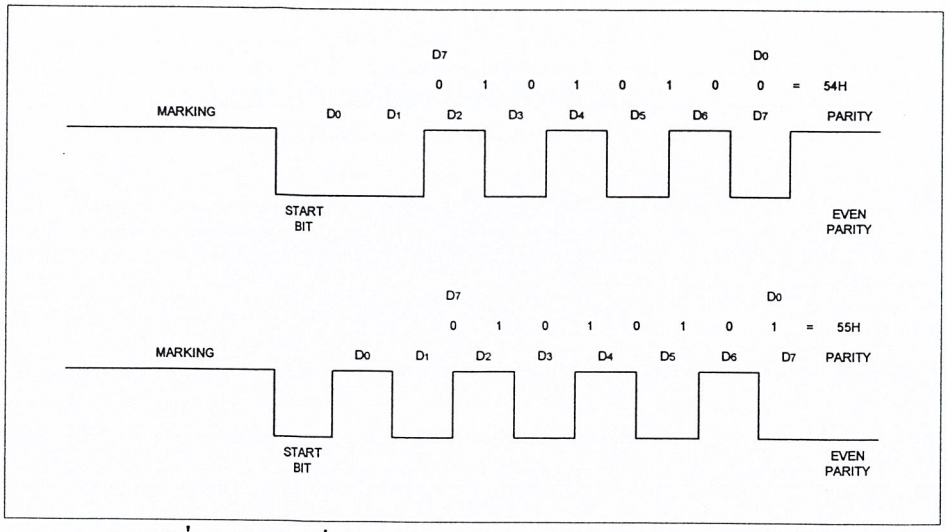
PARITY BIT สำหรับบิตนี้จะทำหน้าที่ในการบอกให้ส่วนรับข้อมูลทราบว่า ข้อมูลที่รับเข้ามานั้นมีความถูกต้องเหมือนกับข้อมูลที่ถูกออกมาหรือไม่ โดยที่บิตนี้จะทำหน้าที่ในการบอกให้ส่วนรับข้อมูลที่ส่งออกมาแต่ละไบต์นั้นมีจำนวนบิตที่เป็น "1" อยู่เป็นจำนวนคี่ หรือว่าเป็นจำนวนคู่ เช่น ข้อมูล 54H หรือ 01010111B จะมีจำนวนบิตที่เป็น "1" อยู่เป็นจำนวนคี่เป็นต้น สำหรับบิตที่ใช้ในการตรวจสอบนี้เรียกว่า "PARITY BIT"



รูปที่ 2.35 การเพิ่ม START BIT เข้าไปก่อนหน้าบิต D0

PARITY BIT นี้จะถูกส่งออกมาโดยอุปกรณ์ส่งข้อมูล ซึ่งบิตนี้จะเป็น “1” หรือ “0” นั้นขึ้นอยู่กับข้อมูลที่ส่งออกมา (D0 – D7) ว่ามีจำนวนบิตที่เป็น “1” เป็นจำนวนคู่หรือคี่ และยังขึ้นกับอุปกรณ์รับส่งข้อมูลด้วยว่าถูกออกแบบไว้ให้รับส่ง PARITY BIT ในลักษณะของ PARITY คู่ หรือ PARITY คี่ อีกด้วย

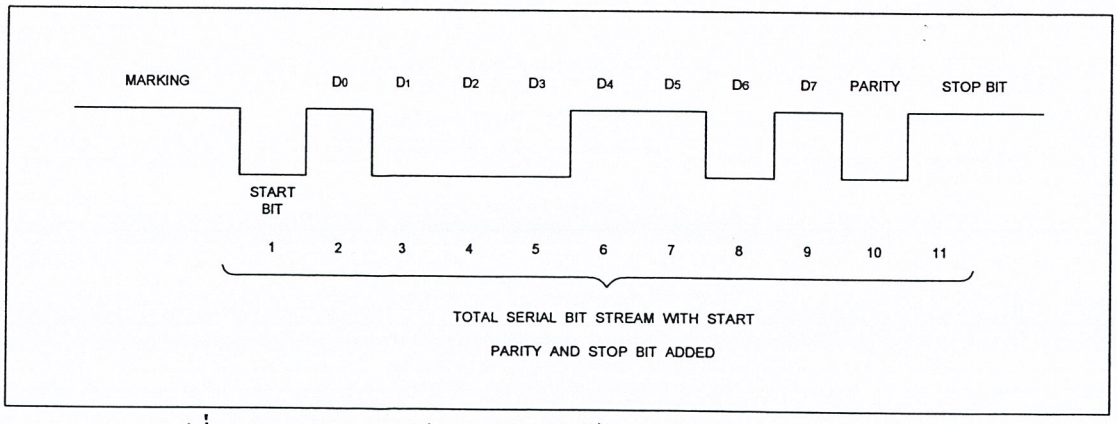
ในกรณีที่อุปกรณ์รับส่งข้อมูลถูกออกแบบไว้ให้เป็น PARITY คู่ อุปกรณ์ส่งข้อมูลก็จะทำการส่ง PARITY BIT ที่เป็นลอจิก “1” ออกไป เมื่อจำนวนบิตที่เป็น “1” ของข้อมูล (D0 – D7) เป็นจำนวนคี่และจะทำการส่ง PARITY BIT เป็นลอจิก “0” เมื่อจำนวนบิตที่เป็น “1” ของข้อมูลเป็นจำนวนคู่ สำหรับ PARITY คี่ก็เช่นกัน คือ PARITY BIT จะเป็น “1” ในกรณีที่จำนวนบิตที่เป็น “1” ของข้อมูลเป็นจำนวนคู่และจะเป็น “0” ในกรณีที่เป็นจำนวนคี่ ในที่นี้จะสมมติว่าอุปกรณ์ถูกออกแบบไว้สำหรับ PARITY คู่ เราต้องการที่จะส่งข้อมูลออกไปให้กับส่วนรับข้อมูลเป็นจำนวน 2 ไบต์ คือ 54H และ 55H เมื่อเราส่งข้อมูล 54H ออกไป ซึ่งมีจำนวนบิตที่เป็น “1” เป็นจำนวนคี่ ดังนั้นในกรณีนี้อุปกรณ์ส่งข้อมูลก็จะทำการส่ง PARITY BIT เป็นลอจิก “1” ตามออกมาด้วยเพื่อที่จะให้ได้จำนวนบิตที่เป็น “1” ของข้อมูล (54H) รวมกับ PARITY BIT แล้วให้ได้เป็นจำนวนคู่ด้วย ส่วนข้อมูล 55H นั้นจำนวนบิตที่เป็น “1” เป็นจำนวนคู่อยู่แล้ว ดังนั้นอุปกรณ์ส่งข้อมูลก็จะส่งข้อมูล PARITY BIT เป็น “0” ให้กับส่วนรับข้อมูล สำหรับส่วนรับข้อมูลนั้นเมื่อทำการรับข้อมูลเข้ามาแล้วก็จะตรวจสอบสัญญาณว่าจำนวนบิตที่เป็น “1” ของข้อมูลรวมกับ PARITY BIT นั้นเป็นจำนวนคู่หรือไม่ ถ้าหากเป็นจำนวนคี่ก็แสดงว่าข้อมูลที่ได้รับเข้ามานี้ความผิดพลาดเกิดขึ้น



รูปที่ 2.36 การเพิ่ม PARITY BIT ลงไปในข้อมูลแต่ละไบต์

สิ่งสำคัญอีกสิ่งหนึ่งก็คือ ถ้าอุปกรณ์ส่งข้อมูลทำการส่งในลักษณะ PARITY คู่หรือคี่ก็ตาม ส่วนรับข้อมูลก็จะต้องทำการรับในลักษณะ PARITY เดียวกับอุปกรณ์ส่งข้อมูลด้วย เช่น ในกรณีที่อุปกรณ์ส่งข้อมูลทำการส่งข้อมูลในลักษณะของ PARITY คู่ อุปกรณ์รับข้อมูลก็จะต้องทำการรับข้อมูลในลักษณะของ PARITY คู่ด้วย เป็นต้น

STOP BIT สำหรับบิตสุดท้ายที่เพิ่มเข้าไปนี้ จะใช้การตรวจสอบจุดสิ้นสุดของข้อมูลบิตนี้จะถูกเพิ่มเข้าไปหลัง PARITY BIT ถ้าอุปกรณ์รับข้อมูลตรวจไม่พบบิตนี้ก็แสดงว่า ข้อมูลที่ได้รับเข้ามานั้นมีความผิดพลาดเกิดขึ้น สำหรับ STOP BIT นี้ อาจมีจำนวนบิตเป็น 1, 1.5, หรือ 2 ก็ได้ รูปที่ 2.36 นั้นจะแสดงข้อมูลทั้ง 8 บิตที่ส่งออกมา รวมทั้ง START, STOP และ PARITY BIT ด้วย ซึ่งจะเห็นว่า สิ่งที่ส่งออกมาในแต่ละไบต์นั้นไม่ได้มีเพียงข้อมูล 8 บิต เท่านั้น แต่อาจจะมีได้ถึง 12 บิต ดังนั้นถ้าเราทำการส่งด้วยอัตรา 2400 BAUD จะต้องใช้เวลาทั้งหมดเป็น 12416 microseconds หรือ 4.99 milliseconds ไม่ใช่ 3328 microseconds ดังที่ได้คำนวณไว้ในตอนต้น



รูปที่ 2.37 รูปแบบของข้อมูลแต่ละไบต์ในการรับส่งข้อมูลแบบอนุกรม

2.5.1 หลักการเบื้องต้นของการรับส่งข้อมูลแบบอนุกรม

สำหรับหลักการเบื้องต้นของการรับส่งข้อมูลแบบอนุกรมที่ได้กล่าวถึงนั้น สามารถสรุปได้เป็นข้อๆ ดังนี้

1. ข้อมูลแบบขนานจากระบบถูกเปลี่ยนให้เป็นข้อมูลแบบอนุกรม เพื่อเตรียมที่จะส่งออกไปให้กับส่วนรับข้อมูล

2 ข้อมูลจะถูกส่งออกไปด้วยอัตราคงที่ค่าหนึ่ง ซึ่งเรียกว่า "BAUD RATE" คือถ้าทำการส่งข้อมูลด้วยอัตรา 1200 BAUD ก็แสดงว่าเป็นการส่งข้อมูลด้วยอัตรา 1200 บิตต่อ 1 วินาที ซึ่งก็คือการส่งข้อมูลโดยใช้ความถี่ 1200 Hz นั่นเอง

3. ข้อมูลอนุกรมจะถูกส่งออกไปทีละบิตโดยทำการส่งบิต D0 เป็นบิตแรก และส่งบิต D7 เป็นบิตสุดท้าย

4. ในขณะที่ยังไม่มีการส่งข้อมูลเข้าไปในสายส่ง สายส่งจะถูกทำให้อยู่ในสภาวะลอจิกใดลอจิกหนึ่ง และเราเรียกสภาวะนี้ว่า "MARKING"

5. อุปกรณ์ส่งข้อมูลจะเพิ่มข้อมูลอีก 1 บิต เข้าไปหน้า D0 ของข้อมูลที่จะส่งให้กับตัวของเครื่องรับ บิตที่เพิ่มเข้าไปนี้เรียกว่า "START BIT" สำหรับบิตนี้จะมีลอจิกตรงข้ามกับลอจิกของ MARKING เป็น "1" ลอจิกของบิตนี้ก็จะเป็น "0"

6. อุปกรณ์ส่งข้อมูลจะทำการเพิ่ม PARITY BIT ให้เข้าไปหลังบิต D7 ของข้อมูล เพื่อใช้ในการตรวจสอบความผิดพลาดของข้อมูลที่เครื่องรับ

7. สำหรับบิตสุดท้ายที่ถูกเพิ่มเข้าไปหลัง PARITY BIT เรียกว่า "STOP BIT" ซึ่งอาจจะมีจำนวน 1, 1.5, หรือ 2 บิตก็ได้ และลอจิกของบิตเป็นลอจิกเดียวกับลอจิกของMARKING

2.6. Microsoft Visual Basic

ปัจจุบันเทคโนโลยีทางด้านคอมพิวเตอร์ ได้ก้าวหน้าอย่างรวดเร็วจากอดีตที่ผ่านมาทั้งด้านฮาร์ดแวร์(Hardware) และซอฟต์แวร์ (Software) โดยฮาร์ดแวร์นั้นเป็นอุปกรณ์ที่ประกอบขึ้นเป็นคอมพิวเตอร์ เช่น CPU ฮาร์ดดิสก์ เป็นต้น ส่วนซอฟต์แวร์นั้นเป็นชุดคำสั่งที่ทำให้คอมพิวเตอร์นั้นทำงานได้ ซึ่งมีการเปลี่ยนแปลงอย่างรวดเร็วมาก และในปัจจุบันได้มีเครื่องมือต่างๆ ที่ช่วยในการพัฒนาโปรแกรมอย่างแพร่หลาย เรียกว่า Microsoft Visual Basic 6 ที่ช่วยให้การพัฒนาซอฟต์แวร์ต่างๆ เป็นไปได้อย่างรวดเร็ว ทั้งยังสามารถพัฒนาซอฟต์แวร์ได้หลายอย่างด้วยกัน ตั้งแต่โปรแกรมธรรมดาทั่วไป โปรแกรมเกี่ยวกับฐานข้อมูล หรือโปรแกรมทางอินเตอร์เน็ต เป็นต้น

สำหรับ VB6 เป็นเครื่องมือที่สร้างโปรแกรมบนระบบปฏิบัติการ Windows ที่ใช้งานได้ง่าย โดยในการสร้างโปรแกรมใน VB6 นั้น จะเป็นการเลือกเครื่องมือต่างๆ ที่มีอยู่มาออกแบบหน้าจอ

โปรแกรมที่เราจะสร้าง ซึ่งเรียกการเขียนโปรแกรมลักษณะนี้ว่า Visual Programming ในการเขียนโปรแกรมนี้ เราจะไม่จำเป็นต้องเขียนคำสั่งต่างๆ มากนัก ก็สามารถสร้างโปรแกรมได้อย่างรวดเร็ว

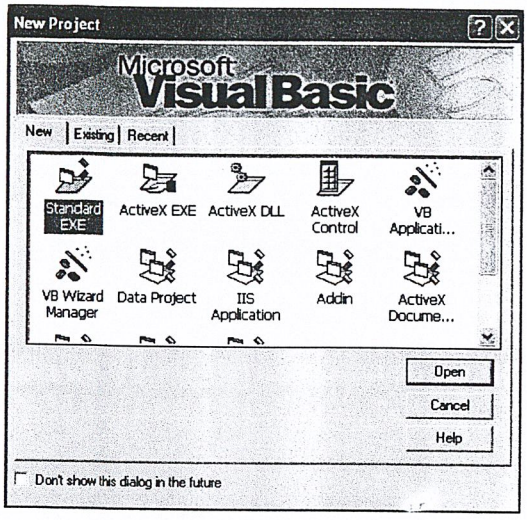
2.6.1 ฮาร์ดแวร์หรืออุปกรณ์ที่เหมาะสมกับการใช้โปรแกรม Visual Basic

การจะสามารถรันโปรแกรมวิชวลเบสิกได้อย่างไม่มีปัญหา จะต้องแน่ใจก่อนว่าฮาร์ดแวร์และซอฟแวร์ของเราพร้อมแล้ว โดยความต้องการของระบบมีดังต่อไปนี้

1. ไมโครซอฟต์วินโดว์ 98 หรือสูงกว่านั้น หรือไมโครซอฟต์เอ็นทีเวอร์กเตชัน 4.0 (Service Pack 3 recommended) หรือมากกว่า
2. เพนเทียม หรือสูงกว่า หรือชิปอัลฟาร์โปรเซสเซอร์ ที่สามารถรันบนไมโครซอฟต์เอ็นทีเวอร์กสเตชั่นได้
3. ไดรฟ์ซีดีรอม
4. การ์ดวีจีเอ หรือสูงกว่า สนับสนุนการแสดงผลระบบวินโดว์
5. แรม 64 MB สำหรับวินโดว์ 98, 64 MB สำหรับไมโครซอฟต์เอ็นทีเวอร์กสเตชัน ยิ่งมากยิ่งดี
6. เมาท์, จอภาพ และอื่นๆ ที่ระบบวินโดว์รองรับ
7. ซอฟแวร์โปรแกรมไมโครซอฟวิชวลเบสิก 6

2.6.2 เริ่มต้นกับโปรแกรมวิชวลเบสิก

สิ่งแรกที่ได้พบเมื่อเราเข้าสู่โปรแกรมวิชวลเบสิก โดยที่โปรแกรมวิชวลเบสิกนั้นจะแสดงไดอะล๊อกบ๊อกซ์ เพื่อให้เราเลือกพัฒนาชนิดของแอพพลิเคชั่นในเบื้องต้น โดยให้เราเลือกพัฒนาแอพพลิเคชั่นชนิด Standard.EXE ก่อน เพราะว่าแอพพลิเคชั่นที่ใช้ทั่วๆไปนั้น เมื่อเราแปลงโค๊ด (Compile) โปรเจ็กต์แล้ว จะได้แอพพลิเคชั่นมีนามสกุลเป็น *.exe เมื่อเลือกชนิดของแอพพลิเคชั่นแล้ว จะเข้าสู่สภาพแวดล้อม (Environment) ของโปรแกรมวิชวลเบสิก ซึ่งในไมโครซอฟต์เรียกว่า Integrated Development Environment – IDE



รูปที่ 2.38 ไดอะล็อกซ์บ๊อกซ์ชนิดของแอพพลิเคชันสำหรับที่ให้โปรแกรมเมอร์เลือกพัฒนา

ตารางที่ 2.2 ตารางสำหรับแอพพลิเคชั่นต่างๆ มีดังต่อไปนี้

ไอคอน	ความหมาย
Standard EXE	ใช้พัฒนาแอพพลิเคชั่นทั่วๆไป และใช้สร้างโปรแกรมแบบกราฟฟิกทั่วๆไป
ActiveX EXE	ใช้พัฒนาแอพพลิเคชั่นที่สามารถใช้งาน และเชื่อมโยงกับอพพลิเคชั่นอื่นๆ ที่สนับสนุนเทคโนโลยีเอคทีฟเอ็กซ์
ActiveX DLL	เป็นแอพพลิเคชั่นชนิดเดียวกับ Active.EXE แต่จะเก็บอยู่ในลักษณะไฟล์ไลบารี่ *.dll (In-Process) ซึ่งไม่สามารถรันได้ด้วยตัวเอง จะต้องถูกเรียกใช้งานจากแอพพลิเคชั่นอื่นๆ
ActiveX Control	ใช้สร้างคอนโทรลแอคทีฟเอ็กซ์ (ActiveX) ขึ้นมาใช้งานเอง
VB Application Wizard	เป็นเครื่องมือที่อำนวยความสะดวกให้สามารถสร้างแอพพลิเคชันได้อย่างรวดเร็ว โดยมันจะสร้างองค์ประกอบเบื้องต้นหลักๆ ของแอพพลิเคชั่น จากขั้นตอนที่เราได้เลือกไว้ แอพพลิเคชั่นที่ได้จะมีความสมบูรณ์ในระดับหนึ่งเท่านั้น เราต้องปรับปรุง เพิ่มเติมและแก้ไขเพื่อทำให้เป็นมาตรฐานยิ่งขึ้น ถือได้ว่าเป็นเครื่องมือที่ผู้เริ่มต้นอาจใช้เครื่องมือตัวนี้ เพื่อศึกษาแนวทางในการพัฒนาแอพพลิเคชันก็ได้

ไอคอน	ความหมาย
VB Wizard Manager	ใช้สำหรับสร้างโปรแกรมการจัดการต่างๆ เช่น การติดต่อกับฐานข้อมูล เป็นต้น
Data Project	เป็นชนิดโปรเจ็กต์ที่เป็นแบบฟอร์ม เพื่อติดต่อกับฐานข้อมูลโดยผ่านทางคอนโทรล ADO Data Control
IIS Application	แอพพลิเคชั่นชนิดที่ใช้กับเว็บเซอร์เวอร์ Web Server)
Addin	สำหรับเพิ่มเติมโปรแกรมอัตถะประโยชน์เข้าไปในสภาวะแวดล้อมของวิชวลเบสิก เพื่อเพิ่มประสิทธิภาพ ซึ่งอาจจะมาจากผู้ผลิตอิสระรายอื่นๆ (Third-Party) หรือเพิ่มเครื่องมืออื่นๆ เข้ามาในสภาวะแวดล้อมของโปรแกรมวิชวลเบสิกก็ได้
ActiveX Document Dll	ใช้สร้างแอพพลิเคชั่นชนิดที่ทำงานบนอินเตอร์เน็ต จะเก็บอยู่ในรูปไฟล์ *.dll ไม่สามารถรันได้ด้วยตัวมันเองต้องให้แอพพลิเคชันอื่นๆ ที่สนับสนุนเทคโนโลยีเอคทีฟเอ็กซ์ (ActiveX) เรียกใช้งาน เช่น Internet Explorer เป็นต้น
Activex Document Exe	ใช้สร้างแอพพลิเคชั่นชนิดที่ทำงานบนอินเตอร์เน็ต แต่จะเก็บอยู่ในรูปไฟล์ *.exe สามารถรันได้ด้วยตัวเอง แต่เซอร์เวอร์ (Server) ด้วยเช่นกัน เช่น Internet Explorer เป็นต้น
DHTML Application	ใช้พัฒนาแอพพลิเคชั่นรูปแบบของเอกสารไดนามิกดีเอ็ชทีเอ็มแอล (DynamicHTML) ซึ่งจะเป็นมาตรฐานใหม่ของการแสดงผลบนเว็บเพจ
VB Enterprise Edition Controls	ใช้สำหรับโหลดสภาวะแวดล้อมของโปรแกรมวิชวลเบสิกในรูปแบบที่ใช้พัฒนาแอพพลิเคชั่นในระดับสถานประกอบการ (Enterprise) ซึ่งโปรแกรมวิชวลเบสิกจะเพิ่มเติมคอนโทรลแอคทีฟเอ็กซ์ (ActiveX) อีกจำนวนหนึ่งขึ้นมาโดยอัตโนมัติ เหมาะสำหรับเครื่องที่มีประสิทธิภาพสูง

2.6.3 ไฟล์ประเภทต่างๆ ที่มีในโปรเจ็กต์ของ VB6

โปรเจ็กต์เป็นไฟล์เก็บฟอร์มและโมดูลต่างๆ เช่น คลาสโมดูล โมดูลของ Active X Control ซึ่งในโปรเจ็กต์หนึ่งจะมีไฟล์โมดูลต่างๆ อยู่ได้หลายไฟล์ โดยโปรเจ็กต์ที่เราสร้างขึ้นจะมีไฟล์ในรูปแบบต่างๆ ที่เป็นไปได้ดังต่อไปนี้

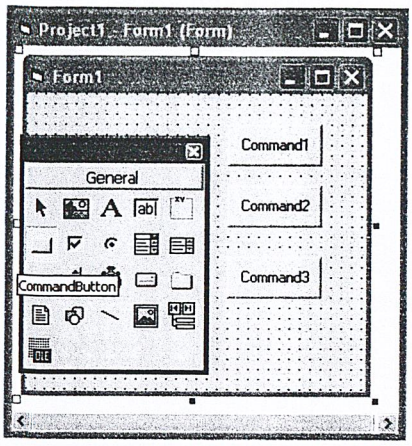
ตารางที่ 2.3 แสดงกลุ่มไฟล์โปรเจ็กต์

ไฟล์กลุ่มโปรเจ็กต์	คำอธิบาย	ส่วนขยายไฟล์
ไฟล์กลุ่มโปรเจ็กต์	เป็นไฟล์ที่ใช้เก็บว่ามีโปรเจ็กต์อะไรเก็บอยู่บ้าง (ต้องมากกว่า 1 โปรเจ็กต์ขึ้นไป	*.vbg
ไฟล์โปรเจ็กต์	เป็นไฟล์หลักโปรเจ็กต์ต่างๆ ของแอพพลิเคชั่นของเรา	*.vbp
ไฟล์ของฟอร์ม	เป็นไฟล์ที่เก็บข้อมูลเกี่ยวกับฟอร์ม เช่น คอนโทรลต่างๆ และคำสั่งจัดการอีเวนต์สำหรับฟอร์มนั้นๆ เป็นต้น เช่น รูปภาพ หรือ ไอคอน เป็นต้น	*.frm
ไฟล์ไบนารี ของฟอร์ม	ส่วนใหญ่จะใช้เก็บค่าคงที่เป็นไบนารีของฟอร์ม เช่น รูปภาพ หรือ ไอคอน เป็นต้น	*.frx
ไฟล์โมดูลมาตรฐาน	ส่วนใหญ่จะใช้เก็บค่าคงที่ ตัวแปร โปรแกรมย่อยที่ให้โมดูลอื่นๆ สามารถเรียกใช้งาน	*.bas
ไฟล์คลาสโมดูล	ใช้ในการสร้างออบเจ็กต์ที่มีลักษณะต่างๆ ตามที่เราต้องการเองได้	*.cls
ไฟล์ของ ActiveX Control	จะเป็นไฟล์ของคอนโทรล Active X ซึ่งเป็นคอนโทรลที่สร้างขึ้นมาเองได้ และสามารถนำไปใช้ในแอพพลิเคชั่นทั่วไปที่สร้างขึ้นมาใหม่ได้	*.ctl
ไฟล์ของ ActiveX Documents	จะเป็นไฟล์ของแอพพลิเคชั่น ที่สามารถนำไปแสดงในโปรแกรมเว็บบราวน์เซอร์ได้	*.dob
ไฟล์ของ Property Pages	จะเป็นไฟล์ของ Property Pages ที่ใช้แสดงคุณสมบัติของคอนโทรลของเรา	*.pag

2.6.4 คอนโทรลที่สำคัญที่ใช้ในโครงงาน

ปุ่มคำสั่ง (Command Button)

ปุ่มคำสั่งเป็นคอนโทรลที่จะทำงานตามที่เรากำหนด โดยเมื่อมีการกดปุ่มคำสั่งซึ่งเราต้องกำหนดสิ่งที่จะตอบสนองในอีเว็นต์ Click



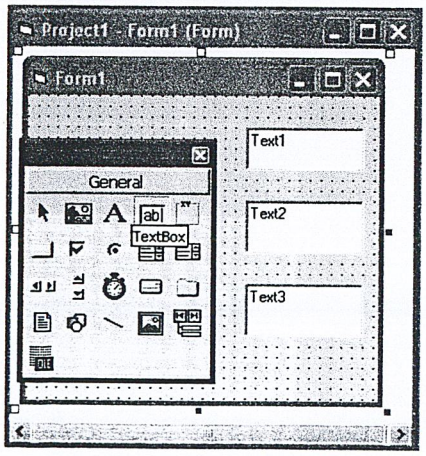
รูปที่ 2.39 คอนโทรลคอมมานด์บัสทอนบนทูลบ๊อกซ์ และปรากฏอยู่บนฟอร์ม

คุณสมบัติที่สำคัญคือ

- **Caption** คือ ข้อความที่ปรากฏบนปุ่มคำสั่ง
- **Enable** คือ การกำหนดให้สามารถกดปุ่มคำสั่งได้ ถ้าหากกำหนดให้เป็น True จะสามารถจะใช้ได้ตามปกติ แต่ถ้ากำหนดเป็น False จะไม่สามารถใช้งานปุ่มคำสั่งนั้นได้ ลักษณะของปุ่มจะเป็นไปเพื่อให้เห็นว่าไม่สามารถใช้ปุ่มคำสั่งนั้นได้ในเวลานั้น

เท็กซ์บ็อกซ์ (Text Box)

เท็กซ์บ็อกซ์เป็นคอนโทรลที่ใช้เพื่อให้ผู้ใช้เดิม หรือเพื่อแก้ไขข้อความ โดยผ่านคีย์บอร์ดเข้าสู่โปรแกรม และยังประยุกต์ใช้เพื่อเป็นการแสดงค่าได้ โดยเราสามารถตอบสนองต่อข้อความที่มีการเปลี่ยนแปลงในเท็กซ์บ็อกซ์ได้ โดยผ่านทางอีเว็นต์ Change



รูปที่ 2.40 คอนโทรลเท็กซ์บ๊อกซ์บนทูลบ๊อกซ์ และที่ปรากฏอยู่บนฟอร์ม

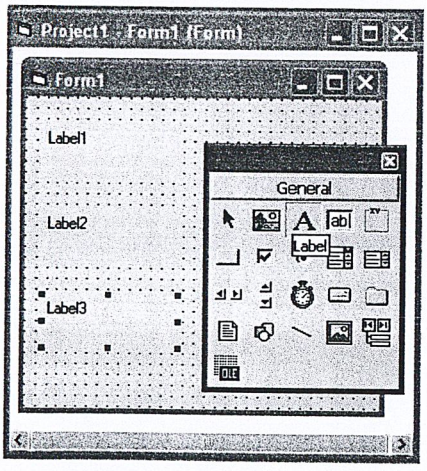
คุณสมบัติที่สำคัญ คือ

- **Text** คือ ข้อความที่แสดงในเท็กซ์บ็อกซ์
- **Enable** คือ การกำหนดให้สามารถเปลี่ยนแปลงค่าในเท็กซ์บ็อกซ์ได้ ถ้ากำหนดเป็น True จะสามารถเปลี่ยนแปลงค่าในเท็กซ์บ็อกซ์ได้ แต่ว่าถ้าเรากำหนดเป็น False ก็จะไม่สามารถทำการเปลี่ยนแปลงค่าในเท็กซ์บ็อกซ์ได้

ลาเบล (Label)

ลาเบล เป็นคอนโทรลที่ใช้ในการแสดงข้อความที่เราต้องการบนฟอร์ม ซึ่งเราจะใช้อธิบายข้อมูลบางอย่างแก่ผู้ใช้ โดยผู้ใช้ไม่สามารถแก้ไขข้อความในลาเบลได้ในตอนขั้นการรันโปรแกรม แต่สามารถแก้ไขข้อมูลได้โดยใช้คำสั่งโปรแกรมตอนรัน และตอนออกแบบโปรแกรมเท่านั้น

ส่วนใหญ่ใช้ลาเบลในการอธิบายการทำงาน รวมทั้งแสดงสถานะบางอย่าง เช่น ใช้ในการอธิบายว่าเท็กซ์บ็อกซ์นี้รับข้อมูลอะไร หรือใช้แสดงข้อมูล, คำอธิบายบนฟอร์ม เป็นต้น

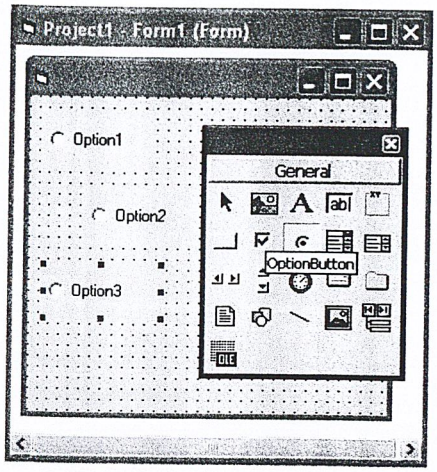


รูปที่ 2.41 คอนโทรลลาเบลบนทูลบ๊อกซ์ และที่ปรากฏอยู่บนฟอร์ม

ออปชั่นบัตตอน (Option Button)

คอนโทรลออปชั่นบัตตอนใช้สำหรับเป็นตัวเลือกให้แก่ผู้ใช้ หากมีหลายออปชั่นบัตตอนบนฟอร์ม ผู้ใช้จะเลือกออปชั่นบัตตอนได้ครั้งละ 1 ตัวเท่านั้น ซึ่งเมื่อเราเลือกออปชั่นบัตตอนหนึ่งจะทำให้ออปชั่นบัตตอนเดิมที่เราเลือกเปลี่ยนไปเป็นไม่ถูกเลือกโดยอัตโนมัติ

เราสามารถใช้คอนโทรลเฟรมในการแบ่งกลุ่มของตัวเลือกต่างๆ โดยที่ในแต่ละกลุ่มผู้ใช้จะเลือกได้เพียงตัวเลือกเดียวเท่านั้น



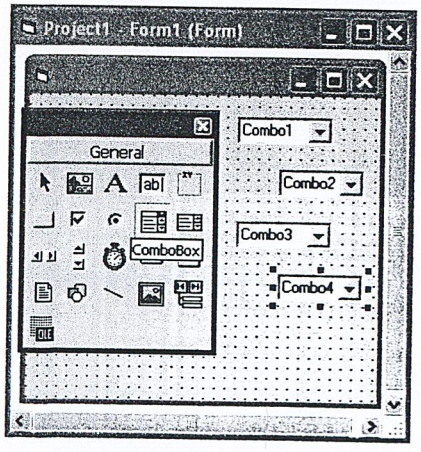
รูปที่ 2.42 คอนโทรลออฟชั่นบัตตอนบนทูลบ๊อกซ์ และที่ปรากฏอยู่บนฟอร์ม

คุณสมบัติที่สำคัญคือ

- **Value** ใช้กำหนดสถานะของออปชั่นบัตตอนว่าถูกเลือกหรือไม่ ถ้าถูกเลือกจะมีค่าเป็น True ถ้าไม่ถูกเลือกจะมีค่าเป็น False
- **Caption** ถ้าข้อความที่กำหนดไว้ข้างออปชั่นบัตตอนเพื่อบอกให้ผู้ใช้รู้
- **Enable** เหมือนกับคอนโทรลอื่นที่กล่าวมาแล้ว

คอมโบบ๊อกซ์ (Combo Box)

คอมโบบ๊อกซ์เป็นคอนโทรลที่แสดงรายการให้ผู้ใช้เลือกใช้ โดยจะสามารถเลือกได้ครั้งละ1 รายการ ถ้ารายการที่แสดงในคอมโบบ๊อกซ์มีมากเกินกว่าที่แสดงได้ทั้งหมด จะมีสกอลบาร์ในแนวตั้งขึ้นมาให้เราเลื่อนดูรายการที่เหลือได้



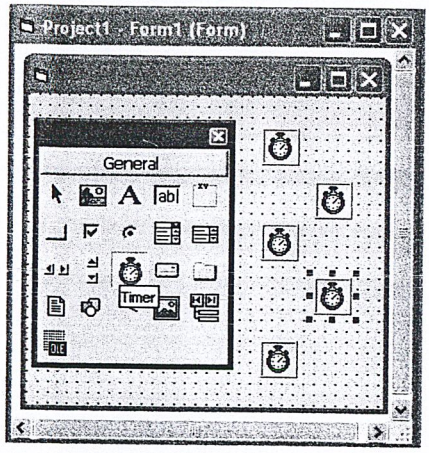
รูปที่ 2.43 คอนโทรลคอมโบบ๊อกซ์เมื่อปรากฏบนทูลบ๊อกซ์ และอยู่บนฟอร์ม

คุณสมบัติที่สำคัญ คือ

- **Enable** เหมือนกับคอนโทรลอื่นที่กล่าวมาแล้ว

ไทม์เมอร์ (Timer)

จุดประสงค์ของการใช้คอนโทรลไทม์เมอร์ คือ จะให้มีการทำงานอย่างเป็นทุกๆ ช่วงเวลาที่เรากำหนด ซึ่งช่วยให้การทำงานบางอย่างถูกทำแบบฉากหลังพร้อมกับมีโปรแกรมอื่นทำงานอยู่ ด้วยคอนโทรลนี้เราไม่สามารถมองเห็นได้ เมื่อรันโปรแกรม (เรียกคอนโทรลประเภทนี้ว่า Non Visual Interface) ซึ่งก็คือจะไม่เห็นตัวคอนโทรลในขณะที่รันโปรแกรมขึ้นมา



รูปที่ 2.44 แสดงคอนโทรลไทล์เมอร์บนทูลบ๊อกซ์ และปรากฏอยู่บนฟอร์ม

คุณสมบัติที่สำคัญ คือ

- **Interval** เป็นคุณสมบัติบอกว่าต้องการให้เกิดอีเวนต์ Timer เว้นช่วงเวลาห่างกันเป็นเท่าไร โดยมีหน่วยเป็นมิลลิวินาที (1/1000 วินาที)
- **Enable** เหมือนกับคอนโทรลอื่นที่กล่าวมาแล้ว

กราฟ (Graph 32.ocx)

คอนโทรลตัวนี้อยู่ใน CD แผ่นที่ 3 ของโปรแกรม Visual Studio เวอร์ชั่น 6.0 ซึ่งไฟล์นี้อยู่ในโฟลเดอร์ \COMMON\TOOLS\VB\CONTROLS ซึ่งจะก่อนใช้คอนโทรลนี้เราจะต้อง Register คอนโทรลนี้โดยเรียกไฟล์ VBCtrls.reg เพื่อให้สามารถนำคอนโทรลนี้มาใช้ได้

คอนโทรล Graph นี้จะสามารถแสดงกราฟได้หลายรูปแบบ เช่น กราฟแท่งก็ได้ ในโครงงานนี้จะใช้กราฟเส้นแสดงผล ซึ่งทำให้ดูผลได้ง่าย, ชัดเจน และใกล้เคียงกับสัญญาณ อินพุตมากที่สุด การกำหนดลักษณะของกราฟทำได้โดยผ่านทางคุณสมบัติ Graph Type

คำสั่งที่สำคัญที่ใช้ในการสร้างกราฟ คือ

Object Name. This Point = I

เป็นการกำหนดจุดที่ใส่ข้อมูล

Object Name. Graph Data = n

เป็นการกำหนดข้อมูลบนแกน Y ที่ตำแหน่ง I

โดยที่ Object Name คือชื่อของออปเจ็กต์กราฟ

I และ n เป็นค่าตัวเลข

Microsoft Comm Control

คอนโทรล MSComm (Communications) เป็นคอนโทรลที่ช่วยในการติดต่อพอร์ตอนุกรม (Serial Port) ซึ่งสามารถทำการส่งข้อมูลผ่านทางพอร์ตอนุกรมได้ด้วยคอนโทรลนี้ เช่น การติดต่อโดยตรงกับบอร์ดอิเล็กทรอนิกส์ ซึ่งคอนโทรล MSComm ที่มากับ Visual Basic จะเป็นคอนโทรลที่ทำงานโดยมีการตอบสนองต่อเหตุการณ์แบบ Event-Driven นั่นคือ คอนโทรลจะทำหน้าที่ในการตรวจสอบการเกิดขึ้น หรือการร้องขอให้เกิดเหตุการณ์ต่างๆ กับพอร์ทอนุกรมโดยอัตโนมัติ และจะมีการแจ้งเตือนให้ทราบทางโพรซีเยอร์เหตุการณ์ จะเช่นเดียวกับคอนโทรลทั่วไปของ Visual Basic นั่นเอง ดังนั้นในการเขียนโค้ดจึงไม่จำเป็นต้องสร้างโพรซีเยอร์ที่ทำหน้าที่คอยตรวจสอบเหตุการณ์ต่างๆ ของพอร์ตอนุกรม ซึ่งจะทำให้ง่ายต่อการใช้งานเป็นอย่างมาก

การเพิ่มคอนโทรล Microsoft Comm Control จะทำได้โดยการเรียกหน้าต่าง Components แล้วทำการเลือกที่ Microsoft Comm Control 6.0

คอนโทรล MSComm มีหน้าที่หลักๆ สำหรับการสื่อสารผ่านพอร์ตอนุกรมดังต่อไปนี้

- หมุนหมายเลขติดต่อกับโทรศัพท์ปลายทางที่กำหนด
- ตรวจสอบการเข้ามาของข้อมูลยังพอร์ตอนุกรมโดยอัตโนมัติ
- ส่งข้อมูลตามที่กำหนดจากโปรแกรมไปยังพอร์ตอนุกรม

ในความเป็นจริงคอนโทรล MSComm ไม่ได้ทำหน้าที่ติดต่อกับพอร์ตอนุกรมโดยตรงแต่ว่าจะทำหน้าที่เรียกใช้ฟังก์ชั่นวินโดวส์ API ซึ่งวินโดวส์นี้จะทำการส่ง หรือรับข้อมูลผ่านทางพอร์ตอนุกรมโดยอาศัยไดรเวอร์ Comm.drv อีกทอดหนึ่ง ดังนั้นจึงสามารถสรุปสั้นๆ ได้ว่า ทุกครั้งที่มีการเรียกใช้คอนโทรล MSComm ก็หมายถึงเรียกใช้ฟังก์ชั่นวินโดวส์ API ซึ่งจะถูกตีความอีกทอดหนึ่ง โดยไดรเวอร์ Comm.drv จากนั้นก็จะส่งผ่านข้อมูลที่ถูกจัดรูปแบบตามมาตรฐานการสื่อสาร (ทั้งนี้ก็ขึ้นกับอุปกรณ์ที่ต่อเข้ากับพอร์ตอนุกรม) ให้กับดีไวซ์ไดรเวอร์อีกทอดหนึ่งนั่นเอง

เหตุการณ์ที่สำคัญของคอนโทรล MSComm คือ CommEvent จะรายงานเหตุการณ์ทุกครั้งที่เกิดข้อผิดพลาดหรือมีการสื่อสาร ซึ่งสามารถอ่านค่าได้เฉพาะขณะทำงานเท่านั้นรูปแบบคำสั่งคือ

Object.CommEvent

คอนโทรล MSComm จะมีการเรียกโพรซีเยอร์เหตุการณ์ OnComm ทุกครั้งที่เกิดผิดพลาดหรือมีการสื่อสารเกิดขึ้น ซึ่งค่าตัวเลขที่เป็นจำนวนเต็ม แสดงถึงข้อผิดพลาดหรือเหตุการณ์ที่มีการสื่อสารดังกล่าว ก็จะถูกจัดเก็บเอาไว้ในคุณสมบัติ CommEvent เสมอ ดังนั้น ถ้าหากต้องการคุณสมบัติที่สำคัญ คือ

- **CommPort** เป็นรายงาน หรือกำหนดหมายเลขของพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ที่ต้องการติดต่อ สำหรับหมายเลขของพอร์ตอนุกรมสามารถมีค่าได้ตั้งแต่ 1 ถึง 16 (ค่าปกติจะเท่ากับ 1) ซึ่งก่อนจะเปิดพอร์ตด้วยคุณสมบัติ Port Open ต้องมีการกำหนดหมายเลขของพอร์ตอนุกรมให้กับคุณสมบัติ CommPort ไม่เป็นความจริง ก็จะเกิดข้อผิดพลาด (Device Unavailable) ทันที ซึ่งในกรณีนี้สามารถแก้ไขได้โดยการกำหนดหมายเลขที่มี ของพอร์ตอนุกรมที่ถูกต้องเสียใหม่ แล้วจึงทำการเปิดพอร์ตอนุกรมอีกครั้งด้วยคุณสมบัติ Port Open

ตารางที่ 2.4 แสดงการตรวจเหตุการณ์ (Event)

ค่าคงที่	ค่าตัวเลข	รายละเอียด
ComEvSend	1	มีจำนวนตัวอักษรในบัฟเฟอร์ด้านส่งออกข้อมูลน้อยกว่าจำนวนตัวอักษรที่กำหนดในคุณสมบัติ
ComEvReceive	2	การรับเข้าจำนวนตัวอักษรที่ถูกกำหนดในคุณสมบัติ Rthreshold ซึ่งจะเกิดขึ้นอย่างต่อเนื่องจนกว่าผู้อ่านจะใช้คุณสมบัติ Input ในการอ่านข้อมูลจากบัฟเฟอร์สำหรับรับข้อมูล
ComEvCTS	3	มีการเปลี่ยนสถานะของสายสัญญาณ Clear to Send
ComEvDSR	4	มีการเปลี่ยนสถานะของสายสัญญาณ Data Set Ready ซึ่งเหตุการณ์นี้จะเกิดขึ้นเมื่อมีการเปลี่ยนสถานะของสายสัญญาณ Data Set Ready จาก 1 เป็น 01 เท่านั้น
	5	มีการเปลี่ยนสถานะของสายสัญญาณ Carrier Data
Com EvRing	6	มีการตรวจพบการเรียกหมายเลข

- **InBuffer Size** เป็นรายงาน หรือกำหนดขนาดบัฟเฟอร์ด้านที่รับเข้า ซึ่งมีหน่วยเป็นไบต์ (โดยปกติ 1 ไบต์จะเท่ากับ 1 ตัวอักษร) การเลือกขนาดของบัฟเฟอร์ด้านรับเข้าที่เหมาะสมนั้นในทางปฏิบัติเป็นสิ่งที่ยากมาก ทั้งนี้ขึ้นอยู่กับความหนาแน่นของข้อมูลที่มีการส่งไปมาในแต่ละครั้งและความเร็วของการสื่อสาร (Transmission Rate) ของพอร์ต

- **Input** คือ การรับข้อมูลจากบัฟเฟอร์ ซึ่งสามารถจะกำหนดค่าได้เฉพาะในขณะทำงานในรูปแบบการใช้งาน คือ

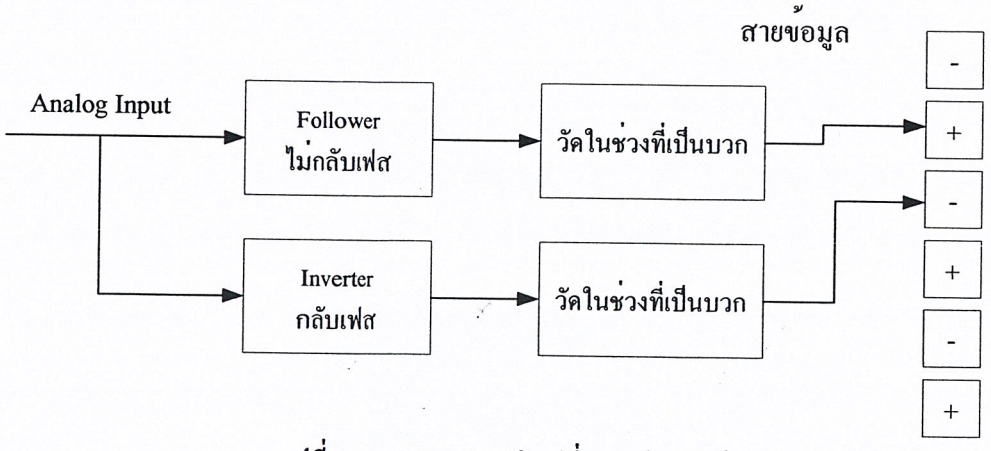
Object.Input

ทุกครั้งที่มีการใช้คุณสมบัติอินพุต ในการอ่านข้อมูลจากบัฟเฟอร์ด้านรับเข้ามานั้น จำนวนของตัวอักษรที่อ่านได้จะถูกกำหนดลงในคุณสมบัติอินพุตเลนทันที ซึ่งถ้าหากว่ากำหนดให้คุณสมบัติอินพุตเลนมีค่าเท่ากับ 0 ซึ่งจะหมายถึงการกำหนดให้คุณสมบัติอินพุตอ่านข้อมูลทั้งหมดจากบัฟเฟอร์ด้านรับเข้านั่นเอง ซึ่งชนิดของข้อมูลที่อ่าน โดยคุณสมบัติอินพุตจะเป็นข้อมูลแบบข้อความ หรือไบนารี ก็ขึ้นกับการกำหนดค่าของคุณสมบัติอินพุตโหมด

- **Output** คือ การส่งข้อมูลไปยังบัฟเฟอร์ด้านส่งออก ซึ่งกำหนดค่าได้เฉพาะในขณะมีการทำงานเท่านั้น รูปแบบการใช้งาน คือ

Object.Output [=Value]

สำหรับชนิดของข้อมูลที่ถูกส่งโดยคุณสมบัติเอาต์พุต จะเป็นข้อมูลแบบข้อความหรือไบนารี ก็ขึ้นกับการกำหนดค่าของคุณสมบัติอินพุตโหมด ดังที่กล่าวมาแล้วข้างต้น



รูปที่ 2.45 แสดงการส่งเปลี่ยนของสายข้อมูล

หน้าที่หลักของโปรแกรมด้านเซิร์ฟเวอร์ คือ ส่งสัญญาณให้ชุดอ่านสัญญาณ เริ่มต้นอ่านสัญญาณในช่วงเวลาหนึ่งจนครบตามจำนวนที่ได้กำหนดเป็นกรอบข้อมูลในที่นี้ใช้ขนาด 66 ไบต์ และเมื่อได้รับการตอบกลับจากชุดอ่านสัญญาณ ก็จะอ่านข้อมูลในบัฟเฟอร์เข้าไปในโปรแกรมจำนวน 66 ไบต์ ค่าที่ได้จะเป็นค่าแบบไบนารี ซึ่งมีค่า ตั้งแต่ 00H จนถึง FFH แล้วนำค่าที่ได้มาแปลงเป็นค่าแรงดัน 0 ถึง 500 มิลลิโวลท์ (ตามค่าของแรงดันอ้างอิง ของ ADC) ตามสมการ

$$V = B = (((B - 0)*5000)/255)$$

โดย V คือ ค่าแรงดันที่แปลงได้ ในช่วง 0 – 5000 มิลลิโวลท์

B คือ ค่าไบนารีที่รับมาจากชุดวัดสัญญาณ

ค่าที่ได้เป็นค่าบวกทั้งหมด แต่โปรแกรมจะเข้าใจว่าค่าที่ได้เป็นค่าบวกสลับกับค่าลบ โดยที่ลำดับที่เป็นคู่ (0, 2, 4, ...) เป็นค่าของ ADC ตัวแรกที่วัดค่าช่วงบวก ถ้าเป็นศูนย์แสดงว่าตำแหน่งนั้นเป็นค่าลบให้ทำการประมาณค่า โดยนำเอาค่าข้างเคียงทั้งสองมาประมาณค่า

$$V(I) = (V(I - 1) + V(I+1)) / 2$$

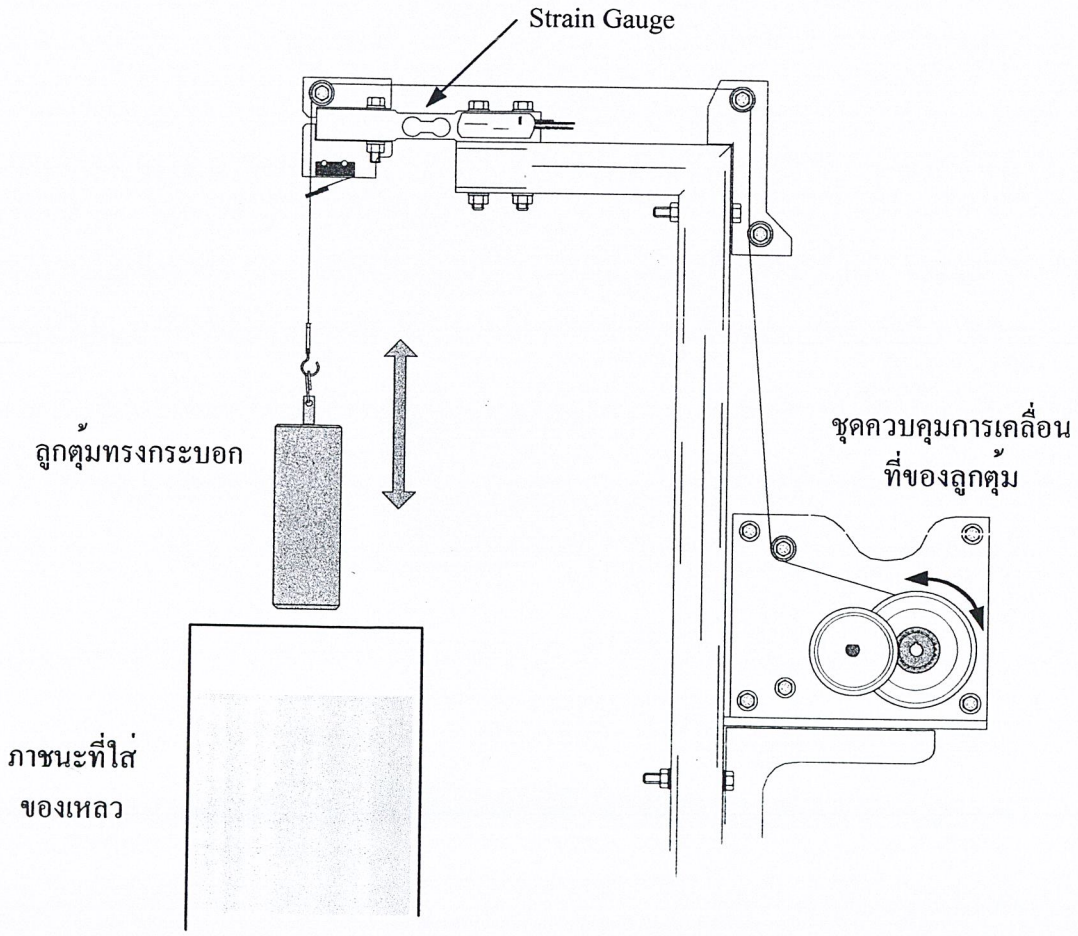
โดยที่ I เป็นตัวชี้ตำแหน่งลำดับที่ของข้อมูล (Index)

และลำดับที่เป็นที่ (1, 3, 5, ...) เป็นค่าของ ADC ตัวที่สองที่วัดค่าช่วงลบ ถ้าเป็นศูนย์แสดงว่าตำแหน่งนั้นเป็นค่าบวก ให้ทำการประมาณค่าโดยเอาค่าข้างเคียงทั้งสองมาประมาณค่าเช่นเดียวกับข้างต้น

บทที่ 3
หลักการและทฤษฎีที่ใช้ในการออกแบบ

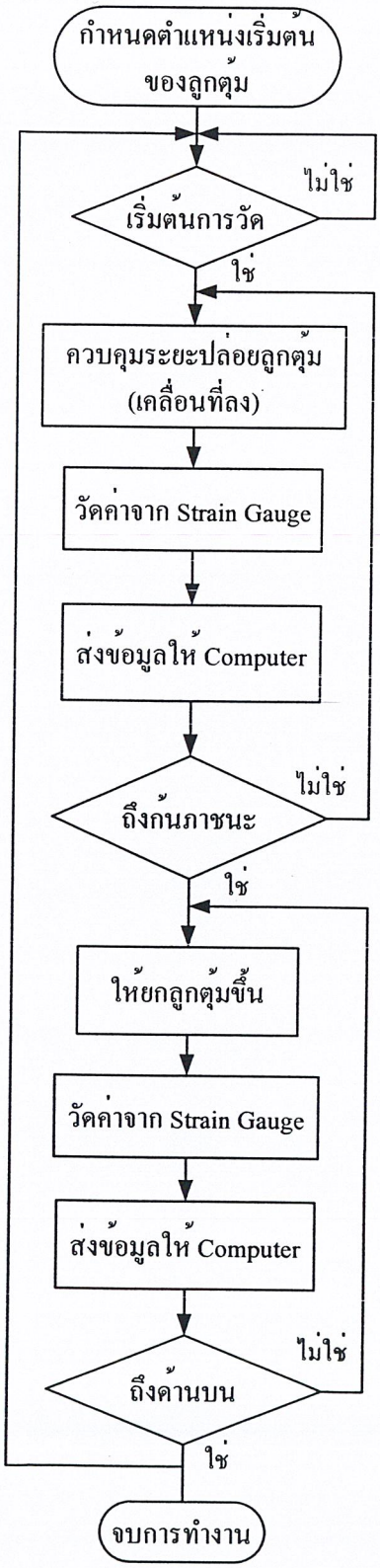
3.1 หลักการการทำงาน

หลักการในการออกแบบ ใช้หลักการการเปลี่ยนแปลงค่าความเครียด (Strain) ที่เกิดขึ้นเมื่อวัตถุทุกแขวนอยู่กับตัว Strain Gauge ที่มีการเปลี่ยนแปลงค่าตลอด เมื่อน้ำหนักของวัตถุที่แขวนอยู่เปลี่ยนแปลง เพื่อวัดการเปลี่ยนแปลงที่เกิดขึ้นของน้ำหนักของวัตถุในสภาวะที่อยู่ในของเหลว แล้วนำสัญญาณที่ได้ ที่ถูกแปลงให้เป็นสัญญาณที่เหมาะสมแล้วส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ เพื่อบันทึกค่าและประมวลผล และจะมีตัวควบคุมระยะการเคลื่อนที่ของวัตถุ โดยใช้การควบคุมมอเตอร์สเต็ปปิ้ง มีการควบคุมการทำงานด้วยไมโครคอนโทรลเลอร์



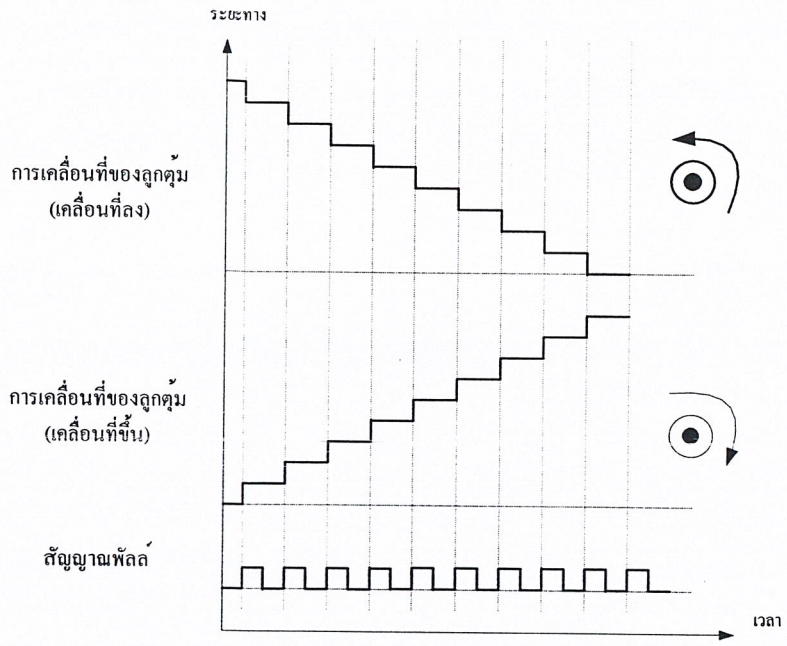
รูปที่ 3.1 แสดงโครงสร้างของเครื่องค่าความถ่วงจำเพาะที่ได้ออกแบบ

โดยหลักการ การทำงานของเครื่องวัดค่าความถ่วงจำเพาะ อธิบายตามลำดับการทำงานตามรูปที่ 3.2

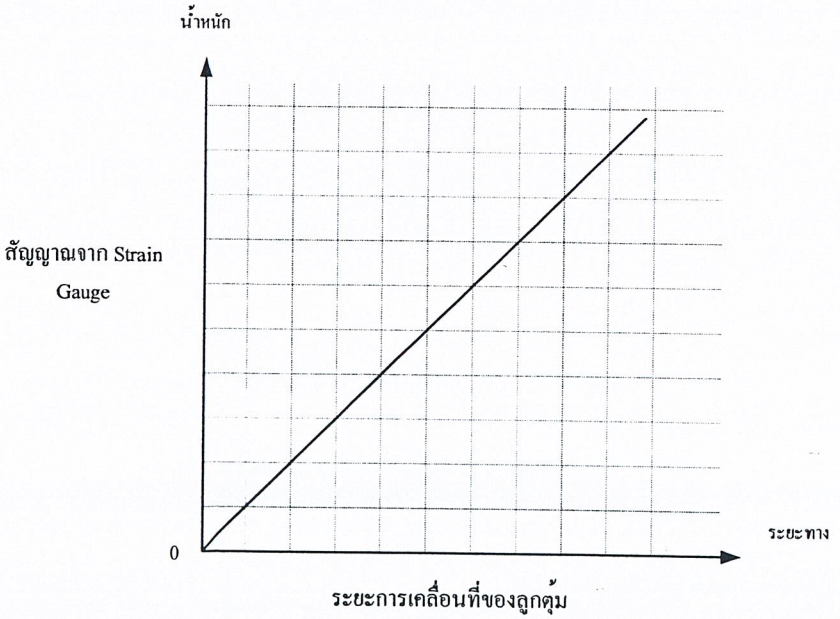


รูปที่ 3.2 แสดง Flow Chart ของการทำงานของเครื่อง

โดยไมโครคอนโทรลเลอร์นั้นจะทำหน้าที่ในการควบคุมการจ่ายสัญญาณพัลส์และควบคุมทิศทางการหมุนให้กับสเต็ปปิ้งมอเตอร์ และรับค่าสัญญาณที่ได้จากตัว Strain Gauge ที่มีการแปลงสัญญาณให้เหมาะสมแล้ว มาทำบันทึกและประมวลผลของความสัมพันธ์ระหว่างพัลส์ที่จ่ายให้กับสเต็ปปิ้งมอเตอร์ เมื่อเทียบเป็นระยะทางกับสัญญาณที่ได้รับจากตัว Strain Gauge เมื่อเทียบเป็นน้ำหนักของวัตถุที่แขวนอยู่ (ลูกตุ้ม)



รูปที่ 3.3 แสดงสัญญาณพัลส์ที่จ่ายให้มอเตอร์สเต็ปปิ้ง กับระยะการเคลื่อนที่ของลูกตุ้ม



รูปที่ 3.4 แสดงความสัมพันธ์ของระยะการเคลื่อนที่ของลูกตุ้ม (เมื่ออยู่ในของเหลว) กับน้ำหนักที่ได้จากลูกตุ้ม (ในอากาศให้เริ่มที่ 0)

เมื่อความสัมพันธ์ของสัญญาณพัลส์ที่จ่ายให้กับสเต็ปปิ้งมอเตอร์นั้นเป็นไปตามแกน x และสัญญาณที่ได้รับจากตัว Strain Gauge เป็นไปตามแกน y นั้นจะเป็นไปตามทฤษฎีของอาร์คิมิดีส

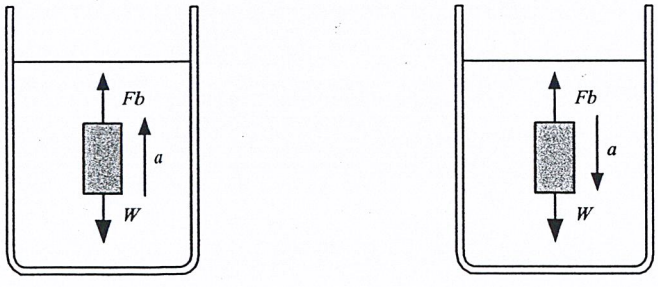
3.2 ทฤษฎีและหลักการอ้างอิง

หลักการอาร์คิมิดีสมีใจความสรุปได้ว่า

" วัตถุที่จมในของเหลว ไม่ว่าจมหมดทุกส่วนหรือจมเป็นบางส่วนก็ตาม จะได้รับแรงพยุงขึ้นด้วยปริมาณแรงที่มีขนาดเท่ากับน้ำหนักของของเหลวเฉพาะส่วนที่ถูกแทนด้วยวัตถุนั้นๆ "

ในการทดสอบหลักการของอาร์คิมิดิสวิธีหนึ่งคือ ทำการยกหรือชั่งวัตถุที่จมอยู่ในน้ำซึ่งจะพบว่าน้ำหนักของวัตถุเบากว่าเมื่อทำการยกหรือชั่งในอากาศเนื่องจากน้ำช่วยออกแรงพยุง หรือแรงลอยตัว (Buoyant Force) ในทิศทางขึ้นบน จึงทำให้วัตถุดังกล่าวเบากว่าปกติ แรงพยุงนี้จะมีขนาดเท่ากับน้ำหนักของของเหลวส่วนที่ถูกแทนที่ด้วยวัตถุ ดังนั้น ถ้าให้ F_B เป็นขนาดของแรงพยุง V ปริมาตรของของเหลวที่เท่ากับปริมาตรส่วนที่จมของวัตถุ และ ρ_f เป็นความหนาแน่นของของเหลว จะได้ $F_B = \rho_f gV$

กรณีที่วัตถุมีปริมาตร V_0 มีความหนาแน่น ρ_0 จมในของเหลวหมดทั้งก้อน ดังรูปที่ 3.5 จะได้ขนาดของแรงลัพธ์ที่กระทำต่อวัตถุเป็น $F_B - W$ แต่ด้วยเนื่องจากว่าขนาดของแรงลอยตัว $F_B = \rho_f gV$ และขนาดน้ำหนักของวัตถุเป็น $W = mg = \rho_0 V_0 g$ จะได้ $F_B - W = (\rho_f - \rho_0)V_0 g$ นั้นก็คือ ถ้าความหนาแน่นของวัตถุน้อยกว่าความหนาแน่นของของเหลว $(\rho_f > \rho_0)$ วัตถุนั้นจะลอยขึ้นสู่ผิวของของเหลว แต่ถ้าหากความหนาแน่นของวัตถุดังกล่าวมากกว่าความหนาแน่นของของไหล $(\rho_0 > \rho_f)$ วัตถุจะเคลื่อนที่ตกลงในของเหลว ตามรูปที่ 3.5 (ก) และ (ข)

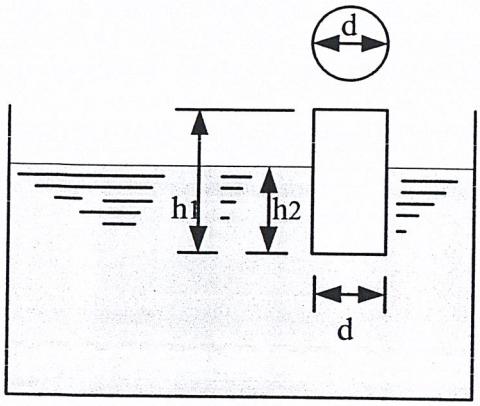


(ก) (ข)

รูปที่ 3.5 (ก) และ (ข) แสดงวัตถุที่จมอยู่ในภาชนะที่มีของเหลวอยู่

โดยรูปที่ 3.5 (ก) เมื่อวัตถุที่มีความหนาแน่นน้อยกว่าของเหลวจมในของเหลวทั้งก้อน จะมีแรงลัพธ์ (ผลต่างของแรงพยุงกับน้ำหนักของวัตถุ) กระทำต่อวัตถุในทิศทางขึ้นสู่บนผิวของเหลว และรูปที่ 3.5 (ข) ถ้าความหนาแน่นของวัตถุมากกว่าความหนาแน่นของของเหลว โดยจะมีแรงลัพธ์กระทำในทิศทางลง วัตถุจึงจมสู่ก้นภาชนะที่บรรจุของเหลวนั้นๆ

ถ้าเราพิจารณาในกรณีที่วัตถุจมในของเหลวเป็นบางส่วน จะเกิดสภาวะสมดุลของแรงพยุงกับน้ำหนักของวัตถุนั้น สมมติว่าวัตถุมีปริมาตร V_0 จมในของเหลวบางส่วนโดยมีปริมาตรส่วนที่จมเป็น V ดังนั้น ขนาดของแรงพยุงในทิศทางขึ้นบนย่อมเท่ากับขนาดน้ำหนัก (แรงโน้มถ่วงของโลก) ของวัตถุในทิศทางลง



รูปที่ 3.6 ลูกตุ้มทรงกระบอกที่จุ่มลงไปในภาชนะที่บรรจุของเหลว

จากรูปที่ 3.6 จะได้ความสัมพันธ์ของลูกตุ้มทรงกระบอก ที่จมอยู่ในภาชนะที่บรรจุของเหลวจะได้สมการดังนี้

แรงพยุง = น้ำหนักวัตถุเมื่อชั่งในอากาศ - น้ำหนักวัตถุเมื่อชั่งในของเหลว

$$F_X = mg - \rho h_2 \frac{\pi d^2}{4} \qquad (3.1)$$

เมื่อ F_X = แรงพยุงหรือแรงผลลัพธ์ทางแนวตั้ง

ρ = ความหนาแน่นของของเหลว

$\frac{\pi d^2}{4}$ = พื้นที่หน้าตัดของวัตถุทรงกระบอก

g = ความโน้มถ่วงของโลก

m = มวลของวัตถุทรงกระบอก

บทที่ 4

ผลการทดลอง

4.1 กล่าวนำ

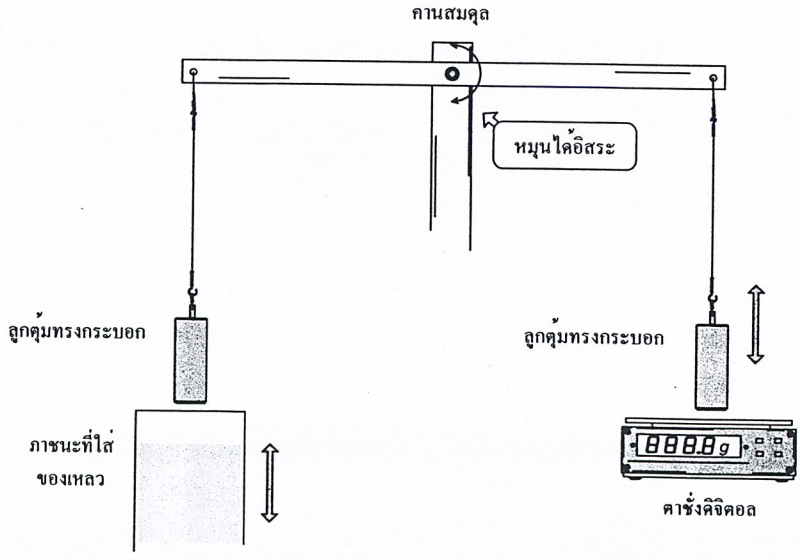
จากความสัมพันธ์ต่างๆ ตามทฤษฎีนั้น ยังสรุปไม่ได้ว่าเป็นจริงตามทฤษฎีหรือไม่ จึงต้องมีการหาข้อสรุปที่เราต้องการ โดยการสร้างเครื่องมือ หรือชุดทดลองขึ้นมา โดยในตอนที่ 1 จะเป็นการทดลองเบื้องต้นเพื่อหาความสัมพันธ์ที่เราได้จากทฤษฎี และในตอนที่ 2 จะเป็นผลทดลองที่ได้จากเครื่องวัดที่สร้างขึ้นมา

4.2 การทดลองตอนที่ 1

จากรูปที่ 4.1 ทำการทดลองหาค่าระยะที่ลูกตุ้มจมลงไปในของเหลว และน้ำหนักที่หายไปของลูกตุ้มที่จมไปในของเหลว (ดูที่ตาชั่งดิจิตอล)

โดยอุปกรณ์ที่ใช้ในการทดลองมีดังนี้

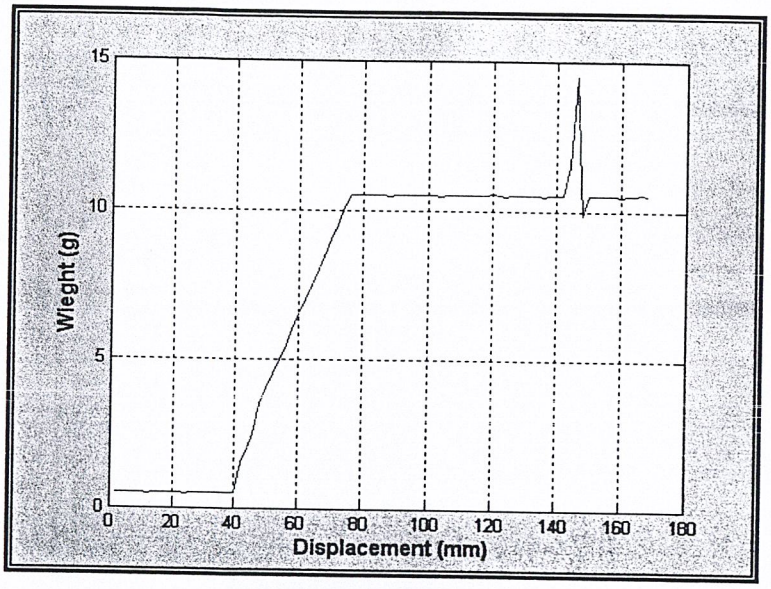
1. ลูกตุ้มทองเหลืองรูปทรงกระบอก	2	ลูก
2. คานสมดุลที่มีการหมุนได้อย่างอิสระ	1	อัน
3. ภาชนะใส่ของเหลว	1	อัน
4. ตาชั่งดิจิตอล	1	เครื่อง



รูปที่ 4.1 ชุดทดลองเบื้องต้น เพื่อหาความสัมพันธ์ของน้ำหนักที่เปลี่ยนแปลงเมื่ออยู่ในของเหลว

ในการทดลองจะเป็นการเคลื่อนภาชนะที่บรรจุของเหลวขึ้น ทำการทดลองของเหลวแต่ละชนิด แล้วนำค่าที่ได้มาพล็อตกราฟความสัมพันธ์ โดยเราให้แกน x เป็นค่าระยะที่ลูกตุ้มอยู่ในของเหลวมีหน่วยเป็นมิลลิเมตร และให้แกน y เป็นค่าของน้ำหนักที่หายไปของลูกตุ้ม (แรง-พยุง) ที่ปรากฏอยู่ที่ตาชั่งดิจิตอลมีหน่วยเป็นกรัม

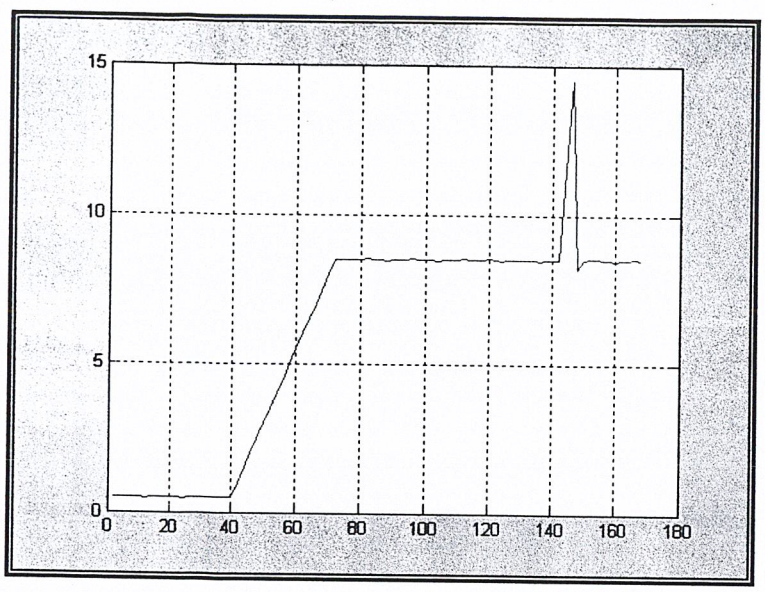
ผลการทดลอง



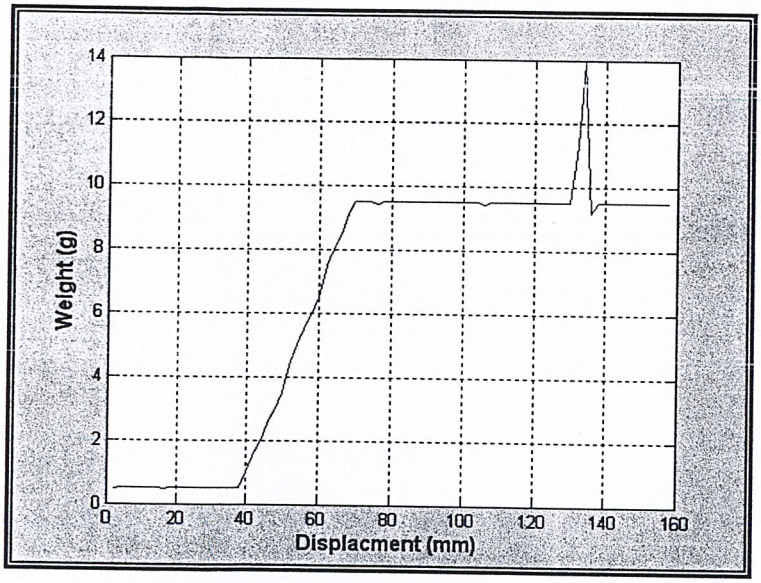
รูปที่ 4.2 ของเหลวที่ทดลองเป็นน้ำ

โดยในกราฟที่ทำการพล็อตขึ้นมานี้ สามารถอธิบายความสัมพันธ์ที่ได้ออกมาเป็นช่วงๆ ได้ดังนี้ โดยจะยกตัวอย่างกราฟรูปที่ 4.2

1. ช่วงที่เริ่มทำการวัด คือ ช่วงที่ลูกตุ้มยังไม่ได้จุ่มลงไปในของเหลว (คือยังอยู่ในอากาศ) ค่าน้ำหนักที่ได้จากตาชั่งดิจิตอลจะมีค่าเท่ากับ 0 g (จากกราฟที่เราทำการพล็อตออกมานั้นเราจะยกระดับของน้ำหนักขึ้นเพื่อให้เห็นการเปลี่ยนแปลง)
2. ช่วงที่ 2 เป็นช่วงที่รูปกราฟที่ได้นั้นมีการเปลี่ยนแปลง โดยกราฟที่ได้จะมีความชันขึ้น (น้ำหนักเพิ่มขึ้น เมื่อระยะเพิ่มขึ้น) จะเป็นช่วงที่ลูกตุ้มทรงกระบอกถูกจุ่มลงไปในของเหลวเรื่อยๆ
3. ช่วงที่ 3 เป็นช่วงที่กราฟไม่มีการเปลี่ยนแปลง (น้ำหนักคงที่ เมื่อระยะเพิ่มขึ้น) เป็นช่วงที่บ่งบอกว่าลูกตุ้มทรงกระบอกนั้นได้จมลงไปในของเหลวทั้งก้อนแล้ว
4. ช่วงที่ 4 จะเป็นช่วงที่กราฟมีความชันสูงสุด จะเป็นช่วงที่ลูกตุ้มนั้นเคลื่อนที่ลงไปจนถึงก้นภาชนะที่บรรจุของเหลว ซึ่งทำให้น้ำหนักมีค่าสูงสุด



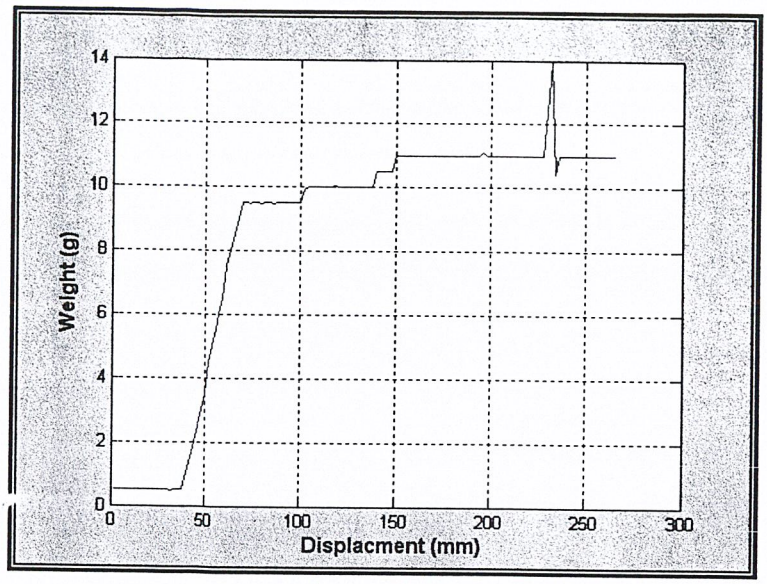
รูปที่ 4.3 ของเหลวที่ทดลองเป็นน้ำมันก๊าด



รูปที่ 4.4 ของเหลวที่ทดลองเป็นน้ำมันปาล์ม

หมายเหตุ

โดยลูกตุ้มที่เราใช้ในการทดลองนั้นเป็นลูกตุ้มทองเหลืองที่มีน้ำหนักเท่ากับ 89.5 กรัม และมีปริมาตรเท่ากับ 10.529 ลูกบาศก์เซนติเมตร



รูปที่ 4.5 ของเหลวที่ทดลองเป็นน้ำมันปาล์ม กับ น้ำ

จากรูปที่ 4.2 ถึง 4.4 จะเป็นกราฟที่คล้ายกัน แต่ในรูปที่ 4.5 จะมีความแตกต่างกันก็ตรงที่ของเหลวที่เราทำการทดลองเป็นของเหลวสองชนิด โดยชั้นแรกจะเป็นน้ำมันปาล์ม และชั้นที่ 2 จะเป็นน้ำ จากรูปที่ 4.5 จะบ่งบอกได้ว่า น้ำมันปาล์มมีความหนาแน่นของของเหลวน้อยกว่าน้ำ เพราะดูจากน้ำหนักที่เปลี่ยนแปลงในสองช่วง โดยจากความสัมพันธ์ที่ได้จากกราฟที่พล็อตจะเป็นความสัมพันธ์ระหว่างน้ำหนักต่อปริมาตร $\frac{g}{cm^3}$ ซึ่งจะเป็นค่าความหนาแน่นของของเหลวนั้นเอง

ตัวอย่าง จากรูปที่ 4.2, 4.3, 4.4

โดยค่าที่วัดได้เมื่อลูกตุ้มอยู่ในภาชนะที่บรรจุของเหลวคือ น้ำ, น้ำมันก๊าด, และน้ำมันปาล์ม จะมีน้ำหนักเท่ากับ 10.5 กรัม 8.5 กรัม และ 9.5 กรัม ตามลำดับ (ดูจากตาชั่งดิจิตอลและกราฟที่ทำการพล็อต)

ดังนั้น

$$\text{ความหนาแน่นของน้ำที่คำนวณ} = \frac{10.5g}{10.529cm^3}$$

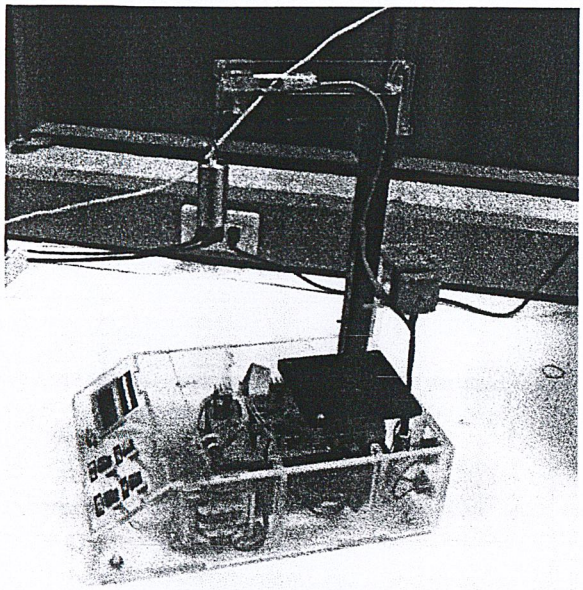
$$= 0.9972g/cm^3$$

$$\text{ความหนาแน่นของน้ำมันก๊าดที่คำนวณ} = \frac{8.5g}{10.529cm^3}$$

$$= 0.8072g/cm^3$$

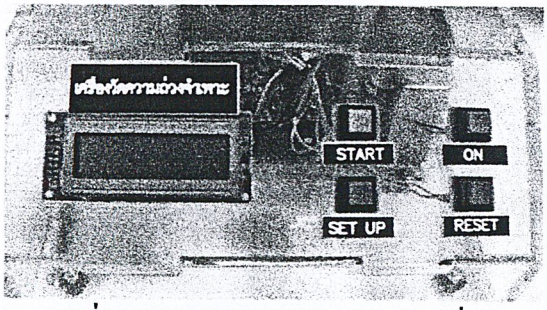
เมื่อความหนาแน่นของน้ำ และ น้ำมัดก๊าดมีค่าเท่ากับ $1g/cm^3$ และ $0.799g/cm^3$ ตามลำดับ โดยเราจะนำความสัมพันธ์ที่ได้จากการทดลองมาสร้างเครื่องวัดค่าความถ่วงจำเพาะในการตอนที่ 2

4.3 การทดลองตอนที่ 2



รูปที่ 4.6 โครงสร้างของเครื่องวัดค่าความถ่วงจำเพาะที่สร้างขึ้น

หลังจากสร้างเครื่องวัดความถ่วงจำเพาะขึ้นมาแล้วทำการทดลอง โดยทำการวัดของเหลวที่หาได้ง่าย และสามารถรู้ค่าความถ่วงจำเพาะของของเหลวนั้น เพื่อมาเปรียบเทียบกับค่าที่ได้จากเครื่องวัดค่าความถ่วงจำเพาะที่สร้างขึ้นมา

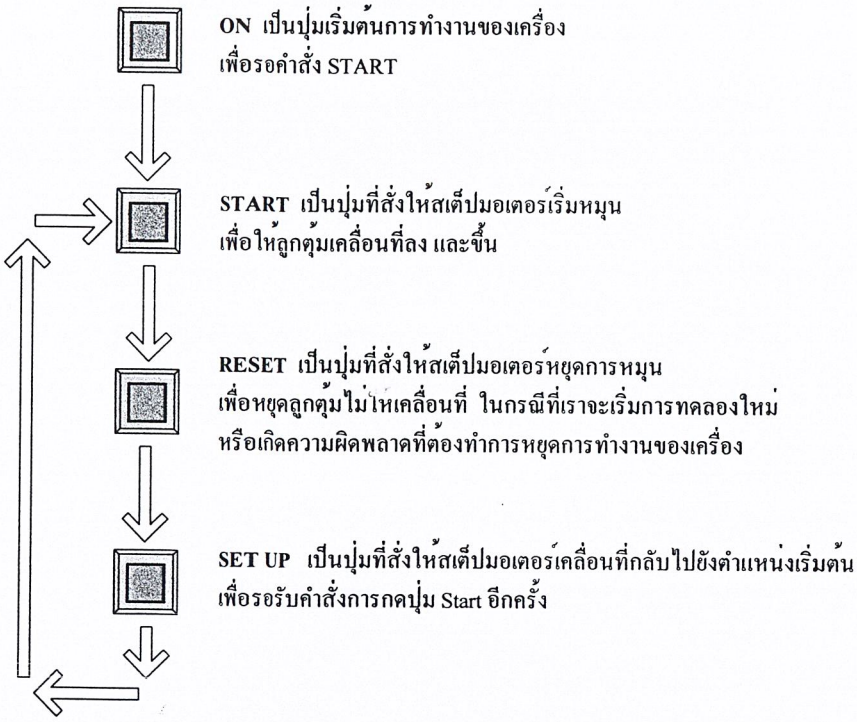


รูปที่ 4.7 แสดงปุ่มการทำงานของเครื่อง

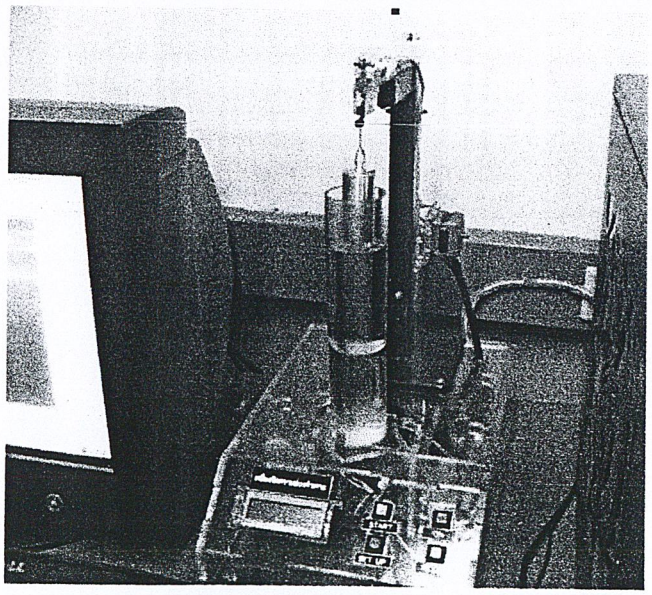
ลำดับขั้นการทดลอง

1. เลือกของเหลวที่จะทำการทดลอง โดยของเหลวนั้นจะต้องเป็นของเหลวที่รู้ค่าความถ่วงจำเพาะแล้วเพื่อจะได้มาทำการเปรียบเทียบกับค่าที่วัดได้จากเครื่องที่เราสร้างขึ้นมา
2. ใส่ของเหลวลงไปในภาชนะที่ใช้ทดลอง ดังรูปที่ 4.9 (รูปที่ 4.9 จะเป็นของเหลวสองชนิด คือ น้ำมันปาล์มกับน้ำ)

3. ทำการเปิดเครื่อง แล้วทำการทดลองตามรูปที่ 4.8 เป็นการแสดงวิธีการใช้เครื่อง

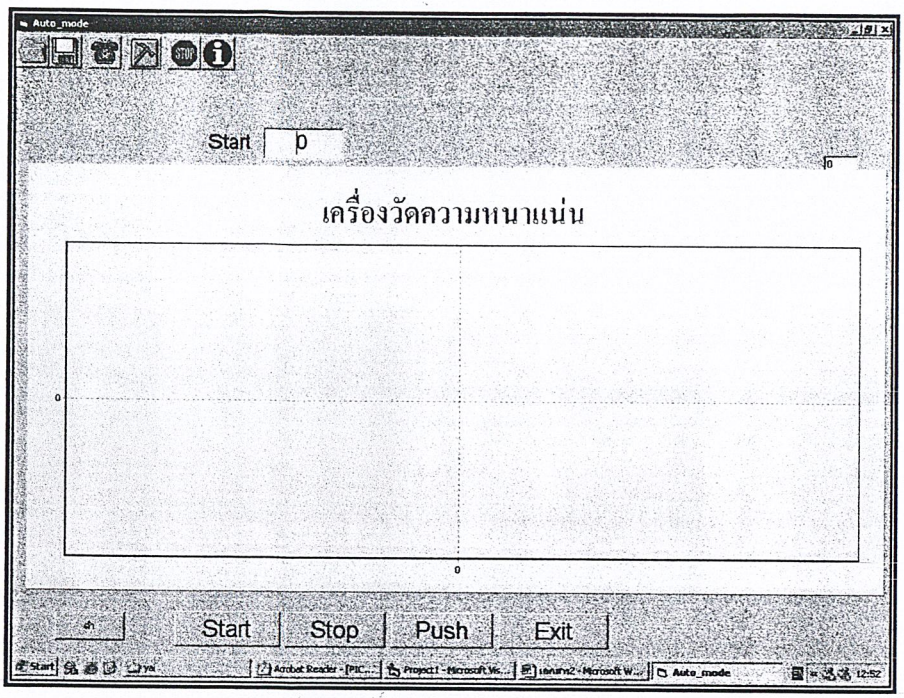


รูปที่ 4.8 แสดงวิธีการทำงานของเครื่องวัดค่าความถ่วงจำเพาะ

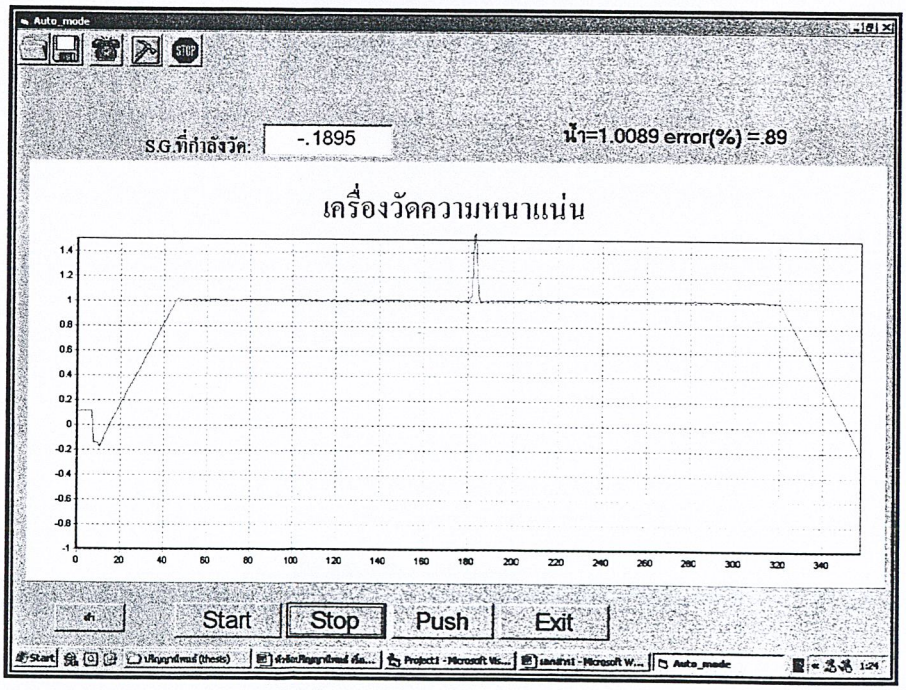


รูปที่ 4.9 แสดงการวัดของเหลวที่ต้องการจะรู้ค่าความถ่วงจำเพาะ

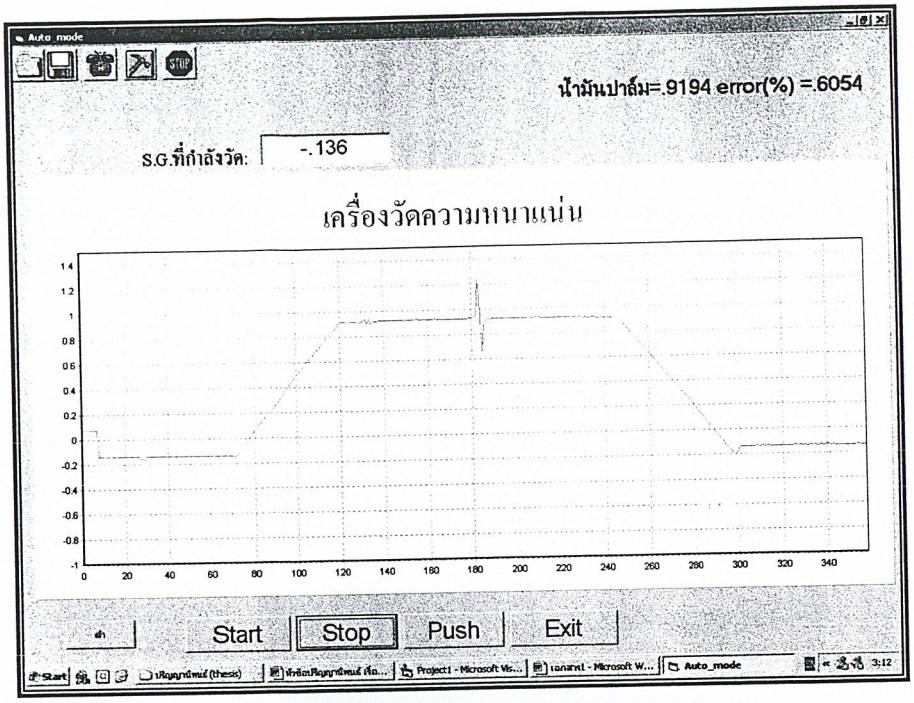
ผลการทดลอง



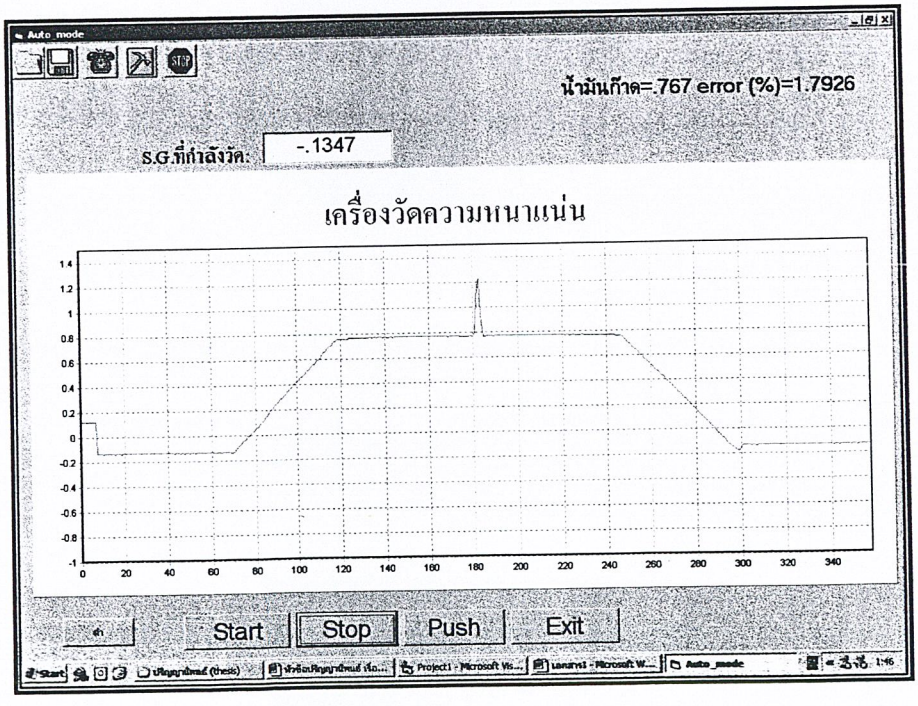
รูปที่ 4.10 แสดงหน้าจอการแสดงผลก่อนเริ่มการทดลอง



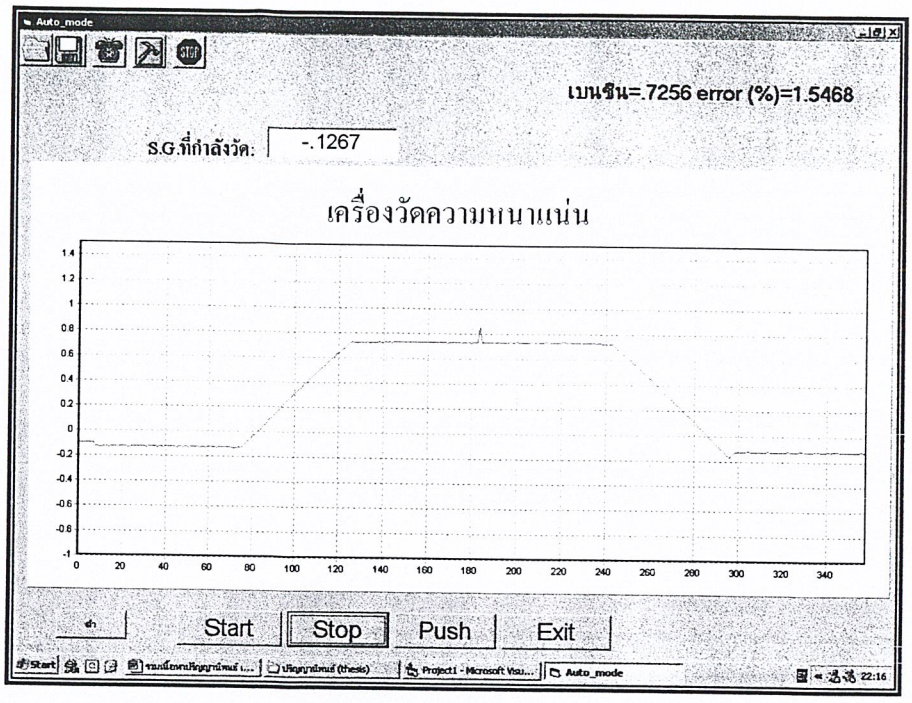
รูปที่ 4.11 ของเหลวที่ทดลองเป็นน้ำ



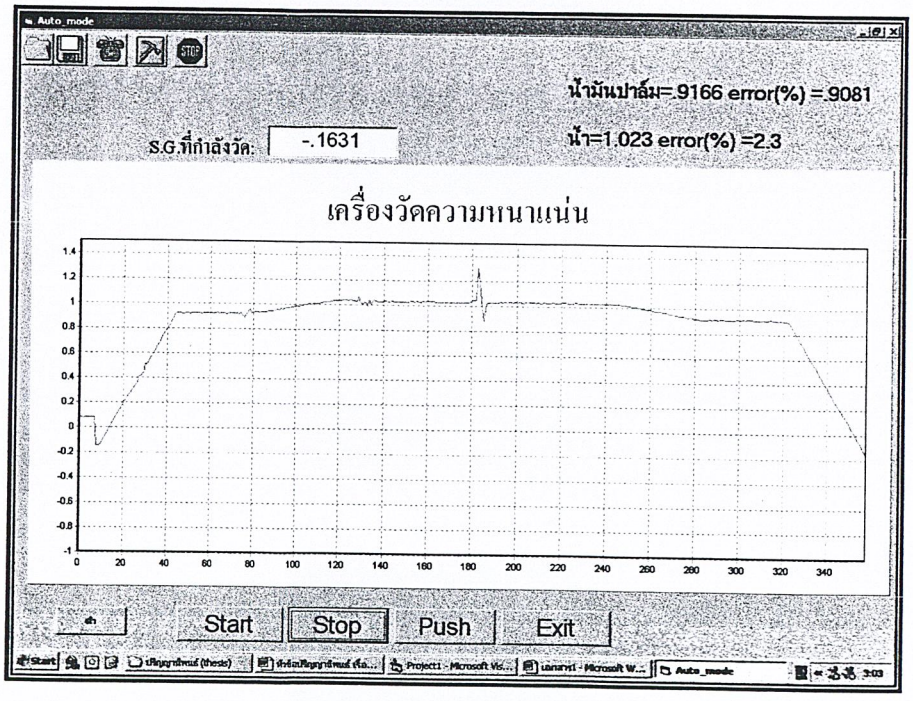
รูปที่ 4.12 ของเหลวที่ทดลองเป็นน้ำมันปาล์ม



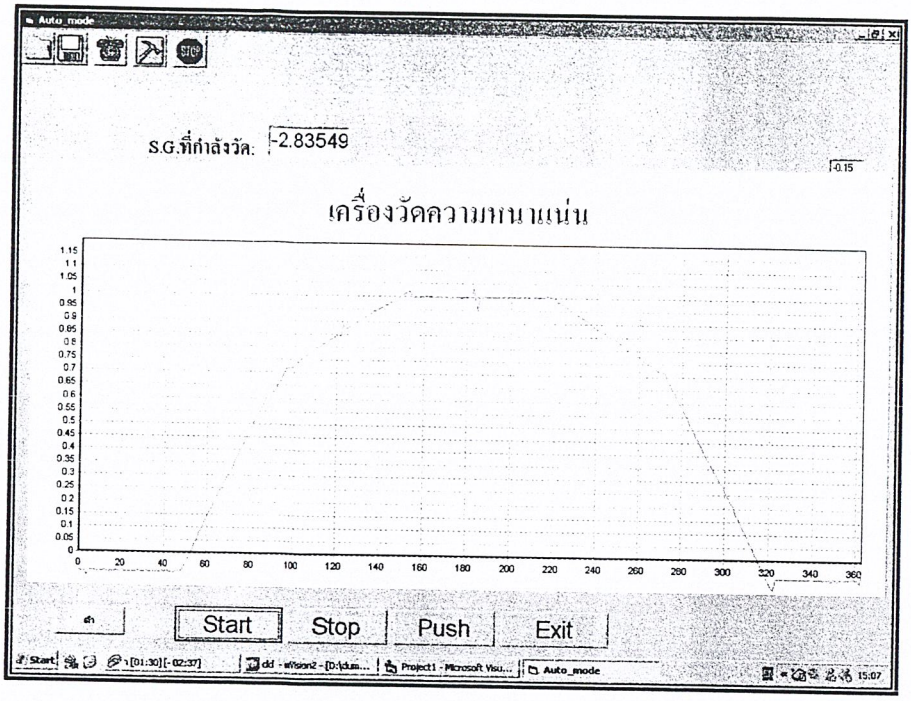
รูปที่ 4.13 ของเหลวที่ทดลองเป็นน้ำมันก๊าด



รูปที่ 4.14 ของเหลวที่ทดลองเป็นน้ำมันเบนซิน



รูปที่ 4.15 ของเหลวที่ทดลองเป็นน้ำ กับ น้ำมันปาล์ม



รูปที่ 4.16 ของเหลวที่ทดลองเป็นน้ำ กับ น้ำมันเบนซิน

ทำการทดลองของเหลวแต่ละชนิดซ้ำ โดยค่าที่ได้จากการวัดจะเป็นค่าตามตารางด้านล่างนี้

ตารางที่ 4.1 ตารางแสดงผลการวัดค่าความถ่วงจำเพาะของของเหลวแต่ละชนิด

ครั้งที่	น้ำ	น้ำมันปาล์ม	น้ำมันก๊าด	น้ำมันเบนซิน
1	1.0089	0.9194	0.7670	0.7256
2	1.0062	0.9207	0.7649	0.7247
3	1.0076	0.9189	0.7663	0.7272
4	1.0063	0.9300	0.7654	0.7265
5	1.0058	0.9217	0.7672	0.7233
6	1.0065	0.9208	0.7684	0.7257
7	1.0087	0.9180	0.7675	0.7263
8	1.0068	0.9193	0.7668	0.7241
9	1.0082	0.9291	0.7697	0.7253
10	1.0074	0.9171	0.7675	0.7284
เฉลี่ย	1.0072	0.9215	0.7671	0.7257
%errer	0.72	0.38	1.78	1.53

บทที่ 5
สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากการออกแบบและทำการทดลองเปรียบเทียบกับค่ามาตรฐาน พบว่าสามารถวัดค่าความถ่วงจำเพาะของของเหลวได้ และสามารถวัดค่าของความถ่วงจำเพาะของของเหลวที่มีการแยกชั้นกันของของเหลวได้ สามารถบอกค่าการเปลี่ยนแปลง และสามารถบอกระยะของของเหลวได้

โดยของเหลวที่ใช้ในการทดลองเป็นของเหลวที่หาได้ง่าย และเราสามารถรู้ค่าของความถ่วงจำเพาะของของเหลวนั้นๆ ได้แก่ น้ำ น้ำมันปาล์ม น้ำมันก๊าด และน้ำมันเบนซิน เป็นตัวอย่างของการทดลอง

5.2 ปัญหาที่เกิดขึ้นและแนวทางแก้ไข

1. ของเหลวที่นำมาทำการทดลอง ในบางครั้งเราไม่สามารถกำหนดค่าของความถ่วงจำเพาะที่แน่นอนได้ เพราะเราใช้ตารางค่าความถ่วงจำเพาะมาอ้างอิงเท่านั้น จึงต้องใช้น้ำกลั่นเป็นของเหลวอ้างอิง
2. การเซทค่าเริ่มต้น นั้นยังมีค่าผิดพลาดอยู่บ้างเล็กน้อย ต้องมีการทดลองหลายๆครั้ง เพื่อให้ค่าที่เซทเป็นค่าที่แน่นอน

5.3 ข้อเสนอแนะ

1. ในการพัฒนาเครื่องวัดประเภท Displacement ให้มีประสิทธิภาพที่ดี อุปกรณ์ในบางอย่างต้องมีความเที่ยงตรงแม่นยำอย่างสูง จึงต้องมีการศึกษารายละเอียดให้เข้าใจพอสมควร
2. ในการทดลองกับของเหลว ควรทดลองหลายชนิด และหลายครั้ง เพื่อเปรียบเทียบความแม่นยำ
3. ควรจะลองหาหลักการอื่นมาเปรียบเทียบความเข้าใจในทฤษฎี หรือหลักการเดิมเป็นการช่วยเพิ่มทักษะให้มากขึ้น

บรรณานุกรม

1. ผศ. วิศรุต ศรีรัตนะ , "อุปกรณ์ตรวจจับ – ส่งสัญญาณ และการแปลงสภาวะสัญญาณ" ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
2. ผศ. เกษตร์ ศิริสันติสัมฤทธิ์, "หลักการของเครื่องมือวัดทางอุตสาหกรรม" ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์ พิมพ์ครั้งที่ 3
3. สักรียา ชิตวงศ์ , "วิศวกรรมอิเล็กทรอนิกส์" ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พิมพ์ครั้งที่ 1
4. สัจจะ จรัสรุ่งรวีวร , "คู่มือการสร้างแอพพลิเคชันด้วย Visual Basic 6" พิมพ์ครั้งที่ 1 สำนักพิมพ์ อินโฟเพรส, 2542
5. ผศ. ธีรวัฒน์ ประกอบผล, "การพัฒนาไมโครคอนโทรลเลอร์ ด้วยภาษาซี" พิมพ์ครั้งที่ 4 สำนักพิมพ์ ส.ส.ท. , 2546

ภาคผนวก ก.

```
//#include<at89x52.h>
//#include <intrins.h>
#include <reg52.h>
#include <absacc.h>
#include <ctype.h>
#include <intrins.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
 #include <stdlib.h>
#include <string.h>
#include <dum.c>
unsigned char Speed_step =   0x90;   /* Speed step motor */
sbit  enable=P1^0;//1.6 pin 15
sbit  Pulse=P1^6;//1.0  pin 11
sbit  run=P1^1;//1.1    pin 14
sbit sw_limit=P1^2;
sbit sw_upper=P1^3;
sbit sw_lowwer=P1^4;
sbit mcp3202_clk  = P2^4;
sbit mcp3202_din  = P2^2;
sbit mcp3202_dout  = P2^3;
sbit mcp3202_cs   = P2^5;
#define SELECT_AD0   P2=  0x04;
#define SELECT_AD8   P2=  0x07;
unsigned int AD_0;
unsigned int End_loop;
unsigned char round;
sbit e = P3^6;                  // Bit control Enable recive data LCD
sbit rs = P3^7;                 // Bit control Command/Data LCD
code unsigned char text1[];          // Store in Program memory
void mcp3202_AD0()
```

```
        {
    unsigned char i1;
        unsigned int out;
    mcp3202_cs = 0;
        mcp3202_clk=1;
    mcp3202_din=1;
        mcp3202_clk=0;
        mcp3202_clk=1;
    mcp3202_din=1;
        mcp3202_clk=0;
        mcp3202_clk=1;
    mcp3202_din=0;
        mcp3202_clk=0;
        mcp3202_clk=1;
    mcp3202_din=0;
        mcp3202_clk=0;
        out=0;
    for (i1=0;i1<=12;i1++)
                {
          mcp3202_clk=1;
                out = out << 1;
      out = out | mcp3202_dout;
                mcp3202_clk=0;
                }
          mcp3202_clk=0;
                AD_0 = out  & 0xfff;
      mcp3202_cs=1;
        }
//----------building to pulse-----------------//

//-----display at LCD-------------------------//
void lcd_command(unsigned char com)
```

```
{
        rs = 0;                 // Write Command to LCD
        e = 1;                  // Start generate pulse clock LCD
        P0 = com;               // Send data to LCD port
        dmsec(1);               // Delay(10)
        e = 0;                  // End pulse clock
        dmsec(1);               // Delay(10)
}
void lcd_text(unsigned char text) //write command
{
        rs = 1;                 // Write Data to LCD
        e = 1;                  // Start generate pulse clock LCD
        P0 = text;              // Send data to LCD port
        dmsec(1);               // Delay
        e = 0;                  // End pulse clock
        dmsec(1);               // Delay
}
void lcd_init()
{
        dmsec(1);               // Delay for initial LCD(500)
        lcd_command(0x38);      // 2 line ,8 bit display ,5*7 dot
        lcd_command(0x0C);              // Display ON , none cursor
//      lcd_command(0x01);      // Clear screen
}
void display()
{
        unsigned char text[]="Specific Gravity";
        unsigned char m;                // Counter for send data
        lcd_init();                     // Initial LCD (From "lcd.h")
        lcd_command(0x02);
        lcd_command(0x80);      // Go to address 00H (From "lcd.h")
        for(m=0;m<17;m++)       // For loop to send data to LCD
```

```
    lcd_text (text[m]);
            lcd_command(0x02);
            lcd_command(0xc0);      // Go to address 00H (From "lcd.h")
            for(m=0;m<17;m++)       // For loop to send data to LCD
    lcd_text (text[m]);
//          mcp3202_AD0();          // Send data to LCD(From "lcd.h")
}

//------------------------//
void Coun_T(unsigned int  step){
End_loop=0;
    do {
                main_ad();
                printf(" %d\n",End_loop);
    }while (End_loop<=step);

}

//------------------------//
void Coun_T_no_print(){
                while(sw_limit!=0){
                }
}
//------------------------//

//*****************************************************************************
**//

//--------control motor--------------------//
void up()
{
```

```
if(sw_upper==0)
        {
                unsigned int m;
                enable=0;
                for(m=1;m<=2;m++)
                {
                        if(m==1)
                                {
                                        run=0;
                                        display();
                                        Speed_step=0xb0;
                                        TL0 = 0x00;
                                        TR0 = 1;
                                        Coun_T(25);//320
        Speed_step=0xb0;
                                        TL0 = 0x00;
                                        Coun_T(1165);

                                        //dmsec(1);//**********chagre1500
                                }

                if(m==2)
                                {
                run=1;
                                        display();
                                        Speed_step=0xb0;
        Coun_T(1165);
                                        Speed_step=0xf0;
        Coun_T(25);
                                        Coun_T_no_print();
                                        run=0;
                                Coun_T(30);
```

```
                                        dmsec(1);//**********chagre1000
                                    TR0 = 0;
                                        Speed_step=0xf0;
                                        TL0 = 0x00;
                                    }
                                }

                }
}

//------------------------------//
void down()
{
        if(sw_lowwer==0)
                {
                 enable=0;
      run=1;
                 Speed_step=0xf0;
                 TL0 = 0x00;
                 TR0 = 1;
                 while(sw_limit!=0)
                        {
                        .

        }
                run=0;
                Coun_T(30);
                dmsec(1);//**********chagre1000
                TR0 = 0;
                Speed_step=0xb0;
                TL0 = 0x00;
                //while(sw_limit!=0)
                //      {
```

```
                //        Coun_T(1);
        //  }
                //        run=0;
                //        Coun_T(70);
                //        dmsec(1);//**********chagre1000
                 }
}

//*****************************************************************//
void sevice_timer0(void) interrupt 1 using 2
{
        round++;                    // Increase counter 1 time
        TH0 = Speed_step;                   // Reload load value for Register TH0 10 millisec
        TL0 = 0x00;                 // Reload load value for Register TH0 10 millisec
        Pulse =~ Pulse;
  if (round==2){
          round=0;End_loop++;
        }
}

//--------------------------//
void main(void)
{
        sw_upper=1;,
        sw_lowwer=1;
        sw_limit=1;
        CKCON = CKCON | 1;
  SCON = 0x52;   /* SCON */     /* setup serial port control */
  TMOD = 0x21;   /* TMOD */     /* hardware (4800 BAUD 11.0592MHZ) Timer0 Mode1*/
  TH1 =  0xfd;   /* TH1 */
        TH0 = Speed_step;                   // Initial load value for Register TH0 10 millisec
        TL0 = 0x00;                 // Initial load value for Register TH0 10 millisec
```

```
        EA =  1;                        // Enable interrupt all
        ET0 = 1;                        // Enable interrupt Timer0
    round=0;
        PCON= PCON |0x80;
    TR1 = 1;
        TR0 = 0;
    Initialize();
                        while(1)
                {
                  enable=1;
                        up();
                        down();

                }
}
```

.....

```
#define OSET_RD    0x90   /* Offset  read command */
#define GAIN_RD    0x92   /* Gain    read command */
#define CONF_RD    0x94   /* Config  read command */
#define CONV_RD    0x96   /* Data    read command */
#define SETU_RD    0x98   /* Setup   read command */

#define OSET_WR    0x80   /* Offset  Write command */
#define GAIN_WR    0x82   /* Gain    Write command */
#define CONF_WR    0x84   /* Config  Write command */
#define CONV_WR    0x86   /* Data    Write command */
#define SETU_WR    0x88   /* Setup   Write command */

sbit SDI  = P2^0;  /* Transfer data from CPU to ADC */
sbit SCLK = P2^1;  /* Shifting data to or from ADC  */
```

```
sbit CS   = P2^6;   /* Chip Select                */
sbit SDO  = P2^7;   /* Transfer data from ADC to CPU */
unsigned char data HIGH,MID,LOW,TEMP;


bit ad;
void  dmsec(unsigned int count)
{
   unsigned char i;
   while(count)
   {
      for(i=1;i<=113;i++);
      count--;
   }
}
void WriteByte(unsigned char dat) {
   unsigned char clock;
   for(clock=0; clock<8; clock++) {
                  SDI  = (dat & 0x80);
                  dat  = (dat << 1);
     SCLK = 1;
     SCLK = 0;
        }
   SDI = 0;
}
/* Read single byte */
unsigned char ReadByte(void) {
   unsigned char clock, dat;
   dat = 0;
   for(clock=0; clock<8; clock++) {
                  dat  = (dat << 1);
                  dat  = (dat | SDO);
     SCLK = 1;
```

```
        SCLK = 0;
            }
            return(dat);
    }
    /* Write register 3 bytes */
    void WriteReg(unsigned char comm,high,mid,low) {
      WriteByte(comm);
      WriteByte(high);
      WriteByte(mid);
      WriteByte(low);
    }
    /* Read register 3 bytes */
    void ReadReg(unsigned char comm) {
      WriteByte(comm);
      HIGH = ReadByte();
      MID  = ReadByte();
      LOW  = ReadByte();
    }
    /* Read configuration register */
    void ReadConf(void) {
      ReadReg(CONF_RD);          /* Read configuration reg   */
      putchar('\n');
      printf("CONF=%02bx%02bx%02bx",HIGH,MID,LOW); /* Print     */

      ReadReg(OSET_RD);          /* Read offset reg          */
      printf(" OFFS=%02bx%02bx%02bx",HIGH,MID,LOW); /* Print    */

      ReadReg(GAIN_RD);          /* Read gain reg            */
      printf(" GAIN=%02bx%02bx%02bx",HIGH,MID,LOW); /* Print    */
    }
    void Initialize(void) {
      unsigned int count;
```

```
    CS   = 0;
    SCLK = 0;
    SDO  = 0;
    SDI  = 0;
    for(count=0; count<2047; count++) {
       SCLK = 1;                    /* Assert SCLK */
       SCLK = 0;                    /* Deassert    */
    }
    SDI = 1;                        /* Assert SDI  */
    for(count=0; count<255; count++) {
       SCLK = 1;                    /* Assert SCLK */
       SCLK = 0;                    /* Deassert    */
    }
    SDI  = 0;
    SCLK = 1;
    SCLK = 0;

    WriteReg(CONF_WR,0x00,0x00,0x80);  /* Reset                    */
    for(count=0; count<2047; count++) {
    }
    WriteReg(CONF_WR,0x00,0x04,0x00);  /* Initialize configuration */
    for(count=0; count<2047; count++) {
    }
    WriteReg(GAIN_WR,0xff,0xff,0xff);
    // WriteReg(GAIN_WR,0x80,0x00,0x00);
          //WriteReg(OSET_WR,0x00,0x50,0x00);
    for(count=0; count<2047; count++) {
    }
    ReadConf();
}

/* Read data conversion */
```

```
unsigned long ReadConv(void) {
   unsigned long l;
   WriteReg(CONF_WR,0x00,0xc6,0x00);
   WriteByte(0xc0);
   do {
      ReadReg(CONF_RD);     // Read Configuration Register
   }while(!(LOW & 0x08));
  ReadReg(CONV_RD);
   l  = HIGH;
   l <<= 8;
   l  |= MID;
   l <<= 8;
   l  |= LOW;
   l >>= 4;
 // if(LOW & 0x01) {   /* Test OF Flag */
   /*  beeph();
    beeph(); */
// }
// if(LOW & 0x02) {   /* Test OD Flag */
   /* beepl();
    beepl(); */
// }
  return(l);
}
float VoltageAdd(unsigned long sn) {
  float y,c,mm;
  if(sn <= 524287) {        /*  y = mx + c ; mm = y2 - y1 / x2 - x1 */
    mm = 7.364845276e-6;    /* mm = ( 3.8660 -4.7mV) / 524288 */
    c = 0.007;              /* mm = 7.364845276e-6 */
    y = (mm*sn) + c;
    return(y);
  }
```

```
    else {
      sn = 1048576 - sn;
      mm = -7.364845276e-6;
      c = 0.002;
      y = (mm*sn) + c;
      return(y);
    }
}

unsigned long loopx(void) {
   unsigned long l;
      l = ReadConv();              /* Read Conversion */
   return(l);
}

void main_ad(void) {
   unsigned long l;
   float v;
      l = loopx();
      v = VoltageAdd(l);
            printf("%2.4fV",v);

}
```

ภาคผนวก ข.

ตาราง ค่าความหนาแน่นของของเหลว

Liquid	Temp	kg/cu.m
1,2-Trichlorotrifluoroethane	25 C	1564
1,2,4-Trichlorobenzene	20 C	1454
1,4-Dioxane	20 C	1033.6
2-Methoxyethanol	20 C	964.6
Acetic Acid	25 C	1049.1
Acetone	25 C	784.58
Acetonitrile	20 C	782.2
Alcohol, ethyl	25 C	785.06
Alcohol, methyl	25 C	786.51
Alcohol, propyl	25 C	799.96
Ammonia (aqua)	25 C	823.35
Analine	25 C	1018.93
Automobile oils	15 C	880 - 940
Benzil	25 C	1079.64
Brine	15 C	1230
Bromine	25 C	3120.4
Butyric Acid	20 C	959
Butane	25 C	599.09
n-Butyl Acetate	20 C	879.6
n-Butyl Alcohol	20 C	809.7
n-Butyl Chloride	20 C	886.2
Caproic acid	25 C	921.06
Carbolic acid	15 C	956.3
Carbon disulfide	25 C	1260.97
Carbon tetrachloride	25 C	1584.39
Carene	25 C	856.99
Castor oil	25 C	956.14
Chloridezene	20 C	1105.8
Chloroform	25 C	1464.73
Citric acid	25 C	1659.51
Coconut oil	15 C	924.27
Cotton seed oil	15 C	925.87
Cresol	25 C	1023.58
Creosote	15 C	1066.83
Crude oil, 48° API	60 F	790
Crude oil, 40° API	60 F	825
Crude oil, 35.6° API	60 F	847
Crude oil, 32.6° API	60 F	862
Crude oil, California	60 F	915
Crude oil, Mexican	60 F	973
Crude oil, Texas	60 F	873
Cumene	25 C	860.19
Cyclohexane	20 C	778.5
Cyclopentane	20 C	745.4
Decane	25 C	726.28
Diesel fuel oil 20 to 60	15 C	820 - 950
Diethyl ether	20 C	714
o-Dichlorobenzene	20 C	1305.8
Dichloromethane	20 C	1326
Diethylene glycol	15 C	1120
Dichloromethane	20 C	1326
Dimethyl Acetamide	20 C	941.5

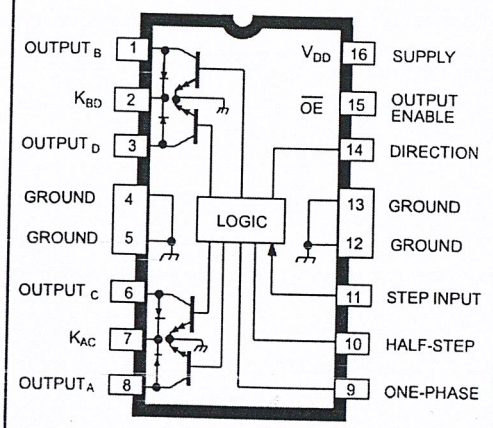
Liquid	Temp	kg/cu.m
Dodecane	25 C	754.63
Ethane	(-)89 C	570.26
Ether	25 C	72.72
Ethylamine	16 C	680.78
Ethyl Acetate	20 C	900.6
Ethyl Ather	20 C	789.2
Ethyl Ether	20 C	713.3
Ethylene Dichloride	20 C	1253
Ethylene glycol	25 C	1096.78
Fluorine refrigerant R-12	25 C	1310.95
Formaldehyde	45 C	812.14
Formic acid 10% concentration	20 C	1025
Formic acid 80% concentration	20 C	1221
Freon - 11	21 C	1490
Freon - 21	21 C	1370
Fuel oil	60 F	890.13
Furan	25 C	1416.03
Furforol	25 C	1154.93
Gasoline, natural	60 F	711.22
Gasoline, Vehicle	60 F	737.22
Gas oils	60 F	890
Glucose	60 F	1350-1440
Glycerin	25 C	1259.37
Glyme	20 C	869.1
Glycerol	25 C	1126.1
Heptane	25 C	679.5
Hexane	25 C	654.83
Hexanol	25 C	810.53
Hexene	25 C	671.17
Hydrazine	25 C	794.52
Iodine	25 C	4927.28
Ionene	25 C	932.27
Isobutyl Alcohol	20 C	801.6
Iso-Octane	20 C	691.9
Isopropyl Alcohol	20 C	785.4
Isopropyl Myristate	20 C	853.2
Kerosene	60 F	817.15
Linolenic Acid	25 C	898.64
Linseed oil	25 C	929.07
Methane	(-)164C	464.54
Methanol	20 C	791.3
Methyl Isoamyl Ketone	20 C	888
Methyl Isobutyl Ketone	20 C	800.8
Methyl n-Propyl Ketone	20 C	808.2
Methyl t-Butyl Ether	20 C	740.5
N-Methylpyrrolidone	20 C	1030.4
Methyl Ethyl Ketone	20 C	804.9
MEK	25 C	802.52
Milk	15 C	1020-1050
Naphtha	15 C	664.77

Liquid	Temp	kg/cu.m
Octane	25 C	797.72
Octane	15 C	917.86
Olive oil	20 C	800-920
Oxygen (liquid)	(-)183C	1140
Palmitic Acid	25 C	850.58
Pentane	20 C	626.2
Pentane	25 C	624.82
Petroleum Ether	20 C	640
Petrol, natural	60 F	711.22
Petrol, Vehicle	60 F	737.22
Phenol	25 C	1072.28
Phosgene	-862	1377.59
Phytadiene	25 C	823.35
Pinene	25 C	856.99
Propane	(-)40 C	583.07
Propane, R-290	25 C	493.53
Propanol	25 C	804.13
Propylene Carbonate	20 C	1200.6
Propylene	25 C	514.35
n-Propyl Alcohol	20 C	803.7
Propylene glycol	25 C	965.27
Pyridine	25 C	978.73
Pyrrole	25 C	965.91
Rape seed oil	20 C	920
Resorcinol	25 C	1268.66
Rosin oil	15 C	980
Sabiname	25 C	812.14
Sea water	25 C	1025.18
Silane	25 C	717.63
Sodium Hydroxide (caustic soda)	15 C	1250
Sorbaldehyde	25 C	895.43
Soya bean oil	15 C	924-928
Stearic Acid	25 C	890.63
Sulphuric Acid 95% conc.	20 C	1839
Sugar solution 68 brix	15 C	1338
Sunflower oil	20 C	920
Styrene	25 C	903.44
Terpinene	25 C	847.38
Tetrahydrofuran	20 C	888
Toluene	20 C	866.9
Toluene	25 C	862.27
Triethylamine	20 C	727.6
Trifluoroacetic Acid	20 C	1489
Turpentine	25 C	868.2
Water, pure (more temperatures)	4 C	1000
Water, sea	77 F	1021.98
Whale oil	15 C	925
o-Xylene	20 C	880.2

5804

Data Sheet
26184.12C

BiMOS II UNIPOLAR STEPPER-MOTOR TRANSLATOR/DRIVER



Dwg. W-194

Note that the UCN5804B (dual in-line package) and UCN5804LB (small outline IC package) are electrically identical and share a common pin number assignment.

ABSOLUTE MAXIMUM RATINGS

Output Voltage, V_{CE} ..... **50 V**
Output Sustaining Voltage, $V_{CE\ (sus)}$ ..... **35 V**
Output Sink Current, I_{OUT} ..... **1.5 A**
Logic Supply Voltage, V_{DD} ..... **7.0 V**
Input Voltage, V_{IN} ..... **7.0 V**
Package Power Dissipation, P_D ..... **See Graph**
Operating Temperature Range, T_A ..... **-20°C to +85°C**
Storage Temperature Range, T_S ..... **-55°C to +150°C**

Combining low-power CMOS logic with high-current and high-voltage bipolar outputs, the UCN5804B and UCN5804LB BiMOS II translator/drivers provide complete control and drive for a four-phase unipolar stepper-motor with continuous output current ratings to 1.25 A per phase (1.5 A startup) and 35 V.

The CMOS logic section provides the sequencing logic, DIRECTION and OUTPUT ENABLE control, and a power-ON reset function. Three stepper-motor drive formats, wave-drive (one-phase), two-phase, and half-step are externally selectable. The inputs are compatible with standard CMOS, PMOS, and NMOS circuits. TTL or LSTTL may require the use of appropriate pull-up resistors to ensure a proper input-logic high.

The wave-drive format consists of energizing one motor phase at a time in an A-B-C-D (or D-C-B-A) sequence. This excitation mode consumes the least power and assures positional accuracy regardless of any winding inbalance in the motor. Two-phase drive energizes two adjacent phases in each detent position (AB-BC-CD-DA). This sequence mode offers an improved torque-speed product, greater detent torque, and is less susceptible to motor resonance. Half-step excitation alternates between the one-phase and two-phase modes (A-AB-B-BC-C-CD-D-DA), providing an eight-step sequence.

The bipolar outputs are capable of sinking up to 1.5 A and withstanding 50 V in the OFF state (sustaining voltages up to 35 V). Ground-clamp and flyback diodes provide protection against inductive transients. Thermal protection circuitry disables the outputs when the chip temperature is excessive.

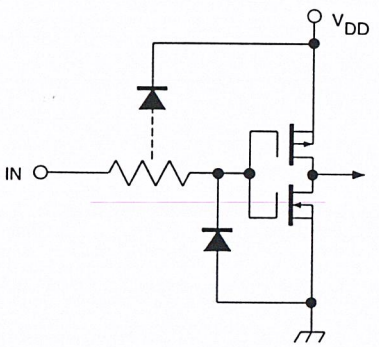
Both devices are rated for operation over the temperature range of -20°C to +85°C. The UCN5804B is supplied in a 16-pin dual in-line plastic batwing package with a copper lead frame and heat-sinkable tabs for improved power dissipation capabilities; the UCN5804LB is supplied in a 16-lead plastic SOIC batwing package with a copper lead frame and heat-sinkable tabs.

FEATURES

- 1.5 A Maximum Output Current
- 35 V Output Sustaining Voltage
- Wave-Drive, Two-Phase, and Half-Step Drive Formats
- Internal Clamp Diodes
- Output Enable and Direction Control
- Power-ON Reset
- Internal Thermal Shutdown Circuitry

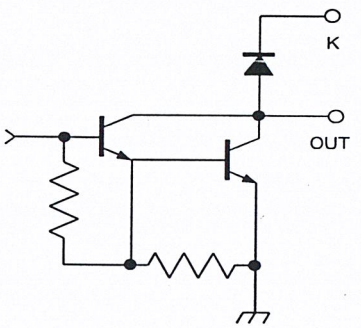
Always order by complete part number, e.g., **UCN5804B**.

TYPICAL INPUT CIRCUIT

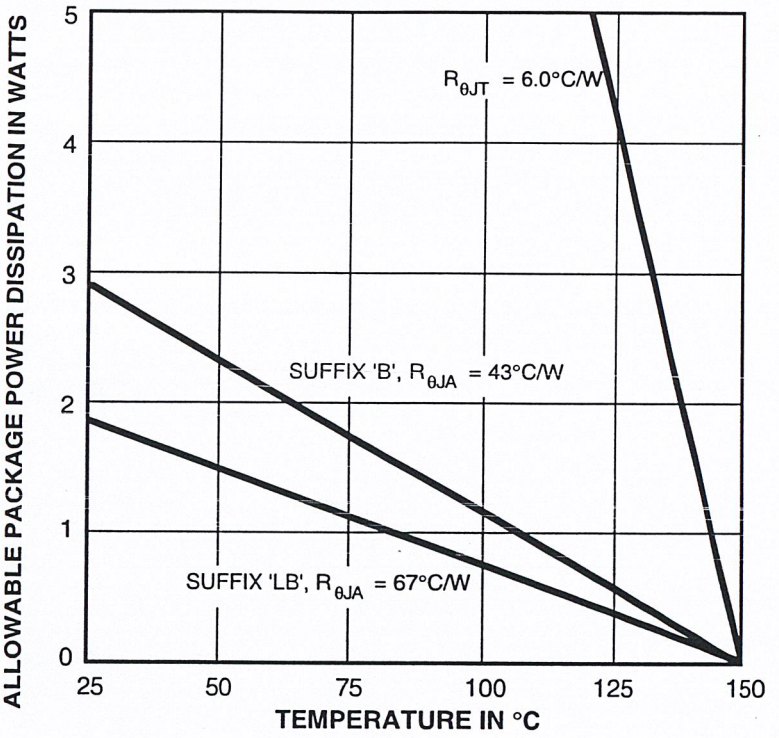


Dwg. EP-010-5

TYPICAL OUTPUT DRIVER



Dwg. EP-021-4



Dwg. GP-049-2

TRUTH TABLE

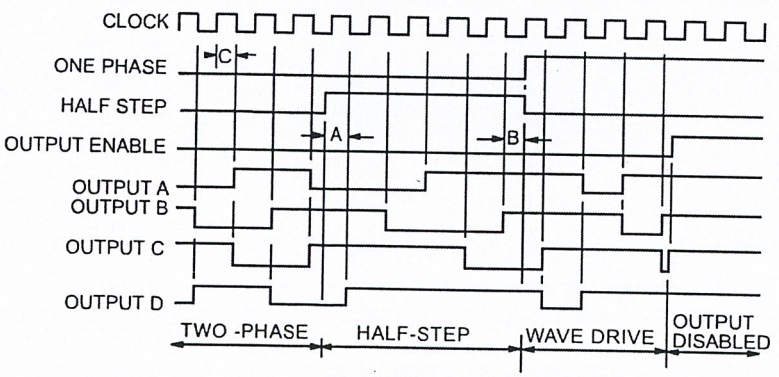
Drive Format	Pin 9	Pin 10
Two-Phase	L	L
One-Phase	H	L
Half-Step	L	H
Step-Inhibit	H	H

Allegro MicroSystems, Inc.
115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000
Copyright © 1987, 1998, Allegro MicroSystems, Inc.

ELECTRICAL CHARACTERISTICS at $T_A = 25°C$, $T_J \leq 150°C$, $V_{DD} = 4.5$ V to 5.5 V (unless otherwise noted).

Characteristic	Symbol	Test Conditions	Limits			
			Min.	Typ.	Max.	Units
Output Leakage Current	I_{CEX}	V_{OUT} = 50 V	—	10	50	µA
Output Sustaining Voltage	$V_{CE(sus)}$	I_{OUT} = 1.25 A, L = 3 mH	35	—	—	V
Output Saturation Voltage	$V_{CE(SAT)}$	I_{OUT} = 700 mA	—	1.0	1.2	V
		I_{OUT} = 1 A	—	1.1	1.4	V
		I_{OUT} = 1.25 A	—	1.2	1.5	V
Clamp Diode Leakage Current	I_R	V_R = 50 V	—	10	50	µA
Clamp Diode Forward Voltage	V_F	I_F = 1.25 A	—	1.5	3.0	V
Input Current	$I_{IN(1)}$	$V_{IN} = V_{DD}$	—	0.5	5.0	µA
	$I_{IN(0)}$	V_{IN} = 0.8 V	—	-0.5	-5.0	µA
Input Voltage	$V_{IN(1)}$	V_{DD} = 5 V	3.5	—	5.3	V
	$V_{IN(0)}$		-0.3	—	0.8	V
Supply Current	I_{DD}	2 Outputs ON	—	20	30	mA
Turn-Off Delay	t_{ON}	50% Step Inputs to 50% Output	—	—	10	µs
Turn-On Delay	t_{OFF}	50% Step Inputs to 50% Output	—	—	10	µs
Thermal Shutdown Temperature	T_J		—	165	—	°C

TIMING CONDITIONS



Dwg. W-110A

A. Minimum Data Set Up Time . . . . . **100 ns**
B. Minimum Data Hold Time . . . . . **100 ns**
C. Minimum Step Input Pulse Width . . . . . **3.0 µs**

APPLICATIONS INFORMATION

Internal power-ON reset (POR) circuitry resets $OUTPUT_A$ (and $OUTPUT_D$ in the two-phase drive format) to the ON state with initial application of the logic supply voltage. After reset, the circuit then steps according to the tables.

The outputs will advance one sequence position on the high-to-low transition of the STEP INPUT pulse. Logic levels on the HALF-STEP and ONE-PHASE inputs will determine the drive format (one-phase, two-phase, or half-step). The DIRECTION pin determines the rotation sequence of the outputs. Note that the STEP INPUT must be in the low state when changing the state of ONE-PHASE, HALF-STEP, or DIRECTION to prevent erroneous stepping.

All outputs are disabled (OFF) when OUTPUT ENABLE is at a logic high. If the function is not required, OUTPUT ENABLE should be tied low. In that condition, all outputs depend only on the state of the step logic.

During normal commutation of a unipolar stepper motor, mutual coupling between the motor windings can force the outputs of the UCN5804B below ground. This condition will cause forward biasing of the collector-to-substrate junction and source current from the output. For many L/R applications, this substrate current is high enough to adversely affect the logic circuitry and cause misstepping. External series diodes (Schottky are recommended for increased efficiency at low-voltage operation) will prevent substrate current from being sourced through the outputs. Alternatively, external ground clamp diodes will provide a preferred current path from ground when the outputs are pulled below ground.

Internal thermal protection circuitry disables all outputs when the junction temperature reaches approximately 165°C. The outputs are enabled again when the junction cools down to approximately 145°C.

WAVE-DRIVE SEQUENCE

Half Step = L, One Phase = H				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	OFF
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	OFF	OFF	ON	OFF
4	OFF	OFF	OFF	ON

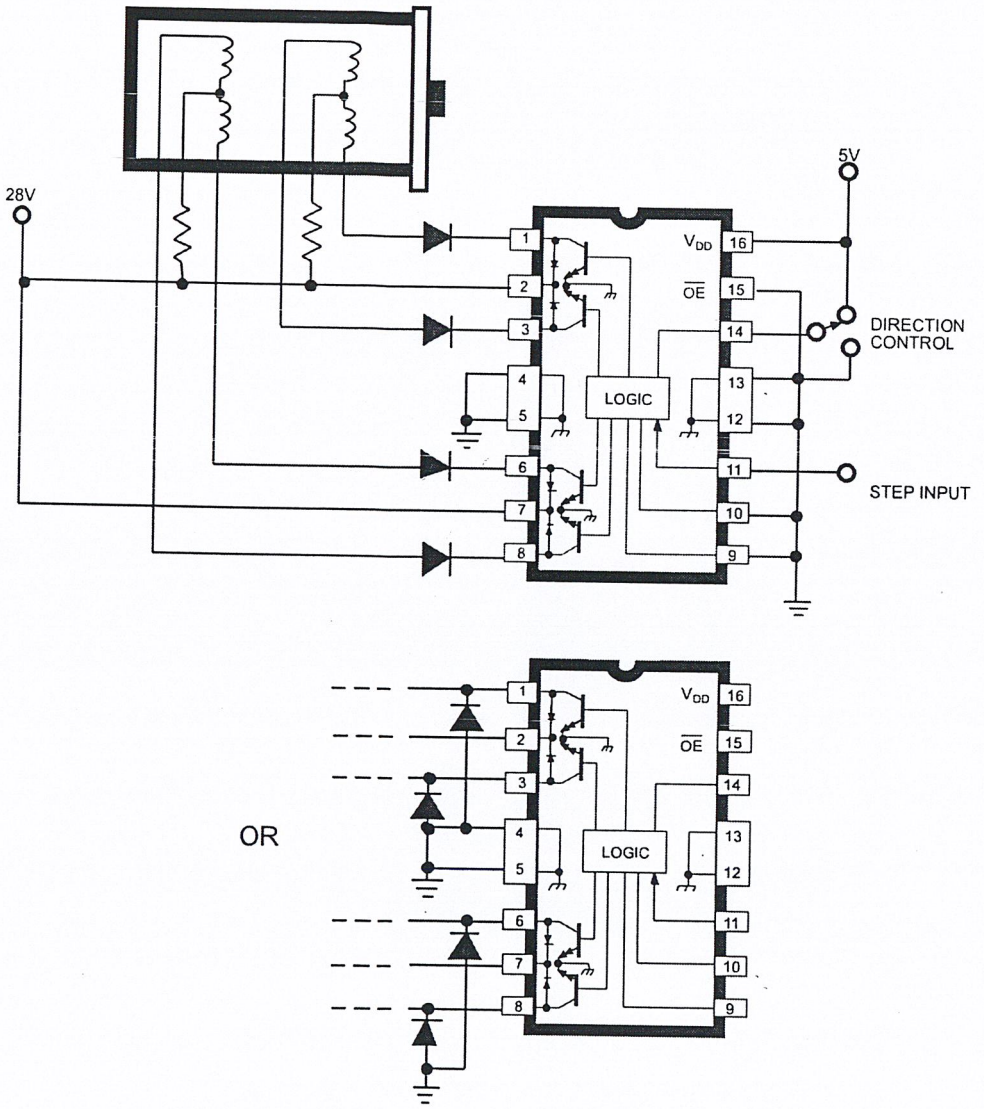
TWO-PHASE DRIVE SEQUENCE

Half Step = L, One Phase = L				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	ON
1	ON	OFF	OFF	ON
2	ON	ON	OFF	OFF
3	OFF	ON	ON	OFF
4	OFF	OFF	ON	ON

HALF-STEP DRIVE SEQUENCE

Half Step = H, One Phase = L				
Step	**A**	**B**	**C**	**D**
POR	ON	OFF	OFF	OFF
1	ON	OFF	OFF	OFF
2	ON	ON	OFF	OFF
3	OFF	ON	OFF	OFF
4	OFF	ON	ON	OFF
5	OFF	OFF	ON	OFF
6	OFF	OFF	ON	ON
7	OFF	OFF	OFF	ON
8	ON	OFF	OFF	ON

TYPICAL APPLICATION
L/R Stepper-Motor Drive

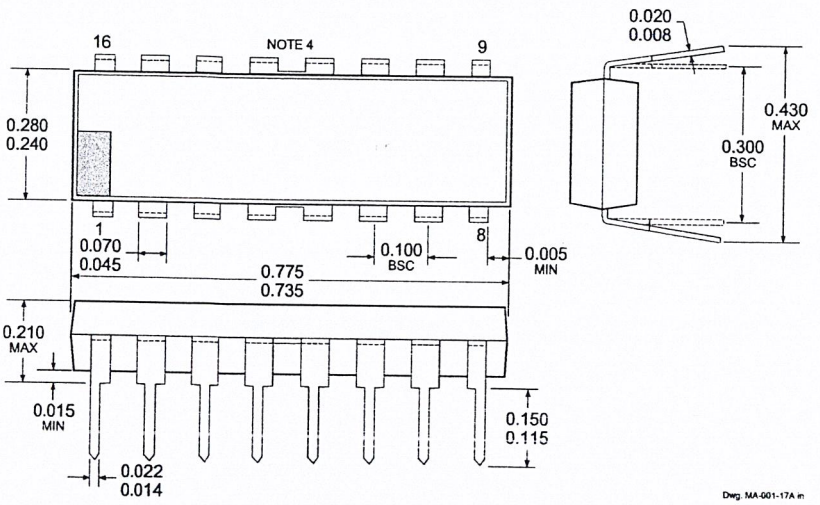


OR

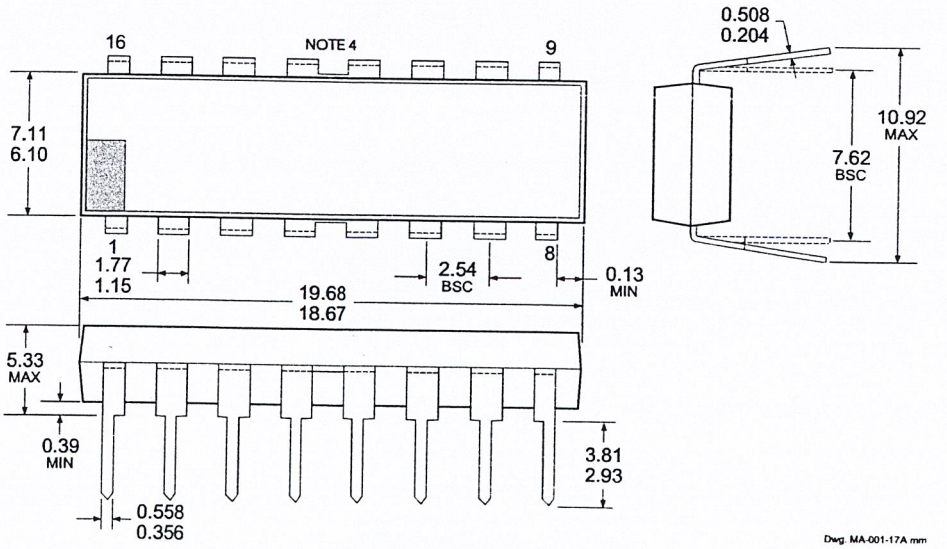
Dwg. EP-029A

UCN5804B

Dimensions in Inches
(controlling dimensions)



Dimensions in Millimeters
(for reference only)



NOTES:
1. Exact body and lead configuration at vendor's option within limits shown.
2. Lead spacing tolerance is non-cumulative
3. Lead thickness is measured at seating plane or below.
4. Webbed lead frame. Leads 4, 5, 12, and 13 are internally one piece.

Allegro® MicroSystems, Inc.
115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000

5804
BiMOS II UNIPOLAR STEPPER-MOTOR TRANSLATOR/DRIVER

MOTOR DRIVERS SELECTION GUIDE

Function	Output Ratings *		Part Number †
INTEGRATED CIRCUITS FOR BRUSHLESS DC MOTORS			
3-Phase Controller/Drivers	±2.0 A	45 V	2936 and 2936-120
Hall-Effect Latched Sensors	10 mA	24 V	3175 and 3177
2-Phase Hall-Effect Sensor/Controller	20 mA	25 V	3235
Hall-Effect Complementary-Output Sensor	20 mA	25 V	3275
2-Phase Hall-Effect Sensor/Driver	900 mA	14 V	3625
2-Phase Hall-Effect Sensor/Driver	400 mA	26 V	3626
Hall-Effect Complementary-Output Sensor/Driver	300 mA	60 V	5275
3-Phase Back-EMF Controller/Driver	±900 mA	14 V	8902–A
3-Phase Controller/DMOS Driver	±4.0 A	14 V	8925
3-Phase Back-EMF Controller/Driver	±1.0 A	7 V	8984
INTEGRATED BRIDGE DRIVERS FOR DC AND BIPOLAR STEPPER MOTORS			
PWM Current-Controlled Dual Full Bridge	±750 mA	45 V	2916
PWM Current-Controlled Dual Full Bridges	±1.5 A	45 V	2917 and 2918
PWM Current-Controlled Dual Full Bridge	±750 mA	45 V	2919
Dual Full-Bridge Driver	±2.0 A	50 V	2998
PWM Current-Controlled Full Bridge	±2.0 A	50 V	3952
PWM Current-Controlled Full Bridge	±1.3 A	50 V	3953
PWM Current-Controlled Microstepping Full Bridges	±1.5 A	50 V	3955 and 3957
PWM Current-Controlled Dual Full Bridge	±800 mA	33 V	3964
PWM Current-Controlled Dual Full Bridge	±650 mA	30 V	3966 and 3968
PWM Current-Controlled Dual Full Bridge	±750 mA	45 V	6219
OTHER INTEGRATED CIRCUIT & PMCM MOTOR DRIVERS			
Unipolar Stepper-Motor Quad Driver	1.8 A	50 V	2544
Unipolar Stepper-Motor Translator/Driver	1.25 A	50 V	5804
Unipolar Stepper-Motor Quad Drivers	1 A	46 V	7024 and 7029
Unipolar Microstepper-Motor Quad Driver	1.2 A	46 V	7042
Voice-Coil Motor Driver	±500 mA	6 V	8932–A
Voice-Coil Motor Driver	±800 mA	16 V	8958
Voice-Coil (and Spindle) Motor Driver	±350 mA	7 V	8984

* Current is maximum specified test condition, voltage is maximum rating. See specification for sustaining voltage limits or over-current protection voltage limits. Negative current is defined as coming out of (sourcing) the output.

† Complete part number includes additional characters to indicate operating temperature range and package style.

Allegro MicroSystems, Inc. reserves the right to make, from time to time, such departures from the detail specifications as may be required to permit improvements in the design of its products. The information included herein is believed to be accurate and reliable. However, Allegro MicroSystems, Inc. assumes no responsibility for its use; nor for any infringements of patents or other rights of third parties which may result from its use.

Allegro® MicroSystems, Inc.

115 Northeast Cutoff, Box 15036
Worcester, Massachusetts 01615-0036 (508) 853-5000

Features

- Compatible with MCS-51® Products
- 8K Bytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 1000 Write/Erase Cycles
- 4.0V to 5.5V Operating Range
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Full Duplex UART Serial Channel
- Low-power Idle and Power-down Modes
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag

Description

The AT89S52 is a low-power, high-performance CMOS 8-bit microcontroller with 8K bytes of in-system programmable Flash memory. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with in-system programmable Flash on a monolithic chip, the Atmel AT89S52 is a powerful microcontroller which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, Watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S52 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 interrupt or hardware reset.

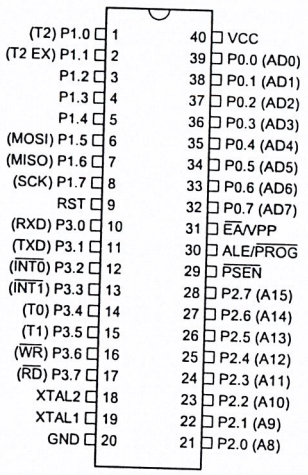
8-bit Microcontroller with 8K Bytes In-System Programmable Flash

AT89S52

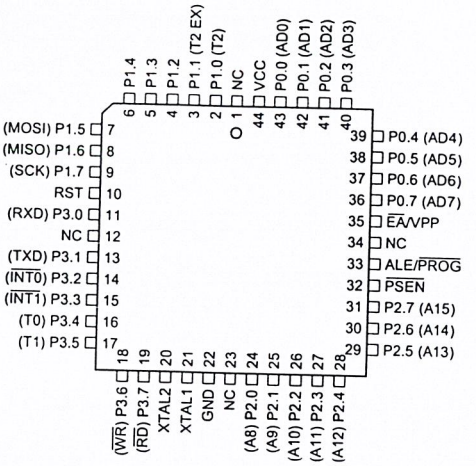
Rev. 1919A-07/01

Pin Configurations

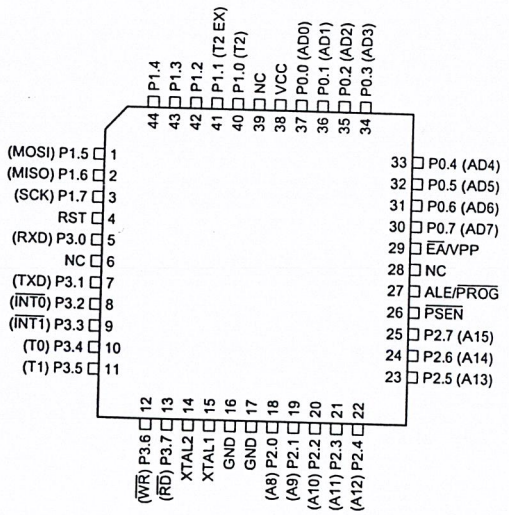
PDIP



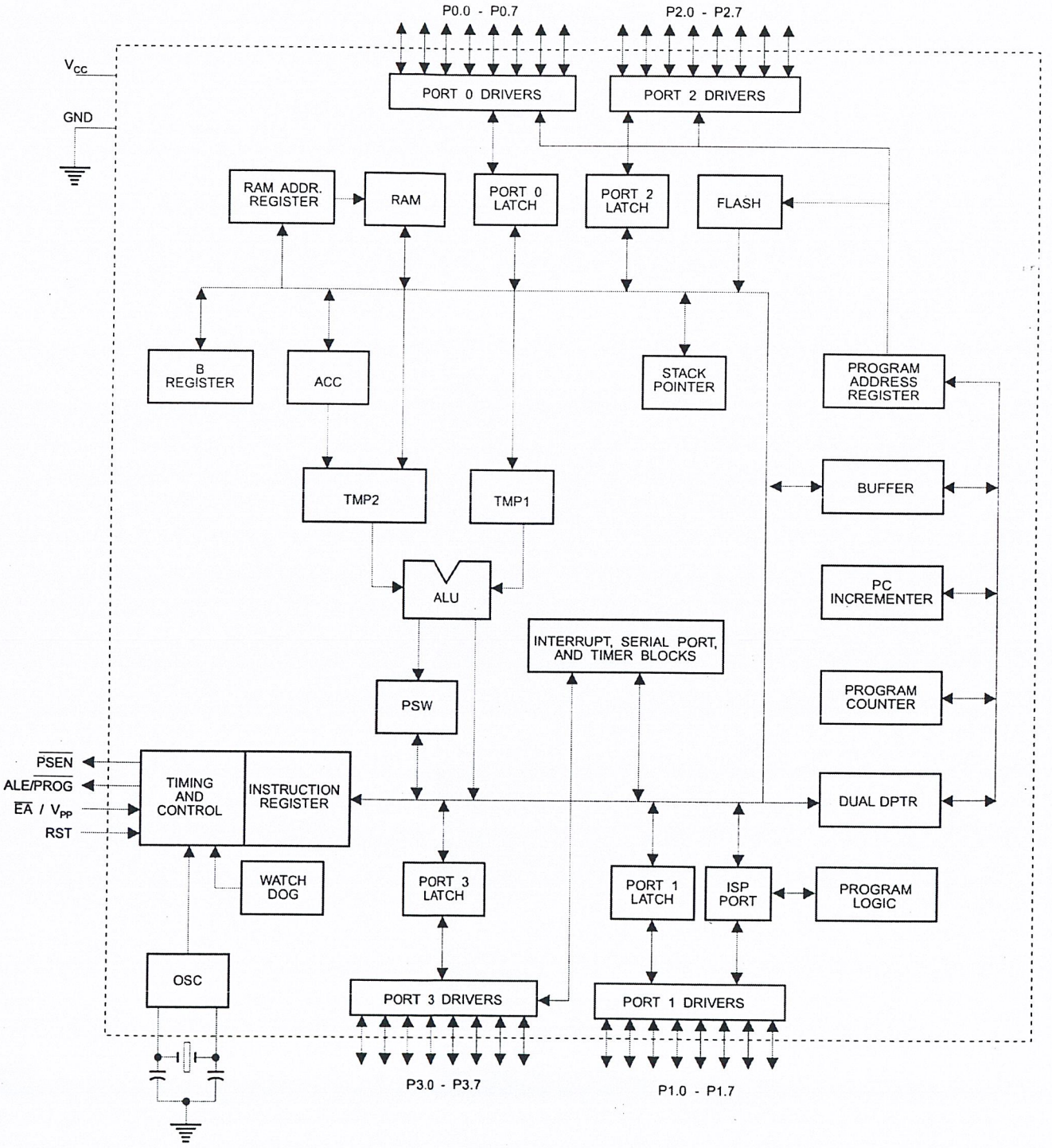
PLCC



TQFP



Block Diagram



Pin Description

VCC
Supply voltage.

GND
Ground.

Port 0
Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1
Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table.

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

Port 2
Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3
Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. 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 AT89S52, as shown in the following table.

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

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)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST
Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives High for 96 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

ALE/$\overline{PROG}$
Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is

weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{\text{PSEN}}$

Program Store Enable ($\overline{\text{PSEN}}$) is the read strobe to external program memory.

When the AT89S52 is executing code from external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory.

$\overline{\text{EA}}$/VPP

External Access Enable. $\overline{\text{EA}}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{\text{EA}}$ will be internally latched on reset.

$\overline{\text{EA}}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming.

XTAL1

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

XTAL2

Output from the inverting oscillator amplifier.

Table 1. AT89S52 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX		0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0		8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000	87H

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 1.

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.

Timer 2 Registers: Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 3) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

Interrupt Registers: The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the six interrupt sources in the IP register.

Table 2. T2CON - Timer/Counter 2 Control Register

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

Bit	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
	7	6	5	4	3	2	1	0

Symbol	Function
TF2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
CP/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Table 3a. AUXR: Auxiliary Register

AUXR Address = 8EH Reset Value = XXX00XX0B

Not Bit Addressable

Bit	7	6	5	4	3	2	1	0
	–	–	–	WDIDLE	DISRTO	–	–	DISALE

– Reserved for future expansion

DISALE Disable/Enable ALE

DISALE	Operating Mode
0	ALE is emitted at a constant rate of 1/6 the oscillator frequency
1	ALE is active only during a MOVX or MOVC instruction

DISRTO Disable/Enable Reset out

DISRTO	
0	Reset pin is driven High after WDT times out
1	Reset pin is input only

WDIDLE Disable/Enable WDT in IDLE mode

WDIDLE	
0	WDT continues to count in IDLE mode
1	WDT halts counting in IDLE mode

Dual Data Pointer Registers: To facilitate accessing both internal and external data memory, two banks of 16-bit Data Pointer Registers are provided: DP0 at SFR address locations 82H-83H and DP1 at 84H-85H. Bit DPS = 0 in SFR AUXR1 selects DP0 and DPS = 1 selects DP1. The user should always initialize the DPS bit to the appropriate value before accessing the respective Data Pointer Register.

Power Off Flag: The Power Off Flag (POF) is located at bit 4 (PCON.4) in the PCON SFR. POF is set to "1" during power up. It can be set and rest under software control and is not affected by reset.

Table 3b. AUXR1: Auxiliary Register 1

AUXR1 Address = A2H Reset Value = XXXXXXX0B

Not Bit Addressable

Bit	7	6	5	4	3	2	1	0
	–	–	–	–	–	–	–	DPS

– Reserved for future expansion

DPS Data Pointer Register Select

DPS	
0	Selects DPTR Registers DP0L, DP0H
1	Selects DPTR Registers DP1L, DP1H

Memory Organization

MCS-51 devices have a separate address space for Program and Data Memory. Up to 64K bytes each of external Program and Data Memory can be addressed.

Program Memory

If the $\overline{EA}$ pin is connected to GND, all program fetches are directed to external memory.

On the AT89S52, if $\overline{EA}$ is connected to V_{CC}, program fetches to addresses 0000H through 1FFFH are directed to internal memory and fetches to addresses 2000H through FFFFH are to external memory.

Data Memory

The AT89S52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. This means that the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions which use direct addressing access of the SFR space.

For example, the following direct addressing instruction accesses the SFR at location 0A0H (which is P2).

```
MOV 0A0H, #data
```

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

```
MOV @R0, #data
```

Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

Watchdog Timer (One-time Enabled with Reset-out)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upsets. The WDT consists of a 13-bit counter and the Watchdog Timer Reset (WDTRST) SFR. The WDT is defaulted to disable from exiting reset. To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, it will increment every machine cycle while the oscillator is running. The WDT timeout period is dependent on the external clock frequency. There is no way to disable the WDT except through reset (either hardware reset or WDT overflow reset). When WDT overflows, it will drive an output RESET HIGH pulse at the RST pin.

Using the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT overflow. The 13-bit counter overflows when it reaches 8191 (1FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 8191 machine cycles. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write-only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is 96xTOSC, where TOSC=1/FOSC. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S52 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down mode.

To ensure that the WDT does not overflow within a few states of exiting Power-down, it is best to reset the WDT just before entering Power-down mode.

Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled. The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S52 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and reenter IDLE mode.

With WDIDLE bit enabled, the WDT will stop to count in IDLE mode and resumes the count upon exit from IDLE.

UART

The UART in the AT89S52 operates the same way as the UART in the AT89C51 and AT89C52. For further information on the UART operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then 'Product Overview'.

Timer 0 and 1

Timer 0 and Timer 1 in the AT89S52 operate the same way as Timer 0 and Timer 1 in the AT89C51 and AT89C52. For further information on the timers' operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then 'Product Overview'.

Timer 2

Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 3. Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

Table 3. Timer 2 Operating Modes

RCLK +TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

Capture Mode

In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1-to-0 transition at external input T2EX also causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 5.

Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 4). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

Figure 5. Timer in Capture Mode

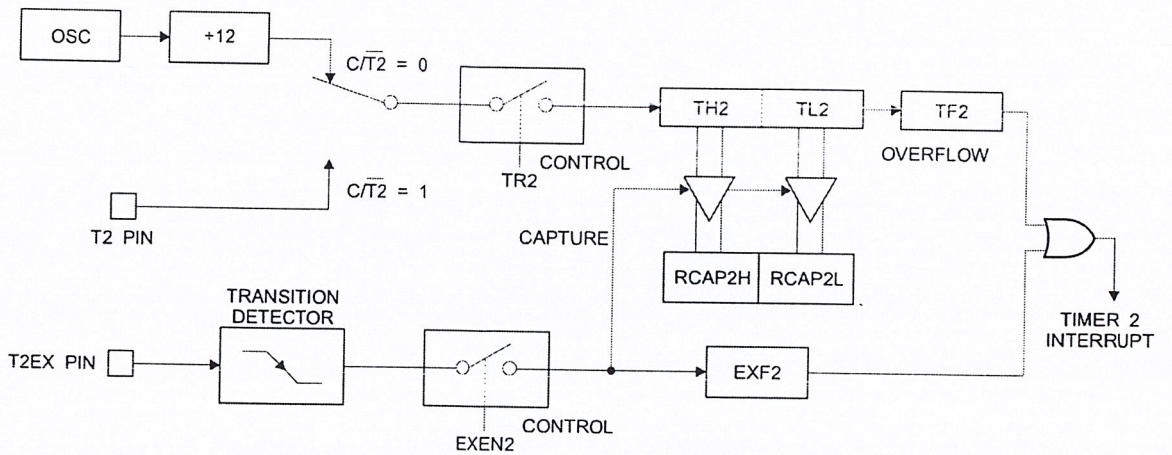


Figure 6 shows Timer 2 automatically counting up when DCEN=0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture ModeRCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 6. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

A logic 0 at T2EX makes Timer 2 count down. The timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows and can be used as a 17th bit of resolution. In this operating mode, EXF2 does not flag an interrupt.

Figure 6. Timer 2 Auto Reload Mode (DCEN = 0)

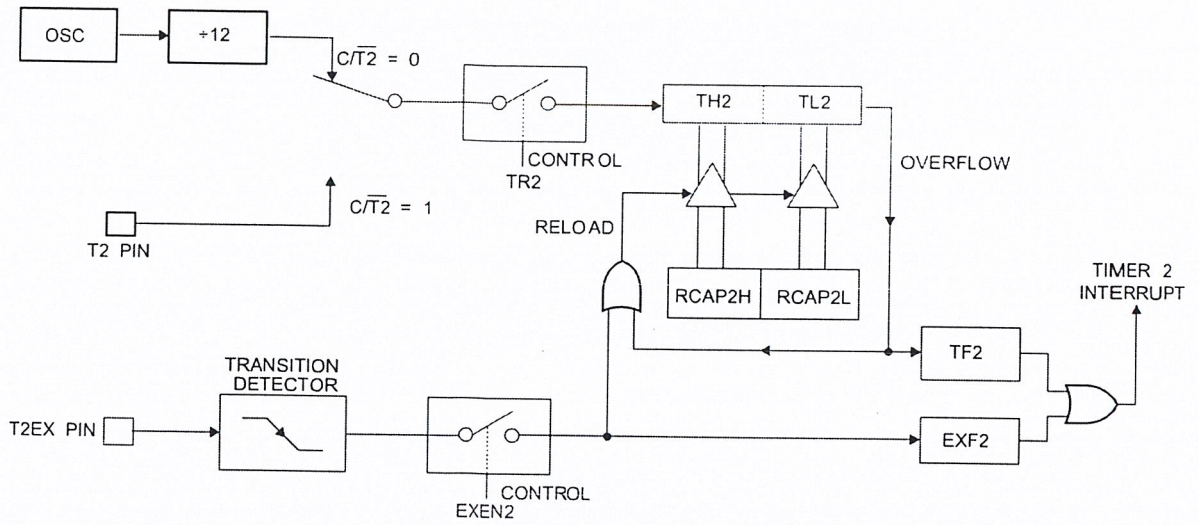


Table 4. T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

Bit	–	–	–	–	–	–	T2OE	DCEN
	7	6	5	4	3	2	1	0

Symbol	Function
–	Not implemented, reserved for future
T2OE	Timer 2 Output Enable bit
DCEN	When set, this bit allows Timer 2 to be configured as an up/down counter

Figure 7. Timer 2 Auto Reload Mode (DCEN = 1)

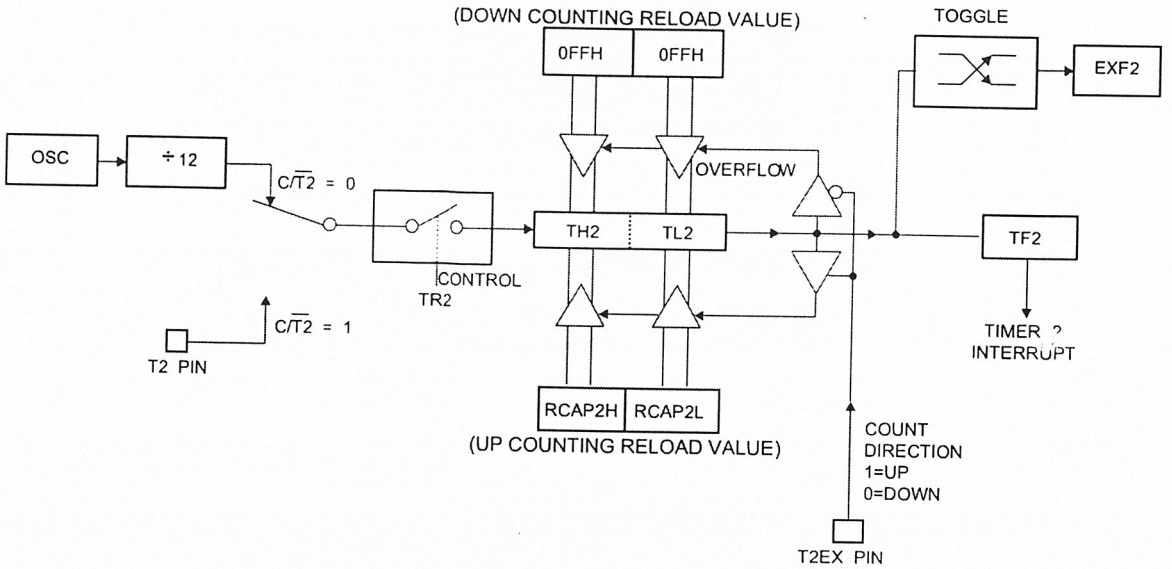
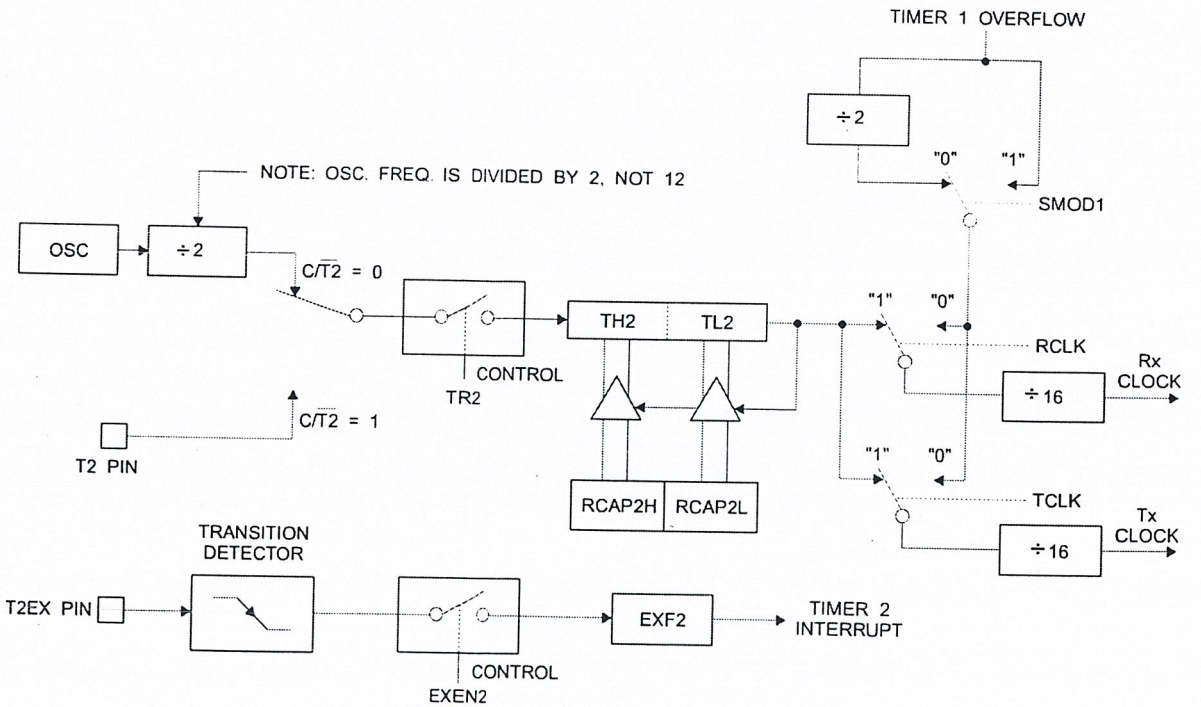


Figure 8. Timer 2 in Baud Rate Generator Mode



Baud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 8.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate according to the following equation.

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either timer or counter operation. In most applications, it is configured for timer operation ($CP/\overline{T2} = 0$). The timer operation is different for Timer 2 when it is used as a baud rate generator. Normally, as a timer, it increments every machine cycle (at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (at 1/2 the oscillator frequency). The baud rate formula is given below.

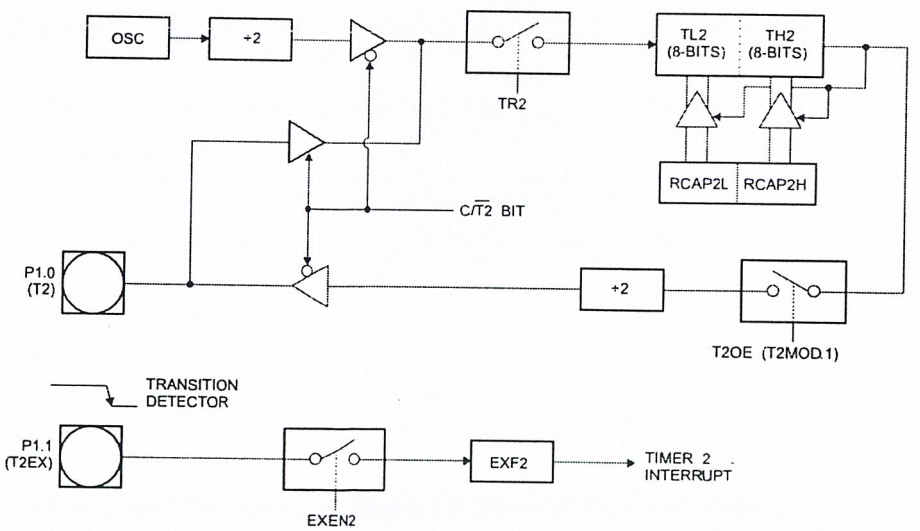
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - \text{RCAP2H,RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 8. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus, when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

Note that when Timer 2 is running (TR2 = 1) as a timer in the baud rate generator mode, TH2 or TL2 should not be read from or written to. Under these conditions, the Timer is incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

Figure 9. Timer 2 in Clock-Out Mode



Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 9. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz at a 16 MHz operating frequency.

To configure the Timer/Counter 2 as a clock generator, bit $C/\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L), as shown in the following equation.

$$\text{Clock-Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times [65536-(\text{RCAP2H,RCAP2L})]}$$

In the clock-out mode, Timer 2 roll-overs will not generate an interrupt. This behavior is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies cannot be determined independently from one another since they both use RCAP2H and RCAP2L.

Interrupts

The AT89S52 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 10.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once.

Note that Table 5 shows that bit position IE.6 is unimplemented. In the AT89S52, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and that bit will have to be cleared in software.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

Table 5. Interrupt Enable (IE) Register

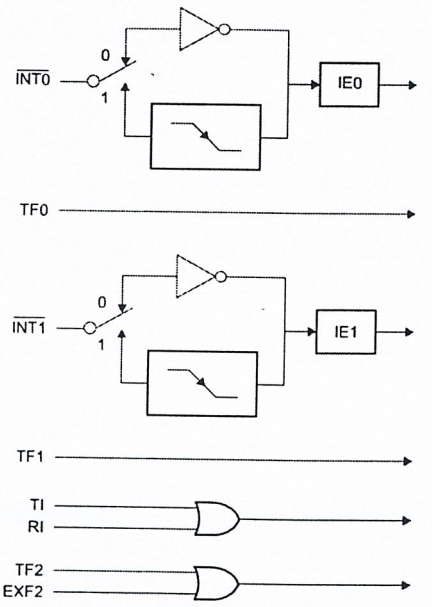
(MSB)							(LSB)
EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	Serial Port interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.
User software should never write 1s to unimplemented bits, because they may be used in future AT89 products.		

Figure 10. Interrupt Sources



Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 11. 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 12. 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.

Idle Mode

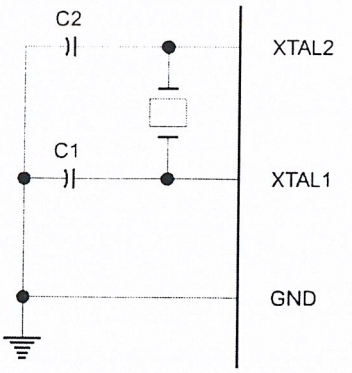
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware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

Note that when idle mode 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 mode is terminated by a reset, the instruction following the one that invokes idle mode should not write 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. Exit from Power-down mode can be initiated either by a hardware reset or by an enabled external interrupt. 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.

Figure 11. Oscillator Connections



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

Figure 12. External Clock Drive Configuration

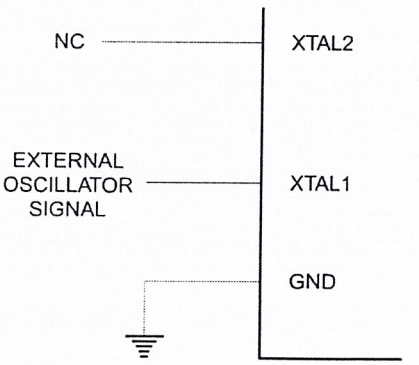


Table 6. Status of External Pins During Idle and Power-down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Program Memory Lock Bits

The AT89S52 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the following table.

Table 7. Lock Bit Protection Modes

Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No program lock features
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash memory is disabled
3	P	P	U	Same as mode 2, but verify is also disabled
4	P	P	P	Same as mode 3, but external execution is also disabled

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value and holds that value until reset is activated. The latched value of $\overline{EA}$ must agree with the current logic level at that pin in order for the device to function properly.

Programming the Flash – Parallel Mode

The AT89S52 is shipped with the on-chip Flash memory array ready to be programmed. The programming interface needs a high-voltage (12-volt) program enable signal and is compatible with conventional third-party Flash or EPROM programmers.

The AT89S52 code memory array is programmed byte-by-byte.

Programming Algorithm: Before programming the AT89S52, the address, data, and control signals should be set up according to the Flash programming mode table and Figures 13 and 14. To program the AT89S52, take the following steps:

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12V.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 50 µs.

Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

$\overline{Data}$ Polling: The AT89S52 features $\overline{Data}$ Polling to indicate the end of a byte write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P0.7. Once the write cycle has been completed, true $\overline{data}$ is valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{Busy}$: The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.0 is pulled low after ALE goes high during programming to indicate $\overline{BUSY}$. P3.0 is pulled high again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The status of the individual lock bits can be verified directly by reading them back.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 100H, and 200H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(100H) = 52H indicates 89S52
(200H) = 06H

Chip Erase: In the parallel programming mode, a chip erase operation is initiated by using the proper combination of control signals and by pulsing ALE/$\overline{PROG}$ low for a duration of 200 ns - 500 ns.

In the serial programming mode, a chip erase operation is initiated by issuing the Chip Erase instruction. In this mode, chip erase is self-timed and takes about 500 ms.

During chip erase, a serial read from any address location will return 00H at the data output.

Programming the Flash – Serial Mode

The Code memory array can be programmed using the serial ISP interface while RST is pulled to V_{CC}. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After RST is set high, the Programming Enable instruction needs to be executed first before other operations can be executed. Before a reprogramming sequence can occur, a Chip Erase operation is required.

The Chip Erase operation turns the content of every memory location in the Code array into FFH.

Either an external system clock can be supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The maximum serial clock (SCK)

frequency should be less than 1/16 of the crystal frequency. With a 33 MHz oscillator clock, the maximum SCK frequency is 2 MHz.

Serial Programming Algorithm

To program and verify the AT89S52 in the serial programming mode, the following sequence is recommended:

1. Power-up sequence:

   Apply power between VCC and GND pins.

   Set RST pin to "H".

   If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 33 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
2. Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 16.
3. The Code array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. The write cycle is self-timed and typically takes less than 1 ms at 5V.
4. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
5. At the end of a programming session, RST can be set low to commence normal device operation.

Power-off sequence (if needed):

Set XTAL1 to "L" (if a crystal is not used).

Set RST to "L".

Turn V_{CC} power off.

Data Polling: The Data Polling feature is also available in the serial mode. In this mode, during a write cycle an attempted read of the last byte written will result in the complement of the MSB of the serial output byte on MISO.

Serial Programming Instruction Set

The Instruction Set for Serial Programming follows a 4-byte protocol and is shown in Table 10.

Programming Interface – Parallel Mode

Every code byte in the Flash array can be programmed 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.

Table 8. Flash Programming Modes

Mode	V_{CC}	RST	$\overline{PSEN}$	ALE/ $\overline{PROG}$	$\overline{EA}$/ V_{PP}	P2.6	P2.7	P3.3	P3.6	P3.7	P0.7-0 Data	P2.4-0	P1.7-0
												Address	
Write Code Data	5V	H	L	(2)	12V	L	H	H	H	H	D_{IN}	A12-8	A7-0
Read Code Data	5V	H	L	H	H	L	L	L	H	H	D_{OUT}	A12-8	A7-0
Write Lock Bit 1	5V	H	L	(3)	12V	H	H	H	H	H	X	X	X
Write Lock Bit 2	5V	H	L	(3)	12V	H	H	H	L	L	X	X	X
Write Lock Bit 3	5V	H	L	(3)	12V	H	L	H	H	L	X	X	X
Read Lock Bits 1, 2, 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
Chip Erase	5V	H	L	(1)	12V	H	L	H	L	L	X	X	X
Read Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	X 0000	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	52H	X 0001	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	06H	X 0010	00H

Notes:
1. Each $\overline{PROG}$ pulse is 200 ns - 500 ns for Chip Erase.
2. Each $\overline{PROG}$ pulse is 200 ns - 500 ns for Write Code Data.
3. Each $\overline{PROG}$ pulse is 200 ns - 500 ns for Write Lock Bits.
4. RDY/$\overline{BSY}$ signal is output on P3.0 during programming.
5. X = don't care.

Figure 13. Programming the Flash Memory (Parallel Mode)

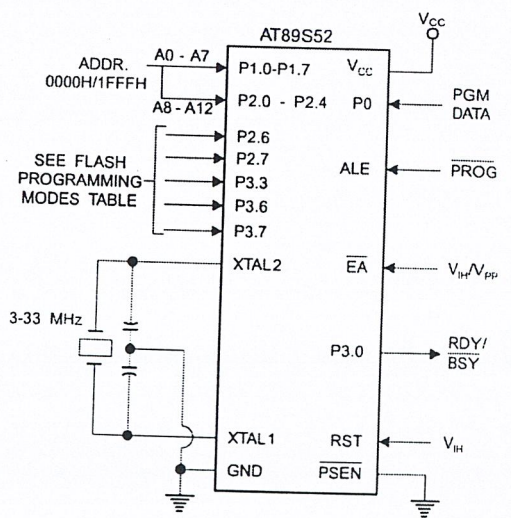
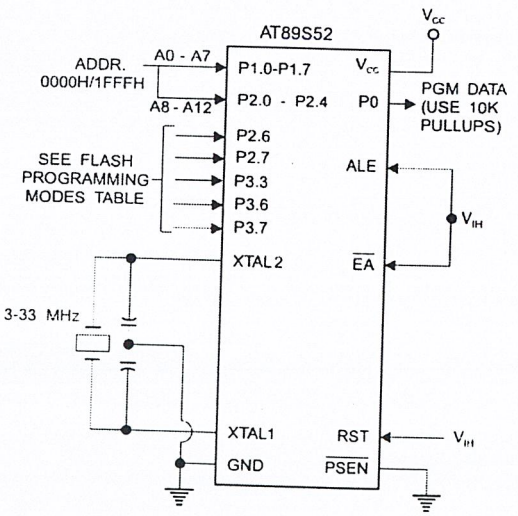


Figure 14. Verifying the Flash Memory (Parallel Mode)



Flash Programming and Verification Characteristics (Parallel Mode)

T_A = 20°C to 30°C, V_{CC} = 4.5 to 5.5V

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	11.5	12.5	V
I_{PP}	Programming Supply Current		10	mA
I_{CC}	V_{CC} Supply Current		30	mA
$1/t_{CLCL}$	Oscillator Frequency	3	33	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHAX}	Address Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHDX}	Data Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
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	0.2	1	µs
t_{AVQV}	Address to Data Valid		$48t_{CLCL}$	
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		50	µs

Figure 15. Flash Programming and Verification Waveforms – Parallel Mode

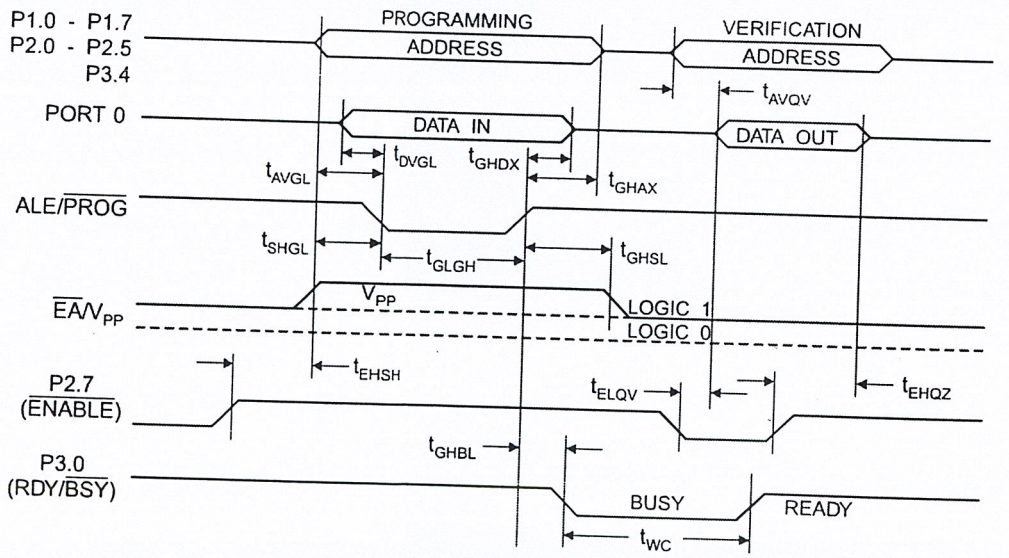
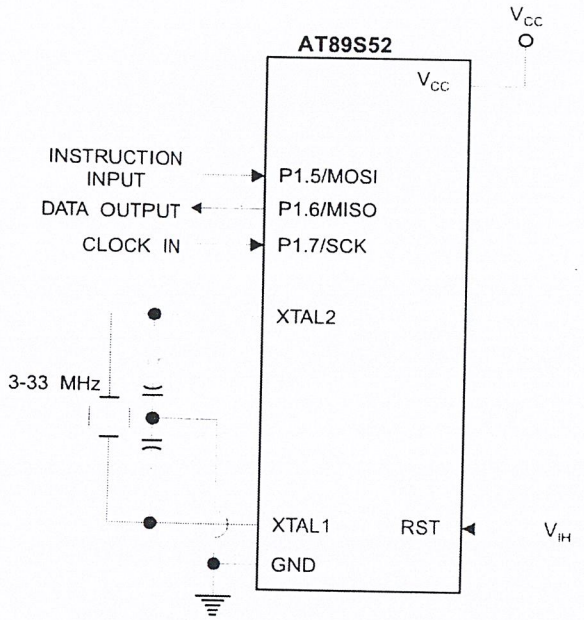


Figure 16. Flash Memory Serial Downloading



Flash Programming and Verification Waveforms – Serial Mode

Figure 17. Serial Programming Waveforms

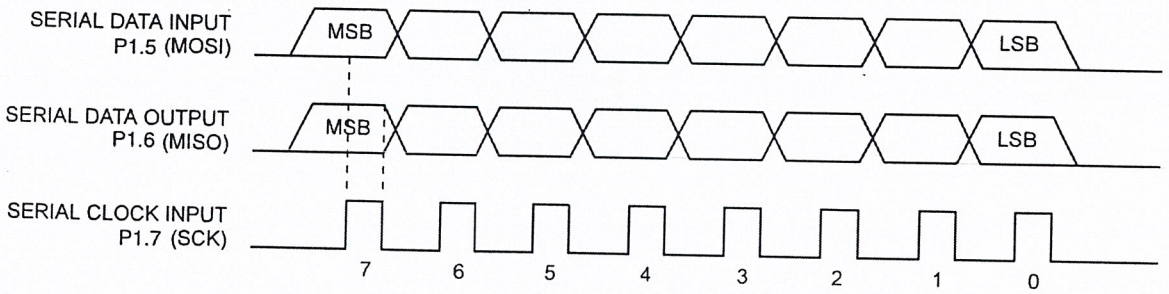


Table 9. Serial Programming Instruction Set

Instruction	Instruction Format				
	Byte 1	Byte 2	Byte 3	Byte 4	Operation
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx 0110 1001 (Output)	Enable Serial Programming while RST is high
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase Flash memory array
Read Program Memory (Byte Mode)	0010 0000	xxx A12 A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Read data from Program memory in the byte mode
Write Program Memory (Byte Mode)	0100 0000	xxx A12 A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Write data to Program memory in the byte mode
Write Lock Bits(2)	1010 1100	1110 00 B1 B2	xxxx xxxx	xxxx xxxx	Write Lock bits. See Note (2).
Read Lock Bits	0010 0100	xxxx xxxx	xxxx xxxx	xx LB3 LB2 LB1 xx	Read back current status of the lock bits (a programmed lock bit reads back as a '1')
Read Signature Bytes(1)	0010 1000	xxx A5 A4 A3 A2 A1 A0	xxx xxxx	Signature Byte	Read Signature Byte
Read Program Memory (Page Mode)	0011 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Read data from Program memory in the Page Mode (256 bytes)
Write Program Memory (Page Mode)	0101 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Write data to Program memory in the Page Mode (256 bytes)

Notes: 1. The signature bytes are not readable in Lock Bit Modes 3 and 4.

2. B1 = 0, B2 = 0 ---> Mode 1, no lock protection
   B1 = 0, B2 = 1 ---> Mode 2, lock bit 1 activated
   B1 = 1, B2 = 0 ---> Mode 3, lock bit 2 activated
   B1 = 1, B1 = 1 ---> Mode 4, lock bit 3 activated

   } Each of the lock bits needs to be activated sequentially before Mode 4 can be executed.

After Reset signal is high, SCK should be low for at least 64 system clocks before it goes high to clock in the enable data bytes. No pulsing of Reset signal is necessary. SCK should be no faster than 1/16 of the system clock at XTAL1.

For Page Read/Write, the data always starts from byte 0 to 255. After the command byte and upper address byte are latched, each byte thereafter is treated as data until all 256 bytes are shifted in/out. Then the next instruction will be ready to be decoded.

Serial Programming Characteristics

Figure 18. Serial Programming Timing

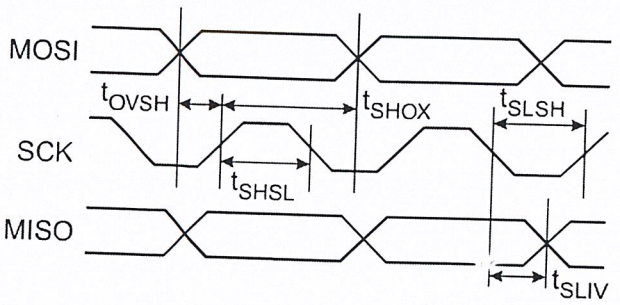


Table 10. Serial Programming Characteristics, T_A = -40° C to 85° C, V_{CC} = 4.0 - 5.5V (Unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0		33	MHz
t_{CLCL}	Oscillator Period	30			ns
t_{SHSL}	SCK Pulse Width High	2 t_{CLCL}			ns
t_{SLSH}	SCK Pulse Width Low	2 t_{CLCL}			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2 t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid	10	16	32	ns
t_{ERASE}	Chip Erase Instruction Cycle Time			500	ms
t_{SWC}	Serial Byte Write Cycle Time			64 t_{CLCL} + 400	µs

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	15.0 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

The values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 4.0V to 5.5V, unless otherwise noted.

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
V_{IL1}	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	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 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low Voltage(1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		10	30	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode(1)	V_{CC} = 5.5V		50	µA

Notes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10 mA
   Maximum I_{OL} per 8-bit port:
   Port 0: 26 mA Ports 1, 2, 3: 15 mA
   Maximum total I_{OL} for all output pins: 71 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.

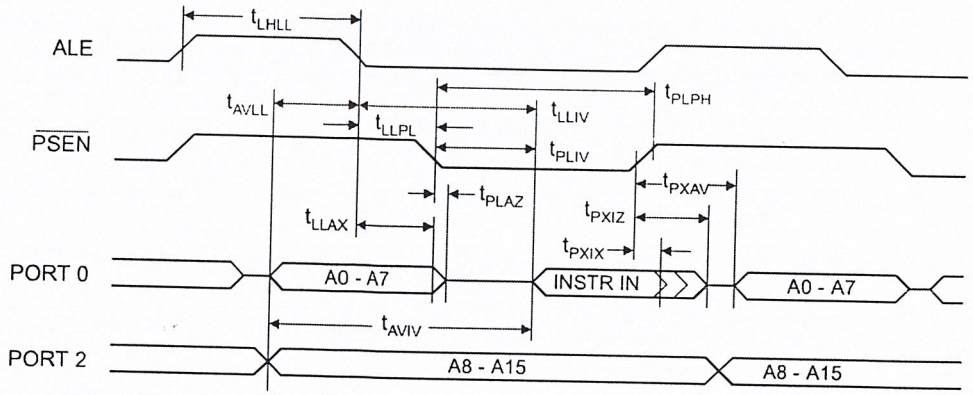
AC Characteristics

Under operating conditions, load capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; load capacitance for all other outputs = 80 pF.

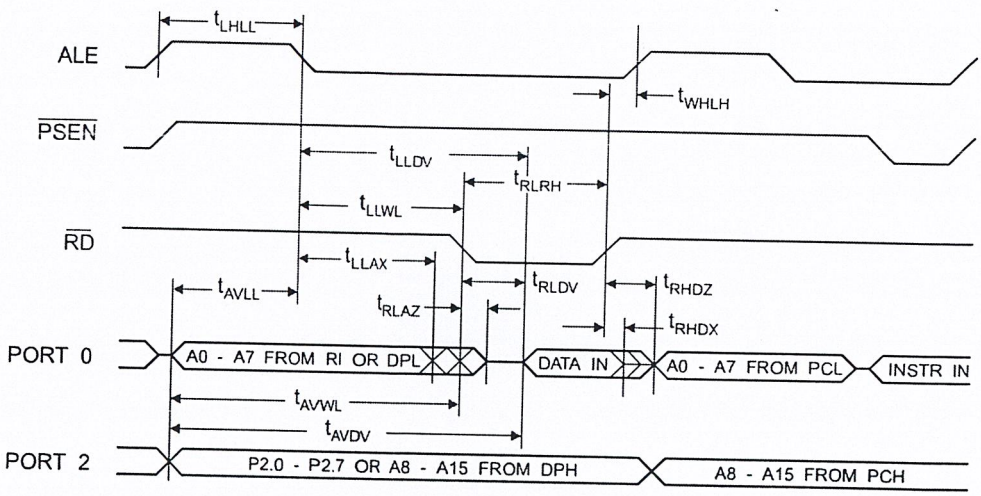
External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	33	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	Address Valid to ALE Low	43		$t_{CLCL}-25$		ns
t_{LLAX}	Address Hold After ALE Low	48		$t_{CLCL}-25$		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		$4t_{CLCL}-65$	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	43		$t_{CLCL}-25$		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}-45$		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}-60$	ns
t_{PXIX}	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input Instruction Float After $\overline{PSEN}$		59		$t_{CLCL}-25$	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	75		$t_{CLCL}-8$		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}-80$	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}-90$	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		97		$2t_{CLCL}-28$	ns
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CLCL}-150$	ns
t_{AVDV}	Address to Valid Data In		585		$9t_{CLCL}-165$	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}-75$		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		$t_{CLCL}-30$		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}-130$		ns
t_{WHQX}	Data Hold After $\overline{WR}$	33		$t_{CLCL}-25$		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	$t_{CLCL}-25$	$t_{CLCL}+25$	ns

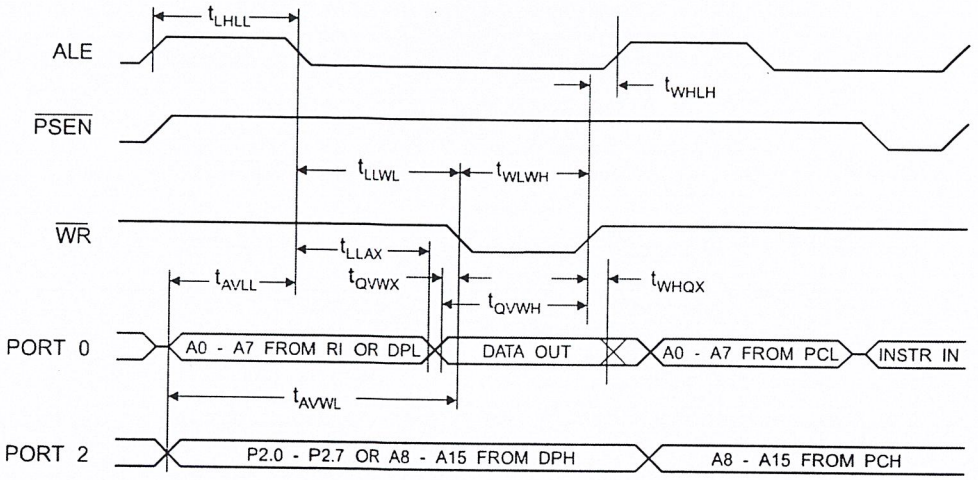
External Program Memory Read Cycle



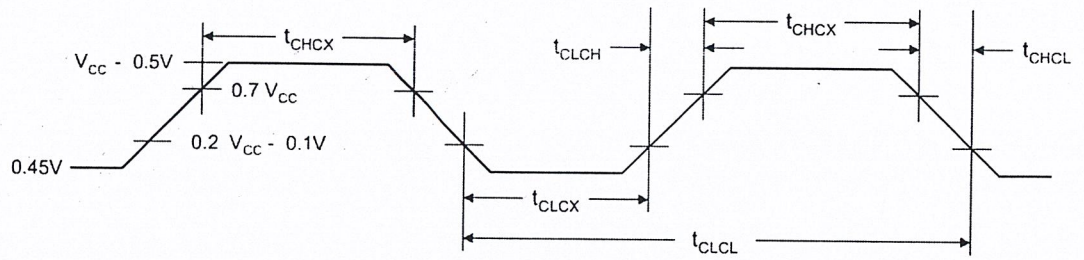
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

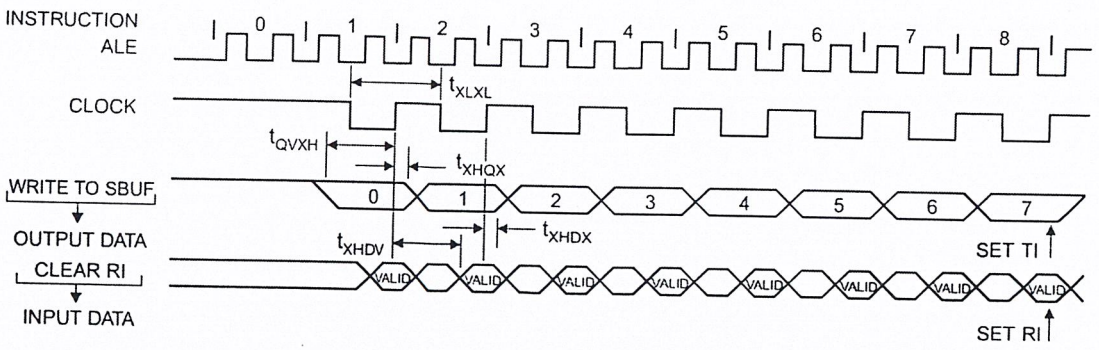
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	33	MHz
t_{CLCL}	Clock Period	30		ns
t_{CHCX}	High Time	12		ns
t_{CLCX}	Low Time	12		ns
t_{CLCH}	Rise Time		5	ns
t_{CHCL}	Fall Time		5	ns

Serial Port Timing: Shift Register Mode Test Conditions

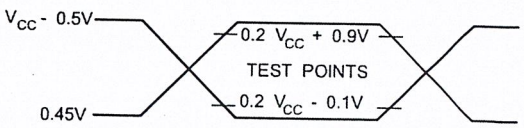
The values in this table are valid for V_{CC} = 4.0V to 5.5V and Load Capacitance = 80 pF.

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		$2t_{CLCL}$-80		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

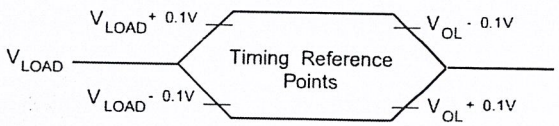


AC Testing Input/Output Waveforms[1]



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 a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.