# การควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

# SPEED CONTROLLING OF DC MOTOR



นายณัฐ วรินทรเวช

นายสุรเดช คชเสนา



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

# SPEED CONTROLLING OF DC MOTOR

NAT WARINTARAWECH

SURADECH KOTCHASENA


A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INDUSTRIAL INSTRUMENTATION TECHNOLOGY

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2001

**ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม**
**คณะวิศวกรรมศาสตร์**
**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**
**ใบรับรองปริญญานิพนธ์**

---

**หัวข้อปริญญานิพนธ์** การควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง
SPEED CONTROLLING OF DC MOTOR

**นักศึกษาผู้จัดทำ** นายณัฐ วรินทรเวช รหัสประจำตัว 41013406
นายสุรเดช คชเสนา รหัสประจำตัว 41013432

**ปริญญา** วิศวกรรมศาสตรบัณฑิต

**สาขาวิชา** วิศวกรรมการวัดคุม

**ปีการศึกษา** 2544

| **อาจารย์ผู้ควบคุมปริญญานิพนธ์** | **ลายมือชื่อ** |
|---|---|
| รศ.วิทยา ทิพย์สุวรรณพร | |

**วัน/เดือน/ปี ที่สอบ** วันพุธ ที่ 24 ตุลาคม พ.ศ. 2544

**สถานที่สอบ** ณ.ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

**ภาควิชารับรองแล้ว**

**(ผศ.ประสิทธิ์ จุลเสรีวงศ์)**
**หัวหน้าภาควิชาฯ**

| | |
|---|---|
| **หัวข้อปริญญานิพนธ์** | การควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง |
| | SPEED CONTROLLING OF DC MOTOR |
| **นักศึกษาผู้จัดทำ** | นายณัฐ วรินทรเวช |
| | นายสุรเดช คชเสนา |
| **อาจารย์ที่ปรึกษา** | รศ.วิทยา ทิพย์สุวรรณพร |
| **พุทธศักราช** | 2544 |


## บทคัดย่อ

โครงงานนี้เป็นการนำเสนอแนวทางอีกทางเลือกหนึ่งในการควบคุมและรักษาระดับความเร็วของมอเตอร์ไฟฟ้ากระแสตรงให้คงที่ เมื่อต้องรับภาระโหลดที่ค่าต่าง ๆ กัน โดยการประยุกต์ใช้คอมพิวเตอร์ร่วมกับไมโครคอนโทรลเลอร์ 8051 ในการควบคุมในรูปแบบ Digital Control System ไมโครคอนโทรลเลอร์ทำหน้าที่เป็นตัวผลิตสัญญาณควบคุมความเร็วรอบโดยรับการสั่งงานจากคอมพิวเตอร์และในขณะเดียวกันคอมพิวเตอร์ก็ทำหน้าที่เป็นตัวคำนวณทางคณิตศาสตร์ การควบคุมจะอยู่ในรูปอัตโนมัติ เมื่อมอเตอร์ต้องรับภาระโหลดความเร็วรอบของตัวมอเตอร์จะลดลง ไมโครคอนโทรลเลอร์จะรับค่าความเร็วรอบจริงเพื่อไปเปรียบเทียบกับค่าที่ต้องการหรือค่า Set Point เกิดค่าความผิดพลาดขึ้นเท่าไหร่คอมพิวเตอร์จะคำนวณและสั่งการให้ตัวไมโครคอนโทรลเลอร์ผลิตสัญญาณควบคุมให้สอดคล้องกับค่าผิดพลาดที่เกิดขึ้น เพื่อควบคุมและรักษาระดับความเร็วรอบให้เป็นไปตามค่าที่ต้องการหรือค่า Set Point

| | |
|---|---|
| **Thesis Title** | Speed Controlling Of DC Motor |
| **Authers** | Mr.Nat Warintarawech |
| | Mr.Suradech Kotchasena |
| **Thesis Advisor** | Assoc.Prof.Vittaya Tipsuwanporn |
| **Year** | 2001 |


## ABSTRACT

This project is present alternative for control and maintain speed of DC Motor, when was vaiitied on load. In accordance with apply computer with microcontroller MCS-8051 were control on Digital Control System, however microcontroller produce the signal for speed control with reception from calculate and a while computer was calculate mathematics pattern. The control will be automatic path when motor decrease on load. Microcontroller recept the real speed compare with set point for calculating error. The computer will calculate and command the microcontroller produce signal control that is agree error value for control and maintain speed of require value or set point.

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงได้ด้วยดีโดยได้รับคำแนะนำและคำปรึกษาตลอดจนถึงแนวทางการแก้ไขปัญหาจาก อาจารย์ รศ.วิทยา ทิพย์สุวรรณพร ตลอดจนเพื่อนๆ ในภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรมทุกคน ที่คอยให้กำลังใจและความห่วงใย

ขอกราบขอบพระคุณบิดา มารดาและวงศาคณาญาติ อันเป็นที่รักยิ่งที่ให้การสนับสนุนอย่างดีเยี่ยมและเป็นแรงบันดาลใจในการทำปริญญานิพนธ์ฉบับนี้

ผู้วิจัยใคร่ที่จะขอมอบคุณค่าตลอดถึงประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ แด่ผู้มีพระคุณทุกท่านไว้ ณ โอกาสนี้

คณะผู้จัดทำ

# สารบัญ

หน้า

# สารบัญรูป

**รูปที่** **หน้า**

# สารบัญรูป (ต่อ)

**รูปที่** **หน้า**

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

ในปัจจุบันได้มีการนำมอเตอร์ไฟฟ้ากระแสตรง ( D.C MOTOR ) มาใช้ในงานอุตสาหกรรมอย่างมากมายซึ่งในการใช้งานมอเตอร์ไฟฟ้ากระแสตรงในงานอุตสาหกรรมบางประเภทนั้นต้องมีการควบคุมความเร็วรอบของมอเตอร์ไฟฟ้ากระแสตรงเพื่อให้เป็นไปตามความต้องการของการผลิตและเพื่อให้ผลผลิตมีประสิทธิภาพมากที่สุดดังนั้นการควบคุมความเร็วให้เป็นไปตามความต้องการจึงเป็นสิ่งจำเป็นอย่างยิ่งและโครงงานนี้ก็เป็นโครงงานที่ศึกษาถึงวิธีการที่จะควบคุมความเร็วรอบของมอเตอร์ไฟฟ้ากระแสตรงโดยการประยุกต์ใช้คอมพิวเตอร์ในการควบคุมร่วมกับไมโครคอนโทรลเลอร์ 8051 ในรูปแบบของการควบคุมโดย Digital Control System

เราจะเห็นได้ว่า ตัวแปรที่จะมีอิทธิพลต่อความเร็วของมอเตอร์ไฟฟ้ากระแสตรงนั้นก็คือ

1. ค่าแรงเคลื่อนไฟฟ้าที่มอเตอร์ได้รับ ( $V_t$ )

$$\omega_m \propto V_t$$

2. จำนวนเส้นแรงแม่เหล็กต่อขั้วของมอเตอร์ ($\phi$ )

$$\omega_m \propto 1/\phi$$

## 1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์นี้จะศึกษาและออกแบบสร้างชุดจำลองการควบคุมมอเตอร์ไฟฟ้ากระแสตรง เพื่อนำเสนอวิธีการควบคุมความเร็วมอเตอร์ไฟฟ้ากระแสตรงด้วยการควบคุม Duty Cycle โดยที่จะรักษาระดับความเร็วของมอเตอร์ไฟฟ้ากระแสตรงให้คงที่ เมื่อต้องรับภาระโหลด

## 1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์เล่มนี้จะกล่าวถึงการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรงและศึกษาความสัมพันธ์ระหว่างค่าแรงดันไฟฟ้ากับค่ากระแสไฟฟ้าของมอเตอร์ เมื่อเกิดการเปลี่ยนแปลงของความเร็วรอบ ทั้งในขณะที่ NO LOAD และ ON LOAD

## 1.4 ขั้นตอนการศึกษา

การทำวิทยานิพนธ์ฉบับนี้เริ่มจากขั้นตอนการศึกษาถึงคุณสมบัติต่างๆของมอเตอร์ไฟฟ้ากระแสตรง การควบคุมแบบ PID การใช้งานไมโครคอนโทรลเลอร์ MCS-51 ตลอดจนวิธีการ Interface ระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ MCS-51 โดยใช้ Visual Basic เขียนหน้าต่างการควบคุมต่าง ๆ เพื่อสั่งการควบคุมและแสดงค่าของผลตอบสนอง

# บทที่ 2
# ความรู้เบื้องต้นเกี่ยวกับการควบคุมความเร็ว DC MOTOR

## 2.1 มอเตอร์ไฟฟ้ากระแสตรง( DC MOTOR )

มอเตอร์ไฟฟ้ากับเครื่องกำเนิดไฟฟ้าต่างกันตรงที่พลังงานที่ป้อนเข้า (Input Power)โดยพลังงานที่ป้อนเข้ามอเตอร์เป็นพลังงานไฟฟ้าและพลังงานที่จ่ายออก ( Output Power ) เป็นพลังงานกล แต่พลังงานที่ป้อนเข้าเครื่องกำเนิดไฟฟ้าเป็นพลังงานกลและพลังงานที่จ่ายออกเป็นพลังงานไฟฟ้า นอกจากนี้ทิศทางการไหลของกระแสไฟฟ้าก็ต่างกันด้วยคือ มอเตอร์ไฟฟ้าจะรับกระแสไฟฟ้าจากภายนอก ส่วนเครื่องกำเนิดไฟฟ้าจะจ่ายกระแสไฟฟ้าออกไปภายนอก

### 2.1.1โครงสร้างของมอเตอร์ไฟฟ้ากระแสตรง

ส่วนประกอบที่สำคัญของมอเตอร์ไฟฟ้ากระแสตรงและเครื่องกำเนิดไฟฟ้ากระแสตรงนั้นเหมือนกัน

**รูปที่ 2.1** แสดงส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง

A – ขดลวดสร้างสนามแม่เหล็ก (Field Winding)

B – ขั้วแม่เหล็ก (Pole Core)

C – ขดลวด Interpole หรือ Commutator Pole

D – Stator Yoke

E – Commutator Sectors

F – ชุดแปรงถ่าน (Brushes)

G – ลวดลัดอาร์มาเจอร์ (Armature Banding Wires)

I – เพลาของตัวหมุน (Rotor Shaft)

J – ช่อง (Slots) ลงขดลวดอาร์มาเจอร์

**ส่วนที่หยุดนิ่งอยู่กับที่** ( Stator ) ประกอบด้วย

ก. *เปลือกนอก* ( Frame หรือ Yoke ) เป็นตัวยึดขั้วแม่เหล็กของส่วนที่อยู่กับที่ พร้อมทั้งทำให้เส้นแรงแม่เหล็กที่เกิดจากขั้วแม่เหล็กวิ่งได้ครบวงจร นอกจากนี้เปลือกนอกยังทำหน้าที่เป็นตัวยึดสำหรับติดตั้งเครื่องจักรรวมถึงเป็นแป้นยึดลูกปืน ( bearing ) สำหรับเพลาของตัวหมุน การทำเปลือกนอกทำได้โดยการขึ้นรูป การใช้เหล็กหล่อ หรือ ใช้วิธีม้วนแผ่นเหล็กแล้วเชื่อมเป็นวง

ข. *แกนเหล็กของขั้วแม่เหล็กและขั้วแม่เหล็ก* ( Pole-core and Pole-shoe ) ทั้งสองส่วนนี้ประกอบกันขึ้นมาเพื่อทำหน้าที่เป็นแกนและขั้วของแม่เหล็ก เพื่อทำให้เกิดสนามแม่เหล็กขึ้นหลังจากที่มีกระแสไฟฟ้าไหลผ่านขดลวดตัวนำที่พันรอบแก่นเหล็ก

ค. *ขดลวดที่พันรอบแกนเหล็ก* ( Pole Coils ) ก็คือ Field Coil ที่ประกอบด้วยลวดกลมหรือลวดแบนที่ทำด้วยทองแดงหลาย ๆ เส้นนำมาทำให้เป็นรูปร่างของขดลวดตัวนำที่ต้องการแล้วนำขดลวดตัวนำนี้สวมทับลงไปบนแกนเหล็กของขั้วแม่เหล็ก ดังนั้นเมื่อมีกระแสผ่านขดลวดนี้ก็ทำให้ขั้วแม่เหล็กมีอำนาจเป็นแม่เหล็กขึ้นมาโดยการผลิตเส้นแรงแม่เหล็กให้เกิดขึ้น เส้นแรงแม่เหล็กนี้จะถูกตัดผ่านตัวนำภายในอาร์มาเจอร์อีกทีหนึ่ง

**ส่วนที่เคลื่อนที่หมุนไปได้รอบตัว** ( Rotor ) ประกอบด้วย

ก. *แกนเหล็กของอาร์มาเจอร์* ( Armature Core ) ทำจากแผ่นเหล็กซิลิกอนหนาประมาณ 0.5 มิลลิเมตร ผิวทั้ง 2 ข้างจะฉาบด้วยฉนวนแล้วนำมาอัดซ้อนเป็นรูปทรงกระบอกจะทำเป็นร่อง ( Slot ) เรียงตามแนวเส้นรอบวงรอบนอกของแกนเหล็กเพื่อที่ใช้พันขดลวดอาร์มาเจอร์ ส่วนตรงกลางก็เจาะรูเป็นวงกลมเพื่อเอาไว้ใส่แกนเหล็ก ( Shaft ) แล้วก็บากเป็นช่องสี่เหลี่ยมของรูหนึ่งเพื่อใส่ตัวยึด ( Lock ) หรือ กุญแจ ( Key ) ทั้งนี้เพื่อไม่ให้เกิดการเคลื่อนที่ขึ้นระหว่างตัว

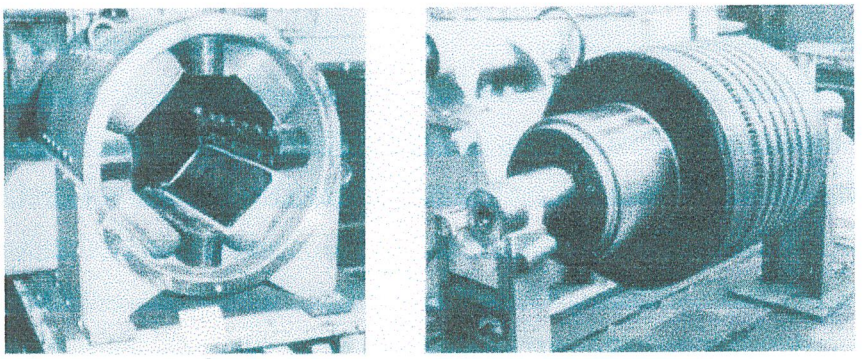
อาร์มาเจอร์กับแกนเหล็ก ( Shaft ) นอกจากนั้นยังเจาะรูอากาศ ( Air hole ) เล็ก ๆ ทะลุผ่านอาร์มาเจอร์เพื่อระบายความร้อนด้วย

ข. *ขดลวดอาร์มาเจอร์* ( Armature winding ) ก็คือขดลวดที่พันอยู่ในร่องของอาร์มาเจอร์

ค. *คอมมิวเตเตอร์* ( Commutator ) มีหน้าที่ คือ เป็นตัวที่เพิ่มความสะดวกในการนำกระแสออกมาจากตัวนำที่พันอยู่ในอาร์มาเจอร์และเป็นตัวกลับกระแสไฟสลับที่เกิดขึ้นภายในอาร์มาเจอร์ให้เป็นกระแสไฟตรง หรือ กระแสที่ไหลไปยังวงจรผ่านนอกในทิศทางเดียวกัน รูปร่างของมันเป็นรูปทรงกระบอก ซี่ต่อซี่ที่ต่อชิดกันนั้นคั่นไว้ด้วยฉนวนหนาที่แข็งแรง แต่ละซี่ต่อไปยังขั้วต่อของคอมมิวเตเตอร์ เพื่อให้ตัวนำที่ฝังอยู่ในอาร์มาเจอร์ยึดเกาะได้มั่นคงอีกทีหนึ่ง การป้องกันไม่ให้ซี่คอมมิวเตเตอร์ต่าง ๆ เหล่านี้ต้องกระเด็นหลุดไปอันเนื่องมาจากแรงหนีศูนย์กลาง จึงใช้ห่วงวงแหวนที่ทำด้วยไมก้ายึดซี่คอมมิวเตเตอร์ทั้งหมดไว้

ง. *แกนหมุน* ( Shaft ) เป็นตัวรับน้ำหนักในส่วนต่าง ๆ ของโรเตอร์ทั้งหมด โดยถ่ายทอดน้ำหนักนี้ไปยังลูกปืน ( Bearing ) ที่รองรับแกนนี้อีกทีหนึ่ง และมีหน้าที่เป็นตัวรับหรือถ่ายทอดการหมุนหรือการเคลื่อนที่ต่าง ๆ ที่เกิดขึ้นกับโรเตอร์ แกนหมุนนี้เป็นที่ยึดเกาะของคอมมิวเตเตอร์ด้วย โดยมีฉนวนที่ทำด้วยไมก้ากั้นไว้ระหว่างคอมมิวเตเตอร์กับแกนหมุน

จ. *แปรงถ่านและลูกปืน* ( Brushes and Bearings ) แท่งถ่านมีหน้าที่เป็นตัวรวบรวมกระแสที่ได้จากคอมมิวเตเตอร์เพื่อส่งต่อไปยังวงจรภายนอก รูปร่างของแปรงถ่านเป็นแท่งสี่เหลี่ยมผืนผ้า ซึ่งทำด้วยผงถ่านอัดแน่นเป็นก้อน แท่งถ่านเหล่านี้ถูกยึดติดอยู่กับที่จับแท่งถ่าน ( Brush Holder) ซึ่งเป็นกล่องสำหรับใส่แท่งถ่านลงไปที่ยึดติดนี้ยึดติดกับเปลือกนอกอีกทีหนึ่ง ฉะนั้นหน้าสัมผัสกับซี่คอมมิวเตเตอร์ ส่วนตรงข้ามก็ถูกกดจากสปริงอีกทีหนึ่ง ตรงด้านที่ถูกกดของแปรงถ่านต่อเข้ากับเส้นลวดทองแดงเล็ก ๆ ที่ถักเป็นเปีย เพื่อส่งต่อกระแสที่ได้จากคอมมิวเตเตอร์นั้นว่ามีมากน้อยเพียงใดส่วนลูกปืนนั้นเป็นตัวที่ใช้สำหรับรับน้ำหนักทั้งหมดที่ได้รับจากตัวหมุน และยังช่วยลดแรงเสียดทานที่แกนหมุนของตัวหมุนกระทำกับลูกปืนนั้น ปกติแล้วลูกปืนนี้จะยึดติดอยู่ที่ฝาครอบทั้งสองด้านที่จะต้องยึดติดกับเปลือกนอกของเครื่องกำเนิดไฟอีกทีหนึ่ง



**รูปที่ 2.2** แสดงส่วนประกอบที่เป็น Stator and Rotor

2.1.2 **ชนิดของมอเตอร์ไฟฟ้ากระแสตรง**

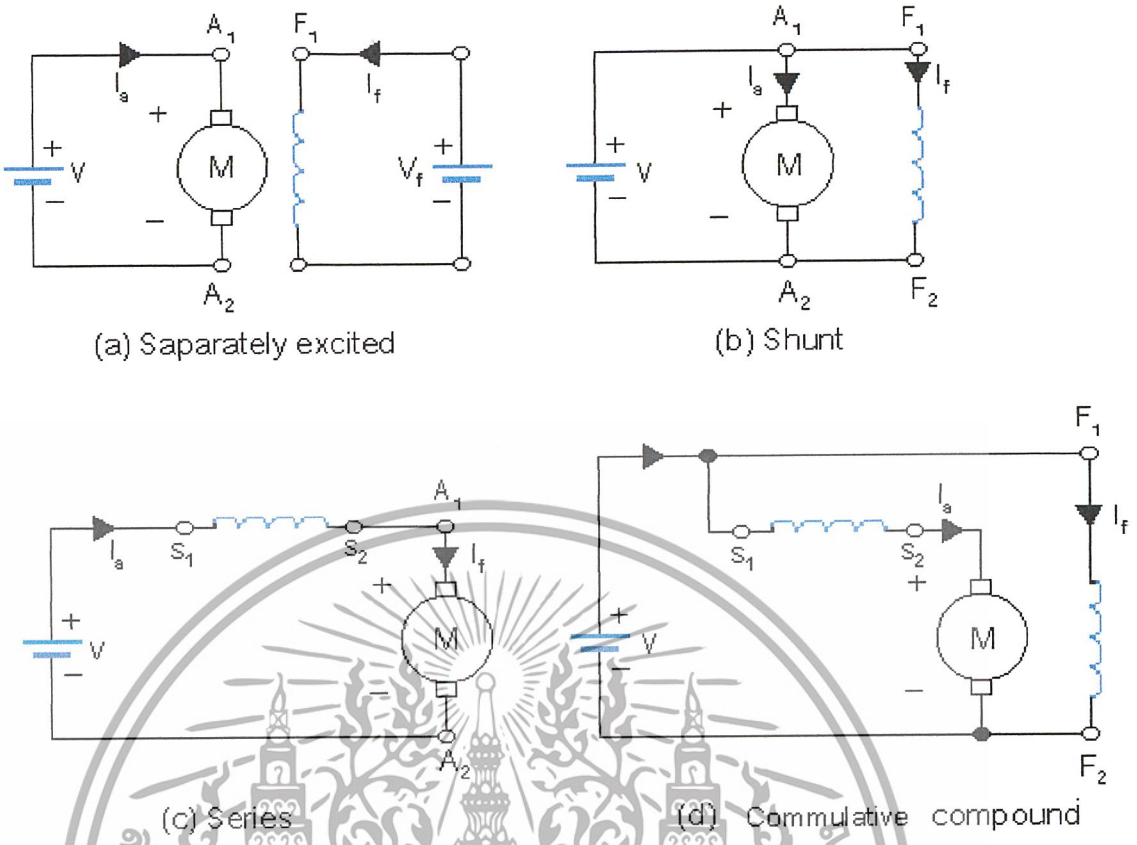
มอเตอร์ไฟฟ้ากระแสตรง แบ่งเป็น 4 ชนิด ดังนี้

1.มอเตอร์ไฟฟ้ากระแสตรงแบบแยกขดกระตุ้น ( Separately Excited DC Motor )

2.มอเตอร์ไฟฟ้ากระแสตรงแบบขนาน ( Shunt DC Motor )

3.มอเตอร์ไฟฟ้ากระแสตรงแบบอนุกรม ( Series DC Motor )

4.มอเตอร์ไฟฟ้ากระแสตรงแบบผสม ( Commulative Compound DC Motor )




**รูปที่ 2.3** แสดงชนิดของมอเตอร์ไฟฟ้ากระแสตรง

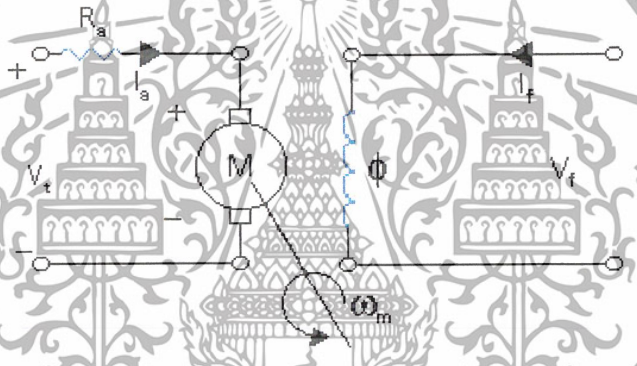
ในกรณีของของมอเตอร์ไฟฟ้ากระแสตรงแบบแยกขดกระตุ้นการควบคุมศักดิ์ดาที่ตกคร่อมอาร์มาเจอร์และขดสนาม ( Field ) นั้นแยกอิสระต่อกัน ส่วนมอเตอร์ไฟฟ้ากระแสตรงแบบขนาน ขดสนามและอาร์มาเจอร์ต่อกับแหล่งจ่ายไฟเดียวกัน การที่จะควบคุมแยกกันทำได้โดยวิธีเดียว คือ การเพิ่มความต้านทาน ( R ) ภายในวงจร แต่เป็นวิธีการควบคุมที่ไม่มีประสิทธิภาพ

ในกรณีของมอเตอร์ไฟฟ้ากระแสตรงแบบอนุกรมกระแสที่ไหลผ่านขดสนาม (Field flux) จึงขึ้นอยู่กับกระแสอาร์มาเจอร์ด้วย ส่วนมอเตอร์ไฟฟ้ากระแสตรงแบบผสม แรงเคลื่อนแม่เหล็กไฟฟ้า ( magnetomotive force ; mmf ) ของสัดส่วนสนามที่ต่ออนุกรมกับมอเตอร์อยู่ มีผลต่อกระแสอาร์มาเจอร์ด้วยและมีทิศทางเดียวกับแรงเคลื่อนแม่เหล็กของขดสนามที่ต่อขนานกับมอเตอร์

### 2.1.3 การควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง ( Speed control of DC Motor )

ในการที่จะศึกษาถึงวิธีในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง ( DC Motor ) นั้น ก่อนอื่นเราจะต้องทราบเสียก่อนว่าตัวแปรต่างๆ ที่จะมีผลต่อความเร็วของมอเตอร์ไฟฟ้ากระแสตรงนั้นมีอะไรบ้าง และตัวแปรแต่ละตัวที่มีอิทธิพลต่อความเร็วของมอเตอร์ไฟฟ้ากระแสตรงที่เราทราบนั้นจะมีข้อดีและข้อเสียอย่างไร ในการที่จะนำไปใช้เพื่อที่จะควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

เพื่อที่เราจะสามารถหาตัวแปรที่จะมีผลต่อความเร็วของมอเตอรไฟฟ้ากระแสตรงได้นั้น เราจะเริ่มพิจารณาจากวงจรสมมูลของมอเตอร์ไฟฟ้ากระแสตรง




**รูปที่** 2.4 แสดงวงจรสมมูลของ DC MOTOR

จากรูปเราจะได้สมการดังนี้

$$V_t = I_a * R_a + E_a$$

และจากสมการแรงเคลื่อนไฟฟ้าเหนี่ยวนำของมอเตอร์ไฟฟ้ากระแสตรงจะได้ว่า

$$E_a = K_a * \phi * \omega_m$$

ดังนั้นจะได้ว่า

$$V_t = I_a * R_a + K_a * \phi * \omega_m$$

โดยที่

$V_t$ = แรงเคลื่อนไฟฟ้าที่มอเตอร์ได้รับ

$E_a$ = แรงเคลื่อนไฟฟ้าที่เหนี่ยวนำขึ้นในอาร์มาเจอร์

$I_a$ = กระแสไฟฟ้าที่ไหลในอาร์มาเจอร์

$R_a$ = ความต้านทานของอาร์มาเจอร์

$K_a$ = ค่าคงที่ = ( Z * P ) / ( 60*a )

Z = จำนวนตัวนำทั้งหมดในอาร์มาเจอร์ที่ต่ออนุกรมกัน

P = จำนวนขั้วแม่เหล็กของมอเตอร์

a = จำนวนแผง ( Path ) ที่ขนานกัน

= จำนวนเส้นแรงแม่เหล็ก/ขั้วมอเตอร์

N = ความเร็วรอบของมอเตอร์

เราสามารถหาความเร็วของมอเตอร์ไฟฟ้ากระแสตรงได้ดังนี้

$$K_a * \phi * \omega_m = V_t - ( I_a * R_a )$$

เพราะฉะนั้นเราจะได้ว่า

$$\omega_m = ( V_t - ( I_a * R_a ) ) / ( K_a * \phi )$$

เนื่องจากว่าเทอม $I_a * R_a$ นั้นมีค่าน้อยมากเมื่อเทียบกับ $V_t$ ทั้งนี้ก็เพราะว่า ค่าของความต้านทานของอาร์มาเจอร์ ( $R_a$ ) นั้นมีค่าน้อยมาก โดยที่ขนาดของความต้านทานจะยิ่งมีค่าน้อย เมื่อขนาดของมอเตอร์ไฟฟ้ากระแสตรงนั้นมีขนาดใหญ่มากขึ้น ดังนั้นเราจะได้ว่า

$$\omega_m = V_t / ( K_a * \phi )$$

นั่นก็คือ $\omega_m \propto V_t$

และ $\omega_m \propto 1 / \phi$

เราจะเห็นได้ว่า ตัวแปรที่จะมีอิทธิพลต่อความเร็วของมอเตอร์ไฟฟ้ากระแสตรงนั้นก็คือ

1. ค่าแรงเคลื่อนไฟฟ้าที่มอเตอร์ได้รับ ( $V_t$ )
2. จำนวนเส้นแรงแม่เหล็กต่อขั้วของมอเตอร์ ($\phi$ )

ดังนั้นการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรงนั้น จึงสามารถทำได้ 2 แบบด้วยกันคือ

1. การควบคุมแรงเคลื่อนไฟฟ้าที่จ่ายให้กับมอเตอร์ ( $V_t$ ) โดยการควบคุมความเร็วแบบนี้ จะเรียกว่า การควบคุมแรงเคลื่อนไฟฟ้าที่จ่ายให้กับมอเตอร์ ( Armature voltage control )

2. การควบคุมจำนวนเส้นแรงแม่เหล็กต่อขั้ว ($\phi$ ) โดยการควบคุมแบบนี้ จะเรียกว่า " การควบคุมโดยใช้เส้นแรงแม่เหล็ก ( Field control ) "

ถ้าเราพิจารณาถึงการเปลี่ยนแปลงค่าของความต้านทานที่อาร์มาเจอร์ เราก็สามารถเปลี่ยนแปลงความเร็วของมอเตอร์ได้เช่นกัน โดยที่เราจะเรียกการควบคุมในลักษณะนี้ว่า " การควบคุมโดยใช้ความต้านทาน ( Rheostatic control ) "

ในการควบคุมความเร็วของมอเตอร์ในลักษณะนี้ เราจะใช้เมื่อต้องการควบคุมความเร็วของมอเตอร์ให้ต่ำหรือช้ากว่าความเร็วที่พิกัด ( Rated speed ) ของมอเตอร์ที่ต้องการควบคุมนั้น โดยที่แรงเคลื่อนไฟฟ้าที่จ่ายให้กับมอเตอร์มีค่าคงที่อยู่ โดยที่ความต้านทานที่ต่ออนุกรมเข้าไปนี้ จะไปมีผลทำให้แรงเคลื่อนไฟฟ้าที่ตกคร่อมอาร์มาเจอร์เปลี่ยนแปลงไป ซึ่งก็คือการควบคุมแรงเคลื่อนไฟฟ้าที่ตกคร่อมอาร์มาเจอร์ในทางอ้อมนั่นเอง ( Indirect voltage control of armature ) ดังนั้นในที่นี้จึงมิได้แบ่งการควบคุมในลักษณะนี้เป็นหัวข้อหนึ่งในการควบคุมความเร็วของมอเตอร์ เนื่องจากการควบคุมความเร็วแบบนี้ เราจะต้องต่อความต้านทานอนุกรมเข้าไปในวงจร ดังนั้นเมื่อเราเปลี่ยนแปลงค่าความต้านทานในขณะที่แรงดันที่จ่ายให้กับมอเตอร์มีค่าคงที่อยู่นั้นก็เท่ากับว่าเราเปลี่ยนแปลงค่าของกระแสนั่นเอง

เนื่องจากว่า

$$T = K_I * \phi * I_a$$

โดยที่

$T$ = แรงบิดที่เกิดขึ้นที่เพลาของมอเตอร์

$K_I$ = ค่าคงที่ในสมการของแรง

$= K_a / ( 2 * \pi )$

$= [( Z * P ) / ( 60 * a )] / ( 2 * \pi )$

$\phi$ = จำนวนเส้นแรงแม่เหล็กต่อขั้ว

$I_a$ = กระแสที่ไหลในอาร์มาเจอร์

เราจะได้ว่า

$$T \propto \phi$$

$$T \propto I_a$$

ดังนั้นเมื่อเราเปลี่ยนแปลงค่าความต้านทานที่ต่ออนุกรมเข้าไปในวงจรอาร์มาเจอร์ ก็จะทำให้แรงบิดที่มอเตอร์เปลี่ยนแปลงไปด้วย ทั้งนี้ก็เพราะว่ากระแสไฟฟ้าที่ไหลในอาร์มาเจอร์เปลี่ยนแปลงไปในขณะที่จำนวนเส้นแรงแม่เหล็กต่อขั้วคงที่อยู่

เนื่องจากความเร็วของมอเตอร์แปรผันโดยตรงกับค่าความต้านทานที่อาร์มาเจอร์ ดังนั้นเมื่อเราเขียนกราฟแสดงความสัมพันธ์ระหว่างความเร็วกระแสไฟฟ้าในวงจรอาร์มาเจอร์ เราจะได้กราฟที่มีลักษณะที่เป็นเส้นตรง


**รูปที่** 2.5 กราฟแสดงความสัมพันธ์ระหว่างความเร็วกับกระแสของอาร์มาเจอร์

ในการควบคุมความเร็วของมอเตอร์ในลักษณะนี้จะไม่ดี ทั้งนี้ก็เพราะว่า

1. เป็นวิธีการที่สิ้นเปลือง เพราะจะมีกำลังไฟฟ้าที่สูญเสียไปอย่างมากในตัว ต้านทาน โดยที่จะสูญเสียไปในรูปของความร้อน และขณะเดียวกันเราจะต้องหาความต้านทานที่มีขนาดใหญ่มากด้วย เพราะเหตุที่ว่ากระแสที่ไหลผ่านอาร์มาเจอร์มีค่ามาก
2. เป็นวิธีที่ค่อนข้างจะแพง
3. เสถียรภาพของวงจรไม่ดี ( Unstable ) ด้วยเหตุผลดังกล่าว จึงทำให้การควบคุมในลักษณะนี้ไม่เป็นที่นิยมกันในปัจจุบัน

## 2.2 อินคริเม้นท์เอนโคดเดอร์

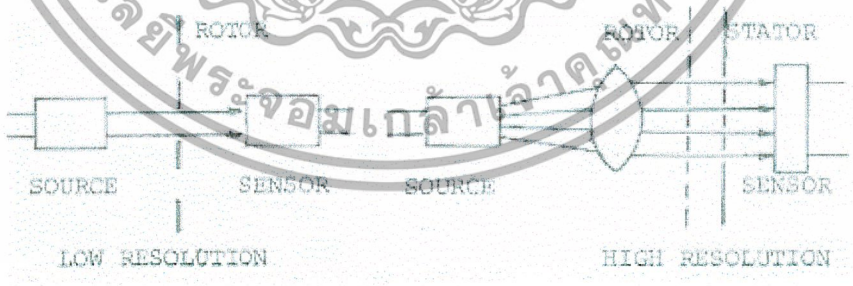
ในระบบการบังคับตำแหน่งหรือความเร็วของมอเตอร์ต้องใช้อินคริเม้นท์เอนโคดเดอร์หรือเอนโคดเดอร์สำหรับรักษาตำแหน่งและสำหรับสร้างสัญญาณป้อนกลับ โดยที่ตัวเอนโคดเดอร์จะสร้างสัญญาณพัลซ์ที่แปรผันตรงกับการหมุนของเพลาซึ่งสามารถนำไปใช้ในการรับรู้ความเร็วของเพลามอเตอร์ในรูปของอัตราจำนวนพัลส์ได้

อินคริเม้นท์เอนโคดเดอร์ประกอบด้วยส่วนที่สำคัญคือตัวกำเนิดแสง จานหมุน ( Rotary disk ) และตัวเซ็นเซอร์ บนแผนจานหมุนทำเป็นช่องโดยรอบและบนแผ่นอยู่กับที่จะมีช่องสำหรับให้แสงผ่านตรงข้ามไปยังตัวเซ็นเซอร์ ถ้าเป็นเอนโคดเดอร์ที่ใช้วัดความเร็วต่ำไม่ต้องมีแผ่นอยู่กับที่ก็ได้ ส่วนตัวกำเนิดแสงอาจจะเป็นหลอดไฟหรือ LED ก็ได้

ความละเอียดของเอนโคดเดอร์คือจำนวนคาบเวลาของสัญญาณเอาท์พุทต่อการหมุนของเพลา 1 รอบ ซึ่งบวกเป็นจำนวนพัลซ์ต่อรอบหรือจำนวนไซเกิลต่อ 360 องศา มุมทางเชิงกลหรือไซเกิลต่อองศา เอนโคดเดอร์ที่ใช้กันทั่วๆไปมีค่าความละเอียดตั้งแต่ 15 ถึง 10,000 พัลซ์ต่อรอบ

ในทางปฏิบัติเนื่องจากแสงที่ออกจากแหล่งกำเนิดเป็นลำแสงเดียว ถ้าเราต้องการให้แสงที่ผ่านช่องไปยังเซ็นเซอร์เป็นเส้นตรงพร้อมๆกัน ( Collimation ) ก็ทำได้โดยใช้เลนซ์หรือพาราโบลิครีแฟลคเตอร์

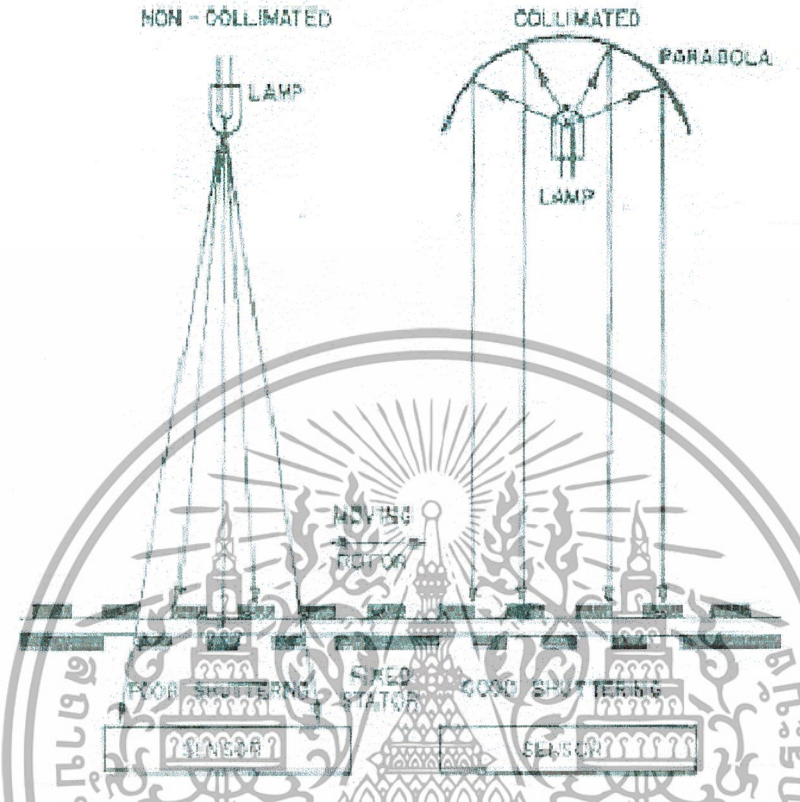

**รูปที่** 2.6 แสดงตัวอย่างกลไกของออฟโตอินคริเม้นท์




**รูปที่** 2.7 แสดงถึงตัวเซ็นเซอร์แบบมีช่องปิดเปิดให้แสงผ่านได้ช่องเดียวและแบบมีหลายช่อง

จำนวนพัลซ์ต่อ 1 รอบของสัญญาณที่เอนโคดเดอร์สร้างออกมาจะเท่ากับจำนวนช่องว่างบนแผ่นจานหมุน และความกว้างของช่องว่างกับความกว้างของแถบทึบระหว่างช่องว่างจะเท่ากัน เพราะฉะนั้นเราสามารถคำนวณหาความกว้างของช่องว่าง ( $\omega$ ) ได้จาก

$$\omega = \frac{\pi D}{2N}$$




**รูปที่** 2.8 แสดงถึงผลของแสงที่เดินในแนวเดียวกันและแสงที่ตกกระจาย

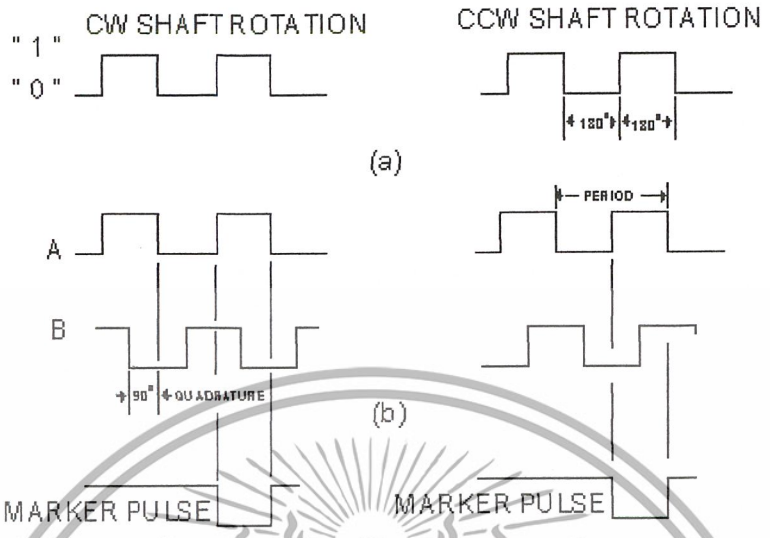
โดยที่

D = เส้นผ่านศูนย์กลางเฉลี่ยของเพ็ทเทิน

N = จำนวนของความละเอียดเป็นพัลซ์ต่อรอบ

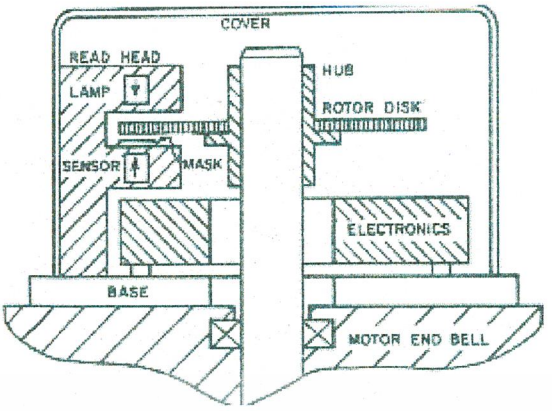
$\omega$ = ความกว้างของช่องว่าง

เอาท์พุทของเอนโคดเดอร์ โดยทั่วๆไป แล้วสัญญาณเอาท์พุทที่ออกจากเอนโคดเดอร์โดยตรงจะมีระดับไม่เพียงพอในการควบคุมหรือสำหรับการประมวลสัญญาณ ดังนั้นจึงต้องมีวงจรขยายและแปลงรูปร่างลูกคลื่นสัญญาณต่อไว้ในตัวเอนโคดเดอร์ด้วยเสมอ สัญญาณลูกคลื่นที่ได้จากตัวเซ็นเซอร์ปกติแล้วจะเป็นรูปสัญญาณสามเหลี่ยมหรือรูปสัญญาณซายน์ขึ้นอยู่กับความละเอียดที่ต้องการ รูปสัญญาณเหล่านี้สามารถทำให้เป็นสัญญาณรูปสี่เหลี่ยมได้โดยการต่อตัวคอมพาราเตอร์เข้ากับลิเนียร์แอมปลิไฟของเอนโคดเดอร์ก็จะได้เอาท์พุทเป็นลูกคลื่นสี่เหลี่ยมตาม




**รูปที่** 2.9 ( a ) แสดงตัวอย่างลูกคลื่นเอาท์พุทสี่เหลี่ยมของอุปกรณ์เอนโคดเดอร์ช่องเดียว
( b ) แสดงตัวอย่างสัญญาณเอนโคดเดอร์ 2 ช่องมีมุมเฟสต่างกัน $90^0$ (สองทาง )

จากรูป ( a ) แสดงถึงลูกคลื่นเอาท์พุทสี่เหลี่ยมของเอนโคดเดอร์ชนิด 1 ช่อง ไม่ว่าเพลาจะหมุนในทิศทางใดก็ได้สัญญาณออกมาเหมือนกันจึงเหมาะที่จะใช้กับงานที่ไม่กำหนดทิศทางเท่านั้น ส่วนในรูป ( b ) แสดงสัญญาณ 2 ชุดที่ได้จากเอนโคดเดอร์ชนิด 2 ช่องเฟสของสัญญาณ 2 ช่องนี้จะต่างกัน 90 องศาทางไฟฟ้าเราเรียกสัญญาณ 2 ช่องนี้ว่าเป็นควอตราเจอร์ ( Quadrature ) กันซึ่งเหมาะที่จะใช้ในการรับรู้ทิศทางการหมุนของเพลาหรือใช้ควบคุมระบบที่ซับซ้อนอื่นๆ จากสัญญาณในรูป( b ) จะเห็นได้ว่าสัญญาณทั้ง 2 ช่องจะเริ่มจาก 0 ถึง 1 และ 1 ถึง 0 ขึ้นอยู่กับทิศทางการหมุนของแผ่นหมุนของเอนโคดเดอร์ ในอินครีเม้นท์เอนโคดเดอร์บางชนิดจะมีพัลซ์ที่แสดงถึงจำนวนรอบของการหมุนสำหรับใช้เป็นศูนย์ในการอ้างอิงพัลซ์ที่ใช้แสดงจำนวนรอบนี้จะเกิดขึ้น 1 พัลซ์ต่อ 1 รอบ โดยทั่วไปแล้วใช้บอกถึงตำแหน่งเชิงกลหรือใช้เป็นสัญญาณเคลียรจำนวนที่นับไว้ในหน่วยเก็บข้อมูล




**รูปที่**2.10 แสดงตัวอย่างของเอนโคดเดอร์

## 2.3 ไมโครคอนโทรลเลอร์ MCS-51

### 2.3.1 โครงสร้างของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีด้วยกันหลายเบอร์ขึ้นอยู่กับ โครงสร้างภายในของมัน บางเบอร์จะมีหน่วยความจำภายในเป็นแบบ ROM บางเบอร์เป็นแบบ EPROM บางเบอร์มี RAM ภายใน 128 ไบต์บางเบอร์ 256 ไบต์ เป็นต้น ซึ่งรายละเอียดจะศึกษาได้จากคู่มือของมัน โดยตรงและลักษณะของขาต่างจะเหมือนกัน คุณสมบัติที่สำคัญของ MCS-51 มีดังนี้

-มีหน่วยความจำ ROM 4K bytes

-มีหน่วยความจำ RAM 128 bytes

-มีพอร์ท I/O ขนาด 8 บิต 4 พอร์ท

-มี Timer 16 บิต 2 ตัว

-สามารถอินเทอร์รัพท์ได้ 5 แหล่ง

-มีวงจรออสซิลเลเตอร์และวงจรนาฬิกาบนชิพ

-มีพอร์ทอนุกรมที่สามารถรับส่งข้อมูลแบบ Full Duplex ความเร็ว

-อ้างหน่วยความจำโปรแกรมภายนอกได้ 64 K

-อ้างหน่วยความจำข้อมูลภายนอกได้ 64 K

-สามารถประมวลผลที่ละบิตได้

-สามารถอ้างหน่วยความจำแบบบิตได้ 210 ตำแหน่ง

-หนึ่งวัฏจักรคำสั่งกินเวลาประมาณ 1 ไมโครวินาที ขณะทำงานด้วย สัญญาณ Clock 12 MHz

### 2.3.2 การรับส่งข้อมูลแบบอนุกรม

พื้นฐานการรับส่งข้อมูล การรับส่งข้อมูลในระบบคอมพิวเตอร์โดยทั่วไปจะหมายถึง การรับส่งข้อมูลแป้นจำนวนไบต์ ๆ ให้กับอุปกรณ์ที่เกี่ยวข้องกับคอมพิวเตอร์ ซึ่งอาจแบ่งประเภทการรับส่งข้อมูลได้ 2 แบบ

1. การรับส่งข้อมูลแบบขนาน ( Parallel )
2. การรับส่งข้อมูลแบบอนุกรม ( Serial )

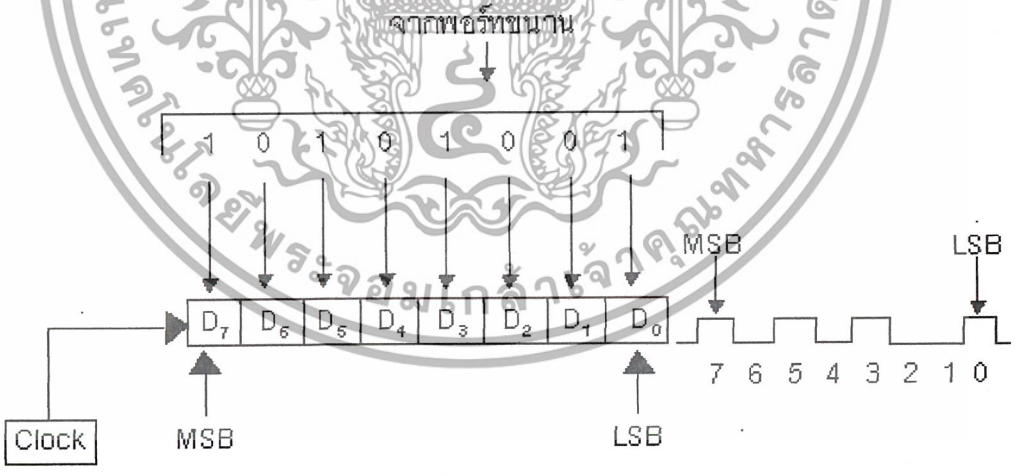
การรับส่งข้อมูลแบบขนานเป็นการรับส่งข้อมูล จำนวน 1 ไบต์ ออกไปทางพอร์ท ในเวลาเดียวกันในระบบคอมพิวเตอร์ 1 ไบต์จะมีจำนวน 8 บิต คือ $D_0 - D_7$ ถ้ามีการส่งข้อมูลแบบขนานจะใช้สายสัญญาณอย่างน้อย 9 เส้น คือ สาย Data 8 เส้นและสายกราวด์ 1 เส้นดังรูป


**รูปที่** 2.11 แสดงการรับส่งข้อมูลแบบขนาน

การรับส่งข้อมูลแบบอนุกรม คือ การรับส่งข้อมูลทีละบิต จนครบ 1 ไบต์ ถ้าต้องการส่งข้อมูล 1 ไบต์ คือ $D_0 - D_7$ อาจจะส่งทีละบิต $D_0$ ออกไปก่อนแล้วตามด้วย $D_1$ เรื่อย ๆ จนถึง $D_7$ การส่งข้อมูลทั้ง 2 แบบ มีข้อดีข้อเสียแตกต่างกัน คือ การส่งข้อมูลแบบขนานสามารถส่งข้อมูลได้รวดเร็ว คือ ส่งข้อมูลทีเดียวจะได้ข้อมูลครบ 1 ไบต์ แต่ถ้าต้องการส่งเป็นระยะทางไกล ๆ จะสิ้นเปลืองสายสัญญาณมาก ถ้าเป็นการส่งข้อมูลแบบอนุกรม เมื่อต้องการส่งข้อมูลเป็นระยะทางไกล ๆ จะช่วยประหยัดสายสัญญาณเนื่องจากจะใช้สายสัญญาณ เพียง 2 เส้น คือ สายสัญญาณกับสายกราวด์ แต่การรับส่งจะใช้เวลานานเนื่องจากการเป็นการส่งทีละบิต

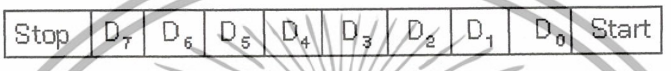
### 2.3.3 รูปแบบของการรับส่งข้อมูลแบบอนุกรม

เมื่อไมโครคอมพิวเตอร์ต้องการจะส่งข้อมูลอนุกรม ตัวไมโครคอมพิวเตอร์จะส่งสัญญาณออกไปทางพอร์ทซึ่งเป็นพอร์ทแบบขนานก่อน จากนั้นจะมีอุปกรณ์มาต่อพอร์ท เพื่อแปลงข้อมูลแบบขนานให้เป็นแบบอนุกรมอีกทีหนึ่ง ( Parallel – to – serial – conversion ) ตัวแปลงข้อมูลนี้อาจพิจารณาง่าย ๆ ว่าเป็น Shift Register ดังรูป เมื่อข้อมูลที่ส่งอยู่ใน Shift Register แล้วตัวต่อสัญญาณนาฬิกาจะเป็นตัวกระตุ้นให้ส่งข้อมูลบิตต่ำออกไปเวลาแรก จากนั้นจะส่งบิตต่อไปออกมา จากรูป จะเป็นการส่งข้อมูล A9H ออกไป




**รูปที่** 2.12 แสดงการส่งข้อมูลแบบอนุกรม

สำหรับตัวรับข้อมูลแบบอนุกรมเมื่อตัวรับข้อมูลจะเป็นการรับเข้ามาใน Shift Register แล้วส่งข้อมูลให้ไมโครคอมพิวเตอร์แบบขนานอีกทีหนึ่ง ( Serial – to – parallel)ระบบคอมพิวเตอร์ในปัจจุบันจะมีตัวแปลง Parallel – to – serial และ serial – to – parallel อยู่ในชิพไอซี เรียกว่า Universal Asynchronous Receiver Transmitter ( UART ) การรับส่งข้อมูลแบบอนุกรมนั้นจะต้องมีการเพิ่มเติมข้อมูลบางอย่างเข้าไปเพื่อให้การรับข้อมูลสามารถทำได้ถูกต้องมากขึ้น โดยการเติมค่าต่าง ๆ ลงไปตามรูปด้านล่าง




**รูปที่** 2.13 แสดงบิตต่าง ๆ ของข้อมูลที่ส่งแบบอนุกรม

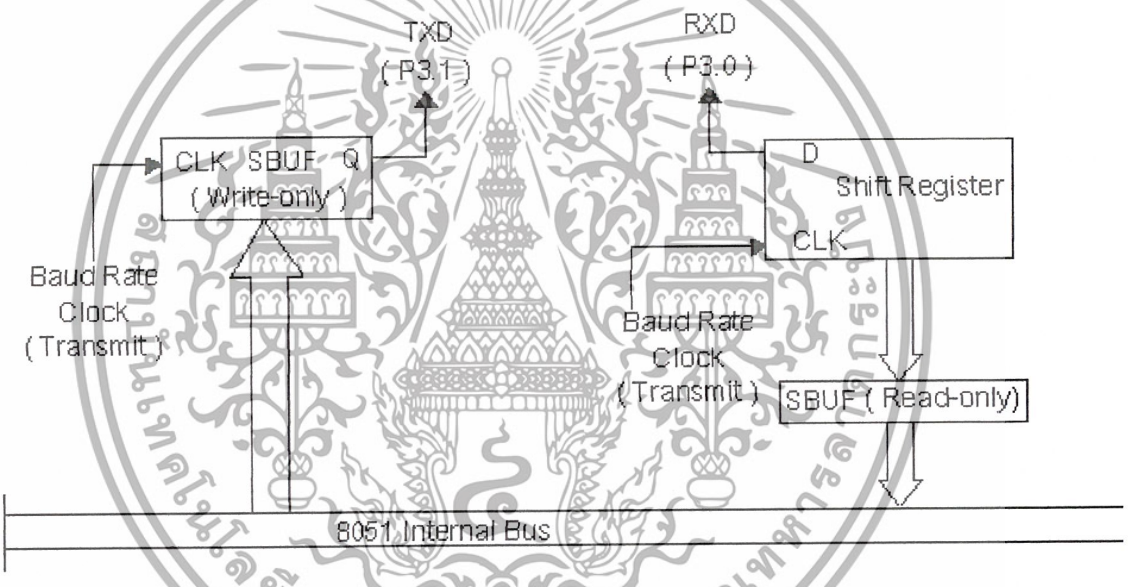
ถ้ามีการส่งข้อมูลแบบ 8 บิต จะต้องส่งบิตแรกออกไปก่อน เรียกว่า **บิตเริ่มต้น** ( Start Bit) ถ้ามีการส่งข้อมูลหลาย ๆ ไบต์ออกมา บิตนี้จะเป็นตัวบอกว่ามีข้อมูลใหม่มาแล้ว โดยทั่วไปบิตเริ่มต้นมักมีระดับลอจิกเป็น " 0 " ต่อจากค่าบิตเริ่มต้นจะเป็นข้อมูลบิต $D_0$ – $D_7$ จากนั้นจะตามด้วยบิตตรวจสอบความถูกต้อง ( Parity Bit ) ถ้าข้อมูล 8 บิตที่ส่งออกมา จำนวนของบิตมีค่าเป็น " 1 " เป็นจำนวนคู่บิตนี้จะมีค่าเป็น " 0 "แต่ถ้าจำนวนของบิตมีค่าเป็น " 1 " เป็นคี่ บิตนี้จะมีค่าเป็น " 1 " จากนั้น ข้อมูลที่ส่งออกไปจะตามด้วยบิตสิ้นสุดข้อมูล ( Stop Bit ) เพื่อเป็นการบอกว่าข่อมูลที่ส่งมา 8 บิตนั้นหมดแล้วตัวบิต Stop อาจมีจำนวนมากกว่า 1 บิต ก็ได้ เช่น $1\frac{1}{2}$ บิต , 2 บิต

การส่งข้อมูลแบบอนุกรมนั้นมีความเร็วของการส่งจะมีค่าเป็น บิตต่อวินาที เรียกว่า " Baud Rate "

### 2.3.4 MCS-51 กับการรับส่งข้อมูลแบบอนุกรม

การรับส่งข้อมูลแบบอนุกรมกับไมโครคอนโทรลเลอร์ MCS-51 นั้น ภายในชิพ MCS-51 จะมี UART อยู่ในตัวแล้ว ซึ่งเป็นข้อดีของไมโครคอนโทรลเลอร์ ถ้าเป็นไมโครคอนโทรลเลอร์ เช่น เบอร์ Z-80 ถ้าต้องการรับส่งข้อมูลแบบอนุกรมจะต้องนำชิพ UART มาประกอบด้วย

พอร์ทอนุกรมของ MCS-51 จะใช้ขา TXD และ RXD ในการรับส่งข้อมูลโดยขาทั้ง 2 จะอยู่ในพอร์ท 3 คือ P3.1 หรือขา 11 เป็น TXD และ P3.0 หรือ ขา 10 เป็น RXD พอร์ทอนุกรมของ MCS-51 สามารถทำงานแบบ Full Duplex ได้ คือ สามารถส่งและรับข้อมูลในเวลาเดียวกันได้ โดยในการรับและส่งข้อมูลจะมีบัฟเฟอร์สำหรับเก็บข้อมูลให้ใช้ รีจีสเตอร์ที่สำคัญในการรับส่งข้อมูล คือSBUF และ SCON ซึ่งเป็นรีจีสเตอร์ที่อยู่ใน Special Function Register โดยรีจีสเตอร์ Serial Port Buffer ( SBUF ) จะอยู่ในตำแหน่ง 99H ถ้าเขียนข้อมูลไปที่ตำแหน่งนี้ จะเป็นการส่งข้อมูลออกทางพอร์ททางพอร์ทอนุกรม และถ้าอ่านข้อมูลจากตำแหน่งนี้จะเป็นการรับข้อมูลจากพอร์ท โดยใน SBUF จะประกอบด้วยบัฟเฟอร์ 2 ตัว สำหรับส่งและรับข้อมูล ดังรูปด้านล่างนี้




**รูปที่** 2.14 การรับส่งข้อมูลระหว่างรีจีสเตอร์กับบัสภายใน

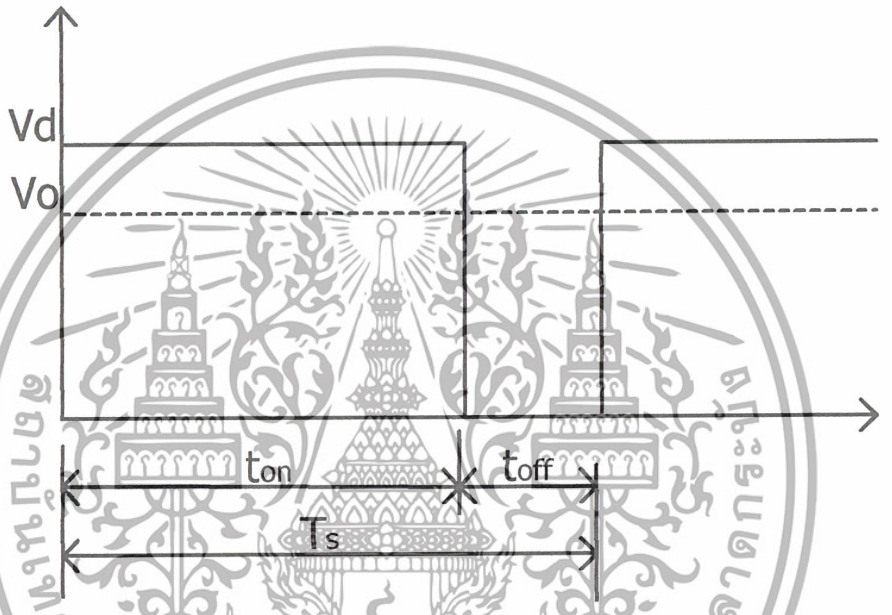
สำหรับ Serial Port Register ( SCON ) ซึ่งอยู่ที่ตำแหน่ง 98H จะเป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ รีจีเตอร์นี้จะทำหน้าที่ควบคุมและบอกสถานะต่าง ๆ ของการรับส่งข้อมูลแบบอนุกรม

สำหรับความเร็วของการส่งข้อมูล ( Baud Rate ) สามารถหาได้จากการหารสัญญาณนาฬิกาที่ใช้กับ MCS-51

# บทที่ 3
# หลักการและทฤษฎีที่ใช้ในการออกแบบ

## 3.1ทฤษฎีที่เกี่ยวข้อง

### หลักการของ Duty Cycle เบื้องต้น




**รูปที่** 3.1 แสดงลักษณะของ Duty Cycle

จากรูปจะได้สมการดังนี้

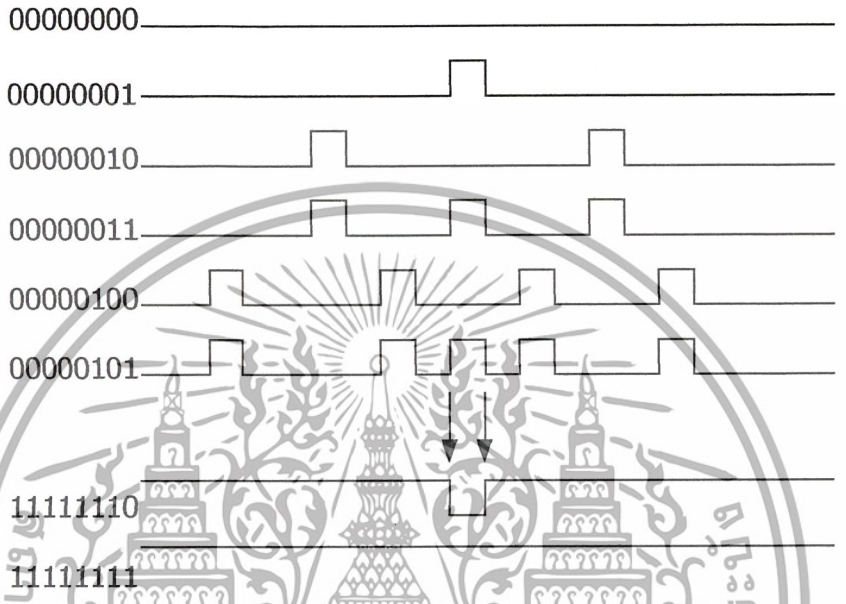
$$Vo = \frac{1}{Ts}\int_{0}^{t_{on}} Vd.dt$$

$$= \frac{Vd.t_{on}}{Ts}$$

$$= Vd.D$$

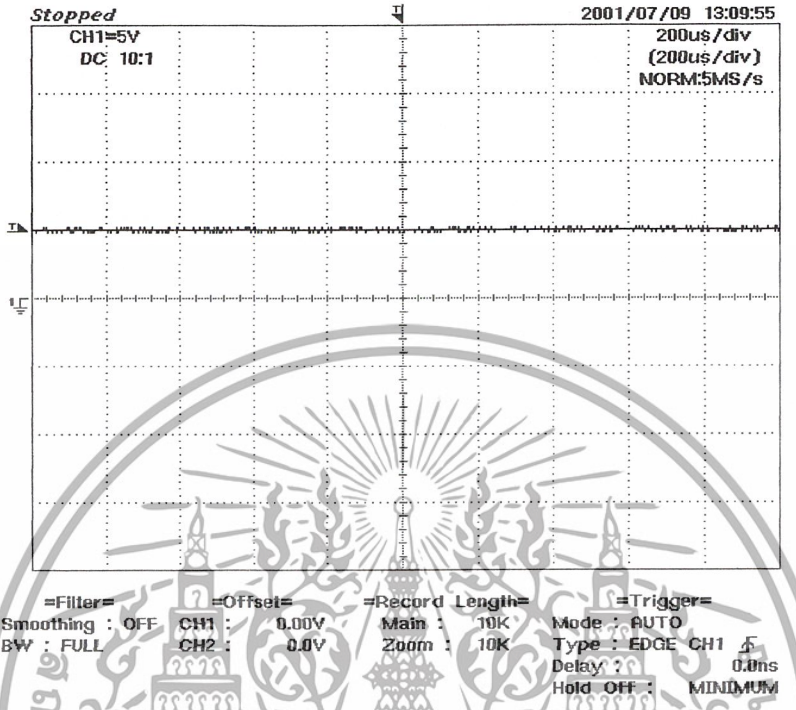
จากสมการเราสามารถควบคุมแรงดันเอาท์พุท ($V_o$) ได้โดยการควบคุมช่วงเวลา ON และ OFF ของสัญญาณ

ลักษณะของสัญญาณควบคุมแบบดิจิตอลซึ่งได้จากไมโครคอนโทรลเลอร์ MCS-51 ที่จะใช้ขับมอเตอร์มีลักษณะดังรูปด้านล่าง

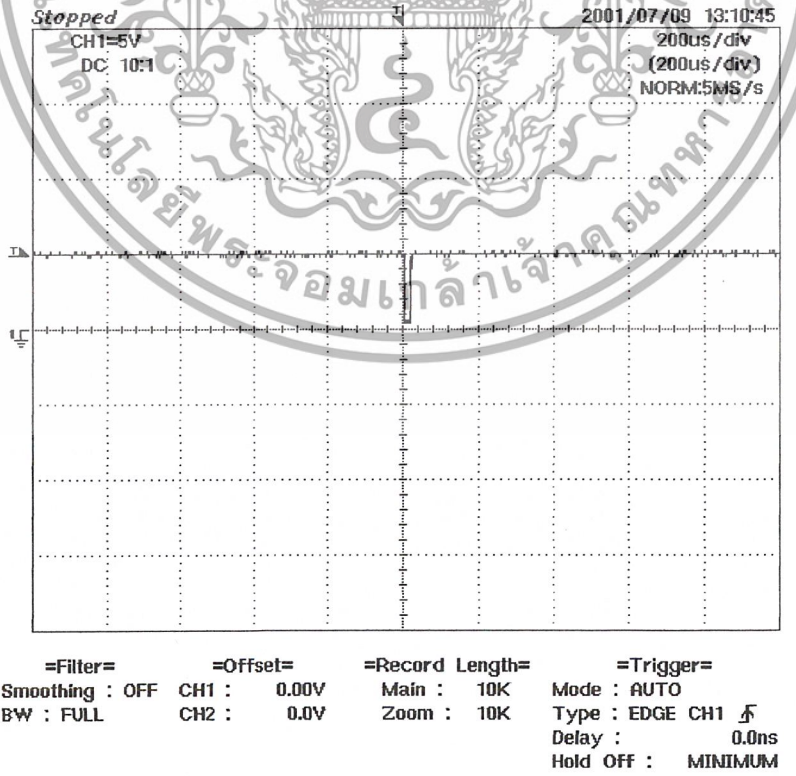



**รูปที่** 3.2 แสดงการควบคุม Duty Cycle โดยลักษณะของสัญญาณดิจิตอล

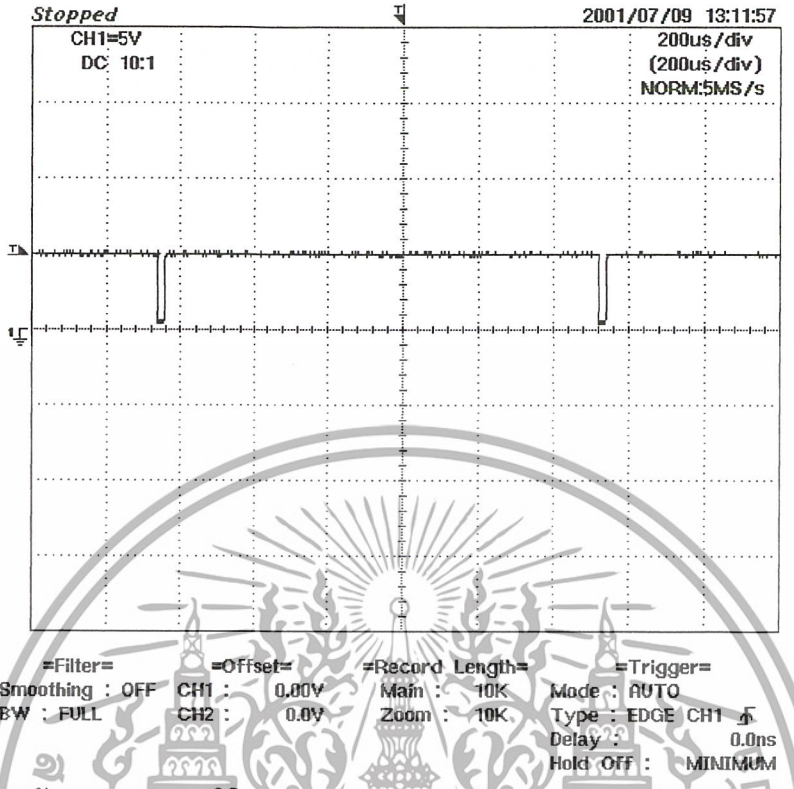
รูปของสัญญาณที่วัดจาก oscilloscope จะมีลักษณะ invert กับสัญญาณมาตรฐานซึ่งมีลักษณะดังรูปด้านล่าง



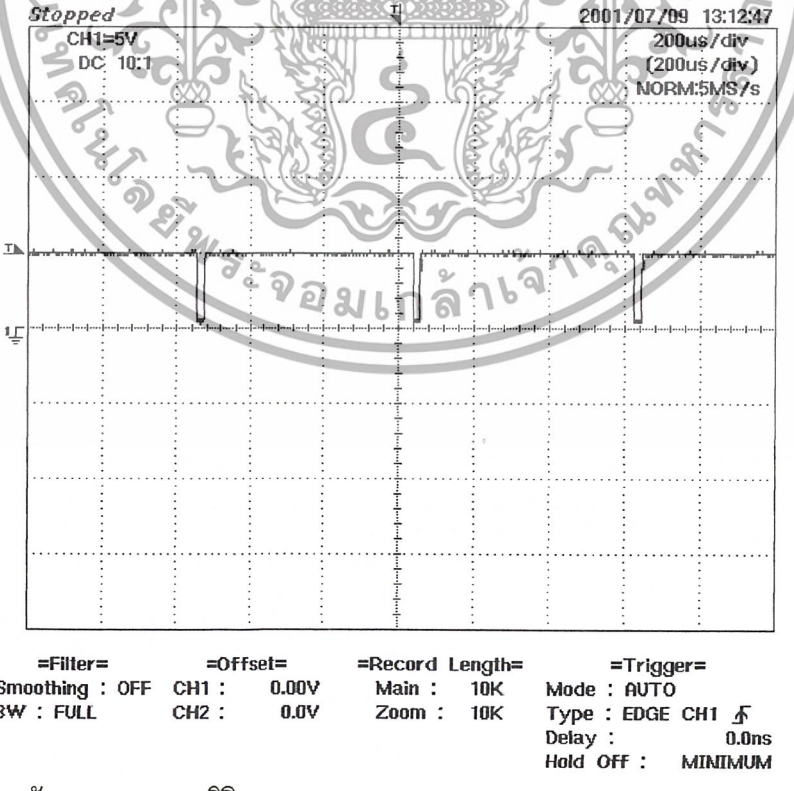

**รูปที่** 3.3 แสดงสัญญาณควบคุมดิจิตอล 00000000



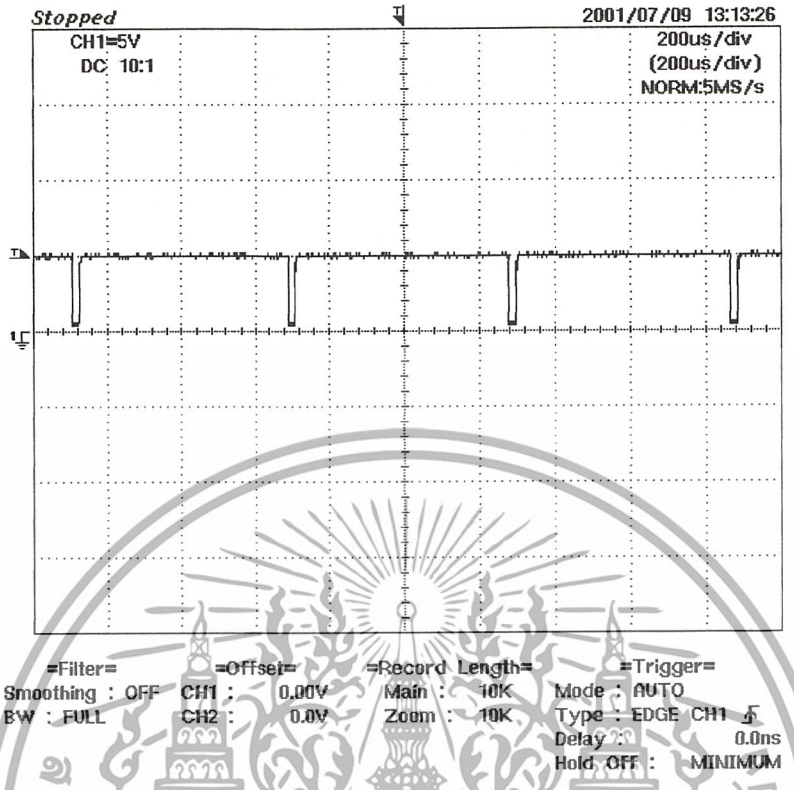

**รูปที่** 3.4 แสดงสัญญาณควบคุมดิจิตอล 00000001



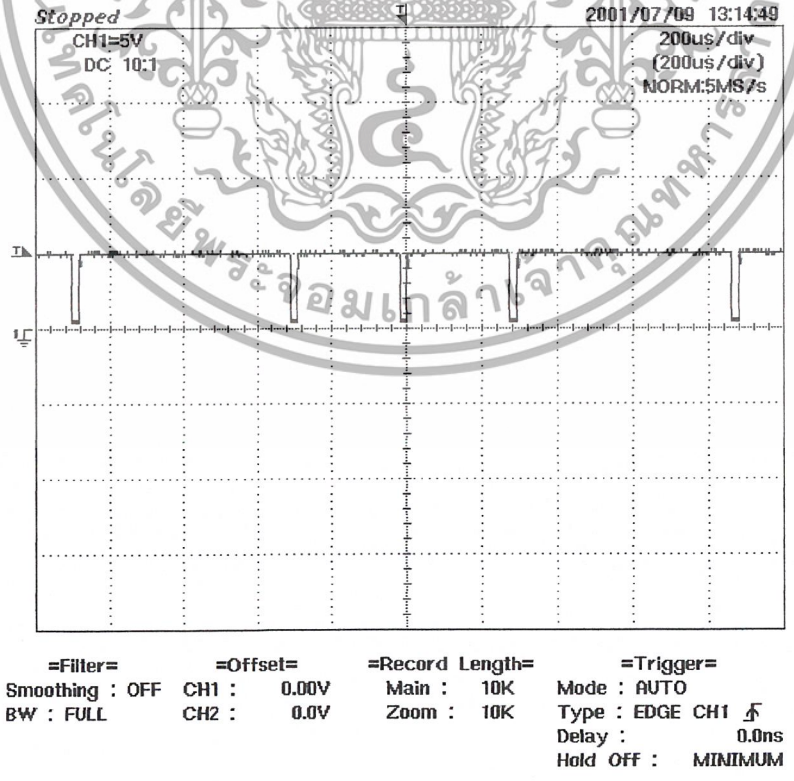

**รูปที่** 3.5 แสดงสัญญาณควบคุมดิจิตอล 00000010



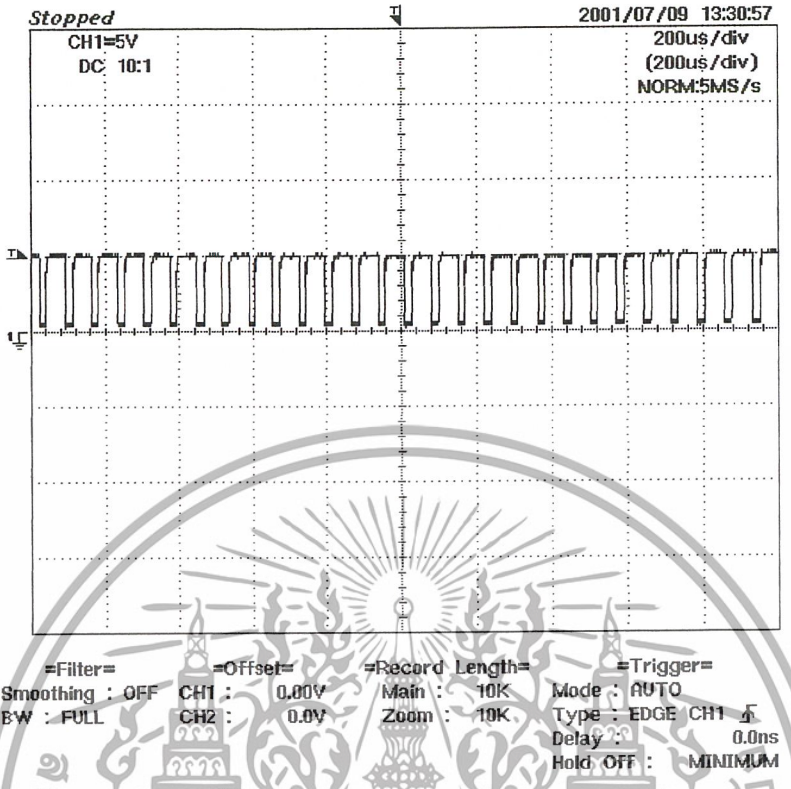

**รูปที่** 3.6 แสดงสัญญาณควบคุมดิจิตอล 00000011



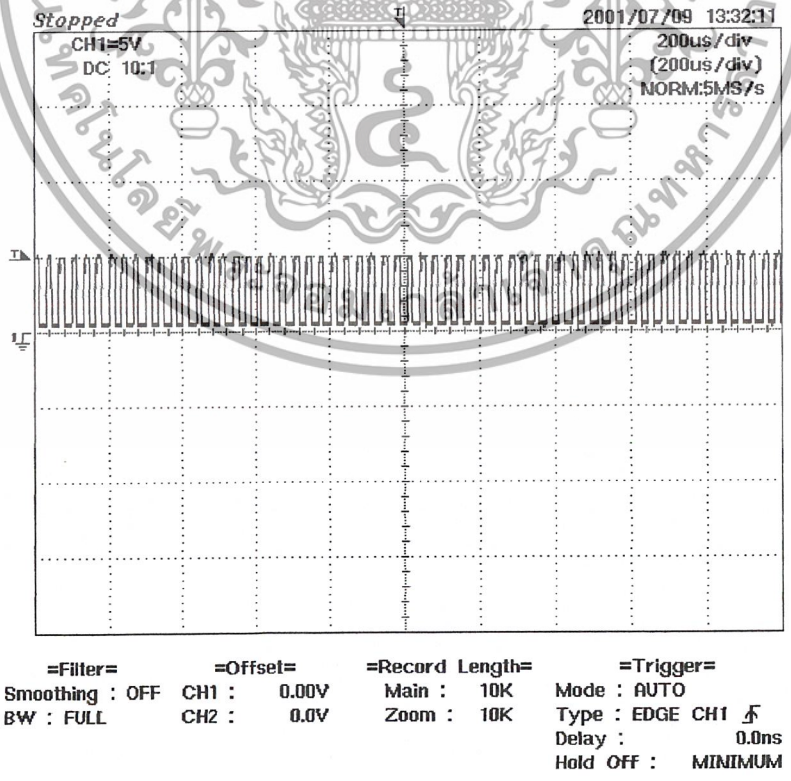

**รูปที่** 3.7 แสดงสัญญาณควบคุมดิจิตอล 00000100



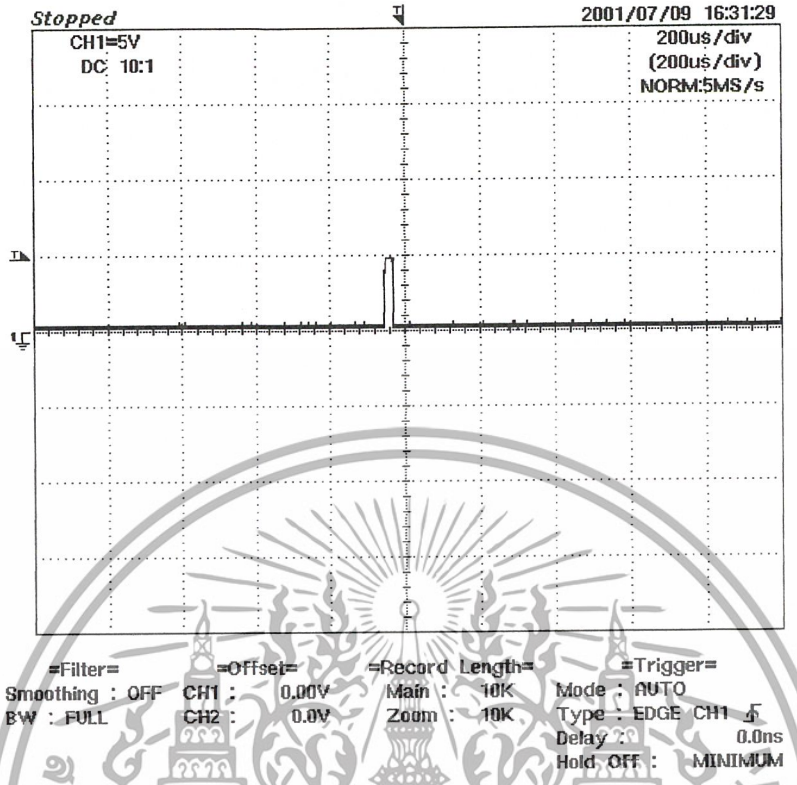

**รูปที่** 3.8 แสดงสัญญาณควบคุมดิจิตอล 000000101



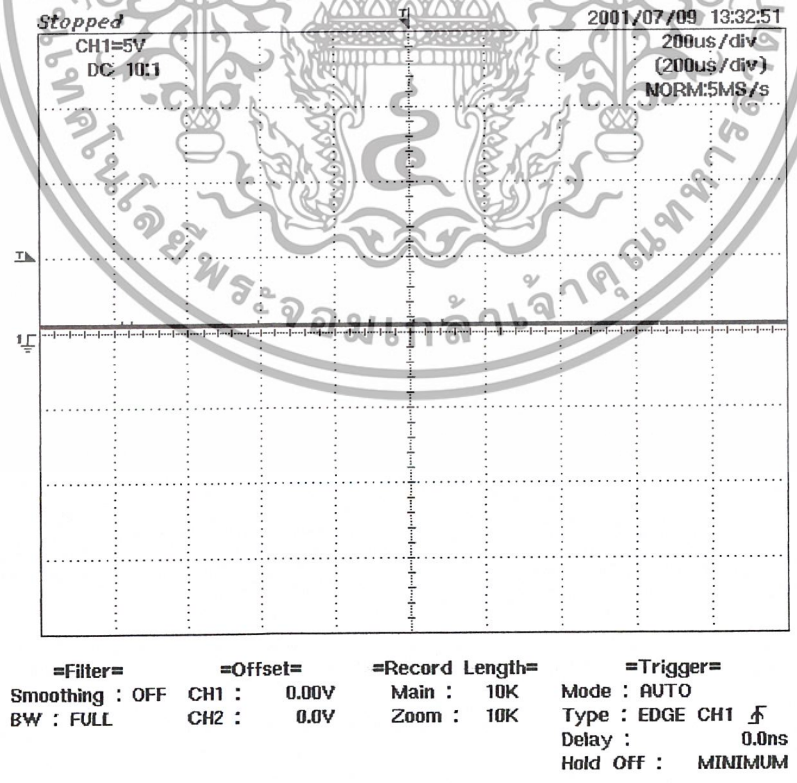

**รูปที่** 3.9 แสดงสัญญาณควบคุมดิจิตอล 01000000




**รูปที่** 3.10 แสดงสัญญาณควบคุมดิจิตอล 10000000




**รูปที่** 3.11 แสดงสัญญาณควบคุมดิจิตอล 11111110

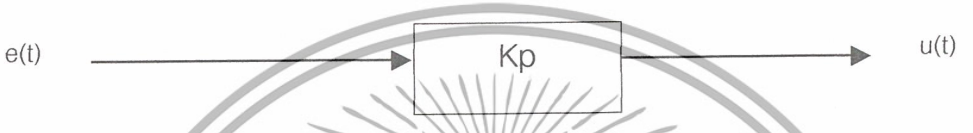



**รูปที่** 3.12 แสดงสัญญาณควบคุมดิจิตอล 11111111

## 3.2 การออกแบบและการสร้างชุดควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

เนื่องจากเราใช้คอมพิวเตอร์เป็นตัวควบคุม ดังนั้นค่าต่างๆ ที่ใช้ในสมการควบคุมต้องทำการแปลงให้อยู่ในรูปที่คอมพิวเตอร์สามารถคำนวณได้โดยใช้พื้นฐานของการควบคุมแต่ละแบบดังนี้

### 3.2.1 การควบคุมเชิงสัดส่วน (Proportional control, P-control)




**รูปที่** 3.13 แสดงการควบคุมเชิงสัดส่วน

จากรูปจะได้ $$u(t) = Kp.e(t)$$

กำหนดให้ Pout = u(t)

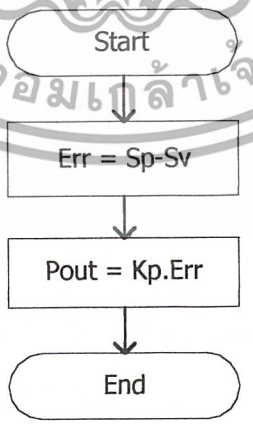
Kp = Kp (อัตราขยายของตัวควบคุมเชิงสัดส่วน)

Err = e(t)

Sp = Set point

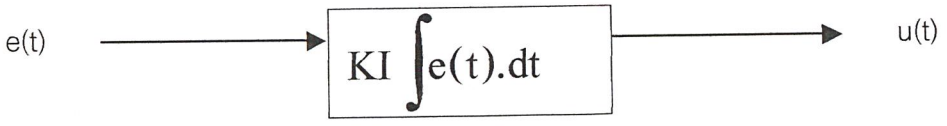
Sv = Sensor Value

ดังนั้น การคำนวณค่าโดยใช้คอมพิวเตอร์ก็สามารถแสดงได้ดัง Flow chart




**รูปที่ 3.14** แสดง Flow Chart การเขียนโปรแกรมการควบคุมเชิงสัดส่วน

### 3.2.2 การควบคุมแบบอินทิกรัล (Integral Control)




**รูปที่** 3.15 แสดงการควบคุมแบบอินทิกรัล

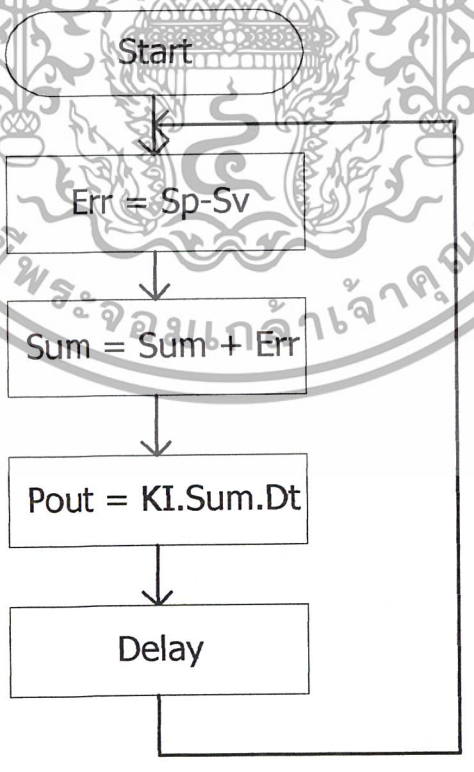
จากรูปจะได้ $u(t) = \int e(t).dt$

กำหนดให้ Sum = $\int e(t)$ (ค่า อินทิกรัลที่ได้จากค่า Err)

Dt = dt (ค่าผลต่างของเวลาในการสุ่มสัญญาณ)

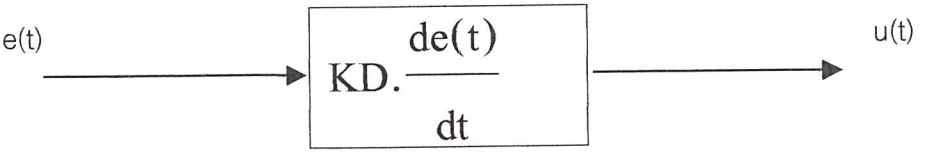
KI = KI(อัตราขยายของการควบคุมแบบอินทิกรัล)

การคำนวณโดยใช้คอมพิวเตอร์สามารถแสดงได้ดัง Flow Chartด้านล่าง




**รูปที่ 3.16** แสดง Flow Chart การเขียนโปรแกรมการควบคุมแบบอินทิกรัล

### 3.2.3 การควบคุมแบบอนุพันธ์ (Derivative control)




**รูปที่** 3.17 แสดงการควบคุมแบบอนุพันธ์

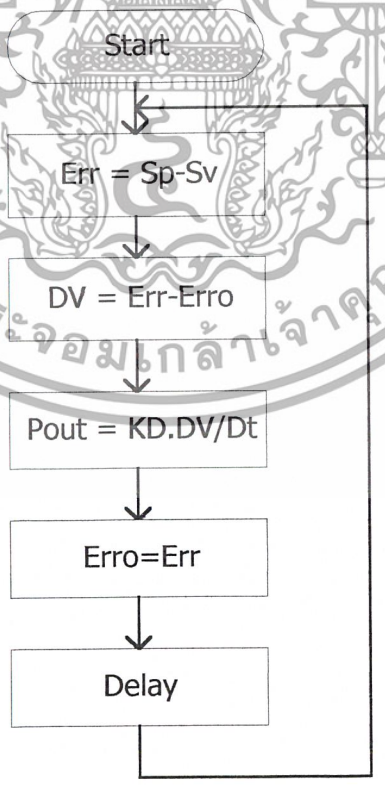
จากรูปจะได้ $u(t) = KD\frac{de(t)}{dt}$

กำหนดให้ Dv = de(t) (ค่าอนุพันธ์ของ Err)

Erro = ค่า Err เก่า

KD = KD (อัตราขยายของตัวควบคุมแบบอนุพันธ์)

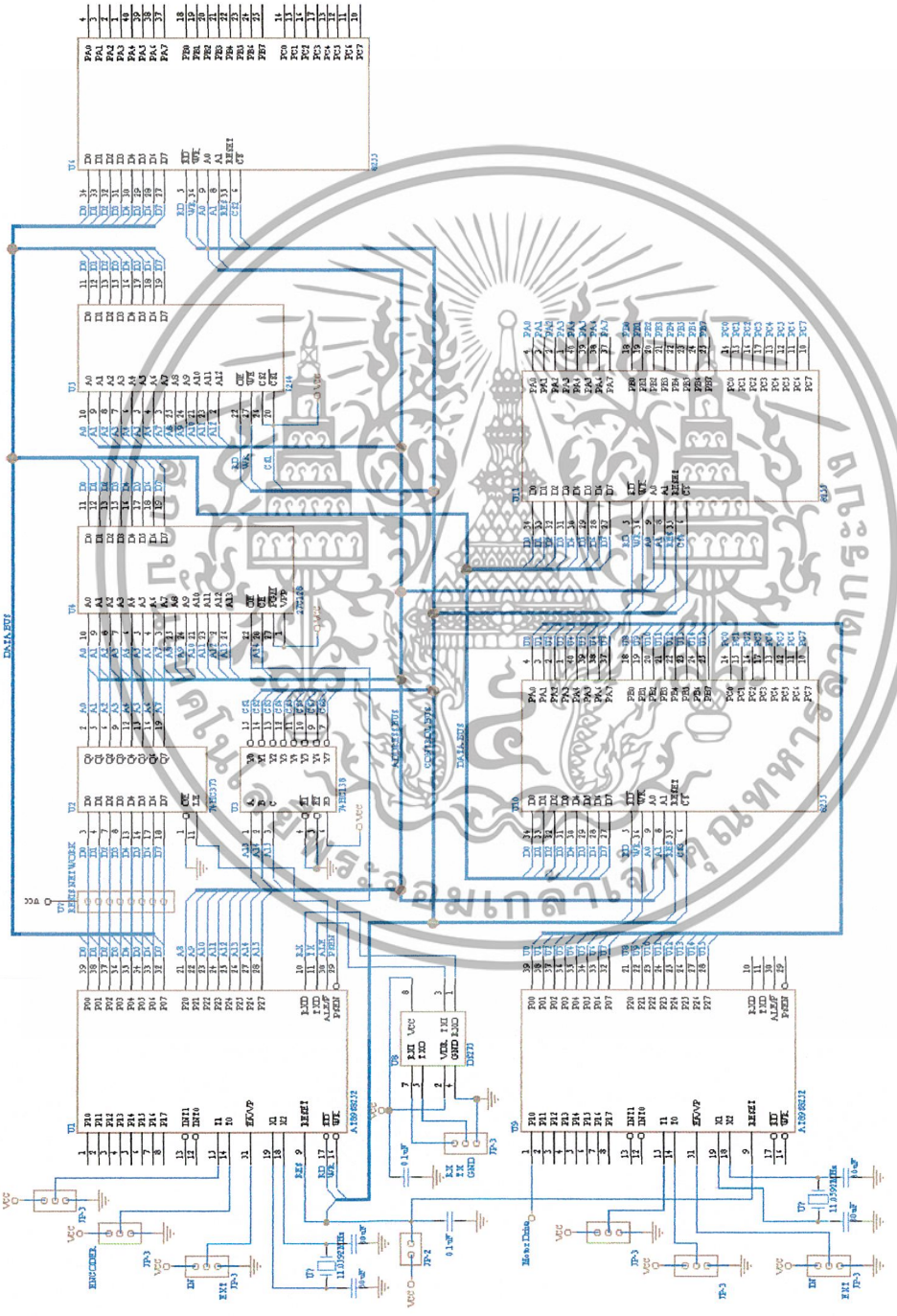
การคำนวณโดยคอมพิวเตอร์สามารถคำนวณได้ดัง Flow Chart ด้านล่าง




**รูปที่** 3.18 แสดง Flow Chart การเขียนโปรแกรมการควบคุมแบบอนุพันธ์

## 3.3 การออกแบบทางฮาร์ดแวร์

### 3.3.1 วงจรชุดควบคุม



**รูปที่ 3.19** แสดงวงจรชุดควบคุม

ในการออกแบบวงจรชุดควบคุมจะใช้ไมโครคอนโทรลเลอร์ 2 ชุด(MCS-51)

**ไมโครคอนโทรลเลอร์ชุดที่ 1**

-ใช้รับส่งข้อมูลระหว่างชุดควบคุมกับคอมพิวเตอร์โดยใช้ IC เบอร์ DS 275

-ใช้นับจำนวนพัลซ์ของ Encoder ผ่านทาง Counter ขา TO(14) ของ MCS -51(Sensor speed)

-ใช้ ON-OFF โหลดโดยใช้ IC เบอร์ 8255 เป็น Interface ระหว่างชุดควบคุมกับชุดโหลด

-ใช้ส่งผ่านค่าการควบคุมความเร็วของมอเตอร์จากคอมพิวเตอร์ไปยังไมโครคอนโทรลเลอร์ชุดที่ 2 โดยใช้ IC เบอร์ 8255 เป็น Interface

**ไมโครคอนโทรลเลอร์ชุดที่ 2**

-ใช้รับค่าควบคุมความเร็วรอบที่ส่งผ่านมาจากไมโครคอนโทรลเลอร์ชุดที่ 1 เพื่อขับมอเตอร์ตามค่าที่ต้องการ

**หน้าที่ IC**

-MCS-51 ใช้เป็นตัวประมวลผลของชุดควบคุม

-74HC373 ใช้เป็นตัว Latch Address

-74HC138 ใช้เป็น Decode ตำแหน่งหน่วยความจำของอุปกรณ์แต่ละตัว

-27128 เป็น IC EPROM ใช้เก็บโปรแกรมที่ใช้ในชุดควบคุม

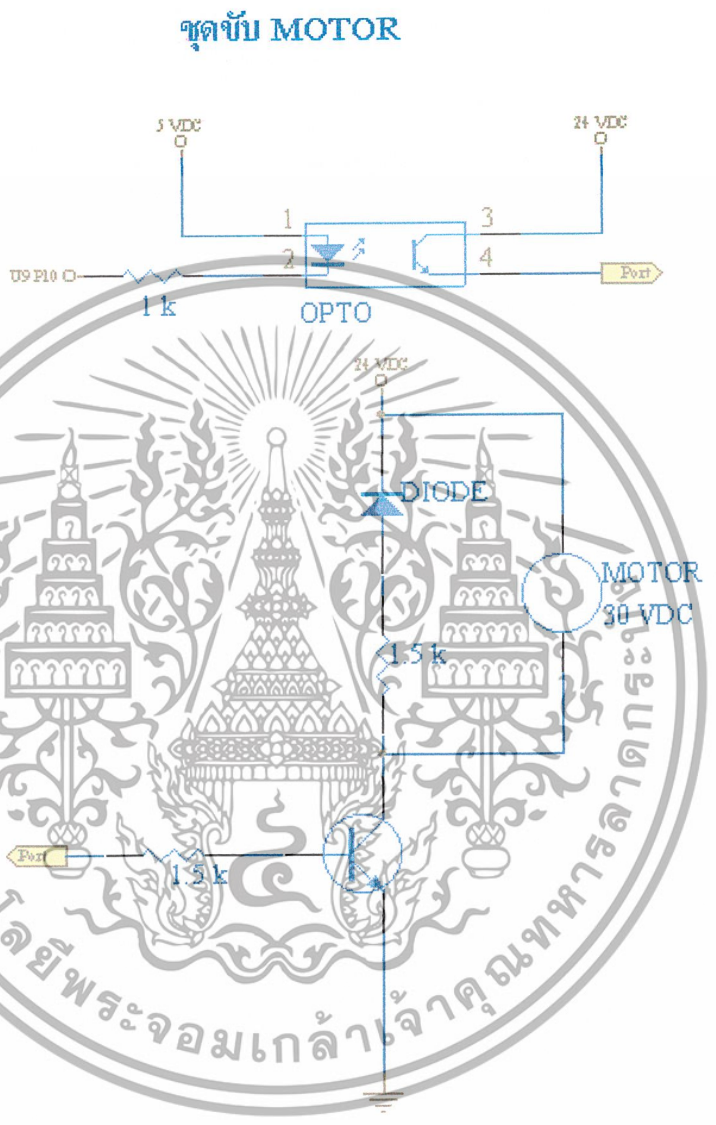
-6264 เป็น RAM ใช้เก็บข้อมูลชั่วคราว

-8255 เป็น I/O ใช้ Interface

-DS 275 เป็น RS 232 drive ใช้เป็นตัวปรับระดับแรงดันสัญญาณให้ได้ตามมาตราฐาน RS 232

### 3.3.2 วงจรชุดขับมอเตอร์

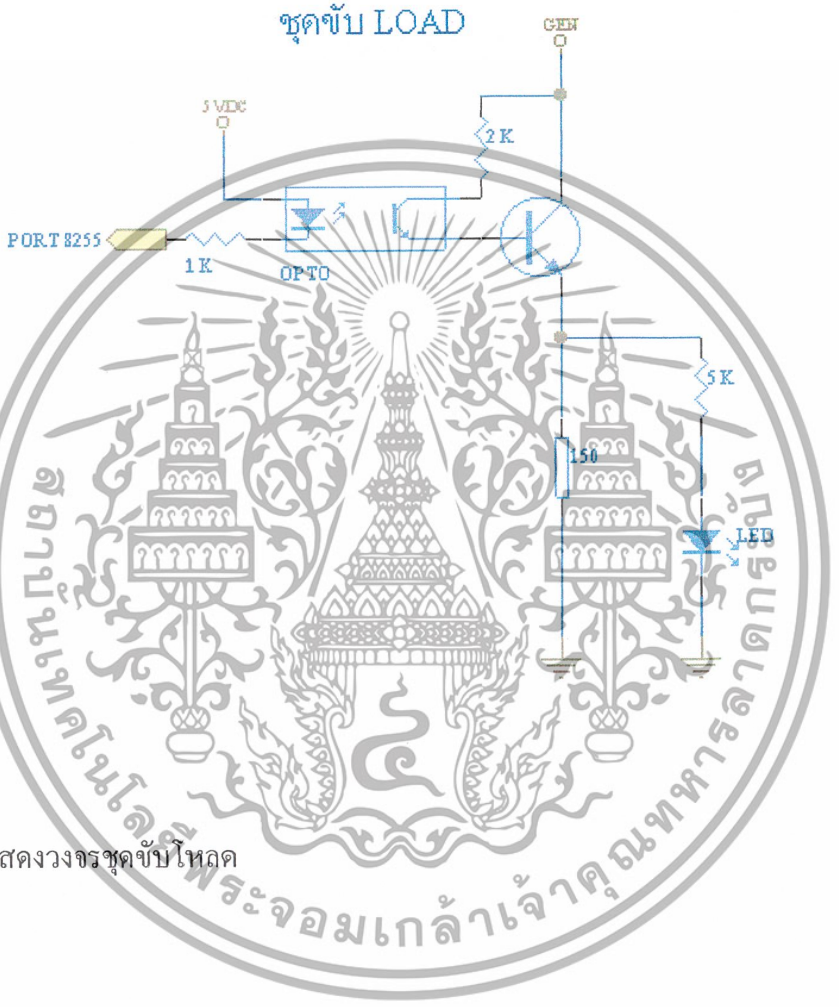
ใช้ทรานซิสเตอร์ เบอร์ 2N3055 เป็นตัวขับมอเตอร์ให้หมุนโดยใช้ IC OPTO เป็น Interface ระหว่างวงจรชุดควบคุมกับวงจรกำลัง




**รูปที่** 3.20 แสดงวงจรชุดขับมอเตอร์

### 3.3.3 วงจรชุดขับโหลด

ใช้ทรานซิสเตอร์เบอร์ 2N6038 เป็นสวิทซ์ชิ่ง ON-OFF โดย OPTO เป็น Interface ระหว่างวงจรควบคุมกับวงจรกำลัง




**รูปที่** 3.21 แสดงวงจรชุดขับโหลด

### 3.4 สรุปแผนผังการทำงานของระบบควบคุม

ในวงจรของการควบคุมความเร็วมอเตอร์มีส่วนประกอบหลักที่สำคัญดังนี้

- คอมพิวเตอร์
- ชุดไมโครคอนโทรลเลอร์ควบคุมและสื่อสารกับคอมพิวเตอร์
- ชุดไมโครคอนโทรลเลอร์ที่ผลิตสัญญาณควบคุม
- มอเตอร์และเครื่องกำเนิดไฟฟ้ากระแสตรง
- ชุดขับโหลดมอเตอร์
- ชุด ON/OFF โหลดและชุดจำลองโหลด

ซึ่งสามารถสรุปเนื้อหาทั้งหกส่วนได้ดังนี้

1. คอมพิวเตอร์ ทำงานโดยการรับค่าจากตัว SENSOR ต่างๆมาจากชุดไมโครคอนโทรลเลอร์ควบคุม และนำเข้าสมการควบคุมแบบต่างๆ แล้วส่งค่าสัญญาณที่ต้องการออกไปให้ชุดไมโครคอนโทรลเลอร์โดยโปรแกรมที่ใช้ จะเขียนด้วย Visual Basic ซึ่งจะมีการแสดงค่าความเร็วรอบของมอเตอร์ การ ON/OFF ของโหลดแต่ละตัว และสามารถสั่งการ ON/OFF ของโหลดผ่าน Soft wear บนคอมพิวเตอร์ได้

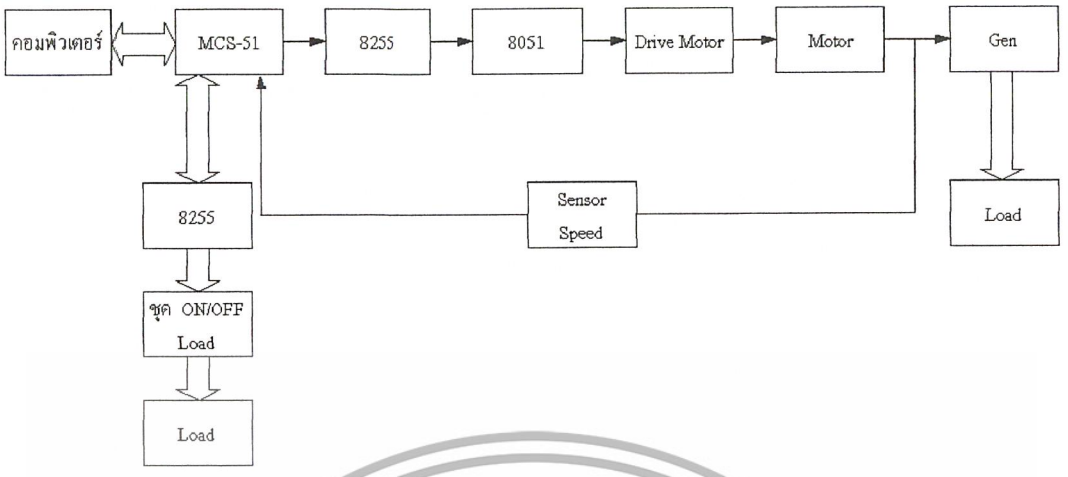
2. ชุดไมโครคอนโทรลเลอร์ควบคุม จะเป็นตัวตรวจจับตรวจจับสัญญาณ ค่าความเร็วรอบ แล้วจะส่งให้กับคอมพิวเตอร์โดยผ่านพอร์ทอนุกรม RS-232 และรับค่า Output จากคอมพิวเตอร์ส่งให้กับชุด ไมโครคอนโทรลเลอร์ผลิตสัญญาณควบคุมผ่าน IC 8255 ซึ่งใช้เป็นตัว ON/OFF โหลดด้วย

3. ชุดไมโครคอนโทรลเลอร์ผลิตสัญญาณควบคุม รับค่าจาก 8255 ที่ต่อกับไมโครคอนโทรลเลอร์ควบคุมมาแปลงเป็นสัญญาณดิจิตอล

4. ชุดขับมอเตอร์จะใช้ทรานซิสเตอร์ ในการขับมอเตอร์ โดยรับสัญญาณดิจิตอลจากชุด ไมโครคอนโทรลเลอร์ผลิตสัญญาณควบคุม

5. มอเตอร์และเครื่องกำเนิดไฟฟ้า โดยใช้ เครื่องกำเนิดไฟฟ้า เป็นโหลดจำลองให้มอเตอร์ซึ่งการต่อเพลาของมอเตอร์เข้ากับเพลาของเครื่องกำเนิดไฟฟ้า

6. ชุด ON/OFF โหลดและชุดจำลองโหลด ใช้ความต้านทาน( R ) เซรามิค 24 ตัว




**รูปที่** 3.22 แสดงแผนผังการทำงานของระบบควบคุมมอเตอร์

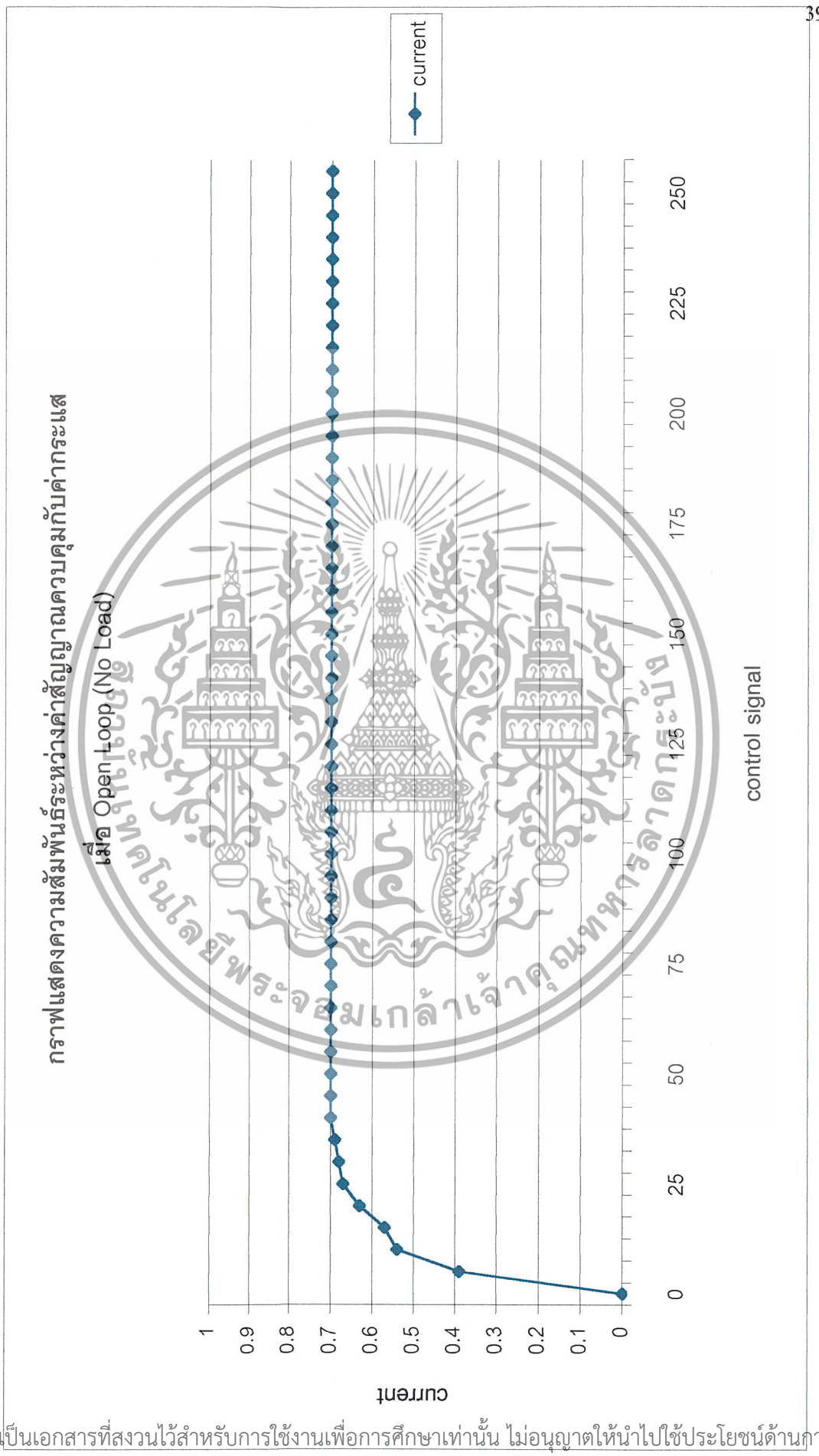
# บทที่ 4
# ผลการทดลองและสรุปผลการทดลอง

ในการทดลองได้กระทำการแบ่งการทดลองออกเป็นสองขั้นตอนคือ ขณะไม่มีภาระโหลด ( NO LOAD ) และขณะมีภาระโหลด( ON LOAD ) โดยในช่วงการทดลองขณะไม่มีภาระโหลด ได้กระทำการป้อนสัญญาณที่ผลิตโดยตัวไมโครคอนโทรลเลอร์ซึ่งสั่งงานมาจากคอมพิวเตอร์ เพื่อทำการทดลองขับมอเตอร์ให้เกิดการหมุน แล้วทำการวัดความเร็วรอบที่เกิดขึ้นว่าแต่ละระดับสัญญาณสามารถจะขับมอเตอร์ให้มีความเร็วรอบออกมาได้เท่าไหร่ โดยใช้ตัวเอนโคดเดอร์ที่ติดอยู่กับตัวมอเตอร์เป็นตัวตัวตรวจจับและได้วัดด้วยเทคโคมิเตอร์ด้วยเพื่อเปรียบเทียบความแตกต่างที่เกิดขึ้นแล้วบันทึกผลการทดลอง รวมทั้งกระทำการวัดค่ากระแสและค่าแรงดันของมอเตอร์ที่ระดับของสัญญาณที่ต่างกันด้วย โดยการทดลองดังกล่าวไม่ได้กระทำการให้มอเตอร์ต้องรับภาระโหลด

ส่วนการทดลองในช่วงที่สองจะเป็นการทดลองควบคุมความเร็วของมอเตอร์เมื่อต้องรับภาระโหลดที่ค่าต่างๆกัน โดยการสั่งการกำหนดโหลดจากคอมพิวเตอร์เช่นเดียวกัน การทดลองกระทำโดยการป้อนค่าความเร็วรอบที่ต้องการผ่านทางหน้าต่างควบคุมบนจอคอมพิวเตอร์ เพื่อให้มอเตอร์มีความเร็วรอบเป็นไปตามที่ต้องการหลังจากนั้นก็สั่งการกำหนดโหลดทีละชุดเพื่อบันทึกการเปลี่ยนแปลง จากการกระทำดังกล่าวเมื่อมอเตอร์ต้องรับภาระโหลดจะพบว่าความเร็วรอบของมอเตอร์ลดลงไม่เป็นไปตามค่าความเร็วรอบที่เราต้องการ แต่ในชั่วขณะหนึ่งเท่านั้นเพราะว่าตัวตรวจจับหรือเอนโคดเดอร์จะตรวจจับค่าที่เกิดการผิดพลาดขึ้นส่งไปยังคอมพิวเตอร์เพื่อให้มีการสั่งการผลิตสัญญาณควบคุมเพื่อป้อนให้แก่ชุดขับมอเตอร์ ให้ทำการรักษาค่าความเร็วรอบที่ต้องการไม่ให้เกิดการเปลี่ยนแปลง แม้ว่าตัวมอเตอร์จะต้องรับภาระโหลดเพิ่มเข้ามาอีกก็ตาม แล้วทำการบันทึกผลของค่ากระแสและค่าแรงดันที่มีการเปลี่ยนแปลงอันเกิดจากการที่ได้มีการชดเชยค่าของสัญญาณที่เกิดขึ้นเพื่อรักษาค่าความเร็วให้เป็นไปตามที่ต้องการ

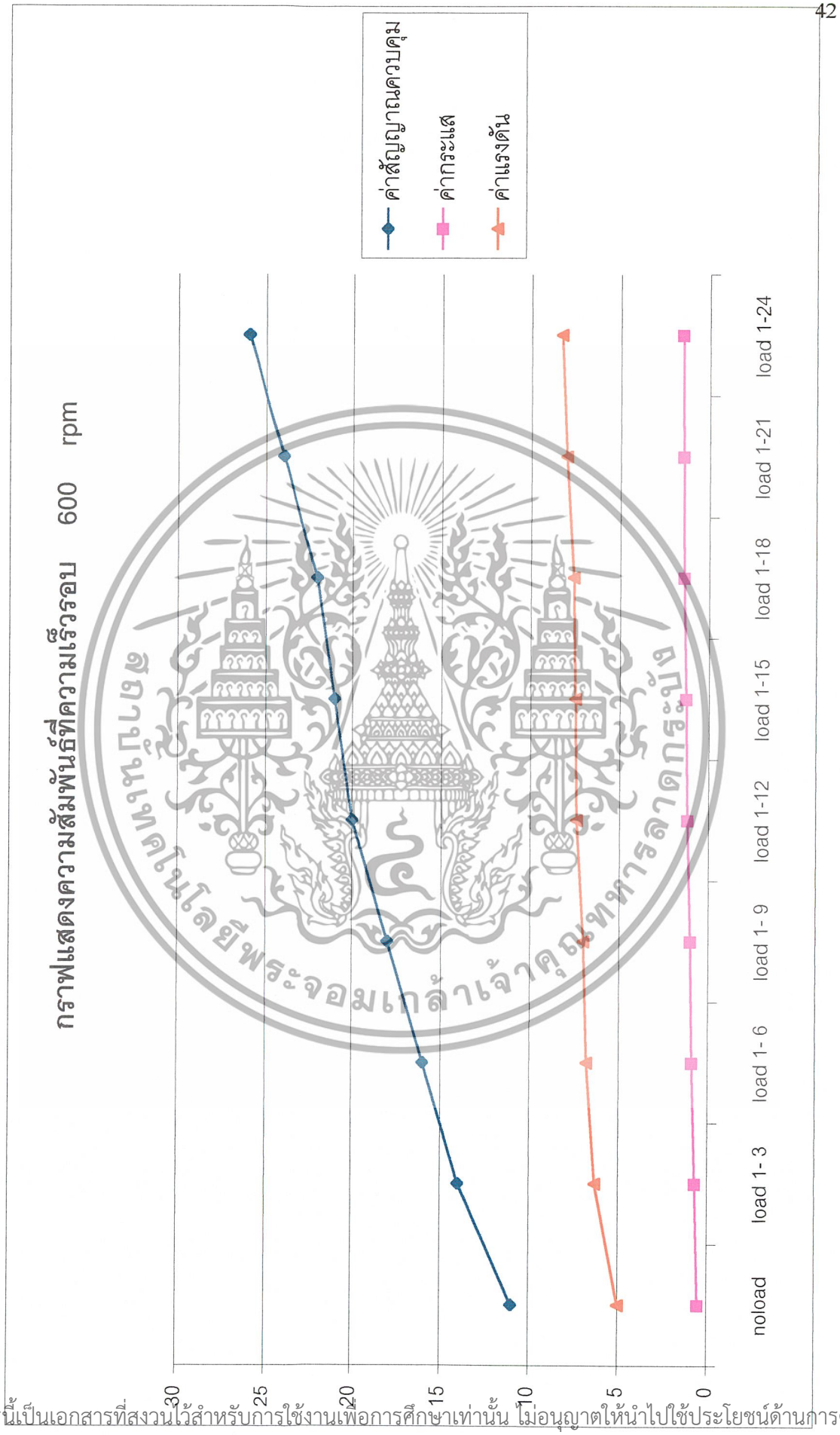
## 4.1 ผลการทดลองเมื่อ OPEN LOOP

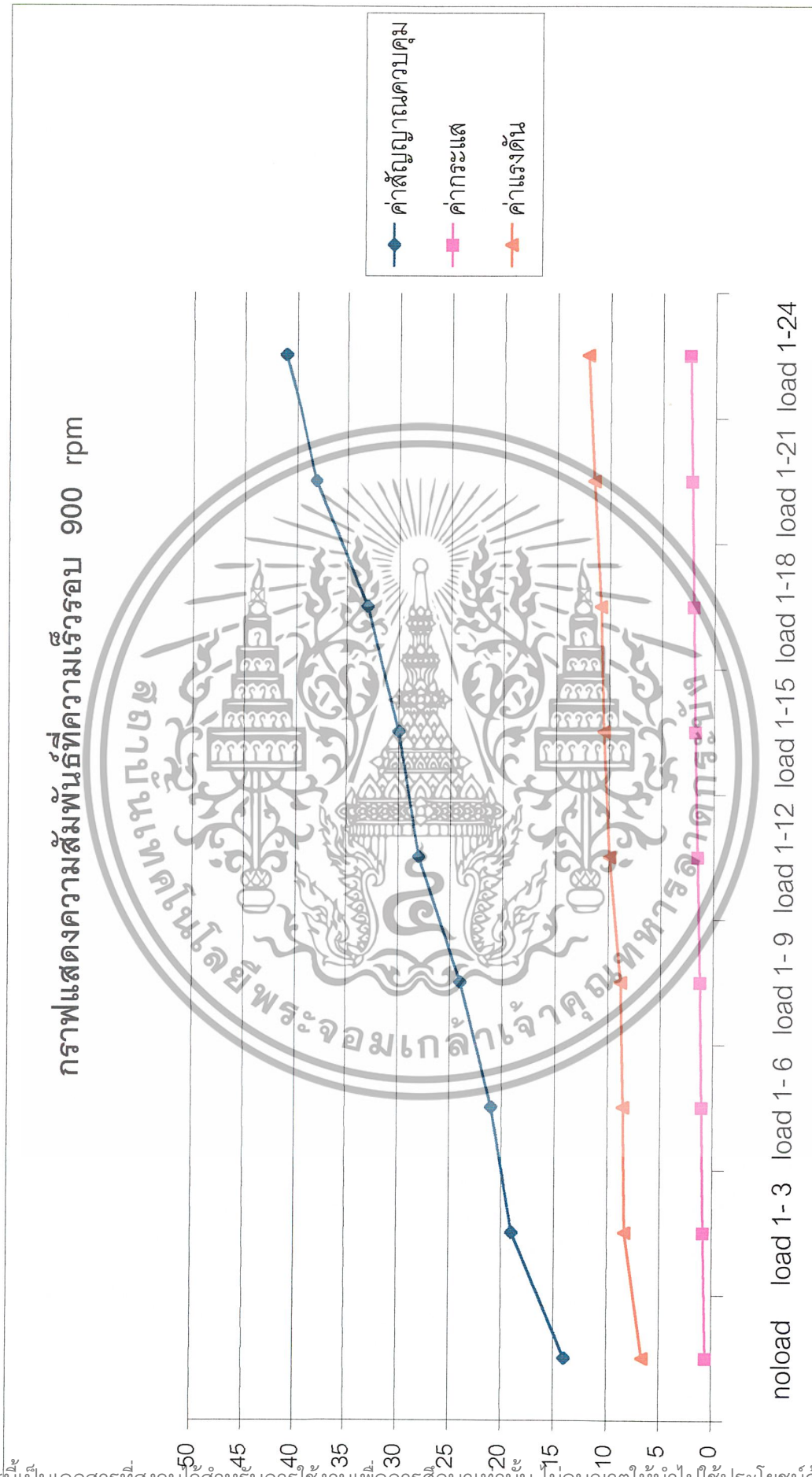
| DIGITAL SIGNAL (DEC) | SPEED (RPM) | | CURRENT (A) | VOLTAGE (V) |
|---|---|---|---|---|
| | Display | Tacometrer | | |
| 5 | 0 | 0 | 0.39 | 1.0 |
| 10 | 578 | 545 | 0.54 | 4.8 |
| 15 | 892 | 978 | 0.57 | 7.7 |
| 20 | 1744 | 1645 | 0.63 | 12 |
| 25 | 2052 | 2055 | 0.67 | 14.8 |
| 30 | 2074 | 2222 | 0.68 | 15.8 |
| 35 | 2340 | 2300 | 0.69 | 16.3 |
| 40 | 2350 | 2370 | 0.70 | 16.8 |
| 45 | 2350 | 2370 | 0.70 | 16.8 |
| 50 | 2350 | 2370 | 0.70 | 16.8 |
| 55 | 2350 | 2382 | 0.70 | 16.9 |
| 60 | 2350 | 2382 | 0.70 | 16.9 |
| 65 | 2353 | 2403 | 0.70 | 17.0 |
| 70 | 2353 | 2410 | 0.70 | 17.0 |
| 75 | 2354 | 2412 | 0.70 | 17.0 |
| " | " | " | " | " |
| " | " | " | " | " |
| 250 | 2357 | 2415 | 0.70 | 17.0 |
| 255 | 2360 | 2420 | 0.70 | 17.0 |

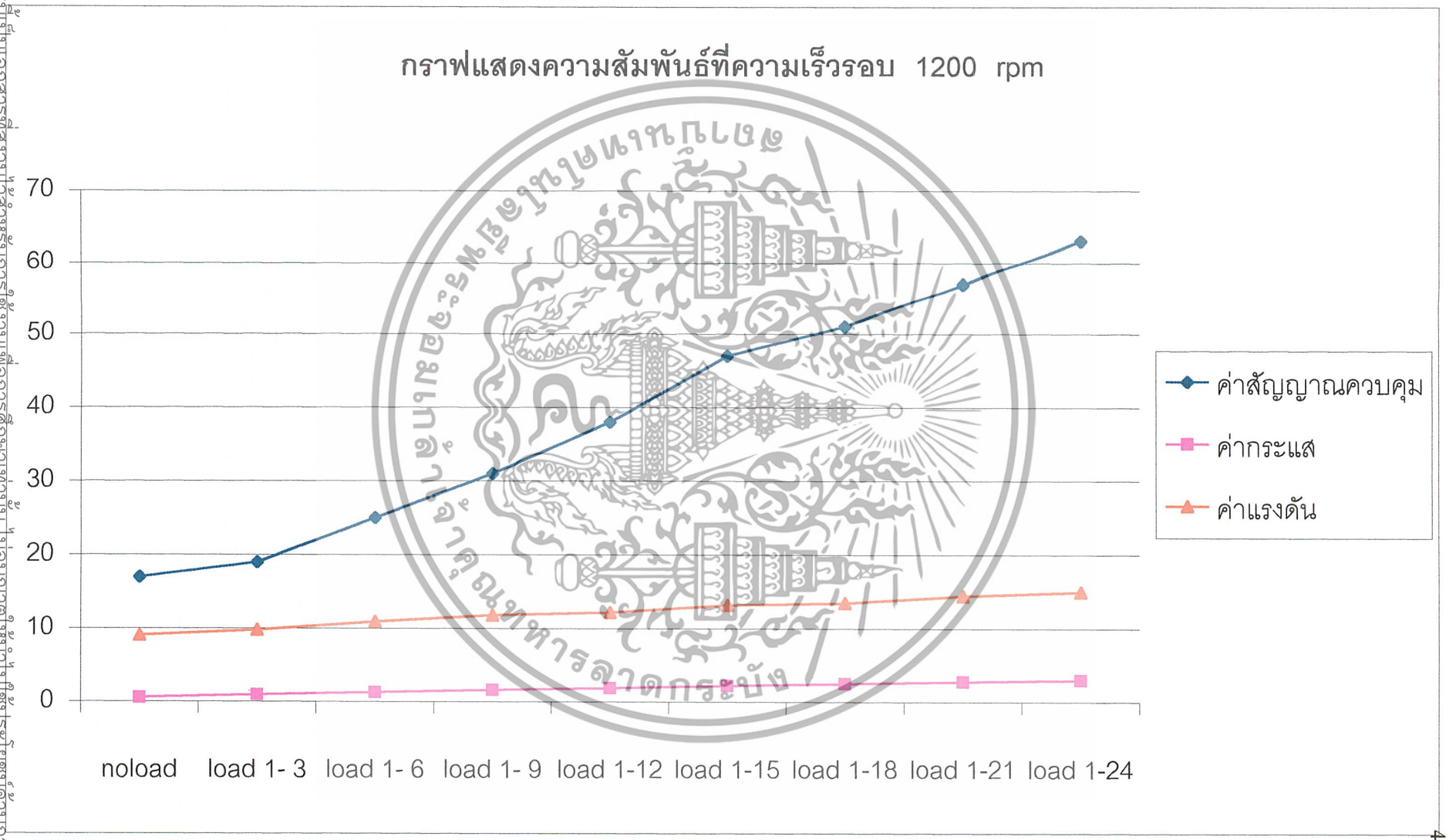


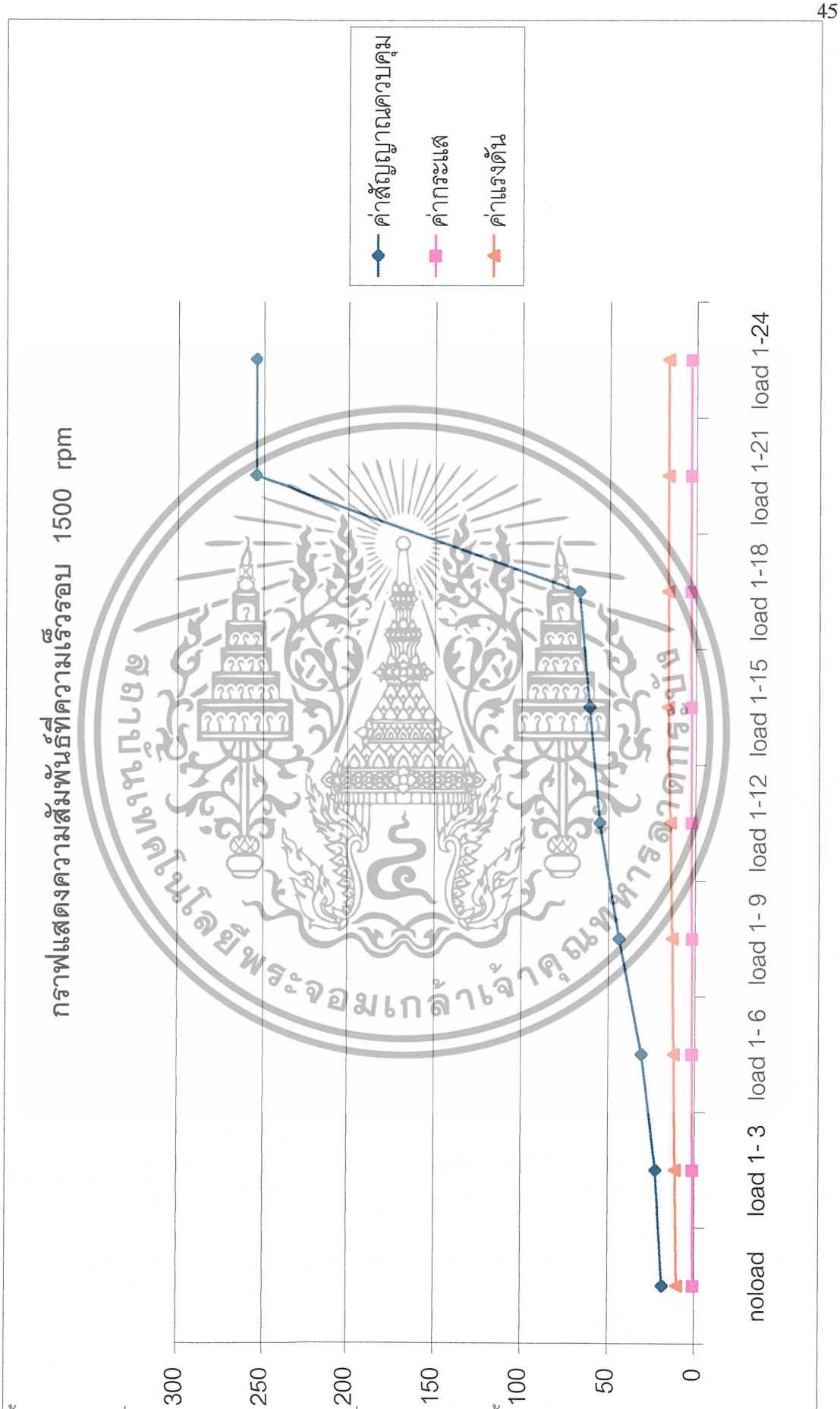
## 4.2 ผลการทดลองเมื่อ CLOSE LOOP

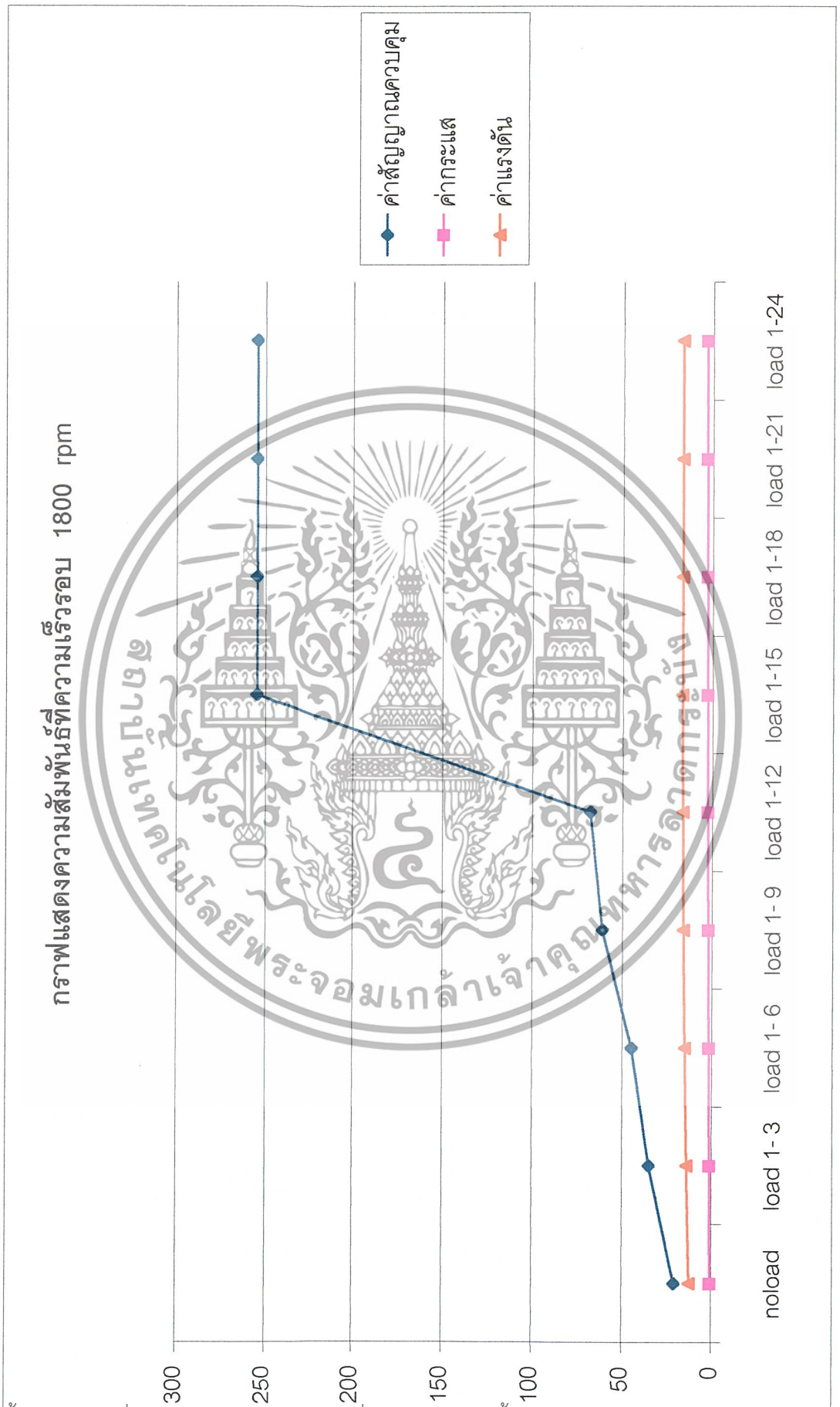
| SET SPEED (RPM) | NO LOAD | | | | | ON LOAD | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Digital Signal | SPEED Display | SPEED Taco | I (A) | V (V) | LOAD 1-3 | | | LOAD 1-6 | | | LOAD 1-9 | | | LOAD 1-12 | | | LOAD 1-15 | | | LOAD 1-18 | | | LOAD 1-21 | | | LOAD 1-24 | | |
| | | | | | | Digital Signal | I (A) | V (V) | Digital Signal | I (A) | V (V) | Digital Signal | I (A) | V (V) | Digital Signal | I (A) | V (V) | Digital Signal | I (A) | V (V) | Digital Signal | I (A) | V (V) | Digital Signal | I (A) | V (V) | Digital Signal | I (A) | V (V) |
| 600 | 11 | 586 | 584 | 0.52 | 5.0 | 14 | 0.70 | 6.3 | 16 | 0.89 | 6.8 | 18 | 1.02 | 7.0 | 20 | 1.20 | 7.4 | 20 | 1.29 | 7.5 | 22 | 1.42 | 7.6 | 24 | 1.46 | 8.0 | 26 | 1.51 | 8.3 |
| 900 | 14 | 893 | 877 | 0.55 | 6.6 | 19 | 0.85 | 8.3 | 21 | 1.04 | 8.5 | 24 | 1.25 | 8.8 | 28 | 1.52 | 9.9 | 30 | 1.78 | 10.5 | 33 | 1.98 | 10.8 | 38 | 2.18 | 11.5 | 41 | 2.37 | 12.1 |
| 1200 | 17 | 1180 | 1208 | 0.58 | 9.1 | 19 | 0.94 | 9.8 | 25 | 1.28 | 10.9 | 31 | 1.59 | 11.8 | 38 | 1.90 | 12.2 | 47 | 2.22 | 13.2 | 51 | 2.49 | 13.5 | 57 | 2.79 | 14.4 | 63 | 2.98 | 15.0 |
| 1500 | 19 | 1465 | 1434 | 0.61 | 11 | 23 | 1.06 | 11.6 | 31 | 1.59 | 12.5 | 44 | 1.87 | 13.3 | 55 | 2.26 | 14.9 | 61 | 2.64 | 16.1 | 67 | 2.94 | 16.5 | 255 | 3.06 | 16.4 | 255 | 3.08 | 16.5 |
| 1800 | 21 | 1750 | 1757 | 0.68 | 12 | 35 | 1.18 | 13.8 | 45 | 1.69 | 15.1 | 61 | 2.12 | 15.9 | 68 | 2.55 | 16.5 | 255 | 2.78 | 16.8 | 255 | 2.96 | 16.6 | 255 | 3.08 | 16.6 | 255 | 3.09 | 16.6 |
| 2100 | 31 | 2078 | 2250 | 0.68 | 16.0 | 58 | 1.36 | 17.3 | 255 | 1.86 | 17.1 | 255 | 2.26 | 17.2 | 255 | 2.59 | 17.8 | 255 | 2.80 | 17.1 | 255 | 2.96 | 16.8 | 255 | 3.09 | 16.6 | 255 | 3.10 | 16.7 |

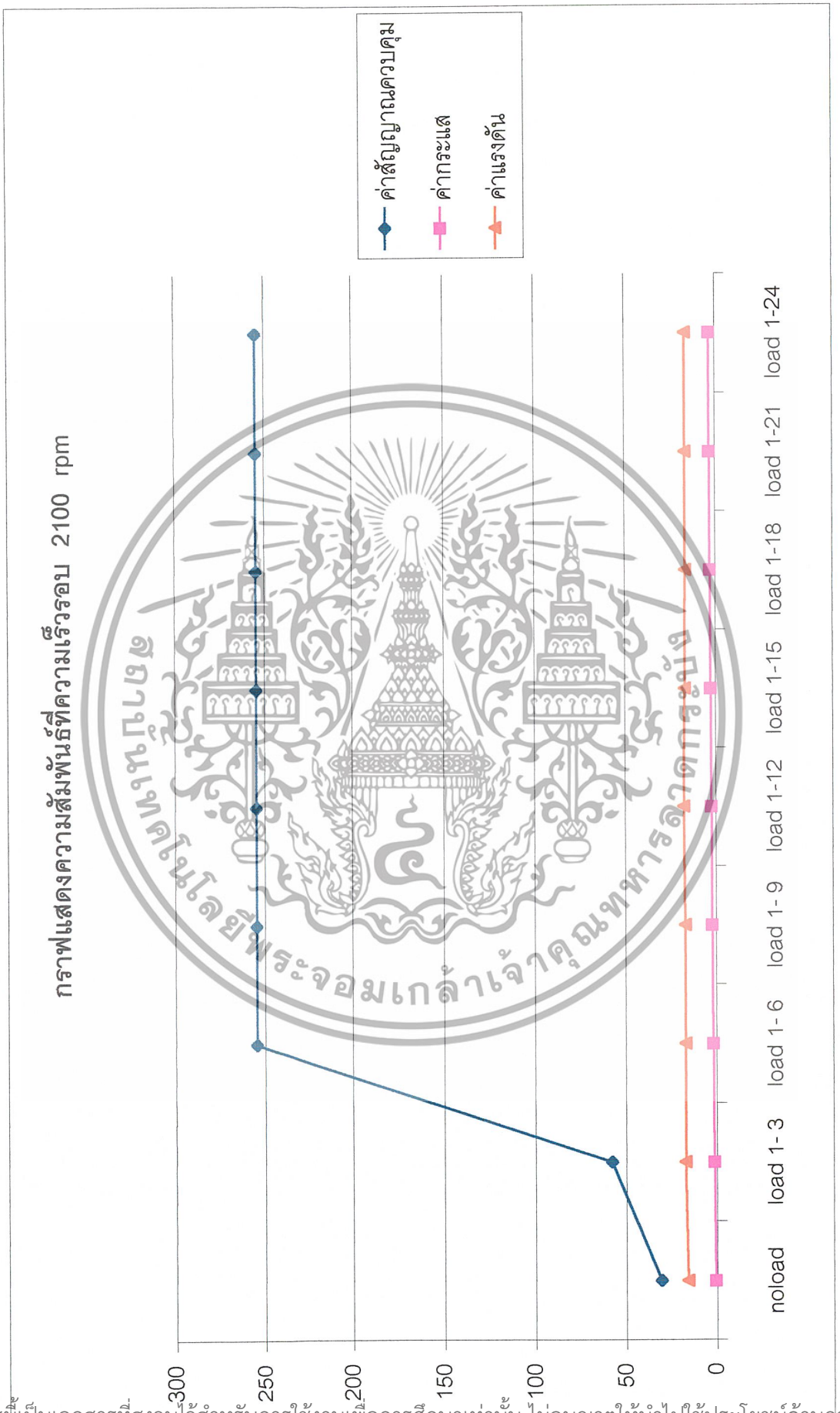


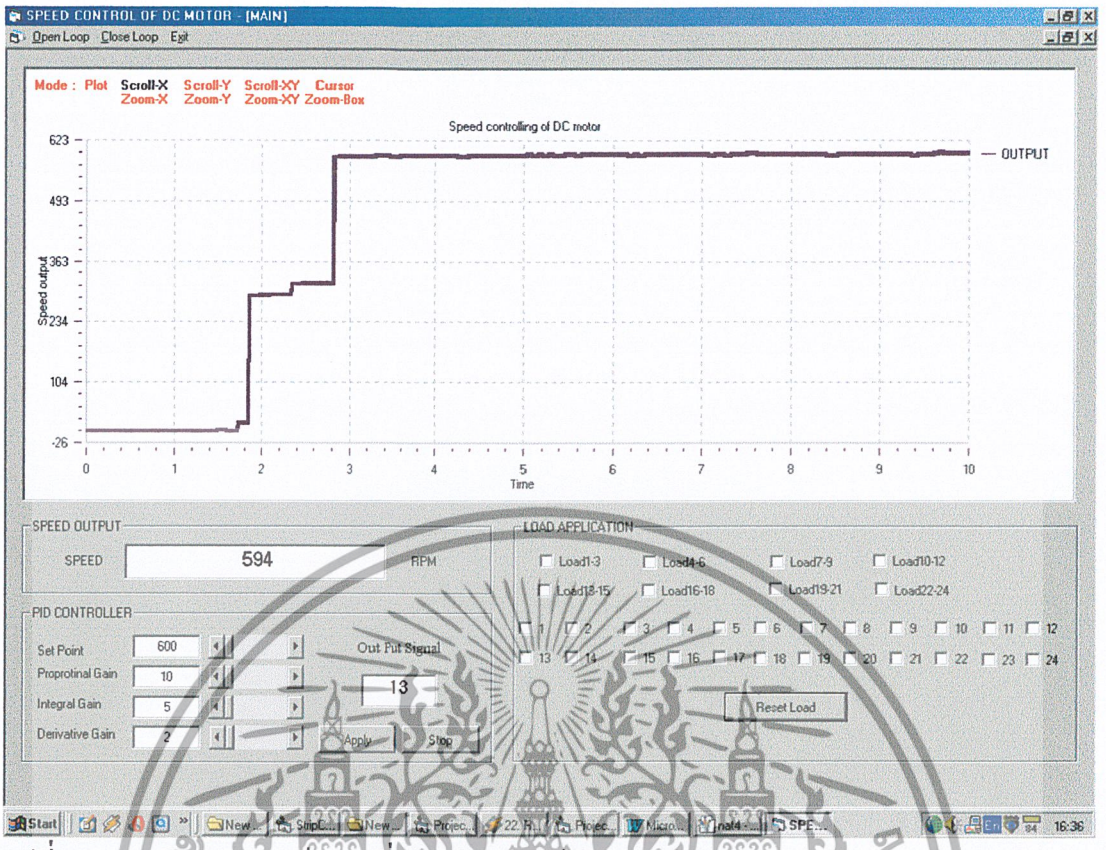



กราฟแสดงความสัมพันธ์ที่ความเร็วรอบ 1200 rpm
70
60
50
40
30
20
10
0
noload
load 1- 3
load 1- 6
load 1- 9
load 1-12
load 1-15
load 1-18
load 1-21
load 1-24
ค่าสัญญาณควบคุม
ค่ากระแส
ค่าแรงดัน

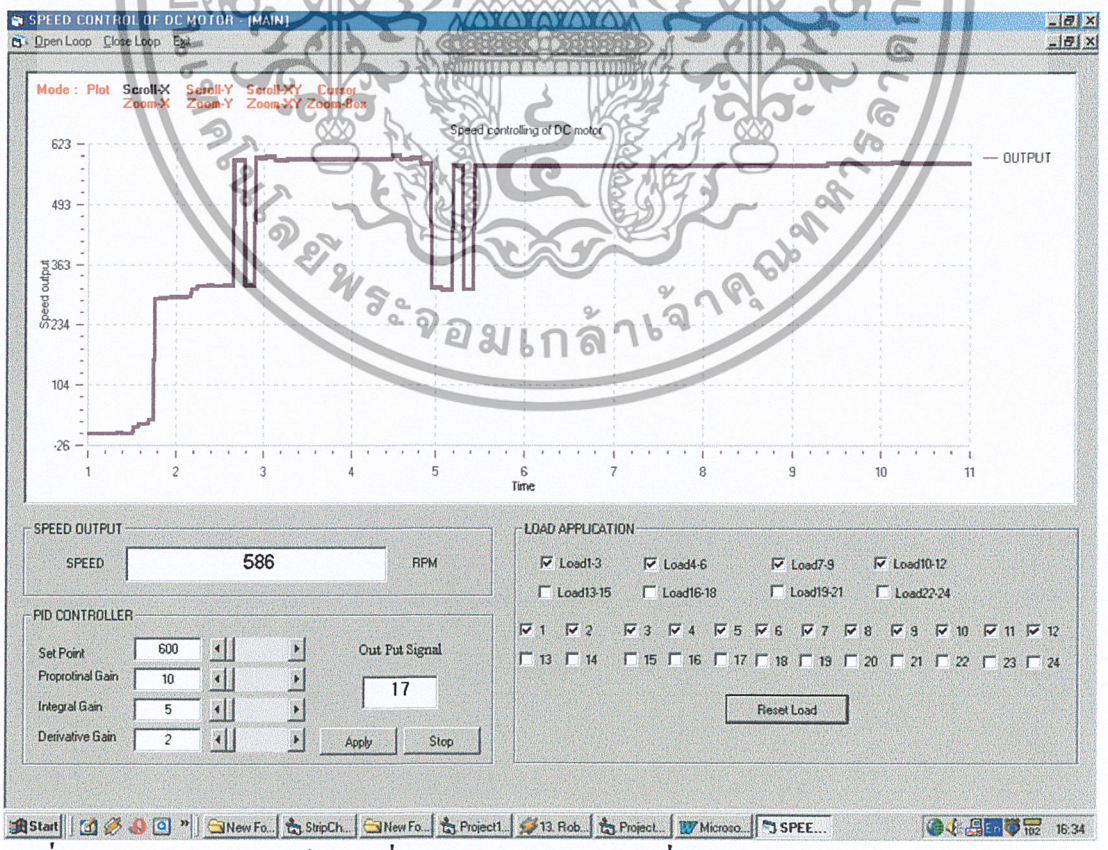




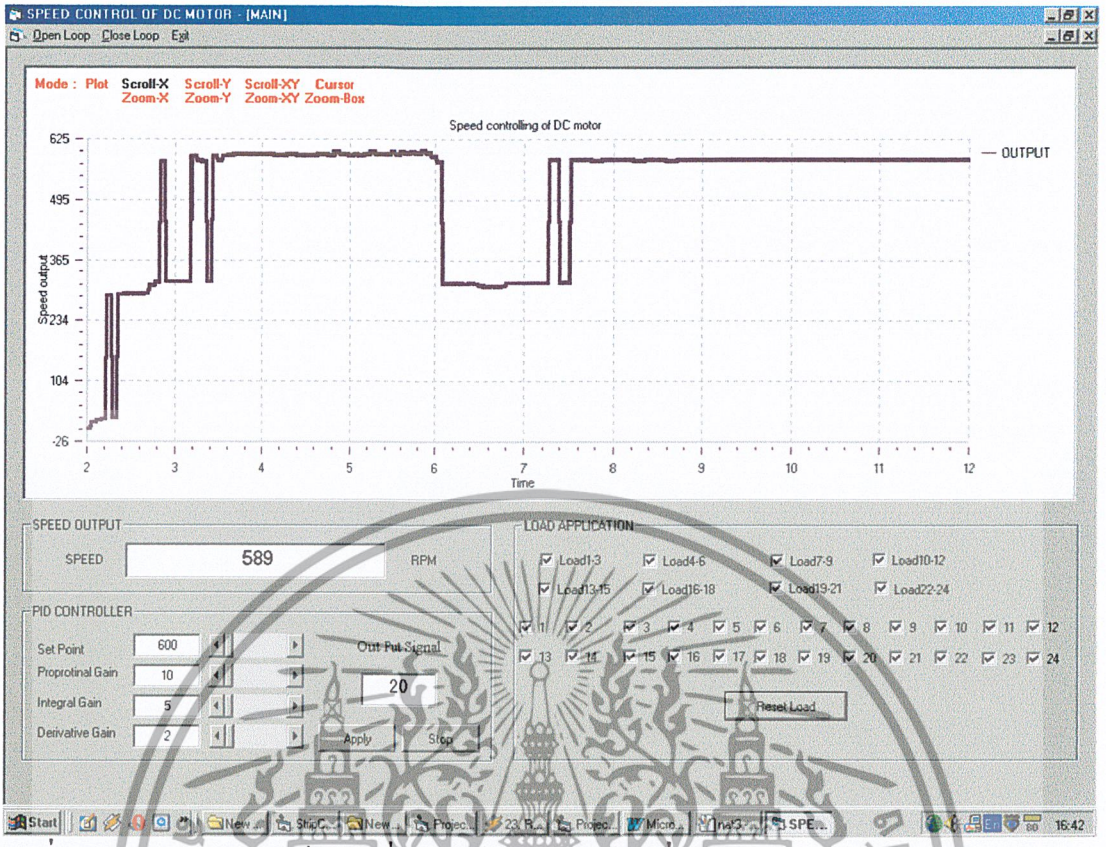




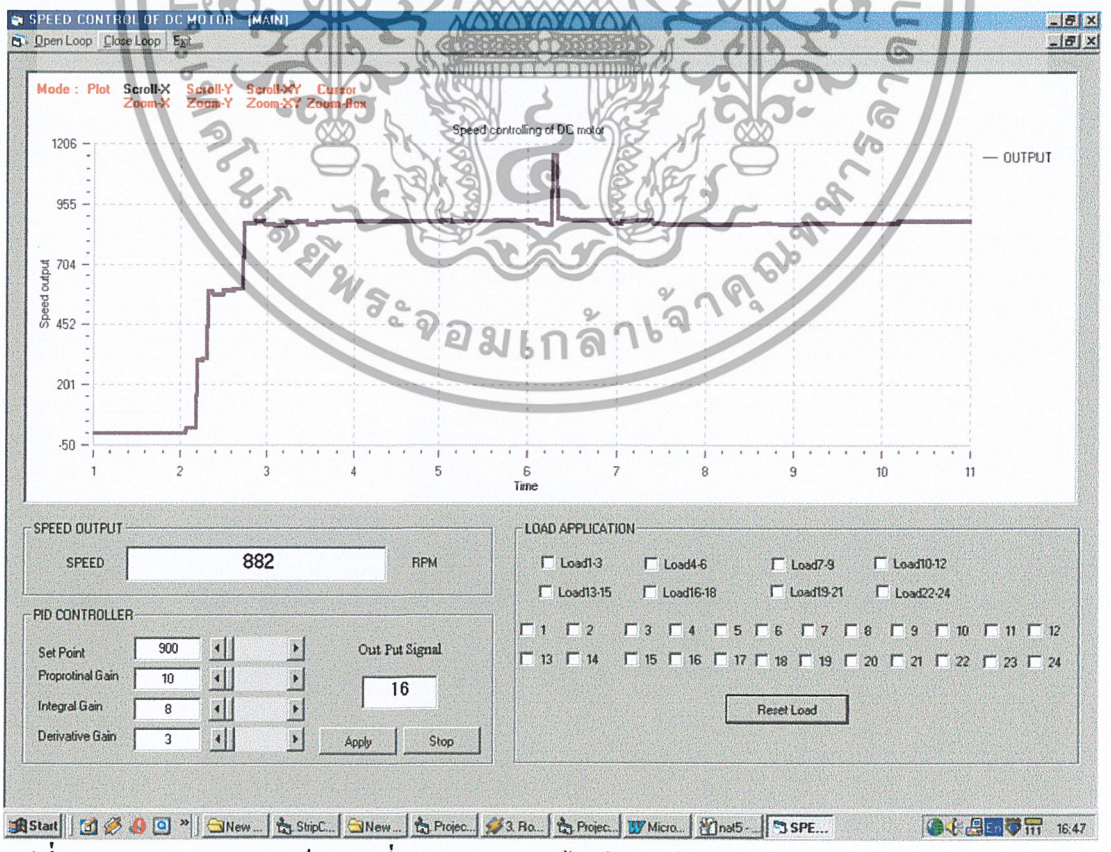

**รูปที่ 4.1** กราฟแสดงความเร็วรอบที่ 600 รอบ ขณะไม่มีภาระโหลด



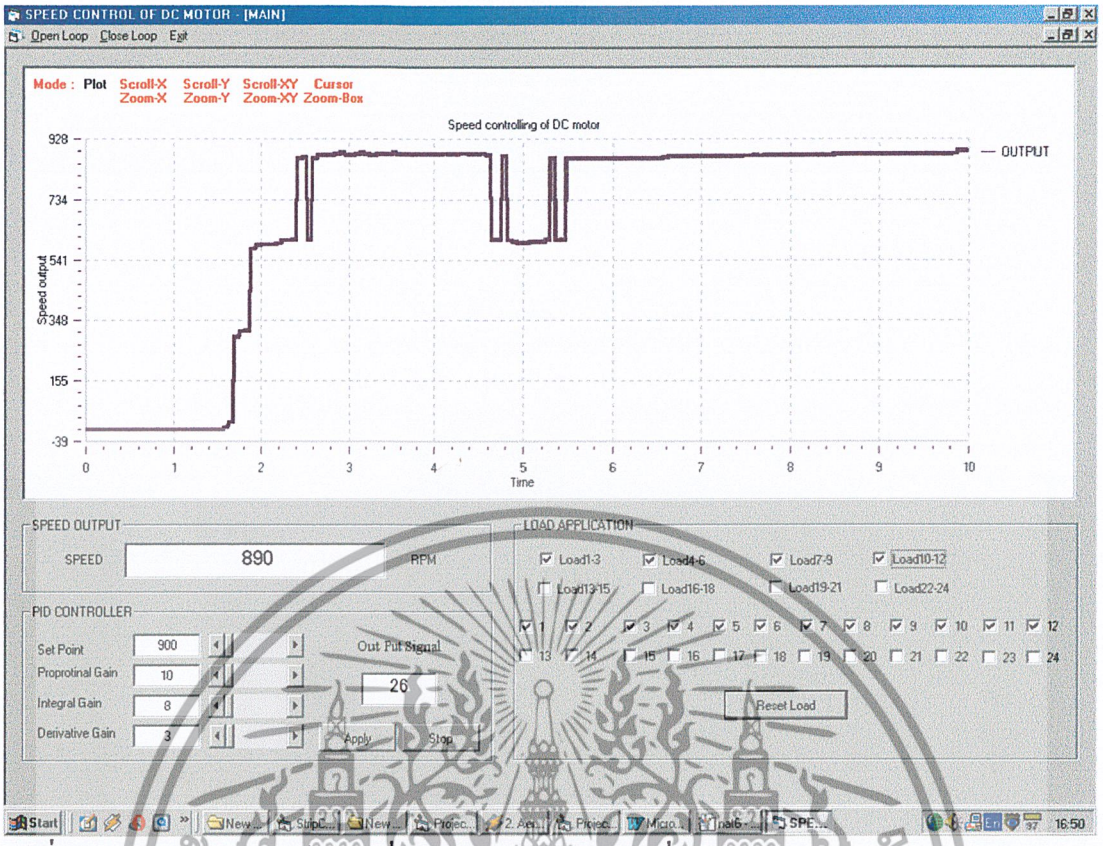

**รูปที่ 4.2** กราฟแสดงความเร็วรอบที่ 600 รอบ ขณะรับภาระที่ 12 ชุด



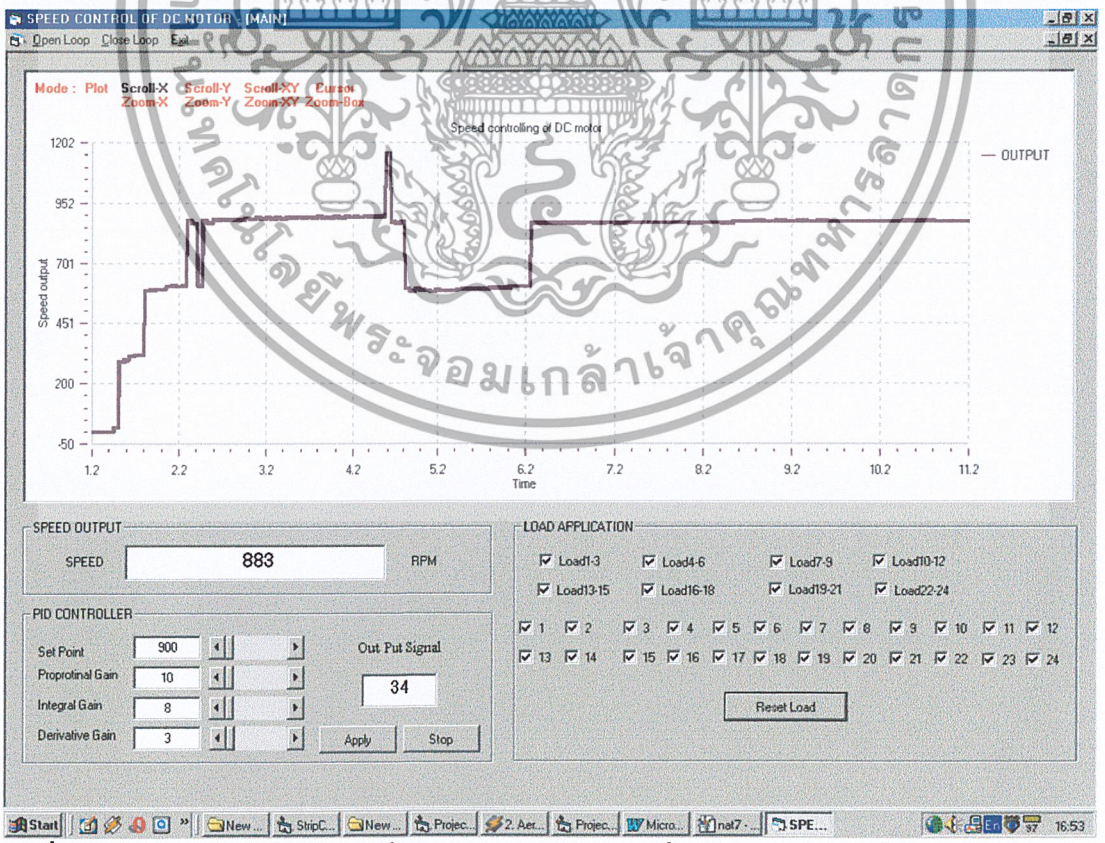

**รูปที่ 4.3** กราฟแสดงความเร็วรอบที่ 600 รอบ ขณะรับภาระที่ 24 ชุด



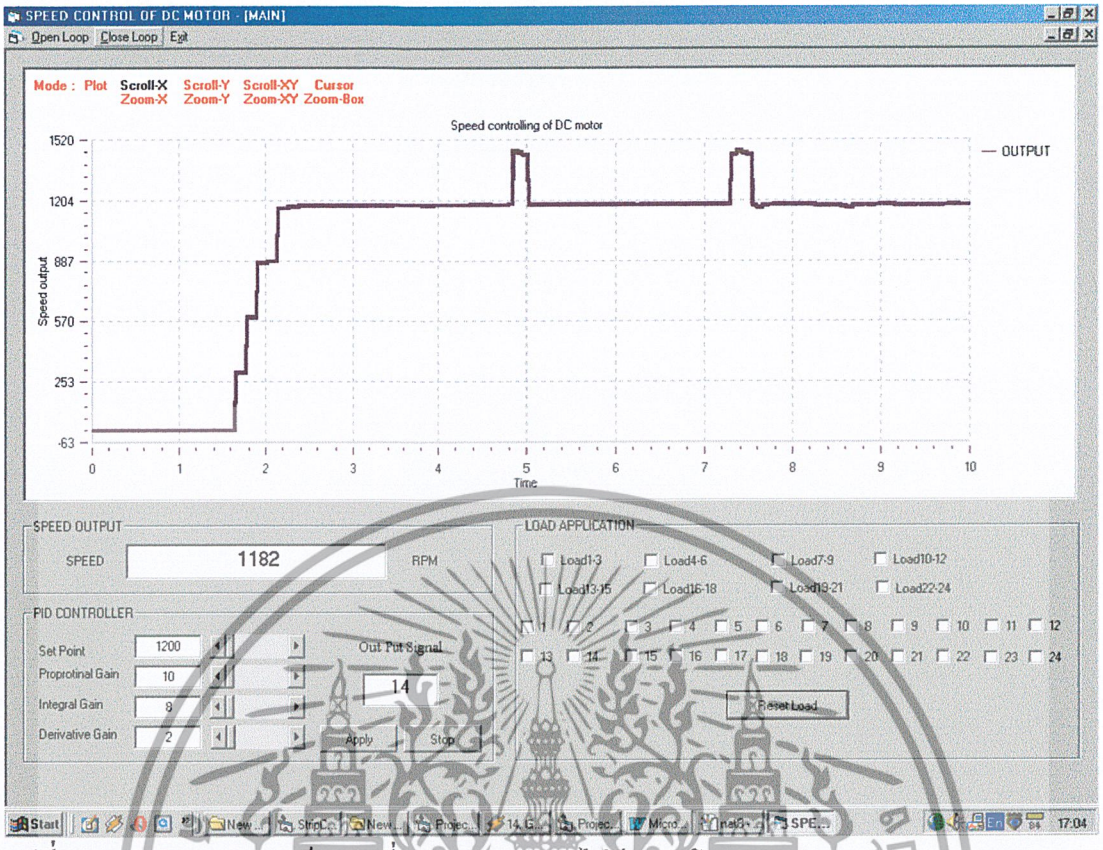

**รูปที่ 4.4** กราฟแสดงความเร็วรอบที่ 900 รอบ ขณะไม่มีภาระโหลด



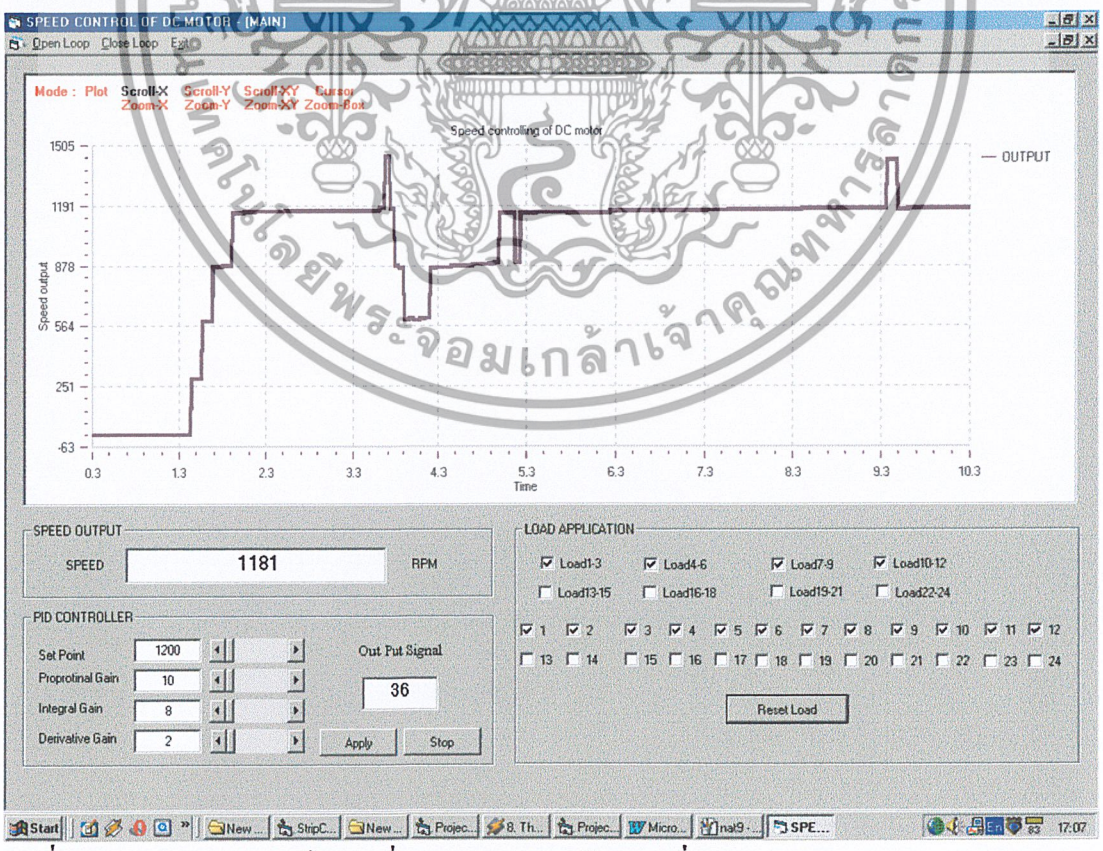

**รูปที่ 4.5** กราฟแสดงความเร็วรอบที่ 900 รอบ ขณะรับภาระที่ 12 ชุด



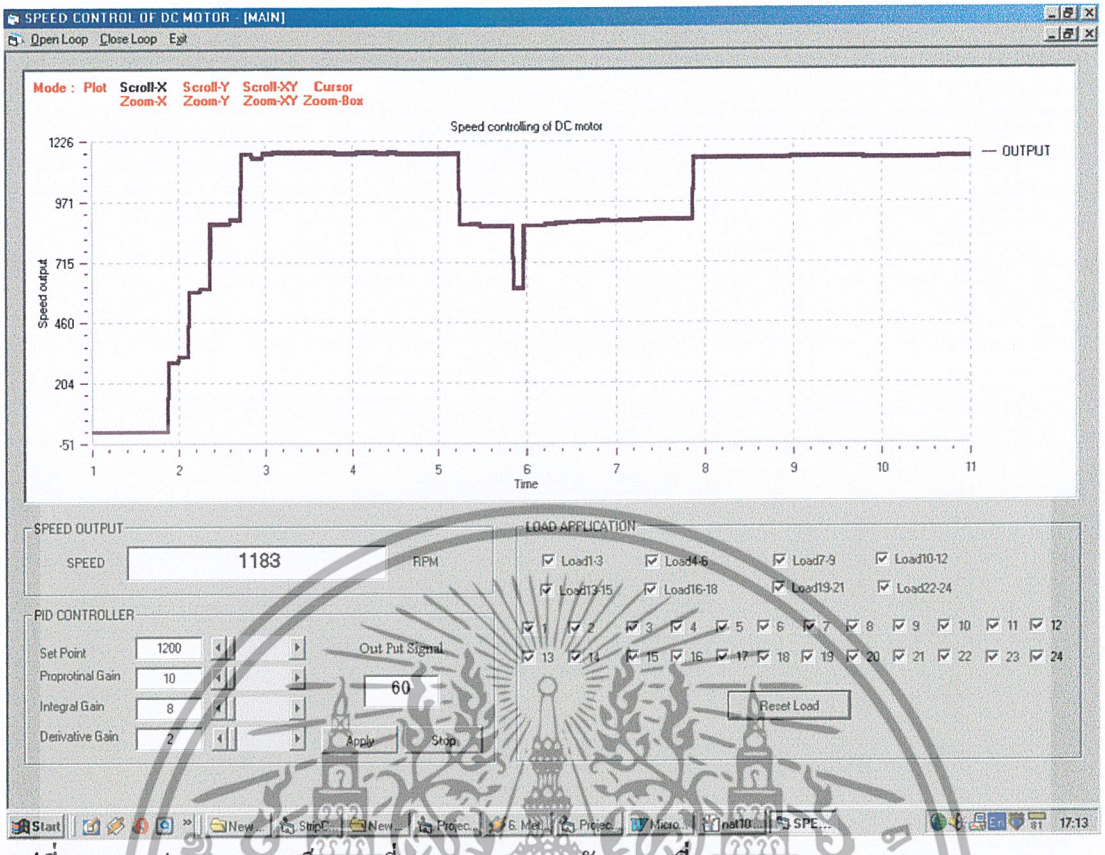

**รูปที่ 4.6** กราฟแสดงความเร็วรอบที่ 900 รอบ ขณะรับภาระที่ 24 ชุด



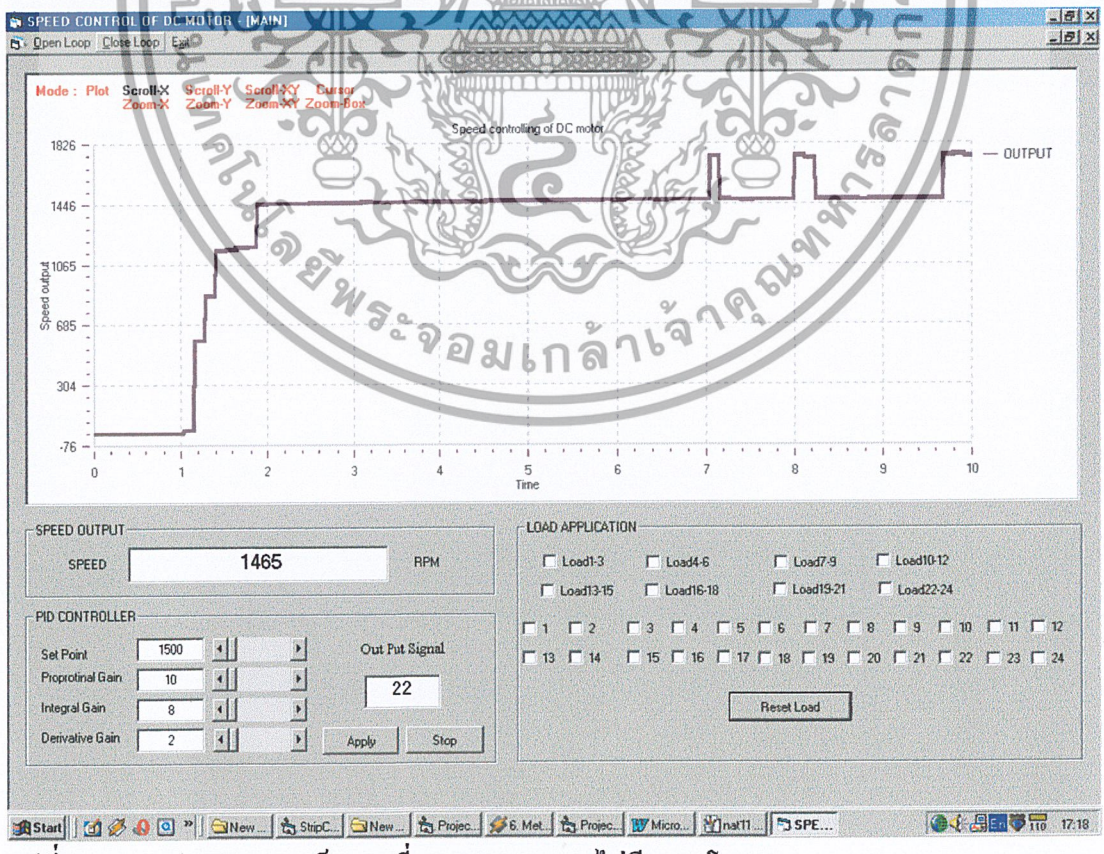

**รูปที่ 4.7** กราฟแสดงความเร็วรอบที่ 1200 รอบ ขณะไม่มีภาระโหลด



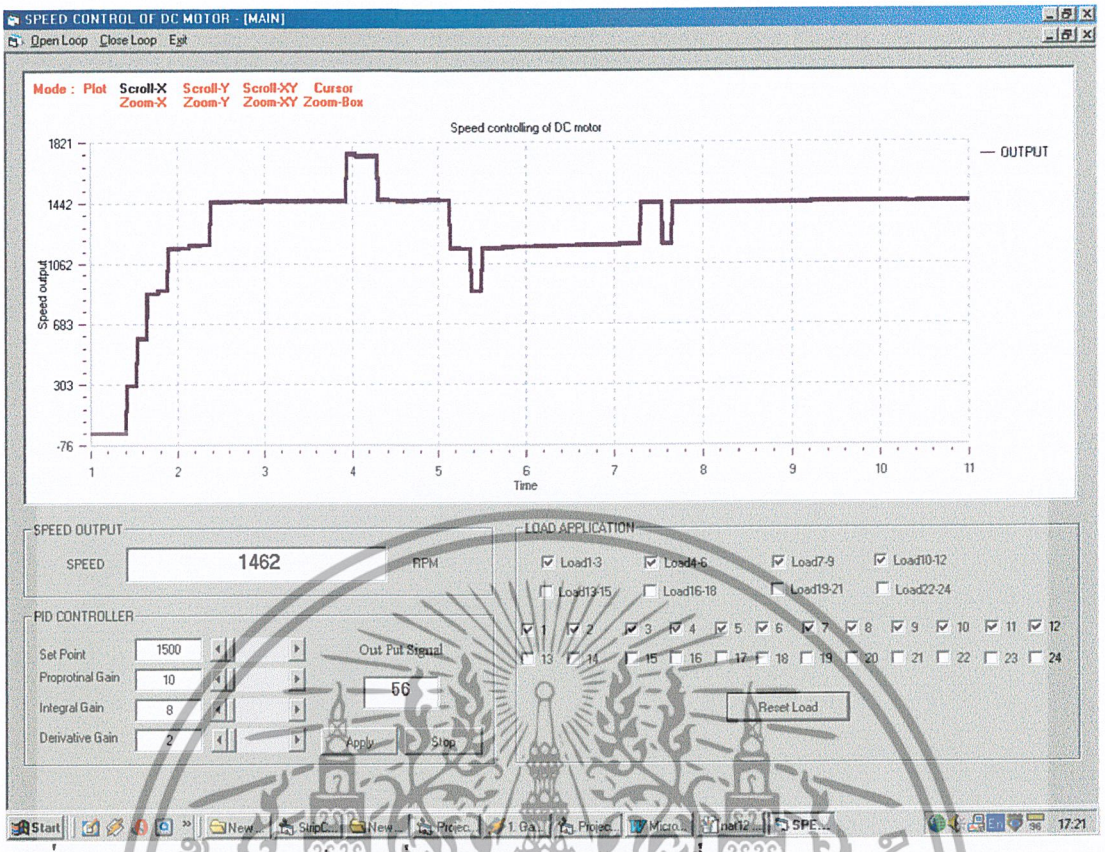

**รูปที่ 4.8** กราฟแสดงความเร็วรอบที่ 1200 รอบ ขณะรับภาระที่ 12 ชุด



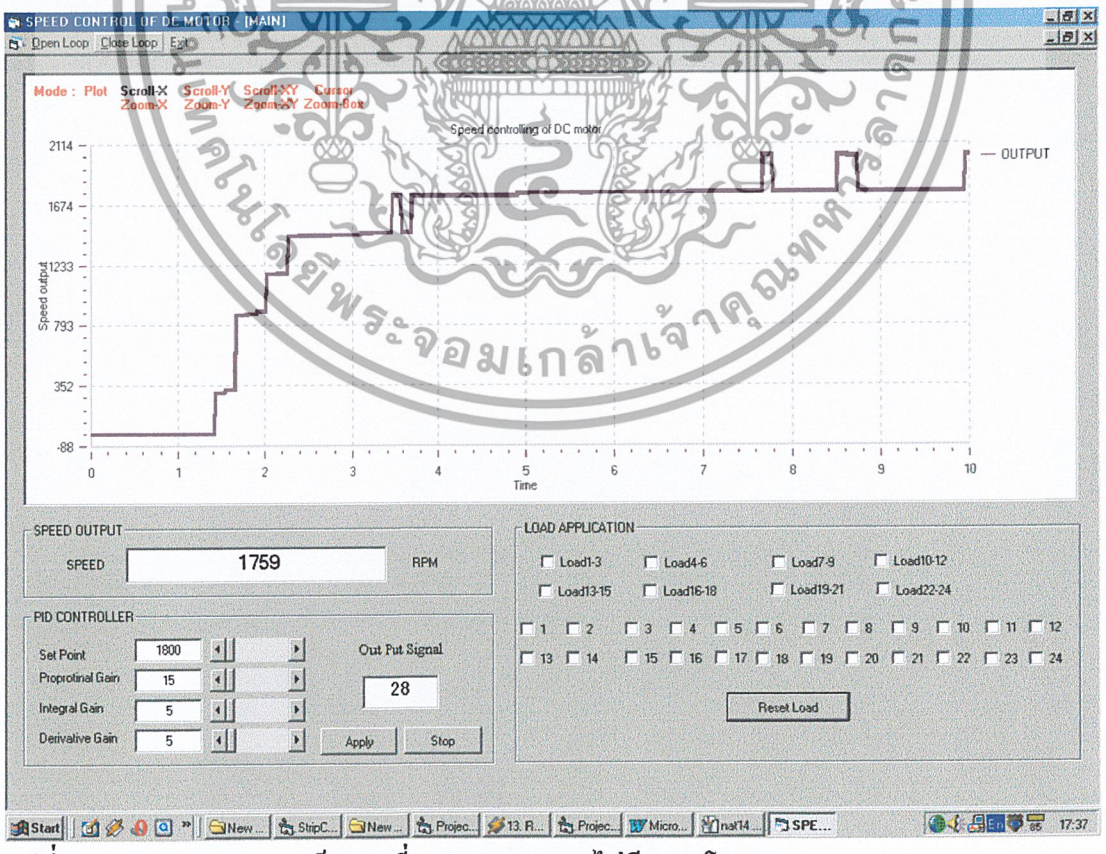

**รูปที่ 4.9** กราฟแสดงความเร็วรอบที่ 1200 รอบ ขณะรับภาระที่ 24 ชุด



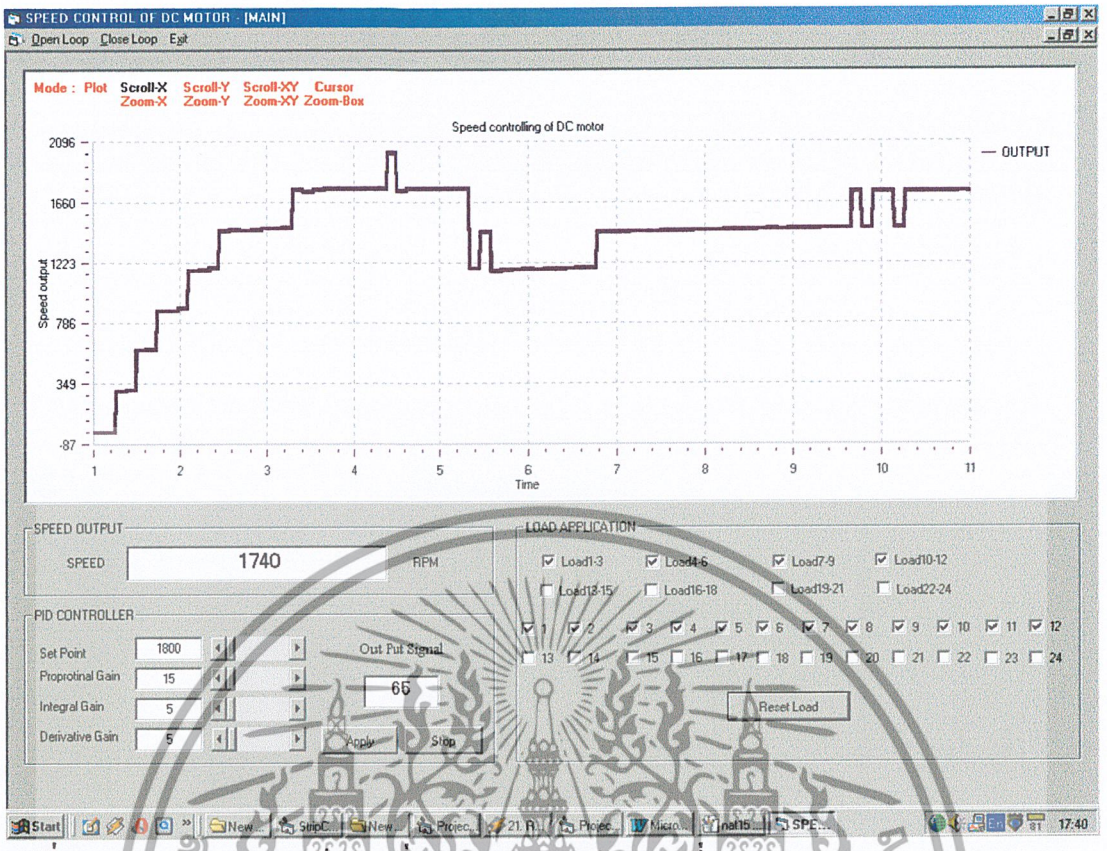

**รูปที่ 4.10** กราฟแสดงความเร็วรอบที่ 1500 รอบ ขณะไม่มีภาระโหลด



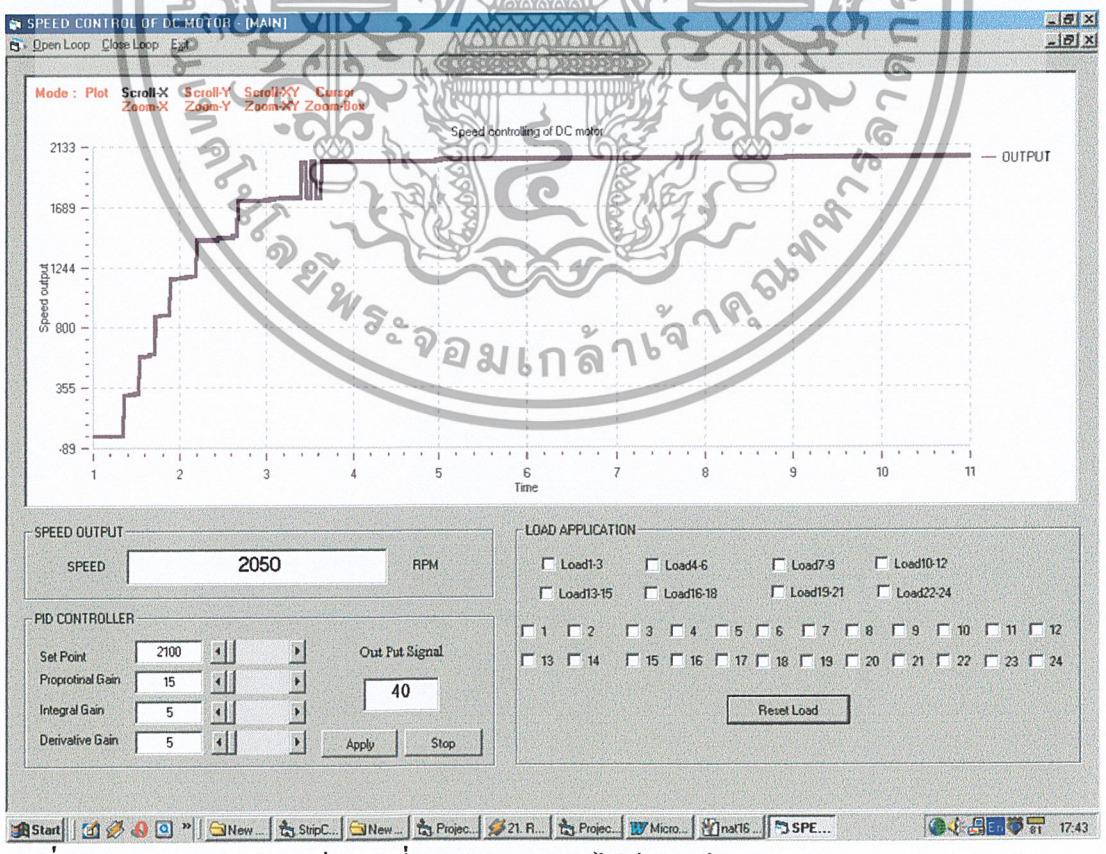

**รูปที่ 4.11** กราฟแสดงความเร็วรอบที่ 1500 รอบ ขณะรับภาระที่ 12 ชุด



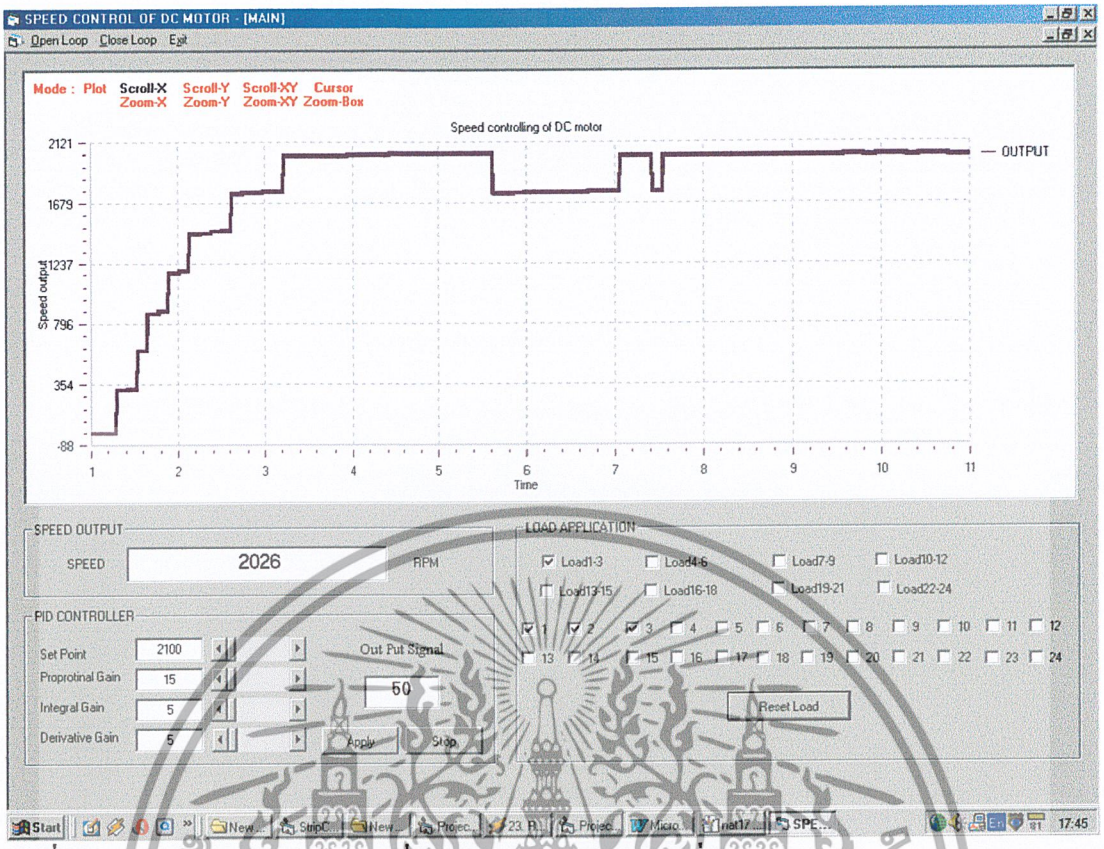

**รูปที่ 4.12** กราฟแสดงความเร็วรอบที่ 1800 รอบ ขณะไม่มีภาระโหลด




**รูปที่ 4.13** กราฟแสดงความเร็วรอบที่ 1800 รอบ ขณะรับภาระที่ 12 ชุด




**รูปที่ 4.14** กราฟแสดงความเร็วรอบที่ 2100 รอบ ขณะไม่มีภาระโหลด




**รูปที่ 4.15** กราฟแสดงความเร็วรอบที่ 2100 รอบ ขณะรับภาระที่ 3 ชุด

## 4.3 สรุปผลการทดลอง

การทดลองในขณะ Open Loop นั้นได้กระทำการทดลองโดยการป้อนสัญญาณควบคุมให้มีค่าระยะความห่างเพิ่มขึ้นครั้งละ 5 ระดับ โดยจะเห็นได้ว่าค่าของสัญญาณและค่าของความเร็วรอบที่เกิดขึ้นจะมีความสัมพันธ์กันตลอดถึงค่าของกระแสและค่าแรงดัน คือค่าความเร็วรอบของตัวมอเตอร์จะเริ่มคงเพียงแค่ได้รับสัญญาณการควบคุมที่ค่าประมาณระดับที่ 35 รวมทั้งค่ากระแสและแรงดันด้วยแสดงว่าสัญญาณควบคุมที่ผลิตขึ้นมาด้วยเลขดิจิตอลในระบบเลขฐานสอง 8 บิตซึ่งได้ทั้งหมด 255 ระดับ ซึ่งสามารถที่จะใช้ควบคุมมอเตอร์ที่มีพิกัดสูงกว่ามอเตอร์ที่ทำการทดลองอยู่ได้

ส่วนการทดลองในขณะ Close Loop จะกำหนดค่า Set Point หรือค่าความเร็วรอบที่ต้องการเพื่อให้มอเตอร์ทำงานหลังจากนั้นก็ทดลองเพิ่มโหลดให้แก่ตัวมอเตอร์ทีละ Step จะสังเกตเห็นว่าความเร็วรอบของมอเตอร์ลดลงแต่ในชั่วขณะหนึ่งเท่านั้น เพราะว่าวงจรควบคุมจะทำการผลิตสัญญาณควบคุมเพื่อป้อนให้แก่วงจรขับมอเตอร์เพื่อขับมอเตอร์ให้รักษาความเร็วรอบให้เป็นไปตามค่า Set Point ที่ต้องการ จากผลการทดลองได้กระทำการวัดทั้งค่าสัญญาณควบคุม ค่ากระแสและค่าแรงดันของมอเตอร์พบว่าจะมีค่าเพิ่มขึ้นเมื่อโหลดเพิ่มขึ้นแต่ค่าความเร็วรอบคงที่ตามค่า Set Point ซึ่งเป็นไปตามที่ต้องการ และจะพบว่าทั้งค่าสัญญาณควบคุม ค่ากระแสตลอดถึงค่าแรงดันของมอเตอร์จะมีค่าคงที่เมื่อค่าความเร็วรอบมอเตอร์เกินพิกัดของมอเตอร์ที่นำมาทำการทดลอง ถ้าหากทำการเพิ่มโหลดให้แก่มอเตอร์อีกค่าความเร็วรอบจะลดลงไม่เป็นไปตามค่า Set Point ตามที่ต้องการเนื่องจากในขณะที่ต้องรับภาระโหลดอยู่นั้นค่าความเร็วรอบจริงของมอเตอร์มีค่าเกินพิกัดความเร็วที่ตัวมอเตอร์จะรับได้แม้ว่าจะกระทำการป้อนสัญญาณควบคุมเพิ่มจนถึงค่าสัญญาณควบคุมที่ระดับ 255 ก็ตาม ดังนั้นการควบคุมความเร็วรอบมอเตอร์ไฟฟ้าจะต้องพิจารณาถึงค่าพิกัดของมอเตอร์ด้วย

# บทที่ 5
# สรุปผลการวิจัยและข้อเสนอแนะ

## 5.1 บทสรุป

จากหลักการการทำงานของมอเตอร์ไฟฟ้ากระแสตรงเพียงแต่เราป้อนไฟกระแสตรงเข้าไป ความเร็วรอบที่เกิดจากการป้อนแรงดันเดียวตลอดมันก็จะหมุนโดยมีค่าคงที่ตลอด แต่ถ้าหากเราต้องการที่จะควบคุมให้มันมีความเร็วต่างๆกันวิธีง่ายๆก็คือการควบคุมกระแสโดยการใช้ ค่าความต้านทานปรับค่าได้ควบคุมค่ากระแสที่ไหลเข้ามอเตอร์ แต่วิธีการนี้จะทำให้มอเตอร์เกิดความร้อนขึ้น จะเห็นว่าถ้าหากเราลองจ่ายกระแสไฟฟ้าเพื่อให้มอเตอร์ทำงานแล้วความเร็วรอบของมันจะไม่ได้เริ่มขึ้นอย่างทันทีทันใดมันต้องใช้เวลานิดหน่อยในการที่จะหมุนไปยังความเร็วสูงสุดของมัน และถ้าหากเราหยุดจ่ายกระแสไฟฟ้าแล้วมันก็ใช่ว่าจะหยุดอย่างทันทีทันใดมันจะค่อยๆลดความเร็วลงมา ถ้าหากว่าเราสามารถจ่ายและหยุดจ่ายกระแสไฟฟ้าได้เร็วพอแล้วละก็เราจะพบว่าความเร็วของมันนั้นก็จะมีค่าคงที่ค่าหนึ่งอยู่ระหว่างความเร็วสูงสุดและการหยุดหมุน แน่นอนเราไม่มีความไวพอแต่อุปกรณ์อิเล็กทรอนิกส์ที่สามารถช่วยเราได้ นี่เองจึงเป็นที่มาของการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง โดยการสร้างสัญญาณที่มีลักษณะเป็น Pulse ที่จะป้อนให้แก่มอเตอร์ไฟฟ้า โดยที่มีช่วงการป้อนและหยุดป้อนสัญญาณที่ต่างๆกันไป

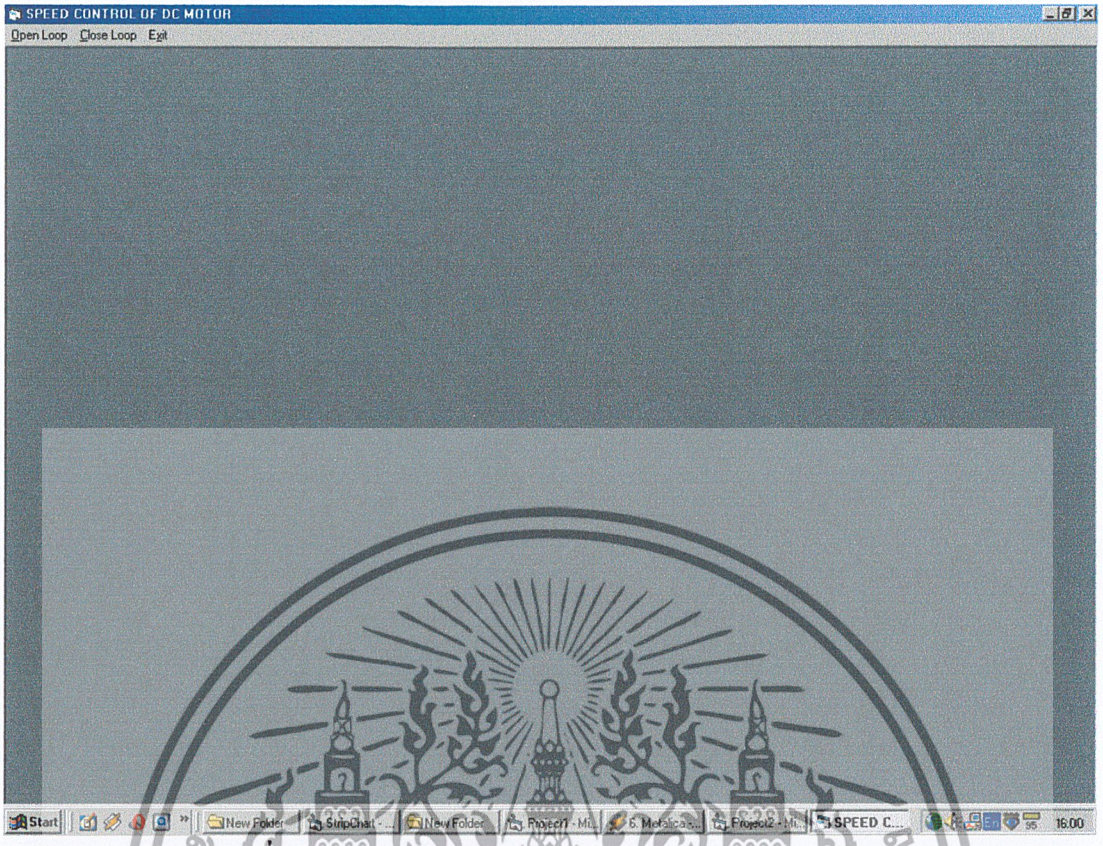
จากผลการวิจัยได้ใช้ไมโครคอนโทรลเลอร์เป็นตัวที่ผลิตสัญญาณควบคุมออกมาเพื่อควบคุมระดับของความเร็วมอเตอร์ไฟฟ้าโดยอาศัยลักษณะของการ on และ off ของระบบดิจิตอลซึ่งมีหลักการทำงานอยู่เพียงแค่ค่าสูงสุดและต่ำสุดเท่านั้น ผลที่ได้ก็เป็นไปตามที่ต้องการ

ในส่วนของการรับภาระโหลดก็สามารถที่จะควบคุมความเร็วของมอเตอร์ได้ในระดับหนึ่งเมื่อต้องรับภาระโหลดที่มอเตอร์ต้องทำการจ่ายกำลังไฟฟ้าให้ โดยที่ทำการป้อนโหลดทีละชุด เพราะที่ค่าความเร็วที่ต้องการบางระดับยังไม่สามารถที่จะควบคุมได้ เนื่องจากในการทดลองได้ใช้มอเตอร์ไฟฟ้ากระแสตรงชนิดอนุกรมมาทำการทดลอง ซึ่งคุณสมบัติของเจ้าตัวมอเตอร์ชนิดนี้มีความไวในการเปลี่ยนแปลงค่าความเร็วรอบต่อค่ากระแสที่ป้อนเข้าไปเร็วมาก แต่จากผลการทดลองที่ได้ก็เป็นที่น่าพอใจในระดับ

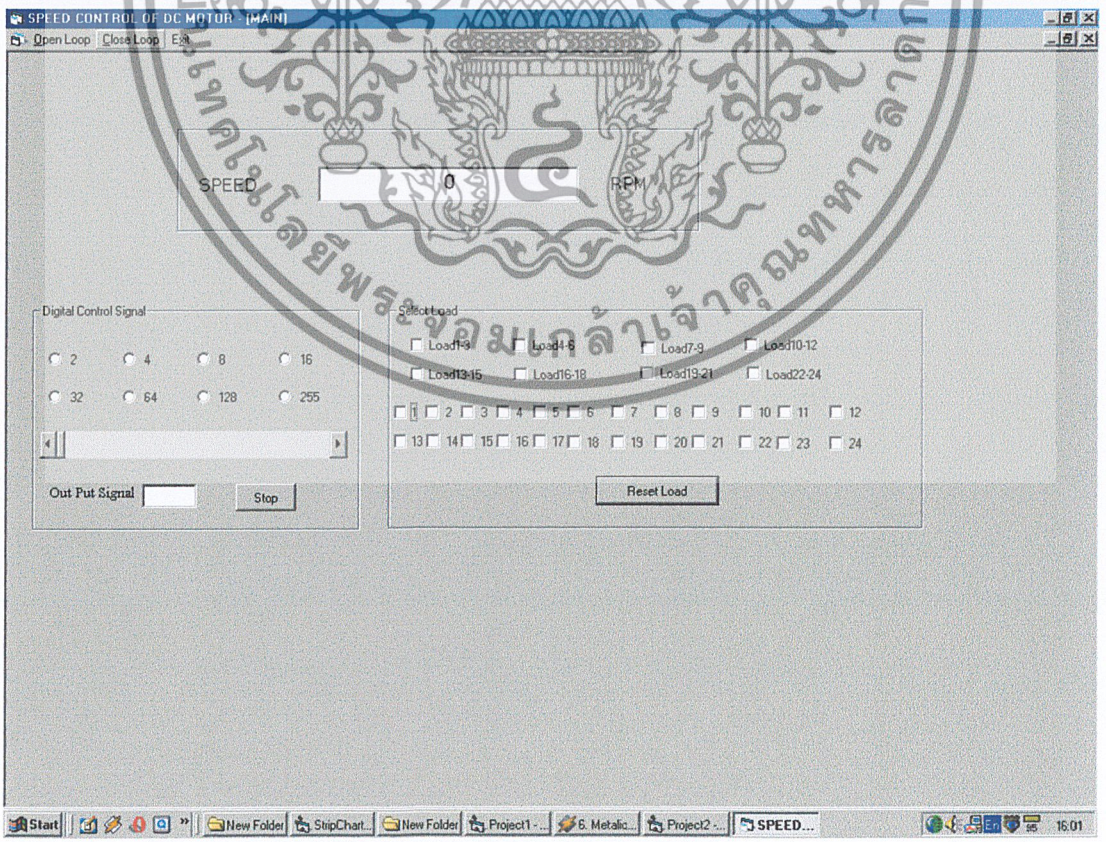
ส่วนในด้านปัญหาก็มีอยู่บ้างดังเช่น สัญญาณที่จะนำมาใช้ในการควบคุมซึ่งผลิตจากตัวไมโครคอนโทรลเลอร์นั้นได้เกิดสัญญาณรบกวนทำให้การควบคุมในช่วงแรกไม่ประสพผลสำเร็จ เนื่องมาจากเกิดกราวด์ลูปขึ้นในวงจรชุดควบคุมแต่ก็สามารถที่แก้ไขได้ โดยการทำให้กราวด์ของอุปกรณ์อิเล็กทรอนิกส์แต่ละตัวต่อเป็นลักษณะสตาร์กราวด์ปัญหาดังกล่าวก็หมดไป

ในการทดลองได้ใช้เพียงมอเตอร์ไฟฟ้ากระแสตรงแบบอนุกรมเท่านั้น ผลที่ได้จึงอาจไม่เป็นที่น่าพอใจ ซึ่งสามารถเห็นได้จากการควบคุมความเร็วที่ได้อยู่เพียงบางค่าเท่านั้น จึงน่าที่จะได้นำมอเตอร์ไฟฟ้ากระแสตรงชนิดอื่นๆมาทดลองแทนบ้าง ผลที่ได้อาจจะมีข้อเปรียบเทียบหรือได้ผลที่แตกต่างกันออกไปเพื่อเป็นแนวทางที่จะเลือกการใช้งานได้อย่างเหมาะสมและมีประสิทธิภาพมากที่สุดในการที่จะนำไปใช้งาน ส่วนในลักษณะของสัญญาณที่ใช้ในการควบคุมนั้นมีเพียงแค่ 8 บิตหากแปลงออกมาก็ได้ 255 ระดับ อาจจะประยุกต์ให้เพิ่มขึ้นเป็น 16 บิตซึ่งหากแปลงออกมาสัญญาณที่ได้จะมีถึง 65535 ระดับเลยทีเดียว การควบคุมก็จะสามารถทำได้กว้างขึ้นและยังใช้ควบคุมกับมอเตอร์ที่มีพิกัดสูงได้อีกด้วย

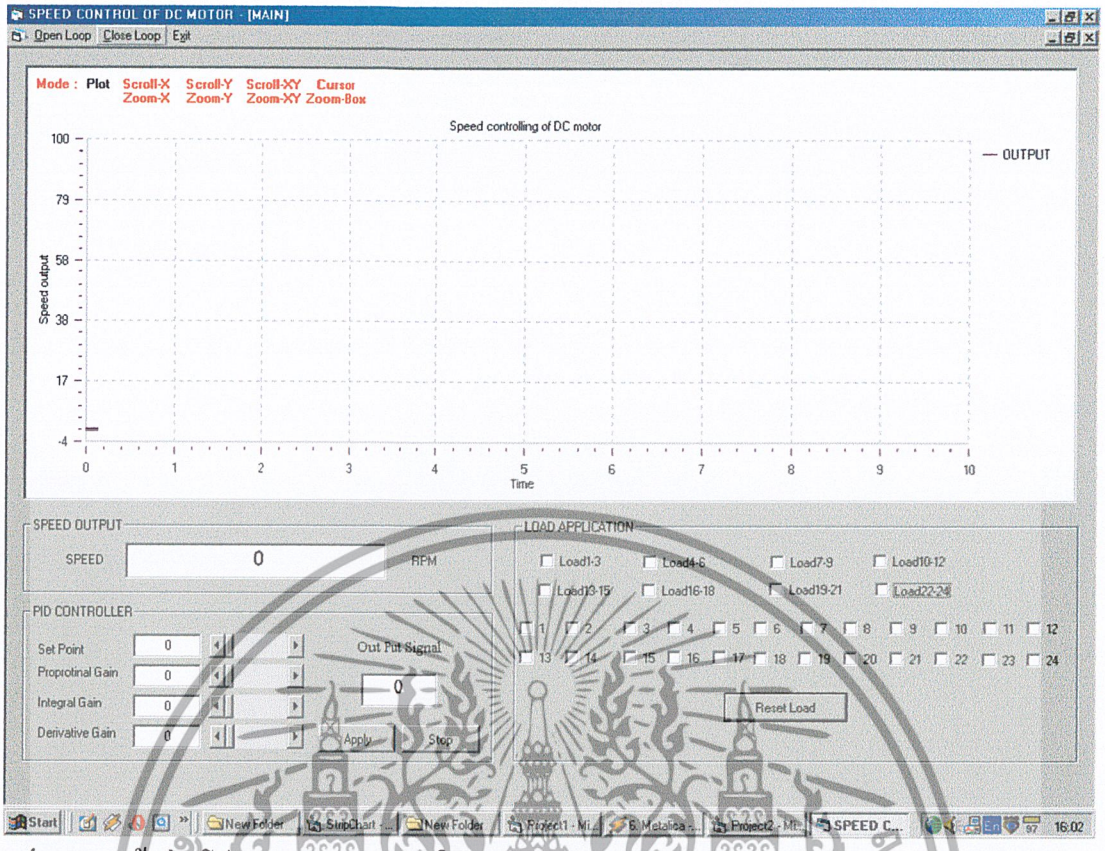

# ภาคผนวก




**รูปแสดง** หน้าต่างเมนูเริ่มต้นโปรแกรมควบคุม




**รูปแสดง** หน้าต่างโปรแกรมการควบคุมในขณะ Open Loop




**รูปแสดง** หน้าต่างโปรแกรมการควบคุมในขณะ Close Loop

MDIFormMain – 1

```
Private Sub mnu_Close_Click()
If Frm_Open.MSComm1.PortOpen = True Then Frm_Open.MSComm1.PortOpen = False
   Frm_Open.Hide
   Frm_Close.Show
If Frm_Close.MSComm1.PortOpen = False Then Frm_Close.MSComm1.PortOpen = True
End Sub

Private Sub mnu_ex_Click()
   End
End Sub

Private Sub mnu_Open_Click()
If Frm_Close.MSComm1.PortOpen = True Then
      Frm_Close.MSComm1.PortOpen = False
Else
End If
      Frm_Close.Hide
      Frm_Open.Show
If Frm_Open.MSComm1.PortOpen = False Then
      Frm_Open.MSComm1.PortOpen = True
Else
End If
End Sub
```

Frm_Open (MAIN.form)

```
Private Sub Command1_Click()
For i = 0 To 7
PA(i).Value = 0
PB(i).Value = 0
PC(i).Value = 0
PD(i).Value = 0
Next i
End Sub

Private Sub Command2_Click()
   HScroll1.Value = 0
   Option1.Value = False
   Option2.Value = False
   Option3.Value = False
   Option4.Value = False
   Option5.Value = False
   Option6.Value = False
   Option7.Value = False
   Option8.Value = False
End Sub

Private Sub HScroll1_Change()
   Label_BRM.Caption = HScroll1.Value
End Sub

Private Sub HScroll1_Scroll()
   Label_BRM.Caption = HScroll1.Value
End Sub
```

```
Private Sub MSComm1_OnComm()
Dim c As String
Dim b As Single
   c = MSComm1.Input
    If c = "" Then Exit Sub
    b = Hex_To_Dec(c)
  Label_SPEED.Caption = Str$(b)
  End Sub

Private Sub Option1_Click()
   HScroll1.Value = 2
End Sub

Private Sub Option2_Click()
   HScroll1.Value = 4
End Sub

Private Sub Option3_Click()
   HScroll1.Value = 8
End Sub

Private Sub Option4_Click()
   HScroll1.Value = 16
End Sub

Private Sub Option5_Click()
   HScroll1.Value = 32
End Sub
```

```
Private Sub Option6_Click()
    HScroll1.Value = 64
End Sub

Private Sub Option7_Click()
    HScroll1.Value = 128
End Sub

Private Sub Option8_Click()
    HScroll1.Value = 255
End Sub

Private Sub PD_Click(Index As Integer)
Select Case Index
Case Is = 0
        PA(0).Value = PD(0).Value
        PA(1).Value = PD(0).Value
        PA(2).Value = PD(0).Value
Case Is = 1
        PA(3).Value = PD(1).Value
        PA(4).Value = PD(1).Value
        PA(5).Value = PD(1).Value
Case Is = 2
        PA(6).Value = PD(2).Value
        PA(7).Value = PD(2).Value
        PB(0).Value = PD(2).Value
Case Is = 3
        PB(1).Value = PD(3).Value
        PB(2).Value = PD(3).Value
        PB(3).Value = PD(3).Value
```

```
Case Is = 4
        PB(4).Value = PD(4).Value
        PB(5).Value = PD(4).Value
        PB(6).Value = PD(4).Value
Case Is = 5
        PB(7).Value = PD(5).Value
        PC(0).Value = PD(5).Value
        PC(1).Value = PD(5).Value
Case Is = 6
        PC(2).Value = PD(6).Value
        PC(3).Value = PD(6).Value
        PC(4).Value = PD(6).Value
Case Is = 7
        PC(5).Value = PD(7).Value
        PC(6).Value = PD(7).Value
        PC(7).Value = PD(7).Value
End Select
End Sub

Private Sub Timer1_Timer()
If MSComm1.PortOpen = False Then Exit Sub
   MSComm1.Output = "@1"
   MSComm1.Output = Dec_To_Hex(HScroll1.Value)
Dim A(0 To 7) As Integer, b(0 To 7) As Integer, c(0 To 7) As Integer
  For i = 0 To 7
   If PA(i).Value = 1 Then
      A(i) = 0
   Else
      A(i) = 2 ^ i
   End If
```

```
If PB(i).Value = 1 Then
    b(i) = 0
  Else
    b(i) = 2 ^ i
  End If
  If PC(i).Value = 1 Then
    c(i) = 0
  Else
    c(i) = 2 ^ i
  End If
  Next i
Dim cc1 As Integer, cc2 As Integer, cc3 As Integer
  cc1 = A(0) + A(1) + A(2) + A(3) + A(4) + A(5) + A(6) + A(7)
  cc2 = b(0) + b(1) + b(2) + b(3) + b(4) + b(5) + b(6) + b(7)
  cc3 = c(0) + c(1) + c(2) + c(3) + c(4) + c(5) + c(6) + c(7)
  MSComm1.Output = "@3"
  MSComm1.Output = Dec_To_Hex(cc1)
  MSComm1.Output = "@4"
  MSComm1.Output = Dec_To_Hex(cc2)
  MSComm1.Output = "@5"
  MSComm1.Output = Dec_To_Hex(cc3)
End Sub

Private Sub Timer2_Timer()
If MSComm1.PortOpen = False Then Exit Sub
MSComm1.Output = "@6"
End Sub
Private Sub Timer3_Timer()
        Text1.Text = Text1.Text + MSComm1.Input
End Sub
```

**Frm_Close (frm_Close.frm)**

```
Private Sub Command1_Click()
For i = 0 To 7
   PA(i).Value = 0
   PB(i).Value = 0
   PC(i).Value = 0
   PD(i).Value = 0
Next i
End Sub

Private Sub Command2_Click()
   SP = Val(Text_SP.Text)
   KP = Val(Text_KP.Text) * 0.05
   KI = Val(Text_KI.Text) * 0.05
   KD = Val(Text_KD.Text) * 0.05
   SUMI = 0
   ERRO = 0
End Sub

Private Sub Command3_Click()
   Text_SP.Text = 0
   Text_KP.Text = 0
   Text_KI.Text = 0
   Text_KD.Text = 0
   HScroll_SP.Value = 0
   HScroll_KP.Value = 0
   HScroll_KI.Value = 0
   HScroll_KD.Value = 0
   Command2.Value = True
End Sub
```

```
Private Sub HScroll_KD_Change()
    Text_KD.Text = Str(HScroll_KD.Value)
End Sub

Private Sub HScroll_KD_Scroll()
    Text_KD.Text = Str(HScroll_KD.Value)
End Sub

Private Sub HScroll_KI_Change()
    Text_KI.Text = Str(HScroll_KI.Value)
End Sub

Private Sub HScroll_KI_Scroll()
    Text_KI.Text = Str(HScroll_KI.Value)
End Sub

Private Sub HScroll_KP_Change()
    Text_KP.Text = Str(HScroll_KP.Value)
End Sub

Private Sub HScroll_KP_Scroll()
    Text_KP.Text = Str(HScroll_KP.Value)
End Sub

Private Sub HScroll_SP_Change()
    Text_SP.Text = Str(HScroll_SP.Value)
End Sub

Private Sub HScroll_SP_Scroll()
    Text_SP.Text = Str(HScroll_SP.Value)
End Sub
```

```
Private Sub HScroll1_Change()
    Label_BRM.Caption = HScroll1.Value
End Sub

Private Sub HScroll1_Scroll()
    Label_BRM.Caption = HScroll1.Value
End Sub

Private Sub MSComm1_OnComm()
Dim c As String
Dim b As Single
    c = MSComm1.Input
      If c = "" Then Exit Sub
      b = Hex_To_Dec(c)
   Label_SPEED.Caption = Str$(b)
End Sub
Private Sub PD_Click(Index As Integer)
Select Case Index
Case Is = 0
        PA(0).Value = PD(0).Value
        PA(1).Value = PD(0).Value
        PA(2).Value = PD(0).Value
Case Is = 1
        PA(3).Value = PD(1).Value
        PA(4).Value = PD(1).Value
        PA(5).Value = PD(1).Value
Case Is = 2
        PA(6).Value = PD(2).Value
        PA(7).Value = PD(2).Value
        PB(0).Value = PD(2).Value
```

```
Case Is = 3
        PB(1).Value = PD(3).Value
        PB(2).Value = PD(3).Value
        PB(3).Value = PD(3).Value
Case Is = 4
        PB(4).Value = PD(4).Value
        PB(5).Value = PD(4).Value
        PB(6).Value = PD(4).Value
Case Is = 5
        PB(7).Value = PD(5).Value
        PC(0).Value = PD(5).Value
        PC(1).Value = PD(5).Value
Case Is = 6
        PC(2).Value = PD(6).Value
        PC(3).Value = PD(6).Value
        PC(4).Value = PD(6).Value
Case Is = 7
        PC(5).Value = PD(7).Value
        PC(6).Value = PD(7).Value
        PC(7).Value = PD(7).Value
End Select
End Sub

Private Sub Timer1_Timer()
If MSComm1.PortOpen = False Then Exit Sub
   MSComm1.Output = "@1"
   MSComm1.Output = Dec_To_Hex(PID(Val(Label_SPEED.Caption)))
Dim A(0 To 7) As Integer, b(0 To 7) As Integer, c(0 To 7) As Integer
  For i = 0 To 7
   If PA(i).Value = 1 Then
      A(i) = 0
```

```
    Else
       A(i) = 2 ^ i
    End If
    If PB(i).Value = 1 Then
       b(i) = 0
    Else
       b(i) = 2 ^ i
    End If
    If PC(i).Value = 1 Then
       c(i) = 0
    Else
       c(i) = 2 ^ i
    End If
    Next i
Dim cc1 As Integer, cc2 As Integer, cc3 As Integer
    MSComm1.Output = "@3"
    cc1 = A(0) + A(1) + A(2) + A(3) + A(4) + A(5) + A(6) + A(7)
    cc2 = b(0) + b(1) + b(2) + b(3) + b(4) + b(5) + b(6) + b(7)
    cc3 = c(0) + c(1) + c(2) + c(3) + c(4) + c(5) + c(6) + c(7)
    MSComm1.Output = Dec_To_Hex(cc1)
    MSComm1.Output = "@4"
    MSComm1.Output = Dec_To_Hex(cc2)
    MSComm1.Output = "@5"
    MSComm1.Output = Dec_To_Hex(cc3)
End Sub
Private Sub Timer2_Timer()
        If MSComm1.PortOpen = False Then Exit Sub
         MSComm1.Output = "@6"
End Sub
```

BRM (BRM.bas)

```
Public SP As Single
Public KP As Single
Public KI As Single
Public KD As Single
Public SUMI As Single
Public ERRO As Single
Public Function Hex_To_Dec(A As String)
    Dim b1 As String
    Dim b2 As String
    Dim b3 As String
    Dim b4 As String
    Dim c1 As Integer
    Dim c2 As Integer
    Dim c3 As Integer
    Dim c4 As Integer
      b1$ = Mid$(A$, 3, 1)
      b2$ = Mid$(A$, 4, 1)
      b3$ = Mid$(A$, 5, 1)
      b4$ = Mid$(A$, 6, 1)
      c1 = Asc(b1$)
      c1 = c1 - 48
      c2 = Asc(b2$)
      c2 = c2 - 48
      c3 = Asc(b3$)
      c3 = c3 - 48
      c4 = Asc(b4$)
      c4 = c4 - 48
      Hex_To_Dec = (c1 * (16 ^ 3)) + (c2 * (16 ^ 2)) + (c3 * 16) + c4
End Function
```

```
Public Function Dec_To_Hex(A As Integer)
   Dim b1 As Integer
   Dim b2  As Integer
      b1 = A \ 16
      b2 = A Mod 16
      Dec_To_Hex = Chr$(b1 + 48) & Chr$(b2 + 48)
End Function

Public Function PID(SV As Single)
Dim ERR As Single
   ERR = (SP - SV) / 255
   SUMI = SUMI + ERR
   DV = ERR - ERRO
   ERRO = ERR
   op = (KP * (ERR)) + (KI * SUMI) + (KD * (DV))
   op = Fix(op)
If op > 255 Then
   op = 255
ElseIf op < 0 Then
   op = 0
Else
End If
   PID = op
   Frm_Close.Label_BRM = op
   End Function
```

**โปรแกรม Interface ระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์**

```
        ORG   0000H
BOOT:

DELAYONSTART:  DJNZ   R1,$
        DJNZ   R0,DELAYONSTART
        LJMP   MAIN
        ORG    0200H

MAIN:       MOV    A,#0FDH          ; Set baud rate 9600
        MOV    TH1,A
        MOV    TL1,A
        MOV    TMOD,#00100100B      ; Timer1 Mode 1
        CLR    ES                   ; Disable serial interupt
        CLR    ET1                  ; Disable timer1 interupt
        SETB   TR1                  ; Set timer1 control
        SETB   TR0
        CLR    ET0
        MOV    SCON,#01010000B      ; Serial mode 1

PA1         EQU    2000H
PB1         EQU    2001H
PC1         EQU    2002H
PCO1        EQU    2003H
PA2         EQU    4000H
PB2         EQU    4001H
PC2         EQU    4002H
PCO2        EQU    4003H
PA3         EQU    6000H
PB3         EQU    6001H
PC3         EQU    6002H
```

```
PCO3        EQU    6003H

SET8255:
            MOV    DPTR,#PCO1
            MOV    A,#90H
            MOVX   @DPTR,A
            MOV    DPTR,#PCO2
            MOV    A,#80H
            MOVX   @DPTR,A
            MOV    DPTR,#PCO3
            MOVX   @DPTR,A

START:
            LCALL  RX_BYTE
            CJNE   A,#'@',START
            LCALL  RX_BYTE
            CJNE   A,#'0',MENU1
            LJMP   MENU_RUN0
MENU1:      CJNE   A,#'1',MENU2
            LCALL  RX_BYTE
            MOV    R6,A
            LCALL  RX_BYTE
            MOV    R7,A
            LJMP   MENU_RUN1
MENU2:      CJNE   A,#'2',MENU3
            LCALL  RX_BYTE
            MOV    R6,A
            LCALL  RX_BYTE
            MOV    R7,A
            LJMP   MENU_RUN2
```

```
MENU3:      CJNE   A,#'3',MENU4
            LCALL  RX_BYTE
            MOV    R6,A
            LCALL  RX_BYTE
            MOV    R7,A
            LJMP   MENU_RUN3
MENU4:      CJNE   A,#'4',MENU5
            LCALL  RX_BYTE
            MOV    R6,A
            LCALL  RX_BYTE
            MOV    R7,A
            LJMP   MENU_RUN4
MENU5:      CJNE   A,#'5',MENU6
            LCALL  RX_BYTE
            MOV    R6,A
            LCALL  RX_BYTE
            MOV    R7,A
            LJMP   MENU_RUN5
MENU6:      CJNE   A,#'6',EN_MENU
            LJMP   MENU_RUN6
EN_MENU:    LJMP   START
            LJMP   START
;***************
MENU_RUN0:
            LJMP   START
;***************
MENU_RUN1:  MOV    DPTR,#PA2
            LCALL  ASCII_TO_HEX
            MOVX   @DPTR,A
            LJMP   START
;***************
```

```
MENU_RUN2:
        MOV    DPTR,#PB2
        LCALL  ASCII_TO_HEX
        MOVX   @DPTR,A
        LJMP   START
;***************
MENU_RUN3:
        MOV    DPTR,#PA3
        LCALL  ASCII_TO_HEX
        MOVX   @DPTR,A
        LJMP   START
;***************
MENU_RUN4:
        MOV    DPTR,#PB3
        LCALL  ASCII_TO_HEX
        MOVX   @DPTR,A
        LJMP   START
;***************
MENU_RUN5:
        MOV    DPTR,#PC3
        LCALL  ASCII_TO_HEX
        MOVX   @DPTR,A
        LJMP   START
;***************
MENU_RUN6:
        MOV    R5,#0E8H
        MOV    TH0,#00H
        MOV    TL0,#00H
LOOP_SEN:   MOV    R4,#03BH
        DJNZ   R4,$
        DJNZ   R5,LOOP_SEN
        MOV    20H,TL0
```

```
        MOV    21H,TH0
        LCALL  SEND_SPEED
        LJMP   START
;****************************************;
; SEND SPEED TO PC
; INPUT = 20H
; OUTPUT = SERIAL PORT
;****************************************;
SEND_SPEED:
        MOV    A,#'@'
        LCALL  TX_BYTE
        MOV    A,#'0'
        LCALL  TX_BYTE
        MOV    A,21H
        LCALL  HEX_TO_ASCII
        MOV    A,R6
        LCALL  TX_BYTE
        MOV    A,R7
        LCALL  TX_BYTE
        MOV    A,20H
        LCALL  HEX_TO_ASCII
        MOV    A,R6
        LCALL  TX_BYTE
        MOV    A,R7
        LCALL  TX_BYTE
        MOV    A,#'*'
        LCALL  TX_BYTE
        RET
```

```
;*****************************************;
; HEX TO ASCII
; INPUT = ACC
; OUTPUT = R6,R7
;*****************************************;
HEX_TO_ASCII:
        MOV    R7,A
        SWAP   A
        ANL    A,#0FH
        ORL    A,#30H
        MOV    R6,A
        MOV    A,R7
        ANL    A,#0FH
        ORL    A,#30H
        MOV    R7,A
        RET
;*****************************************
; ASCII TO HEX
; INPUT = R6,R7
; OUTPUT = A
;*****************************************
ASCII_TO_HEX:
        MOV    A,R7
        ANL    A,#0FH
        MOV    R7,A
        MOV    A,R6
        ANL    A,#0FH
        SWAP   A
        ORL    A,R7
        RET
```

```
;****************************************;
;       DELAY                  ;
;****************************************;
DELAY:      PUSH   00
            PUSH   01
            MOV    R1,#0FH
LOOPDE:     MOV    R0,#08H
            DJNZ   R0,$
            DJNZ   R1,LOOPDE
            POP    01
            POP    00
            RET
;*************************
;* Send 1-Byte to RS-232 *
;* Input  : ACC          *
;* Output : Serial port *
;*************************
;
TX_BYTE:    PUSH   IE
            CLR    TI
            MOV    SBUF,A
            JNB    TI,$
            CLR    TI
            POP    IE
            RET
```

```
;***************************
;* Receive Data From RS-232 *
;* Input  : Serial Port  *
;* Output : ACC          *
;***************************
;
RX_BYTE:   PUSH  IE
           JNB   RI,$          ; Wait data
           CLR   RI
           MOV   A,SBUF
           POP   IE
           RET
             END
```

## โปรแกรมผลิตสัญญาณควบคุมใน MCS 51

```
                ORG   0000H
BOOT:
DELAYONSTART:   DJNZ  R1,$
                DJNZ  R0,DELAYONSTART
                LJMP  START
                ORG   0200H
START:          MOV   R0,P0
                MOV   DPTR,#BRMTABLE
                MOV   R1,#00H
LOOPBRM:        MOV   A,R1
                MOVC  A,@A+DPTR
                ANL   A,R0
                JZ    NP
                MOV   P1,#00H
                LJMP  EP
NP:             MOV   P1,#01H
EP:             LCALL DELAY
                DJNZ  R1,LOOPBRM
                LJMP  START
DELAY:          MOV   R5,#0E8H
LOOPDELAY:      MOV   R4,#03BH
                DJNZ  R4,$
                DJNZ  R5,LOOPDELAY
                RET
```

BRMTABLE:

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,08H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,04H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,08H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,02H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,08H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,04H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,08H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,01H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,08H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,04H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,08H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,02H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,08H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,04H

DB 80H,40H,80H,20H,80H,40H,80H,10H

DB 80H,40H,80H,20H,80H,40H,80H,08H

```
DB  80H,40H,80H,20H,80H,40H,80H,10H
DB  80H,40H,80H,20H,80H,40H,80H,00H
END
```

## Features

- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Downloadable Flash Memory
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 2K Bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 4V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Nine Interrupt Sources
- Programmable UART Serial Channel
- SPI Serial Interface
- Low-power Idle and Power-down Modes
- Interrupt Recovery From Power-down
- Programmable Watchdog Timer
- Dual Data Pointer
- Power-off Flag

## Description

The AT89S8252 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of downloadable Flash programmable and erasable read only memory and 2K bytes of EEPROM. 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 downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with downloadable Flash on a monolithic chip, the Atmel AT89S8252 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S8252 provides the following standard features: 8K bytes of downloadable Flash, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable 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 AT89S8252 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.

The downloadable Flash can be changed a single byte at a time and is accessible through the SPI serial interface. Holding RESET active forces the SPI bus into a serial programming interface and allows the program memory to be written to or read from unless Lock Bit 2 has been activated.

ATMEL

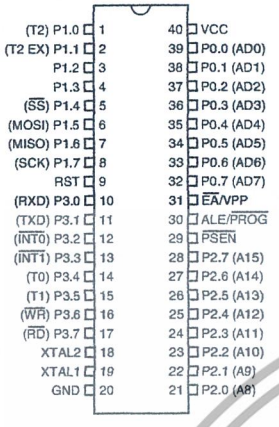
# 8-bit Microcontroller with 8K Bytes Flash

## AT89S8252

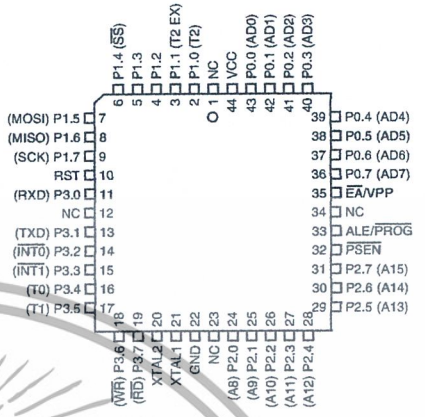
Rev. 0401E–02/00

## Pin Configurations

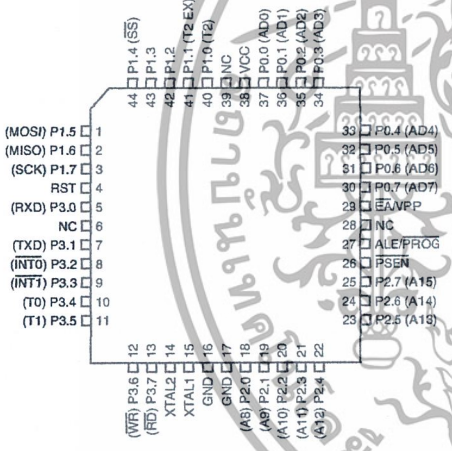
PDIP




PLCC




PQFP/TQFP




## Pin Description

### VCC

Supply voltage.

### GND

Ground.

### Port 0

Port 0 is an 8-bit open drain bbi-didirectional 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.

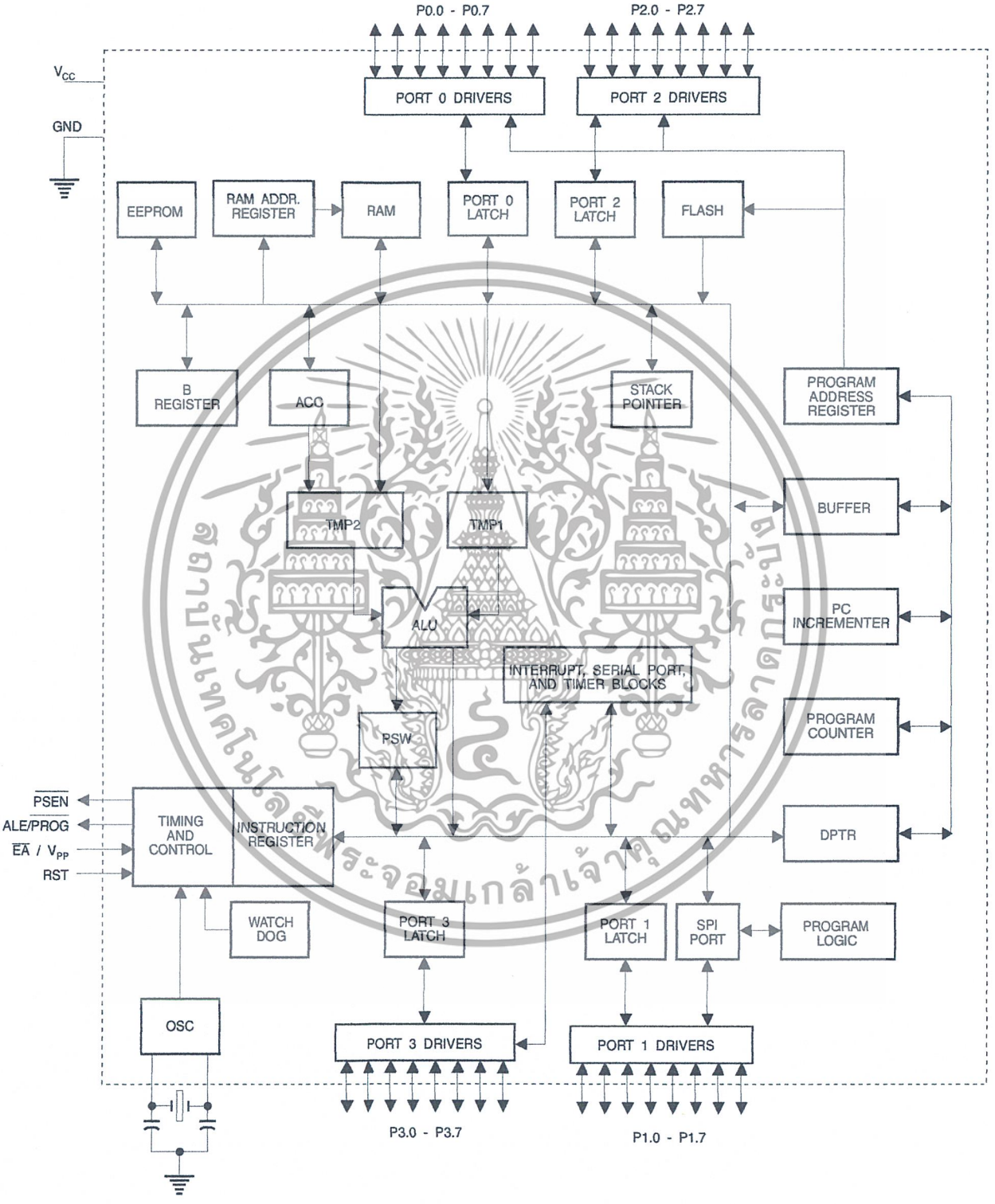
Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

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 bi-directional 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.

## Block Diagram



Some Port 1 pins provide additional functions. 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.

## Pin Description

Furthermore, P1.4, P1.5, P1.6, and P1.7 can be configured as the SPI slave port select, data input/output and shift clock input/output pins as shown in the following table.

| 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) |
| P1.4 | $\overline{SS}$ (Slave port select input) |
| P1.5 | MOSI (Master data output, slave data input pin for SPI channel) |
| P1.6 | MISO (Master data input, slave data output pin for SPI channel) |
| P1.7 | SCK (Master clock output, slave clock input pin for SPI channel) |

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

### Port 2

Port 2 is an 8-bit bi-directional 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 bi-directional 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 AT89S8252, 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.

### ALE/$\overline{PROG}$

Address Latch Enable 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.

### $\overline{PSEN}$

Program Store Enable is the read strobe to external program memory.

When the AT89S8252 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

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

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external pro-

gram memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ 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 when 12-volt programming is selected.

**XTAL1**

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

**XTAL2**

Output from the inverting oscillator amplifier.

**Table 1.** AT89S8252 SFR Map and Reset Values

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0F8H | | | | | | | | | 0FFH |
| 0F0H | B<br>00000000 | | | | | | | | 0F7H |
| 0E8H | | | | | | | | | 0EFH |
| 0E0H | ACC<br>00000000 | | | | | | | | 0E7H |
| 0D8H | | | | | | | | | 0DFH |
| 0D0H | PSW<br>00000000 | | | | | SPCR<br>000001XX | | | 0D7H |
| 0C8H | T2CON<br>00000000 | T2MOD<br>XXXXXX00 | RCAP2L<br>00000000 | RCAP2H<br>00000000 | TL2<br>00000000 | TH2<br>00000000 | | | 0CFH |
| 0C0H | | | | | | | | | 0C7H |
| 0B8H | IP<br>XX000000 | | | | | | | | 0BFH |
| 0B0H | P3<br>11111111 | | | | | | | | 0B7H |
| 0A8H | IE<br>0X000000 | | SPSR<br>00XXXXXX | | | | | | 0AFH |
| 0A0H | P2<br>11111111 | | | | | | | | 0A7H |
| 98H | SCON<br>00000000 | SBUF<br>XXXXXXXX | | | | | | | 9FH |
| 90H | P1<br>11111111 | | | | | | WMCON<br>00000010 | | 97H |
| 88H | TCON<br>00000000 | TMOD<br>00000000 | TL0<br>00000000 | TL1<br>00000000 | TH0<br>00000000 | TH1<br>00000000 | | | 8FH |
| 80H | P0<br>11111111 | SP<br>00000111 | DP0L<br>00000000 | DP0H<br>00000000 | DP1L<br>00000000 | DP1H<br>00000000 | SPDR<br>XXXXXXXX | PCON<br>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 9) 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.

**Table 2.** T2CON—Timer/Counter 2 Control Register

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

Bit Addressable

| | TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/$\overline{T2}$ | CP/$\overline{RL2}$ |
|---|---|---|---|---|---|---|---|---|
| Bit | 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 overflows 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. |

**Watchdog and Memory Control Register** The WMCON register contains control bits for the Watchdog Timer (shown in Table 3). The EEMEN and EEMWE bits are used to select the 2K bytes on-chip EEPROM, and to enable byte-write. The DPS bit selects one of two DPTR registers available.

**Table 3.** WMCON—Watchdog and Memory Control Register

WMCON Address = 96H — Reset Value = 0000 0010B

| | PS2 | PS1 | PS0 | EEMWE | EEMEN | DPS | WDTRST | WDTEN |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| PS2<br>PS1<br>PS0 | Prescaler Bits for the Watchdog Timer. When all three bits are set to "0", the watchdog timer has a nominal period of 16 ms. When all three bits are set to "1", the nominal period is 2048 ms. |
| EEMWE | EEPROM Data Memory Write Enable Bit. Set this bit to "1" before initiating byte write to on-chip EEPROM with the MOVX instruction. User software should set this bit to "0" after EEPROM write is completed. |
| EEMEN | Internal EEPROM Access Enable. When EEMEN = 1, the MOVX instruction with DPTR will access on-chip EEPROM instead of external data memory. When EEMEN = 0, MOVX with DPTR accesses external data memory. |
| DPS | Data Pointer Register Select. DPS = 0 selects the first bank of Data Pointer Register, DP0, and DPS = 1 selects the second bank, DP1 |
| WDTRST<br>RDY/$\overline{BSY}$ | Watchdog Timer Reset and EEPROM Ready/$\overline{Busy}$ Flag. Each time this bit is set to "1" by user software, a pulse is generated to reset the watchdog timer. The WDTRST bit is then automatically reset to "0" in the next instruction cycle. The WDTRST bit is Write-Only. This bit also serves as the RDY/$\overline{BSY}$ flag in a Read-Only mode during EEPROM write. RDY/$\overline{BSY}$ = 1 means that the EEPROM is ready to be programmed. While programming operations are being executed, the RDY/$\overline{BSY}$ bit equals "0" and is automatically reset to "1" when programming is completed. |
| WDTEN | Watchdog Timer Enable Bit. WDTEN = 1 enables the watchdog timer and WDTEN = 0 disables the watchdog timer. |

**SPI Registers** Control and status bits for the Serial Peripheral Interface are contained in registers SPCR (shown in Table 4) and SPSR (shown in Table 5). The SPI data bits are contained in the SPDR register. Writing the SPI data register during serial data transfer sets the Write Collision bit, WCOL, in the SPSR register. The SPDR is double buffered for writing and the values in SPDR are not changed by Reset.

**Interrupt Registers** The global interrupt enable bit and the individual interrupt enable bits are in the IE register. In addition, the individual interrupt enable bit for the SPI is in the SPCR register. Two priorities can be set for each of the six interrupt sources in the IP register.

**Dual Data Pointer Registers** To facilitate accessing both internal EEPROM 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 WMCON 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 reset under software control and is not affected by RESET.

**Table 4.** SPCR—SPI Control Register

SPCR Address = D5H — Reset Value = 0000 01XXB

| | SPIE | SPE | DORD | MSTR | CPOL | CPHA | SPR1 | SPR0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| SPIE | SPI Interrupt Enable. This bit, in conjunction with the ES bit in the IE register, enables SPI interrupts: SPIE = 1 and ES = 1 enable SPI interrupts. SPIE = 0 disables SPI interrupts. |
| SPE | SPI Enable. SPI = 1 enables the SPI channel and connects $\overline{SS}$, MOSI, MISO and SCK to pins P1.4, P1.5, P1.6, and P1.7. SPI = 0 disables the SPI channel. |
| DORD | Data Order. DORD = 1 selects LSB first data transmission. DORD = 0 selects MSB first data transmission. |
| MSTR | Master/Slave Select. MSTR = 1 selects Master SPI mode. MSTR = 0 selects Slave SPI mode. |
| CPOL | Clock Polarity. When CPOL = 1, SCK is high when idle. When CPOL = 0, SCK of the master device is low when not transmitting. Please refer to figure on SPI Clock Phase and Polarity Control. |
| CPHA | Clock Phase. The CPHA bit together with the CPOL bit controls the clock and data relationship between master and slave. Please refer to figure on SPI Clock Phase and Polarity Control. |
| SPR0<br>SPR1 | SPI Clock Rate Select. These two bits control the SCK rate of the device configured as master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the oscillator frequency, $F_{osc.}$, is as follows:<br>SPR1 SPR0 SCK = $F_{osc.}$ divided by<br>0 0 4<br>0 1 16<br>1 0 64<br>1 1 128 |

**Table 5.** SPSR – SPI Status Register

SPSR Address = AAH — Reset Value = 00XX XXXXB

| | SPIF | WCOL | – | – | – | – | – | – |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Symbol | Function |
|---|---|
| SPIF | SPI Interrupt Flag. When a serial transfer is complete, the SPIF bit is set and an interrupt is generated if SPIE = 1 and ES = 1. The SPIF bit is cleared by reading the SPI status register with SPIF and WCOL bits set, and then accessing the SPI data register. |
| WCOL | Write Collision Flag. The WCOL bit is set if the SPI data register is written during a data transfer. During data transfer, the result of reading the SPDR register may be incorrect, and writing to it has no effect. The WCOL bit (and the SPIF bit) are cleared by reading the SPI status register with SPIF and WCOL set, and then accessing the SPI data register. |

**Table 6.** SPDR – SPI Data Register

SPDR Address = 86H — Reset Value = unchanged

| | SPD7 | SPD6 | SPD5 | SPD4 | SPD3 | SPD2 | SPD1 | SPD0 |
|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

## Data Memory – EEPROM and RAM

The AT89S8252 implements 2K bytes of on-chip EEPROM for data storage and 256 bytes of RAM. The upper 128 bytes of RAM occupy a parallel space to the Special Function Registers. That means 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 that use direct addressing access 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.

The on-chip EEPROM data memory is selected by setting the EEMEN bit in the WMCON register at SFR address location 96H. The EEPROM address range is from 000H to 7FFH. The MOVX instructions are used to access the EEPROM. To access off-chip data memory with the MOVX instructions, the EEMEN bit needs to be set to "0".

The EEMWE bit in the WMCON register needs to be set to "1" before any byte location in the EEPROM can be written. User software should reset EEMWE bit to "0" if no further EEPROM write is required. EEPROM write cycles in the serial programming mode are self-timed and typically take 2.5 ms. The progress of EEPROM write can be monitored by reading the RDY/$\overline{BSY}$ bit (read-only) in SFR WMCON. RDY/$\overline{BSY}$ = 0 means programming is still in progress and RDY/$\overline{BSY}$ = 1 means EEPROM write cycle is completed and another write cycle can be initiated.

In addition, during EEPROM programming, an attempted read from the EEPROM will fetch the byte being written with the MSB complemented. Once the write cycle is completed, true data are valid at all bit locations.

## Programmable Watchdog Timer

The programmable Watchdog Timer (WDT) operates from an independent oscillator. The prescaler bits, PS0, PS1 and PS2 in SFR WMCON are used to set the period of the Watchdog Timer from 16 ms to 2048 ms. The available timer periods are shown in the following table and the actual timer periods (at $V_{CC}$ = 5V) are within ±30% of the nominal.

The WDT is disabled by Power-on Reset and during Power-down. It is enabled by setting the WDTEN bit in SFR WMCON (address = 96H). The WDT is reset by setting the WDTRST bit in WMCON. When the WDT times out without being reset or disabled, an internal RST pulse is generated to reset the CPU.

**Table 7.** Watchdog Timer Period Selection

| WDT Prescaler Bits | | | |
|---|---|---|---|
| PS2 | PS1 | PS0 | Period (nominal) |
| 0 | 0 | 0 | 16 ms |
| 0 | 0 | 1 | 32 ms |
| 0 | 1 | 0 | 64 ms |
| 0 | 1 | 1 | 128 ms |
| 1 | 0 | 0 | 256 ms |
| 1 | 0 | 1 | 512 ms |
| 1 | 1 | 0 | 1024 ms |
| 1 | 1 | 1 | 2048 ms |

## Timer 0 and 1

Timer 0 and Timer 1 in the AT89S8252 operate the same way as Timer 0 and Timer 1 in the AT89C51, AT89C52 and AT89C55. For further information, see the October 1995 Microcontroller Data Book, page 2-45, section titled, "Timer/Counters."

## 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 8.

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.

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.

**Table 8.** 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) |

## 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 1.

**Figure 1.** Timer 2 in Capture Mode

## 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 9). 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 2 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 RCAP2H 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 3. 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 2.** Timer 2 in Auto Reload Mode (DCEN = 0)

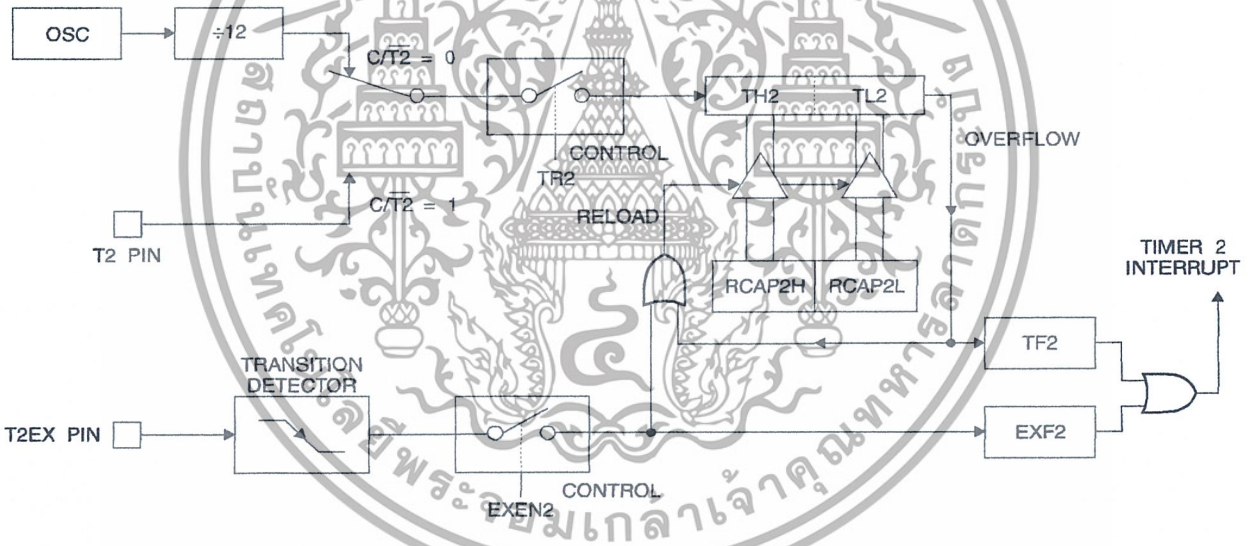



**Table 9.** T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

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

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

**Figure 3.** Timer 2 Auto Reload Mode (DCEN = 1)

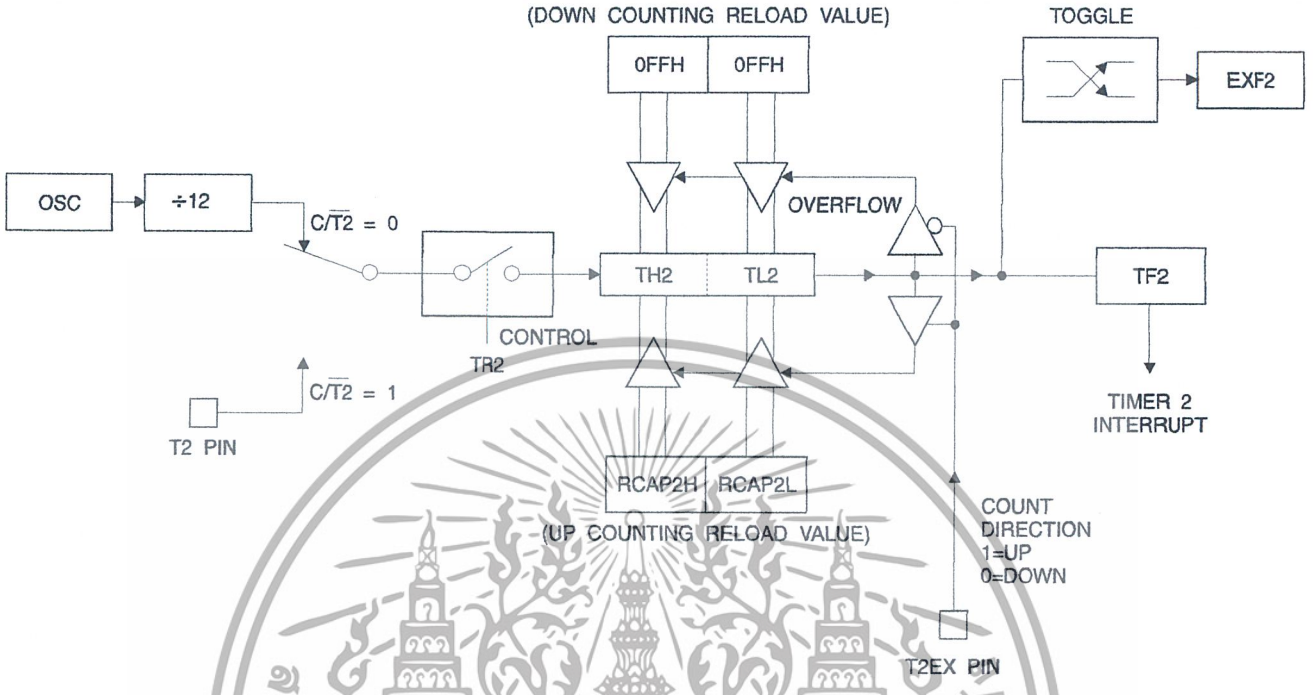
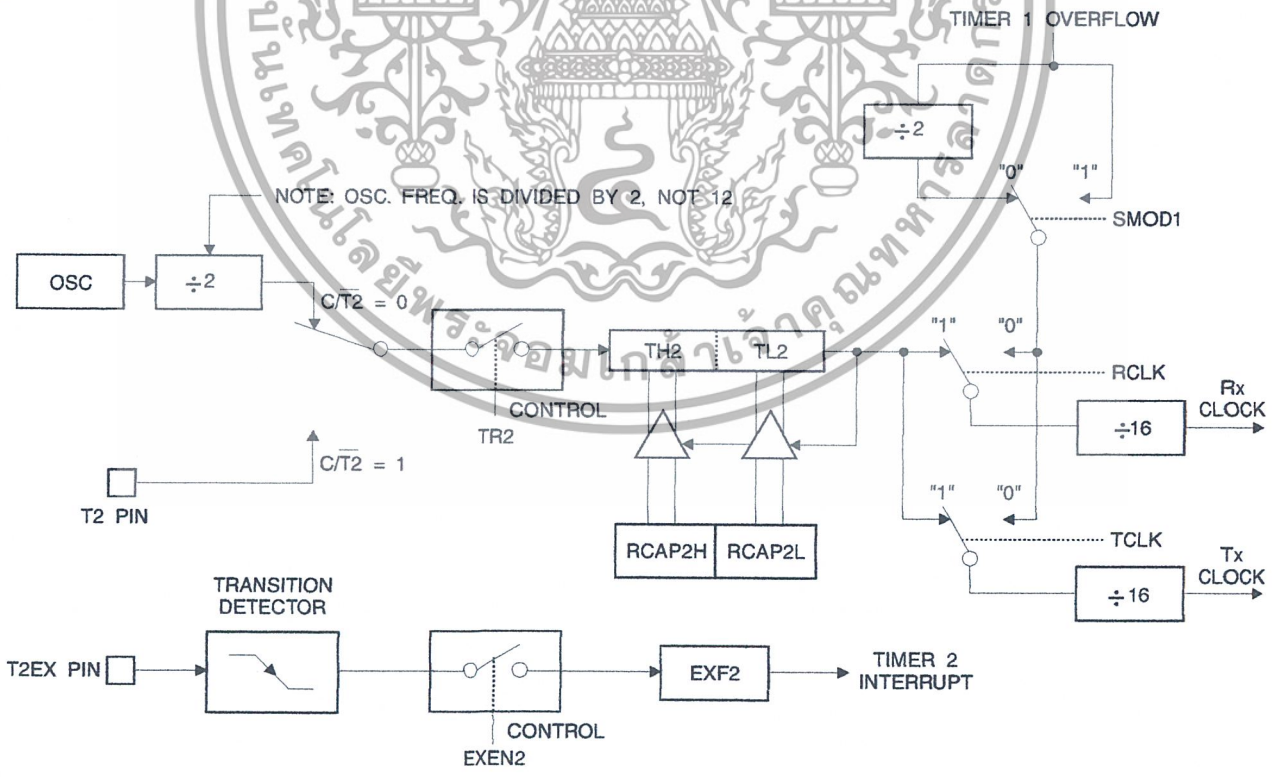



**Figure 4.** Timer 2 in Baud Rate Generator Mode



## 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 4.

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 4. 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.

## Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. 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 rollovers 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.

**Figure 5.** Timer 2 in Clock-out Mode

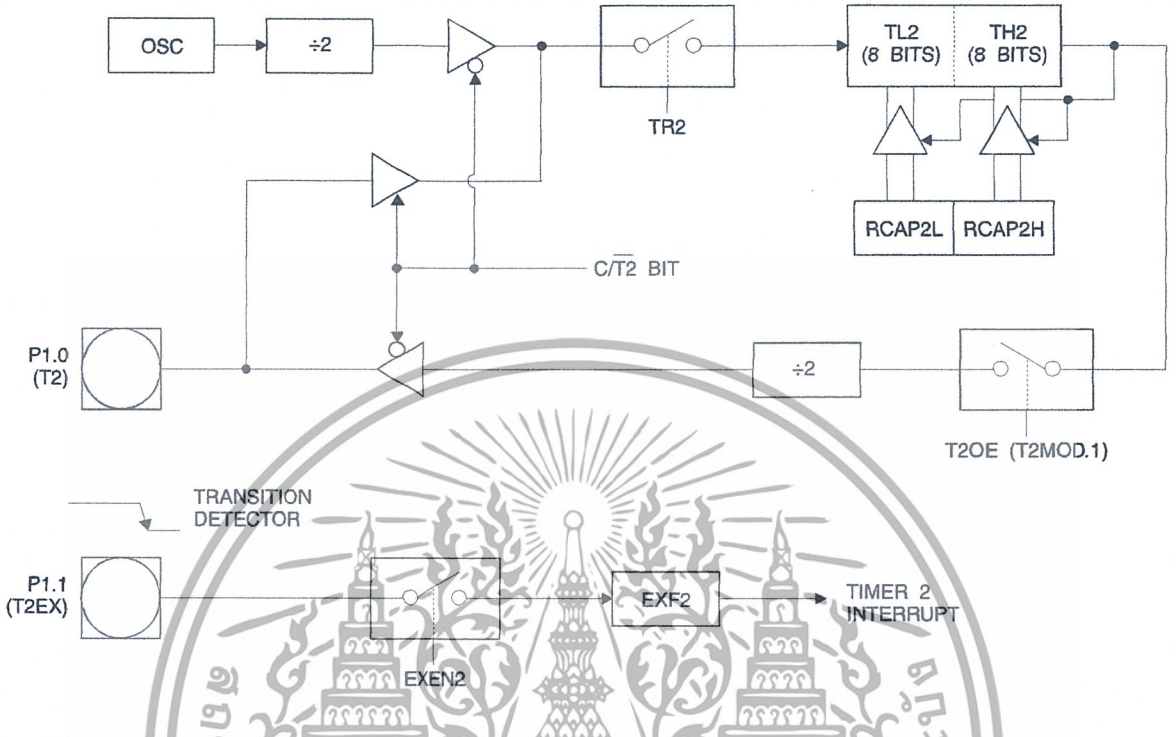
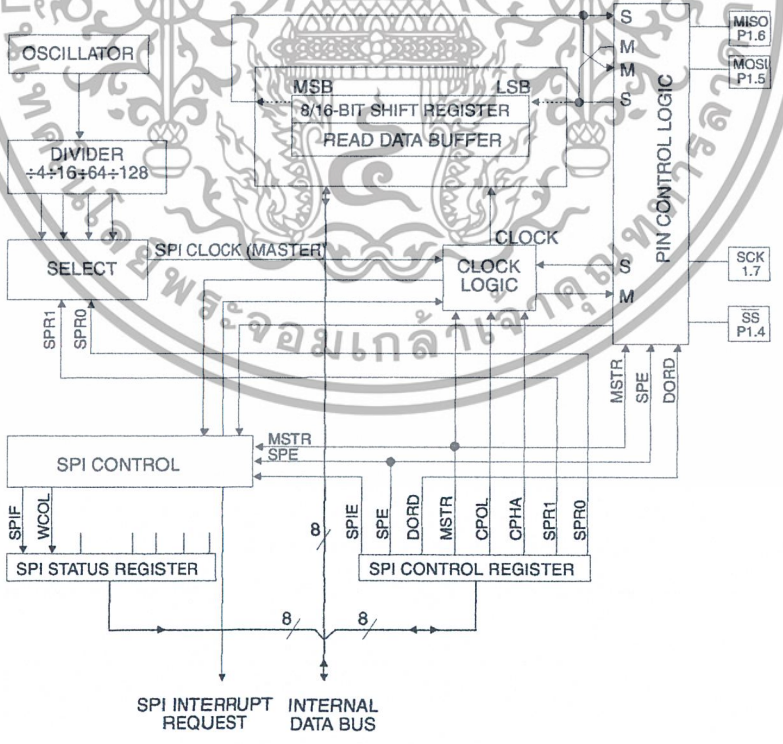



**Figure 6.** SPI Block Diagram



## UART

The UART in the AT89S8252 operates the same way as the UART in the AT89C51, AT89C52 and AT89C55. For further information, see the October 1995 Microcontroller Data Book, page 2-49, section titled, "Serial Interface."

## Serial Peripheral Interface

The serial peripheral interface (SPI) allows high-speed synchronous data transfer between the AT89S8252 and peripheral devices or between several AT89S8252 devices. The AT89S8252 SPI features include the following:

- Full-Duplex, 3-Wire Synchronous Data Transfer
- Master or Slave Operation
- 1.5 MHz Bit Frequency (max.)
- LSB First or MSB First Data Transfer
- Four Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wakeup from Idle Mode (Slave Mode Only)

The interconnection between master and slave CPUs with SPI is shown in the following figure. The SCK pin is the clock output in the master mode but is the clock input in the slave mode. Writing to the SPI data register of the master CPU starts the SPI clock generator, and the data written shifts out of the MOSI pin and into the MOSI pin of the slave CPU. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If both the SPI interrupt enable bit (SPIE) and the serial port interrupt enable bit (ES) are set, an interrupt is requested.

The Slave Select input, $\overline{SS}$/P1.4, is set low to select an individual SPI device as a slave. When $\overline{SS}$/P1.4 is set high, the SPI port is deactivated and the MOSI/P1.5 pin can be used as an input.

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 8 and Figure 9.

**Figure 7.** SPI Master-slave Interconnection

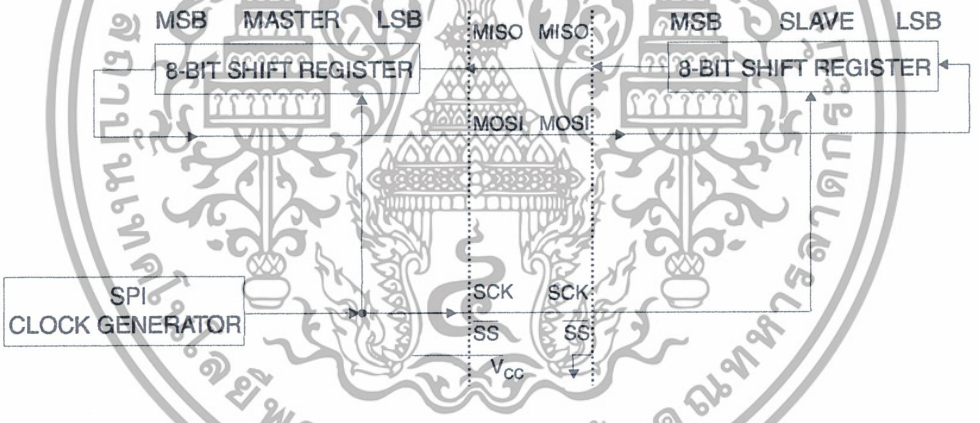
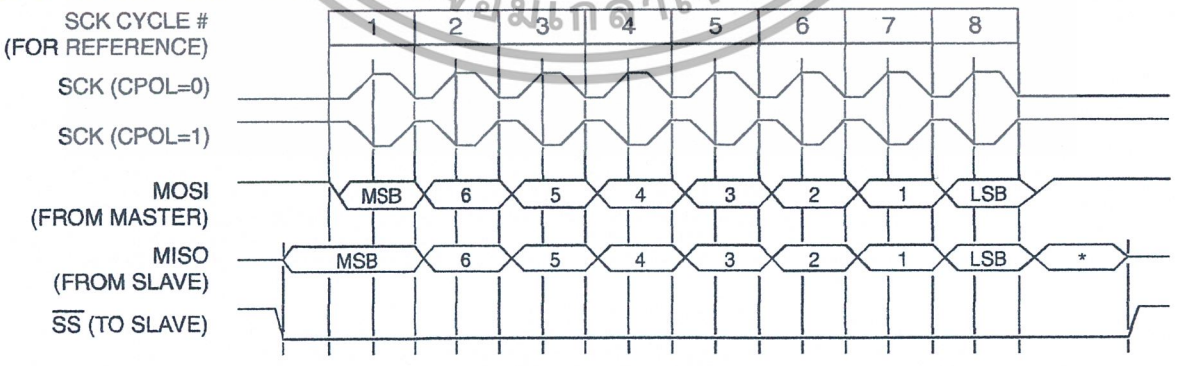
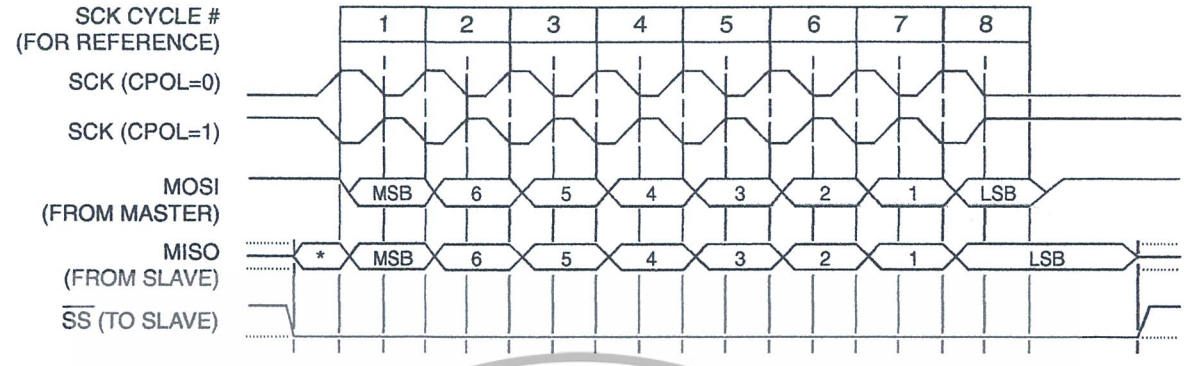



**Figure 8.** SPI transfer Format with CPHA = 0




*Not defined but normally MSB of character just received

**Figure 9.** SPI Transfer Format with CPHA = 1




*Not defined but normally LSB of previously transmitted character

## Interrupts

The AT89S8252 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 10 shows that bit position IE.6 is unimplemented. In the AT89C51, 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 10.** 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 | SPI and UART 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

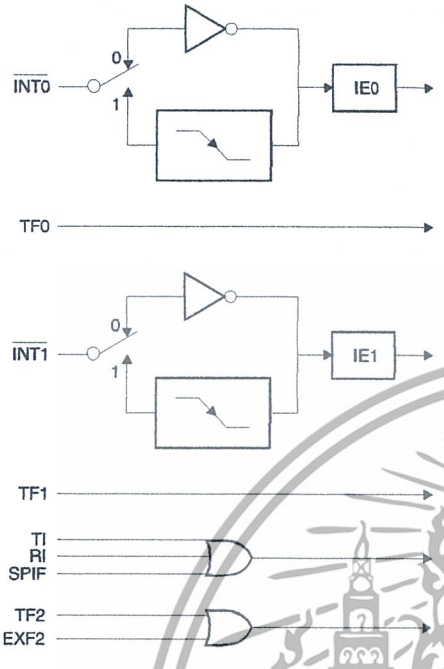
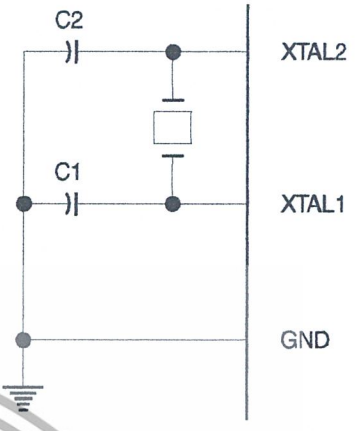
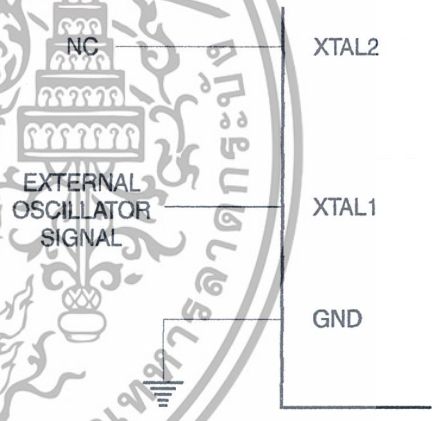



**Figure 11.** Oscillator Connections




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

**Figure 12.** External Clock Drive Configuration




## 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.

## 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 |

## 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 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.

To exit power-down via an interrupt, the external interrupt must be enabled as level sensitive before entering power-down. The interrupt service routine starts at 16 ms (nominal) after the enabled interrupt pin is activated.

## Program Memory Lock Bits

The AT89S8252 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.

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.

Once programmed, the lock bits can only be unprogrammed with the Chip Erase operations in either the parallel or serial modes.

## Lock Bit Protection Modes[1][2]

| Program Lock Bits | | | | |
|---|---|---|---|---|
| | LB1 | LB2 | LB3 | Protection Type |
| 1 | U | U | U | No internal memory lock feature. |
| 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 (parallel or serial mode) is disabled. |
| 3 | P | P | U | Same as Mode 2, but parallel or serial verify are also disabled. |
| 4 | P | P | P | Same as Mode 3, but external execution is also disabled. |

Notes: 1. U = Unprogrammed
2. P = Programmed

## Programming the Flash and EEPROM

Atmel's AT89S8252 Flash Microcontroller offers 8K bytes of in-system reprogrammable Flash Code memory and 2K bytes of EEPROM Data memory.

The AT89S8252 is normally shipped with the on-chip Flash Code and EEPROM Data memory arrays in the erased state (i.e. contents = FFH) and ready to be programmed. This device supports a High-voltage (12V) Parallel programming mode and a Low-voltage (5V) Serial programming mode. The serial programming mode provides a convenient way to download the AT89S8252 inside the user's system. The parallel programming mode is compatible with conventional third party Flash or EPROM programmers.

The Code and Data memory arrays are mapped via separate address spaces in the serial programming mode. In the parallel programming mode, the two arrays occupy one contiguous address space: 0000H to 1FFFH for the Code array and 2000H to 27FFH for the Data array.

The Code and Data memory arrays on the AT89S8252 are programmed byte-by-byte in either programming mode. An auto-erase cycle is provided with the self-timed programming operation in the serial programming mode. There is no need to perform the Chip Erase operation to reprogram any memory location in the serial programming mode unless any of the lock bits have been programmed.

In the parallel programming mode, there is no auto-erase cycle. To reprogram any non-blank byte, the user needs to use the Chip Erase operation first to erase both arrays.

**Parallel Programming Algorithm:** To program and verify the AT89S8252 in the parallel programming mode, the following sequence is recommended:

1. Power-up sequence:

   Apply power between $V_{CC}$ and GND pins.

   Set RST pin to "H".

   Apply a 3 MHz to 24 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
2. Set $\overline{PSEN}$ pin to "L"

   ALE pin to "H"

   $\overline{EA}$ pin to "H" and all other pins to "H".
3. Apply the appropriate combination of "H" or "L" logic levels to pins P2.6, P2.7, P3.6, P3.7 to select one of the programming operations shown in the Flash Programming Modes table.
4. Apply the desired byte address to pins P1.0 to P1.7 and P2.0 to P2.5.

   Apply data to pins P0.0 to P0.7 for Write Code operation.
5. Raise $\overline{EA}/V_{PP}$ to 12V to enable Flash programming, erase or verification.
6. Pulse ALE/$\overline{PROG}$ once to program a byte in the Code memory array, the Data memory array or the lock bits. The byte-write cycle is self-timed and typically takes 1.5 ms.
7. To verify the byte just programmed, bring pin P2.7 to "L" and read the programmed data at pins P0.0 to P0.7.
8. Repeat steps 3 through 7 changing the address and data for the entire 2K or 8K bytes array or until the end of the object file is reached.
9. Power-off sequence:

   Set XTAL1 to "L".

   Set RST and $\overline{EA}$ pins to "L".

   Turn $V_{CC}$ power off.

In the parallel programming mode, there is no auto-erase cycle and to reprogram any non-blank byte, the user needs to use the Chip Erase operation first to erase both arrays.

**Data Polling:** The AT89S8252 features $\overline{DATA}$ Polling to indicate the end of a write cycle. During a write cycle in the parallel or serial programming mode, an attempted read of the last byte written will result in the complement of the written datum on P0.7 (parallel mode), and on the MSB of the serial output byte on MISO (serial mode). Once the write cycle has been completed, true data are 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 in the parallel programming mode can also be monitored by the RDY/$\overline{BSY}$ output signal. Pin P3.4 is pulled Low after ALE goes High during programming to indicate $\overline{BUSY}$. P3.4 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 or Data byte can be read back via the address and data lines for verification. The state of the lock bits can also be verified directly in the parallel programming mode. In the serial programming mode, the state of the lock bits can only be verified indirectly by observing that the lock bit features are enabled.

**Chip Erase:** Both Flash and EEPROM arrays are erased electrically at the same time. In the parallel programming mode, chip erase is initiated by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The Code and Data arrays are written with all "1"s in the Chip Erase operation.

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 16 ms.

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

**Serial Programming Fuse:** A programmable fuse is available to disable Serial Programming if the user needs maximum system security. The Serial Programming Fuse can only be programmed or erased in the Parallel Programming Mode.

*The AT89S8252 is shipped with the Serial Programming Mode enabled.*

**Reading the Signature Bytes:** The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows:

(030H) = 1EH indicates manufactured by Atmel
(031H) = 72H indicates 89S8252

## Programming Interface

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

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

## Serial Downloading

Both the Code and Data memory arrays can be programmed using the serial SPI bus 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 program/erase operations can be executed.

An auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction unless any of the lock bits have been programmed. The Chip Erase operation turns the content of every memory location in both the Code and Data arrays into FFH.

The Code and Data memory arrays have separate address spaces:

0000H to 1FFFH for Code memory and 000H to 7FFH for Data memory.

Either an external system clock is 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 1/40 of the crystal frequency. With a 24 MHz oscillator clock, the maximum SCK frequency is 600 kHz.

## Serial Programming Algorithm

To program and verify the AT89S8252 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 24 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 40.
3. The Code or Data array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. The selected memory location is first automatically erased before new data is written. The write cycle is self-timed and typically takes less than 2.5 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 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.

## Serial Programming Instruction

The Instruction Set for Serial Programming follows a 3-byte protocol and is shown in the following table:

## Instruction Set

| Instruction | Input Format | | | Operation |
|---|---|---|---|---|
| | Byte 1 | Byte 2 | Byte 3 | |
| Programming Enable | 1010 1100 | 0101 0011 | xxxx xxxx | Enable serial programming interface after RST goes high. |
| Chip Erase | 1010 1100 | xxxx x100 | xxxx xxxx | Chip erase both 8K & 2K memory arrays. |
| Read Code Memory | aaaa a001 | low addr | xxxx xxxx | Read data from Code memory array at the selected address. The 5 MSBs of the first byte are the high order address bits. The low order address bits are in the second byte. Data are available at pin MISO during the third byte. |
| Write Code Memory | aaaa a010 | low addr | data in | Write data to Code memory location at selected address. The address bits are the 5 MSBs of the first byte together with the second byte. |
| Read Data Memory | 00aa a101 | low addr | xxxx xxxx | Read data from Data memory array at selected address. Data are available at pin MISO during the third byte. |
| Write Data Memory | 00aa a110 | low addr | data in | Write data to Data memory location at selected address. |
| Write Lock Bits | 1010 1100 | x x111 | xxxx xxxx | Write lock bits.<br>Set LB1, LB2 or LB3 = "0" to program lock bits. |

Note:
1. DATA polling is used to indicate the end of a write cycle which typically takes less than 2.5 ms at 5V.
2. "aaaaa" = high order address.
3. "x" = don't care.

## Flash and EEPROM Parallel Programming Modes

| Mode | RST | $\overline{PSEN}$ | ALE/$\overline{PROG}$ | $\overline{EA}/V_{PP}$ | P2.6 | P2.7 | P3.6 | P3.7 | Data I/O P0.7:0 | Address P2.5:0 P1.7:0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Serial Prog. Modes | H | h(1) | h(1) | x | | | | | | |
| Chip Erase | H | L | ⌒ (2) | 12V | H | L | L | L | X | X |
| Write (10K bytes) Memory | H | L | ⌒ | 12V | L | H | H | H | DIN | ADDR |
| Read (10K bytes) Memory | H | L | H | 12V | L | L | H | H | DOUT | ADDR |
| Write Lock Bits: | H | L | ⌒ | 12V | H | L | H | L | DIN | X |
| Bit - 1 | | | | | | | | | P0.7 = 0 | X |
| Bit - 2 | | | | | | | | | P0.6 = 0 | X |
| Bit - 3 | | | | | | | | | P0.5 = 0 | X |
| Read Lock Bits: | H | L | H | 12V | H | H | L | L | DOUT | X |
| Bit - 1 | | | | | | | | | @P0.2 | X |
| Bit - 2 | | | | | | | | | @P0.1 | X |
| Bit - 3 | | | | | | | | | @P0.0 | X |
| Read Atmel Code | H | L | H | 12V | L | L | L | L | DOUT | 30H |
| Read Device Code | H | L | H | 12V | L | L | L | L | DOUT | 31H |
| Serial Prog. Enable | H | L | ⌒ (2) | 12V | L | H | L | H | P0.0 = 0 | X |
| Serial Prog. Disable | H | L | ⌒ (2) | 12V | L | H | L | H | P0.0 = 1 | X |
| Read Serial Prog. Fuse | H | L | H | 12V | H | H | L | H | @P0.0 | X |

Notes:
1. "h" = weakly pulled "High" internally.
2. Chip Erase and Serial Programming Fuse require a 10 ms $\overline{PROG}$ pulse. Chip Erase needs to be performed first before reprogramming any byte with a content other than FFH.
3. P3.4 is pulled Low during programming to indicate RDY/$\overline{BSY}$.
4. "X" = don't care

**Figure 13.** Programming the Flash/EEPROM Memory

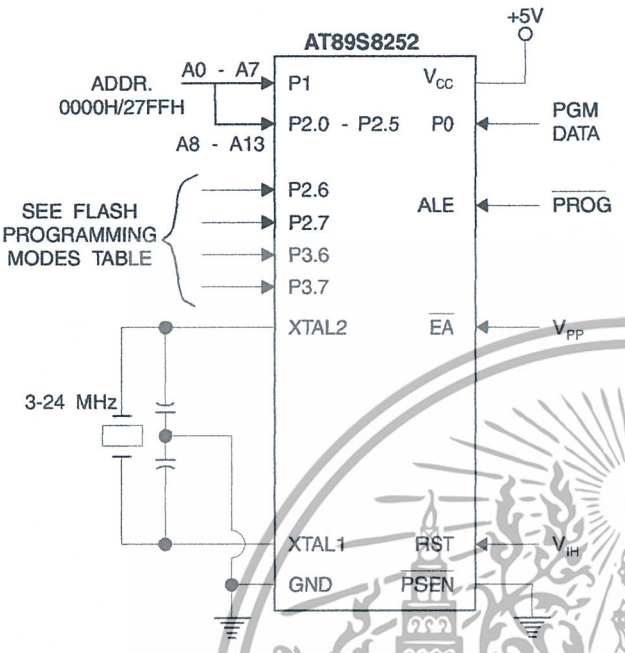



**Figure 14.** Verifying the Flash/EEPROM Memory

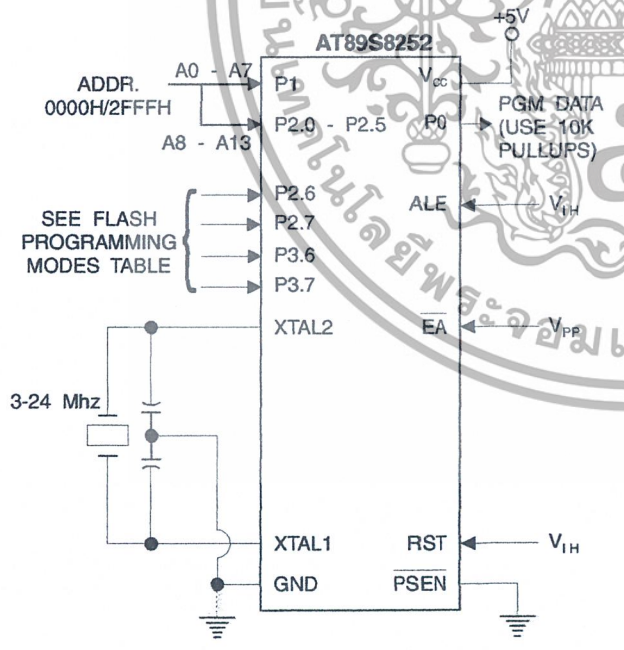
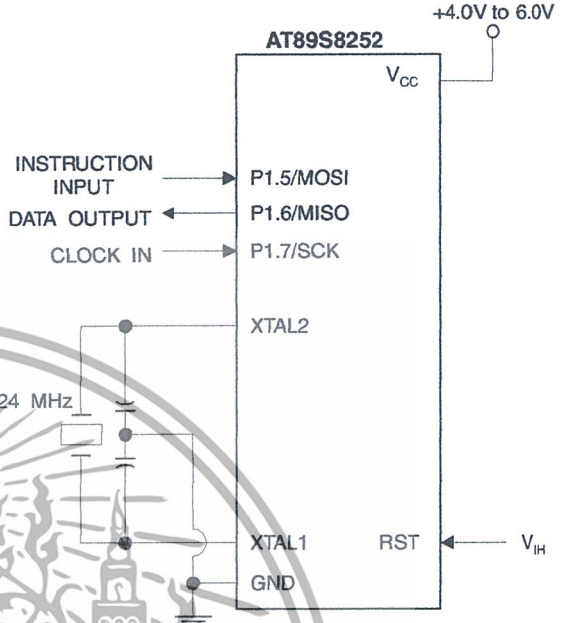



**Figure 15.** Flash/EEPROM Serial Downloading

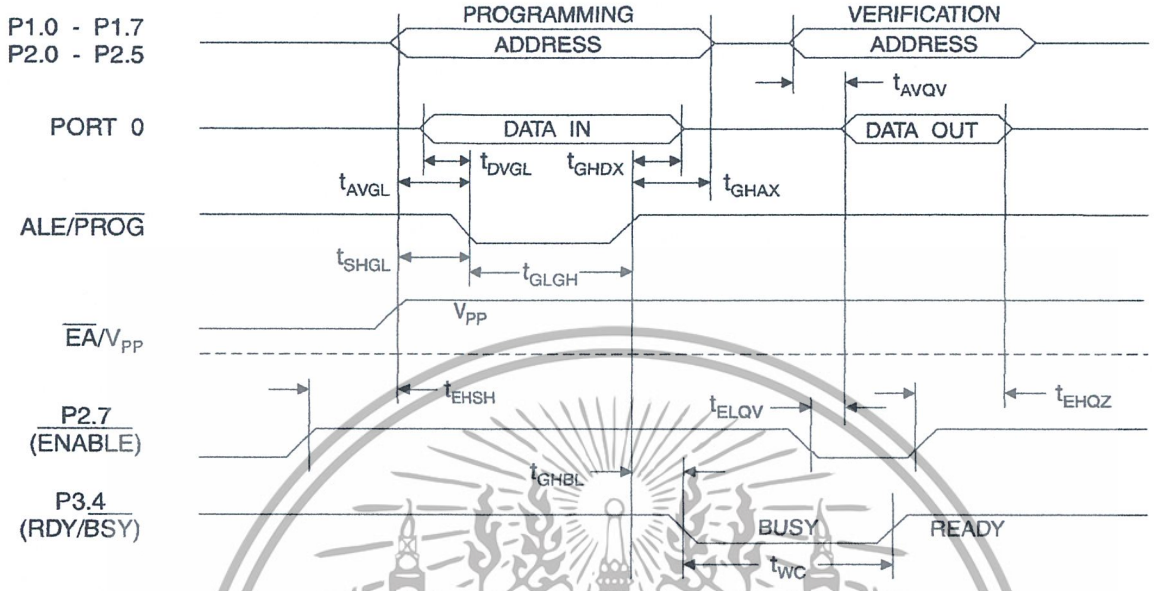


## Flash Programming and Verification Characteristics – Parallel Mode

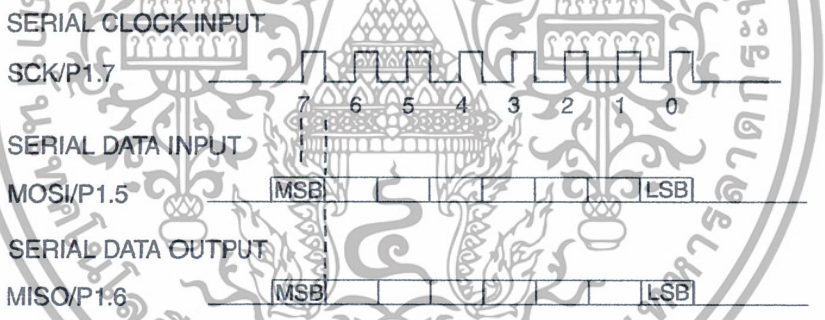
$T_A$ = 0°C to 70°C, $V_{CC}$ = 5.0V ± 10%

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $V_{PP}$ | Programming Enable Voltage | 11.5 | 12.5 | V |
| $I_{PP}$ | Programming Enable Current | | 1.0 | mA |
| $1/t_{CLCL}$ | Oscillator Frequency | 3 | 24 | 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 |
| $t_{GLGH}$ | $\overline{PROG}$ Width | 1 | 110 | 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 | | 2.0 | ms |

## Flash/EEPROM Programming and Verification Waveforms – Parallel Mode




## Serial Downloading Waveforms



## Absolute Maximum Ratings*

| | |
|---|---|
| Operating Temperature | -55°C to +125°C |
| Storage Temperature | -65°C to +150°C |
| Voltage on Any Pin with Respect to Ground | -1.0V to +7.0V |
| Maximum Operating Voltage | 6.6V |
| DC Output Current | 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 $V_{CC}$ = 5.0V ± 20%, 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 Hifh-voltage | (Except XTAL1, RST) | 0.2 $V_{CC}$ + 0.9 | $V_{CC}$ + 0.5 | V |
| $V_{IH1}$ | Input Hifh-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 | | 0.5 | V |
| $V_{OL1}$ | Output Low-voltage (1) (Port 0, ALE, $\overline{PSEN}$) | $I_{OL}$ = 3.2 mA | | 0.5 | V |
| $V_{OH}$ | Output Hifh-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 Hifh-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 Pull-down Resistor | | 50 | 300 | KΩ |
| $C_{IO}$ | Pin Capacitance | Test Freq. = 1 MHz, $T_A$ = 25°C | | 10 | pF |
| $I_{CC}$ | Power Supply Current | Active Mode, 12 MHz | | 25 | mA |
| | | Idle Mode, 12 MHz | | 6.5 | mA |
| | Power-down Mode (2) | $V_{CC}$ = 6V | | 100 | µA |
| | | $V_{CC}$ = 3V | | 40 | 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

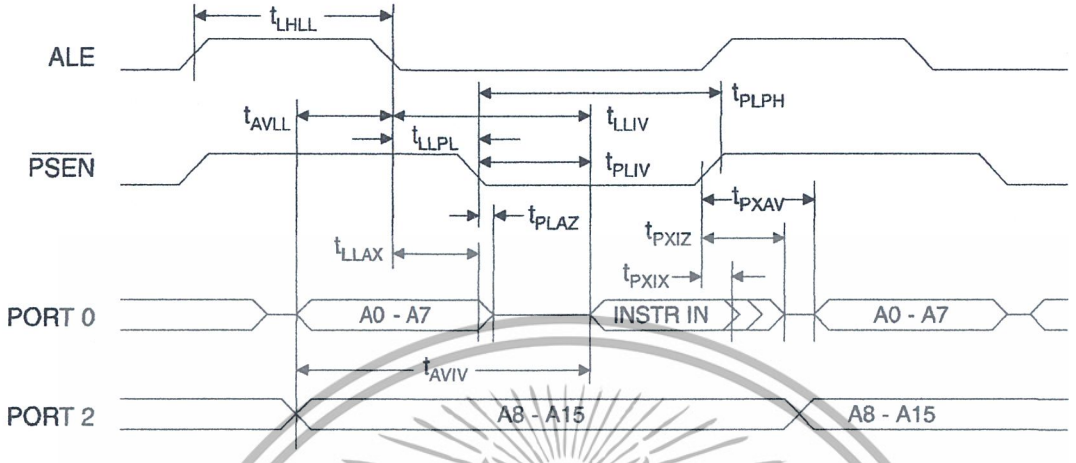
## AC Characteristics

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

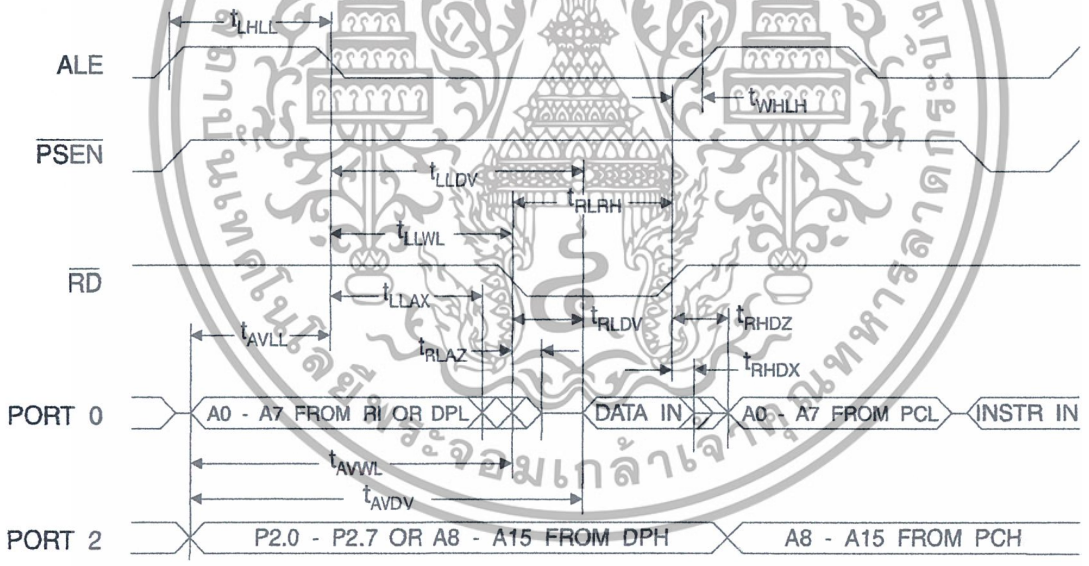
## External Program and Data Memory Characteristics

| Symbol | Parameter | Variable Oscillator | | Units |
|---|---|---|---|---|
| | | Min | Max | |
| $1/t_{CLCL}$ | Oscillator Frequency | 0 | 24 | MHz |
| $t_{LHLL}$ | ALE Pulse Width | $2t_{CLCL} - 40$ | | ns |
| $t_{AVLL}$ | Address Valid to ALE Low | $t_{CLCL} - 13$ | | ns |
| $t_{LLAX}$ | Address Hold after ALE Low | $t_{CLCL} - 20$ | | ns |
| $t_{LLIV}$ | ALE Low to Valid Instruction In | | $4t_{CLCL} - 65$ | ns |
| $t_{LLPL}$ | ALE Low to $\overline{PSEN}$ Low | $t_{CLCL} - 13$ | | ns |
| $t_{PLPH}$ | $\overline{PSEN}$ Pulse Width | $3t_{CLCL} - 20$ | | ns |
| $t_{PLIV}$ | $\overline{PSEN}$ Low to Valid Instruction In | | $3t_{CLCL} - 45$ | ns |
| $t_{PXIX}$ | Input Instruction Hold after $\overline{PSEN}$ | 0 | | ns |
| $t_{PXIZ}$ | Input Instruction Float after $\overline{PSEN}$ | | $t_{CLCL} - 10$ | ns |
| $t_{PXAV}$ | $\overline{PSEN}$ to Address Valid | $t_{CLCL} - 8$ | | ns |
| $t_{AVIV}$ | Address to Valid Instruction In | | $5t_{CLCL} - 55$ | ns |
| $t_{PLAZ}$ | $\overline{PSEN}$ Low to Address Float | | 10 | ns |
| $t_{RLRH}$ | $\overline{RD}$ Pulse Width | $6t_{CLCL} - 100$ | | ns |
| $t_{WLWH}$ | $\overline{WR}$ Pulse Width | $6t_{CLCL} - 100$ | | ns |
| $t_{RLDV}$ | $\overline{RD}$ Low to Valid Data In | | $5t_{CLCL} - 90$ | ns |
| $t_{RHDX}$ | Data Hold after $\overline{RD}$ | 0 | | ns |
| $t_{RHDZ}$ | Data Float after $\overline{RD}$ | | $2t_{CLCL} - 28$ | ns |
| $t_{LLDV}$ | ALE Low to Valid Data In | | $8t_{CLCL} - 150$ | ns |
| $t_{AVDV}$ | Address to Valid Data In | | $9t_{CLCL} - 165$ | ns |
| $t_{LLWL}$ | ALE Low to $\overline{RD}$ or $\overline{WR}$ Low | $3t_{CLCL} - 50$ | $3t_{CLCL} + 50$ | ns |
| $t_{AVWL}$ | Address to $\overline{RD}$ or $\overline{WR}$ Low | $4t_{CLCL} - 75$ | | ns |
| $t_{QVWX}$ | Data Valid to $\overline{WR}$ Transition | $t_{CLCL} - 20$ | | ns |
| $t_{QVWH}$ | Data Valid to $\overline{WR}$ High | $7t_{CLCL} - 120$ | | ns |
| $t_{WHQX}$ | Data Hold after $\overline{WR}$ | $t_{CLCL} - 20$ | | ns |
| $t_{RLAZ}$ | $\overline{RD}$ Low to Address Float | | 0 | ns |
| $t_{WHLH}$ | $\overline{RD}$ or $\overline{WR}$ High to ALE High | $t_{CLCL} - 20$ | $t_{CLCL} + 25$ | ns |

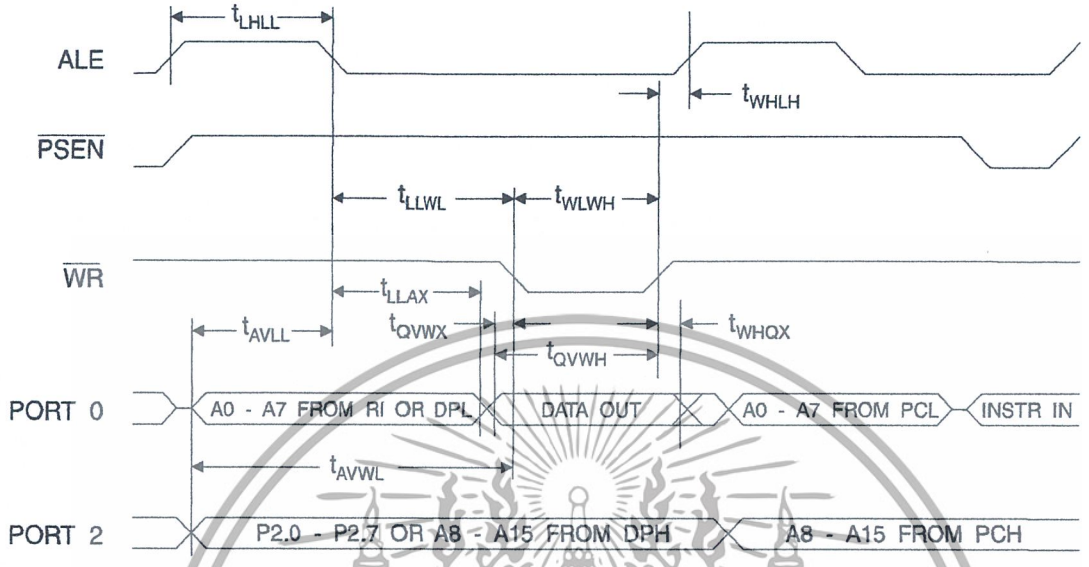
## External Program Memory Read Cycle



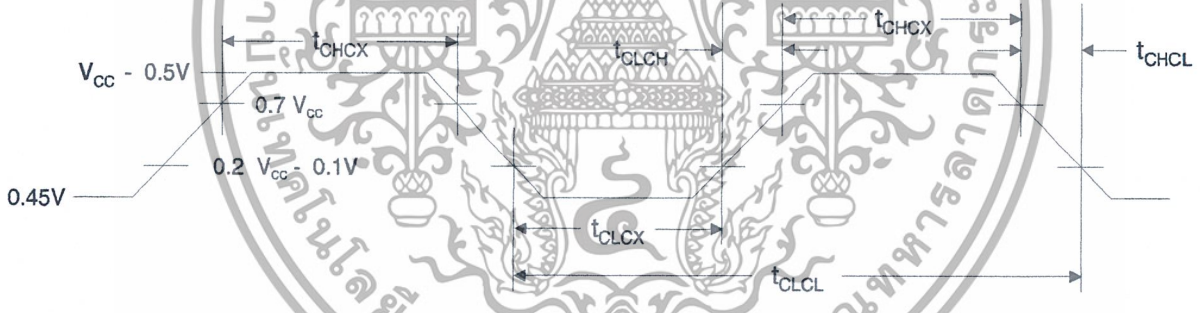

## External Data Memory Read Cycle



## External Data Memory Write Cycle




## External Clock Drive Waveforms




## External Clock Drive

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

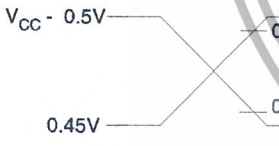
## Serial Port Timing: Shift Register Mode Test Conditions

The values in this table are valid for $V_{CC}$ = 4.0V to 6V and Load Capacitance = 80 pF.

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

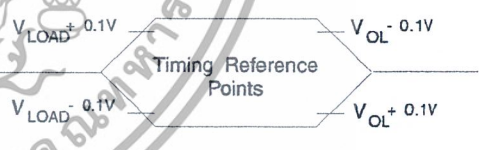
## Shift Register Mode Timing Waveforms


## AC Testing Input/Output Waveforms[1]




Notes: 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.

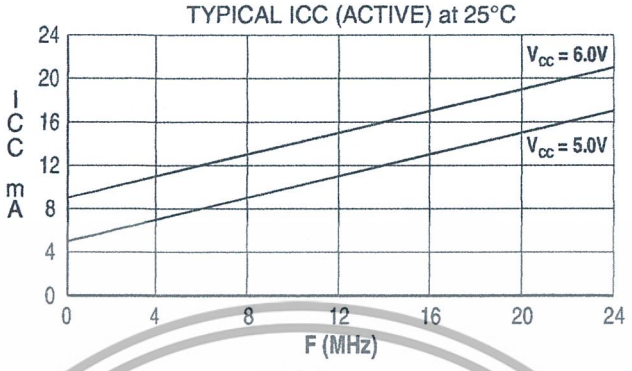
## Float Waveforms[1]




Notes: 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.

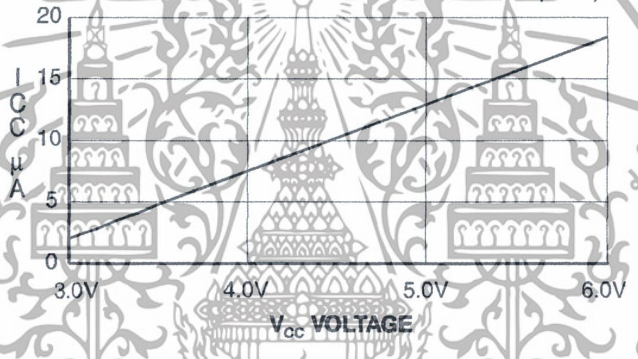
**AT89S8252**

TYPICAL ICC (ACTIVE) at 25°C



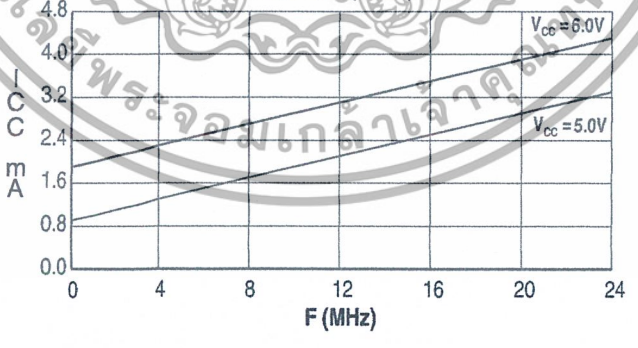

**AT89S8252**

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




**AT89S8252**

TYPICAL ICC (IDLE) at 25°C




Notes: 1. XTAL1 tied to GND for Icc (power-down)
2. Lock bits programmed

## Ordering Information

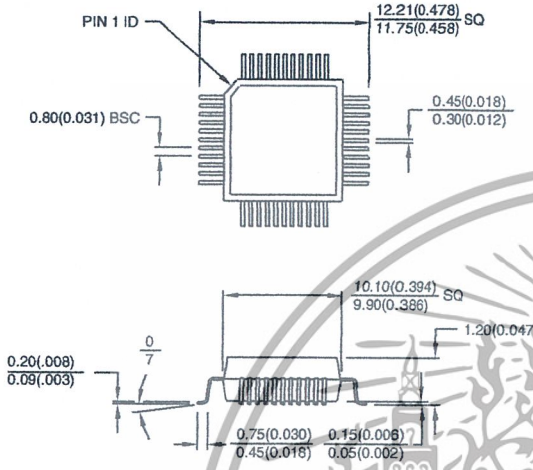
| Speed (MHz) | Power Supply | Ordering Code | Package | Operation Range |
|---|---|---|---|---|
| 24 | 4.0V to 6.0V | AT89S8252-24AC<br>AT89S8252-24JC<br>AT89S8252-24PC<br>AT89S8252-24QC | 44A<br>44J<br>40P6<br>44Q | Commercial<br>(0°C to 70°C) |
| | 4.0V to 6.0V | AT89S8252-24AI<br>AT89S8252-24JI<br>AT89S8252-24PI<br>AT89S8252-24QI | 44A<br>44J<br>40P6<br>44Q | Industrial<br>(-40°C to 85°C) |
| 33 | 4.5V to 5.5V | AT89S8252-33AC<br>AT89S8252-33JC<br>AT89S8252-33PC<br>AT89S8252-33QC | 44A<br>44J<br>40P6<br>44Q | Commercial<br>(0°C to 70°C) |

= Preliminary Information

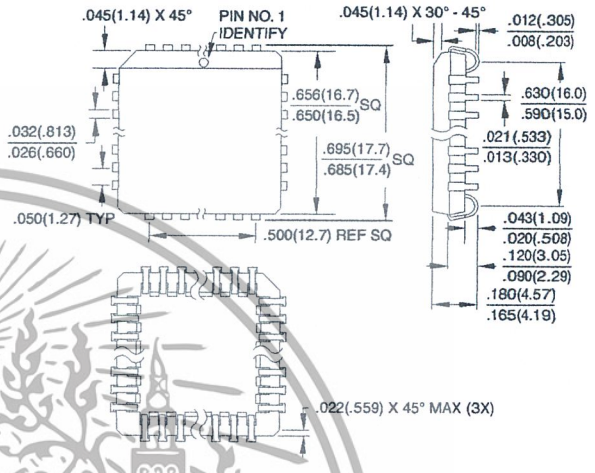
| Package Type | |
|---|---|
| 44A | 44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP) |
| 44J | 44-lead, Plastic J-leaded Chip Carrier (PLCC) |
| 40P6 | 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP) |
| 44Q | 44-lead, Plastic Gull Wing Quad Flatpack (PQFP) |

## Packaging Information

**44A,** 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flatpack (TQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-026 ACB

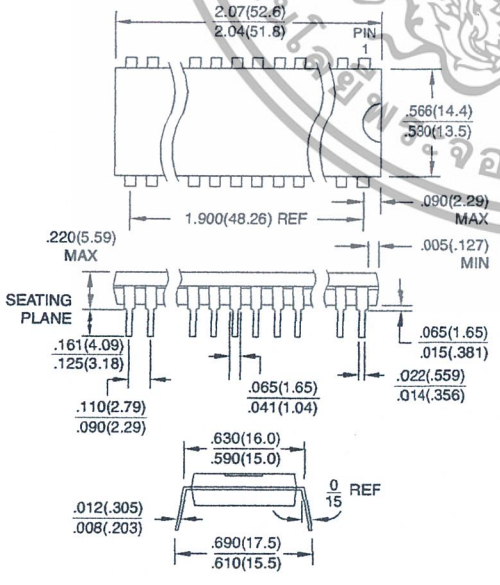



**44J,** 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-018 AC

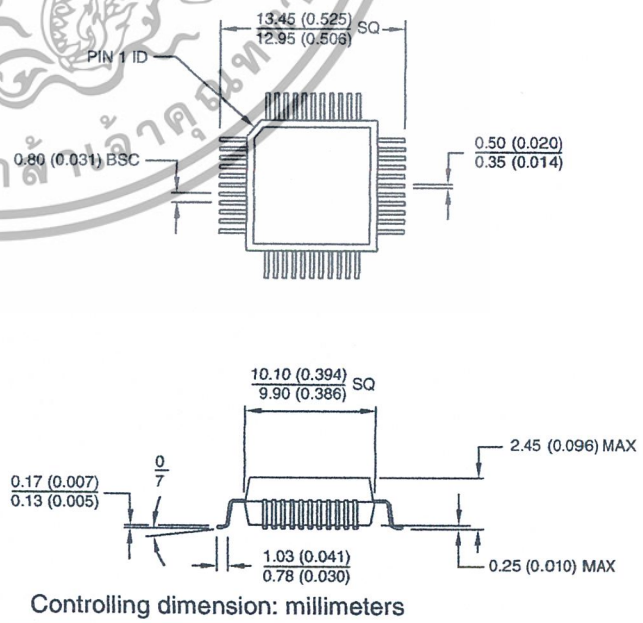



Controlling dimension: millimeters

**40P6,** 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)




**44Q,** 44-lead, Plastic Quad Flat Package (PQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-022 AB




Controlling dimension: millimeters

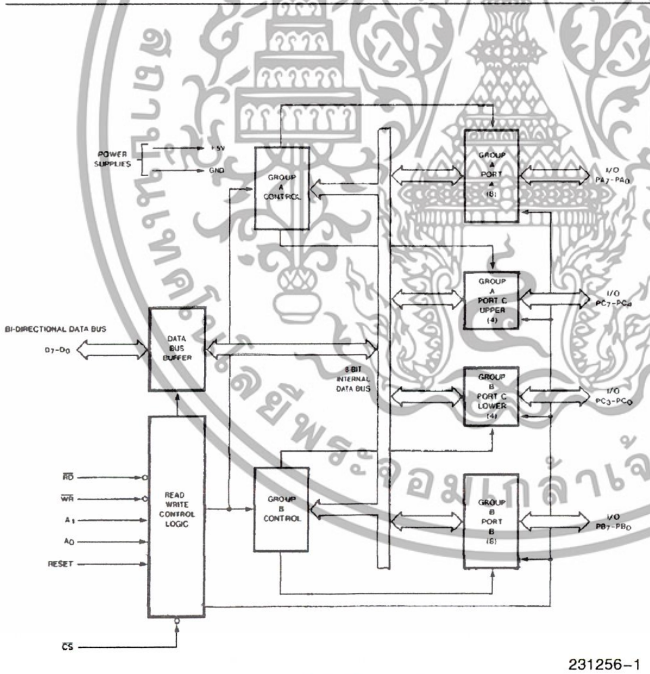
# 82C55A
# CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- **Compatible with all Intel and Most Other Microprocessors**
- **High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188**
- **24 Programmable I/O Pins**
- **Low Power CHMOS**
- **Completely TTL Compatible**
- **Control Word Read-Back Capability**
- **Direct Bit Set/Reset Capability**
- **2.5 mA DC Drive Capability on all I/O Port Outputs**
- **Available in 40-Pin DIP and 44-Pin PLCC**
- **Available in EXPRESS**
  - **— Standard Temperature Range**
  - **— Extended Temperature Range**

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

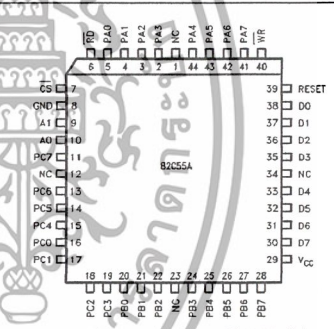
In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.

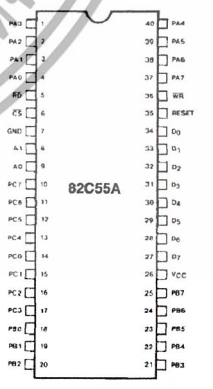



231256-1

**Figure 1. 82C55A Block Diagram**




231256-31




231256-2

**Figure 2. 82C55A Pinout**

Diagrams are for pin reference only. Package sizes are not to scale.

October 1995

Order Number: 231256-004

**Table 1. Pin Description**

| Symbol | Pin Number Dip | Pin Number PLCC | Type | Name and Function |
|---|---|---|---|---|
| $PA_{3-0}$ | 1–4 | 2–5 | I/O | **PORT A, PINS 0–3:** Lower nibble of an 8-bit data output latch/ buffer and an 8-bit data input latch. |
| $\overline{RD}$ | 5 | 6 | I | **READ CONTROL:** This input is low during CPU read operations. |
| $\overline{CS}$ | 6 | 7 | I | **CHIP SELECT:** A low on this input enables the 82C55A to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and WR are ignored otherwise. |
| GND | 7 | 8 | | **System Ground** |
| $A_{1-0}$ | 8–9 | 9–10 | I | **ADDRESS:** These input signals, in conjunction $\overline{RD}$ and $\overline{WR}$, control the selection of one of the three ports or the control word registers. |
| $PC_{7-4}$ | 10–13 | 11,13–15 | I/O | **PORT C, PINS 4–7:** Upper nibble of an 8-bit data output latch/ buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. |
| $PC_{0-3}$ | 14–17 | 16–19 | I/O | **PORT C, PINS 0–3:** Lower nibble of Port C. |
| $PB_{0-7}$ | 18–25 | 20–22, 24–28 | I/O | **PORT B, PINS 0–7:** An 8-bit data output latch/buffer and an 8-bit data input buffer. |
| $V_{CC}$ | 26 | 29 | | **SYSTEM POWER:** + 5V Power Supply. |
| $D_{7-0}$ | 27–34 | 30–33, 35–38 | I/O | **DATA BUS:** Bi-directional, tri-state data bus lines, connected to system data bus. |
| RESET | 35 | 39 | I | **RESET:** A high on this input clears the control register and all ports are set to the input mode. |
| $\overline{WR}$ | 36 | 40 | I | **WRITE CONTROL:** This input is low during CPU write operations. |
| $PA_{7-4}$ | 37–40 | 41–44 | I/O | **PORT A, PINS 4–7:** Upper nibble of an 8-bit data output latch/ buffer and an 8-bit data input latch. |
| NC | | 1, 12, 23, 34 | | No Connect |

| $A_1$ | $A_0$ | $\overline{RD}$ | $\overline{WR}$ | $\overline{CS}$ | Input Operation (Read) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | Port A - Data Bus |
| 0 | 1 | 0 | 1 | 0 | Port B - Data Bus |
| 1 | 0 | 0 | 1 | 0 | Port C - Data Bus |
| 1 | 1 | 0 | 1 | 0 | Control Word - Data Bus |
| | | | | | **Output Operation (Write)** |
| 0 | 0 | 1 | 0 | 0 | Data Bus - Port A |
| 0 | 1 | 1 | 0 | 0 | Data Bus - Port B |
| 1 | 0 | 1 | 0 | 0 | Data Bus - Port C |
| 1 | 1 | 1 | 0 | 0 | Data Bus - Control |
| | | | | | **Disable Function** |
| X | X | X | X | 1 | Data Bus - 3 - State |
| X | X | 1 | 1 | 0 | Data Bus - 3 - State |

## 82C55A FUNCTIONAL DESCRIPTION

### General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

### Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

### Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

### Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7–C4)
Control Group B - Port B and Port C lower (C3–C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

### Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

**Port A.** One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

**Port B.** One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

**Port C.** One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.

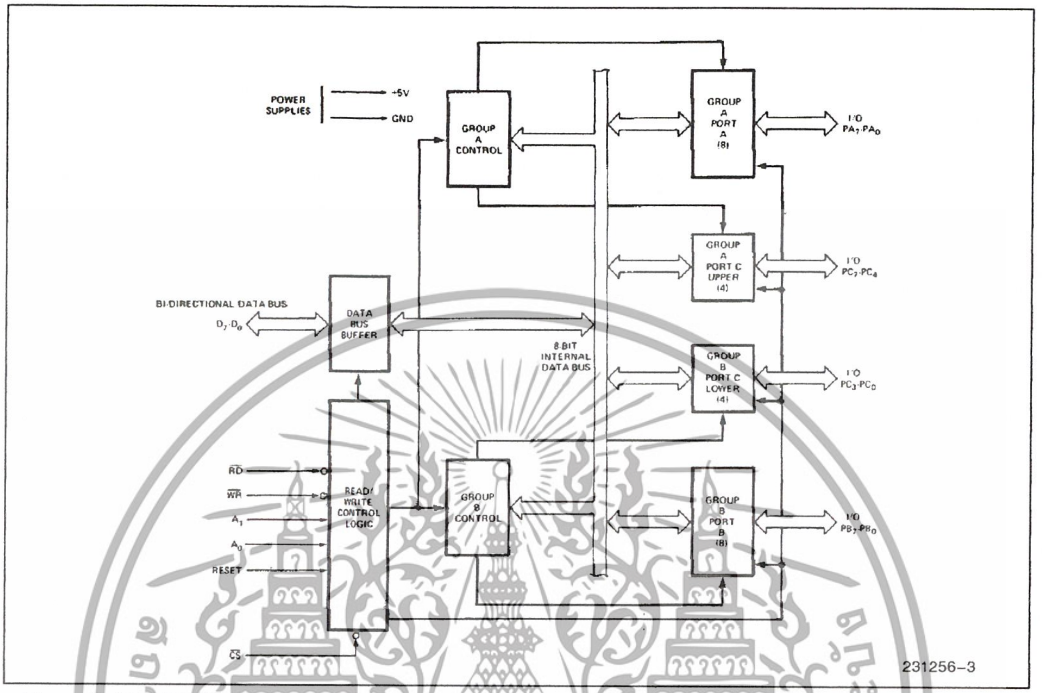
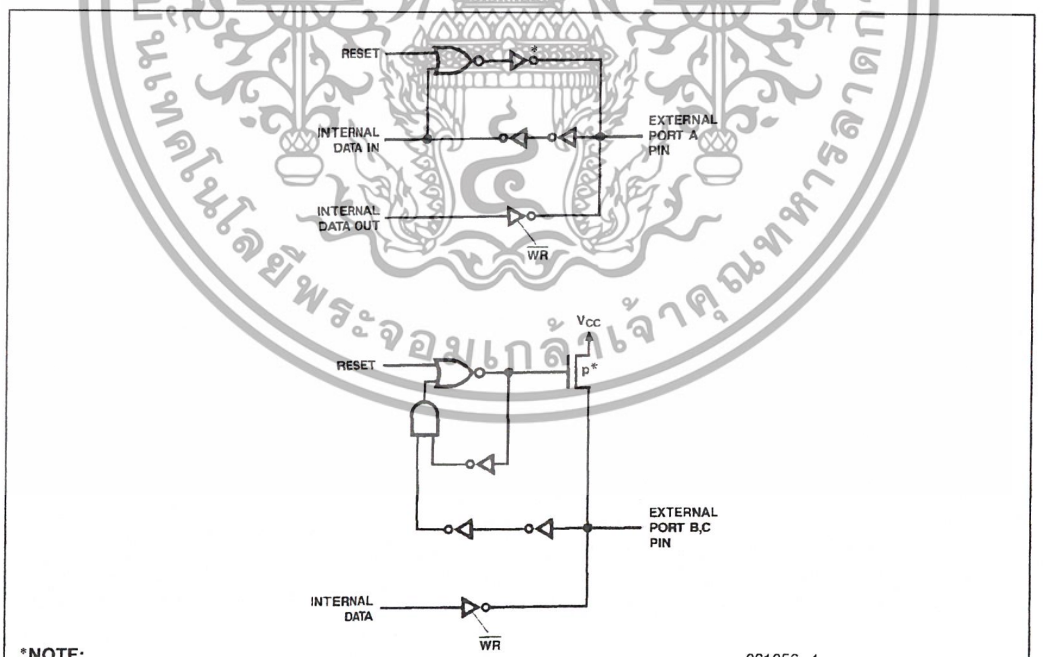



**Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions**




***NOTE:**
Port pins loaded with more than 20 pF capacitance may not have their logic level guaranteed following a hardware reset.

**Figure 4. Port A, B, C, Bus-hold Configuration**

## 82C55A OPERATIONAL DESCRIPTION

### Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 — Basic input/output
Mode 1 — Strobed Input/output
Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

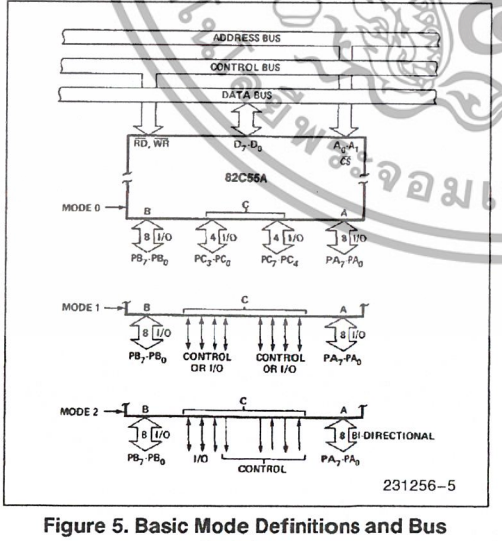



**Figure 5. Basic Mode Definitions and Bus Interface**

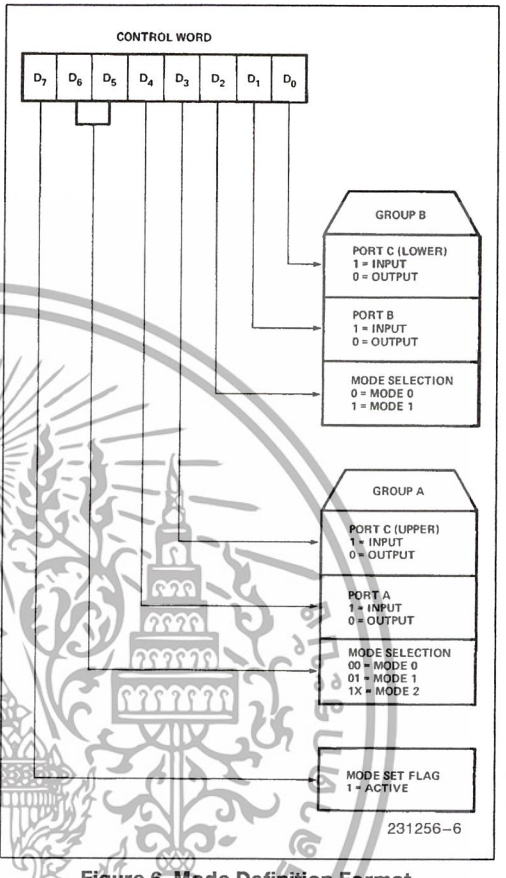



**Figure 6. Mode Definition Format**

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

### Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

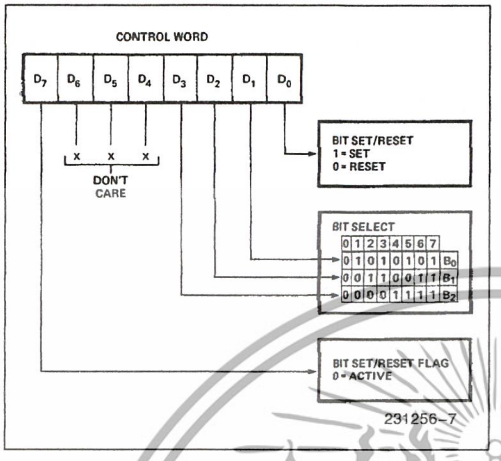



**Figure 7. Bit Set/Reset Format**

**Interrupt Control Functions**

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is SET—Interrupt enable
(BIT-RESET)—INTE is RESET—Interrupt disable

**Note:**
All Mask flip-flops are automatically reset during mode selection and device Reset.

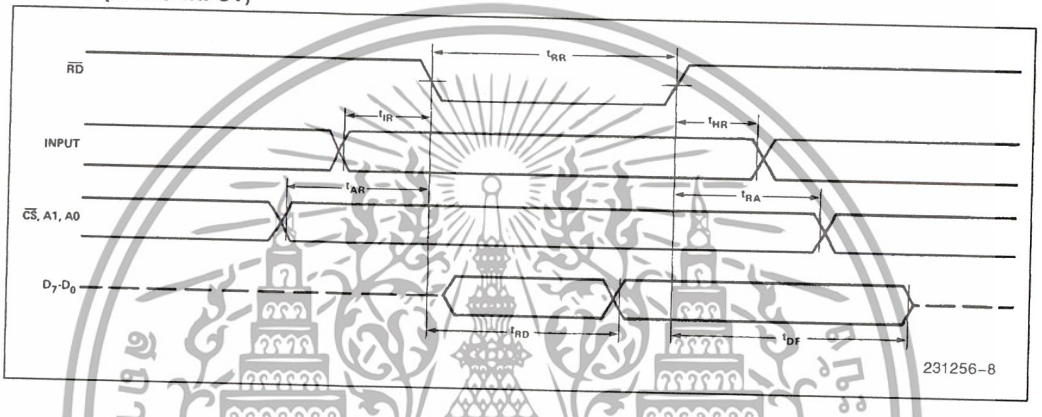
### Operating Modes

**Mode 0 (Basic Input/Output).** This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)

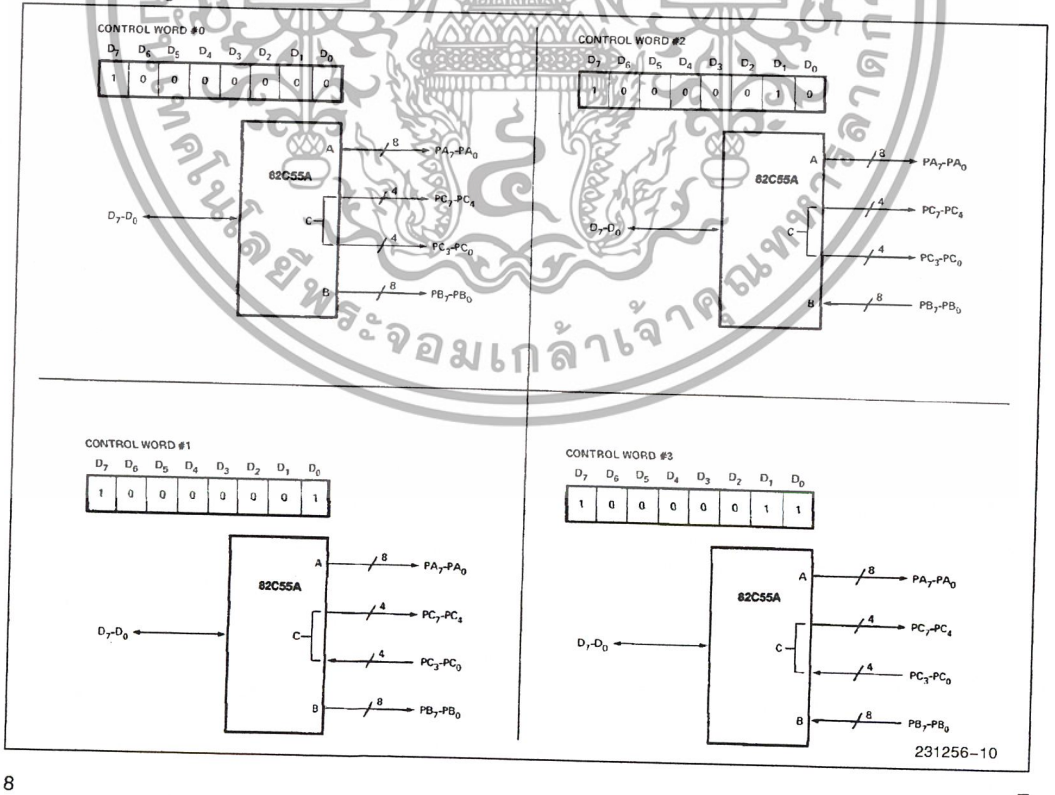



MODE 0 (BASIC OUTPUT)

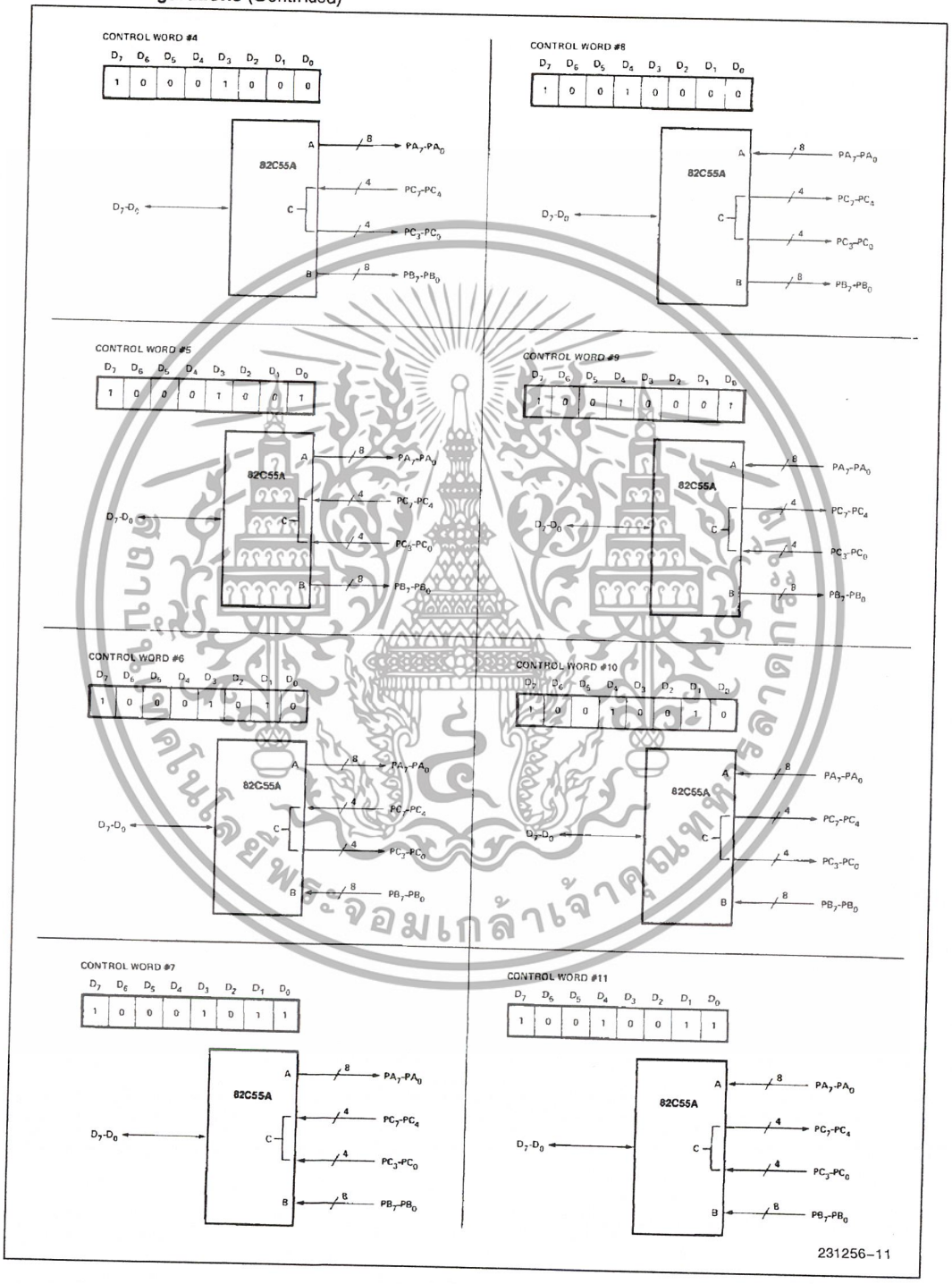
**MODE 0 Port Definition**

| A | | B | | GROUP A | | | GROUP B | |
|---|---|---|---|---|---|---|---|---|
| $D_4$ | $D_3$ | $D_1$ | $D_0$ | PORT A | PORT C (UPPER) | # | PORT B | PORT C (LOWER) |
| 0 | 0 | 0 | 0 | OUTPUT | OUTPUT | 0 | OUTPUT | OUTPUT |
| 0 | 0 | 0 | 1 | OUTPUT | OUTPUT | 1 | OUTPUT | INPUT |
| 0 | 0 | 1 | 0 | OUTPUT | OUTPUT | 2 | INPUT | OUTPUT |
| 0 | 0 | 1 | 1 | OUTPUT | OUTPUT | 3 | INPUT | INPUT |
| 0 | 1 | 0 | 0 | OUTPUT | INPUT | 4 | OUTPUT | OUTPUT |
| 0 | 1 | 0 | 1 | OUTPUT | INPUT | 5 | OUTPUT | INPUT |
| 0 | 1 | 1 | 0 | OUTPUT | INPUT | 6 | INPUT | OUTPUT |
| 0 | 1 | 1 | 1 | OUTPUT | INPUT | 7 | INPUT | INPUT |
| 1 | 0 | 0 | 0 | INPUT | OUTPUT | 8 | OUTPUT | OUTPUT |
| 1 | 0 | 0 | 1 | INPUT | OUTPUT | 9 | OUTPUT | INPUT |
| 1 | 0 | 1 | 0 | INPUT | OUTPUT | 10 | INPUT | OUTPUT |
| 1 | 0 | 1 | 1 | INPUT | OUTPUT | 11 | INPUT | INPUT |
| 1 | 1 | 0 | 0 | INPUT | INPUT | 12 | OUTPUT | OUTPUT |
| 1 | 1 | 0 | 1 | INPUT | INPUT | 13 | OUTPUT | INPUT |
| 1 | 1 | 1 | 0 | INPUT | INPUT | 14 | INPUT | OUTPUT |
| 1 | 1 | 1 | 1 | INPUT | INPUT | 15 | INPUT | INPUT |

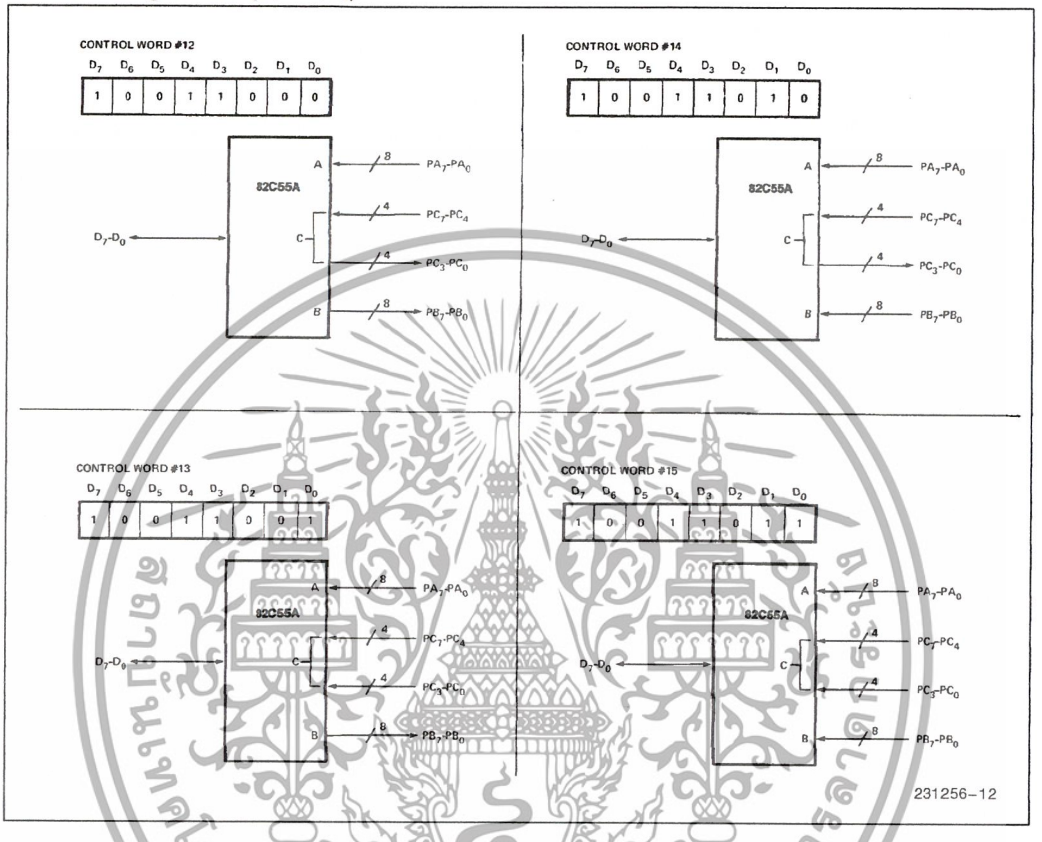
**MODE 0 Configurations**



**MODE 0 Configurations** (Continued)



**MODE 0 Configurations** (Continued)




## Operating Modes

**MODE 1 (Strobed Input/Output).** This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic functional Definitions:

- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

### Input Control Signal Definition

**$\overline{STB}$ (Strobe Input).** A "low" on this input loads data into the input latch.

### IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by $\overline{STB}$ input being low and is reset by the rising edge of the $\overline{RD}$ input.

### INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

**INTE A**

Controlled by bit set/reset of $PC_4$.

**INTE B**

Controlled by bit set/reset of $PC_2$.

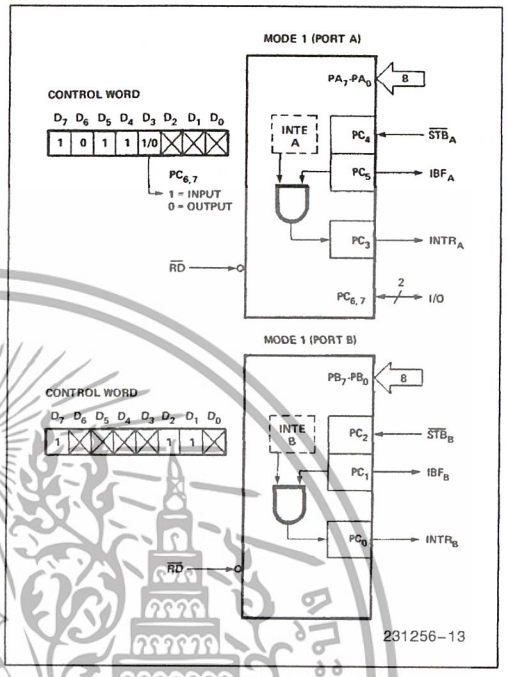



Figure 8. MODE 1 Input

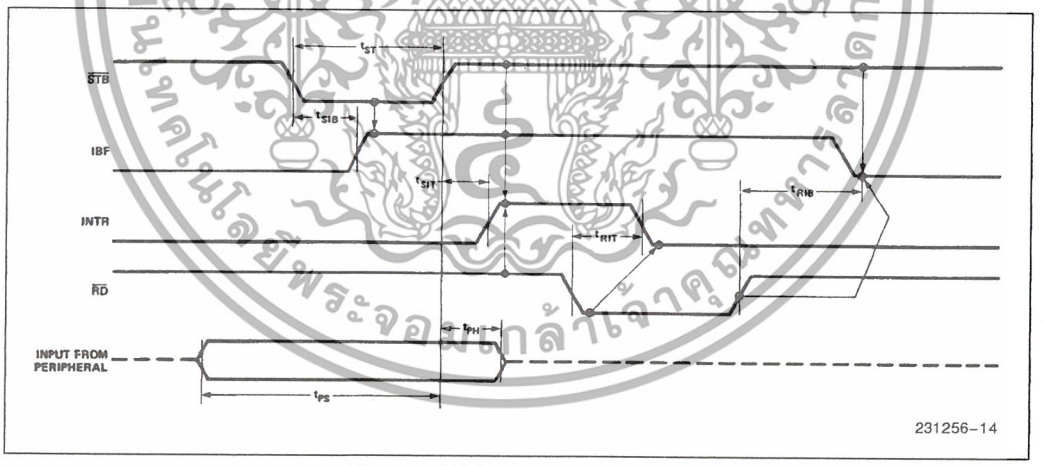



Figure 9. MODE 1 (Strobed Input)

### Output Control Signal Definition

**$\overline{OBF}$ (Output Buffer Full F/F).** The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{OBF}$ F/F will be set by the rising edge of the $\overline{WR}$ input and reset by $\overline{ACK}$ Input being low.

**$\overline{ACK}$ (Acknowledge Input).** A "low" on this input informs the 82C55A that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

**INTR (Interrupt Request).** A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{ACK}$ is a "one", $\overline{OBF}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{WR}$.

**INTE A**

Controlled by bit set/reset of $PC_6$.

**INTE B**

Controlled by bit set/reset of $PC_2$.

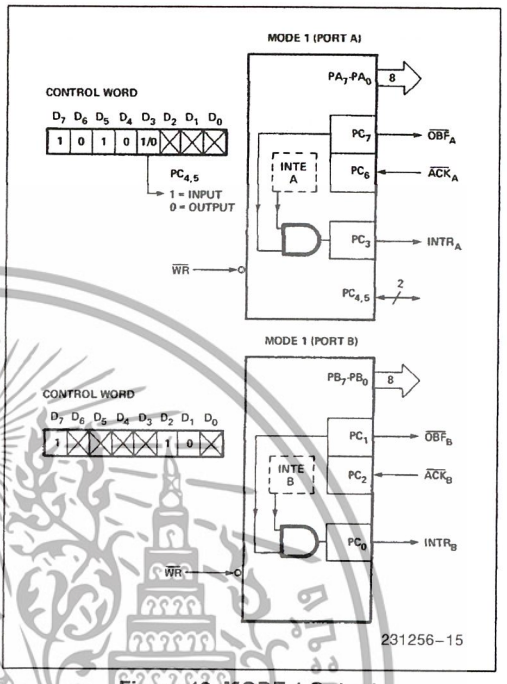



Figure 10. MODE 1 Output

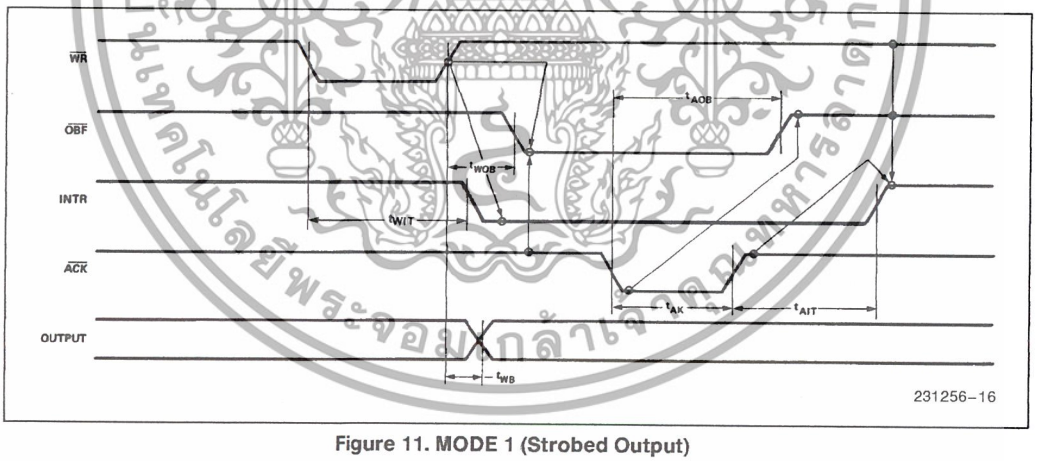



Figure 11. MODE 1 (Strobed Output)

### Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

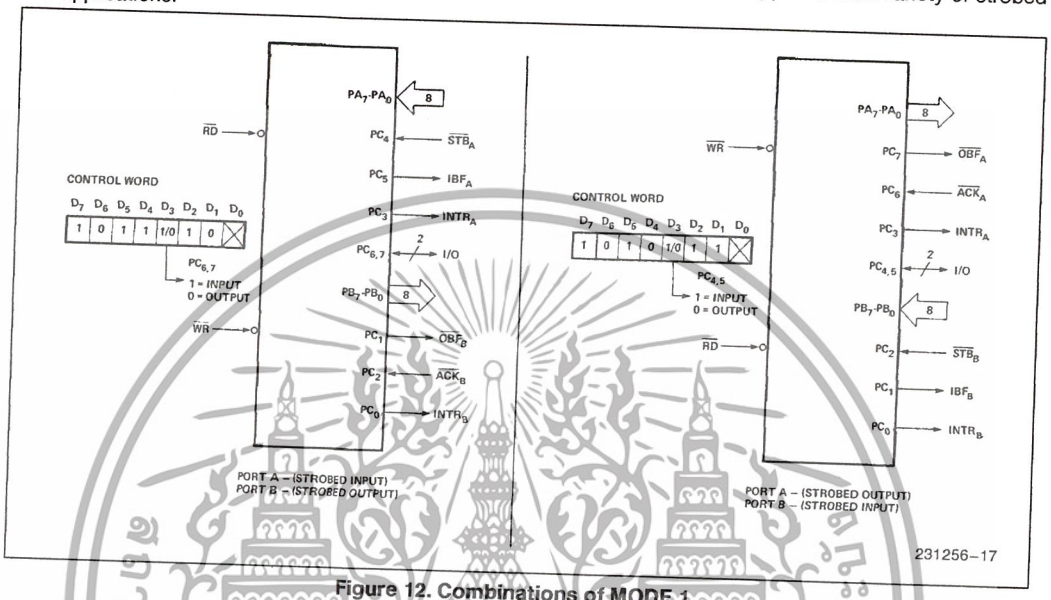



**Figure 12. Combinations of MODE 1**

### Operating Modes

**MODE 2 (Strobed Bidirectional Bus I/O).** This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only**.
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

### Bidirectional Bus I/O Control Signal Definition

**INTR (Interrupt Request).** A high on this output can be used to interrupt the CPU for input or output operations.

### Output Operations

**$\overline{OBF}$ (Output Buffer Full).** The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to port A.

**$\overline{ACK}$ (Acknowledge).** A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

**INTE 1 (The INTE Flip-Flop Associated with $\overline{OBF}$).** Controlled by bit set/reset of $PC_6$.

### Input Operations

**$\overline{STB}$ (Strobe Input).** A "low" on this input loads data into the input latch.

**IBF (Input Buffer Full F/F).** A "high" on this output indicates that data has been loaded into the input latch.

**INTE 2 (The INTE Flip-Flop Associated with IBF).** Controlled by bit set/reset of $PC_4$.

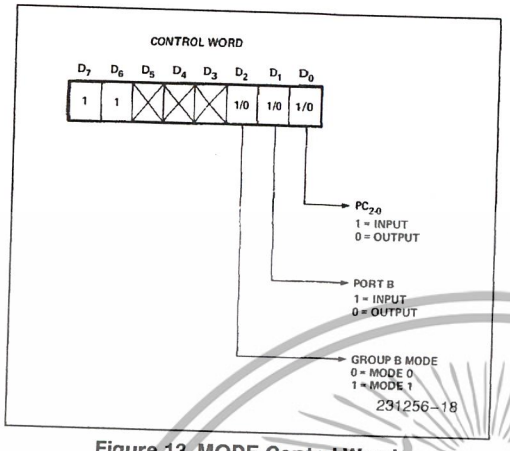



Figure 13. MODE Control Word

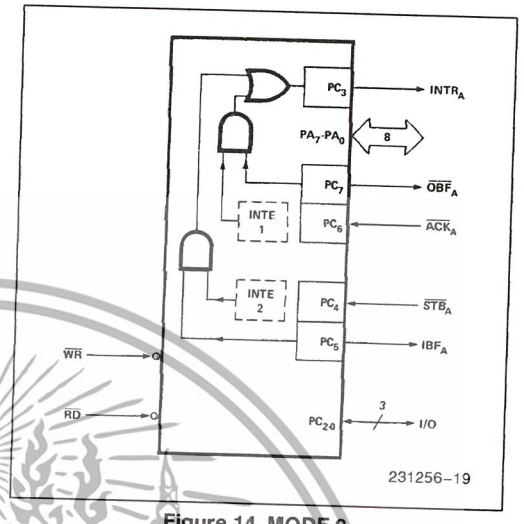



Figure 14. MODE 2

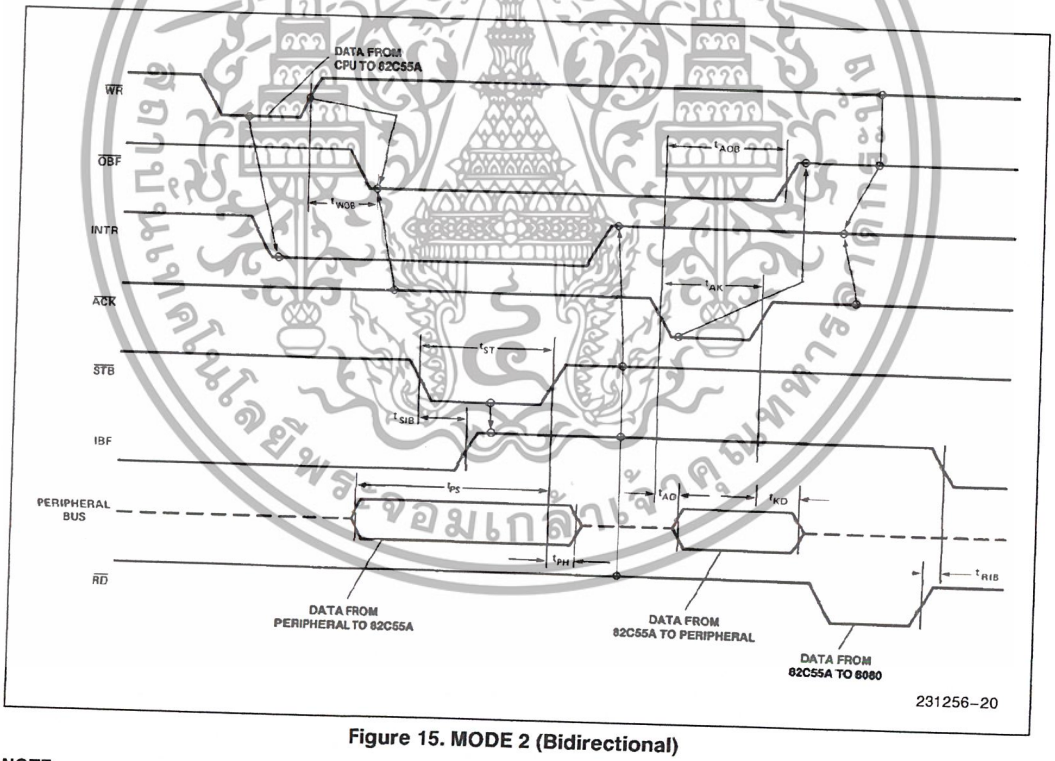



Figure 15. MODE 2 (Bidirectional)

**NOTE:**
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$, and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
($INTR = IBF \bullet \overline{MASK} \bullet \overline{STB} \bullet \overline{RD} + \overline{OBF} \bullet \overline{MASK} \bullet \overline{ACK} \bullet \overline{WR}$)

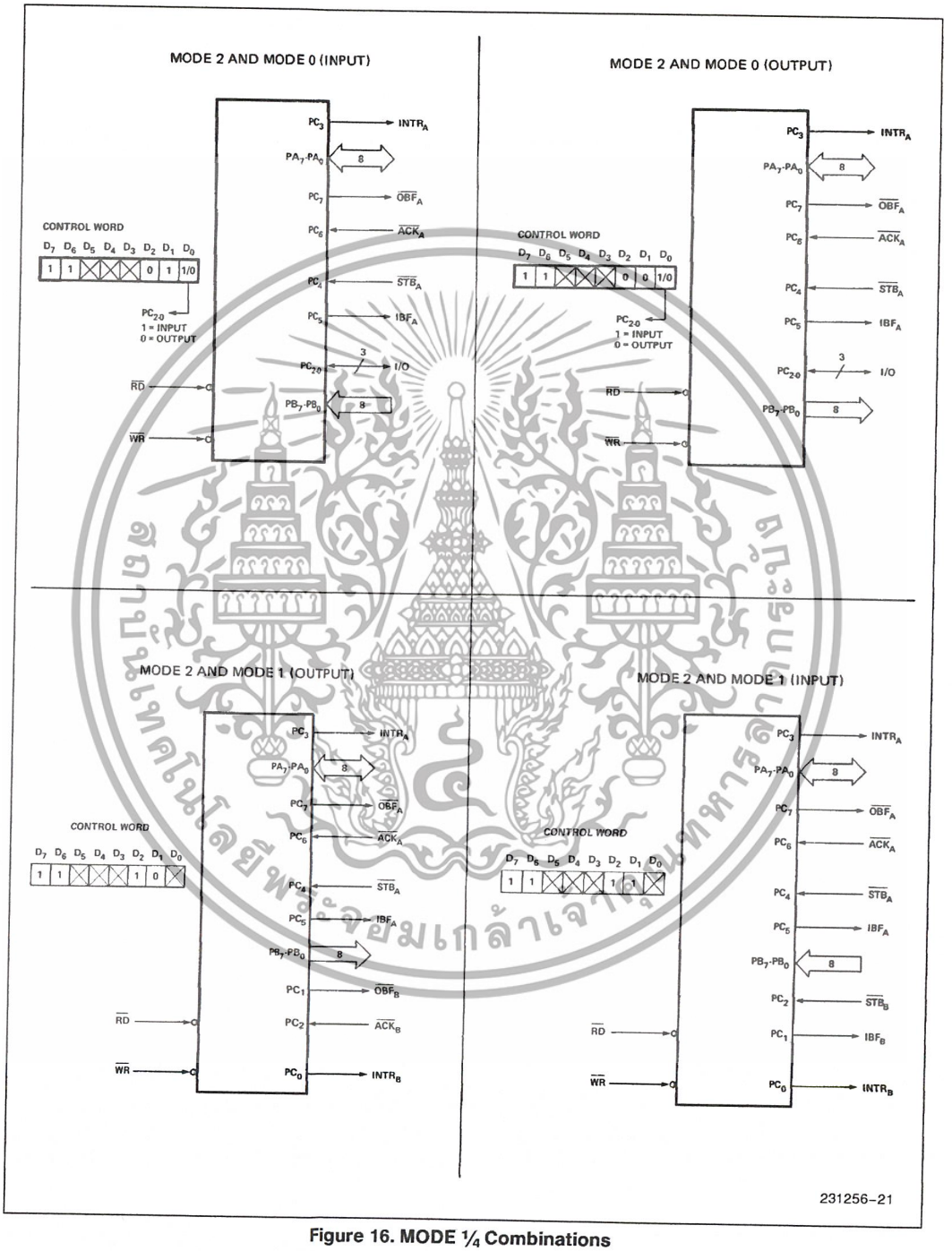



**Figure 16. MODE 1/4 Combinations**

**Mode Definition Summary**

| | MODE 0 | | MODE 1 | | MODE 2 | |
|---|---|---|---|---|---|---|
| | IN | OUT | IN | OUT | GROUP A ONLY | |
| $PA_0$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_1$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_2$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_3$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_4$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_5$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_6$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_7$ | IN | OUT | IN | OUT | ⟷ | |
| $PB_0$ | IN | OUT | IN | OUT | — | MODE 0 OR MODE 1 ONLY |
| $PB_1$ | IN | OUT | IN | OUT | — | |
| $PB_2$ | IN | OUT | IN | OUT | — | |
| $PB_3$ | IN | OUT | IN | OUT | — | |
| $PB_4$ | IN | OUT | IN | OUT | — | |
| $PB_5$ | IN | OUT | IN | OUT | — | |
| $PB_6$ | IN | OUT | IN | OUT | — | |
| $PB_7$ | IN | OUT | IN | OUT | — | |
| $PC_0$ | IN | OUT | $INTR_B$ | $INTR_B$ | I/O | |
| $PC_1$ | IN | OUT | $IBF_B$ | $\overline{OBF}_B$ | I/O | |
| $PC_2$ | IN | OUT | $\overline{STB}_B$ | $\overline{ACK}_B$ | I/O | |
| $PC_3$ | IN | OUT | $INTR_A$ | $INTR_A$ | $INTR_A$ | |
| $PC_4$ | IN | OUT | $\overline{STB}_A$ | I/O | $\overline{STB}_A$ | |
| $PC_5$ | IN | OUT | $IBF_A$ | I/O | $IBF_A$ | |
| $PC_6$ | IN | OUT | I/O | $\overline{ACK}_A$ | $\overline{ACK}_A$ | |
| $PC_7$ | IN | OUT | I/O | $\overline{OBF}_A$ | $\overline{OBF}_A$ | |

**Special Mode Combination Considerations**

There are several combinations of modes possible. For any combination, some or all of the Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the $\overline{ACK}$ and $\overline{STB}$ lines, will be placed on the data bus. In place of the $\overline{ACK}$ and $\overline{STB}$ line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 18.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including INTR, IBF and $\overline{OBF}$) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{ACK}$ and $\overline{STB}$ lines, associated with Port C are not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the $\overline{ACK}$ and $\overline{STB}$ lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 18.

**Current Drive Capability**

Any output on Port A, B or C can sink or source 2.5 mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

### Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

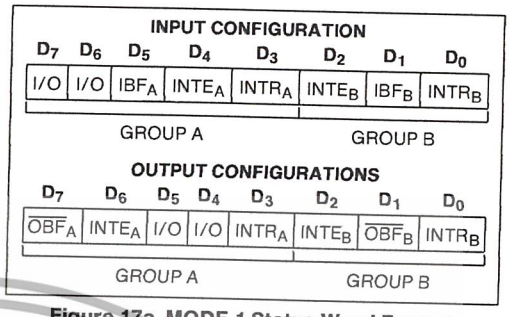



**Figure 17a. MODE 1 Status Word Format**

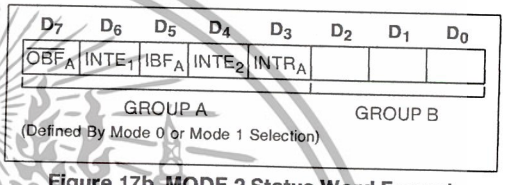



**Figure 17b. MODE 2 Status Word Format**

| Interrupt Enable Flag | Position | Alternate Port C Pin Signal (Mode) |
|---|---|---|
| INTE B | PC2 | $\overline{ACK}_B$ (Output Mode 1) or $\overline{STB}_B$ (Input Mode 1) |
| INTE A2 | PC4 | $\overline{STB}_A$ (Input Mode 1 or Mode 2) |
| INTE A1 | PC6 | $\overline{ACK}_A$ (Output Mode 1 or Mode 2) |

**Figure 18. Interrupt Enable Flags in Modes 1 and 2**

## ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to + 70°C
Storage Temperature . . . . . − 65°C to + 150°C
Supply Voltage . . . . . − 0.5 to + 8.0V
Operating Voltage . . . . . + 4V to + 7V
Voltage on any Input . . . . . GND − 2V to + 6.5V
Voltage on any Output . . GND − 0.5V to $V_{CC}$ + 0.5V
Power Dissipation . . . . . 1 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

## D.C. CHARACTERISTICS

$T_A$ = 0°C to 70°C, $V_{CC}$ = +5V ±10%, GND = 0V ($T_A$ = −40°C to +85°C for Extended Temperture)

| Symbol | Parameter | Min | Max | Units | Test Conditions |
|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | −0.5 | 0.8 | V | |
| $V_{IH}$ | Input High Voltage | 2.0 | $V_{CC}$ | V | |
| $V_{OL}$ | Output Low Voltage | | 0.4 | V | $I_{OL}$ = 2.5 mA |
| $V_{OH}$ | Output High Voltage | 3.0 | | V | $I_{OH}$ = −2.5 mA |
| | | $V_{CC}$ − 0.4 | | V | $I_{OH}$ = −100 μA |
| $I_{IL}$ | Input Leakage Current | | ±1 | μA | $V_{IN}$ = $V_{CC}$ to 0V (Note 1) |
| $I_{OFL}$ | Output Float Leakage Current | | ±10 | μA | $V_{IN}$ = $V_{CC}$ to 0V (Note 2) |
| $I_{DAR}$ | Darlington Drive Current | ±2.5 | (Note 4) | mA | Ports A, B, C<br>$R_{ext}$ = 500Ω<br>$V_{ext}$ = 1.7V |
| $I_{PHL}$ | Port Hold Low Leakage Current | +50 | +300 | μA | $V_{OUT}$ = 1.0V<br>Port A only |
| $I_{PHH}$ | Port Hold High Leakage Current | −50 | −300 | μA | $V_{OUT}$ = 3.0V<br>Ports A, B, C |
| $I_{PHLO}$ | Port Hold Low Overdrive Current | −350 | | μA | $V_{OUT}$ = 0.8V |
| $I_{PHHO}$ | Port Hold High Overdrive Current | +350 | | μA | $V_{OUT}$ = 3.0V |
| $I_{CC}$ | $V_{CC}$ Supply Current | | 10 | mA | (Note 3) |
| $I_{CCSB}$ | $V_{CC}$ Supply Current-Standby | | 10 | μA | $V_{CC}$ = 5.5V<br>$V_{IN}$ = $V_{CC}$ or GND<br>Port Conditions<br>If I/P = Open/High<br>O/P = Open Only<br>With Data Bus = High/Low<br>$\overline{CS}$ = High<br>Reset = Low<br>Pure Inputs = Low/High |

**NOTES:**
1. Pins $A_1$, $A_0$, $\overline{CS}$, $\overline{WR}$, $\overline{RD}$, Reset.
2. Data Bus; Ports B, C.
3. Outputs open.
4. Limit output current to 4.0 mA.

## CAPACITANCE

$T_A$ = 25°C, $V_{CC}$ = GND = 0V

| Symbol | Parameter | Min | Max | Units | Test Conditions |
|---|---|---|---|---|---|
| $C_{IN}$ | Input Capacitance | | 10 | pF | Unmeasured pins returned to GND $f_c$ = 1 MHz(5) |
| $C_{I/O}$ | I/O Capacitance | | 20 | pF | |

**NOTE:**
5. Sampled not 100% tested.

## A.C. CHARACTERISTICS

$T_A$ = 0° to 70°C, $V_{CC}$ = +5V ±10%, GND = 0V

$T_A$ = −40°C to +85°C for Extended Temperature

### BUS PARAMETERS

### READ CYCLE

| Symbol | Parameter | 82C55A-2 Min | 82C55A-2 Max | Units | Test Conditions |
|---|---|---|---|---|---|
| $t_{AR}$ | Address Stable Before $\overline{RD}$ ↓ | 0 | | ns | |
| $t_{RA}$ | Address Hold Time After $\overline{RD}$ ↑ | 0 | | ns | |
| $t_{RR}$ | $\overline{RD}$ Pulse Width | 150 | | ns | |
| $t_{RD}$ | Data Delay from $\overline{RD}$ ↓ | | 120 | ns | |
| $t_{DF}$ | $\overline{RD}$ ↑ to Data Floating | 10 | 75 | ns | |
| $t_{RV}$ | Recovery Time between $\overline{RD}$/$\overline{WR}$ | 200 | | ns | |

### WRITE CYCLE

| Symbol | Parameter | 82C55A-2 Min | 82C55A-2 Max | Units | Test Conditions |
|---|---|---|---|---|---|
| $t_{AW}$ | Address Stable Before $\overline{WR}$ ↓ | 0 | | ns | |
| $t_{WA}$ | Address Hold Time After $\overline{WR}$ ↑ | 20 | | ns | Ports A & B |
| | | 20 | | ns | Port C |
| $t_{WW}$ | $\overline{WR}$ Pulse Width | 100 | | ns | |
| $t_{DW}$ | Data Setup Time Before $\overline{WR}$ ↑ | 100 | | ns | |
| $t_{WD}$ | Data Hold Time After $\overline{WR}$ ↑ | 30 | | ns | Ports A & B |
| | | 30 | | ns | Port C |

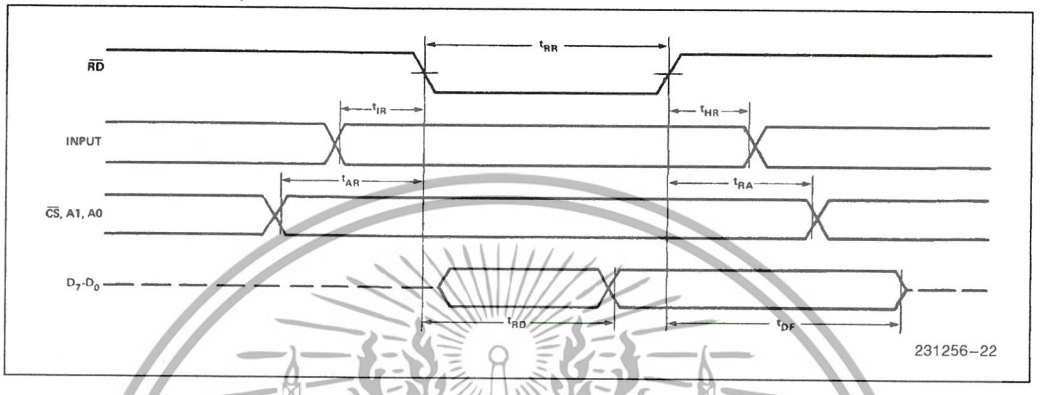
## OTHER TIMINGS

| Symbol | Parameter | 82C55A-2 | | Units Conditions | Test |
|---|---|---|---|---|---|
| | | Min | Max | | |
| $t_{WB}$ | $\overline{WR}$ = 1 to Output | | 350 | ns | |
| $t_{IR}$ | Peripheral Data Before $\overline{RD}$ | 0 | | ns | |
| $t_{HR}$ | Peripheral Data After $\overline{RD}$ | 0 | | ns | |
| $t_{AK}$ | $\overline{ACK}$ Pulse Width | 200 | | ns | |
| $t_{ST}$ | $\overline{STB}$ Pulse Width | 100 | | ns | |
| $t_{PS}$ | Per. Data Before $\overline{STB}$ High | 20 | | ns | |
| $t_{PH}$ | Per. Data After $\overline{STB}$ High | 50 | | ns | |
| $t_{AD}$ | $\overline{ACK}$ = 0 to Output | | 175 | ns | |
| $t_{KD}$ | $\overline{ACK}$ = 1 to Output Float | 20 | 250 | ns | |
| $t_{WOB}$ | $\overline{WR}$ = 1 to $\overline{OBF}$ = 0 | | 150 | ns | |
| $t_{AOB}$ | $\overline{ACK}$ = 0 to $\overline{OBF}$ = 1 | | 150 | ns | |
| $t_{SIB}$ | $\overline{STB}$ = 0 to IBF = 1 | | 150 | ns | |
| $t_{RIB}$ | $\overline{RD}$ = 1 to IBF = 0 | | 150 | ns | |
| $t_{RIT}$ | $\overline{RD}$ = 0 to INTR = 0 | | 200 | ns | |
| $t_{SIT}$ | $\overline{STB}$ = 1 to INTR = 1 | | 150 | ns | |
| $t_{AIT}$ | $\overline{ACK}$ = 1 to INTR = 1 | | 150 | ns | |
| $t_{WIT}$ | $\overline{WR}$ = 0 to INTR = 0 | | 200 | ns | see note 1 |
| $t_{RES}$ | Reset Pulse Width | 500 | | ns | see note 2 |

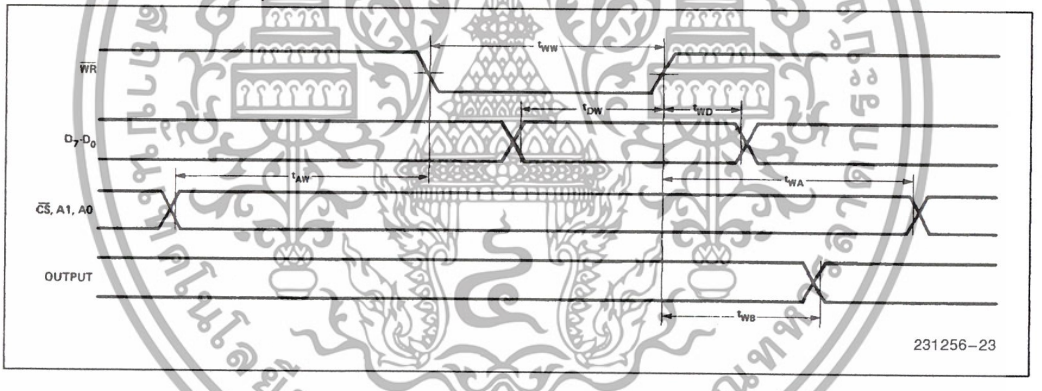
**NOTE:**
1. INTR ↑ may occur as early as $\overline{WR}$ ↓.
2. Pulse width of initial Reset pulse after power on must be at least 50 μSec. Subsequent Reset pulses may be 500 ns minimum. The output Ports A, B, or C may glitch low during the reset pulse but all port pins will be held at a logic "one" level after the reset pulse.

## WAVEFORMS

### MODE 0 (BASIC INPUT)

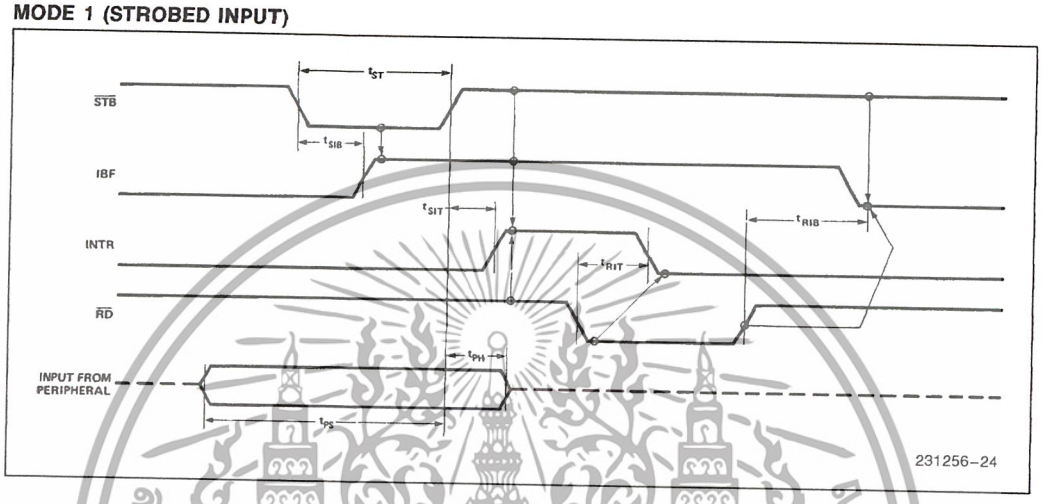



### MODE 0 (BASIC OUTPUT)

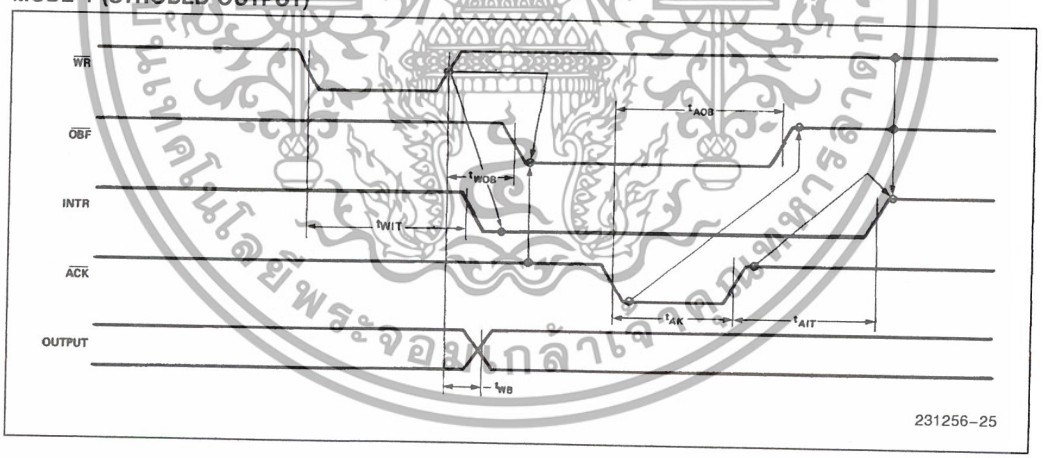


## WAVEFORMS (Continued)

### MODE 1 (STROBED INPUT)

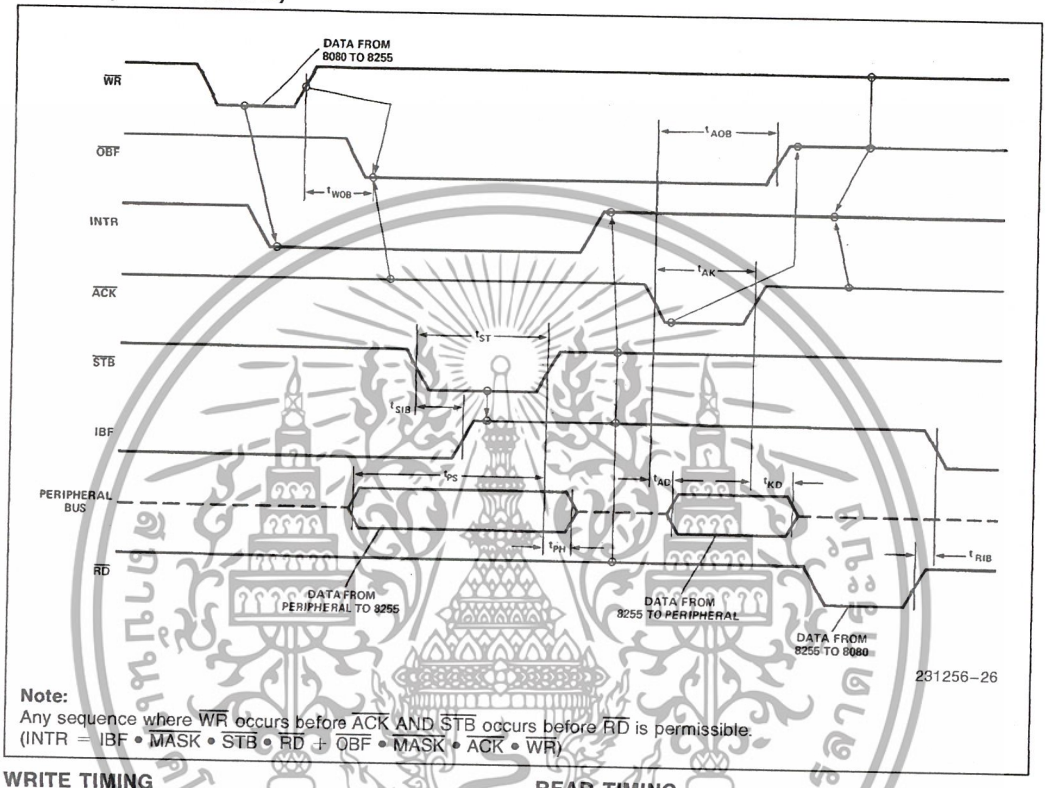



### MODE 1 (STROBED OUTPUT)



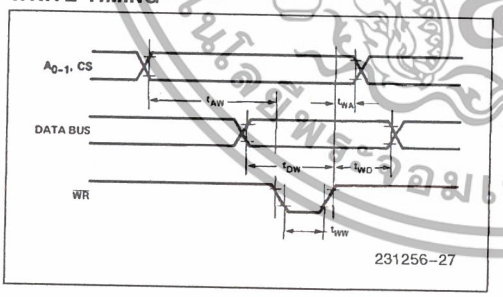
## WAVEFORMS (Continued)

### MODE 2 (BIDIRECTIONAL)

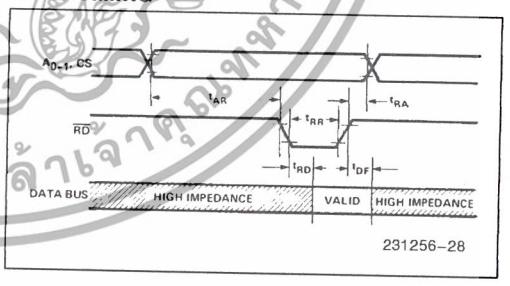



**Note:**
Any sequence where $\overline{WR}$ occurs before ACK AND $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + OBF • MASK • $\overline{ACK}$ • $\overline{WR}$)

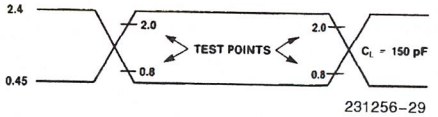
### WRITE TIMING




### READ TIMING

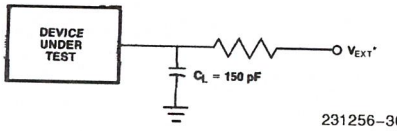



### A.C. TESTING INPUT, OUTPUT WAVEFORM




A.C. Testing Inputs Are Driven At 2.4V For A Logic 1 And 0.45V For A Logic 0 Timing Measurements Are Made At 2.0V For A Logic 1 And 0.8 For A Logic 0.

### A.C. TESTING LOAD CIRCUIT




*$V_{EXT}$ Is Set At Various Voltages During Testing To Guarantee The Specification. $C_L$ Includes Jig Capacitance.

DALLAS SEMICONDUCTOR

# DS275
# Line-Powered RS-232 Transceiver Chip

www.dalsemi.com

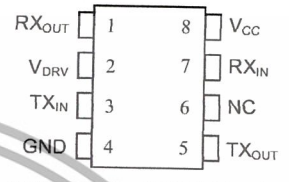
## FEATURES

- Low-power serial transmitter/receiver for battery-backed systems
- Transmitter steals power from receive signal line to save power
- Ultra-low static current, even when connected to RS-232-E port
- Variable transmitter level from +5 to +12 volts
- Compatible with RS-232-E signals
- Available in 8-pin, 150 mil wide SOIC package (DS275S)
- Low-power CMOS

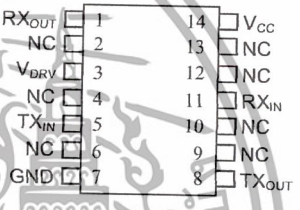
## ORDERING INFORMATION

| | |
|---|---|
| DS275 | 8-pin DIP |
| DS275S | 8-pin SOIC |
| DS275E | 14-pin TSSOP |

## PIN ASSIGNMENT




DS275 8-Pin DIP (300-mil)
DS275 8-Pin SOIC (150-mil)




DS275E 14-Pin TSSOP

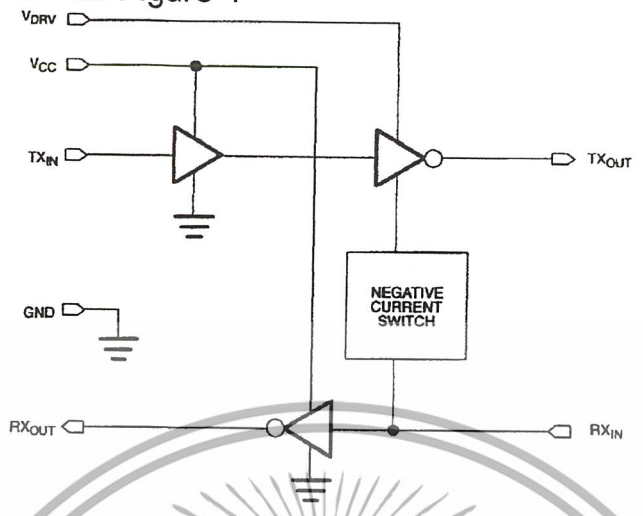
## PIN DESCRIPTION

| | |
|---|---|
| $RX_{OUT}$ | - RS-232 Receiver Output |
| $V_{DRV}$ | - Transmit driver +V |
| $TX_{IN}$ | - RS-232 Driver Input |
| GND | - System Ground (0V) |
| $TX_{OUT}$ | - RS-232 Driver Output |
| NC | - No Connection |
| $RX_{IN}$ | - RS-232 Receive Input |
| $V_{CC}$ | -System Logic Supply (+5V) |

## DESCRIPTION

The DS275 Line-Powered RS-232 Transceiver Chip is a CMOS device that provides a low-cost, very low-power interface to RS-232 serial ports. The receiver input translates RS-232 signal levels to common CMOS/TTL levels. The transmitter employs a unique circuit which steals current from the receive RS-232 signal when that signal is in a negative state (marking). Since most serial communication ports remain in a negative state statically, using the receive signal for negative power greatly reduces the DS275's static power consumption. This feature is especially important for battery-powered systems such as laptop computers, remote sensors, and portable medical instruments. During an actual communication session, the DS275's transmitter will use system power (5-12 volts) for positive transitions while still employing the receive signal for negative transitions.

## DS275 BLOCK DIAGRAM Figure 1




## OPERATION

Designed for the unique requirements of battery-backed systems, the DS275 provides a low-power half-duplex interface to an RS-232 serial port. Typically, a designer must use an RS-232 device which uses system power during both negative and positive transitions of the transmit signal to the RS-232 port. If the connector to the RS-232 port is left connected for an appreciable time after the communication session has ended, power will statically flow into that port, draining the battery capacity. The DS275 eliminates this static current drain by stealing current from the receive line ($RX_{IN}$) of the RS-232 port when that line is at a negative level (marking). Since most asynchronous communication over an RS-232 connection typically remains in a marking state when data is not being sent, the DS275 will not consume system power in this condition. System power would only be used when positive-going transitions are needed on the transmit RS-232 output ($TX_{OUT}$) when data is sent. However, since synchronous communication sessions typically exhibit a very low duty-cycle, overall system power consumption remains low.

## RECEIVER SECTION

The $RX_{IN}$ pin is the receive input for an RS-232 signal whose levels can range from ±3 to ±15 volts. A negative data signal is called a mark while a positive data signal is called a space. These signals are inverted and then level-shifted to normal +5-volt CMOS/TTL logic levels. The logic output associated with $RX_{IN}$ is $RX_{OUT}$ which swings from +Vcc to ground. Therefore, a mark on $RX_{IN}$ produces a logic 1 at $RX_{OUT}$; a space produces a logic 0.

The input threshold of $RX_{IN}$ is typically around 1.8 volts with 500 millivolts of hysteresis to improve noise rejection. Therefore, an input positive-going signal must exceed 1.8 volts to cause $RX_{OUT}$ to switch states. A negative-going signal must now be lower than 1.3 volts (typically) to cause $RX_{OUT}$ to switch again. An open on $RX_{IN}$ is interpreted as a mark, producing a logic 1 at $RX_{OUT}$.

## TRANSMITTER SECTION

$TX_{IN}$ is the CMOS/TTL-compatible input for digital data from the user system. A logic 1 at $TX_{IN}$ produces a mark (negative data signal) at $TX_{OUT}$ while a logic 0 produces a space (positive data signal). As mentioned earlier, the transmitter section employs a unique driver design that uses the $RX_{IN}$ line for swinging to negative levels. The $RX_{IN}$ line must be in a marking or idle state to take advantage of this design; if $RX_{IN}$ is in a spacing state, $TX_{OUT}$ will only swing to ground. When $TX_{OUT}$ needs to transition to a positive level, it uses the $V_{DRV}$ power pin for this level. $V_{DRV}$ can be a voltage supply between 5 to 12

volts, and in many situations it can be tied directly to the +5 volt $V_{CC}$ supply. *It is important to note that $V_{DRV}$ must be greater than or equal to $V_{CC}$ at all times.*

The voltage range on $V_{DRV}$ permits the use of a 9-volt battery in order to provide a higher voltage level when $TX_{OUT}$ is in a space state. When $V_{CC}$ is shut off to the DS275 and $V_{DRV}$ is still powered (as might happen in a battery-backed condition), only a small leakage current (about 50-100 nA) will be drawn. If $TX_{OUT}$ is loaded during such a condition, $V_{DRV}$ will draw current only if $RX_{IN}$ is not in a negative state. During normal operation ($V_{CC}$=5 volts), $V_{DRV}$ will draw less than 2 uA when $TX_{OUT}$ is marking. Of course, when $TX_{OUT}$ is spacing, $V_{DRV}$ will draw substantially more current—about 3 mA, depending upon its voltage and the impedance that $TX_{OUT}$ sees.

The $TX_{OUT}$ output is slew rate-limited to less than 30 volts/us in accordance with RS-232 specifications. In the event $TX_{OUT}$ should be inadvertently shorted to ground, internal current-limiting circuitry prevents damage, even if continuously shorted.

## RS-232 COMPATIBILITY

The intent of the DS275 is not so much to meet all the requirements of the RS-232 specification as to offer a low-power solution that will work with most RS-232 ports with a connector length of less than 10 feet. As a prime example, the DS275 will not meet the RS-232 requirement that the signal levels be at least ±5 volts minimum when terminated by a 3 kΩ load and $V_{DRV}$ = +5 volts. Typically a voltage of 4 volts will be present at $TX_{OUT}$ when spacing. However, since most RS-232 receivers will correctly interpret any voltage over 2 volts as a space, there will be no problem transmitting data.

## APPLICATIONS INFORMATION

The DS275 is designed as a low-cost, RS-232-E interface expressly tailored for the unique requirements of battery-operated handheld products. As shown in the electrical specifications, the DS275 draws exceptionally low operating and static current. During normal operation when data from the handheld system is sent from the $TX_{OUT}$ output, the DS275 only draws significant $V_{DRV}$ current when $TX_{OUT}$ transitions positively (spacing). This current flows primarily into the RS-232 receiver's 3-7 kΩ load at the other end of the attaching cable. When $TX_{OUT}$ is marking (a negative data signal), the $V_{DRV}$ current falls dramatically since the negative voltage is provided by the transmit signal from the other end of the cable. This represents a large reduction in overall operating current, since typical RS-232 interface chips use charge-pump circuits to establish both positive and negative levels at the transmit driver output.

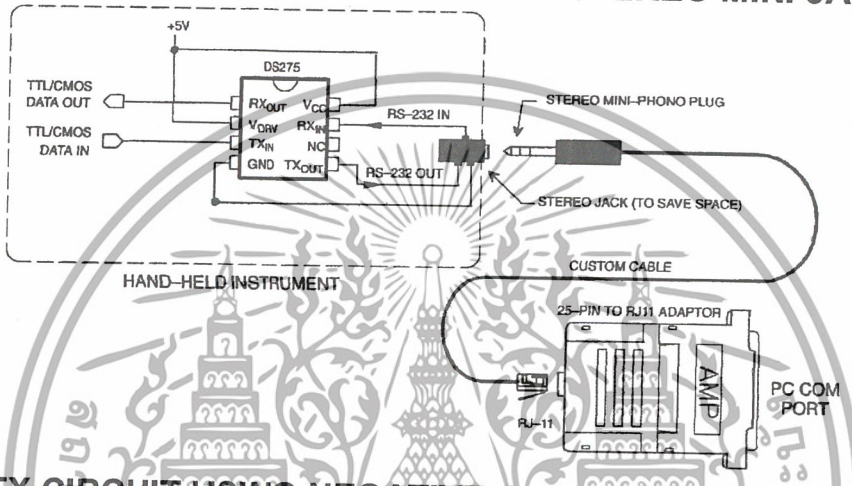
To obtain the lowest power consumption from the DS275, observe the following guidelines. First, to minimize $V_{DRV}$ current when connected to an RS-232 port, always maintain $TX_{IN}$ at a logic 1 when data is not being transmitted (idle state). This will force $TX_{OUT}$ into the marking state, minimizing $V_{DRV}$ current. Second, $V_{DRV}$ current will drop to less than 100 nA when $V_{CC}$ is grounded. Therefore, if $V_{DRV}$ is tied directly to the system battery, the logic +5 volts can be turned off to achieve the lowest possible power state.

## FULL-DUPLEX OPERATION

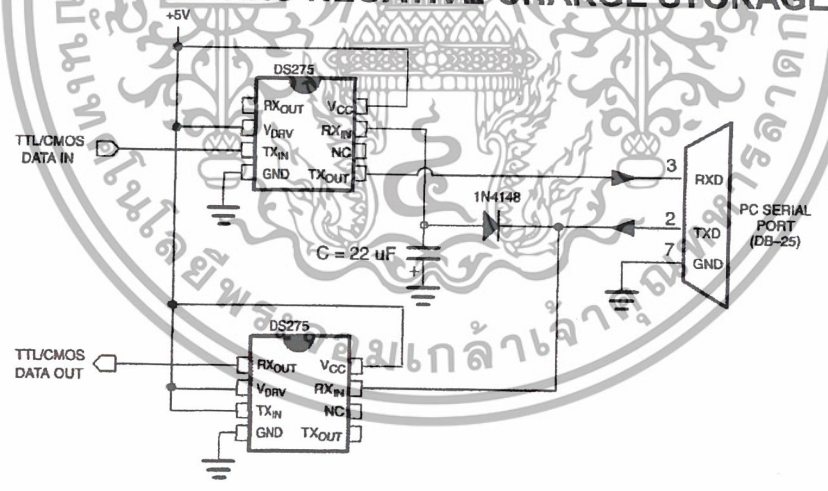
The DS275 is intended primarily for half-duplex operation; that is, $RX_{IN}$ should remain idle in the marking state when transmitting data out $TX_{OUT}$ and visa versa. However, the part can be operated full-duplex with most RS-232-E serial ports since signals swinging between 0 and +5V will usually be correctly interpreted by an RS-232-E receiver device. The 5-volt swing occurs when $TX_{OUT}$ attempts to swing negative while $RX_{IN}$ is at a positive voltage, which turns on an internal weak pulldown to ground for the $TX_{OUT}$ driver's negative reference. So, transmit mark signals at $TX_{OUT}$ may have voltage jumps from some negative value (corresponding to $RX_{IN}$ marking) to approximately ground. One possible

problem that may occur in this case is if the receiver at the other end requires a negative voltage for recognizing a mark. In this situation, the full-duplex circuit shown in Figure 3 can be used as analternative. The 22 µF capacitor forms a negative-charge reservoir; consequently, when the TXD line is spacing (positive), TXOUT still has a negative source available for a time period determined by the capacitor and the load resistance at the other end (3-7 kΩ). This circuit was tested from 150-19,200 bps with error-free operation using a SN75154 Quad Line Receiver as the receiver for the TXOUT signal. Note that the SN75154 can have a marking input threshold below ground; hence there is the need for TXOUT to swing both positive and negative in full-duplex operation with this device.

## HANDHELD RS-232-C APPLICATION USING A STEREO MINI-JACK Figure 2




## FULL-DUPLEX CIRCUIT USING NEGATIVE-CHARGE STORAGE Figure 3




## NOTE:

The capacitor stores negative charge whenever the TXD signal from the PC serial port is in a marking data state (a negative voltage that is typically -10 volts). The top DS275's TXOUT uses this negative charge reservoir when it is in a marking state. The capacitor will discharge to 0 volts when the TXD line is spacing (and TXOUT is still marking) at a time constant determined by its value and the value of the load resistance reflected back to TXOUT. However, when TXD is marking the capacitor will quickly charge back to -10 volts. Note that TXD remains in a marking state when idle, which improves the performance of this circuit.

## ABSOLUTE MAXIMUM RATINGS*

| | |
|---|---|
| $V_{CC}$ | -0.3 to +7.0 volts |
| $V_{DRV}$ | -0.3 to +13.0 volts |

| | |
|---|---|
| $RX_{IN}$ | ±15 volts |
| $TX_{IN}$ | -0.3 to $V_{CC}$ + 0.3 volts |
| $TX_{OUT}$ | ±15 volts |
| $RX_{OUT}$ | -0.3 to $V_{CC}$ + 0.3 volts |
| Storage Temperature | -55°C to +125°C |
| Operating Temperature | 0°C to 70°C |

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

## RECOMMENDED DC OPERATING CONDITIONS

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS | NOTES |
|---|---|---|---|---|---|---|
| Logic Supply | $V_{CC}$ | 4.5 | 5.0 | 5.5 | V | 1 |
| Transmit Driver Supply | $V_{DRV}$ | 4.5 | 5-12 | 13.0 | V | 1 |
| Logic 1 Input | $V_{IH}$ | 2.0 | | $V_{CC}$+0.3 | V | 2 |
| Logic 0 Input | $V_{IL}$ | -0.3 | | +0.8 | V | |
| RS-232 Input Range ($RX_{IN}$) | $V_{RS}$ | -15 | | +15 | V | |
| Dynamic Supply Current | | | | | | |
| $TX_{IN}$ = $V_{CC}$ | $I_{DRV1}$ | | 400 | 800 | µA | 3 |
| | $I_{CC1}$ | | 40 | 100 | µA | |
| $TX_{IN}$ = GND | $I_{DRV1}$ | | 3.8 | 5.0 | µA | |
| | $I_{CC1}$ | | 40 | 100 | µA | |
| Static Supply Current | | | | | | |
| $TX_{IN}$ = $V_{CC}$ | $I_{DRV2}$ | | 1.5 | 10.0 | µA | 4 |
| | $I_{CC2}$ | | 10.0 | 15.0 | µA | |
| $TX_{IN}$ = GND | $I_{DRV2}$ | | 3.8 | 5.0 | mA | |
| | $I_{CC2}$ | | 10.0 | 20.0 | µA | |
| Driver Leakage Current ($V_{CC}$=0V) | $I_{DRV3}$ | | 0.05 | 1.0 | µA | 5 |

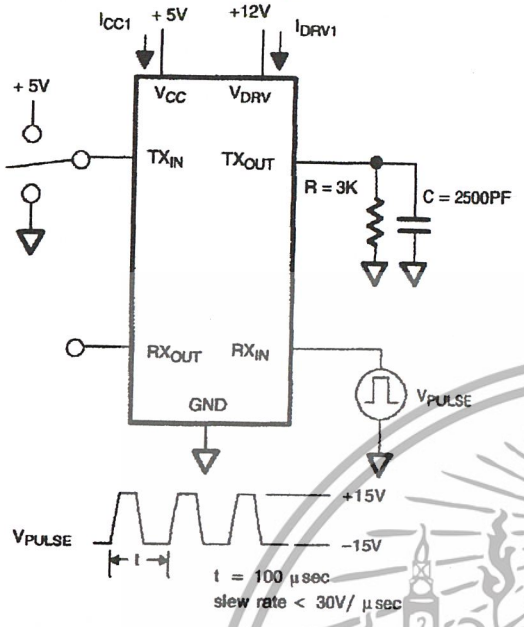
## DC ELECTRICAL CHARACTERISTICS (0°C to 70°C; $V_{CC}$ = $V_{DRV}$ = 5V ± 10%)

| PARAMETER | SYMBOL | MIN | TYP | MAX | UNITS | NOTES |
|---|---|---|---|---|---|---|
| $TX_{OUT}$ Level High | $V_{OTXH}$ | 3.5 | 4.0 | 5.0 | V | 6 |
| $TX_{OUT}$ Level Low | $V_{OTXL}$ | -8.5 | -9.0 | | V | 7 |
| $TX_{OUT}$ Short Circuit Current | $I_{SC}$ | | +60 | +85 | mA | |
| $TX_{OUT}$ Output Slew Rate | $t_{SR}$ | | | 30 | V/µs | |
| Propagation Delay | $t_{PD}$ | | 5 | | µs | 8 |
| $RX_{IN}$ Input Threshold Low | $V_{TL}$ | 0.8 | 1.2 | 1.6 | V | |
| $RX_{IN}$ Input Threshold High | $V_{TH}$ | 1.6 | 2.0 | 2.4 | V | |
| $RX_{IN}$ Threshold Hysteresis | $V_{HYS}$ | 0.5 | 0.8 | | V | 9 |
| $RX_{OUT}$ Output Current @ 2.4V | $I_{OH}$ | -1.0 | | | mA | |
| $RX_{OUT}$ Output Current @ 0.4V | $I_{OL}$ | | | 3.2 | mA | |

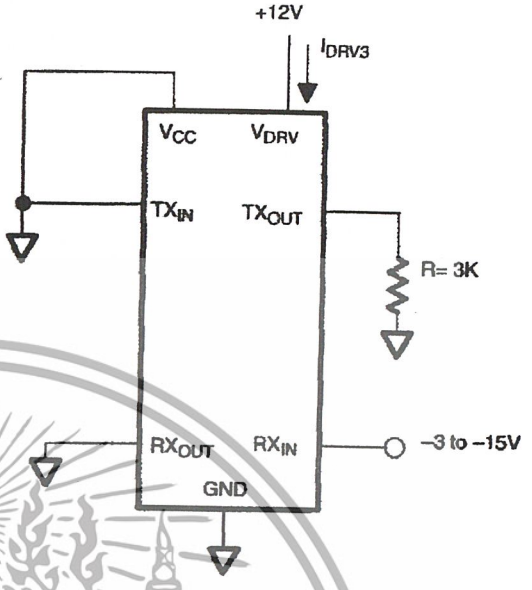
## NOTES:

1. $V_{DRV}$ must be greater than or equal to $V_{CC}$.
2. $V_{CC}$ = $V_{DRV}$ = 5V ± 10%.
3. See test circuit in Figure 4.
4. See test circuit in Figure 5.
5. See test circuit in Figure 6.
6. $TX_{IN}$ = $V_{IL}$ and $TX_{OUT}$ loaded by 3 kΩ to ground.
7. $TX_{IN}$ = $V_{IH}$, $RX_{IN}$ = -10 volts and $TX_{OUT}$ loaded by 3 kΩ to ground.
8. $TX_{IN}$ to $TX_{OUT}$ - see Figure 7.
9. $V_{HYS}$ = $V_{TH}$ - $V_{TL}$.

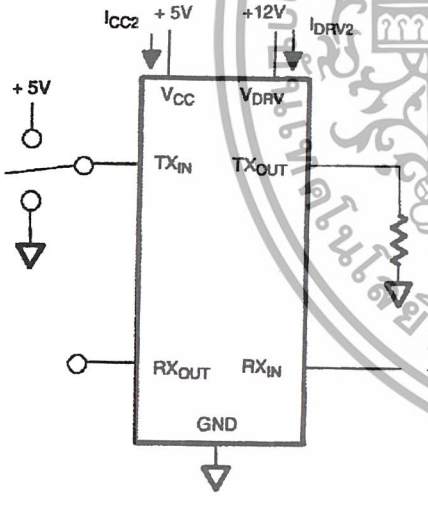
## DYNAMIC OPERATING CURRENT TEST CIRCUIT Figure 4



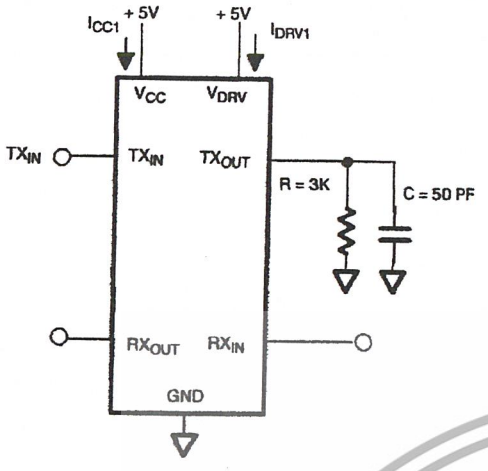

## DRIVER LEAKAGE TEST CIRCUIT Figure 6

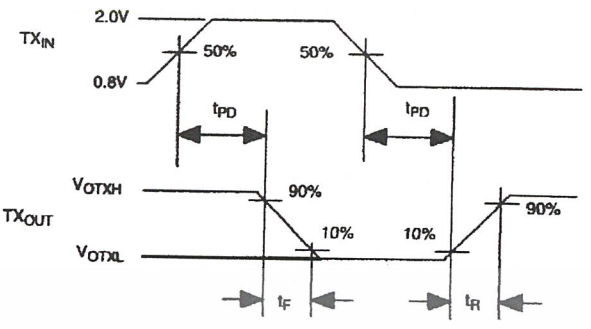



## STATIC OPERATING CURRENT TEST CIRCUIT Figure 5



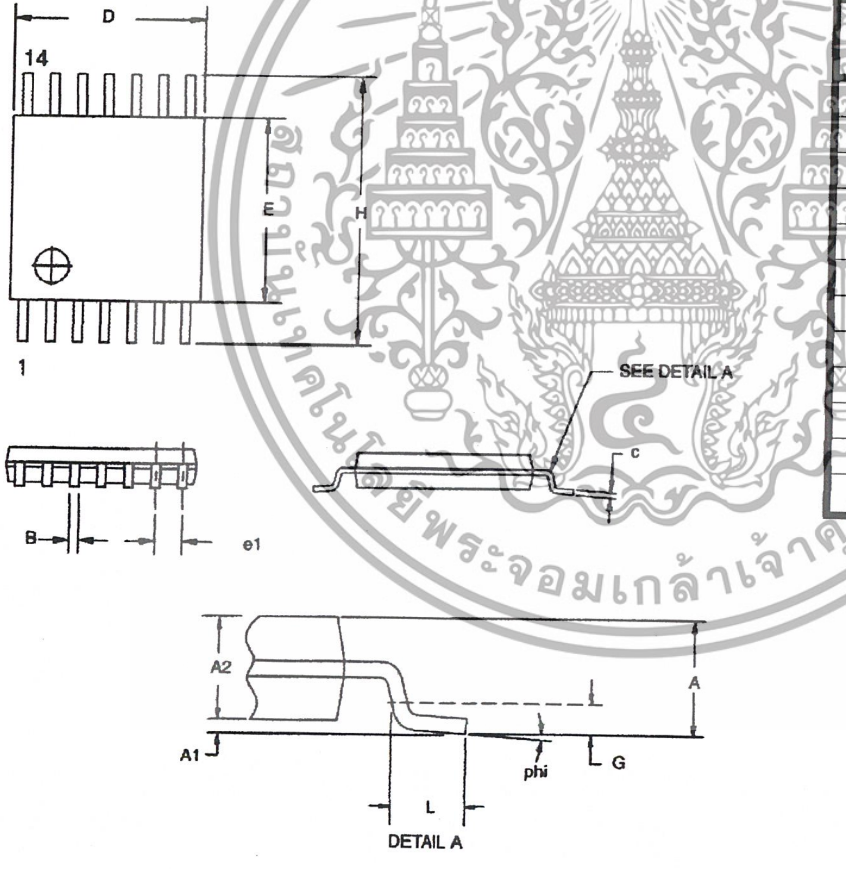
## PROPAGATION DELAY TEST CIRCUIT Figure 7




$$t_{SR} = \frac{0.8(V_{OTXH} - V_{OTXL})}{t_F \text{ or } t_R}$$

## DS275E 14-PIN TSSOP




| DIM | 14-PIN | |
|---|---|---|
| | MIN | MAX |
| A MM | - | 1.10 |
| A1 MM | 0.05 | - |
| A2 MM | 0.75 | 1.05 |
| B MM | 0.18 | 0.30 |
| C MM | 0.09 | 0.18 |
| D MM | 4.90 | 5.10 |
| E MM | 4.40 NOM | |
| e1 MM | 0.65 BSC | |
| G MM | 0.25 REF | |
| H MM | 6.25 | 6.55 |
| L MM | 0.50 | 0.70 |
| phi | 0° | 8° |