รถสำรวจเคลื่อนที่

Survey Vehicle

โดย

นาย พัฒน์พงษ์ ดิวสร้อย

นาย วรพจน์ ศรีโลพันธุ์

ว่าที่ ร.ต. สาธิต ทองมี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

รถสำรวจเคลื่อนที่

Survey Vehicle

โดย

นาย พัฒน์พงษ์ ติวสร้อย 42015227

นาย วรพจน์ ศรีโลพันธุ์ 42015233

ว่าที่ ร.ต. สาธิต ทองมี 42015243

อาจารย์ที่ปรึกษา

อาจารย์ พลศาสตร์ เลิศประเสริฐ

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง รถสำรวจเคลื่อนที่

ผู้จัดทำ นาย พัฒน์พงษ์ ติวสร้อย 42015227

นาย วรพจน์ ศรีโลพันธุ์ 42015233

ว่าที่ ร.ต. สาธิต ทองมี 42015243

..... อาจารย์ที่ปรึกษา

(อ. พลศาสตร์ เลิศประเสริฐ)

รถสำรวจเคลื่อนที่

SURVEY VEHICLE

นาย พัฒน์พงษ์ ติวสร้อย 42015227

นาย วรพจน์ ศรีโลพันธุ์ 42015233

ว่าที่ ร.ต. สาธิต ทองมี 42015243

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการตรวจสอบ

.....อาจารย์ที่ปรึกษา

(อ. พลศาสตร์ เลิศประเสริฐ)

รถสำรวจเคลื่อนที่

นาย พัฒน์พงษ์ ติวสร้อย 42015227
นาย วรพจน์ ศรีโลพันธุ์ 42015233
ว่าที่ ร.ต. สาธิต ทองมี 42015243
อาจารย์พลศาสตร์ เลิศประเสริฐ อาจารย์ที่ปรึกษา
ภาคเรียนที่ 2 ปีการศึกษา 2544

บทคัดย่อ

โครงงานนี้เป็นการควบคุมการขับเคลื่อนรถสำรวจ ซึ่งตัวรถเป็น DC Motor 2 ตัว สามารถเคลื่อนที่ได้ตามการบังคับของฐานควบคุม (Computer Base) อีกทั้งยังมีกล้องติดอยู่กับตัวรถโดยส่งภาพมายังฐานควบคุมเพื่อให้ผู้ที่อยู่ในฐานสามารถบังคับรถให้วิ่งในทิศทางที่ผู้ควบคุมต้องการ อีกทั้งการส่งข้อมูลสามารถส่งข้อมูลได้ในระยะไกล การออกแบบวงจรที่ใช้งานในโครงงานนี้ใช้ทั้งวงจร Digital และ Analog ให้ทำงานร่วมกัน

Survey Vehicle

Mr. Pudtanapong Tiesroy 42015227

Mr. Worapod Silopon 42015233

Sub^{lt} Satit Thongmee 42015243

Advisor Ponsart Lertprasert

2nd Semester, 2001

Abstract

This project is Survey Vehicle. Which had driven by 2 DC Motors. The car can move relate to the unit control base such direction and speed, when the car had a problem unit control base will received image problem from the car. This project has successfully been designed with the combination of analog circuit and digital circuit as well as the overall system.

สารบัญ

บทที่ 1
บทนำ

ในปัจจุบันรถสำรวจได้เข้ามามีบทบาท แทนที่มนุษย์ในการทำงานที่เสี่ยงภัยหรือในสถานที่ที่อันตราย เช่นการกู้ระเบิด การสำรวจเหมืองแร่ หรือแม้แต่สำรวจดาวอังคาร จึงมีความจำเป็นอย่างยิ่งยวดที่รถสำรวจจะต้องมีประสิทธิภาพมากที่สุดโดยสามารถตอบสนองความต้องการของมนุษย์ได้ทุกอย่างเช่น ต้องสามารควบคุมการทำงานของรถสำรวจในระยะไกลได้ และส่งข้อมูลกลับมาได้อย่างครบถ้วน รถสำรวจนับว่ามีบทบาทในด้านการสำรวจทางวิศวกรรม ไม่ว่าจะเป็นการสำรวจเหมือง,พื้นที่กู้ระเบิดและอีกมากมาย ล้วนจะต้องมีคุณสมบัติในการเคลื่อนที่ได้อย่างมีประสิทธิภาพ มีการแลกเปลี่ยนข่าวสารข้อมูลระหว่างสถานีควบคุมและตัวรถ ไม่ว่าจะเป็นข้อมูลภาพ, อุณหภูมิ,ปริมาณสารเคมี,ความเข้มแสง และอาจมีกลไกอย่างอื่นเช่น แขนกลเป็นต้น ในการออกแบบรถนั้นจะเน้นหนักไปในเรื่องการส่งข้อมูลจากเวิร์กเตชั่น ไปยังตัวรถ เพื่อที่จะให้รถปฏิบัติตามคำสั่งที่เราได้ออกแบบไว้

เป็นที่ยอมรับโดยทั่วไปว่าประเทศชั้นนำทางเทคโนโลยี ได้ตระหนักถึงความสำคัญในการพัฒนารถสำรวจมาทำงานแทนมนุษย์เพื่อลดความเสี่ยงในการทำงาน โครงงานนี้อาจเป็นโครงงานเล็กๆแต่ได้เล็งเห็นความสำคัญในจุดนี้ ได้นำเอาสิ่งที่มีอยู่แล้วภายในประเทศมาพัฒนาและวิจัยให้เกิดสิ่งที่มีประโยชน์แม้ว่าอาจจะยังไม่ดีพอ แต่เมื่อค่อยๆพัฒนาไปแล้วเชื่อแน่ว่าจะต้องได้รับผลสำเร็จแน่นอน

1.1 วัตถุประสงค์

1. เพื่อศึกษาหลักการและการประยุกต์ใช้งานในด้านการเขียนโปรแกรมคอมพิวเตอร์เพื่อเป็นแนวทางในการพัฒนาโปรแกรมโดยใช้โปรแกรม Borland C++ Builder
2. เพื่อศึกษาหลักการและการประยุกต์ใช้งานการควบคุมวงจรอิเล็กทรอนิกส์ผ่านเครื่องไมโครคอมพิวเตอร์ส่วนบุคคล
3. เพื่อศึกษาหลักการและประยุกต์ใช้งานการควบคุม ดีซีมอเตอร์ (DC Motor)
4. เพื่อศึกษาหลักการและประยุกต์ใช้งานของกล้อง ซีซีดี (CCD) ที่ควบคุมผ่านระบบไมโครคอมพิวเตอร์
5. เพื่อศึกษาหลักการและประยุกต์ใช้ไมโครคอนโทรลเลอร์

1.2 ขอบเขตของโครงงาน

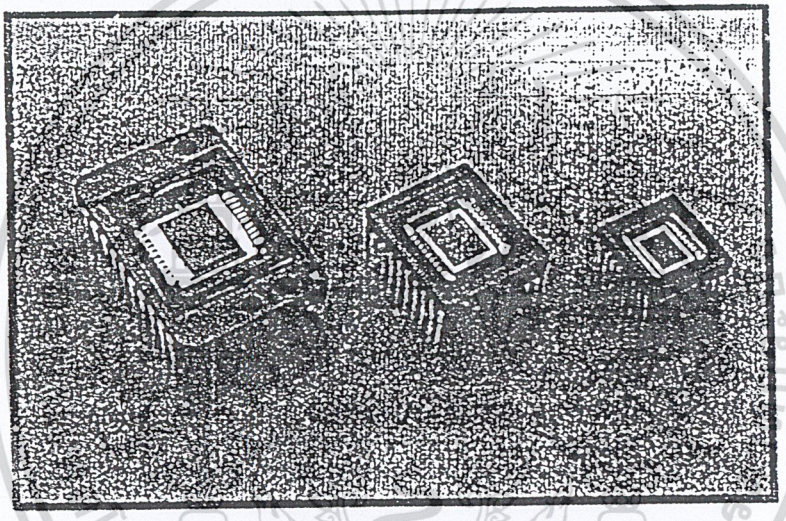
โครงงานรถสำรวจ มีระยะควบคุมไม่เกิน 20 เมตรจากเครื่องคอมพิวเตอร์ สามารถควบคุมรถสำรวจได้ในระยะไกลโดยมีกล้องติดอยู่กับตัวรถเพื่อที่จะส่งสัญญาณภาพมายังฐานควบคุมโดยที่ผู้ควบคุมสามารถบังคับความเร็วและทิศทางได้อย่างถูกต้อง

1.3 หลักการเบื้องต้น

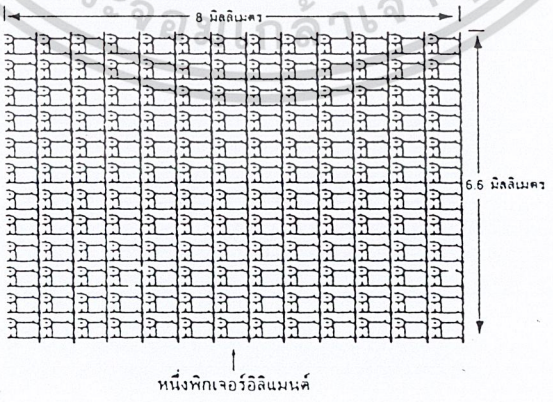
รถสำรวจเป็นการแสดงสัญญาณภาพที่ส่งจากตัวรถไปฐานควบคุม โดยผู้ควบคุมรถไม่ต้องออกมาติดตามรถที่อยู่นอกฐานโดยสามารถที่จะสังเกตุการเคลื่อนที่ได้โดยการมองภาพจากจอมอนิเตอร์ อีกทั้งยังสามารถประยุกต์ได้ในหลายลักษณะเช่น การตั้งเวลาในการบันทึกภาพเคลื่อนไหวเพื่อใช้ในการรักษาความปลอดภัยเป็นต้น

บทที่ 2
ระบบรับและแสดงรับและแสดงภาพ

ในส่วนประกอบของระบบการรับและแสดงภาพนั้นส่วนที่ขาดไม่ได้นั่นคือ กล้องโทรทัศน์ดิจิตอล และเครื่องคอมพิวเตอร์ที่ใช้ในการแสดงผลภาพ ดังนั้นจึงควรทำความเข้าใจกับหลักการพื้นฐานของกล้องโทรทัศน์ดิจิตอล (Charge Coupled Device หรือเรียกสั้นๆว่า CCD) ส่วนในการนำสัญญาณภาพโทรทัศน์จากกล้องดิจิตอลมาแสดงบนหน้าจอคอมพิวเตอร์นั้นต้องอาศัยซอฟแวร์ในการติดต่อ สัญญาณจากกล้องให้ติดต่อกับคอมพิวเตอร์ได้แล้วจึงนำภาพมาแสดงได้ ดังนั้นหลักการพื้นฐานของระบบรับและแสดงภาพสามารถที่จะแบ่งได้เป็นสองอย่าง คือ การทำงานของกล้องโทรทัศน์ดิจิตอล และ การทำงานทางซอฟแวร์ที่ใช้แสดงผลภาพโทรทัศน์



รูปที่ 2.1 (ก) กล้องโทรทัศน์ CCD ขนาดต่างๆ



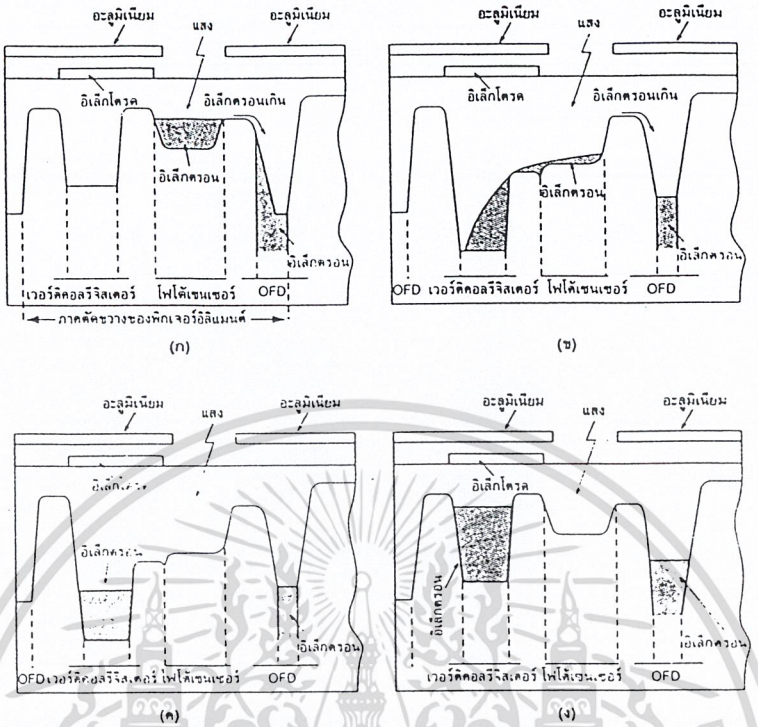
รูปที่ 2.1 (ข) ส่วนรับภาพและพิคเจอร์อิลีเมนต์

2.1 หลักการพื้นฐานของกล้องโทรทัศน์

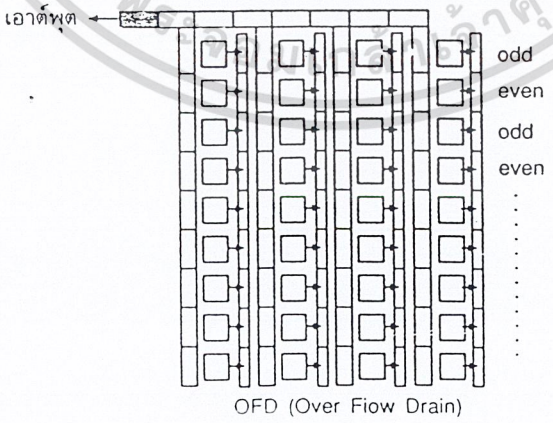
CCD หรืออุปกรณ์ Charge Couple Device ก็คือ VLSI (Very Large Scale IC) ที่บรรจุด้วยจุดรับภาพมากกว่า 250,000 จุด อยู่บนแผ่นตรวจจับแสงในพื้นที่ขนาดเล็กเพียง 8.8 x 6.6 ตารางมิลิเมตร เป็นผลจากการพัฒนาของเทคโนโลยีทางด้านอุปกรณ์สารกึ่งตัวนำ CCD จะให้อิเล็กตรอนเมื่อมีแสงสว่างตกกระทบตัวมัน ปริมาณอิเล็กตรอนที่เกิดขึ้นจะเปลี่ยนแปลงตามสัดส่วนกับความเข้มของแสงที่ได้รับ โดยแบ่งการทำงานภายใน CCD เป็น 3 ขั้นตอนดังนี้

- กำเนิดอิเล็กตรอนจากแสงที่ได้รับ (Detect Incident Light) จุดรับภาพแต่ละจุดจะให้กำเนิดอิเล็กตรอนตามความเข้มของแสงที่ได้รับซิลค์อะลูมิเนียมของจุดรับภาพ ตามรูปที่ 2.2 จะทำหน้าที่บังแสงไม่ให้ตกไปยังส่วนอื่นๆ ยกเว้นบริเวณตัวตรวจจับแสง
- เก็บรักษาอิเล็กตรอน (Store Induct Charge) ค่าความต่างศักดิ์ ดังแสดงในรูปที่ 2.2 จะอยู่ภายในของจุดรับภาพแต่ละจุด อิเล็กตรอนที่ถูกเก็บอยู่ในตัวจับแสงจำนวนมากจะไหลลงสู่ OFD (Over Flow Drain) และขจัดปัญหาการเกิดหางของภาพ (Becoming) ซึ่งเกิขึ้นเสมอสำหรับกล้องชนิดหลอดด้วยวิธีนี้จะสามารถรักษาระดับสูงสุดของจำนวนอิเล็กตรอนให้คงที่แม้ว่าจะมีแสงสว่างมากเกินไป
- ถ่ายเทอิเล็กตรอน ขณะที่มีแรงดันป้อนให้กับอิเล็กโตรดของจุดรับภาพส่วนลึกของแต่ละแรงดันภายใต้อิเล็กโตรดจะเพิ่มขึ้น อิเล็กตรอนที่ถูกเก็บไว้จะเริ่มถ่ายเทลงมาให้กับรีจิสเตอร์ในแนวตั้ง (Vertical Register) ดังรูปที่ 2.2 (ก) ซึ่งเปรียบเสมือนประตูเขื่อนถูกเปิดออก ขณะที่อิเล็กตรอนถูกถ่ายความลึกของแต่ละแรงดันจุดน้อยลงและกลับคืนค่ามาเหมือนเดิม การถ่ายเทอิเล็กตรอนจากโฟโตเซ็นเซอร์ไปสู่รีจิสเตอร์ในแนวตั้งได้สิ้นสุดลง ในระหว่างการถ่ายเทอิเล็กตรอนภายในอยู่ จะยังไม่มีการกำเนิดอิเล็กตรอนใหม่ ถึงแม้ว่าตัวตรวจจับแสงกำลังรับแสงอยู่ก็ตาม เป็นเพราะว่าความเร็วในการถ่ายเทนั้นสูงกว่าการเกิดอิเล็กตรอน การถ่ายเทอิเล็กตรอนของ CCD นี้สามารถตัดปัญหาการเผาไหม้และภาพที่มีลักษณะเป็นดาวหางได้

ทั้ง 3 ขั้นตอนนั้นเป็นหลักการทำงานพื้นฐานของตัวรับตัวรับภาพแบบ CCD (CCD Image Sensor) คือขั้นตอนแรกรับแสงและกำเนิดอิเล็กตรอนและเก็บมันเอาไว้ และขั้นตอนสุดท้ายอิเล็กตรอนที่ถูกเก็บไว้นั้นจะถูกถ่ายเทผ่านรีจีสเตอร์ในแนวตั้ง (Vertical Shift Register) และรีจีสเตอร์ในแนวระดับ (HoriZontal Shift Register) แล้วส่งไปวงจรเอาท์พุทที่เรียกว่า อินเตอร์ไลน์ทรานสเฟอร์ซีซีดี อิเมจเซนเซอร์ (Interling Transter CCD Imgage Sensor)



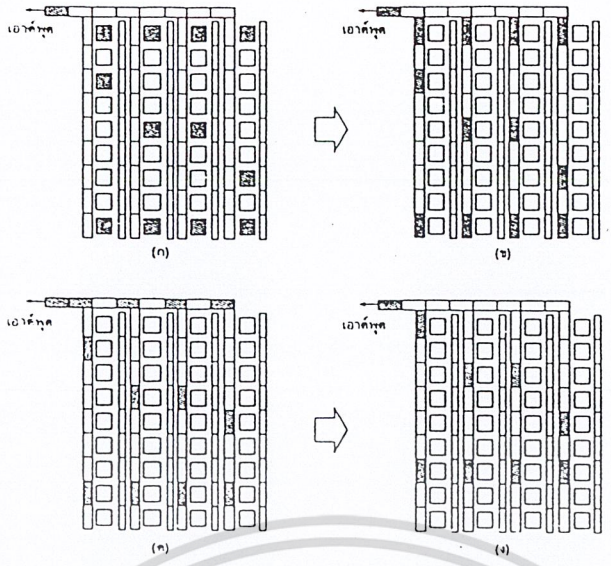
รูปที่ 2.2 ภายใน CCD ฟิกเจอร์อิลีเมนต์จะมีอิเล็กตรอนเกิดขึ้นและถ่ายเทไปทาง OFD



รูปที่ 2.3 ทิศทางการถ่ายเทของอิเล็กตรอนใน OFD หรือ Overflow Drain

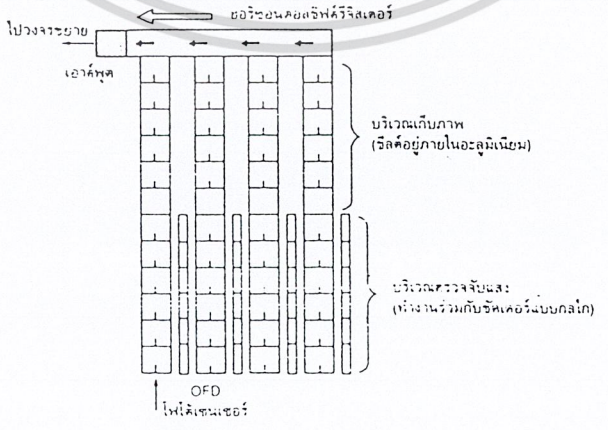
บริษัทโซนี่ได้ทำการอินเตอร์ไลน์ทรานสเฟอร์ให้กับตัวรับภาพ CCD รูปที่ 2.5 แสดงภาพโครงสร้างของอินเตอร์ไลน์ทรานสเฟอร์ CCD อิเมจเซ็นเซอร์ เพื่อให้เห็นการฉายแสงลงบนพื้นที่ CCD นั้นเหมือนกับปรากฏบนจอมอนิเตอร์จากรูปรีจีสเตอร์ในแนวตั้งและ OFD ได้เชื่อมต่อประสานกับรีจีสเตอร์ในแนวระดับ ข้อพิเศษทางเทคนิคของโครงสร้าง CCD แบบนี้ ก็คือลดอาการของภาพที่มีสีเลอะ (Smear) หรือเป็นหิมะที่เกิดขึ้นถูกถ่ายเทไปยังรีจิสเตอร์ซึ่งเวอร์ติคอลชีฟต์รีจิสเตอร์และฮอลริซอนตอนชีฟรีจิสเตอร์เหมือนกับถนนและอิเล็กตรอนทู่กถ่ายเทก็เหมือนรถที่วิ่งผ่านถนน เคลื่อนที่ไปย่างมีลำดับต่อเนื่องกันตลอด ตามรูปที่ 2.4 ไปสู่ภาคขยายอินเตอร์ไลน์ทรานสเฟอร์ CCD ดีกว่าอุปกรณ์อื่นๆ อย่างไรเมื่อเปรียบเทียบระหว่างอินเตอร์ไลน์ทรานสเฟอร์ CCD อิเมจเซ็นเซอร์มีโครงสร้างที่ง่ายกว่า แต่มีข้อเสียอยู่หลายประการดังนี้คือ

1. มีขนาดใหญ่ทำให้ต้นทุนการผลิตสูง
2. เนื่องจากการถ่ายเทอิเล็กตรอนระหว่างโฟโตเซ็นซิ่งกับสตอเรจเป็นไปด้วยความรวดเร็วมาก ดังนั้นจึงทำให้รายละเอียดทางด้านแนวตั้ง (Vertical) ลดลงซึ่งเปรียบเสมือนกับคนที่หิ้วน้ำเต็มถังแล้ววิ่งด้วยความเร็ว น้ำย่อมจะมีการกระฉอกออกไปบ้าง ทำให้ประสิทธิภาพที่ได้เสียไป
3. ในระหว่างการถ่ายเทอิเล็กตรอนโฟโตเซ็นซิ่งรับหน้าที่เป็นรีจีสเตอร์แบบกลไก (Mechanical Shutter) ใช้เพื่อตัดแสงในช่วงการถ่ายเทอิเล็กตรอน มิฉะนั้นภาพจะเลอะเลือน อย่างไรก็ตามชัตเตอร์แบบกลไก ทำให้เกิดปัญหาหลายประการ เช่น ทำให้การผลิตยากขึ้น ความแน่นอนในการทำงานต่ำลง และเกิดนอยส์เป็นต้น รูปที่ 2.6 เป็นรูปของเฟรมทรานสเฟอร์อิเมจเซ็นเซอร์ ข้อได้เปรียบระหว่างกล้อง CCD แบบ 3 ชิบเหนือกว่ากล้อง 3 หลอดในเรื่องความไวแสงและภาพเกิดรอยไหม้ (Image Burning) น้อย ใช้กำลังไฟน้อย และมีความทนทานแต่กล้อง 3 หลอดก็มีข้อดีตรงที่มีความชัดเจนสูงกว่า ดังนั้นกล้อง CCD แบบ 3 ชิบจึงเหมาะกับงานประเภทงานที่ถ่ายภาพนอกสถานที่ ซึ่งต้องการกล้องที่มีความต้านต่อการเผาไหม้สูง มีความไวและความทนทานเข็งแรงเป็นต้นส่วนกล้อง 3 หลอด CCD นั้นจะใช้เมื่อต้องการคุณภาพของภาพเท่านั้น

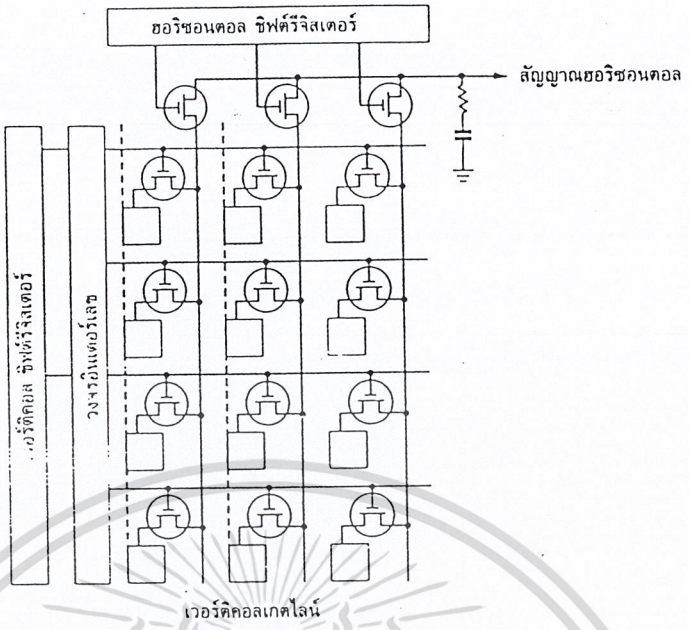


รูปที่ 2.4 การถ่ายเทอิเล็กตรอนโดยผ่านทางรีจอสเตอร์ในแนวตั้งและรีจอสเตอร์ในแนวระดับ

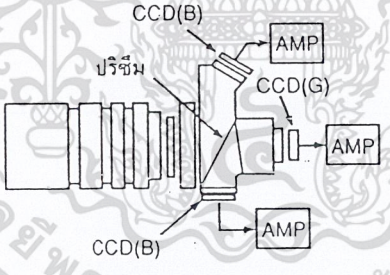
รูปที่ 2.5 การถ่ายเทอิเล็กตอรนจะทำให้เกิดมีอิเล็กตรอนสูญเสีย



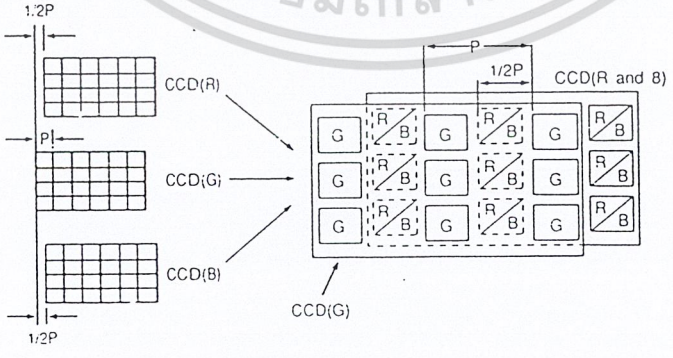
รูปที่ 2.6 โครงสร้างของ CCD แบบเฟรมทรานสฟอร์ม



รูปที่ 2.7 โครงสร้างของตัวรับแบบ MOS



(ก) ตำแหน่งของ CCD ภายในกล้อง



(ข) CCD สีแดงและสีน้ำเงินจะมีระบบต่างกับ CCD สีเขียว 1/2 พิคซ์

รูปที่ 2.8 เทคโนโลยีชดเชย Original Spatial Effect ของโซนี่

ข้อเปรียบเทียบระหว่างกล้อง CCD แบบ 3 ชิพกับกล้องแบบ MOS ทั่วไป

2.1.1 ตัวรับภาพแบบมอส (MOS Image Sensor)

ตัวรับภาพแบบมอสจะใช้ทรานซิสเตอร์แบบมอสเป็นตัวสวิทชิ่งคล้ายกับหน่วยความจำแบบ DRAM ดังรูปที่ 2.7 แม้ว่าอุปกรณ์ชนิดนี้จะสามารถเก็บประจุได้เป็นจำนวนมาก แต่มันก็ทำให้เกิดสัญญาณรบกวนมากเช่นเดียวกันและตัวรับภาพ CCD ก็มีคุณสมบัติเหนือกว่ามากในด้านของความไวแสง (Sensitivity) และนอยส์ (Fixed Pattern Noise) โดยใช้เทคโนโลยี Original Spatial Effect ของโซนี่โดยการติดตั้ง CCD สีแดงและสีน้ำเงินให้มีระยะแตกต่างจากสีเขียว ½ พิตช์ (Pitch) ดังรูปที่ 2.8 ด้วยเทคนิคอันนี้จะทำให้ได้พิกเจอร์อิลีเมนต์เป็น 2 เท่าและทำให้ได้ความชัดเจนของภาพสูงสุด

2.1.2 ประโยชน์และคุณภาพของกล้อง CCD มีดังต่อไปนี้

1. ภาพที่ได้มีการสูญเสียน้อย
2. มีความไวสูง
3. มีน้ำหนักเบา
4. ให้ภาพที่ละเอียดและคมชัด
5. กล้องไม้เป็นรูปรอยไหม้เมื่อถูกแสงอาทิตย์
6. ไม่เกิดภาพล้า (No Lag)
7. ไม่เกิดภาพเป็นรูปดาวหาง
8. เมื่อสัญญาณแสงต่ำไม่ต้องตั้งปรับ (No Registraion)

ปัจจุบันกล้องโทรทัศน์ส่วนใหญ่จะใช้ CCD เป็นตัวรับภาพ จึงควรทราบหลักการทำงานและโครงสร้างโทรทัศน์ CCD หากต้องการศึกษาลงไปถึงระดับโครงสร้างของสารกึ่งตัวนำก็สามารถศึกษาเพิ่มเติมได้จากแหล่งข้อมูลทางอินทอร์เน็ต

บทที่ 3
พอร์ตขนาน

การที่จะนำเอาสัญญาณอะนาลอกจากตัวตรวจจับมาแสดงปริมาณการเปลี่ยนแปลงทางฟิสิกส์อาทิ ความดัน , อุณหภูมิ , ความเข้มของแสงน้ำหนัก , สถานะสวิตช์ ,การเคลื่อนไหว เป็นต้น หรือนำมาบันทึกไว้แล้วส่งออกไปเพื่อใช้ในการควบคุมทางฟิสิกส์ ซึ่งอุปกรณ์ที่ว่าหมายถึง มอเตอร์ ,ไซเรน,รีเลย์และสเต็บมอเตอร์ การที่จะทำเช่นนี้ได้ ต้องผ่านกระบวนการแปลงข้อมูลทางฟิสิกส์ที่เป็นข้อมูลอะนาลอกไปเป็นข้อมูลดิจิตอล เพื่อให้สามารถติดต่อกับคอมพิวเตอร์ได้และข้อมูลสามารถนำไปแสดงผลบนจออมพิวเตอร์หรือบันทึกไว้ในหน่วยความจำของคอมพิวเตอร์ ส่วนการจะนำข้อมูลออกมาเพื่อควบคุมอุปกรณ์ทางฟิสิกส์ ต้องเปลี่ยนข้อมูลดิจิตอลไปเป็นข้อมูลอะนาลอก การติดต่อระหว่างฮาร์ดแวร์และเครื่องคอมพิวเตอร์ เรียกว่าการอินเตอร์เฟส การอินเตอร์เฟสสามารถทำได้หลายวิธี

- เชื่อมต่อผ่านการ์ดอินพุทเอาท์พุท ซึ่งใช้วิธีติดตั้งการ์ดลงในสล๊อตภายในเครื่องคอมพิวเตอร์
- เชื่อมต่อผ่านพอร์ตทอนุกรม
- เชื่อมต่อผ่านพอร์ตขนาน
- เชื่อมต่อผ่านระบบมาตฐานอื่นๆ เช่น พอร์ต USB(Universal Serial Bus),พอร์ต SCSI หรือพอร์ต GAME เป็นต้น

3.1 พอร์ตและการใช้งาน

ในการเขียนโปรแกรมเพื่อใช้งานพอร์ตต่างๆ ภายในเครื่องคอมพิวเตอร์จำเป็นที่ต้องรู้ถึงตำแหน่งการอ้างถึงของอุปกรณ์อินพุท/เอาท์พุทที่ต้องการติดต่อด้วย ในการอ้างตำแหน่งพอร์ตจะถูกออกแบบให้ใช้สายสัญญาณแอดเดรสบัสเพียง 10 เส้น คือ A0-A9 ดังนั้นจึงสามารถอ้างตำแหน่งพอร์ตได้ถึง 1024 พอร์ท ซึ่งจะแบ่งออกเป็น 2 กลุ่ม คือพอร์ตที่มีแอดเดรสอยู่ในช่วง 000H-0FFH จะใช้งานบนเมนบอร์ดสำหรับชิพซัพพอร์ทเท่านั้น เช่น 8259 (Interrupt Controller) , 8237 (DMA Controller) , 8253 (Timer&Counter) และกลุ่มที่มีแอดเดรสอยู่ในช่วง 100H-3FFH จะใช้งานกับการ์ดขยายต่างๆ ที่เสียบในสล๊อต

หมายเลขพอร์ต	การใช้งาน
000H-01FH	ตัวควบคุม DMA 1 ,8237A-5
020H-021H	Interrupt mask register
022H-03FH	(ห้ามใช้) ตัวควบคุม 8259
040H-043H	เวลา /และตัวนับเวลา
060H	ลำโพง (บิท 0 และ บิท 1)
060H-06FH	ตัวควบคุมพอร์ตขนานและคีย์บอร์ด 8255
070H-07FH	Real Time Clock,Nmi ของระบบ
080H-09FH	DMA page register 74LS162
0A0H-0BFH	ตัวควบคุมอินเตอร์รัพพ์ 2 8259 (สเลฟ)
0F0H	เคลียร์แมทโครโปรเซสเซอร์
0F1H	รีเซ็ทแมทโครโปรเซสเซอร์
0F8H-0FFH	แมทโครโปรเซสเซอร์
1F0H-1F8H	ฮาร์ดดิสก์
200H-207H	เกมอินพุท/เอ้าท์พุท
278H-27FH	เครื่องพิมพ์ขนานพอร์ต 2
2F8H-2FFH	เครื่องพิมพ์อนุกรมพอร์ต 2
300H-31FH	การ์ดโปรโตไทป์(prototype)
360H-36FH	สงวนไว้
378H-37FH	เครื่องพิมพ์ขนานพอร์ต 1
380H-38FH	SDLC, ไบต์ซิงโครไนซ์ 2
3A0H-3AFH	ไบต์ซิงโครไนซ์ 1
3B0H-3BFH	อะแดปเตอร์โมโนโครมและเครื่องพิมพ์
3C0H-3CFH	สงวนไว้
3D0H-3DFH	อแดปเตอร์สีและกราฟฟิก
3F0H-3F7H	ตัวควบคุมดิสก์ไดร์ฟ
3F8H-3FFH	พอร์ตอนุกรม1

ตารางที่ 3.1 แสดงการใช้งานพอร์ตที่ตำแหน่งต่างๆ

จากตาราง 3.1 ถ้าในระบบของเครื่องคอมพิวเตอร์ไม่มีการใช้งานอุปกรณ์นั้น เราสามารถนำแอดเดรสของพอร์ตนั้นมาใช้งานได้ ในโครงงานนี้เลือกใช้พอร์ตขนาน เพราะมีข้อดีคือ

1. ในด้านความปลอดภัย เนื่องจากไม่ต้องถอดฝาเครื่องคอมพิวเตอร์เพื่อติดตั้งการ์ดใดๆ
2. ในด้านเข้ากันได้กับเครื่องคอมพิวเตอร์ส่วนใหญ่ เพราะพอร์ตขนานจะมีติดตั้งในเครื่อง คอมพิวเตอร์ทุกเครื่อง แม้กระทั่งเครื่องคอมพิวเตอร์โน้ตบุ๊กซึ่งไม่มีสล็อตในการติดตั้งการ์ดเหมือนเครื่องพีซี
3. สะดวกในการใช้งานและประหยัดพื้นที่ เพราะไม่ต้องเพิ่มการ์ดใดๆ
4. จำนวนช่องสัญญาณอินพุท/เอาท์พุท โดยปกติพอร์ตขนานจะมีอินพุท 5 ขา และเอาท์พุท 12 ขา ซึ่งเพียงพอที่จะนำไปใช้งานต่างๆ อีกทั้งสามารถขยายจำนวนพอร์ตเพิ่มขึ้นได้
5. ความเร็วในการสื่อสารข้อมูล พอร์ตขนานจะมีความเร็วเท่ากับการติดต่อระบบบัสโดยตรง และมีความเร็วมากกว่าการติดต่อผ่านพอร์ตอนุกรม
6. อะไหล่และชิ้นส่วนประกอบ คอนเน็กเตอร์และสายเชื่อมต่อต่างๆ ของการเชื่อมต่อผ่านพอร์ตขนานสามารถหาได้ง่ายและราคาไม่แพง

3.2 ข้อมูลเบื้องต้นเกี่ยวกับพอร์ตขนาน

พอร์ตขนาน(Parallel port) บางครั้งเรียกว่า พอร์ตเครื่องพิมพ์ (Printer port) พอร์ตขนานจะมีการถ่ายทอดข้อมูลของพอร์ตเป็นแบบขนาน มีความเร็วในการติดต่อข้อมูลเท่ากับการติดต่อกับระบบบัสโดยตรง มีอัตราการถ่ายทอดข้อมูลสูงกว่าการถ่ายทอดข้อมูลแบบอนุกรมประมาณ 8-10 เท่า และมีการประมวลผลข้อมูลขนาด 8 บิท ตำแหน่งของอินพุท/เอาท์พุทของพีซี (PC I/O) จะทำการจับจองเนื้อที่สำหรับพอร์ตขนานเอาไว้ 3 ย่าน คือ

Address	Notes:
3BCh - 3BFh	Used for Parallel Ports which were incorporated in to Video Cards and now, commonly an option for Ports controlled by BIOS. - Doesn't support ECP addresses.
378h - 37Fh	Usual Address For LPT 1
278h - 27Fh	Usual Address For LPT 2

ตารางที่ 3.2 แสดงการจับจองเนื้อที่สำหรับพอร์ตขนาน

มาตฐานของพอร์ตขนานจะมีรีจิสเตอร์อยู่ 3 ตัว รีจิสเตอร์ทั้งสามจะเข้าจับจองเนื้อที่รีจิสเตอร์ละแอดเดรสของทั้งสามย่านพอร์ตขนาน แอดเดรสแรกจะถูกเรียกว่าแอดเดรสหลัก (Base address) ซึ่งรีจิสเตอร์ทั้งสามประกอบด้วย

Address	Port Name	Read/Write
Base + 0	Data Port (SPP)	Write
Base + 1	Status Port (SPP)	Read
Base + 2	Control Port (SPP)	Write
Base + 3	Address Port (EPP)	Read/Write
Base + 4	Data Port (EPP)	Read/Write
Base + 5	Undefined (16/32bit Transfers)	-
Base + 6	Undefined (32bit Transfers)	-
Base + 7	Undefined (32bit Transfers)	-

ตารางที่ 3.3 แสดงตำแหน่งของอินพุท/เอาท์พุทของพีซี (PC I/O) สำหรับพรับพอร์ตขนาน

1. ดาต้ารีจีสเตอร์ (Data register) อยู่ตำแหน่งแอดเดรสหลัก (Base address) ทำหน้าที่ เขียนอย่างเดียว

Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write (Note-1)	Bit 7	Data 7 (Pin 9)
			Bit 6	Data 6 (Pin 8)
			Bit 5	Data 5 (Pin 7)
			Bit 4	Data 4 (Pin 6)
			Bit 3	Data 3 (Pin 5)
			Bit 2	Data 2 (Pin 4)
			Bit 1	Data 1 (Pin 3)
			Bit 0	Data 0 (Pin 2)

ตารางที่ 3.4 แสดงดาต้ารีจีสเตอร์

2. สเตตัสรีจีสเตอร์ (Status registers) อยู่ตำแหน่งแอดเดรสหลักบวกหนึ่ง (Base address+1) ทำหน้าที่อ่านอย่างเดียว

Offset	Name	Read/Write	Bit No.	Properties
Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

ตารางที่ 3.5 แสดงสเตตัสรีจีสเตอร์

3.คอนโทรลรีจีสเตอร์ (Control register) จะอยู่ตำแหน่งแอดเดรสหลักบวกสอง (Base address+2) ทำหน้าที่อ่านและเขียน

Offset	Name	Read/Write	Bit No.	Properties
Base + 2	Control	Read/Write	Bit 7	Bit 7 Unused
			Bit 6	Unused
			Bit 5	Enable bi-directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

ตารางที่ 3.6 แสดงคอนโทรลรีจีสเตอร์

3.3 การตรวจสอบพอร์ตขนาน

ทุกครั้งระหว่างที่ทำการเปิดเครื่อง(Boot-up) BIOS จะทำการตรวจสอบอุปกรณ์ที่ต่อไว้กับพอร์ตขนานว่าอยู่ที่ตำแหน่งไหน พอร์ตแรกจะถูกกำหนด ให้เป็น LPT1 และพอร์ตที่สองจะถูกกำหนดให้เป็น LPT2 ไปเรื่อยๆ ซึ่ง BIOS จะเก็บตำแหน่งแอดเดรสหลักของแต่ละพอร์ตดังต่อไปนี้

Start Address	Function
0000:0408	LPT1's Base Address
0000:040A	LPT2's Base Address
0000:040C	LPT3's Base Address
0000:040E	LPT4's Base Address (Note 1)

ตารางที่ 3.7 แสดงตำแหน่งพอร์ตขนาน

การตรวจสอบสามารถตรวจสอบโดยใช้โปรแกรม Debug ป้อนคำสั่ง –d 40:0008 ซึ่งเป็นตำแหน่งแรกของพอร์ต

```
Microsoft(R) Windows 98
   (C)Copyright Microsoft Corp 1981-1999.

C:\WINDOWS>cd command

C:\WINDOWS\COMMAND>debug
-d 40:0008
0040:0000                                78 03 00 00 00 00 1E 02   x.......
0040:0010  27 C4 00 80 02 80 00 20-00 00 2E 00 2E 00 34 45   '.....  ......4K
0040:0020  30 52 3A 27 30 52 30 52-30 52 30 48 0D E0 64 20   0R:'0R0R0R0H..d
0040:0030  65 12 62 30 75 1E 67 22-0D 1C 64 20 20 39 00 80   e.b0u.g"..d  9..
0040:0040  00 00 C0 00 00 00 00 00-00 03 50 00 00 14 00 00   ..........P.....
0040:0050  00 0E 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:0060  0E 0D 00 D4 03 29 30 A4-17 05 85 FF 04 00 01 00   .....)0.........
0040:0070  00 00 00 00 00 02 00 00-14 14 14 3C 01 01 01 01   ...........<....
0040:0080  1E 00 3E 00 18 19 00 60                           ..>....`
-q
```

รูปที่ 3.1 แสดงการตรวจสอบตำแหน่งพอร์ตขนาน

จากรูปที่ 3.1 ข้อมูลที่วงกลมไว้ คือ 378H เป็นค่าแอดเดรสหลักของ LPT1 ค่าแอดเดรสที่มีค่าเป็นศูนย์แสดงว่า พอร์ตของ LPT หมายเลขนั้นไม่สามารถติดต่อได้ จะเห็นว่าเครื่องพีซีเครื่องนี้ไม่มี LPT2-4 หรือสามารถตรวจตำแหน่งของพอร์ตได้จากการเขียนโปรแกรมตรวจสอบพอร์ต (TestPort.cpp)

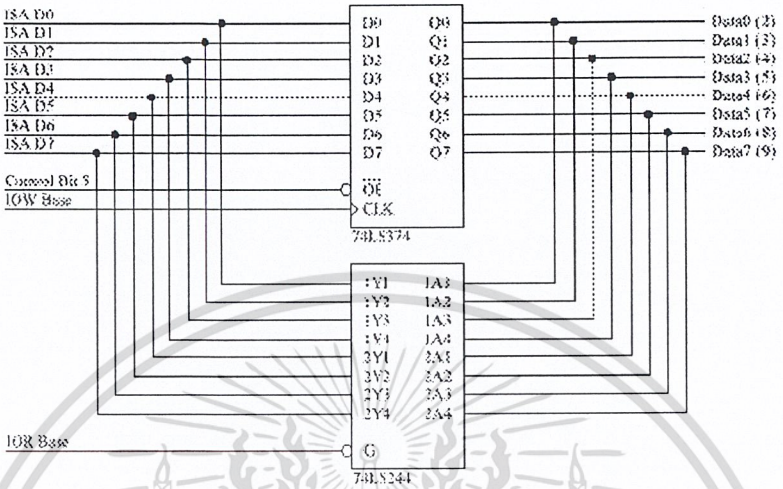
3.4 การใช้งานพอร์ตขนาน

สำหรับพอร์ตขนานแบบมาตฐาน การอ่านข้อมูล ผู้ใช้สามารถนำพอร์ตอินพุทจากพอร์ตสเตตัสและพอร์ตคอนโทรลมาใช้งานรับค่าข้อมูล ส่วนการเขียนข้อมูล สามารถนำพอร์ตดาต้าร่วมกับพอร์ตคอนโทรลได้(พอร์ตคอนโทรลสามารถเป็นได้ทั้งพอร์ตอินพุทและเอ้าพุท) ตำแหน่งของสายสัญญาณบนพอร์ตขนานจะมีหน้าที่และทิศทาง (Direction) ดังตารางที่ 3.8

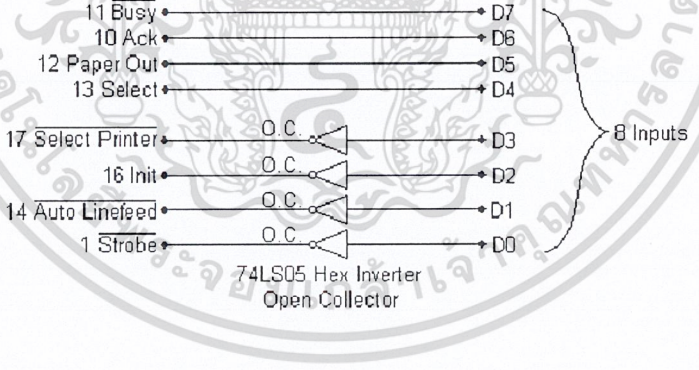
Pin No (DType25)	**Pin No (Centronics)**	SPP Signal	**Direction In/out**	**Register Hardware**
1	1	nStrobe	In/Out	Control
2	2	Data 0	Out	Data
3	3	Data 1	Out	Data
4	4	Data 2	Out	Data
5	5	Data 3	Out	Data
6	6	Data 4	Out	Data
7	7	Data 5	Out	Data
8	8	Data 6	Out	Data
9	9	Data 7	Out	Data
10	10	nAck	In	Status
11	11	Busy	In	Status
12	12	Paper-Out PaperEnd	In	Status
13	13	Select	In	In
14	14	nAuto-Linefeed	In/Out	Control
15	32	nError / nFault	In	In
16	31	nInitialize	In/Out	Control
17	36	nSelect-Printer	In/Out	Control
18-25	19-30	Ground	Gnd	

ตารางที่ 3.8 แสดงข้อมูลสายสัญญาณของพอร์ตขนาน

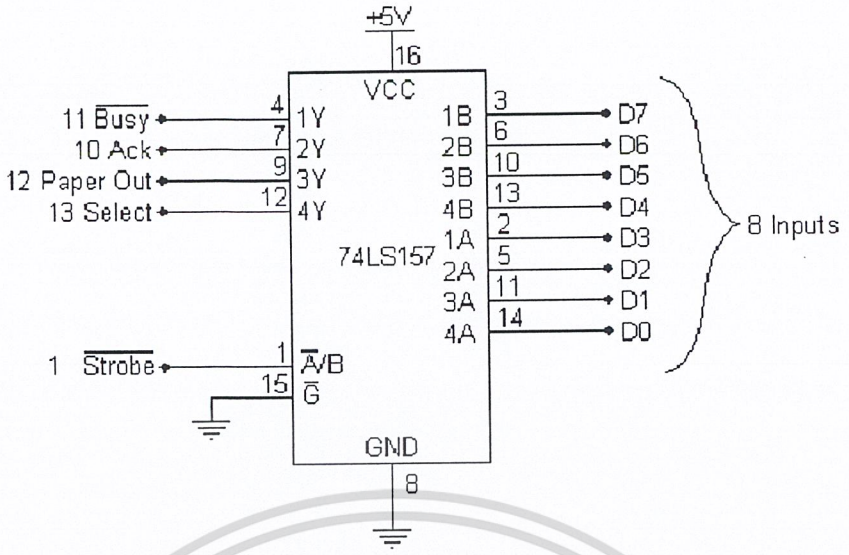
ตัวอย่างการต่อใช้งานพอร์ตขนาน



รูปที่ 3.2 แสดงการต่อใช้งานพอร์ตขนานเพื่อส่งข้อมูล



รูปที่ 3.3 แสดงการต่อใช้งานพอร์ตขนานเพื่อรับข้อมูลขนาด 8 บิท



รูปที่ 3.4 แสดงการต่อใช้งานพอร์ตขนานเพื่อรับข้อมูลขนาด 8 บิท แบบมัลติเพล็ก

3.5 การเขียนโปรแกรมติดต่อกับพอร์ตขนาน

พอร์ตขนานจะมีลักษณะเช่นเดียวกับอุปกรณ์อินพุทเอาท์พุทตัวอื่นๆ คือเมื่อต้องการติดต่อจะต้องกำหนดแอดเดรสที่ต้องการติดต่อด้วย เมื่อต้องการติดต่อกับพอร์ตขนานในตำแหน่งนั้นๆ ยกตัวอย่างการเขียนโปรแกรมเพื่อส่งค่าลอจิก "1" ออกไปทุกบิทของพอร์ต Data ของ LPT1 และอ่านค่าจากพอร์ตขนานมายังคอมพิวเตอร์ผ่านทางพอร์ต Status ของ LPT1 สามารถเขียนโปรแกรมดังนี้

3.5.1 ภาษา QBASIC

การส่งค่าข้อมูลออกไปยังพอร์ตขนาน

```
OUT &H378,&HFF
```

โดยที่ &H ที่แสดงนั้นหมายถึงตัวเลขฐานสิบหก

378 เป็นแอดเดรสของรีจีสเตอร์ Data สำหรับ LPT1

FF เป็นข้อมูลฐานสิบหกมีลอจิก "11111111" ในฐานสอง

การอ่านค่าข้อมูลจากพอร์ตขนาน

Temp=INP(&H379)

โดยที่ INP(_) เป็นคำสั่งสำหรับการอ่านข้อมูล

379 เป็นตำแหน่งแอดเดรสของรีจีสเตอร์ STATUS สำหรับ LPT1 ในตัวเลขฐานสิบหก

Temp เป็นตัวแปรที่ใช้เก็บข้อมูลที่อ่านได้จากพอร์ตขนาน

3.5.2 ภาษา ASSEMBLY

การส่งค่าข้อมูลออกไปยังพอร์ตขนาน

```
mov dx,378h
mov al,ffh
out dx,al
```

การอ่านค่าข้อมูลจากพอร์ตขนาน

```
mov dx,379h
in al,dx
```

3.5.3 ภาษา PASCAL

การส่งค่าข้อมูลออกไปยังพอร์ตขนาน

Port[378H]:=FFH

การอ่านค่าข้อมูลจากพอร์ตขนาน

```
Temp :=Port[379H]
```

บทที่ 4
การใช้งาน MCS –51

4.1 จัดขาของไมโครคอนโทรลเลอร์ 8051

V_{cc} : สำหรับแหล่งจ่ายไฟฟ้า (+5v.)

V_{ss} : สำหรับต่อกราวด์

P0 : เป็นขาพอร์ต 0 ของ 8051 ที่มีขนาด 8 บิตชนิดสองทิศทาง ซึ่งแต่ละบิตสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไปหากต้องการให้เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล “1” ไปยังบิตนั้น โดยแต่ละบิตเมื่อเป็นเอาต์พุตจะสามารถต่อพ่วงกับอุปกรณ์ TTL แบบ LS ได้ 8 ตัว และยังเป็นขาให้สัญญาณ Multiplex ระหว่างสัญญาณข้อมูลกับสัญญาณ Address 8 บิตแรก ในกรณีที่ใช้หน่วยความจำภายนอก

P1 : เป็นขาพอร์ต 1 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional ซึ่งแต่ละบิตสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไปหากต้องการให้เป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล “1” ไปยังบิตนั้น และสามารถต่อพ่วงกับอุปกรณ์ LS TTL ได้ 4 ตัว

P2 : เป็นขาพอร์ต 2 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional เช่นเดียวกับพอร์ต 1 นอกจากนี้พอร์ต2นี้ยังทำหน้าที่ให้สัญญาณ Address 8 บิตบน ในกรณีที่ใช้หน่วยความจำภายนอก ในกรณีอ้าง Address หน่วยความจำขนาด 16 บิต ดังนั้นขณะที่ใช้หน่วยความจำภายนอก จะต้องไม่มีการเขียนข้อมูลใด ๆ ไปที่พอร์ต 2 จะทำให้เกิดความผิดพลาดการทำงานได้

P3 : เป็นขาพอร์ต 3 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional เช่นเดียวกับขาพอร์ต 1 และพอร์ต 2 แต่พอร์ต 3 นี้จะมีหน้าที่พิเศษดังตารางที่ 2.1

ขาพอร์ต	หน้าที่พิเศษ
P3.0	R x D (สำหรับรับข้อมูลแบบอนุกรม)
P3.1	T x D (สำหรับส่งข้อมูลแบบอนุกรม)
P3.2	INT0 (ขาอินเตอร์รัพท์ภายนอก 0)
P3.3	INT1 (ขาอินเตอร์รัพท์ภายนอก 1)
P3.4	T0 (ขาอินพุตของ Timer 0)
P3.5	T1 (ขาอินพุตของ Timer 1)
P3.6	WR (สำหรับสัญญาณเขียนหน่วยความจำข้อมูลภายนอก)
P3.7	RD (สำหรับสัญญาณอ่านหน่วยความจำข้อมูลภายนอก)

ตารางที่ 4.1 หน้าที่พิเศษของขาต่าง ๆ ของ PORT 3

ดังนั้น เมื่อมีการใช้สัญญาณดังกล่าว จึงไม่ควรเขียนข้อมูลไปที่พอร์ต 3 จะทำให้การทำงานของ 8051 ผิดพลาดได้

RST : เป็นขาสำหรับรีเซตการทำงานของ 8051 โดยการให้ลอจิกหนึ่งเป็นเวลาอย่างน้อย 2 ช่วง Machine Cycle

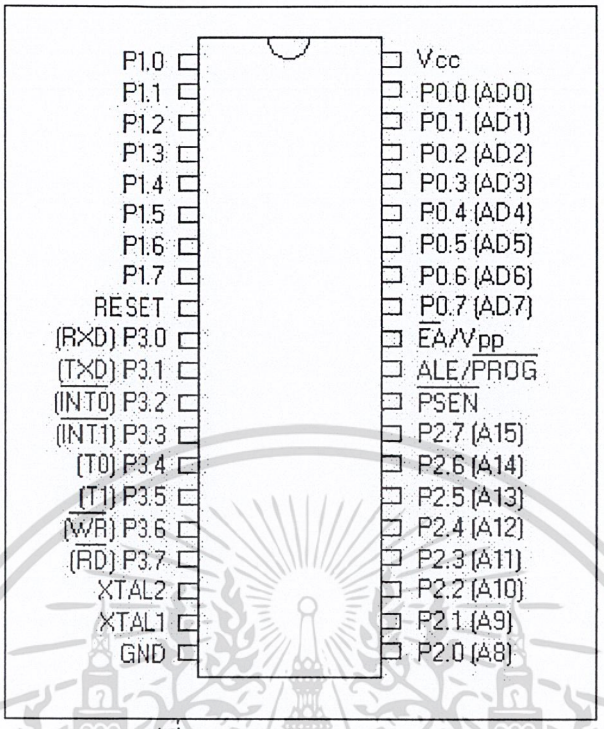
ALE : เป็นขาที่ใช้ในการควบคุมการแลตซ์ของขา พอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก

PSEN : เป็นขาสัญญาณเพื่อร้องขอติดต่อหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านหน่วยความจำโปรแกรมภายนอก

EA : เป็นขาใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมภายนอกหรือภายในไมโครคอนโทรลเลอร์ โดยที่ให้ลอจิก 0 จะอ่านหน่วยความจำโปรแกรมภายนอก และลอจิก 1 จะอ่านหน่วยความจำโปรแกรมภายใน

XTAL1 : ขาเข้าของวงจรกำเนิดความถี่อ้างอิงภายในของ 8051

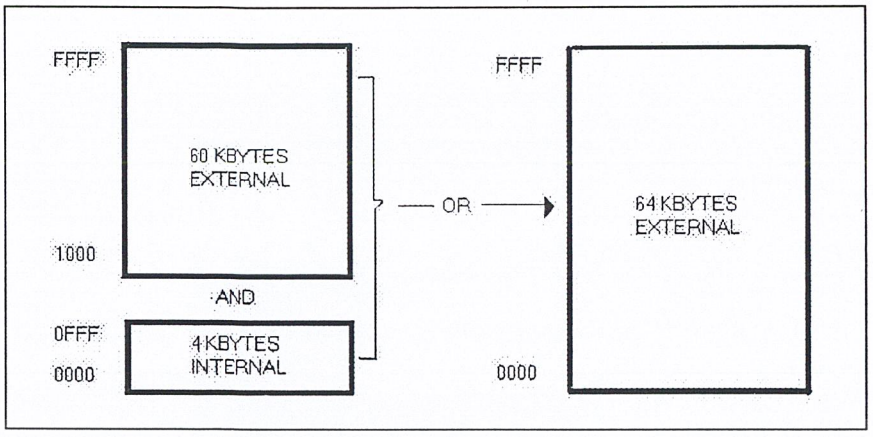
XTAL2 : ขาออกของวงจรกำเนิดความถี่อ้างอิงภายในของ 8051



รูปที่ 4.2 การจัดขาของ 8051

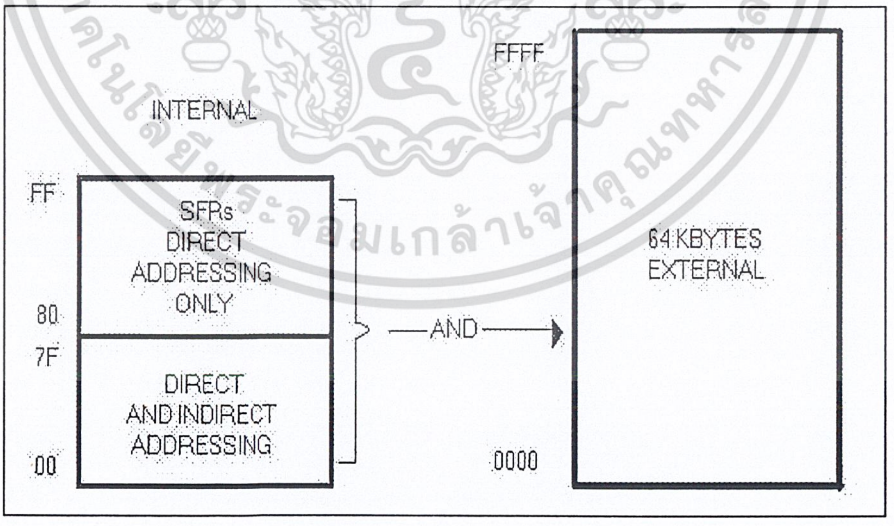
4.2 สร้างหน่วยความจำของ 8051

ดังที่กล่าวมาแล้ว 8051 จะแบ่งหน่วยความจำออกเป็นสองส่วน ได้แก่ หน่วยความจำสำหรับโปรแกรมและหน่วยความจำสำหรับเก็บข้อมูล โดยมีขนาดของแต่ละส่วนเท่ากับ 64 กิโลไบต์ ในส่วนของหน่วยความจำโปรแกรมจะเป็นส่วนหน่วยความจำสำหรับอ่านอย่างเดียว โดยที่ 8051 จะใช้สัญญาณ PSEN ในการอ่านเท่านั้น แต่หน่วยความจำข้อมูลของ 8051 จะสามารถอ่านและเขียนได้โดยใช้สัญญาณ RD และ WR ตามลำดับ แต่อย่างไรก็ตาม ผู้ใช้สามารถรวมหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลเข้าด้วยกันได้ โดยนำสัญญาณ RD และ PSEN มาต่อเข้าวงจร AND GATE สำหรับสร้างสัญญาณในการอ่านหน่วยความจำ นอกจากนี้หน่วยความจำโปรแกรมยังแบ่งออกเป็นภายนอกและภายในของ 8051 ดังแสดงในรูปที่ 4.2 รูปที่ 4.3 โดยรูปที่ 4.2 แสดงหน่วยความจำโปรแกรมในกรณีที่เลือกใช้หน่วยความจำภายนอกและภายใน ในด้านซ้ายมือเป็นส่วนหนึ่งของหน่วยความจำโปรแกรมภายในที่มีขนาด 4 กิโลไบต์ของ 8051 ส่วนที่เหลือจะเป็นหน่วยความจำภายนอก ส่วนด้านขวามือแสดงหน่วยความจำโปรแกรมเมื่อเลือกให้ติดต่อหน่วยความจำภายนอกทั้งหมด



รูปที่ 4.2 แสดงหน่วยความจำโปรแกรมของ 8051

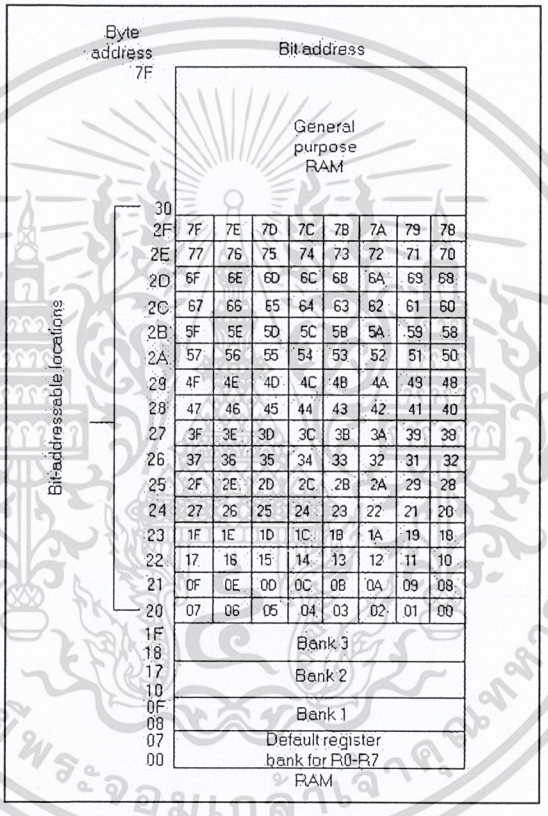
สำหรับหน่วยความจำข้อมูลของ 8051 สามารถแบ่งออกเป็นภายนอกและภายในโดยหน่วยความจำภายนอกแสดงไว้ด้านขวามือของรูปที่ 4.3 ซึ่งมีขนาด 64 กิโลไบต์ ส่วนหน่วยความจำข้อมูลภายในแสดงไว้ด้านซ้ายของรูปที่ 4.3 โดยหน่วยความจำภายในของ 8051 แบ่งออกเป็นสองส่วน ได้แก่ ส่วนของหน่วยความจำข้อมูลที่สามารถอ้างอิงแบบ Direct และ Indirect ซึ่งมีขนาด 128 ไบต์ กับหน่วยความจำที่อ้างอิงได้เฉพาะแบบ Direct หรือในส่วนนี้จะเรียกอีกแบบหนึ่งว่า SFR (Special Function Register) โดยจะแบ่งกล่าวได้ดังนี้



รูปที่ 4.3 แสดงหน่วยความจำข้อมูลของ 8051

ในส่วนของหน่วยความจำข้อมูลภายในที่อ้างอิงแบบ direct และ Indirect นั้นจะสามารถแบ่งออกได้ 3 ส่วน ดังแสดงในรูปที่ 4.4 โดยมีรายละเอียดดังนี้

- ส่วนที่ 1 เรียกว่า Register Banks 0-3 ซึ่งอยู่ที่ตำแหน่งความจำข้อมูลภายใน ตั้งแต่ 00H ถึง 1FH จำนวน 32 ไบต์ โดยจะแบ่งออกเป็นชุด ชุดละ 8 ไบต์จำนวน 4 ชุด ซึ่งแต่ละชุดจะมีชื่อเรียกเป็น R0 ถึง R7 จะเป็น Register ที่ใช้งาน โดยเมื่อ 8051 ถูกรีเซต Register Bank 0 จะถูกเลือกใช้
- ส่วนที่ 2 เรียกว่า Bit Addressable Area ซึ่งมีขนาด 16 ไบต์ที่ตำแหน่งหน่วยความจำข้อมูล 20H ถึง 2FH ในส่วนนี้สามารถที่จะอ้างอิงข้อมูลได้เป็นระดับบิตถึง 128 บิต โดยการอ้างอิงตำแหน่งโดยตรงในลักษณะบิต ตั้งแต่ตำแหน่ง 00H ถึง 7FH



รูปที่ 4.4 แสดงหน่วยความจำข้อมูลภายใน

- ส่วนที่ 3 เรียกว่า Scratch Pad Area จะอยู่ที่ตำแหน่งตั้งแต่ 30H ถึง 7FH ซึ่งเป็นบริเวณหน่วยความจำข้อมูลภายในเอนกประสงค์ที่ผู้ใช้สามารถใช้ได้โดยตรงนอกจากนี้ยังสามารถใช้หน่วยความจำข้อมูลบริเวณนี้สำหรับการเก็บข้อมูลแบบ Stack ได้ด้วย

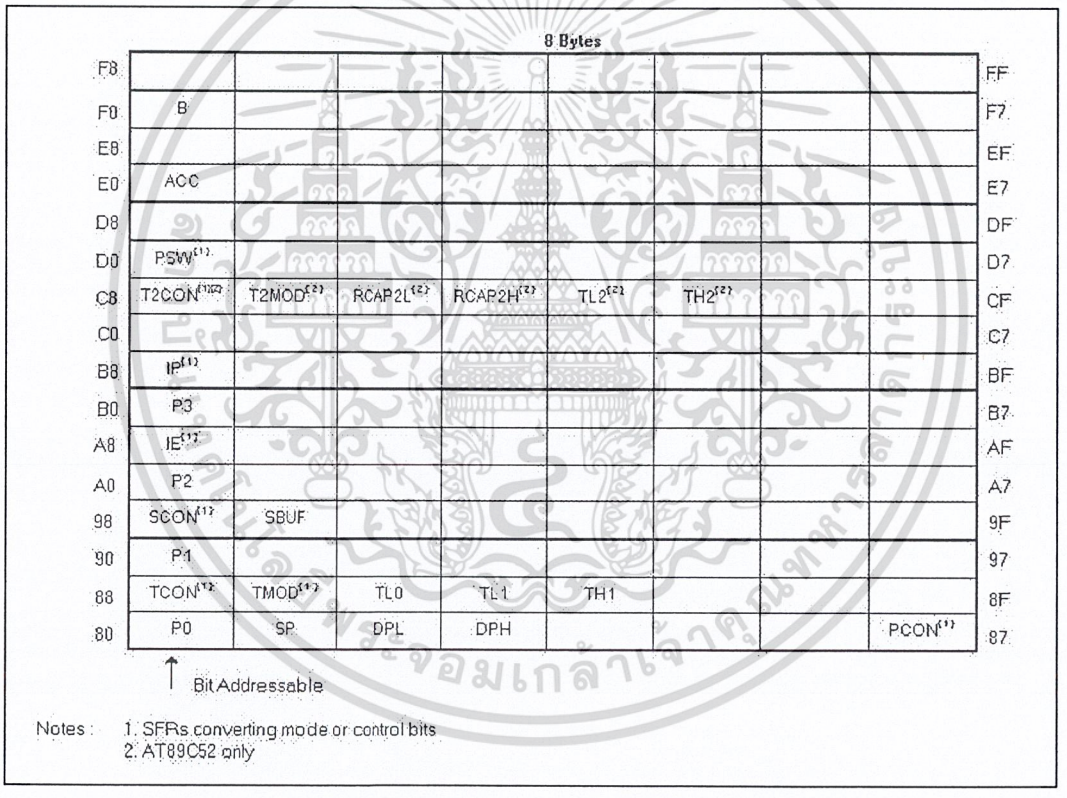
ในส่วนของหน่วยความจำข้อมูลภายในที่ใช้อ้างอิงแบบ Direct เพียงอย่างเดียวหรือที่เรียกว่า SFR ซึ่งเป็นส่วนสำหรับเก็บหรือกำหนดการทำงานภายในของ 8051 ดังแสดงในรูปที่ 4.5

ในส่วนของบริเวณนี้จะมีขนาด 128 ไบต์แต่ในการใช้งานนั้นใช้ได้เฉพาะตำแหน่งซึ่งแสดงไว้ในรูปที่ 4.5เท่านั้น หากผู้ใช้อ้างตำแหน่งนอกเหนือจากนี้นั้นจะได้ข้อมูลที่คาดเดาไม่ได้ โดยแต่ละตำแหน่งจะมีหน้าที่ดังนี้

ACC : เป็น Accumulator ซึ่งเป็นรีจีสเตอร์สำหรับการประมวลผลทางคณิตศาสตร์และลอจิก โดยผู้ใช้สามารถอ้างอิงได้ในรูปแบบของไบต์หรือระดับบิตได้

B : เป็นรีจีสเตอร์พิเศษสำหรับใช้กับคำสั่งในการคูณหรือหาร นอกจากนี้ยังใช้เป็นรีจีสเตอร์สำหรับเก็บพักข้อมูลได้

PSW : เป็นรีจีสเตอร์ Program Status Word หรือแฟลกจะแสดงสถานะการทำงานของ 8051 สำหรับการตรวจสอบซึ่งจะอธิบายรายละเอียดในภายหลัง



รูปที่ 4.5 แสดงรายละเอียดของ Special Function Register

SP : เป็นรีจีสเตอร์สำหรับชี้หน่วยความจำข้อมูลภายในสำหรับการเก็บแบบ Stack

DPTR : เป็นรีจีสเตอร์ขนาด 16 บิต โดยแบ่งเป็น 8 บิตบนและ 8 บิตล่าง ให้สำหรับชี้ตำแหน่งของหน่วยความจำข้อมูลภายนอกหรือสำหรับการอ่านตารางข้อมูลของหน่วยความจำโปรแกรม

P0 : เป็นรีจีสเตอร์สำหรับพอร์ต 0 ของ 8051

P1 : เป็นรีจีสเตอร์สำหรับพอร์ต 1 ของ 8051

P2 : เป็นรีจีสเตอร์สำหรับพอร์ต 2 ของ 8051

P3 : เป็นรีจีสเตอร์สำหรับพอร์ต 3 ของ 8051

IP : เป็นรีจีสเตอร์สำหรับกำหนดลำดับความสำคัญของการอินเตอร์รัพท์ของ 8051

IE : เป็นรีจีสเตอร์สำหรับกำหนดการรับหรือไม่รับการอินเตอร์รัพท์ของ 8051

TMOD : เป็นรีจีสเตอร์สำหรับควบคุมหน้าที่ของ Timer/Counter ของ 8051

TCON : เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ Timer/Counter ของ 8051

T2CON : เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ Timer/Counter 2 ของ 8052

TH0 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 0 8บิตบน

TL0 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 0 8บิตล่าง

TH1 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 1 8บิตบน

TL1 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 1 8บิตล่าง

TH2 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 2 8บิตบนของ 8052

TL2 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 2 8บิตล่างของ 8052

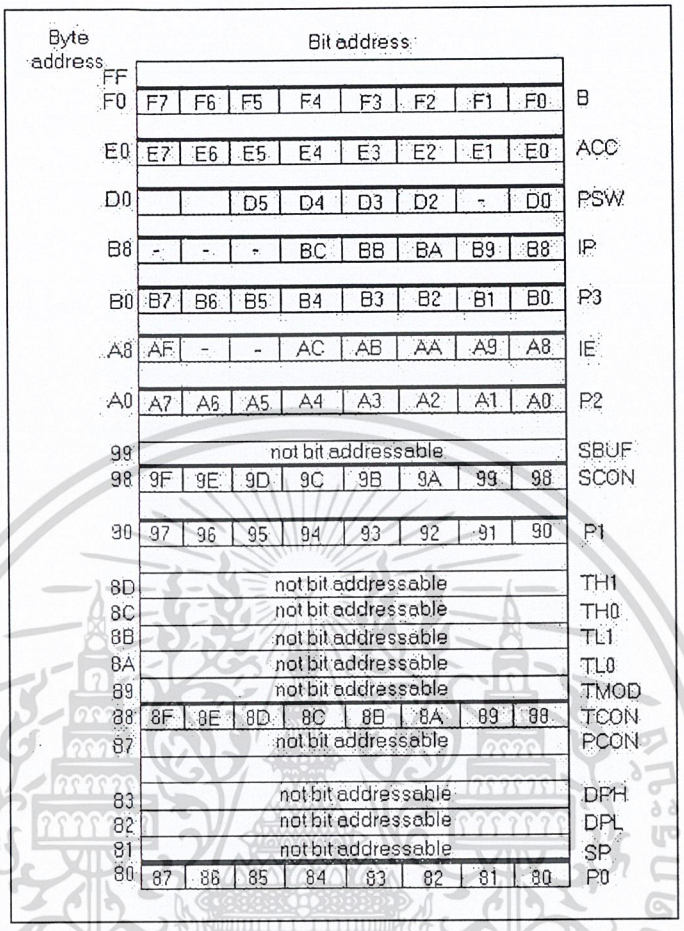
RCAP2H : เป็น Capture Register ของ Timer/Counter 2 8บิตบนของ 8052

SCON : เป็นรีจีสเตอร์สำหรับควบคุมการรับส่งข้อมูลแบบอนุกรมของ 8051

SBUF : เป็นรีจีสเตอร์สำหรับเก็บพักข้อมูลที่ได้จากการรับส่งข้อมูลแบบอนุกรมของ 8051

PCON : เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ 8051 ด้านเกี่ยวกับการใช้กำลังไฟฟ้า

ในส่วนของรีจีสเตอร์ SFR นี้สามารถที่จะอ้างอิงในระดับบิตได้โดยตำแหน่งการอ้างอิงระดับบิตแสดงไว้ในตารางต่อไปนี้



รูปที่ 4.6 แสดงตำแหน่งการอ้างอิงระดับบิตของรีจีสเตอร์ SFR

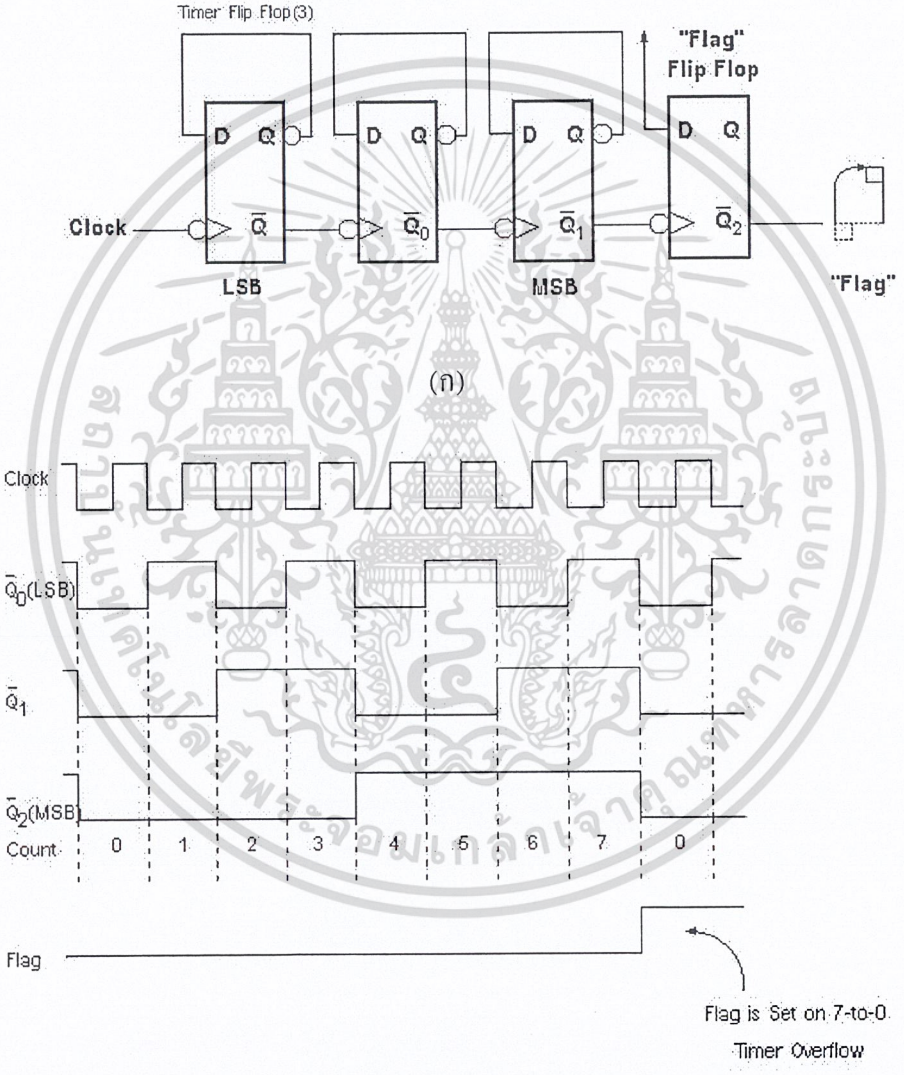
4.3 TIMER

ตัว Timer อาจพิจารณาได้ง่าย ๆ ว่าเป็นตัวฟลิปฟลอปมาต่อเรียงกันโดยมี Clock เป็นอินพุต สำหรับเอาต์พุตที่ออกมาจากฟลิปฟลอปแต่ละตัวจะถูกหารด้วย 2 พิจารณาการต่อฟลิปฟลอปตามรูปที่ 4.7 ถ้าใส่ Clock เข้าไปในฟลิปฟลอปตัวแรก ความถี่ของ Clock ที่ออกจากเอาต์พุตตัวแรกจะถูกหารด้วย 2 และเอาต์พุตนี้จะต่อกับฟลิปฟลอปตัวที่สอง และสัญญาณที่ออกมาจะถูกหารด้วย 2 อีก ดังนั้น ถ้ามีฟลิปฟลอปต่ออยู่ n Stages จะหารสัญญาณนาฬิกาได้ 2^n ถ้าให้เอาต์พุต Stage สุดท้ายของ Timer เป็น Overflow Flip-Flop หรือ Flag และจะให้เอาต์พุตออกมาเมื่อการนับเป็น Overflow เช่น ถ้าเป็นตัวนับแบบ 16 บิต (มีฟลิปฟลอปต่ออยู่ 16 ตัว) วงจรจะนับตั้งแต่ 0000H ถึง FFFFH เมื่อ ฟลิปฟลอปเปลี่ยนจาก FFFFH เป็น 0000H จะให้บิต Overflow ออกมา

พิจารณารูป 4.7(ก) เป็น 3-bit Timer โดยฟลิปฟลอปแต่ละตัวจะนำขา Q มาต่อกับ D ซึ่งอาจเรียกว่าเป็นการใช้ฟลิปฟลอปแบบ Divide-by-two Mode โดยความถี่ของสัญญาณที่ได้จาก

ฟลิปฟลอปแต่ละตัวจะมีค่าหารสองจากสัญญาณนาฬิกาที่เข้ามา เมื่อนับไปถึงค่า 111 (หรือ $Q_2 = 1$, $Q_1 = 1$, $Q_0 = 1$) และเปลี่ยนกลับมาเป็น 000 จะให้บิต Flag ออกมา ดังแสดงในรูปที่ 4.7(ข)

ใน MCS – 51 จะมีตัวจับเวลาอยู่ภายในชิพ ถ้าเป็นเบอร์ 8051 หรือ 8031 จะมี 2 ตัว คือ Timer 0 และ Timer 1 แต่ถ้าเป็นเบอร์ 8052 จะมีเพิ่มอีกหนึ่งตัวคือ Timer 2 รีจีสเตอร์ต่าง ๆ ที่เกี่ยวข้องกับการใช้ Timer แสดงได้ดังตารางที่ 4.2 ซึ่งจะเห็นว่ามีรีจีสเตอร์บางตัวสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย นอกจากนี้ตัว Timer สามารถใช้เป็นตัวนับ (Counter) ได้อีกด้วย โดยการโปรแกรมในรีจีสเตอร์ TMOD



รูปที่ 4.7 รีจีสเตอร์ที่ใช้เป็น Timer

รีจีสเตอร์	หน้าที่	ตำแหน่ง	สามารถอ้างอิงตำแหน่งบิต
TCON	Control	88H	Yes
TMOD	Mode	89H	No
TL_0	Timer 0 Low-byte	8AH	No
TL_1	Timer 1 Low-byte	8BH	No
TH_0	Timer 0 High-byte	8CH	No
TH_1	Timer 1 High-byte	8DH	No
T_2CON*	Timer 2 Control	C8H	Yes
$RCAP_2L$*	Timer 2 Low-byte Capture	CAH	No
$RCAP_2H$*	Timer 2 High-byte Capture	CBH	No
TL_2*	Timer 2 Low-byte	CCH	No
TH_2*	Timer 2 High-byte	CDH	No

* มีในเบอร์ 8032 / 8052

ตารางที่ 4.2 รีจีสเตอร์ที่ใช้เป็น Timer

4.3.1 Timer Mode Register (TMOD)

ตัวรีจีสเตอร์ TMOD เป็นรีจีสเตอร์ควบคุม Timer จะแบ่งออกเป็น 2 กลุ่ม กลุ่มละ 4 บิต โดย 4 บิตบนจะเป็นการควบคุม Timer 1 ส่วน 4 บิตล่างจะเป็นการควบคุม Timer 0 ความหมายของแต่ละบิตดูในตารางที่ 4.3 ซึ่งตัวรีจีสเตอร์นี้เป็นตัวเลือกการทำงานว่าจะให้ตัว Time/Counter ทำงานในโหมดใด และเป็น Timer หรือ Counter รีจีสเตอร์ TCON ไม่สามารถจะโปรแกรมเข้าไปในระดับบิตได้ (Not Bit-Addressable) ซึ่งการใช้งานมักจะโปรแกรมเข้าไปครั้งเดียวในตำแหน่งเริ่มต้นของโปรแกรม

บิต	ชื่อ	Timer	ความหมาย
7	GATE	1	Gate bit ถ้าบิตนี้เซตวงจรจะทำงาน เมื่อ INT1 เป็น High
A	C/T	1	เป็นบิตเลือก Counter / Timer 1 = ใช้เป็น Counter 0 = ใช้เป็น Timer
5	M_1	1	Mode bit 1 (ดูตาราง 5-3)
4	M_0	1	Mode bit 0 (ดูตาราง 5-3)
3	GATE	0	บิต Gate ของ Timer 0
2	C/T	0	บิตเลือก Counter / Timer ของ Timer 0
1	M_1	0	Timer 0 M_1 bit
0	M_0	0	Timer 0 M_0 bit

ตารางที่ 4.3 รีจีสเตอร์ TMOD (Timer Mode)

M_1	M_0	Mode	ความหมาย
0	0	0	ใช้เป็น Timer แบบ 13-bit (8048 Mode)
0	1	1	ใช้เป็น Timer แบบ 16-bit
1	0	2	ใช้เป็น Timer แบบ 8-bit Auto-reload Mode
1	1	3	Split Timer Mode : แยก Timer 0 ออกเป็น Timer 8 บิตสองตัวคือ TL_0 และ TH_0 โดยไม่ใช้ Timer 1

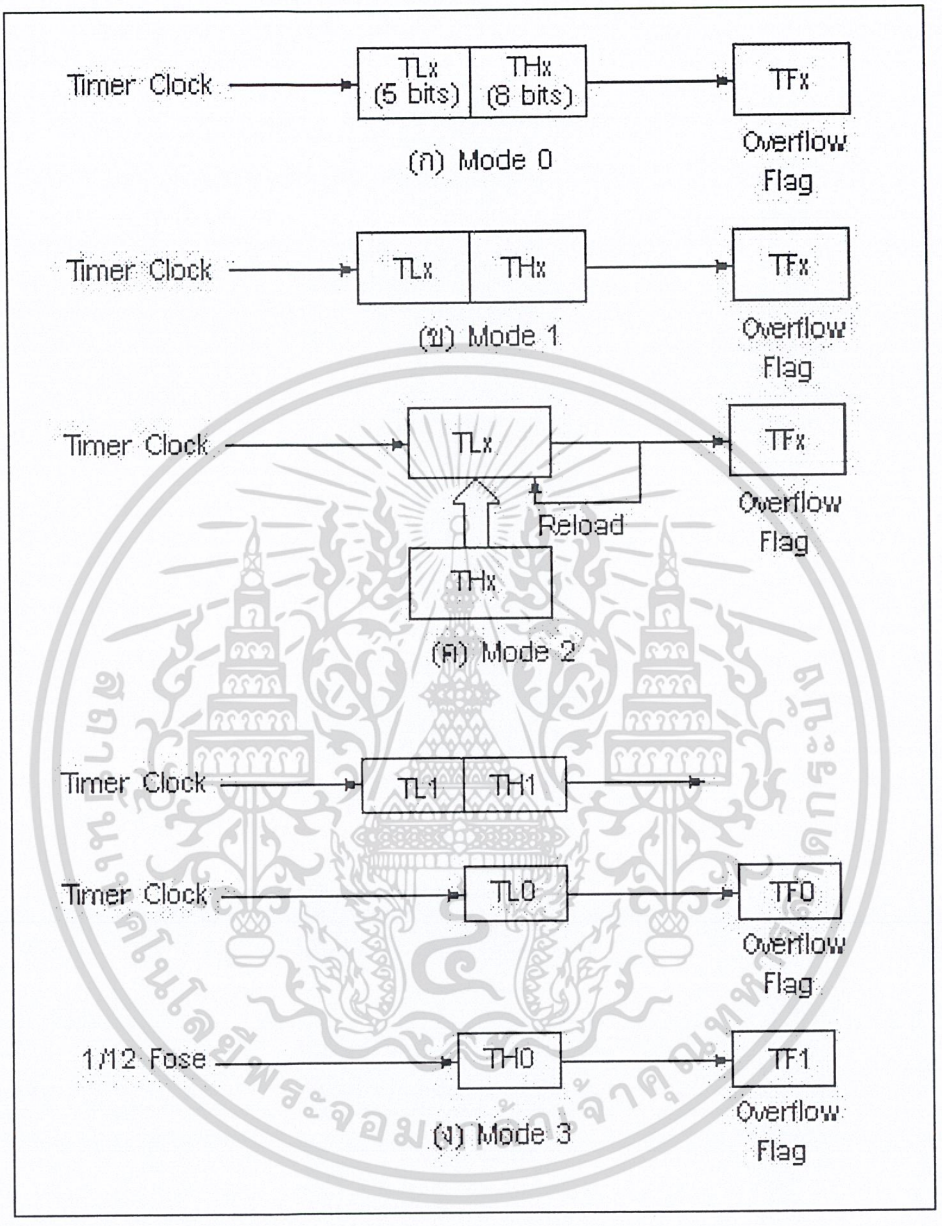
ตารางที่ 4.4 การใช้ Timer โหมดต่าง ๆ

4.3.2 Timer Control Register (TCON)

รีจีสเตอร์ TCON เป็นรีจีสเตอร์ที่บอกสถานะและควบคุมบิต Timer 0 และ Timer 1 ซึ่งดูได้จากตารางที่ 4.5 รีจีสเตอร์นี้สามารถเข้าถึงข้อมูลระดับบิตได้

บิต	ชื่อ	ตำแหน่งบิต	ความหมาย
TCON.7	TF1	8FH	บิตแฟลกแสดงการโอเวอร์โฟลว์ของ Timer 1 จะ Set โดย Hardware และ Clear โดย Software
TCON.6	TR1	8EH	บิตควบคุมการปิด-เปิด Timer 1 Set และ Clear โดย Software
TCON.5	TF0	8DH	แฟลกแสดงการโอเวอร์โฟลว์ของ Timer 0
TCON.4	TR0	8CH	บิตควบคุมการปิด-เปิด Timer 0
TCON.3	IE1	8BH	บิตแฟลกแสดงการอินเทอร์รัพท์จาก INT1 จะ Set โดย Hardware และสามารถ Clear ได้ด้วย Software
TCON.2	IT1	8AH	บิตเลือกชนิดของสัญญาณอินเทอร์รัพท์จากอินเทอร์รัพท์ภายนอก INT1 สามารถ Set และ Clear ได้ด้วย Software
TCON.1	IE0	89H	บิตแฟลกแสดงการอินเทอร์รัพท์จาก INT0
TCON.0	IT0	88H	บิตเลือกชนิดของสัญญาณอินเทอร์รัพท์จากอินเทอร์รัพท์ภายนอก INT0

ตารางที่ 4.5 แสดงความหมายแต่ละบิตของรีจีสเตอร์ TCON (Timer Control)



รูปที่ 4.8 การทำงานของ Timer ในโหมดต่าง

4.3.3 Timer Mode And Overflow Flag

เมื่อใช้ Timer 0 และ Timer 1 จะต้องใช้รีจีสเตอร์คู่ TLx และ THx โดยค่า x จะเป็นตัวบอกว่าเป็น Timer 0 หรือ Timer 1 การใช้ Timer สามารถใช้งานได้หลายโหมด ดังแสดงในรูปที่ 4.2 ซึ่งเราสามารถเซตค่าโหมดการทำงานได้ โดยการโปรแกรมในรีจีสเตอร์ TMOD

13-Bit Timer Mode (Mode 0)

การทำงานในโหมด 0 นี้จะเป็นการใช้ Timer แบบ 13 บิต ดังแสดงในรูป 4.8(ก) ซึ่งจะใช้ 5 บิตล่างของ TLx โดยไม่สนใจ 3 บิตที่เหลือ และ 8 บิต ของ THx การทำงานในโหมดนี้ เมื่อบิตของ TLx นับไปจนเป็น "1" ทุกบิตจะส่ง Clock 1 ถูกให้ THx นับต่อและเมื่อนับเป็น "1" ทุกบิต และเปลี่ยนกลับเป็น "0" จะเกิด Overflow Flag เกิดขึ้น

16-Bit Timer Mode (Mode 1)

การทำงานในโหมดนี้จะเหมือนกับการทำงานในโหมด 0 แต่เป็น Timer แบบ 16 บิต ซึ่งการนับจะเริ่มตั้งแต่ 0000H,0001H,0002H ไปเรื่อย ๆ และจะเกิด Overlflow ขึ้น เมื่อมีการเปลี่ยนจาก FFFFH เป็น 0000H ดังรูปที่ 4.8(ข) ซึ่งเป็นการเซต Overflow Flag และค่านี้จะเกิดขึ้นในบิต TFx ของรีจีส-เตอร์ TCON ซึ่งสามารถอ่านและเขียนด้วยโปรแกรม

การใช้ตัว Timer นี้ค่าของบิตสูงสุด (MSB) คือค่าบิต 7 ของ THx ส่วนบิตต่ำสุด (LSB) คือ บิต 0 ของ TLx บิต LSB จะเป็น Toggles เมื่อมีสัญญาณอินพุตเข้ามา ถูกหารด้วย 2 ดังนั้นจะพบว่าบิต MSB จะ Toggles ด้วยค่าความถี่ของสัญญาณอินพุตหารด้วย 65,536 (2^{16}) และค่า Timer รีจีสเตอร์นี้ (TLx/THx) สามารถอ่านและเขียนได้ด้วยการโปรแกรม ดังนั้นสามารถนำไปประยุกต์ใช้งานได้ตามต้องการ

8-Bit Auto – Reload Mode (Mode 2)

การทำงานในโหมด 2 เรียกอีกอย่างหนึ่งว่า 8-bit Auto – reload Mode โดยใช้ Timer ไบต์ต่ำ (TLx) เป็น Timer แบบ 8 บิต เมื่อไบต์ต่ำเกิด Overflows หรือเกิดการเปลี่ยนแปลงจาก FFH เป็น 00H จะมีการโหลดค่าที่เก็บไว้ในไบต์สูง (THx) ไปเก็บไว้ในไบต์ต่ำ (TLx) ซึ่งจะเป็นค่าเริ่มต้นของการนับครั้งต่อไป นิยมใช้สร้างเป็นฐานเวลาที่สามารถโปรแกรมได้ การทำงานในโหมดนี้แสดงดังรูปที่ 4.8(ค)

Split Timer Mode (Mode 3)

การทำงานในโหมด 3 นี้ ตัว Timer 1 จะไม่ทำงาน ตัว Timer 0 จะแยกเป็น 2 ตัว ตัวละ 8 บิต คือ TL0 และ TH0 เมื่อ Timer เกิด Overflows จะมีการเซตบิต TF0 และ TF1 ดังแสดงในรูปที่ 4.8(ง)

การทำงานในโหมด 3 นี้ Timer 1 จะไม่ถูกใช้งานแต่เราสามารถสวิตช์ให้ Timer 1 ไปทำงานในโหมดอื่นได้ แต่การทำงานของ Timer 1 จะไม่มีการอินเทอร์รัพท์เกิดขึ้น เพราะบิต TF1 ถูกใช้ในการนับของ TH0 ในการทำงานของโหมด 3 ไปแล้ว เราอาจมองว่าถ้าให้ Timer ทำงานใน

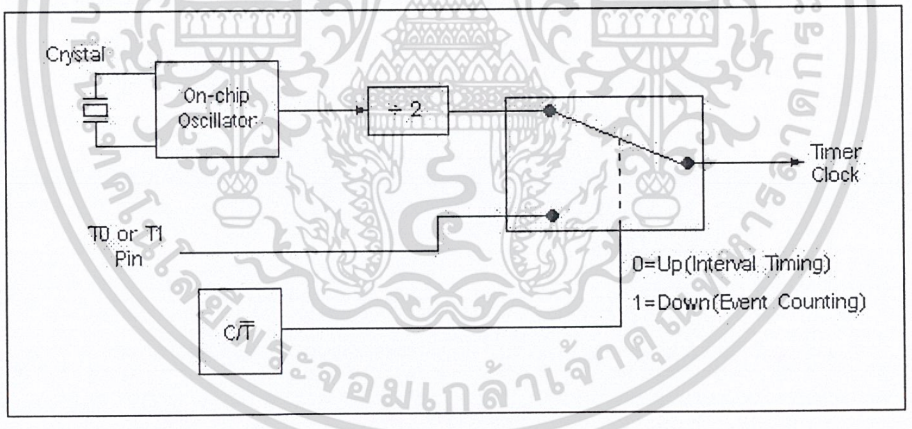
โหมด 3 ทำให้เรามี Timer เพิ่มขึ้น คือ TH_0 และ TL_0 ใน Timer 0 โหมด 3 และโปรแกรมให้ Timer 1 ไปทำงานในโหมดอื่น ๆ

4.3.4 Clocking Source

ในรูปที่ 4.8 ไม่ได้แสดงว่า Timer Clock นำมาจากที่ใดซึ่งการใช้ Timer นี้สามารถใช้ได้ 2 หน้าที่ คือเป็นตัวจับเวลา (Timer) และเป็นตัวนับ (Counter) ซึ่งสามารถโปรแกรมได้โดยการเซต หรือรีเซตบิต C / Tในรีจีสเตอร์ TMOD

การใช้เป็นตัวจับเวลา (Timer)

ถ้าบิต C / Tใน TMOD เป็นลอจิก "0" จะเป็นการเลือกให้ Timer นำ Clock มาจากวงจร Oscillator ในชิพ ซึ่งสัญญาณนาฬิกาจะเข้ามาทุก ๆ Machine Cycle หรืออาจกล่าวได้ว่าค่าใน THx และ TLx จะมีค่าเพิ่มขึ้นด้วยอัตราการนับแต่ละครั้งใช้เวลาเท่ากับ 1/12 ของความถี่ของสัญญาณนาฬิกาที่ใช้บนชิพ ดังแสดงในรูปที่ 4.9 ถ้า MCS – 51 ใช้สัญญาณนาฬิกา 12 MHz การนับจะมีความถี่เท่ากับ 1 MHz



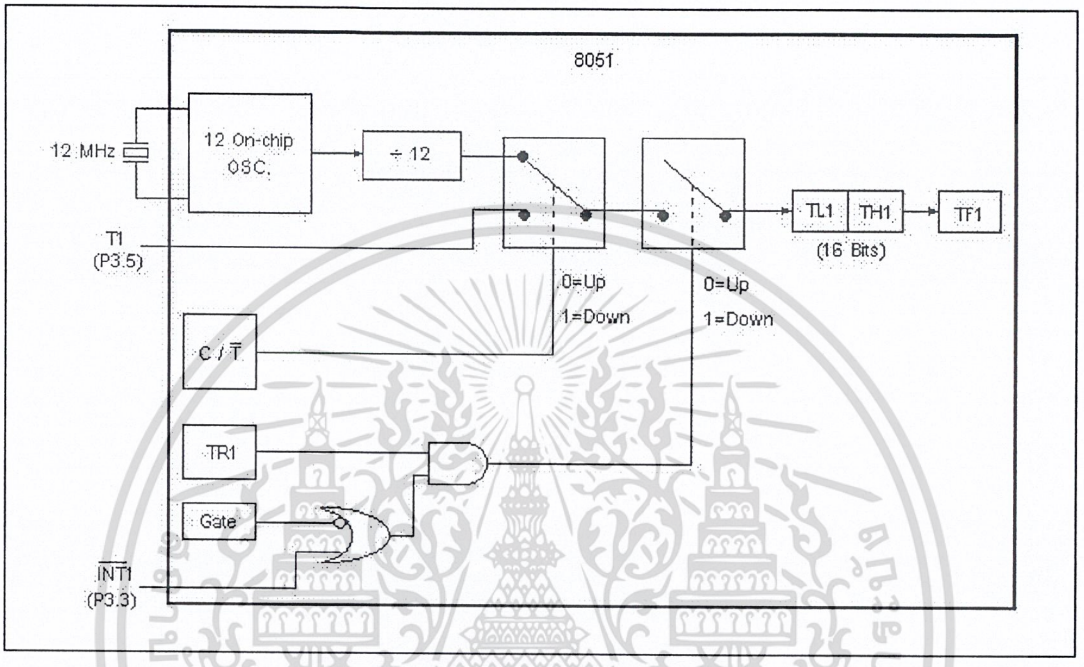
รูปที่ 4.9 ความถี่ของสัญญาณนาฬิกาที่เข้าขา Timer

4.3.5 การใช้เป็นตัวนับ (Counter)

ถ้าบิต C / T เป็น "1" ตัว Timer จะนำ Clock มาจากภายนอกโดยใช้ขา $P_{3.4}$ หรือ T_0 เป็นขา Input Clock ให้กับ Timer 0 และใช้ขา $P_{3.5}$ หรือ T_1 เป็น Input Clock ให้กับ Timer 1 ดังรูปที่ 4.9 หรืออาจมองว่า ถ้าจะให้นับอะไรสัญญาณที่จะนับให้ต่อกับขา T_0 และ T_1 ในการใช้เป็น Counter สัญญาณที่เข้ามามีการเปลี่ยนแปลงจาก "1" เป็น "0" จะทำให้วงจรนับ TLx มีค่าเพิ่มขึ้น 1 ภายใน MCS – 51 นี้จะตรวจสอบขาอินพุต T_0 และ T_1 ในช่วงเวลาเฟส 2 ของ State 5 (S_5P_2) ถ้าพบว่ามีค่า

สัญญาณนาฬิกาความถี่ 1 MHz เมื่อ INT0 ลงเป็น “0” ตัว Timer “ Gate Off “ สัญญาณที่ได้จะมีความกว้างของสัญญาณนาฬิกา 1 μS ส่งเข้าไปใน TL0/TH0

รูปที่ 4.11 จะเป็นระบบที่สมบูรณ์ของ Timer 1 เมื่อทำงานในโหมด 1 ซึ่งเป็น 16-bit Timer โดยใช้รีจีสเตอร์ TL1 / TH1 และ Overflow Flag TF1 ในรูปจะเห็นถึงการควบคุมแหล่งกำเนิด Clock การเริ่มทำงาน และการหยุดทำงาน



รูปที่ 4.11 ระบบทั้งหมดของ Timer 1

4.3.6 Initializing And Accessing Timer Register

การใช้งาน Timer เริ่มแรกจะต้องโปรแกรมเพื่อเลือกโหมดการทำงานของ Timer ก่อนเมื่อเริ่มใช้งานก็โปรแกรมให้ เริ่มทำงาน, หยุดทำงาน, อ่าน และ เคลียร์ค่า Flag Bits อ่านค่า Timer Registers ตามลำดับ เพื่อนำไปประยุกต์การใช้งานต่อไป

TMOD คือ รีจีสเตอร์ที่ต้องโปรแกรม โดยเซตโหมดการทำงานก่อน ตัวอย่างเช่น ถ้าให้ Timer 1 เป็น 16-bits Timer (โหมด 1) นับสัญญาณนาฬิกาบนชิพ สามารถเขียนคำสั่งได้ดังนี้

```
MOV TMOD, #00010000B
```

ผลที่ได้จากคำสั่งข้างบนคือ เซตบิต M1 = 0 และ M0 = 1 ซึ่งเป็นการเลือกโหมด 1 และให้ C / T= 0 และ GATE = 0 ซึ่งเป็นการใช้สัญญาณนาฬิกาจากภายในหรือใช้เป็น Timer และตัว Timer นี้จะยังไม่ทำงาน ถ้าบิตควบคุม TR1 ยังไม่ได้เซต

ถ้าให้ Timer นี้นับขึ้นโดยใช้รีจีสเตอร์ TL1 / TH1 และจะเซตบิต Overflow Flag เมื่อรีจีสเตอร์เปลี่ยนจาก FFFFH เป็น 0000H โดยให้นับเวลาไป 100 μS หรือให้ TL1 / TH1 นับสัญญาณนาฬิกาได้ 100 ลูก ดังนั้นค่าเริ่มต้นของ TL1 / TH1 จะไม่เริ่มที่ 0000H จะต้องเริ่มที่ FFFFH ลบด้วย

100 ลูก หรือ FF9CH เพื่อให้นับไปถึง FFFFH และเปลี่ยนเป็น 0000H ได้สัญญาณนาฬิกา 100 ลูก พอดี สามารถเขียนคำสั่งได้ดังนี้

```
MOV TL1 , # 9CH
MOV TH1 , # 0FFH
```

ถ้าให้ Timer เริ่มทำงานก็ให้บิตควบคุมดังนี้

```
SETB TR1
```

จากนั้นบิต Overflow Flag จะส่งออกมาหลังเวลาผ่านไป 100 μS ซึ่งเราสามารถเขียนโปรแกรมเป็นโปรแกรมวนลูป 100 μS ได้ โดยตรวจสอบบิต TF1 ว่าถูกเซตหรือไม่ ถ้าไม่เซตก็ให้วนลูปต่อไปดังนี้

```
CLR TR1
CLR TF1
```

การใช้แบบ Readind a Timer "On the Fly"

การใช้งานแบบประยุกต์บางงานจะต้องอ่านค่าจาก Timer Register เนื่องจากตัว Timer Register มีขนาด 2 ไบต์ ถ้าหากไบต์ต่ำเกิด Overflow จะทดเข้าไบต์สูง ถ้าหากเขียนโปรแกรมให้อ่านค่าจากไบต์ต่ำก่อน แล้วจึงอ่านไบต์สูงข้อมูลที่ได้อาจเกิดข้อผิดพลาดได้เนื่องจากไบต์ต่ำมีการเปลี่ยนแปลงเร็วกว่าไบต์สูง การอ่านข้อมูลควรอ่านจากไบต์สูงก่อน แล้วจึงกลับมาอ่านไบต์ต่ำ จากนั้นอ่าน ข้อมูลไบต์สูงอีกครั้ง ถ้าค่าไบต์สูงที่อ่านได้ไม่มีการเปลี่ยนแปลงให้ใช้ค่านั้นได้เลย แต่ถ้ามีการเปลี่ยน-แปลงให้อ่านอีกครั้ง ถ้าต้องการอ่านข้อมูลจาก TL1 / TH1 เข้าในรีจีสเตอร์ R6 / R7 อาจเขียนโปรแกรมได้ดังนี้

```
AGAIN :  MOV A, TH1
         MOV R6, TL1
         CJNE A, TH1, AGAIN
         MOV R7, A
```

4.3.7 Short Intervals And Long Intervals

ถ้า MCS – 51 ทำงานที่ความถี่สัญญาณนาฬิกา 12 MHz ถ้าให้ Timers ใช้วงจร Oscillator บนชิพ สัญญาณนาฬิกาจะถูกหารด้วย 12 และ Timer จะทำงานด้วยความถี่ 1 MHz ถ้าต้องการใช้โปรแกรมสร้างสัญญาณนาฬิกาออกมาอาจทำได้โดยง่าย ซึ่งพิจารณาจากการทำงานชุดคำสั่งต่าง ๆ ของ MCS – 51 ใน 1 Machine Cycle จะใช้เวลา 1μS ในตารางที่ 4.6 จะแสดงความกว้างของสัญญาณที่สร้างขึ้นจาก MCS – 51 ที่ทำงานด้วย Crystal ความถี่ 12 MHz

Maximum Interval in Microseconps	Technique
≈10	Software Tuing
256	8 – bit Timer with Auto-reload
65536	8 – bit Timer
No Limit	16 – bit Timer Plus Software Loops

ตารางที่ 4.6 ค่าสูงสุดของการใช้ Timer โหมดต่าง ๆ

4.4 การอินเตอร์รัพท์

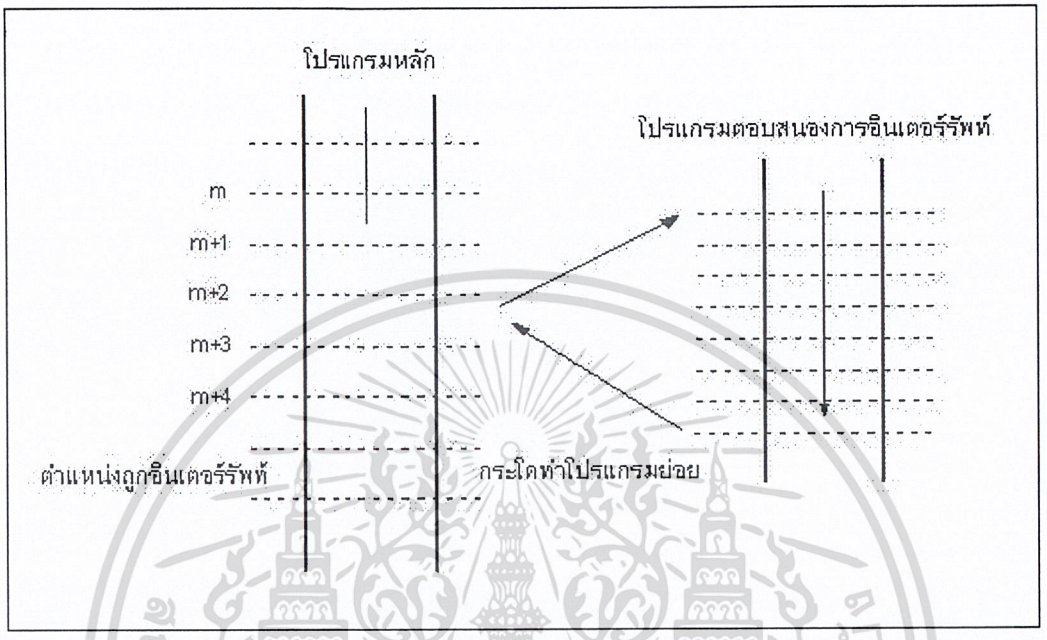
การทำงานของระบบคอมพิวเตอร์ โดยทั่วไปมักมีอุปกรณ์ภายนอกต่อร่วมอยู่ ถ้าคอมพิวเตอร์ต้องการทำงานกับอุปกรณ์ภายนอกจะต้องคอยตรวจสอบอุปกรณ์เหล่านั้นเสมอ ตัวอย่างเช่น ถ้าหากให้คอมพิวเตอร์พอร์ทหนึ่งต่ออยู่กับหลอด LED 7 ส่วน อีกพอร์ทหนึ่งต่อกับสวิตช์ ถ้าระบบของเราทำงานเป็นนาฬิกาเดินไปให้คอยตรวจสอบสวิตช์ด้วยว่ามีการกดหรือยัง การทำงานแบบนี้เรียกว่า Polling Method คือตัวไมโครโปรเซสเซอร์จะต้องคอยตรวจสอบอุปกรณ์อินพุตตลอดเวลาว่ามีข้อมูลเข้ามาหรือยัง การทำงานแบบนี้ ถ้ามีอุปกรณ์ภายนอกหลายตัวระบบต้องตรวจสอบอุปกรณ์ภายนอกหลายตัว ทำให้เสียเวลาในการทำงานหลักไป การทำงานอีกแบบหนึ่งจะให้ CPU ทำงานหลัก ถ้ามีการกดสวิตช์เมื่อไรให้นาฬิกาหยุดเดินทันที การทำงานในลักษณะนี้ CPU ไม่ต้องเสียเวลาในการตรวจอุปกรณ์ภายนอก ถ้าอุปกรณ์ภายนอกต้องการติดต่อกับ CPU อุปกรณ์ภายนอกจะส่งสัญญาณมาบอก CPUเอง ระบบนี้เรียกว่า การอินเทอร์รัพท์ (Interrupt)

4.4.1 ขบวนการเกิดอินเตอร์รัพท์

ถ้าหากคอมพิวเตอร์กำลังทำงานโปรแกรมหลักอยู่เมื่อมีการอินเทอร์รัพท์เข้ามา คอมพิวเตอร์จะละทิ้งโปรแกรมหลัก แต่ไปทำงานโปรแกรมตอบสนองการอินเตอร์รัพท์ (Interrupt Service Routine) เมื่อทำโปรแกรมตอบสนองอินเตอร์รัพท์เสร็จคอมพิวเตอร์จะกลับมาทำโปรแกรมเดิมพิจารณารูปที่ 4.12

ถ้า CPU กำลังทำงานโปรแกรมหลักอยู่ เช่นกำลังทำคำสั่งในตำแหน่งของหน่วยความจำที่ m, m+1, m+2 ไปเรื่อย ๆ โดย PC จะชี้ที่ตำแหน่งที่จะอ่านค่าคำสั่งถัดมา เมื่อโปรแกรมทำงานมาถึงตำแหน่งที่ m+3 แล้วเกิดการอินเตอร์รัพท์ขึ้น (ขณะนั้น PC อยู่ที่ m+4) โปรแกรมจะต้องทำงาน

โปรแกรมตอบสนองการอินเตอร์รัพท์ โดยย้าย PC ไปที่ตำแหน่งที่เก็บโปรแกรมตอบสนองการอินเตอร์รัพท์ จากนั้นจะเก็บค่า PC เดิมลงในหน่วยความจำสแตค เมื่อคอมพิวเตอร์ทำงานโปรแกรมตอบสนองการอินเตอร์รัพท์เสร็จสิ้นลง จะคืนค่าใน สแตค (m+4) ให้กับ PC ทำโปรแกรมหลักต่อไป



รูปที่ 4.12 ขั้นตอนการทำงานของโปรแกรมเมื่อถูกอินเตอร์รัพท์

4.4.2 สัญญาณอินเตอร์รัพท์

แหล่งกำเนิดสัญญาณอินเตอร์รัพท์ที่ใช้กับ MCS – 51 มีสองชนิดคือ อินเตอร์รัพท์ภายในและภายนอก โดยอินเทอร์รัพท์ภายในจะเกิดขึ้นจากภายในตัว MCS – 51 เอง ได้แก่สัญญาณจากไทเมอร์แฟลก 0 (TF0) ไทเมอร์แฟลก 1 (TF1) และพอร์ทอนุกรม สำหรับอินเตอร์รัพท์ภายนอกเกิดจากสัญญาณที่กระตุ้นเข้ามาทางขา INT0 และ INT1 เมื่อมีสัญญาณอินเตอร์รัพท์จากแหล่งต่าง ๆ เข้ามา เราสามารถโปรแกรมได้ว่าจะให้ MCS – 51 ยอมให้มีการอินเตอร์รัพท์ได้หรือไม่ โดยการโปรแกรมไปที่ รีจีสเตอร์ IE (Interrupt Enable) และถ้ามีสัญญาณอินเตอร์รัพท์มากจากแหล่งต่าง ๆ หลายแหล่งพร้อมกันเราสามารถจัดลำดับได้ว่า จะให้อินเตอร์รัพท์ใดเกิดก่อน โดยการโปรแกรมไปที่ อินเตอร์รัพท์ไพออริตี้ IP (Interrupt Priority) รีจีสเตอร์ทั้งสองตัวมีรายละเอียดดังนี้

Interrupt Enables

เป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ใช้สำหรับกำหนดค่าว่าถ้าเกิดการอินเตอร์รัพท์จากแหล่งต่าง ๆ จะทำอินเตอร์รัพท์เหล่านั้นหรือไม่ โดยรายละเอียดของบิตต่าง ๆ มีดังตารางที่ 4.7

บิต 7 บิต 0

EA	X	ET2	ES	ET1	EX1	ET0	EX0

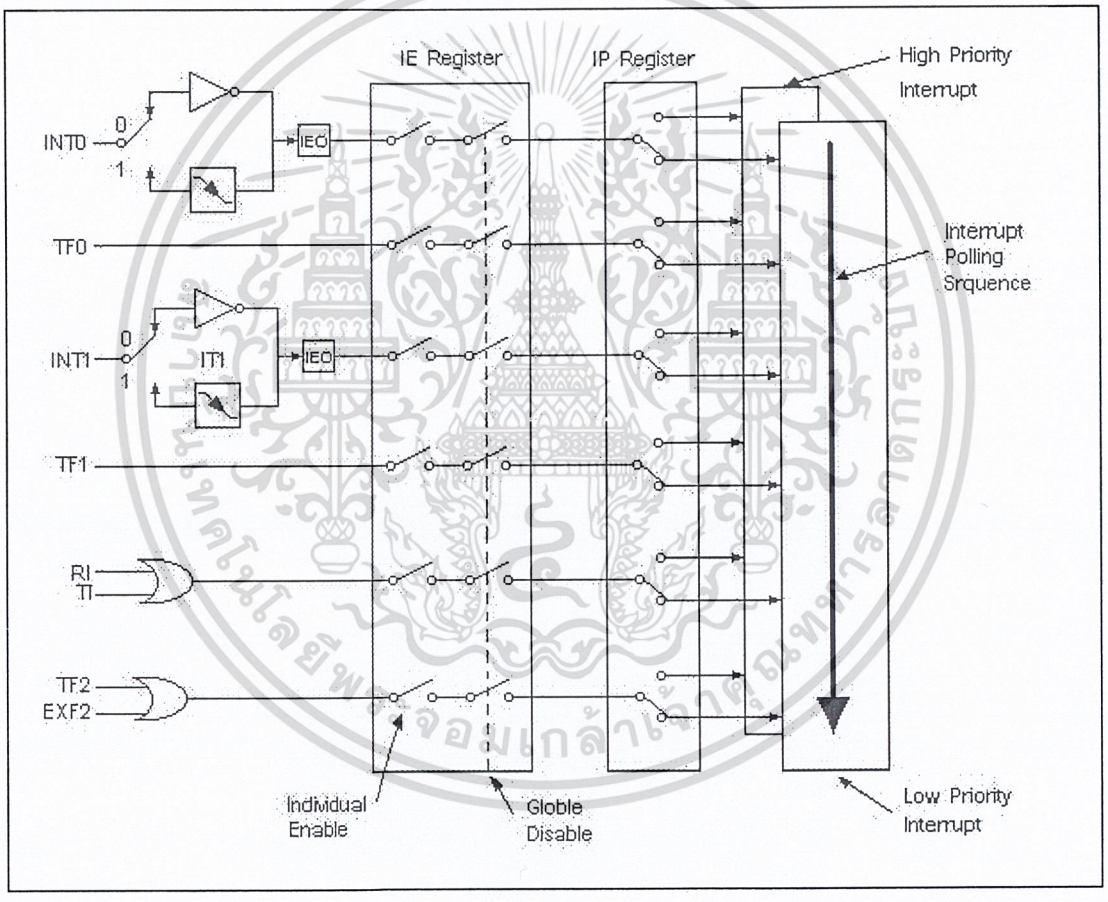
บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IE.7	EA	AFH	ถ้าเซตยอมให้มีการอินเตอร์รัพท์
IE.6	-	AEH	ไม่ใช้งาน
IE.5	ET2	ADH	Enable อินเตอร์รัพท์จาก Timer 2 (ใช้กับ 8052)
IE.4	ES	ACH	Enable อินเตอร์รัพท์จากพอร์ทอนุกรม
IE.3	ET1	ABH	Enable อินเตอร์รัพท์จาก Timer 1
IE.2	EX1	AAH	Enable อินเตอร์รัพท์จาก INT1
IE.1	ET0	A9H	Enable อินเตอร์รัพท์จาก Timer 0
IE.0	EX0	A8H	Enable อินเตอร์รัพท์จาก INT0

ตารางที่ 4.7 บิตต่าง ๆ ของรีจีสเตอร์ IE

Interrupt Priority

เป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ใช้ในการจัดลำดับความสำคัญของการอินเทอร์รัพท์ซึ่งสามารถจัดได้สองลำดับ ถ้าเป็น “1” หมายความว่ามีความสำคัญสูงสุด ถ้าเป็น “0” หมายความว่ามีลำดับความสำคัญต่ำสุด ความหมายของบิตต่าง ๆ แสดงได้ดังตารางที่ 4.8 ถ้าหากกำหนดให้มีความสำคัญเป็น “1” เหมือนกันหมด MCS – 51 จะจัดลำดับความสำคัญใหม่ดังนี้

ลำดับ	อินเตอร์รัพท์
1 (สูงสุด)	IE0
2	TF0
3	IE1
4	TF1
5 (ต่ำสุด)	Serial Port



รูปที่ 4.13 รีจิสเตอร์ต่าง ๆ ที่เกี่ยวข้องกับการอินเตอร์รัพท์

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IP.7	-	-	ไม่ใช้งาน
IP.6	-	-	ไม่ใช้งาน
IP.5	PT2	0BDH	ใช้กับ Timer 2 (8052)
IP.4	PS	0BCH	ใช้กับพอร์ทอนุกรม
IP.3	PT1	0BBH	ใช้กับ Timer 1
IP.2	PX1	0BAH	ใช้กับอินเทอร์รัพท์จาก INT1
IP.1	PT0	0B9H	ใช้กับ Timer 0
IP.0	PX0	0B8H	ใช้กับอินเทอร์รัพท์จาก INT0

ตารางที่ 4.8 บิตและหน้าที่ต่าง ๆ ของรีจีสเตอร์ IP

จากรูปที่ 4.13 แสดงการอินเตอร์รัพท์จากแหล่งต่าง ๆ ที่มีผลกับ MCS – 51 ถ้าเป็นเบอร์ 8051 8031 จะถูกอินเตอร์รัพท์ได้ 5 แหล่ง ถ้าเป็นเบอร์ 8052,8032 จะถูกอินเตอร์รัพท์ได้ 6 แหล่ง โดยเพิ่มอินเตอร์รัพท์จาก Timer 2 ในรูปที่ 4.13 จะแสดงให้เห็นว่า ถ้า MCS – 51 จะถูกอินเตอร์รัพท์ได้จะต้องเซตค่า Global Enable ในรีจีสเตอร์ IE นอกจากนี้ยังกำหนดได้ว่าจะให้อินเตอร์รัพท์ใดเกิดได้ โดยการเซตค่า Interrupt Enable ของอินเตอร์รัพท์จากแหล่งต่าง ๆ ในรีจีสเตอร์ IE จากรูปยังแสดงให้เห็นอีกว่าเมื่อมีการอินเทอร์รัพท์เข้ามาจะมีผลต่อแฟลกใด เช่นถ้า INT0 เป็น “1” บิต IE0 จะเป็น “1” หมายความว่าถูกอินเตอร์รัพท์ โดยแฟลกต่าง ๆ ที่มีผลจากการถูกอินเตอร์รัพท์แสดงได้ดังตารางที่ 4.9

อินเตอร์รัพท์	แฟลก	ประกอบอยู่ในรีจีสเตอร์
External 0	IE0	TCON.1
External 1	IE1	TCON.3
Timer 1	TF1	TCON.7
Timer 0	TF0	TCON.5
Serial port	T1	SCON.1
Serial port	RI	SCON.0
Timer 2	TF2	T2CON.7 (8052)
Timer 2	EXF2	T2CON.6 (8052)

ตารางที่ 4.9 แฟลกที่จะทำงานเมื่อถูกอินเตอร์รัพท์

จากตารางจะเห็นว่า ถ้ามีการอินเตอร์รัพท์จากภายนอกเข้ามา ตัวที่จะอินเตอร์รัพท์ MCS – 51 คือ บิตแฟลก IE0 ซึ่งอยู่ในรีจีสเตอร์ TCON ถ้ามีการสื่อสารแบบอนุกรม เมื่อข้อมูลถูกส่งไปหมด แล้วจะอินเตอร์รัพท์ MCS – 51 ทางบิตแฟลก TI ถ้ารับข้อมูลหมดแล้วจะอินเตอร์รัพท์ MCS – 51 ทางบิตแฟลก RI ซึ่งอยู่ในรีจีสเตอร์ SCON และถ้าใช้ Timer 0 ในการนับเมื่อเกิด Overflow สามารถ อินเตอร์รัพท์ MCS – 51 ได้ทางบิต TF0

4.4.3 การทำงานของระบบหลังถูกอินเตอร์รัพท์

เมื่อ MCS – 51 ถูกอินเทอร์รัพท์จะต้องกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์โดยตำแหน่งที่จะกระโดดไปเรียกว่า อินเตอร์รัพท์เวกเตอร์ (Interrupt Vectors) เมื่อทำโปรแกรมตอบสนองการอินเตอร์รัพท์เรียบร้อยแล้ว MCS – 51 จะกระโดดมาทำงานยังตำแหน่งเดิม โดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์จะต้องเก็บค่าตำแหน่งเดิมไว้ โดยเก็บค่าPCลงหน่วยความ-จำสแตคซึ่งอยู่ที่หน่วยความจำที่ถูกชี้โดยรีจีสเตอร์ SP เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัพท์เสร็จแล้วจะคืนค่าในหน่วยความจำสแตคให้ PC ตามเดิม ค่าอินเตอร์รัพท์เวกเตอร์ของ MCS – 51 แสดงได้ดังตารางที่ 4.10

อินเตอร์รัพท์	อินเตอร์รัพท์เวกเตอร์
System Reset	0000H
External 0	0003H
Timer 0	000BH
External 1	0013H
Timer 1	001BH
Serial Port	0023H
Timer 2	002BH

ตารางที่ 4.10 อินเตอร์รัพท์เวกเตอร์ของอินเตอร์รัพท์ต่าง ๆ

จากตารางจะเห็นว่าถ้าระบบถูกอินเตอร์รัพท์จากภายนอกทาง INT0 ตัว MCS – 51 จะกระโดดไปทำงานที่ตำแหน่ง 0003H ถ้าระบบถูกอินเตอร์รัพท์จาก Timer 0 จะกระโดดไปทำงานตำแหน่ง 000BH

4.4.4 การออกแบบโปรแกรมอินเตอร์รัพท์

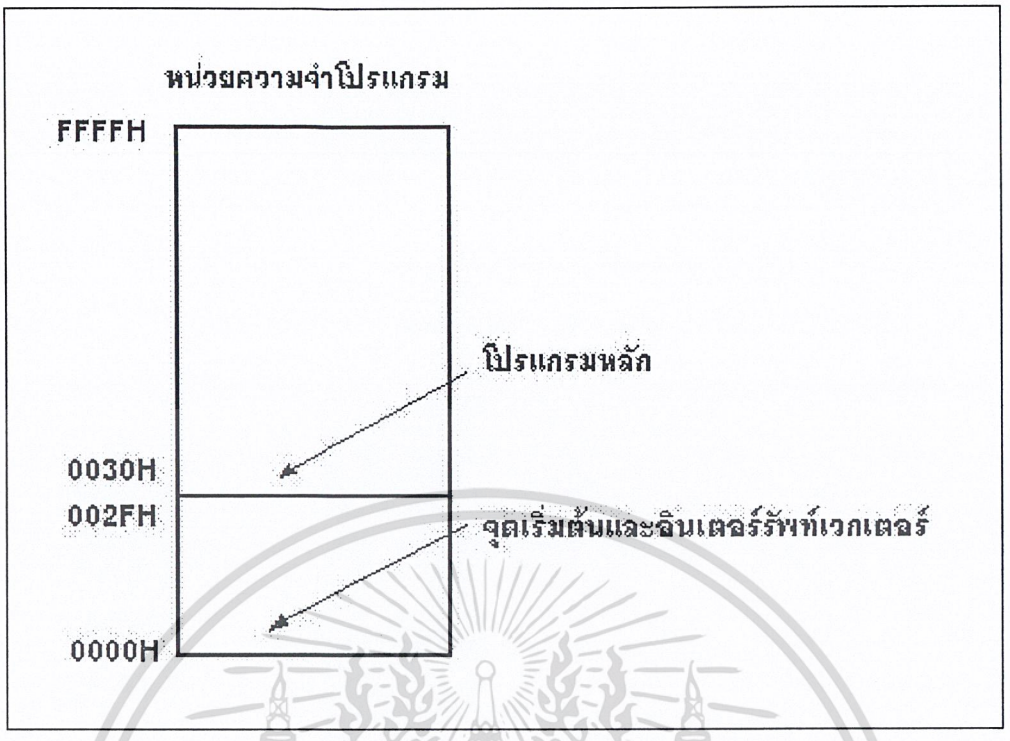
ในการเขียนโปรแกรมหลัก (Main Program) จะต้องกำหนดค่าว่าจะให้ MCS – 51 ถูกอินเตอร์รัพท์ด้วยอะไร และจะให้ MCS – 51 ถูกอินเตอร์รัพท์ได้หรือไม่ โดยการโปรแกรมค่าต่าง ๆ ใน IE รีจีสเตอร์ ถ้ามีการอินเทอร์รัพท์จากสองแหล่งขึ้นไปควรมีการจัดลำดับความสำคัญในรีจีสเตอร์ IP ดังนั้นในโปรแกรมหลักจะต้องมีการโปรแกรมต่อไปนี้

1. โปรแกรมค่าในรีจีสเตอร์ IE
2. โปรแกรมค่าในรีจีสเตอร์ IP

สำหรับโปรแกรมตอบสนองการอินเทอร์รัพท์ถือว่าเป็นโปรแกรมย่อยโปรแกรมหนึ่ง แต่จะต้องจบโปรแกรมย่อยด้วยค่า RETI (Return From Interrupt)

จากตารางอินเตอร์รัพท์เวกเตอร์ จะเห็นว่าถ้ากด Reset หรือให้ระบบเริ่มทำงาน โปรแกรมจะเริ่มทำงานที่ตำแหน่ง 0000H และจะเห็นว่า ตำแหน่งที่เก็บโปรแกรมหลักมีโอกาส อย่างมากที่จะทับกับหน่วยความจำโปรแกรมที่เก็บค่าอินเทอร์รัพท์เวกเตอร์ที่ตำแหน่ง 0003H ถ้าโปรแกรมยาวมากอาจจะไปทับตำแหน่ง 000BH ได้ซึ่งเป็นตำแหน่งของอินเทอร์รัพท์เวกเตอร์ของ Timer 0 ดังนั้นในการเขียนโปรแกรมหลัก ภายใน 3 ตำแหน่งแรก คือ 0000H,0001H,0002H จะต้องกระโดดไปที่อื่นก่อนเพื่อให้ข้ามอินเทอร์รัพท์เวกเตอร์ไป ซึ่งอาจเขียนโปรแกรมได้ดังนี้

```
        ORG   0000H          ; เริ่มต้นโปรแกรม
        LJMP  MAIN           ; กระโดดไปโปรแกรมหลัก
        .....                ; เพื่อหนีอินเทอร์รัพท์เวกเตอร์
        .....
        ORG   0030H          ; ตำแหน่งเริ่มต้นของโปรแกรม
MAIN :  .....                ; เริ่มต้นโปรแกรมหลัก
        .....
```



รูปที่ 4.14 การจัดตำแหน่งโปรแกรมในหน่วยความจำ

จากตัวอย่างโปรแกรมจะเห็นว่า เมื่อเริ่มต้นโปรแกรมหรือระบบถูกรีเซต ระบบจะทำงานตำแหน่งแรก คือคำสั่งกระโดดไปโปรแกรมหลัก ซึ่งอยู่ต่อจากโปรแกรมตอบสนองการอินเตอร์รัพท์ที่อยู่ที่ตำแหน่ง 0030H

โปรแกรมตอบสนองการอินเตอร์รัพท์แบบสั้น

จากตารางอินเตอร์รัพท์เวกเตอร์ จะเห็นว่าที่เก็บโปรแกรมอินเตอร์รัพท์แต่ละแหล่งจะห่างกัน 8 ไบต์ ดังนั้นถ้ามีการอินเตอร์รัพท์จากแหล่งต่าง ๆ หลาย ๆ แหล่งและโปรแกรมตอบสนองการอินเตอร์รัพท์บางโปรแกรมมีขนาดยาวเกิน 8 ไบต์ จะทำให้โปรแกรมไปทับกับตำแหน่งของโปรแกรมตอบสนองการอินเทอร์รัพท์ของอินเทอร์รัพท์ถัดไป แต่ถ้าโปรแกรมตอบสนองการอินเทอร์รัพท์ไม่ยาวมากเกินไปเราสามารถเขียนไปในตำแหน่งนั้นได้เลยดังโปรแกรมต่อไปนี้

```
        ORG         0000H
        LJMP        MAIN  ; กระโดดไปโปรแกรมหลัก
        ORG         000BH ; ตำแหน่งเริ่มต้นของอินเตอร์รัพท์ Timer 0
TOISR :.....
        .....
        RETI              ; กลับโปรแกรมหลัก
MAIN : .....              ; โปรแกรมหลัก
        .....
```

จากตัวอย่างโปรแกรมจะใช้อินเตอร์รัพท์จาก Timer 0 เมื่อระบบเริ่มทำงานจะทำตำแหน่ง 0000H โดยกระโดดไปโปรแกรมหลักซึ่งอยู่ที่ตำแหน่งต่อจากโปรแกรมตอบสนองการอินเทอร์รัพท์เมื่อมีการอินเทอร์รัพท์ Timer 0 ระบบจะทำโปรแกรมตำแหน่งที่ 000BH ซึ่งเป็นอินเตอร์รัพท์เวกเตอร์ของ Timer 0 โดยโปรแกรมตอบสนองการอินเตอร์รัพท์จะจบด้วยคำสั่ง RETI เพื่อกลับสู่โปรแกรมหลักต่อไป

โปรแกรมตอบสนองการอินเตอร์รัพท์ขนาดใหญ่

ในกรณีที่มีการอินเตอร์รัพท์จากหลายแหล่ง และโปรแกรมตอบสนองการอินเตอร์รัพท์แต่ละโปรแกรมยาวเกิน 8 ไบต์ เราไม่สามารถเขียนโปรแกรมตอบสนองการอินเทอร์รัพท์ไว้ที่ตำแหน่งของ อินเทอร์รัพท์เวกเตอร์ได้ ซึ่งจะแก้ปัญหานี้ได้โดยกำหนดให้ตำแหน่งของอินเทอร์รัพท์เวกเตอร์ให้ทำโปรแกรมกระโดด โดยกระโดดไปที่ตำแหน่งเก็บโปรแกรมตอบสนองการอินเทอร์รัพท์ที่เขียนไว้ที่ตำแหน่งอื่นดังตัวอย่าง ต่อไปนี้

```
        ORG   0000H ; เริ่มโปรแกรมของระบบ
        LJMP  MAIN  ; กระโดดไปโปรแกรมหลัก
        ORG   000BH ; ตำแหน่งของอินเตอร์รัพท์ Timer 0
        LJMP  LED1  ; กระโดดไปโปรแกรมตอบสนองการอินเตอร์รัพท์ชื่อ LED1
        ORG   0030H ; ตำแหน่งหลังอินเตอร์รัพท์เวกเตอร์
MAIN : .....        ; โปรแกรมหลัก
        .....
LED1 : .....        ; โปรแกรมตอบสนองการอินเตอร์รัพท์ Timer 1
        .....
        RETI        ; กลับสู่โปรแกรมหลัก
```

จากโปรแกรมจะเห็นว่า เมื่อระบบทำงาน จะต้องทำที่ตำแหน่ง 0000H โดยกระโดดไปทำโปรแกรมหลักที่ตำแหน่งต่อจาก 0030H เพราะตำแหน่งดังกล่าวข้ามอินเตอร์รัพท์เวกเตอร์จากแหล่งต่าง ๆ ไปแล้ว เมื่อมีการอินเตอร์รัพท์จาก Timer 0 โปรแกรมจะต้องทำงานที่ตำแหน่ง 000BH แต่โปรแกรมตอบสนองการอินเตอร์รัพท์ยาวมาก ที่ตำแหน่ง 000BH จึงให้ทำโปรแกรมกระโดดโดยกระโดดไปที่โปรแกรมตอบสนองการอินเตอร์รัพท์ชื่อ LED1 ซึ่งอยู่ท้ายโปรแกรม เมื่อจบโปรแกรมจะจบด้วยคำสั่ง RETI เพื่อกลับไปโปรแกรมหลักต่อไป

บทที่ 5
มอเตอร์กระแสตรง

5.1 หลักการทำงานของมอเตอร์มอเตอรกระแสตรง

ดีซีมอเตอร์เป็นทรานสดิวเซอร์แรงบิดซึ่งมีการออกแบบให้มีคุณลักษณะพิเศษคือแรงบิดของดีซีมอเตอร์จะเป็นสัดส่วนโดยตรงกับกระแสอาร์มาเจอร์ แรงบิดของเพลาของดีซีมอเตอร์จะได้จากผลระหว่างสนามแม่เหล็กและขดลวดตัวนำ ในที่นี้ กระแสที่ไหลในขดลวดตัวนำจะสร้างฟิลด์ที่ประกอบด้วยเส้นแรงแม่เหล็ก ϕ และขดลวดตัวนำเหล่านั้นอยู่ห่างจากศูนย์กลางการหมุนเท่ากับ r ความสัมพันธ์ระหว่างแรงบิดของเพลาและกระแสเท่ากับ

$$T = K\phi I \qquad (5\text{-}1)$$

เมื่อ T คือ แรงบิดของเพลามีหน่วยเป็นนิวตัน-เมตร

ϕ คือ เส้นแรงแม่เหล็กมีหน่วยเป็นเวเบอร์

I คือ กระแสเป็นแอมแปร์

และ K คือ ค่าคงที่

ดังนั้นแรงบิดของเพลาจะเป็นสัดส่วนโดยตรงกับผลคูณของเส้นแรงแม่เหล็กและกระแสเมื่อขดลวดตัวนำเคลื่อนที่ในสนามแม่เหล็กก็จะทำให้เกิดโวลท์เต็จตกคร่อมตัวมันเอง โวลท์เต็จนี้จะเป็นสัดส่วนกับความเร็วของเพลาของมอเตอร์และต้านการไหลของกระแส ความสัมพันธ์ระหว่างโวลท์เต็จย้อนกลับนี้และความเร็วของเพลามอเตอร์คือ

$$E = K\phi\omega \qquad (5\text{-}2)$$

เมื่อ E คือแรงดันย้อนกลับ emf มีหน่วยเป็นโวลต์

ϕ คือ เส้นแรงแม่เหล็กมีหน่วยเป็นเวเบอร์

ω คือ ความเร็วของมอเตอร์มีหน่วยเป็นเรเดียน/วินาที

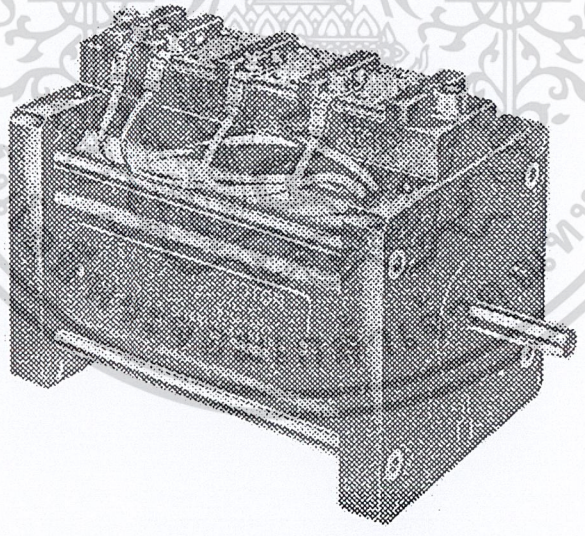
5.2 ประเภทของมอเตอร์กระแสตรง

ดีซีมอเตอร์สามารถแบ่งออกได้เป็นหลายประเภทขึ้นอยู่กับลักษณะวิธีการสร้างสนามแม่เหล็กของตัวมอเตอร์และขึ้นอยู่กับพื้นฐานการออกแบบโครงสร้างของอาร์มาเจอร์ การแบ่งประเภทตามลักษณะการจ่ายสนามแม่เหล็กแยกออกได้เป็น 2 แบบคือ

1. ดีซีมอเตอร์แบบปรับเส้นแรงแม่เหล็กได้
2. ดีซีมอเตอร์แบบเส้นแรงแม่เหล็กมีค่าคงที่

ถ้าเราพิจารณาแยกประเภทตามลักษณะการออกแบบโครงสร้างอาร์มาเจอร์สามารถแยกออกได้เป็น 3 แบบ คือ

1. ดีซีมอเตอร์แบบอาร์มาเจอร์เป็นแกนเหล็ก
2. ดีซีมอเตอร์แบบอาร์มาเจอร์ที่มีขดลวดพันอยู่บนพื้นผิว
3. ดีซีมอเตอร์แบบอาร์มาเจอร์เป็นขดลวดหมุน



รูปที่ 5.1 เซอร์โวมอเตอร์แบบโรเตอร์เป็นแกนเหล็ก

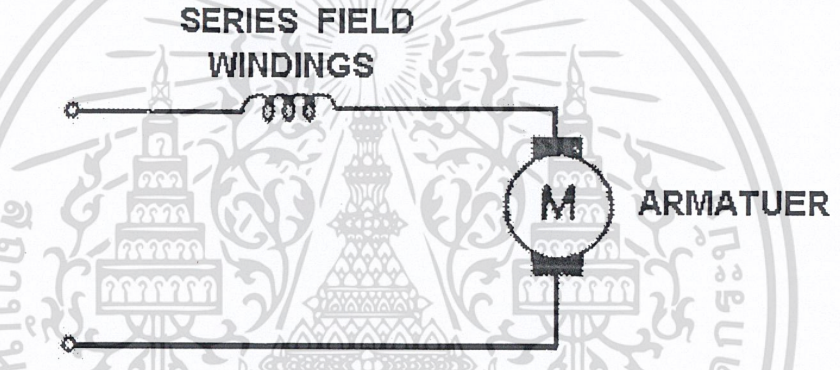
5.2.1 ดีซีมอเตอร์แบบปรับเส้นแรงแม่เหล็กได้

ดีซีมอเตอร์แบบปรับเส้นแรงแม่เหล็กได้ยังแบ่งแยกได้เป็น 2 แบบคือ

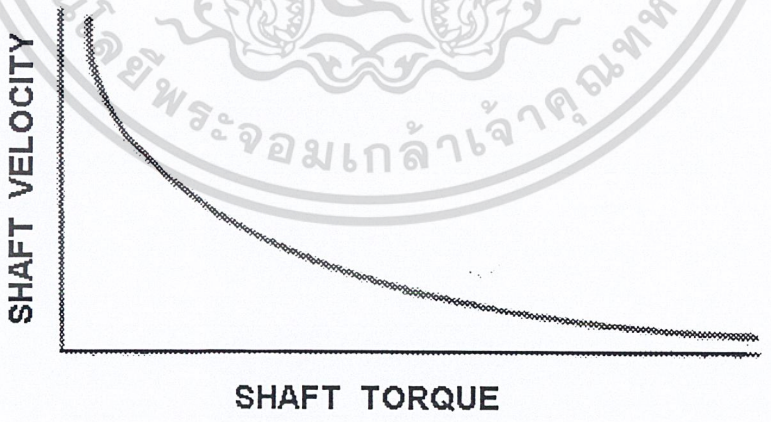
ก). แบบขดลวดสนามแม่เหล็กต่ออนุกรมกับขดลวดอาร์มาเจอร์

ข). แบบขดลวดสนามแม่เหล็กแยกกระตุ้น

ตัวอย่างของมอเตอร์แบบขดลวดสนามแม่เหล็กต่ออนุกรมแสดงดังในรูป 5.2 มอเตอร์แบบนี้จะมีเส้นแรงแม่เหล็กเป็นสัดส่วนกับกระแสดังนั้นเส้นแรงของสนามแม่เหล็กจึงสามารถปรับค่าได้ และเราจะได้ความสัมพันธ์ระหว่างความเร็วและแรงบิดเป็นนอนลิเนียร์ดังแสดงในรูป 5.3



รูปที่ 5.2 ดีซีมอเตอร์แบบอาร์มาเจอร์ต่ออนุกรมกับขดลวดสนามแม่เหล็ก

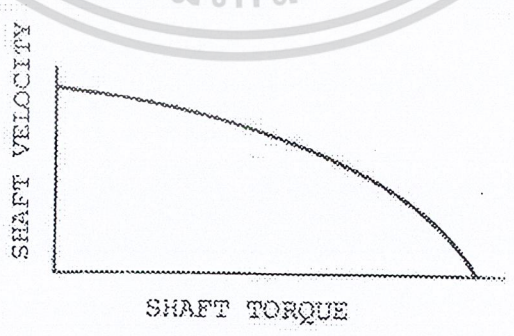


รูปที่ 5.3 คุณสมบัติระหว่างความเร็วและแรงบิดของดีซีมอเตอร์อนุกรมภายใต้

5.2.1.1 ภาวะโวลต์เทจคงที่

มอเตอร์ต่อดังกล่าวจะใช้งานในภาวะเฉพาะเมื่อต้องการแรงบิดสูงที่ความเร็วต่ำ และแรงบิดต่ำที่ความเร็วสูงเช่นระบบการขับเคลื่อนที่ของรถลากตัวอย่างของดีซีมอเตอร์แบบขดลวดสนามแม่เหล็กแยกกระตุ้นแสดงในรูปที่5.4 ดีซีมอเตอร์แบบนี้มักนิยมเรียกกันว่ามอเตอร์ขนาน (shunt motor) มอเตอร์แบบนี้สามารถปรับเส้นแรงแม่เหล็กได้อย่างอิสระต่อกระแสของอาร์มาเจอร์ยังผลให้สามารถควบคุมพารามิเตอร์ของมอเตอร์ให้มีค่าคงที่ได้ตลอดช่วงพิสัยที่กว้าง มอเตอร์นี้มักจะใช้งานในกรณีระบบบังคับการเคลื่อนที่ต้องการแรงบิดสูง ในรูปที่ 5.5 แสดงถึงคุณสมบัติระหว่างแรงบิดกับความเร็วของชันท์มอเตอร์ภายใต้ภาวะการกระตุ้นสนามแม่เหล็กคงที่และอาร์มาเจอร์โวลท์เต็จคงที่

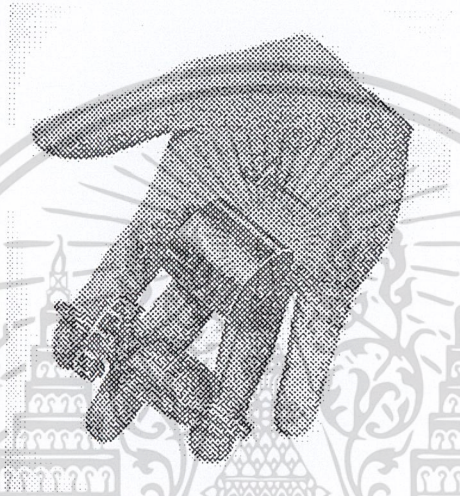
รูปที่ 5.4 ดีซีมอเตอร์แบบแยกปรับสนามแม่เหล็กได้



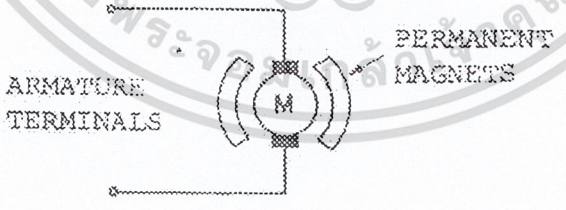
รูปที่ 5.5 แสดงคุณสมบัติระหว่างความเร็วและแรงบิดของชันท์มอเตอร์

5.2.2 ดีซีมอเตอร์แบบเส้นแรงแม่เหล็กคงที่

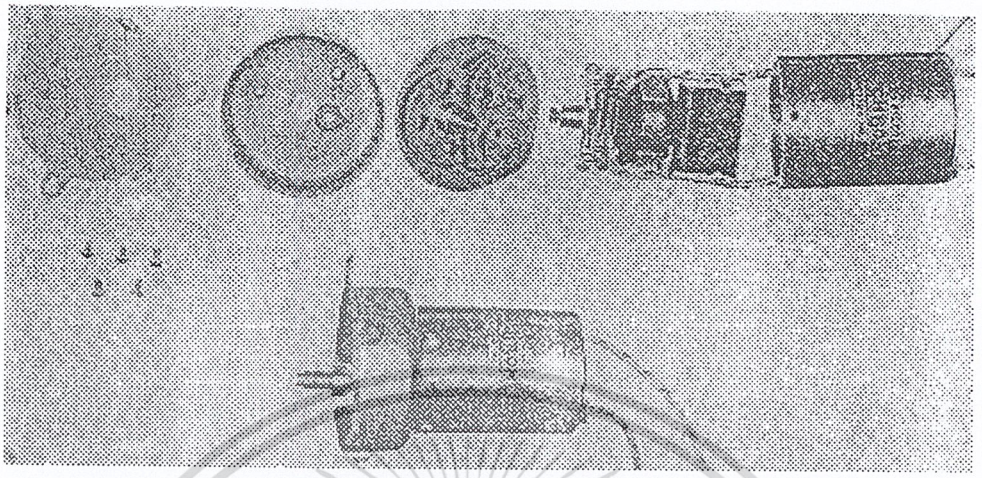
ระบบการกระตุ้นฟิลด์ของมอเตอร์โดยทั่วไปในปัจจุบันมักใช้เป็นแบบ แม่เหล็กถาวร ดังแสดงในรูป 5.6 ในระบบนี้เส้นแรงของฟิลด์มีค่าคงที่ดังนั้น อัตราส่วนระหว่างกระแสอาร์มาเจอร์และแรงบิดจะมีค่าคงที่



รูปที่ 5.6 ชันท์มอเตอร์แบบแม่เหล็กถาวร



รูปที่ 5.7 ดีซีมอเตอร์แบบฟิลด์เป็นแม่เหล็กถาวร



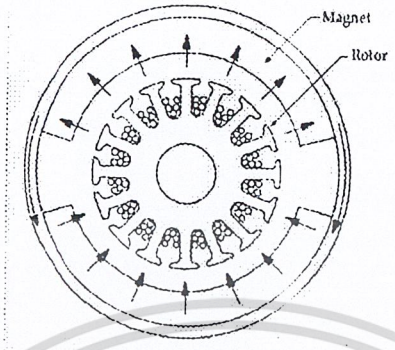
รูปที่ 5.8 ดีซีมอเตอร์แบบแม่เหล็กถาวรและเกียร์บล็อคสำหรับลดความเร็ว

5.2.2.1 ข้อดีของมอเตอร์แบบฟิลด์แม่

เหล็กถาวรซึ่งเหนือกว่ามอเตอร์แบบมีโครงสร้างของฟิลด์ด้วยการพันของขดลวดคือ ไม่มีกำลังสูญเสียในฟิลด์ มีประสิทธิภาพสูงกว่าและมีขนาดเล็กกว่าเมื่อเทียบมอเตอร์ที่มีขนาดของกำลังม้าเท่ากัน นอกจากนั้นความสัมพันธ์เชิงเส้นยังให้ค่าของกระแสอาร์มาเจอร์ที่สูงกว่าดีซีมอเตอร์แบบฟิลด์เป็นขดลวด การประยุกต์ใช้งานเหมาะสมกับระบบที่ต้องการแรงบิดของโหลดสูง

5.2.3 ดีซีมอเตอร์แบบอาร์มาเจอร์เป็นแกนเหล็ก

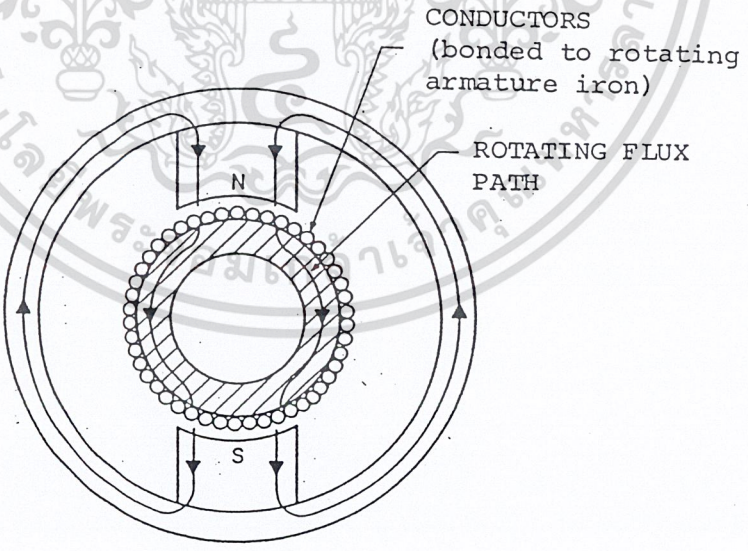
โครงสร้างของโรเตอร์และสเตทเตอร์ของมอเตอร์แบบแกนเหล็กแสดงดังในรูปที่ 5.9 โครงสร้างของมอเตอร์แบบนี้มีโมเม้นท์ของแรงเฉื่อยสูงที่สุดและมีค่าอินดัคแต้นซ์ของโรเตอร์สูงที่สุดด้วยดังนั้นมอเตอร์นี้จึงมีปริมาณการจุความร้อนได้สูง และสามารถจะทนโอเวอร์โหลดได้ในระยะเวลาที่ยาวนานโดยไม่ทำให้มอเตอร์เสียหาย



รูปที่ 5.9 แสดงรูปหน้าตัดของดีซีมอเตอร์แบบอาร์มาเจอร์เป็นแกนเหล็กส่วนฟิลด์เป็นแม่เหล็ก

5.2.4 ดีซีมอเตอร์แบบอาร์มาเจอร์มีขดลวดพันอยู่บนพื้นผิว

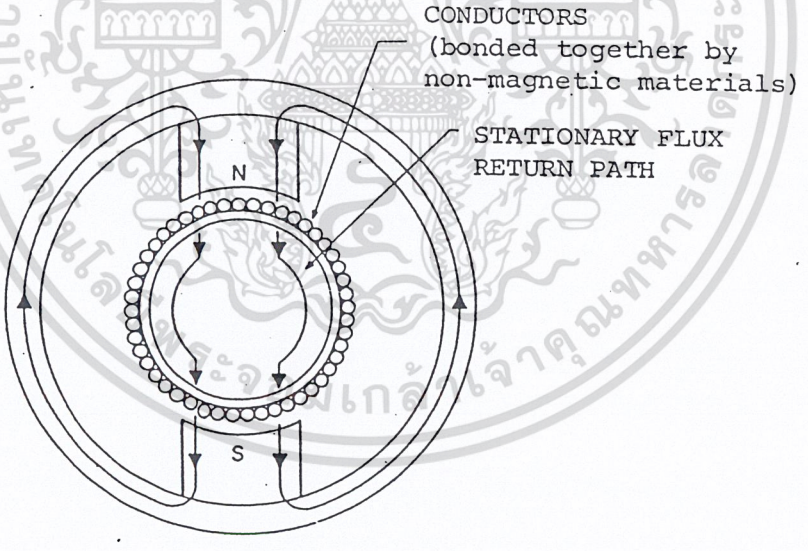
ในรูปที่ 5.10 แสดงถึงการออกแบบของโรเตอร์ที่มีขดลวดพันอยู่บนพื้นผิว โดยไม่มีฝาสลอตทำให้ได้อินดัคแต้นซ์ของโรเตอร์ต่ำกว่าแบบแกนเหล็ก ข้อเสียคือ ทำให้ขนาดของมอเตอร์แบบนี้ใหญ่ขึ้นและราคาแพงกว่าแบบแกนเหล็กด้วย



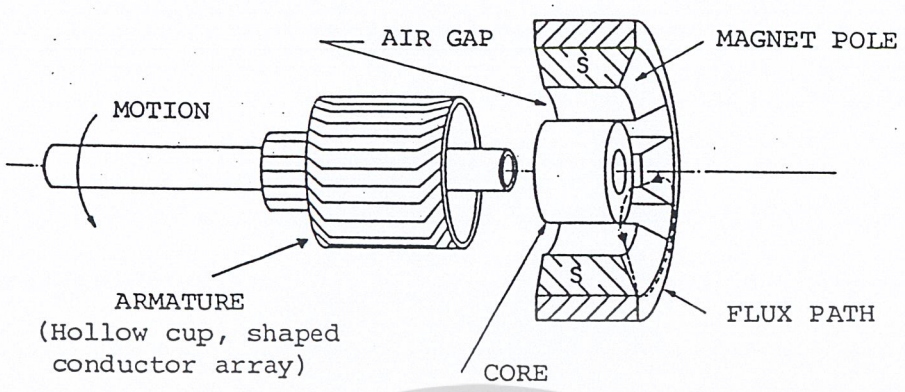
รูปที่ 5.10 แสดงรูปหน้าตัดของดีซีมอเตอร์แบบมีขดลวดบนพื้นผิวและฟิลด์เป็นแม่เหล็กถาวร

5.2.5 ดีซีมอเตอร์แบบอาร์มาเจอร์เป็นขดลวดหมุน

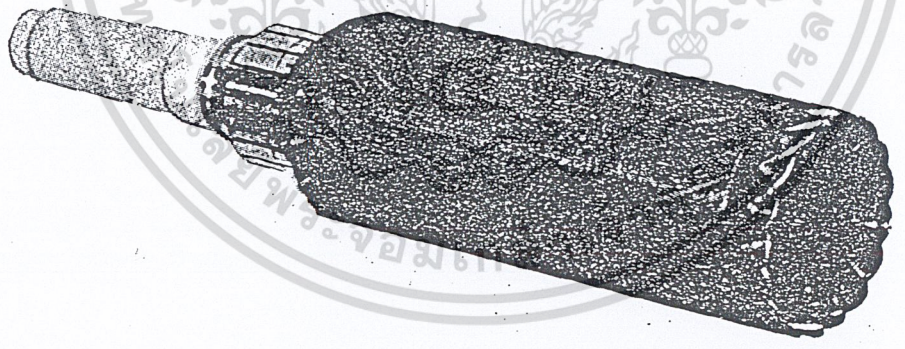
มอเตอร์แบบขดลวดหมุนนี้ได้รับการออกแบบเพื่อให้มีโมเม้นของแรงเฉื่อยน้อยมากดังแสดงในรูปที่ 5.11 และรูปหน้าตัดซึ่งมองด้านข้างของมอเตอร์ดังกล่าวแสดงในรูปที่ 5.12 นอกจากนั้นในรูป 5.13 และรูป 5.14 เป็นรูปถ่ายของอาร์มาเจอร์แบบเป็นขดลวดเคลื่อนที่ และรูปลักษณะโครงสร้างภายนอกและโครงสร้างของแม่เหล็กมอเตอร์แบบนี้มีช่องว่างอากาศ (air-gap) ระหว่างแม่เหล็กมากกว่ามอเตอร์ทั้งสองแบบที่กล่าวมาแล้ว ดังนั้นจำเป็นต้องออกแบบให้โครงสร้างของแม่เหล็กให้ใหญ่ขึ้นเพื่อให้ได้ช่องว่างของอากาศระหว่างเส้นแรงแม่เหล็กที่เท่ากันกับของมอเตอร์ทั้งสองแบบดังกล่าว ดังนั้นราคาของมอเตอร์แบบนี้จึงมีราคาแพง นอกจากนั้นโครงสร้างของโรเตอร์มีความจุความร้อนต่ำมากถ้าหากเกิดโอเวอร์โหลดก็จะทำให้มอเตอร์เสียได้ง่ายและโรเตอร์ลักษณะนี้จะมีค่าอินดัคแต้นซีต่ำมากคือน้อยกว่า 10 ไมโครเฮนรี่



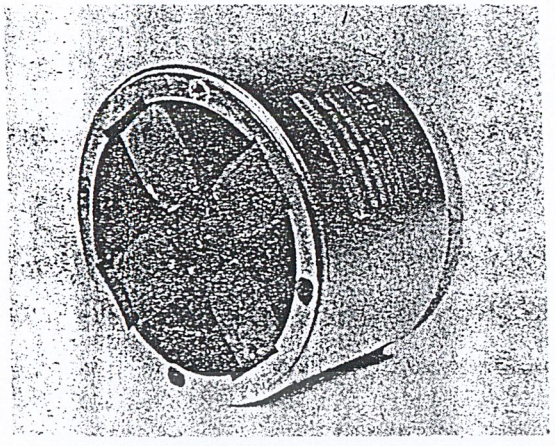
รูปที่ 5.11 หน้าตัดของดีซีมอเตอร์แบบมีโรเตอร์เป็นขดลวดเคลื่อนที่และฟิลด์เป็นแม่เหล็กถาวร



รูปที่ 5.12 หน้าตัดของดีซีมิเตอร์แบบมีโรเตอร์เป็นขดลวดเคลื่อนที่

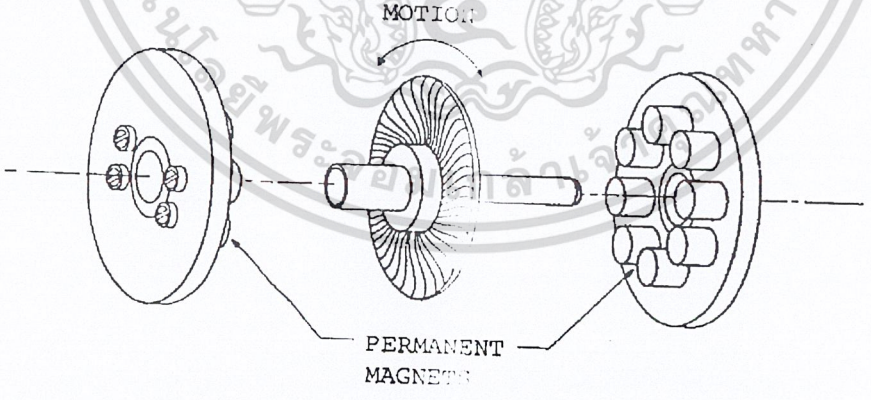


รูปที่ 5.13 อาร์มาเจอร์แบบเป็นขดลวดเคลื่อนที่ซึ่งมีรูปร่างเป็นถ้วยตรงกระบอกและเพลาเอาท์พุททำด้วยเซรามิคอาลูมิเนียมเพื่อทนต่อแรงดึงได้สูง



รูปที่ 5.14 รูปร่างและโครงสร้างแม่เหล็กของมอเตอร์แบบขดลวดเคลื่อนที่

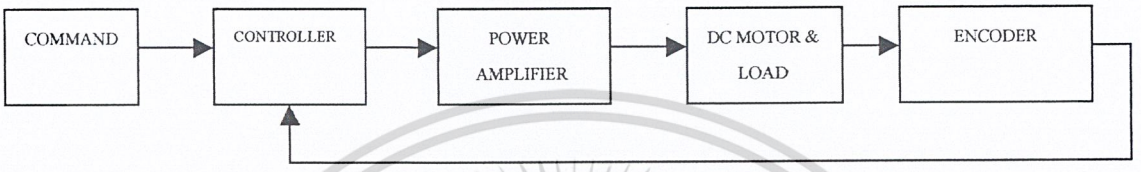
ในรูปที่ 5.15 แสดงถึงมอเตอร์แบบขดลวดเคลื่อนที่อีกลักษณะหนึ่งซึ่งมีโครงสร้างของอาร์มาเจอร์เป็นรูปร่างจานซึ่งทำขึ้นจากขดลวดตัวนำซ้อนกันหลาย ๆ ชั้น ซึ่งเรามักจะเรียกกันว่า "printed motor" ในปัจจุบันดีซีมอเตอร์แบบโรเตอร์เป็นขดลวดหมุนนี้ให้คุณสมบัติการทำงานที่ดีเยี่ยมเหมาะสมสำหรับเป็นตัวขับเคลื่อนที่ในระบบการบังคับตำแหน่ง และยังให้อัตราส่วนระหว่างแรงบิดและแรงเฉื่อยได้สูง และมีค่าอินดักแต้นซ์ต่ำที่สุดเมื่อเทียบกับมอเตอร์แบบอื่น ๆ นอกจากนั้นความสามารถในการเพิ่มอัตราเร่งยังกระทำได้สูง 10^6 เรเดียน/วินาที^{2}



รูปที่ 5.15 ดีซีมอเตอร์แบบโรเตอร์ขดลวดหมุนมีรูปร่างเป็นจาน

5.3 ระบบควบคุมมอเตอร์กระแสตรง

ส่วนประกอบพื้นฐานของระบบคอนโทรลดีซีมอเตอร์แสดงได้ในบล็อคไดอะแกรมของรูป 1.3 ซึ่งประกอบด้วยบล็อคที่สำคัญ 4 บล็อคคือ



รูปที่5.16 บล็อคไดอะแกรมระบบคอนโทรลดีซีมอเตอร์

ตัวคอนโทรลเลอร์ เป็นส่วนของระบบที่ทำให้เกิดสัญญาณคอนโทรลเลอร์ไปบังคับดีซีมอเตอร์และโหลด คอนโทรลเลอร์ที่ให้สัญญาณคอนโทรลเป็นสัญญาณอนาลอก เรียกว่าอนาลอกคอนโทรลเลอร์ ส่วนคอนโทรลเลอร์ที่ให้สัญญาณคอนโทรลเป็นสัญญาณดิจิตอลเราเรียกว่าดิจิตอลคอนโทรลเลอร์

วงจรไดร์ฟเวอร์ เป็นส่วนประกอบของระบบที่อยู่ระหว่างตัวคอนโทรลเลอร์กับดีซีมอเตอร์และโหลดมีหน้าที่ปรับรูป และขยายสัญญาณให้เหมาะสมก่อนที่จะป้อนเข้าไปขับดีซีมอเตอร์และโหลด วงจรไดร์ฟเวอร์ส่วนใหญ่ได้แก่เพาเวอร์แอมปริไฟ ซึ่งอาจจะแบ่งย่อยออกเป็น ลิเนียร์เพาเวอร์แอมป์ปลิ-ไฟและพัลส์โมดูเลทชั่นแอมปลิไฟ

ฟีดแบ็ดทรานสดิวเซอร์หรือเอนโคดเดอร์ เป็นสิ่งประดิษฐ์ที่ใช้รับรู้หรือดีเท็คสัญญาณเอาท์พุทที่ต้องการ โดยไม่มีผลของการโหลดดิ้ง สัญญาณที่ดีเท็คได้นี้จะป้อนกลับไปเปรียบเทียบกับสัญญาณอ้างอิงทำให้ได้สัญญาณ Error ฟีดแบ็ดทรานสดิวเซอร์แบ่งออกได้เป็น 2 แบบ คือ อนาลอกทรานสดิวเซอร์ คือ สิ่งประดิษฐ์ใช้เปลี่ยนพลังงานรูปหนึ่งให้เป็นสัญญาณอนาลอกได้แก่พวก ทาโคเยนเนอเรตอร์ โปเทนทิโอมิเตอร์และซิงโคร เป็นต้น ส่วนฟีดแบ็คทรานสดิวเซอร์อีกแบบหนึ่งคือ ดิจิตอลทราน สดิวเซอร์ เป็นสิ่งประดิษฐ์ที่ใช้เปลี่ยนพลังงานรูปหนึ่งให้เป็นสัญญาณดิจิตอล ได้แก่พวกอินครีเม้นท์เอนโคดเดอร์รีโซลเวอร์ แมกนิติคพิคอัพ เป็นต้น

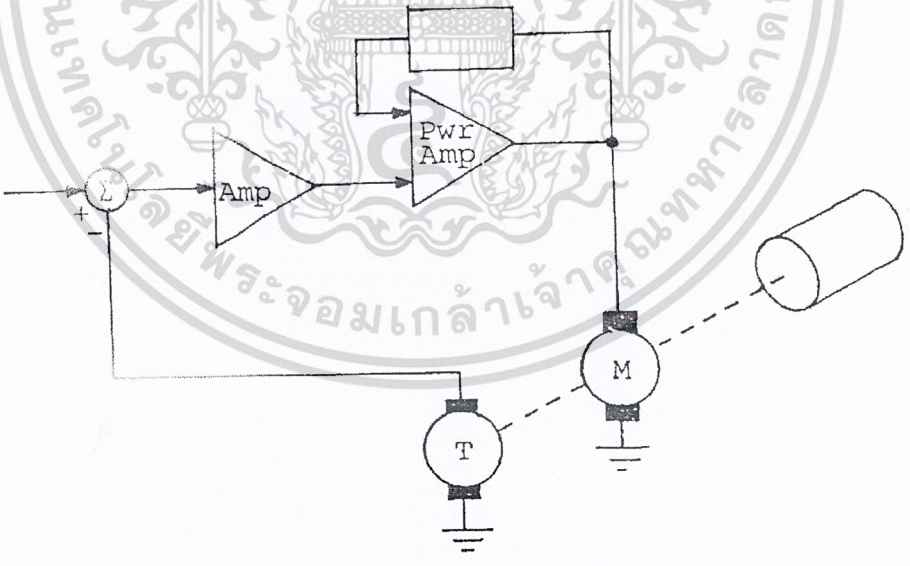
ดีซีมอเตอร์และโหลด คือระบบที่ถูกคอนโทรลหรือส่วนที่ออกแรงทำงานซึ่งจะเป็นเครื่องจักรกล (ดีซีมอเตอร์) ดีซีมอเตอร์ในที่นี้เป็นแบบแม่เหล็กถาวรที่มีคุณสมบัติการทำงานสูง มีอาร์มาเจอร์อินดัคแต้นซ์และแรงเฉื่อยของโรเตอร์ต่ำ

5.3.1 ลักษณะการควบคุมของระบบดีซีมอเตอร์

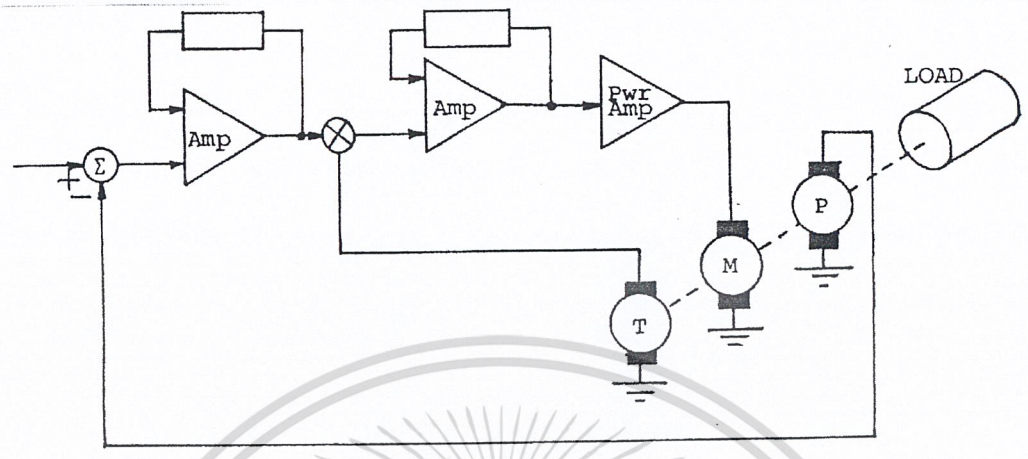
ระบบการคอนโทรลดีซีมอเตอร์สามารถที่จะจำแนกลักษณะการคอนโทรลออกได้เป็น 2 แบบ คือระบบอนาลอกคอนโทรลและระบบดิจิตอลคอนโทรล

5.3.1.1 ระบบอนาลอกคอนโทรล

ในคอนโทรลูปของระบบอะนาลอกคอนโทรล เอาท์พุทของระบบจะถูกวัดค่าหรือดีเท็คค่าได้เป็นสัดส่วนกับสัญญาณไฟฟ้าที่เป็นสัญญาณอนาลอก เช่นระบบที่มีการป้อนกลับด้วยทาโคมิเตอร์ในรูปที่ 5.16 โวลท์เตจเอาท์พุทของทาโคมิเตอร์จะเป็นอนาลอกของความเร็วมอเตอร์ ในทำนองเดียวกันระบบการบังคับตำแหน่งดังรูปที่ 5.17 ไดนามิควาริเอเบิ้ล (การเปลี่ยนแปลงตำแหน่งของโหลด) จะเป็นสัดส่วนกับโวลท์เตจเอาท์พุทที่ได้จากโปเทนมิเตอร์ นั้นคือตำแหน่งเอาท์พุทของระบบเป็นอะนาลอกของไดนามิกวาริเอเบิ้ล

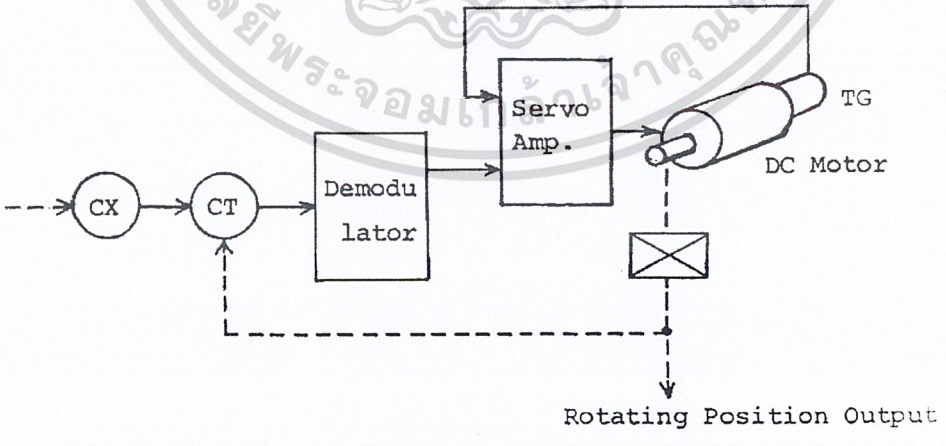


รูปที่ 5.17 บล็อคไดอะแกรมของระบบการบังคับความเร็วที่มีทาโคมิเตอร์เป็นตัวป้อนกลับ



รูปที่ 5.18 บล็อคไดอะแกรมของระบบการบังคับตำแหน่งที่มีทาโคมิเตอร์เป็นตัวเสริมเสถียรภาพให้ดีขึ้น

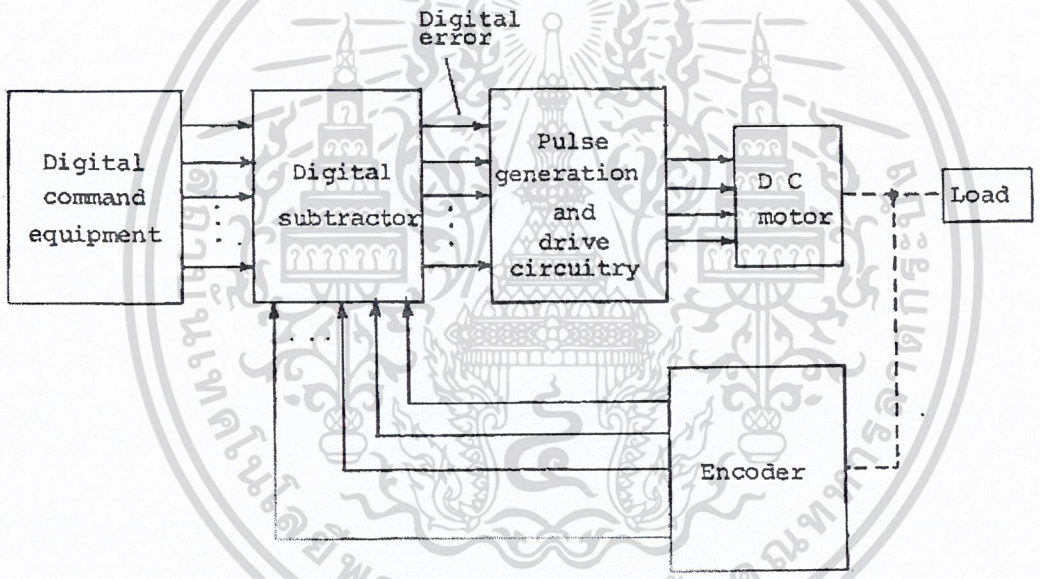
สรุปได้ว่าระบบอนาลอกคอนโทรลลูฟที่มีฟีดแบ็คทรานสดิวเซอร์เป็นอนาลอกทรานสดิวเซอร์วัดค่าไดนามิควาริเอเบิ้ลออกเป็นสัญญาณอนาลอกป้อนกลับไปยังตัวอนาลอกคอนโทรลเอลร์เพื่อคอนโทรลให้ได้คุณสมบัติการทำงานเป็นไปตามที่ต้องการ ตัวอย่างของระบบอนาลอกคอนโทรลของระบบดีซีมอเตอร์แสดงได้ดังในรูปที่ 5.19



รูปที่ 5.19 ระบบการคอนโทรลตำแหน่งด้วยสัญญาณซิงโคร

5.3.1.2 ระบบดิจิตอลคอนโทรล

ระบบดิจิตอลคอนโทรลคือระบบที่คอนโทรลลูฟมีฟีดแบ็คทรานสดิวเซอร์เป็นดิจิตอลทรานสดิวเซอร์ ซึ่งสามารถวัดค่าไดนามิกวาริเอเบิ้ลออกเป็นสัญญาณดิจิตอลหรือในรูปของสัญญาณเอนโคดดิ้งของไบนารีคือสัญญาณเอาท์พุทของทรานสดิวเซอร์จะเป็นโค๊ดไบนารีป้อนกลับไปยังตัวดิจิตอลคอนโทรลเลอร์เพื่อคอนโทรลให้คุณสมบัติการทำงานของระบบเป็นไปตามที่ต้องการ บล็อคไดอะแกรมของระบบดิจิตอลคอนโทรลของดีซีมอเตอร์แสดงได้ในรูปที่ 5.19



รูปที่ 5.20 บล็อคไดอะแกรมของระบบดิจิตอลคอนโทรลของดีซีมอเตอร์

5.3.2 วิธีการควบคุมมอเตอร์

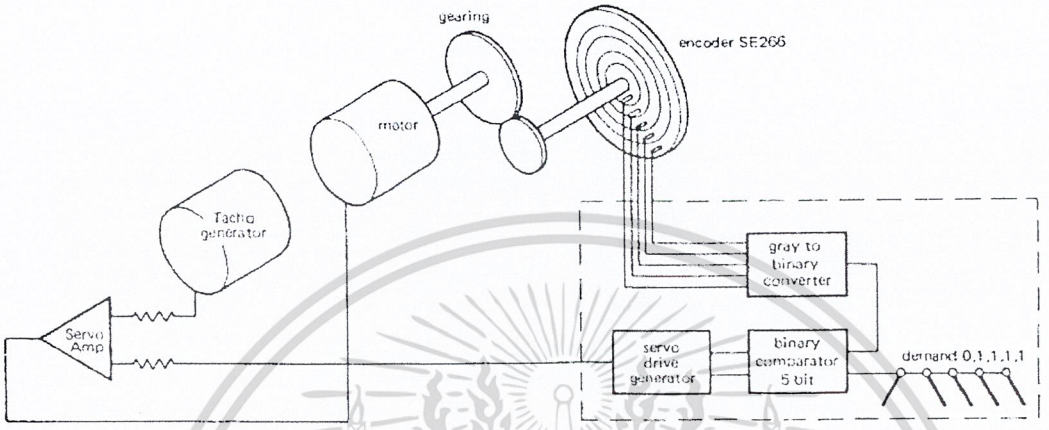
5.3.2.1 การคอนโทรลมอเตอร์ด้วยดีซีมอเตอร์ด้วยดีซีโวลท์เต็จ

ก). เฟสคอนโทรลโดยใช้ไทริสเตอร์ (SCR)

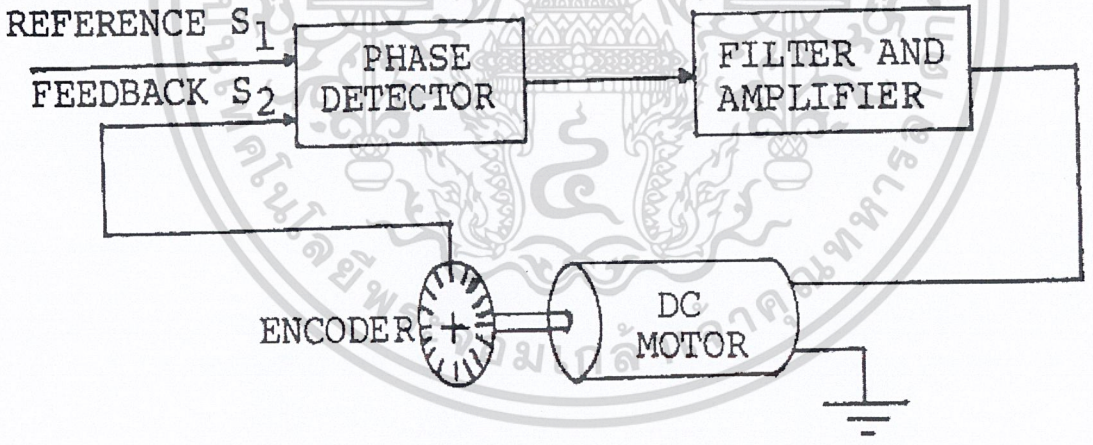
ข). โวลท์เต็จคอนโทรลโดยใช้ทรายซิสเตอร์

ค). ออน-ออฟ คอนโทรลโดยใช้ทรานซิสเตอร์และไทริสเตอร์

ง). ไทม์เร โซคอนโทรล



รูปที่ 5.21 ระบบการคอนโทรลดีซีมอเตอร์ด้วยตัวเลขไบนารี



รูปที่ 5.22 ระบบการบังคับความเร็วดีซีมอเตอร์แบบเฟสล็อคลูป

5.3.2.2 การคอนโทรลมอเตอร์ด้วยเอซีโวลท์เต็จ

ก). เฟสคอนโทรลด้วยไทรริสเตอร์ (TRIAC)

ข). อินติกรอลคอนโ?รลด้วยไทรริสเตอร์

5.3.2.3 การคอนโทรลมอเตอร์ด้วยการคอนโทรลความถี่

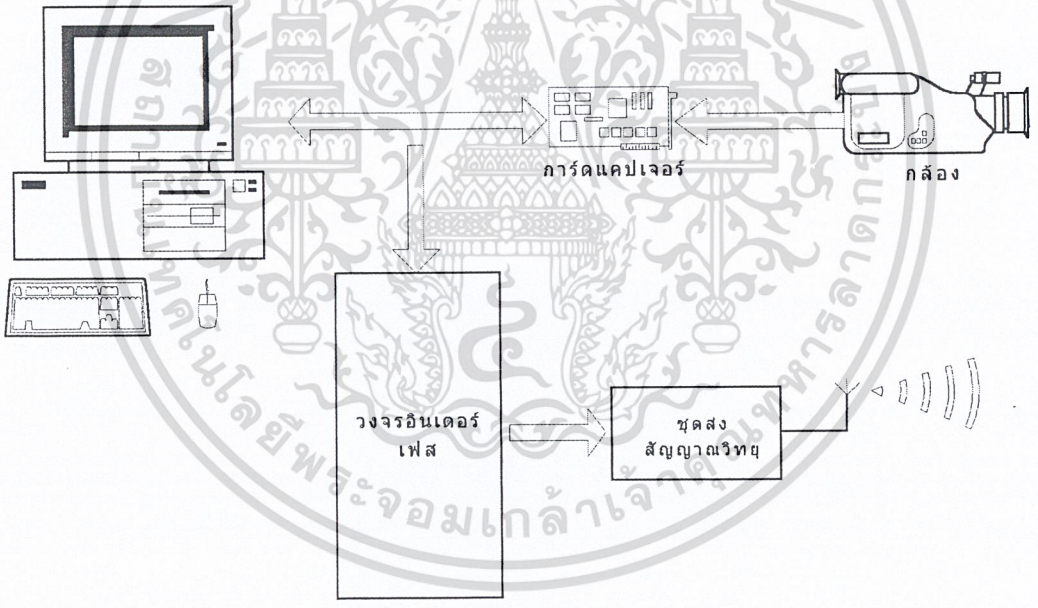
ก). อินเวอทเทอด้วยทรานซิสเตอร์

ข). อินเวอเทอด้วยไทรริสเตอร์

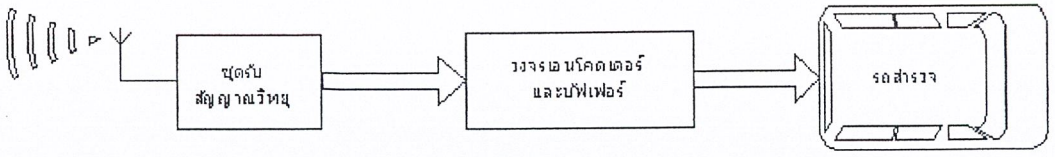
ค). เฟสล็อคลูฟคอนโทรล

บทที่ 6
การออกแบบส่วนโครงสร้าง

ในขั้นตอนการออกแบบโครงงาน ได้แบ่งเป็น 2 ส่วนคือส่วนของฮาร์ดแวร์ และส่วนของโปรแกรมควบคุม ส่วนของฮาร์ดแวร์ประกอบด้วยตัวรถสำรวจซึ่งมีวงจรขับมอเตอร์กล้องที่ติดอยู่กับตัวรถเพื่อส่งภาพมายังฐานควบคุม ส่วนของภาคอินเตอร์เฟสโดยในโครงงานนี้เลือกการอินเตอร์เฟสทางพอร์ทขนาน ซึ่งสามารถนำไปใช้กับเครื่องคอมพิวเตอร์อื่นๆได้ง่าย เนื่องจากเป็นมาตรฐานที่มีใช้ทั่วไป ในส่วนโปรแกรมควบคุม ผู้ทำโครงงานเลือกใช้การทำงานภายใต้ระบบปฏิบัติการ WINDOWS 98 SE โดยใช้คอมไพล์เลอร์(Compliler) ของ Boland C++ Builder 5.0 (BCB 5.0) เพราะมีเครื่องมือ(Component) และการเขียนโปรแกรมที่ง่ายในการใช้งาน ในบทนี้จะกล่าวรายละเอียดของส่วนฮาร์ดแวร์ซึ่งประกอบด้วย



รูปที่ 6.1 ภาพแสดงในส่วนของฐานควบคุม



รูปที่ 6.2 ภาพแสดงในส่วนที่ติดอยู่กับตัวรถสำรวจ

1. ส่วนประมวลผลและควบคุม (Processor and Control)
2. วงจรอินเตอร์เฟส
3. ชุดรับส่งสัญญาณวิทยุ(ความถี่ 49 MHz)
4. วงจรดีโคดเดอร์และบัฟเฟอร์(Decode And Buffer)
5. แหล่งจ่ายพลังงาน (Power Supply)
6. กล้องวีดีโอ
7. รถสำรวจ

6.1 การทำงาน

จากรูปที่ 6.1 เครื่องคอมพิวเตอร์ทำหน้าที่ประมวลผลคือสร้างสัญญาณพัลส์เพื่อใช้ในการเข้ารหัสโดยสัญญาณพัลส์ที่ได้จะถูกส่งออกทางชุดอินเตอร์เฟสสัญญาณ(ถูกเข้ารหัสเรียบร้อยแล้ว) จะถูกส่งไปยังชุดส่งสัญญาณวิทยุส่งออกอากาศต่อไป สัญญาณวิทยุที่ส่งออกมาจากชุดส่งจะถูกส่งไปยังชุดรับสัญญาณที่ติดอยู่กับตัวรถ

จากรูปที่ 6.2 เมื่อวงจรภาครับได้รับสัญญาณจะทำการนำสัญญาณที่ได้รับมานั้นแปลงให้เป็นสัญญาณพัลส์ตามที่เราต้องการ จากนั้นพัลส์ที่ได้จะถูกส่งไปยังวงจรดีโคดเดอร์และวงจรบัฟเฟอร์(Decoder And Buffer) วงจรดีโคดเดอร์และวงจรบัฟเฟอร์จะเปลี่ยนสัญญาณพัลส์ให้เป็นสัญญาณ ข้อมูล(เลขฐานสอง) เพื่อนำไปใช้ในการควบคุมรถสำรวจโดยผ่านวงจรไดร์ฟมอเตอร์ ในส่วนถัดไป เมื่อมีสัญญาณเข้ามาที่อินพุทของวงจรไดร์ฟมอเตอร์ วงจรไดร์ฟมอเตอร์จะทราบทันทีว่าเป็นสัญญาณที่สั่งให้ตัวมันทำงานเช่นไรโดยรหัสได้ถูกเขียนเอาไว้ในตัวไมโครคอนโทรลเลอร์ ยกตัวอย่างเช่น มีสัญญาณ 0010 เข้ามาที่อินพุทของวงจรไดร์ฟซึ่งเป็นสัญญาณที่สั่งให้รถเลี้ยวซ้าย ตัวไมโครคอนโทรลเลอร์จะรับค่าอินพุทนี้มาทำการประมวลผลจากนั้นจึงส่งสัญญาณบังคับเลี้ยวให้กับมอเตอร์เพื่อให้รถเลี้ยวตามคำสั่งที่ได้รับต่อไป

6.2 ส่วนประมวลผลและควบคุม (Processor and Control)

ในการประมวลผล เลือกใช้เครื่องคอมพิวเตอร์เพื่อการประมวลผล โดยมีอุปกรณ์ประกอบที่มีคุณสมบัติอย่างต่ำ ดังนี้

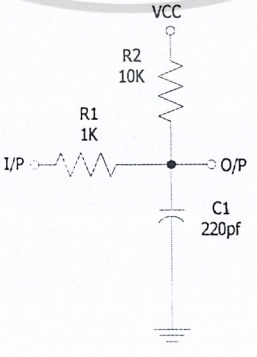
1. เครื่องคอมพิวเตอร์รุ่น Pentium 166 MMX ขึ้นไป
2. หน่วยความจำไม่ควรต่ำกว่า 32 MB
3. จอภาพแบบ Super VGA ขึ้นไป
4. คีย์บอร์ดและเมาส์
5. การ์ดแคปเจอร์ (Capture Card) เพื่อรับภาพจากกล้อง CCD

เนื่องจากโปรแกรมควบคุมจะทำงานภายใต้ระบบปฏิบัติการ WINDOWS 98 SE จึงต้องมีคุณสมบัติของเครื่องคอมพิวเตอร์ที่สามารถทำงานภายใต้ระบบปฏิบัติการโดยไม่มีปัญหา

6.3 วงจรอินเตอร์เฟส

เนื่องจากโครงงานนี้ใช้การอินเตอร์เฟสทางพอร์ทขนาน ซึ่งมีข้อดีคือไม่ต้องสร้างการ์ดอินเตอร์เฟสเพิ่มเติม ซึ่งเป็นวงจรที่เป็นตัวกลางในการติดต่อ-ควบคุมอุปกรณ์ต่างๆ กับเครื่องคอมพิวเตอร์ภายในโครงงาน ในการออกแบบได้สร้างวงจรอินเตอร์เฟสพอร์ทขนานซึ่งสามารถใช้งานพอร์ทดาต้า, พอร์ทสเตตัส และพอร์ทคอนโทรลได้ทั้งหมด โดยนำสายสัญญาณต่างๆซึ่งผ่านวงจรกรองสัญญาณรบกวนแล้วเข้าสู่วงจรต่างๆภายในโครงงาน

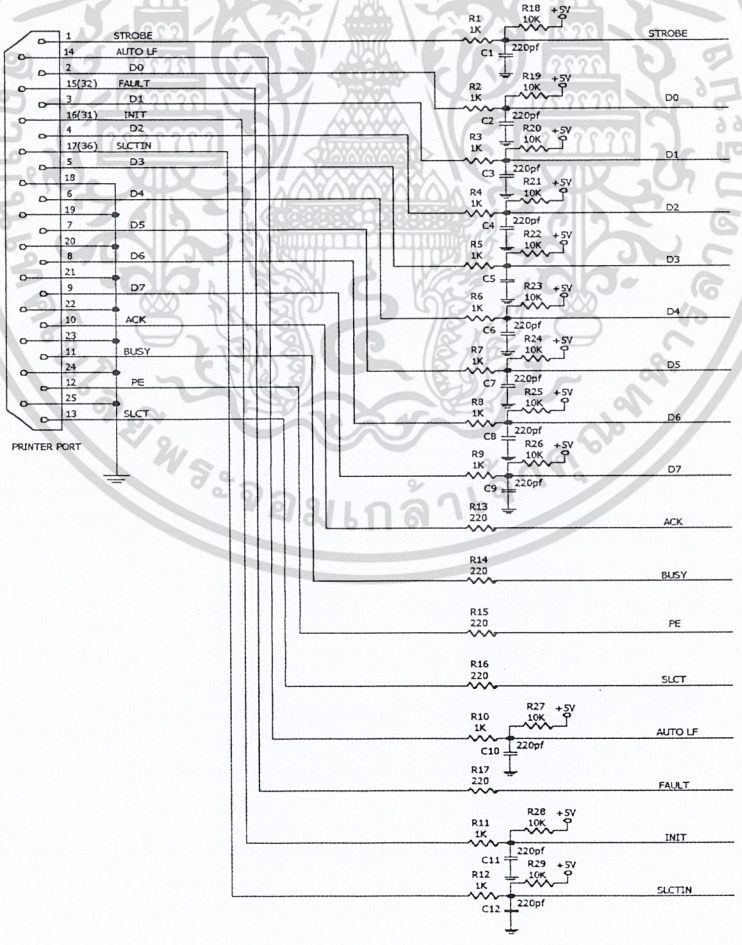
เนื่องจากสายเคเบิลของพรินเตอร์สามารถเกิดสัญญาณสอดแทรก(interference) ได้ง่าย ซึ่งสัญญาณรบกวนนี้อาจเกิดในรูปสัญญาณรูปเดือยแหลมหรือสไปค์(spike) เกิดเป็นแรงดันเกินสั้นๆ ซึ่งอาจทำให้วงจรทำงานผิดพลาดได้ สามารถแก้ปัญหานี้ได้โดยต่อวงจรกรองความถี่ต่ำผ่าน โดยมีตัวต้านทาน R_1 และ C_1 ซึ่งจะทำหน้าที่ชอร์ตสัญญาณสไปค์ที่เกิดจากความถี่สั้นๆลงกราวด์ไป ดังวงจรในรูปที่ 6.3



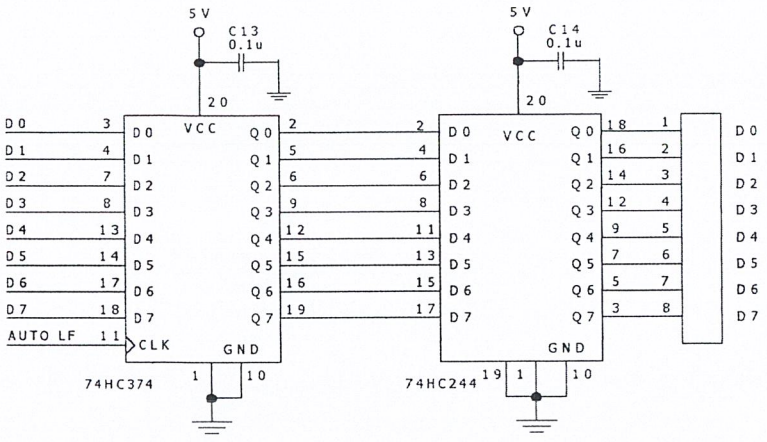
รูปที่ 6.3 แสดงวงจรกรองความถี่ต่ำผ่าน

ในการเขียนโปรแกรมต้องกำหนดกว้างของพัลซ์ต่ำสุด 1.25 ไมโครวินาที ถึงจะผ่านวงจรกรองความถี่ไปได้ และจากวงจรใช้ไอซีชนิดซีมอส ซึ่งไม่สามารถที่จะใช้ต่อโดยตรงกับเอาท์พุทของการ์ดเครื่องพิมพ์ที่เป็นชนิดทีทีแอลได้ ทั้งนี้ก็เพราะว่าระดับสัญญาณลอจิก "1" ของอุปกรณ์ทีที-แอลใช้แรงดันต่ำกว่าอุปกรณ์ที่ใช้ซีมอส ดังนั้น R2 ที่ต่อระหว่างขาสัญญาณที่ออกจากการ์ดเครื่องพิมพ์กับไฟเลี้ยง 5 โวลท์ เพื่อพูลอัพ(pull up) ระดับสัญญาณของทีทีแอลให้ตรงกับซีมอส

ทางด้านพอร์ทดาต้า จะมี IC_1 ซึ่งภายในจะมีดีฟลิบฟลอบ(D-flipflop) ทำงานเป็นวงจรป้องกัน ข้อมูลที่เข้ามายังฟลิบฟลอบคือข้อมูลที่เขียนโปรแกรมผ่านทางด้านเอาท์พุททั้ง 8 ของ พอร์ทดาต้า (D0-D7) และข้อมูลนี้จะถูกทำค้างสถานะ(latch) โดยสัญญาณ "AUTO LF" เอาท์พุทของ IC_1 ที่ถูกทำให้ค้างสถานะ จะถูกส่งเข้าไปยัง IC_2 ซึ่งเป็นไอซีที่ใช้ขับเอาท์พุท(Driver) ดังรูปที่ 6.4 และรูปที่ 6.5ในการออกแบบเลือกใช้ไอซีเบอร์ 74HC374 ทำหน้าที่ค้างสถานะข้อมูล โดยใช้สัญญาณนาฬิกา (CLK ขาที่ 11) จากขาสัญญาณ AUTO LF โดยในการส่งข้อมูลจะใช้โปรแกรมเป็นตัวควบคุม และใช้ไอซี 74HC244 เป็นบัฟเฟอร์(Buffer) ป้องกันวงจรจากอุปกรณ์ภายนอกอีกชั้นหนึ่ง ส่วน C ค่า 0.1uF จะป้องกันสัญญาณรบกวนจากแหล่งจ่ายไฟ



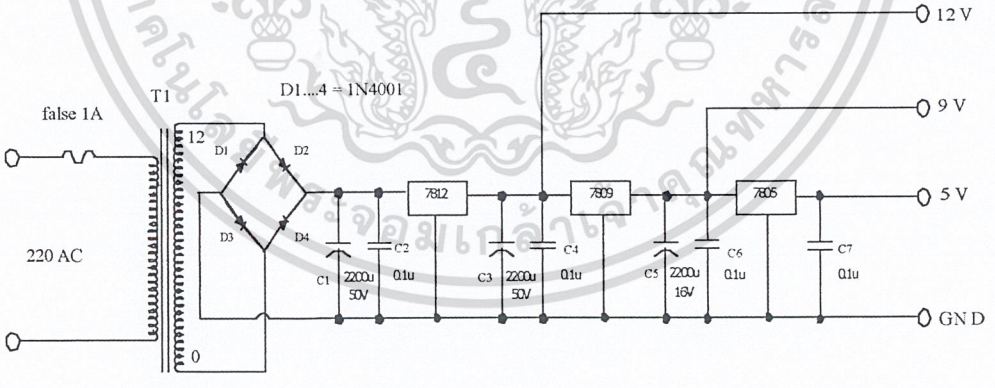
รูปที่ 6.4 แสดงการต่อวงจรกรองความถี่ต่ำผ่านเข้ากับพอร์ทขนาน



รูปที่ 6.5 แสดงวงจรดิจิตอลเอาท์พุท

6.4 แหล่งจ่ายพลังงาน

ภายในโครงงาน มีแหล่งจ่ายกำลังงานไฟฟ้ากระแสตรงเพื่อเลี้ยงวงจรในส่วนต่างๆ ถึง 3 ระดับ กล่าวคือ 5V ,9Vและ 12 V โดยที่ไฟเลี้ยง 5V จะจ่ายไปเลี้ยงไอซีต่างๆ ไฟเลี้ยง 9V จ่ายไปเลี้ยงวงจรตรวจจับการเคลื่อนไหว และ 12V เพื่อขับสเต็ปปิ้งมอเตอร์ ดังวงจรในรูปที่ 6.6



รูปที่ 6.6 แสดงวงจรจ่ายพลังงาน

6.5 กล้องวีดีโอ

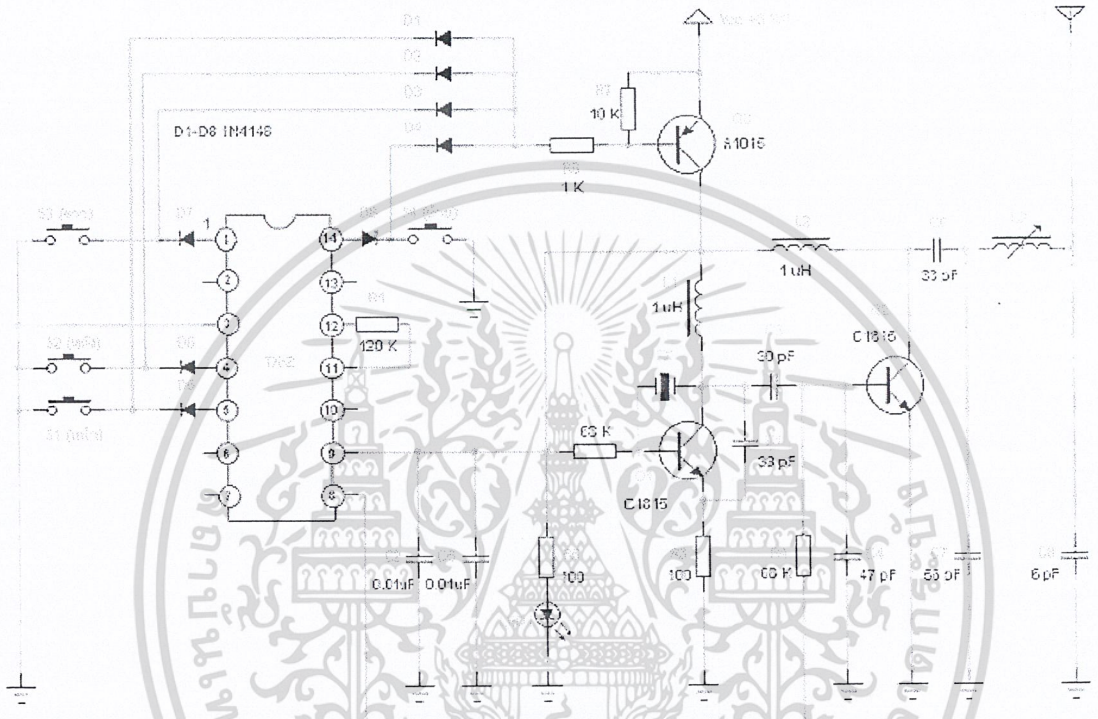
ในโครงงานนี้เลือกใช้กล้อง CCD ซึ่งเป็นแบบขาว-ดำ ที่มีขนาดเล็กและน้ำหนักเบา โดยมีคุณสมบัติดังนี้

- กล้อง CCD แบบอินฟาเรด แสดงผลระบบขาว-ดำ
- ทำงานที่ระดับแรงดัน 12 โวลท์ดีซี
- มีความละเอียดของสัญญาณแบบเดียวกับโทรทัศน์ คือเส้นสแกนทางแนวนอน 380 เส้น และเส้นสแกนทางแนวตั้ง 450 เส้น

สำหรับการรับภาพจากกล้อง CCD จะใช้การ์ดอินเตอร์เฟส TV รุ่น Flyvideo ของบริษัท Lifeview ซึ่งการ์ด Capture ตัวนี้จะทำหน้าที่แปลงสัญญาณภาพที่รับเข้ามาจากกล้อง ให้เป็นสัญญาณดิจิตอลเพื่อให้คอมพิวเตอร์สามารถแสดงและประมวลผลภาพได้

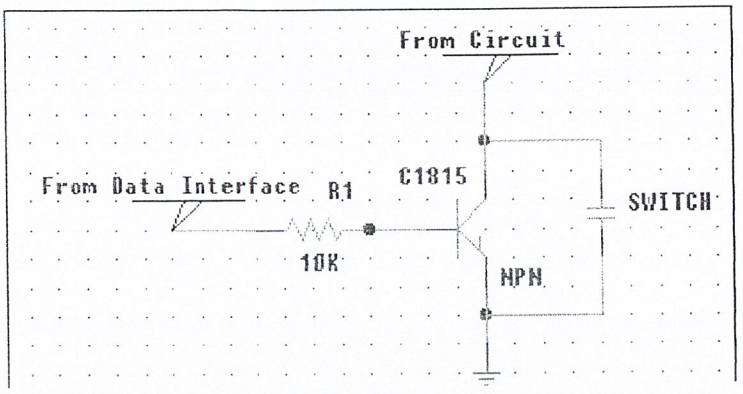
6.6 การออกแบบชุดเข้ารหัส(Encoder)

เนื่องด้วยโครงงานนี้ใช้การส่งข้อมูลระยะไกลแต่ชุดส่งข้อมูลระยะไกลนั้นทางผู้จัดทำได้ทดลองสร้างขึ้นมาหลายชุดแต่ไม่ประสบผลสำเร็จจึงจำเป็นต้องใช้ชุดรับส่งสำเร็จของรถบังคับวิทยุ (รถกระป๋อง) โดยนำชุดรับส่งนี้มาทำการแก้ไขดัดแปลงให้สามารถทำการรับส่งข้อมูลที่ต้องการได้ ชุดส่งของรถบังคับนั้นมีวงจรดังรูปที่ 6.7



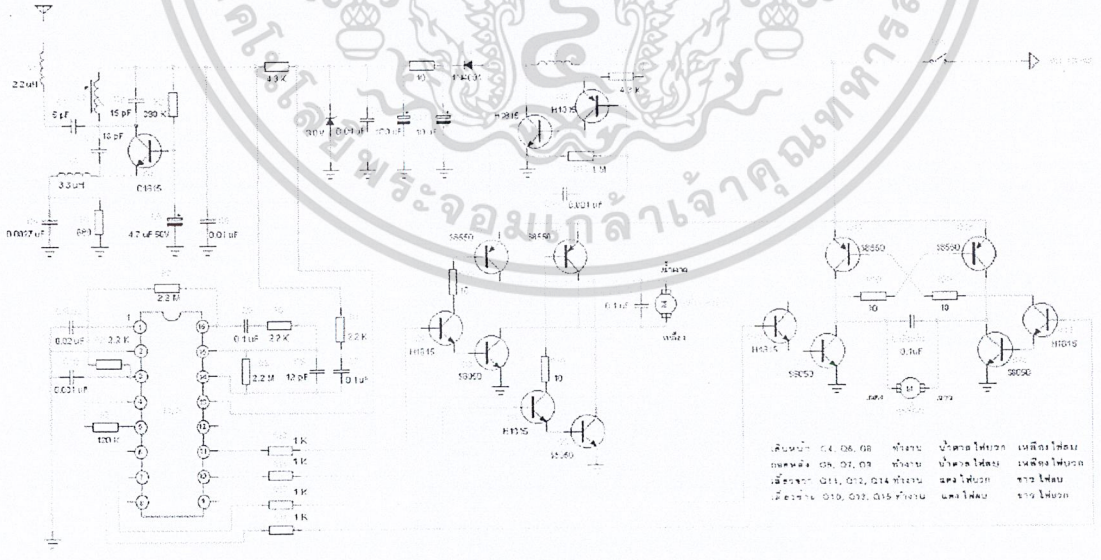
รูปที่ 6.7 วงจรภาคส่งของรถบังคับวิทยุ

จากรูปที่ 6.7 จะเห็นได้ว่าเราสามารถแก้ไขวงจรบางตำแหน่งเพื่อที่จะได้ส่งรหัสของการบังคับได้ตามที่เราต้องการ จากวงจรเราเห็นได้ว่าเมื่อเรากดสวิตช์จะทำให้ขาของไอซีที่ต่ออยู่กับสวิตช์ต่อลงกราวด์ก็เท่ากับว่าเราไบอัสสถานะศูนย์ให้กับวงจรส่ง ด้วยเหตุนี้จึงทำการต่อเชื่อมต่อสวิตช์ด้วยชุดอินเทอร์เฟส โดยนำทรานซิสเตอร์มาต่อรวมด้วยเราก็จะได้วงจรภาคส่งแบบไร้สายโดยควบคุมด้วยคอมพิวเตอร์ ดังรูป 6.8



รูปที่ 6.8 แสดงการต่อชุดวงจรทรานซิสเตอร์

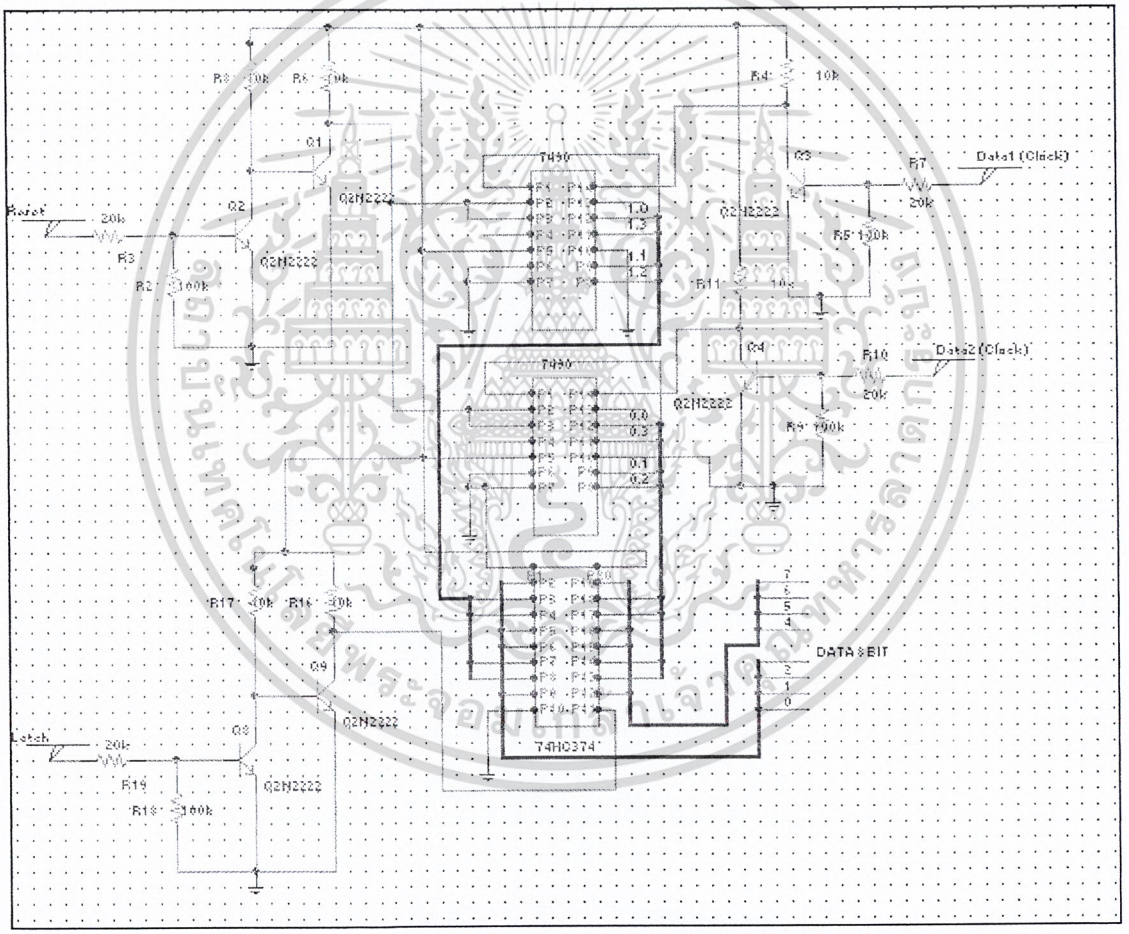
จากรูปเราเห็นได้ว่าเมื่อชุดอินเทอร์เฟสของคอมพิวเตอร์มีสัญญาณออกมาเป็น “1” จะทำให้ทรานซิสเตอร์ทำงาน เทียบได้กับการกดสวิตช์ แต่เมื่อสัญญาณเป็น “0” ทรานซิสเตอร์จะไม่ทำงานดังนั้นเราจึงใช้วงจรนี้ต่อกับสวิตช์ทั้ง4ตัว เพื่อที่จะได้เขียนโปรแกรมในการเข้ารหัสซึ่งใช้ในการส่งสัญญาณต่อไป



รูปที่ 6.9 วงจรภาครับของรถบังคับวิทยุ

6.7 การออกแบบชุดถอดรหัส(Decoder)

ในการรับสัญญาณที่ได้รับมาจากวงจรชุดส่ง เราต้องทำการสร้างวงจรเพื่อถอดรหัสของข้อมูลที่ส่งมา ซึ่งรหัสที่ส่งมานั้นเป็นขบวนพัลส์โดยถูกกำหนดโดยโปรแกรมคอมพิวเตอร์ จากรูปที่ 6.7 เมื่อสั่งให้โปรแกรมส่งข้อมูลที่มีสถานะเป็น “1” ออกสู่พอร์ตปรินท์เตอร์(Printer) โดยสมมุติว่าส่งออกไปยังตัวทรานซิสเตอร์ที่ต่อขนานอยู่กับสวิตช์ตัวที่3 ซึ่งเป็นการสั่งให้รถเลี้ยวขวา ในขณะเดียวกันกับที่ภาครับ ขา 6 ของไอซีอาเอ็ก 2 (RX-2) จะมีสัญญาณ “1” ออกมาเช่นเดียวกันดังนั้นเราจึงได้สัญญาณออกไปใช้ในวงจรถอดรหัสต่อไป

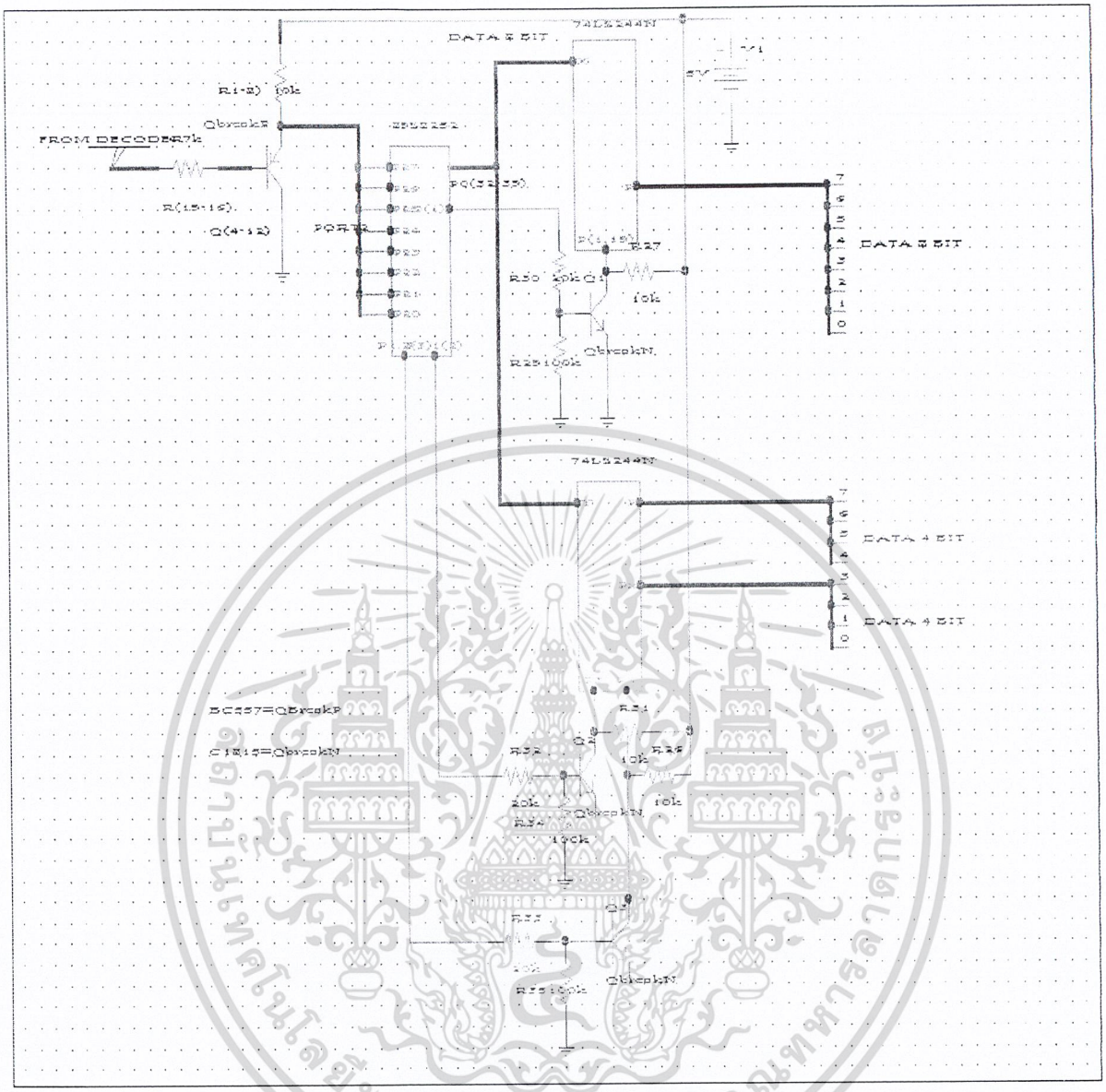


รูปที่ 6.10 วงจรถอดรหัส

COUNT	OUTPUT			
	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

ตารางที่ 6.1 ตรรกของไอซี 7490

จากวงจรรูปที่ 6.10 เมื่อเราป้อนสัญญาณพัลส์เข้าที่ขา 14 ของไอซี 7490 ไอซีตัวนี้จะทำการนับตามจำนวนของพัลส์ที่ป้อนเข้ามาดังตารางที่ 6.1 สมมุติว่าเราป้อนพัลส์เข้ามา 2 พัลส์ ก็จะทำให้ไอซี 7490 มีค่าที่ขาเอ้าท์ เป็น 0001 ซึ่งเป็นเลขฐานสองสี่บิต เราสามารถนำค้าต้าที่ได้ไปใช้ในการควบคุมอุปกรณ์ต่างๆตามที่เราต้องการได้ จากรูปที่ 6.10 ไอซี 74HC374 ใช้ทำหน้าที่เป็นตัวแลท (Latch) และบัฟเฟอร์(Buffer)



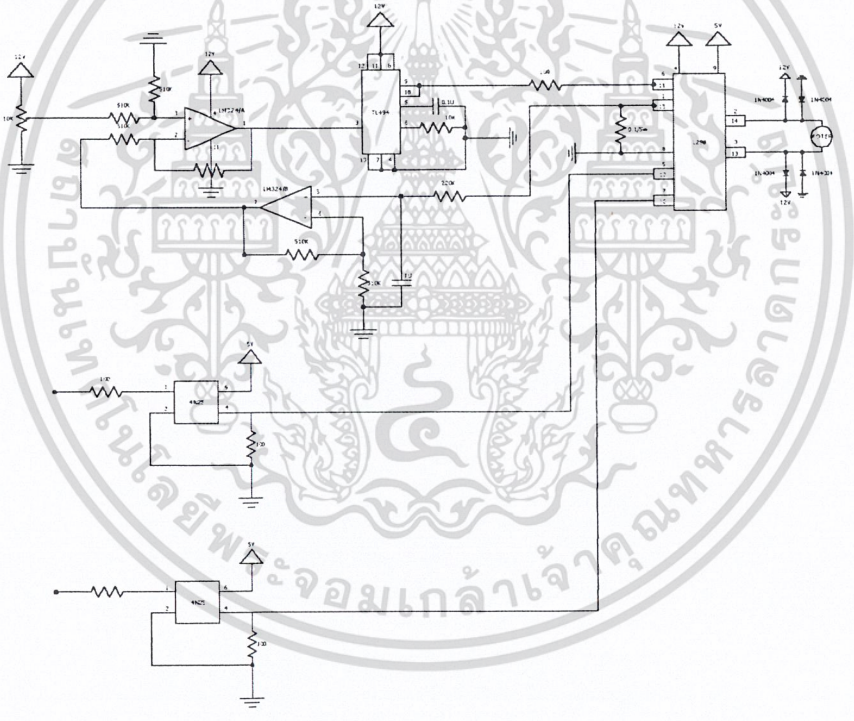
รูปที่ 6.11 วงจรภาคบับเฟอร์

จากวงจรดังรูปที่ 6.11 เป็นวงจรบัฟเฟอร์(Buffer) ซึ่งประกอบด้วยไมโครคอนโทรลเลอร์เบอร์ 89S8252 ทำหน้าที่เป็นตัวจัดส่งแยกข้อมูลให้กับช่องข้อมูลต่างๆทั้ง 3 ช่องตามที่เราเขียนคำสั่งโปรแกรมให้กับตัวไมโครคอนโทรลเลอร์ โดยการเขียนโปรแกรมภาษาแอสเซมบลี ในลำดับท้ายสุดเราก็จะได้ข้อมูลออกมาตามที่ต้องการ

6.8 การออกแบบส่วนไดร์มอเตอร์

เนื่องจากในโครงงานเลือกใช้มอเตอร์กระแสตรงในการขับเคลื่อน ทำให้ใช้กระแสมากโดยเฉพาะในช่วงเริ่มต้นของการทำงาน เพราะฉะนั้นจึงต้องมีวงจรที่ทำหน้าที่จ่ายกระแสให้กับมอเตอร์ โดยแต่ละส่วนจะแยกการทำงานแต่ละตัว และในการเคลื่อนที่ของมอเตอร์ในช่วงที่ใช้กระแสต่างกันจะมีความเร็วในการเคลื่อนที่ต่างกัน ทำให้ต้องใช้พัลวิดท์มอดดูเลชั่นแอมปลิไฟล์เข้ามาช่วยในการไดร์มอเตอร์ ในการออกแบบได้แยกการทำงานออกดังนี้

6.8.1 วงจรควบคุมทิศทางการหมุนและจ่ายกำลังมอเตอร์



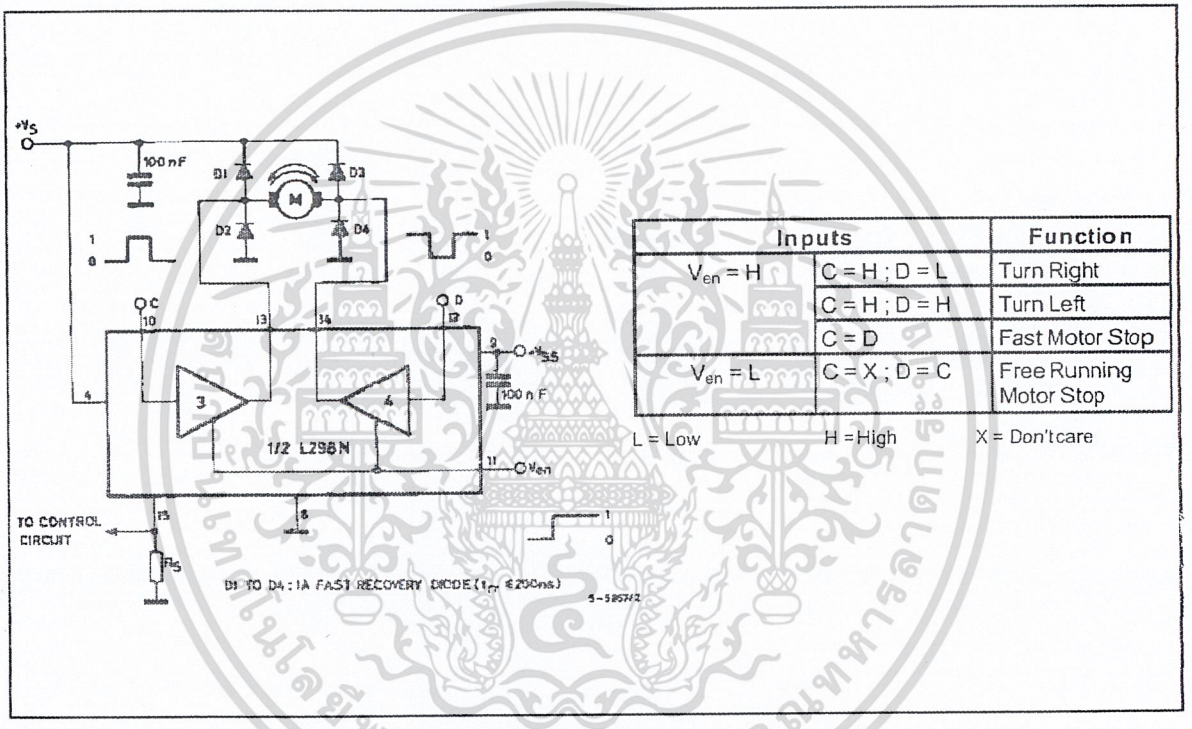
รูปที่ 6.12 วงจรควบคุมทิศทางการหมุนและจ่ายกำลังมอเตอร์

หน้าที่ของอุปกรณ์แต่ละตัว

- D_1 ,D_2 ,D_3 ,D_4 ใช้สำหรับป้องกันคอลเล็คเตอร์-อิมิตเตอร์จังค์ชั่นของเพาเวอร์ทรานซิสเตอร์ภายใน

ตัว IC เบอร์ L298 จาก inductive kicks และสวิทชิ่งสไปค์

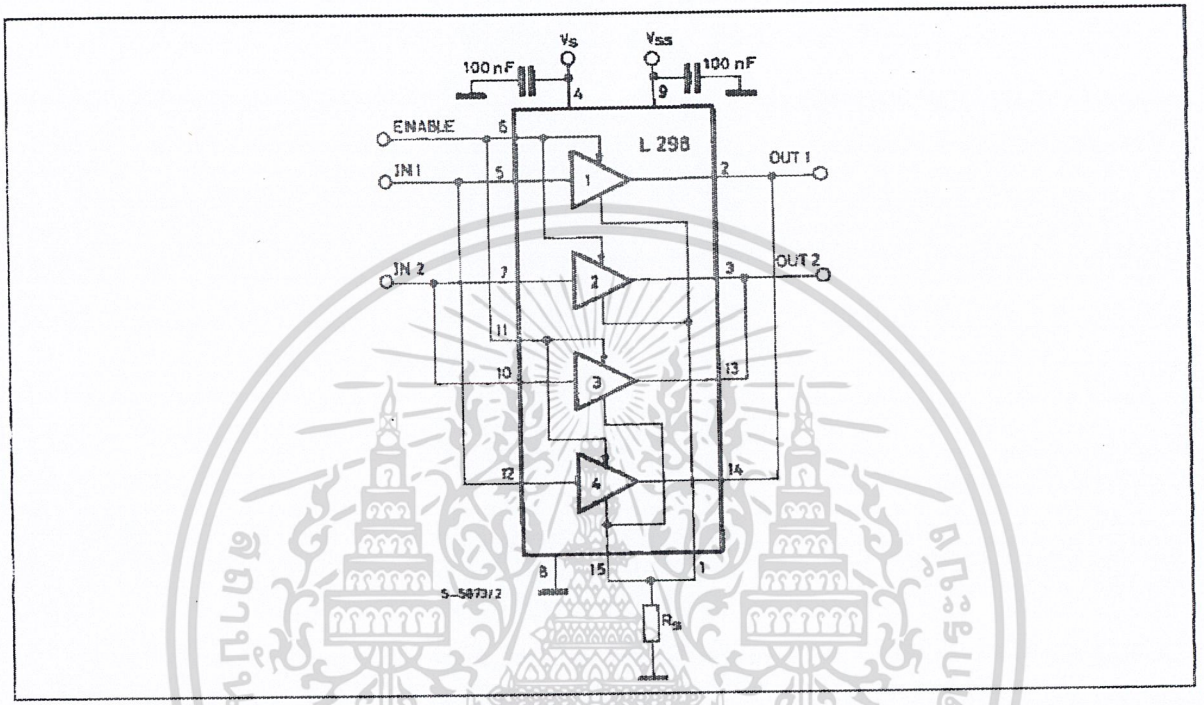
- R_{sense} ใช้สำหรับดีเท็คความเร็วของมอเตอร์



รูปที่ 6.13 Bidirectional DC Motor Control

จากรูปที่ 6.13 จะแสดงการทำงานของ IC เบอร์ L298 เมื่อ Ven มีสถานะลอจิก High คือมีพัลซ์บวกมากระตุ้นการทำงานขา C ลักษณะลอจิก High ขา D มีสถานะลอจิก Low ทำให้กระแสไหลจากขา 13 ไปยัง dc motor ทำให้มอเตอร์หมุนไปทางขวา ในทางกลับกันถ้าขา C มีสถานะลอจิก Low และขา D มีลักษณะลอจิก High กระแสก็จะไหลสวนทางกับครั้งแรกทำให้มอเตอร์ไหลกลับทิศทาง โดยจะหมุนไปทางซ้าย แต่ถ้าหากระดับลอจิกที่ขา C และ D มีระดับเดียวกัน จะทำให้มอเตอร์หยุดหมุนในทันที

พิจารณาที่ Ven มีสถานะลอจิก Low ถ้าขา C เป็น Low หรือ High ขา D มีสถานะเหมือนกับ C มอเตอร์จะมีสถานะ หยุดหมุนแบบ Free Running



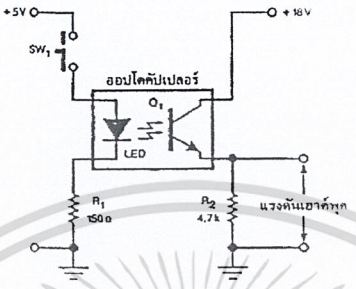
รูปที่ 6.14 แสดงการต่อเอาท์พุทขนานในการใช้งานกระแสสูง

จากรูปแสดงการนำ IC L298มาต่อขนานกัน โดยขา enable ขา 6 ต่อกับขา 11 ,ขาอินพุท1 ขา 5 ต่อกับขา 12 ,ขาอินพุท 2 ขา 7 ต่อกับขา 10 ,ขาเอาท์พุท 1 ขา 2 ต่อกับขา 14 ,ขาเอาท์พุท 3 ขา ต่อกับขา 12 โดยจะสังเกตุเห็นว่าเป็นการต่อขนานทำให้กระที่สามารถจ่ายให้โหลดนั้นมีค่าเพิ่มมากขึ้น ในการทำโปรเจคครั้งนี้ก็ใช้การต่อแบบขนาน เพราะมอเตอร์ต้องจ่ายกระแสมากในการทำงาน

6.8.2 วงจรคัปปลิ้ง

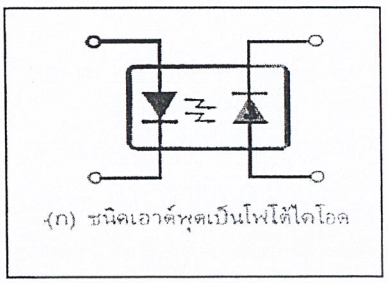
สำหรับการ์ดอินเตอร์เฟสที่ใช้ในการเชื่อมต่อระหว่างคอมพิวเตอร์และวงจรไดร์มอเตอร์นี้ ได้นำออปโต้คัปเปลอร์เข้ามาใช้ เพื่อช่วยป้องกันการลัดวงจรจากภายนอกที่อาจส่งผลกระทบต่อคอมพิวเตอร์ ได้ซึ่งชนิดและหลักการทำงานของออปโต้คัปเปลอร์ (Opto-Coupler) คือออปโต้คัปเปลอร์เป็นอุปกรณ์ที่ประกอบด้วย LED ซึ่งปกติเป็นชนิดอินฟาเรด และโฟโต้ทรานซิสเตอร์หรือโฟโต้ได้โอดซึ่งถูกผลิตมาคู่กันรวมอยู่ในตัวเดียวกันจากรูปเป็นวงจรพื้นฐานของออปโต้คัปเปลอร์

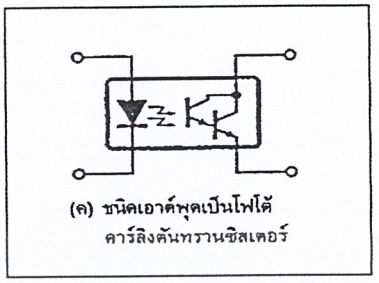
โดยด้าน LED จะเป็นอินพุทของวงจรและด้านโฟโต้ทรานซิสเตอร์ เป็นเอาท์พุทของวงจร ดังรูปที่ 6.15

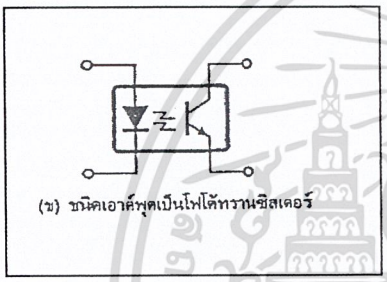


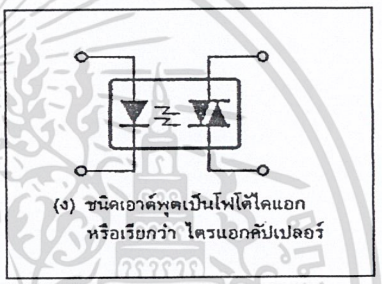
รูปที่ 6.15 วงจรออปโตคัปเปิล

จากรูปเมื่อสวิตซ์ SW1 เปิดวงจร LED และทรานซิสเตอร์ Q1 จะยังไม่ทำงาน แต่เมื่อสวิตซ์ SW1 ปิดวงจรจะมีกระแสไหลผ่าน LED และตัวต้านทาน R2 ซึ่งจะทำให้ Q1 ทำงานด้วยและจะได้แรงดันเอาท์พุทตกคร่อมที่ตัวต้านทาน R2 จะเห็นได้ว่าเอาท์พุทของวงจรถูกควบคุมโดยส่วนอินพุท แต่ทั้งสองส่วนถูกแยกออกจากกันทางไฟฟ้าอย่างสิ้นเชิง (ซึ่งเป็นหลักการสำคัญของออปโต้คัปเปลอร์) วงจรนี้สามารถนำไปประยุกต์ใช้ได้ทั้งสัญญาณดิจิตอลและสัญญาณอะนาลอกการทำลายกราวด์ลูประหว่างวงจรสองวงจรอาจทำได้โดยใช้ออปโต้คัปเปลอร์โดย LED จะคัปปลิ้งผ่านแสงไปยังโฟโต้ทรานซิสเตอร์ไดโอด หรือไทริสเตอร์ซึ่งอยู่ภายในภาชนะเดียวกันทำให้กราวด์ลูปถูกตัดออกจากกันโดยสิ้นเชิง เนื่องจากการคัปปลิ้งระหว่างวงจรทั้งสองจะเกิดขึ้นโดยมีแสงเป็นตัวกลางเท่านั้นออปโต้คัปเปลอร์มีหลายแบบต่างกันตามชนิดเอาท์พุท ดังแสดงในรูป 6.16

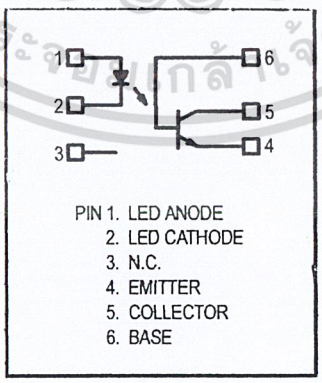








รูปที่ 6.16 แสดงสัญลักษณ์ของออปโต้คัปเปลอร์แบบต่าง ๆ



รูปที่ 6.17 แสดงวงจรภายในและขาของ IC 4N25

-IC 4N25 ภายในเป็นวงจรออปโต้คัปเปลอร์ ทำหน้าที่คัปปลิ้งสัญญาณควบคุมจากคอมพิวเตอร์กับวงจรจ่ายกำลัง โดยขา 1 เป็นขา LED อาโนด ,ขา 2 เป็น LED คาโถด,ขา 3 ไม่ต่อ,ขา 4 เป็นขาอิมิเตอร์,ขา 5 เป็นขาคอลเลคเตอร์,ขา 6 เป็นขาเบส ดังรูปที่ 6.17

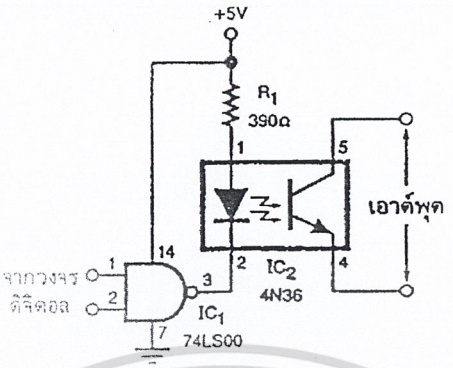
6.8.2.1 อัตราการถ่ายทอด

อัตราการถ่ายทอดกระแสไฟฟ้า(Current Transfer Ratio : CTR)เป็นพารามิเตอร์ที่วัดถึงปริมาณกระแสไฟฟ้าที่ถูกถ่ายทอดจากส่วนอินพุทของวงจรซึ่งใช้ LED อินฟาเรด ไปยังส่วนเอาท์พุทของเป็นอุปกรณ์ใด ๆ โดยคิดในหน่วยเปอร์เซ็นต์ของการแยกกันทางไฟฟ้าอย่างสมบูรณ์ ซึ่งสามารถติดออกมา เป็นอัตราการขยายกระแสหรืออัตราการสูญเสียทางไฟฟ้าก็ได้ เมื่อเทียบแล้วทรานซิสเตอร์ก็คือ การคิดหาอัตราการขยายกระแส หรือ h_{FE} นั่นเอง ซึ่งเปรียบเทียบกระแสอินพุทคือ I_B และกระแสเอาท์พุทคือ I_C แต่ I_B ในที่นี้คือกระแสไบอัสตรงของ LED อินฟราเรดหรือ IF ในรูปเป็นวงจรออปโต้คัปเปลอร์ที่ส่วนเอาท์พุทเป็นโฟโต้ทรานซิสเตอร์

ค่า CTR ถ้าเป็น 100% หมายความว่ากระแสอินพุทเท่ากับกระแสเอาท์พุทพอดี หากต่ำกว่า 100% หมายความว่าเกิดความสูญเสียขึ้นภายในตัวออปโต้คัปเปลอร์ ถ้าเกิน 100% แสดงว่าออปโต้คัปเปลอร์ตัวนั้นมีความหมายในการขยายกระแสไฟฟ้า

6.8.2.2 การใช้ลอจิกเกตในการควบคุมออปโต้คัปเปลอร์

การนำไอซีลอจิกเกตมาเชื่อมกับออปโต้คัปเปลอร์ทำได้โดยง่ายโดยต่อเข้ากับทางภาคอินพุท เพื่อควบคุมการทำงานของไดโอดอินฟาเรดภายในออปโต้คัปเปลอร์ ทั้งนี้เนื่องจากแรงดันไฟฟ้าเลี้ยงไอซีลอจิกเกตและวงจรดิจิตอลโดยส่วนใหญ่จะมีค่า 5V ซึ่งไม่สูงนัก ทำให้ใช้เป็นไฟเลี้ยงภาคอินพุทของออปโต้คัปเปลอร์ร่วมกันได้ เพียงต่อตัวต้านทานอนุกรมกับไดโอดอินฟราเรด เพื่อจำกัดกระแสเท่านั้น ซึ่งแสดงดังรูป

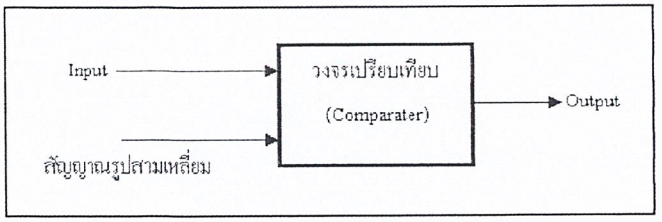


รูปที่ 6.18 การใช้ไอซีลอจิกเกตตระกูล TTL เพื่อควบคุมออปโต้คัปเปลอร์

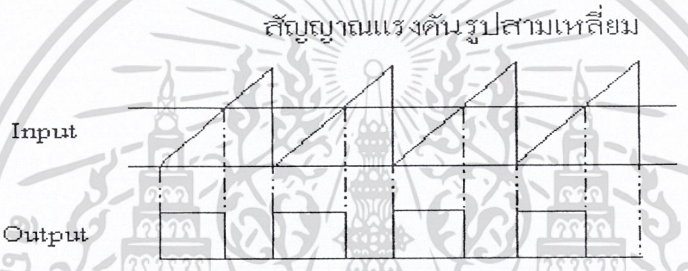
ลอจิกเกตที่ใช้เป็นไอซีตระกูล TTL จะมีความสามารถในการจ่ายกระแสสูงพอสมควร แต่ถ้าเป็นลอจิกตระกูล CMOS ลอจิกเกตตระกูลนี้จะมีความสามารถในการจ่ายกระแสเอาท์พุทต่ำกว่าตระกูล TTL จึงจำเป็นต้องต่อวงจรขับกระแสเอาท์พุทเพิ่มเติม

6.8.3 วงจรพัลส์วิธมอดูเลชั่น (Pulse Widih Modulation)

เป็นวงจรควบคุมความกว้างของพัลส์ โดยการเปรียบเทียบค่าแรงดันระหว่างแรงดันอินพุทที่เข้ามากับแรงดันของสัญญาณสามเหลี่ยมที่มีขนาดและความถี่คงที่ ถ้าแรงดันอินพุทมีค่ามากกว่าแรงดันสามเหลี่ยมจะทำให้แรงดันเอาท์พุทมีค่าเป็น “1” และถ้าแรงดันอินพุทมีค่าน้อยแรงดันสามเหลี่ยมจะทำให้แรงดันเอาท์พุมมีค่าเป็น ”0” ดังแสดงในรูป



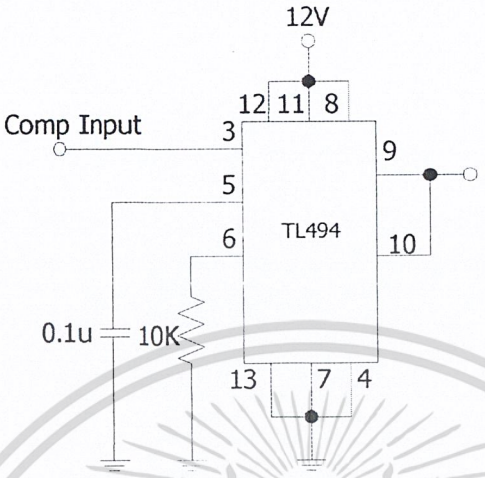
ก). บล็อกไดอะแกรม



ข). ลักษณะของรูปสัญญาณ

รูปที่ 6.19 หลักการของวงจรพัลส์วิธมอดูเลชั่น

เรามักใช้ความถี่ของพัลซ์ในระบบ PWM ที่มีค่าสูงกว่า 1 KHz (มักมีค่าถึง 10 KHz) และความถี่นี้ขึ้นอยู่กับความต้องการของผลตอบสนองของระบบต่อแบนวิดท์ อินดัคแต้นซ์ของมอเตอร์และคุณสมบัติการสูญเสียเพาเวอร์ในตัวมอเตอร์ ในเวลาเดียวกันเราต้องคำนึงถึง Adio noise ที่เกิดจากขดลวด ฮีทซิงค์ และส่วนของโครงสร้างประกอบของตัวมอเตอร์ที่มีเสียงดังออกมา เราสามารถประยุกต์ใช้งานของระบบ PWM โดยเพิ่มความถี่ของพัลซ์ให้ถึงจุดที่หูของมนุษย์ไม่ได้ยินเสียงได้ ในส่วนของวงจรเราได้ใช้ไอซีเบอร์ TL 494 ทำหน้าที่เป็นวงจรควบคุมความกว้างของพัลส์ โดยกำหนดความถี่ในการออสซิสเลทโดยใช้ค่าของ R_T และ C_T ดังรูปที่ 6.20

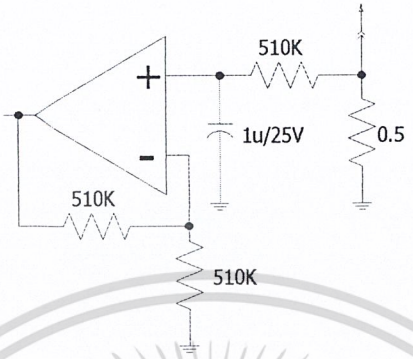


รูปที่ 6.20 แสดงวงจรพัลส์วิธมอดูเลชั่น

ซึ่งจากวงจรเราใช้ค่า $R_T = 10k\Omega$ และ $C_T = 0.1uF$ ดังนั้นจะได้ค่าความถี่ออสซิลเลทประมาณ 1.1kHz ซึ่งความถี่ออสซิเลทนี้หาได้จาก $f_{osc}=1.1/(R_T.C_T)$ ความกว้างของพัลส์จะเปลี่ยนไปตามค่าแรงดันอินพุทที่เข้ามาซึ่งความกว้างของพัลส์จะมีผลกับความเร็วมอเตอร์คือ เมื่อพัลส์มีความกว้างมากจะทำให้ความเร็วในการหมุนของมอเตอร์เร็วขึ้นในทางตรงกันข้ามถ้าความกว้างของพัลส์น้อยจะทำให้ความเร็วในการหมุนของมอเตอร์ช้าลงเอาท์พุทที่ได้ก็จะนำไปผ่านวงจรควบคุมทิศทางต่อไป

6.8.4 วงจรวัดความเร็วของมอเตอร์

มอเตอร์จะดึงกระแสจากวงจรไดว์เวอร์เป็นสัดส่วนกับความเร็วของมอเตอร์ กระแสนี้จะรับรู้ได้ด้วยตัวความต้านทาน Rsense ต่อกับจุดร่วมของขา 1 และขา 15 ซึ่งเป็นขาร่วมอิมิเตอร์ภายใน IC L298 กับกราวด์ ดังรูป 6.21



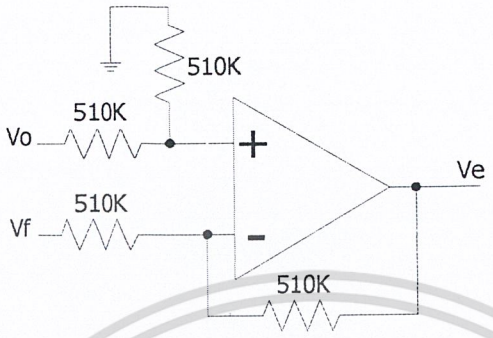
Rsense=0.5 โอห์ม

รูปที่ 6.21 วงจรนอนอินเวอร์ทแอมปลิไฟเออร์

จากรูป วงจรนอนอินเวอร์ทแอมปลิไฟเออร์ที่ใช้ขยายโวลท์เตจตกคร่อม ความต้านทาน Rsense อัตราขยายของนอนอินเวอร์ทเตอร์ขึ้นอยู่กับค่ากระแสของมอเตอร์ ซึ่งเป็นตัวแปรของแรงดันอินพุทของวงจร non-inverting amp นั่นเอง โดยค่าอัตราการขยายนั้นคำนวณได้จากสูตร Av=1+(Rf/Ri) โดยในที่นี้ Rf=510K และ Ri=510K เพราะฉนั้นอัตราการขยายเท่ากับ 2 จะได้ Vo=2Vin

6.8.5 วงจรดีเท็คเออร์เรอร์ (ดิฟเฟอเรนเชียล แอมปลิไฟเออร์)

วงจรดีเท็คเออร์เรอร์จะเปรียบเทียบโวลท์เตจจากแรงดันอ้างอิงในที่นี้คือ R ปรับค่าที่ต่อจากแหล่งจาก 12 โวลท์นั่นเอง โดยดูได้จากวงจรไดรมอเตอร์ ซึ่งจะเปรียบเทียบกับโวลท์เตจทางด้านเอาท์พุทของวงจรแปลงความเร็วเป็นโวลท์เตจเพื่อให้ได้สัญญาณเออร์เรอร์ โดยผ่านวงจรดิฟเฟอเรนเชียลแอมปลิไฟเออร์ดังรูป 6.22



รูปที่ 6.22 วงจรดิฟเฟอเรนเชียล แอมปลิไฟเออร์

เอาท์พุทโวลท์เตจ V_e ของวงจรคือ Ve=Vo-Vf

เมื่อ Vo= แรงดันอ้างอิง

Vf= โวลท์เตจป้อนกลับที่เป็นสัดส่วนกับความเร็วมอเตอร์

Ve= เออร์เรอร์โวลท์เตจ

สำหรับอัตราการขยายของวงจรดิฟเฟอเรนเชียลนั้น ได้จาก

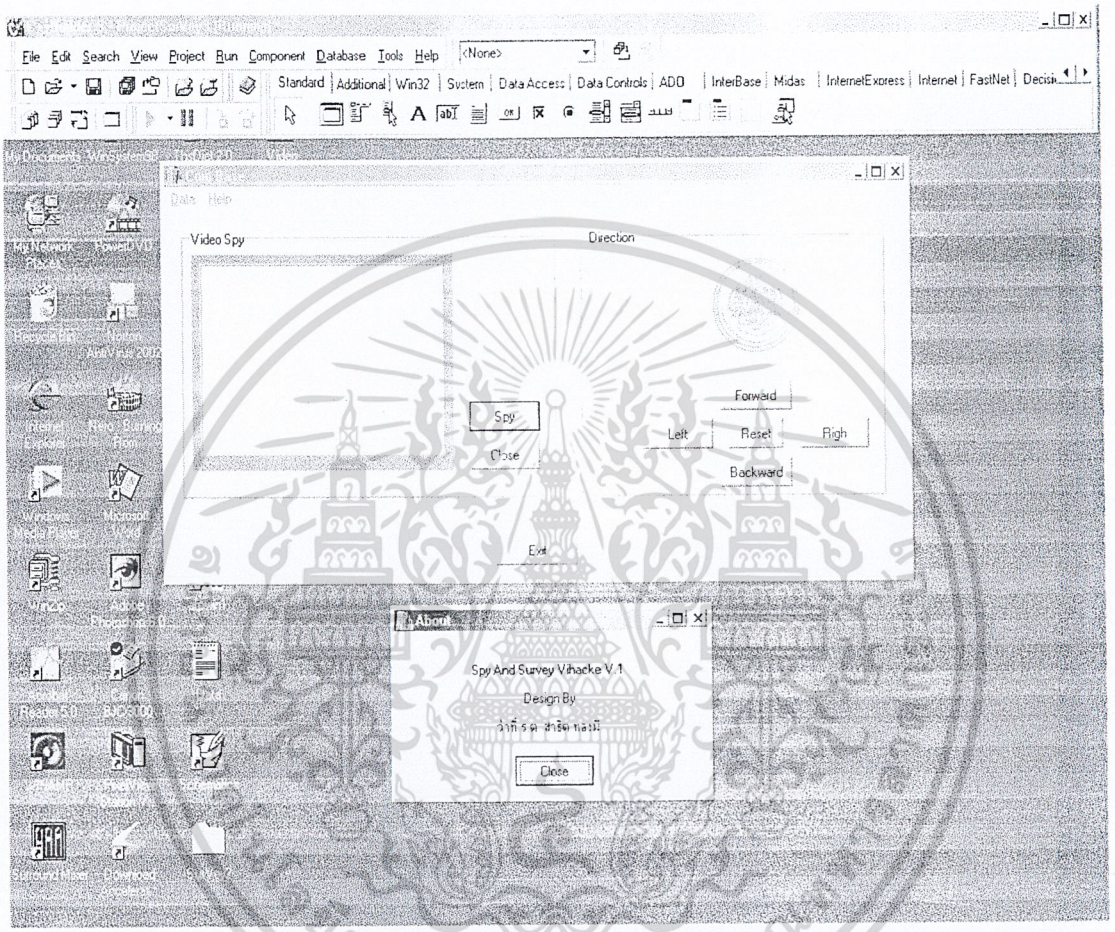
Ve=[R3/(R1+R3)][(R2+R4)/R2]Vo-[(R4/R2)]Vf

แต่ในวงจรนี้ค่า R1=R2=R3=R4=R

เพราะฉนั้น Ve=(1/2)(2).Vo-Vf=Vo-Vf

6.9 การเขียนโปรแกรมควบคุมรถ

โปรแกรมควบคุมรถ(Program Survey Vehicle Control : PSVC) ในการเขียนโปรแกรมควบคุมรถสำรวจเราใช้แอปพลิเคชั่นโปรแกรมชื่อโปรแกม “ซีพัสพัสบิวเดอร์ (C++ Builder)” เป็นตัวเขียนโปรแกรมควบคุมโดยมีลักษณะดังรูปที่ 1



รูปที่ 6.23 แสดงโปรแกรมควบคุมรถสำรวจ

จากรูปที่ 6.33 โปรแกรม PSVC ประกอบด้วย 3 ส่วนคือ

1. **ส่วนสัญญาณภาพ**

ประกอบด้วย

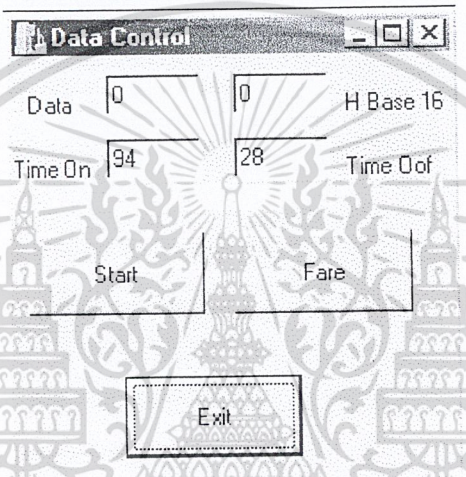
Spy คือ ปุ่มควบคุมการเปิดสัญญาณภาพ

Close คือ ปุ่มควบคุมการปิดสัญญาณภาพ

2. ส่วนควบคุมทิศทาง

ประกอบด้วย

Forward	คือ ปุ่มบังคับให้รถเดินหน้า
Backward	คือ ปุ่มบังคับให้รถถอยหลัง
Left	คือ ปุ่มบังคับให้รถเลี้ยวซ้าย
Right	คือ ปุ่มบังคับให้รถเลี้ยวขวา
Reset	คือ ปุ่มบังคับให้รถหยุด



รูปที่ 6.24 แสดงส่วนของดาต้าคอนโทรล

3. ส่วนควบคุมสัญญาณของพัลส์ที่ใช้ในการส่ง

ประกอบด้วย

Data คือ ข้อมูลที่เราต้องการให้ส่งไปยังภาครับเป็นเลขฐาน 16 ตั้งแต่ 0-9 H

Time on คือ คาบเวลาก่อนที่สัญญาณจะมีค่าเป็น “1” (mS)

Time oof คือ คาบเวลาก่อนที่สัญญาณจะมีค่าเป็น “0” (mS)

ในการกำหนดค่า Time on หรือ Time oof นั้นถ้าเรากำหนดให้มีค่าให้มีค่าน้อยๆ จะทำให้อัตราการส่งข้อมูลสามารถทำการส่งได้เร็ว แต่ถ้ากำหนดให้เร็วจนเกินไปจะทำให้การรับส่งข้อมูลผิดพลาดได้โดยวงจรดีโคดเดอร์จะไม่สามารถรับรู้สถานะของพัลส์ข้อมูลได้

บทสรุปและวิจารณ์

จากการทำโครงงานที่ผ่านมาได้ผลการทดลองเป็นที่น่าพอใจ กล่าวคือสามารถความคุมการเคลื่อนที่ของรถลาดตระเวนโดยการส่งข้อมูลขนาด 8 บิท ให้เป็นตามการบังคับควบคุมของคอมพิวเตอร์อีกทั้งยังสามารถส่งภาพมายังคอมพิวเตอร์ได้

ส่วนปัญหาที่พบในการทำงานคือ

1. เนื่องจากใช้มอเตอร์ในการขับเครื่อนสองตัวทำให้การความคุมให้มอเตอร์ทั้งสองทำงานที่ความเร็วเท่าๆกันทำได้ยาก ต้องพยายามในการปรับความเร็วของมอเตอร์ทั้งสองตัวให้เร็วเท่าๆกัน

2. เนื่องจากอัตรากำลังส่งของเครื่องควบคุมมีกำลังต่ำ จึงไม่สามารถส่งได้ไม่ไกล และส่งที่ความถี่ต่ำจึงถูกรบกวนได้ง่าย

3. น้ำหนักของตัวรถทั้งหมดมีน้ำหนักมากจึงทำให้การเคลื่อนที่ช้า และยังทำให้เวลาในการทำงานต่ำ

4. เนื่องจากวงจรทั้งหมดมีขนาดใหญ่ ทำให้ต้องสิ้นเปลืองพลังงานเป็นจำนวนมาก ระยะเวลาในการทำงานจึงต่ำ

5. การส่งขอ้มูลจากคอมพิวเตอร์ทำได้ช้าจึงทำให้การความคุมการทำงานของรถไม่สามารถตอบสนองได้เร็วเท่าที่ควรจะเป็น และต้องสั่งให้โปรแกรมในการควบคุมการทำงานของมอเตอร์ต้องค้างค่าข้อมูลได้

เอกสารอ้างอิง

1. ฐิรพงษ์ นามแถ่ง, "DAQดาต้าแอกควิซิชันบนพีซี",วารสารเซมิคอนดัคเตอร์,ฉบับที่186,2541, หน้า 158-162.
2. ฐิรพงษ์ นามแถ่ง, "DAQดาต้าแอกควิซิชันบนพีซี",วารสารเซมิคอนดัคเตอร์,ฉบับที่187,2541, หน้า 146-152.
3. ศภุชัย พุคยาภรณ์, "เครื่องวัดระยะทาง", วารสารเซมิคอนดัคเตอร์,ฉบับที่179,2536,หน้า 24-30.
4. Bruce Mielke, "Integrated Computer Graphics",St.Paul,West Publishing Company,1991
5. Haim Levkowitz and Gabor T. Herman, "Color Scales for Image Data",IEEE Computer Graphics and Application,january,1992,pp.72-80
6. James D.Foley,Andries Van Dam,Steven K.Feiner and John F.Hughes, "Computer Graphics Principles and Displays" ,secound edition,Singapore,Addison-Wesley Publishing Company,1990
7. James D.Foley and Jack Grimes, "Using Color in Computer Graphics", IEEE Computer Graphics and Application,january,1988,pp.25-27
8. Neubert,Herman K.P., " Instrumemt transducers",Oxford,Clarrondon Press,1975

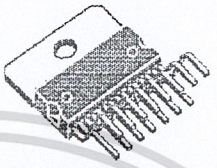
ภาคผนวก ก.
ข้อมูลอุปกรณ์

1. L 298
2. AT89S8252

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

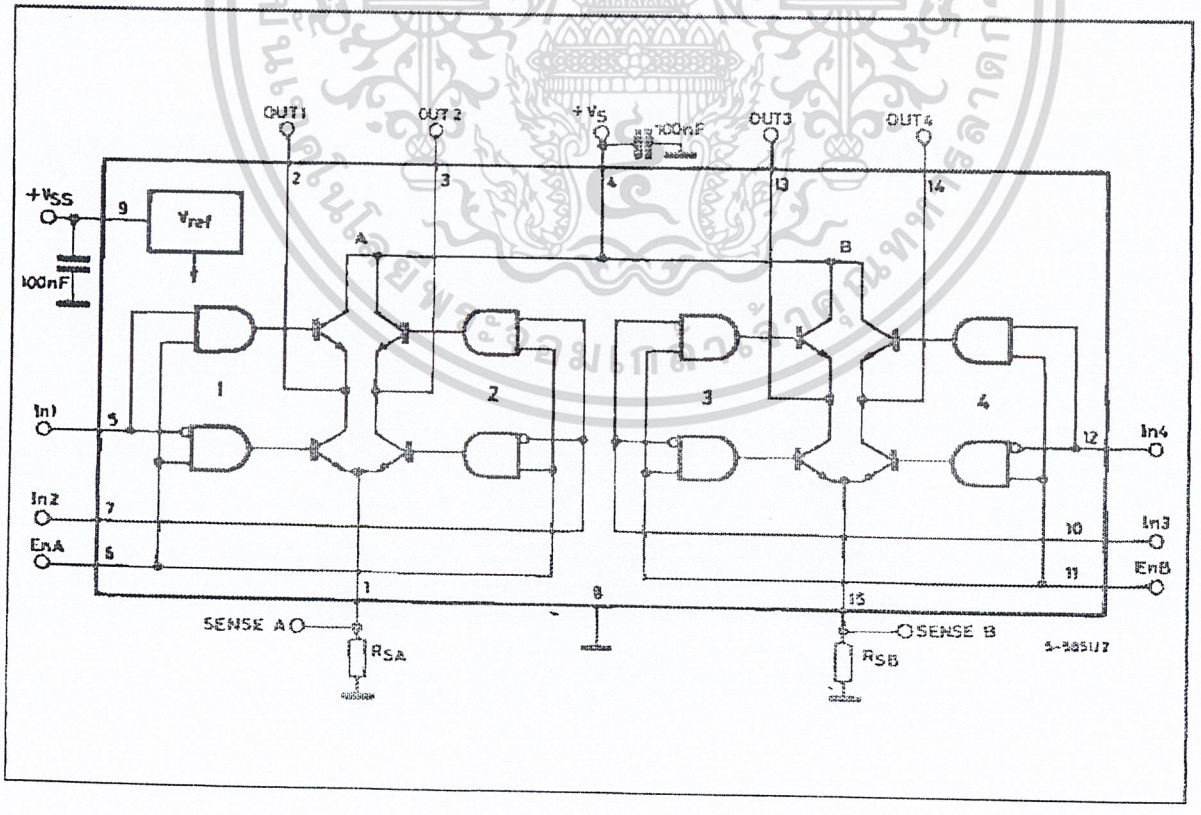
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

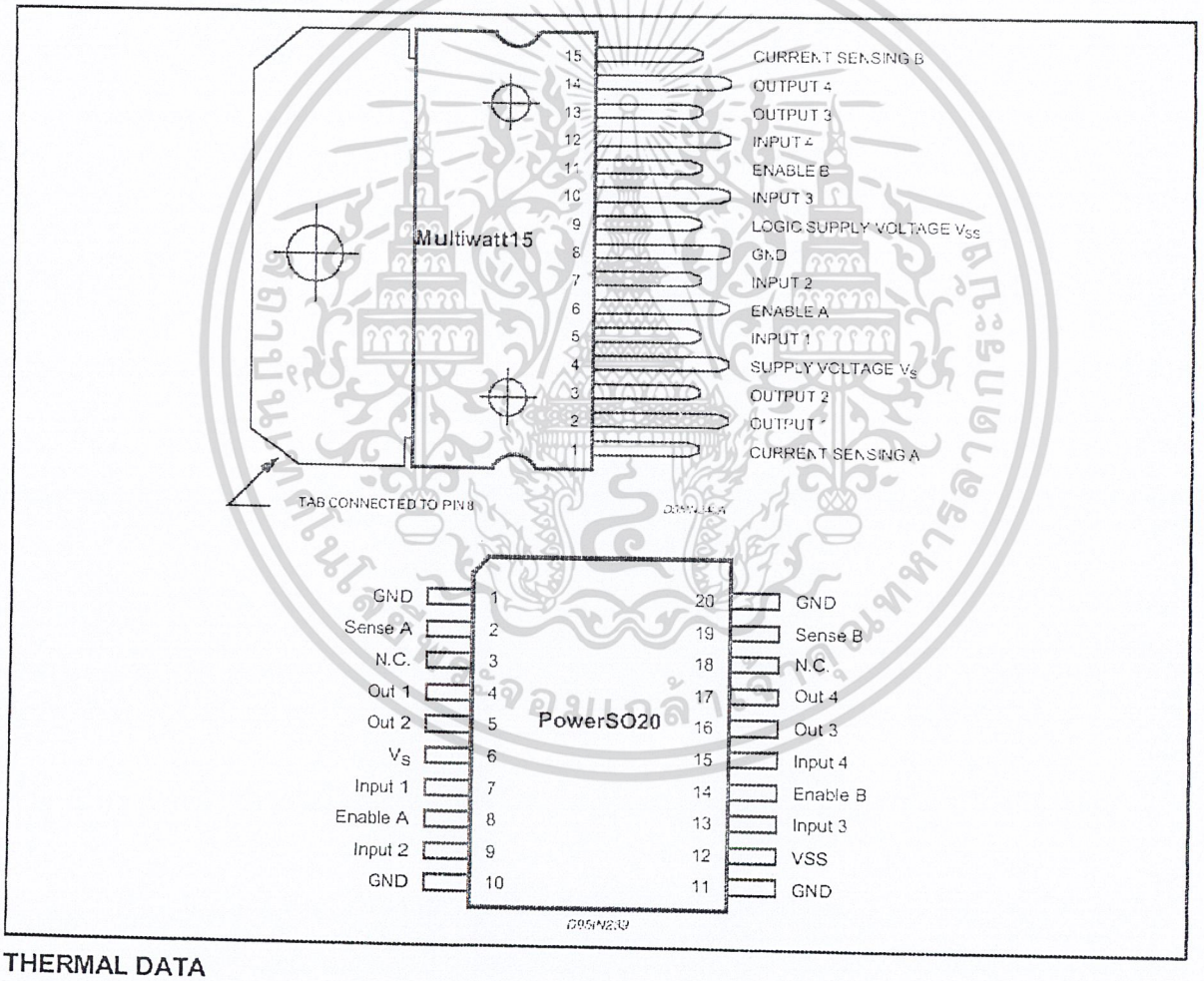
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

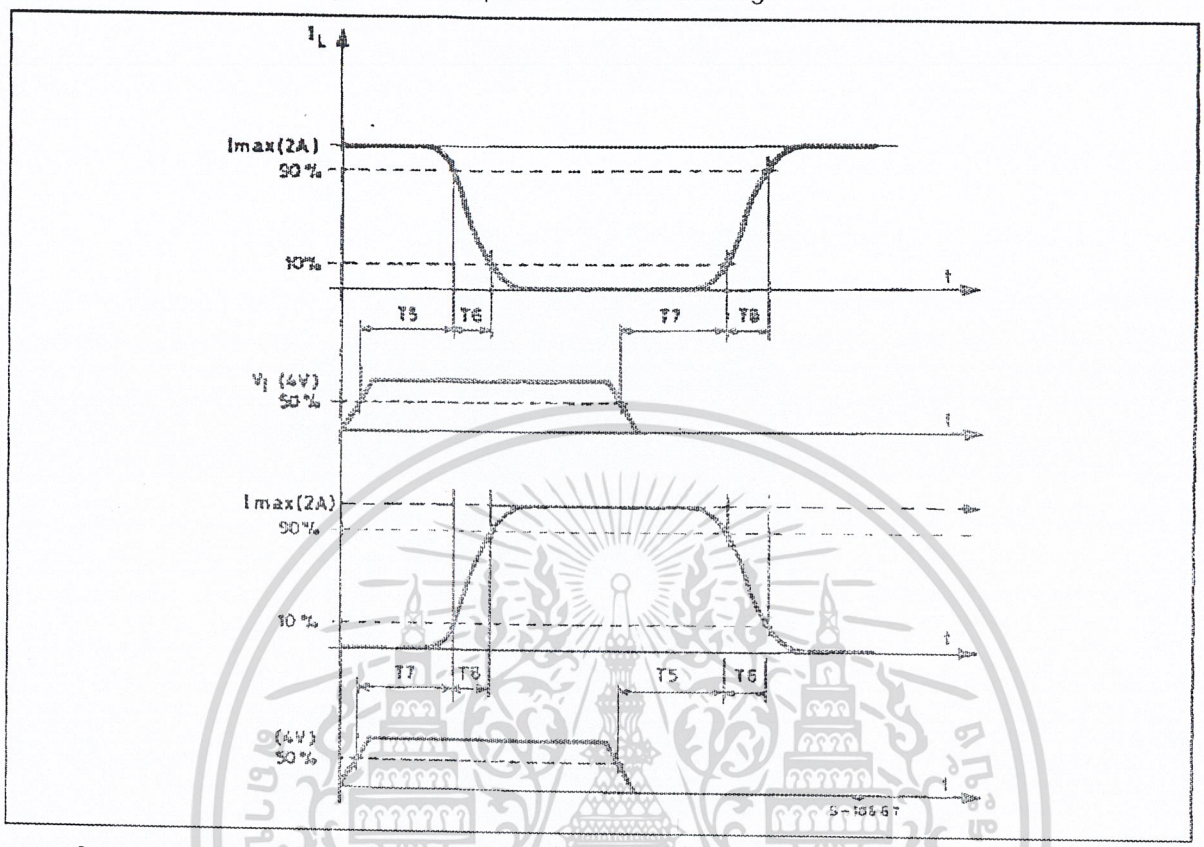
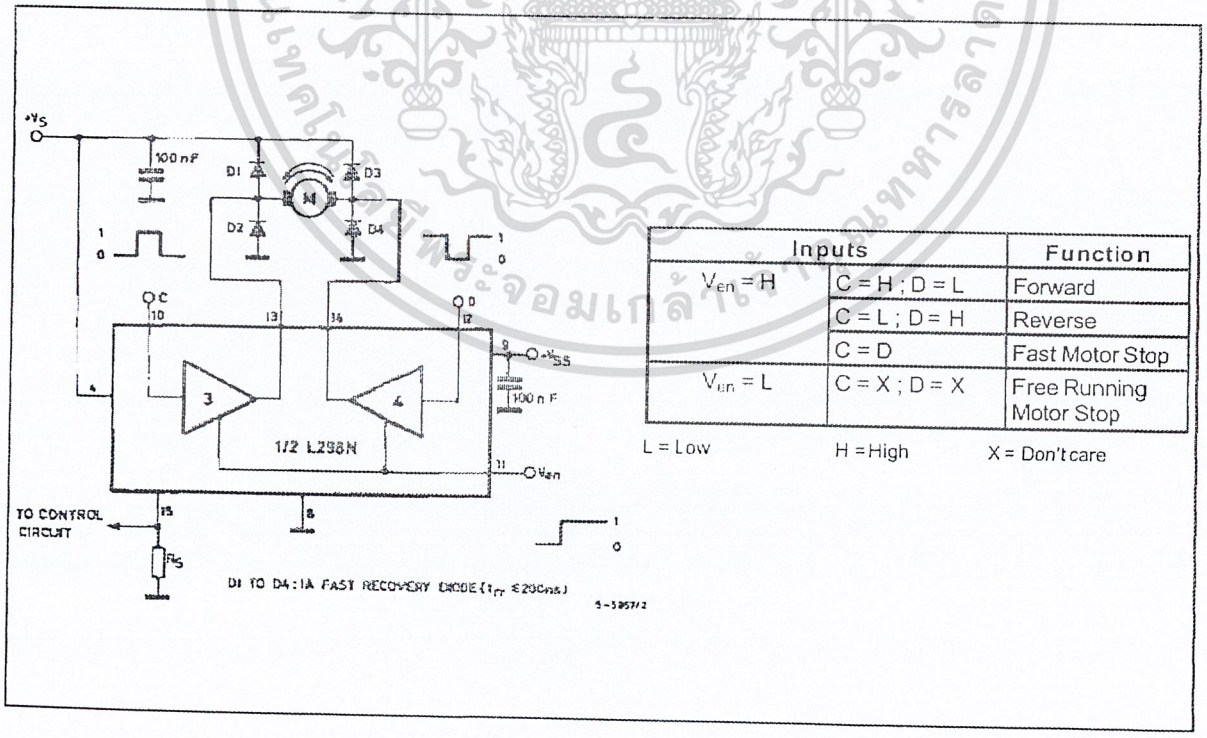


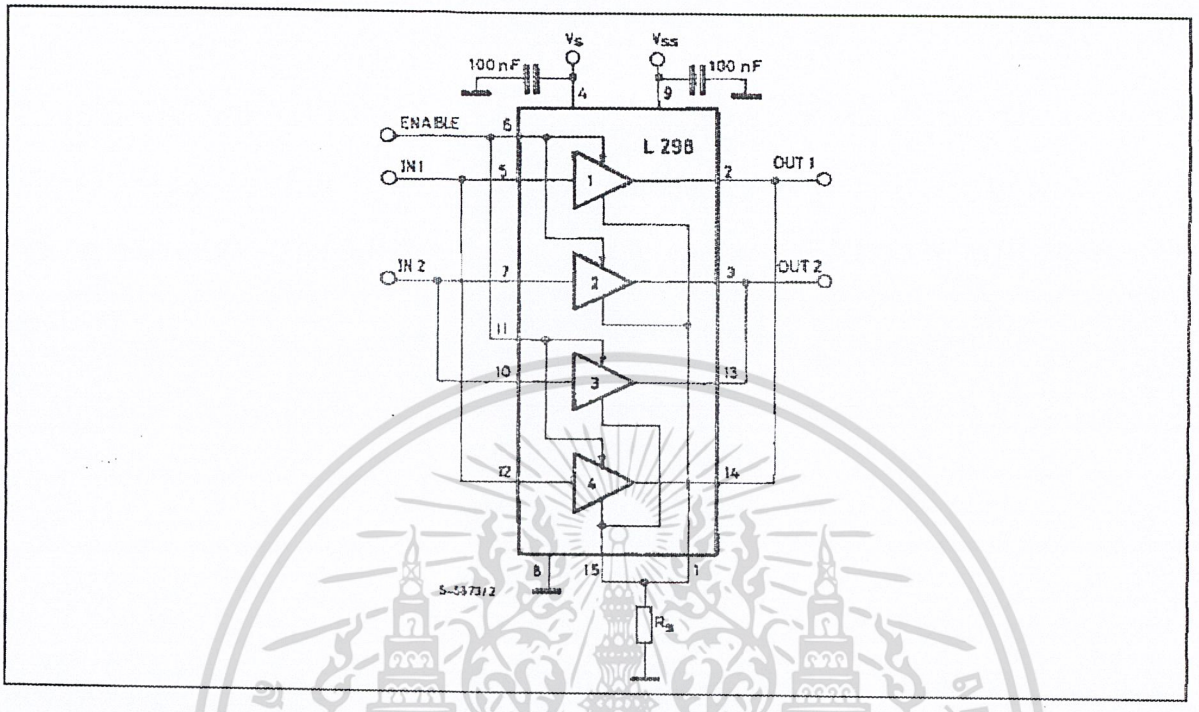
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements ($t_{rr} \leq 200$ nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

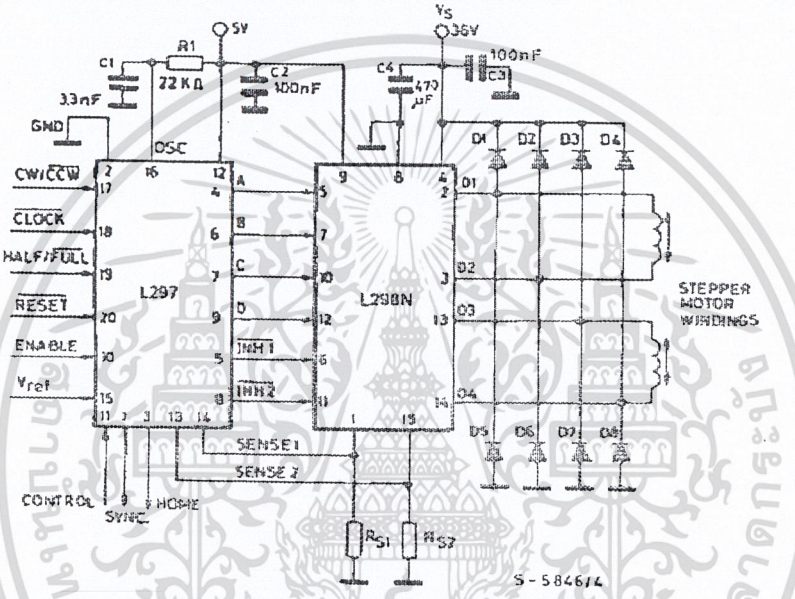
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2$ V @ I = 2 A ; trr ≤ 200 ns

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
- 4.0V 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

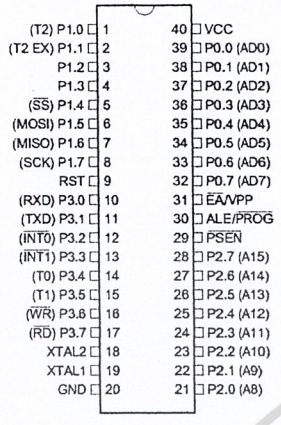
8-Bit Microcontroller with 8K Bytes Flash

AT89S8252

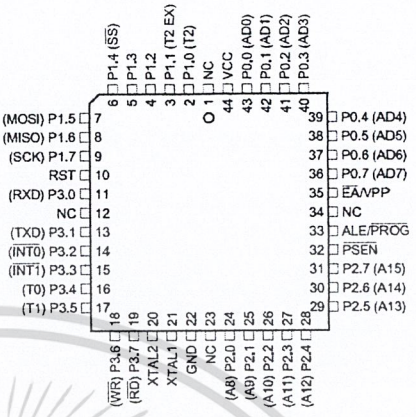
0401D-A–12/97

Pin Configurations

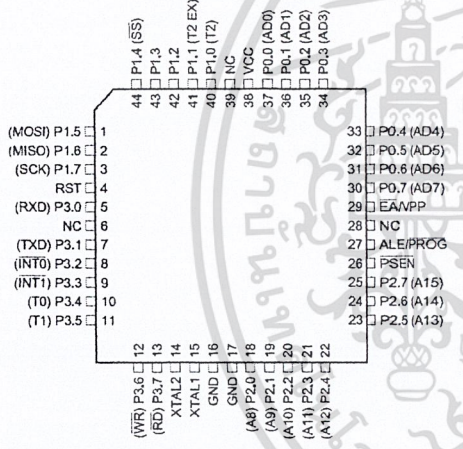
PDIP



PLCC



PQFP/TQFP



Pin Description

V_{CC}
Supply voltage.

GND
Ground.

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

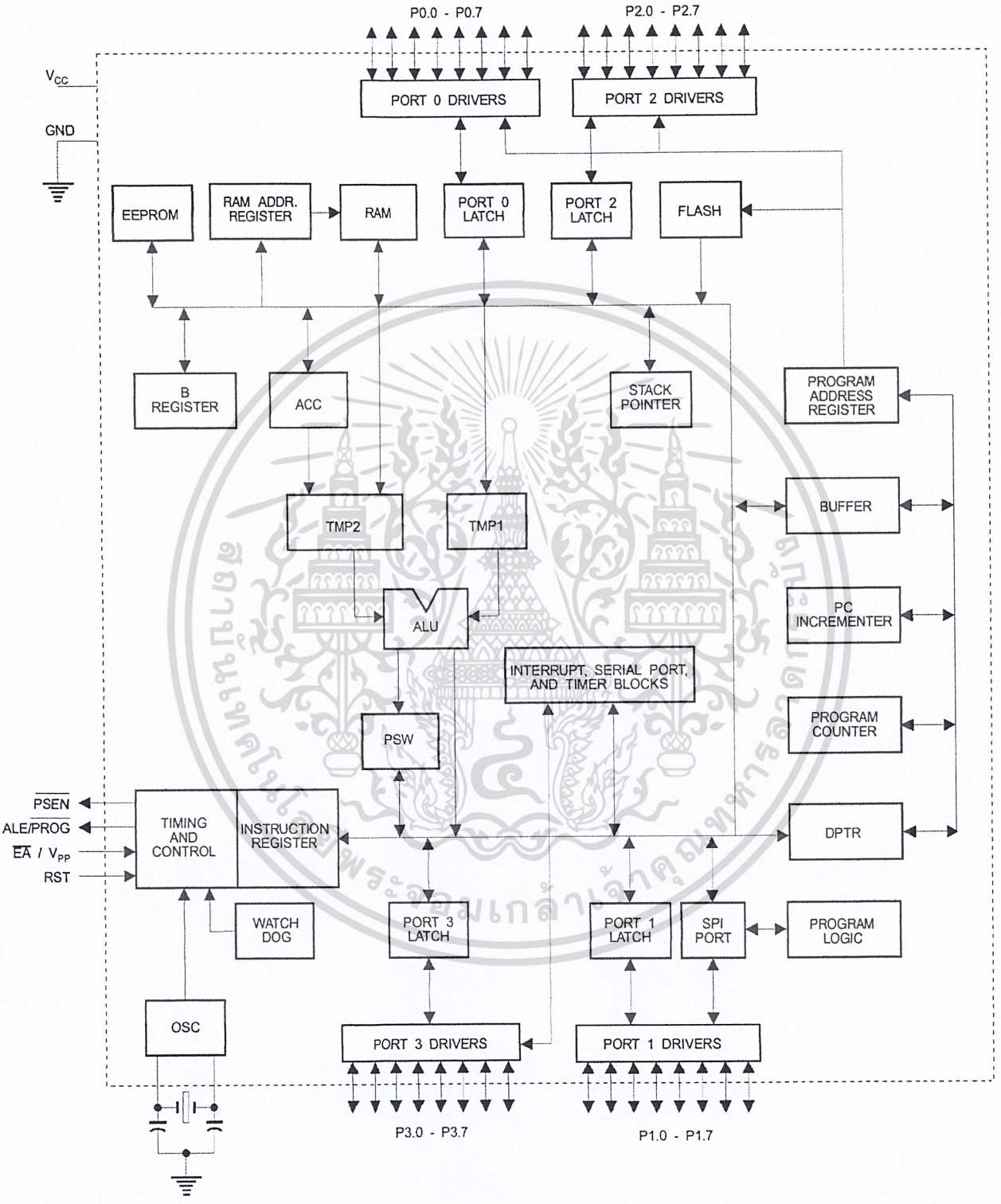
Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

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

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

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.

Block Diagram



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

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

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

Port 3

Port 3 is an 8 bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the 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.

$\overline{EA}$/V_{PP}

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

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.

Table 1. AT89S8252 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000					SPCR 000001XX			0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000		SPSR 00XXXXXX						0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111						WMCON 00000010		97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	SPDR XXXXXXXX	PCON 0XXX0000	87H

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.

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.

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.

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.

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 3. WMCON—Watchdog and Memory Control Register

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

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

Symbol	Function
PS2 PS1 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 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.

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 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: SPR1 SPR0 SCK = $F_{osc.}$ divided by 0 0 4 0 1 16 1 0 64 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

Bit	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
	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			Period (nominal)
PS2	PS1	PS0	
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

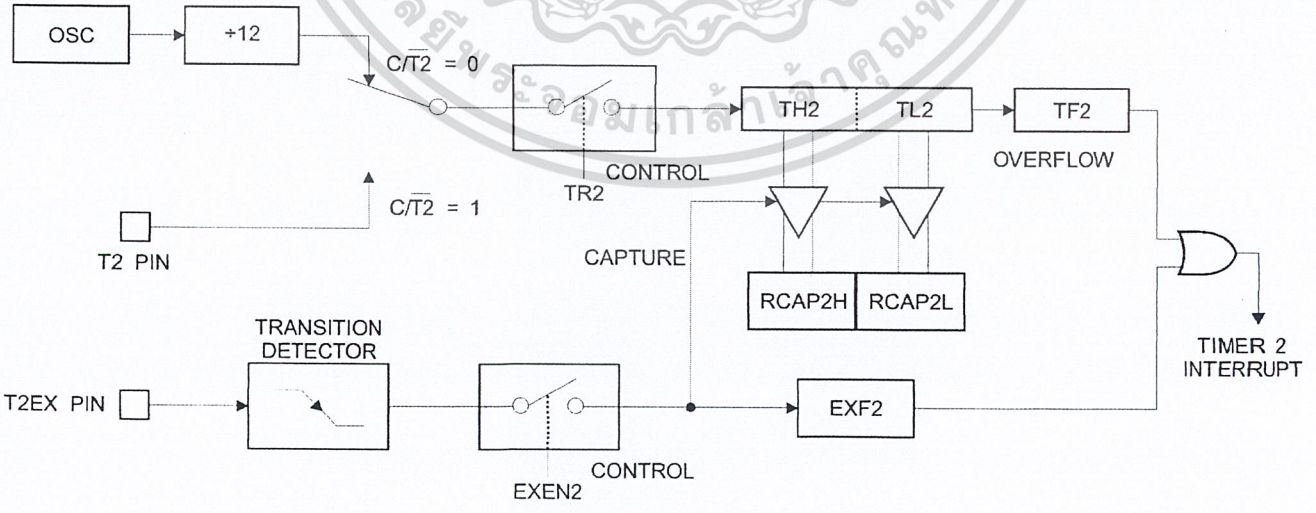
In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16 bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1-to-0 transition at external 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.

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

Figure 1. Timer 2 in Capture Mode



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)

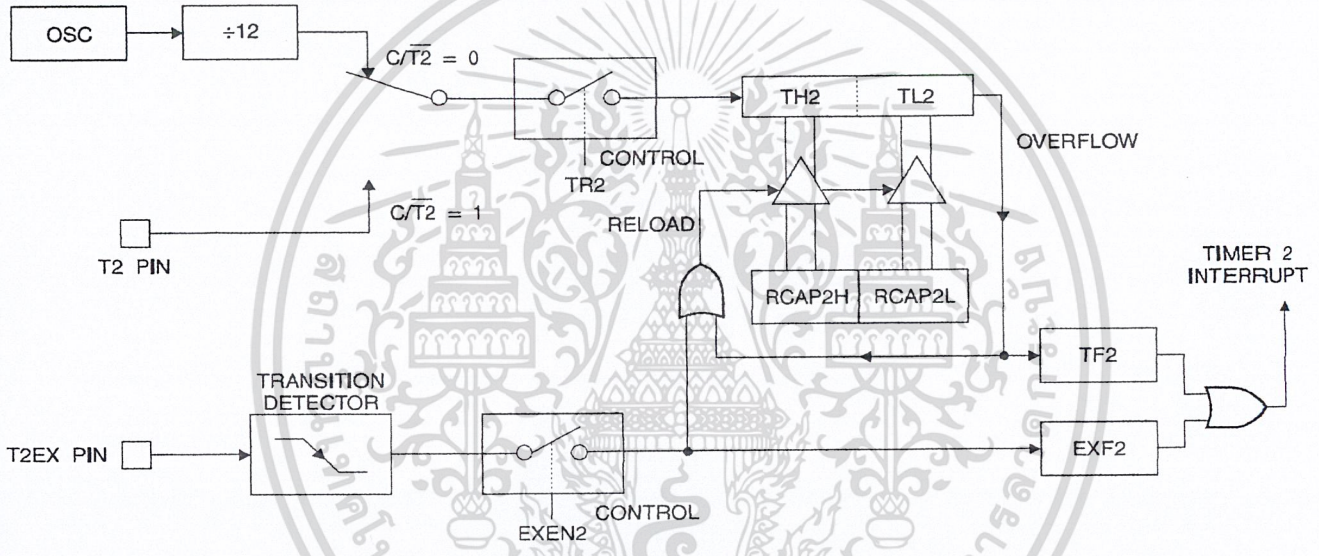


Table 9. T2MOD—Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

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

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)

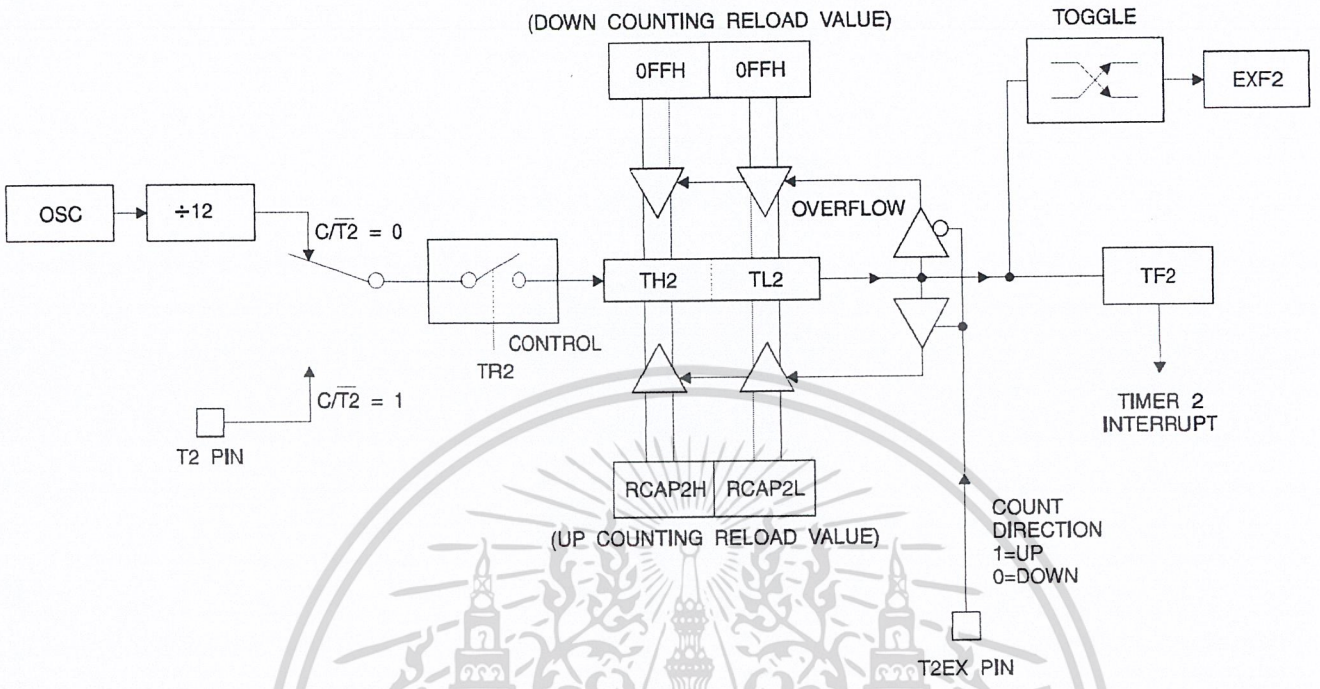
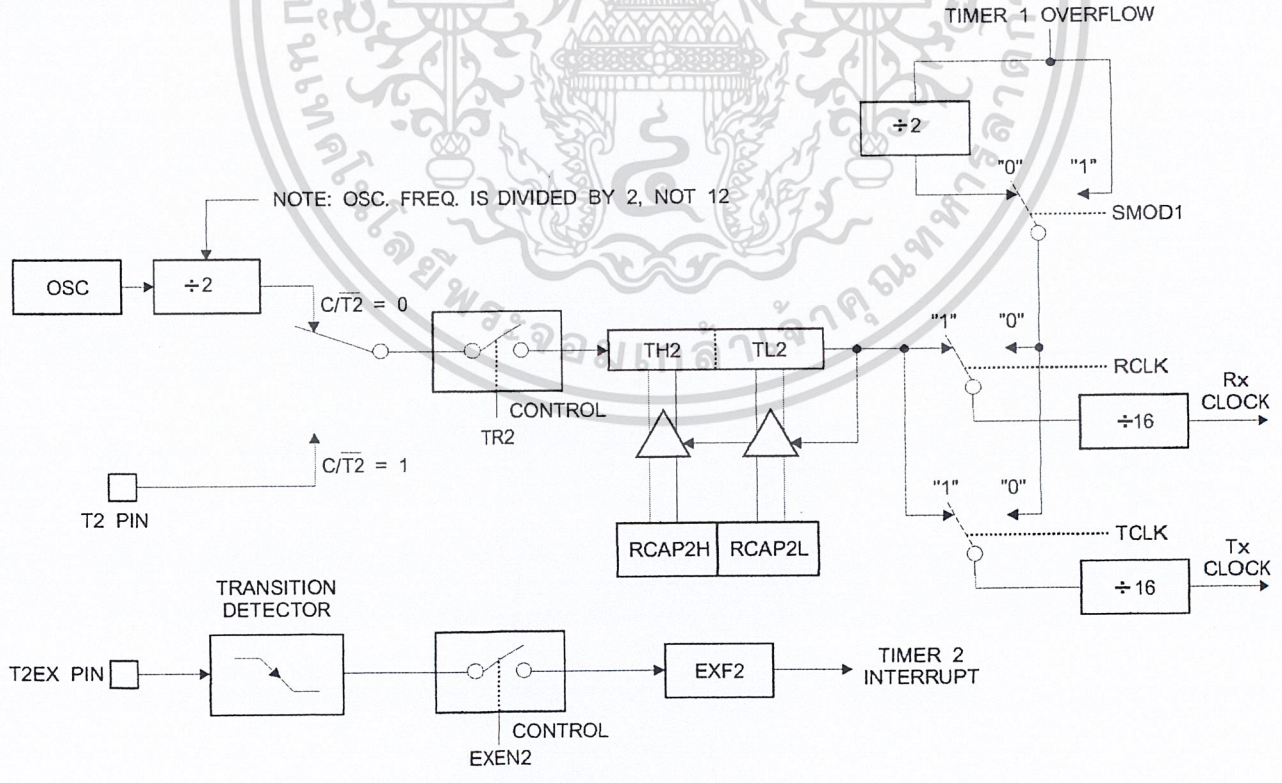


Figure 4. Timer 2 in Baud Rate Generator Mode



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.

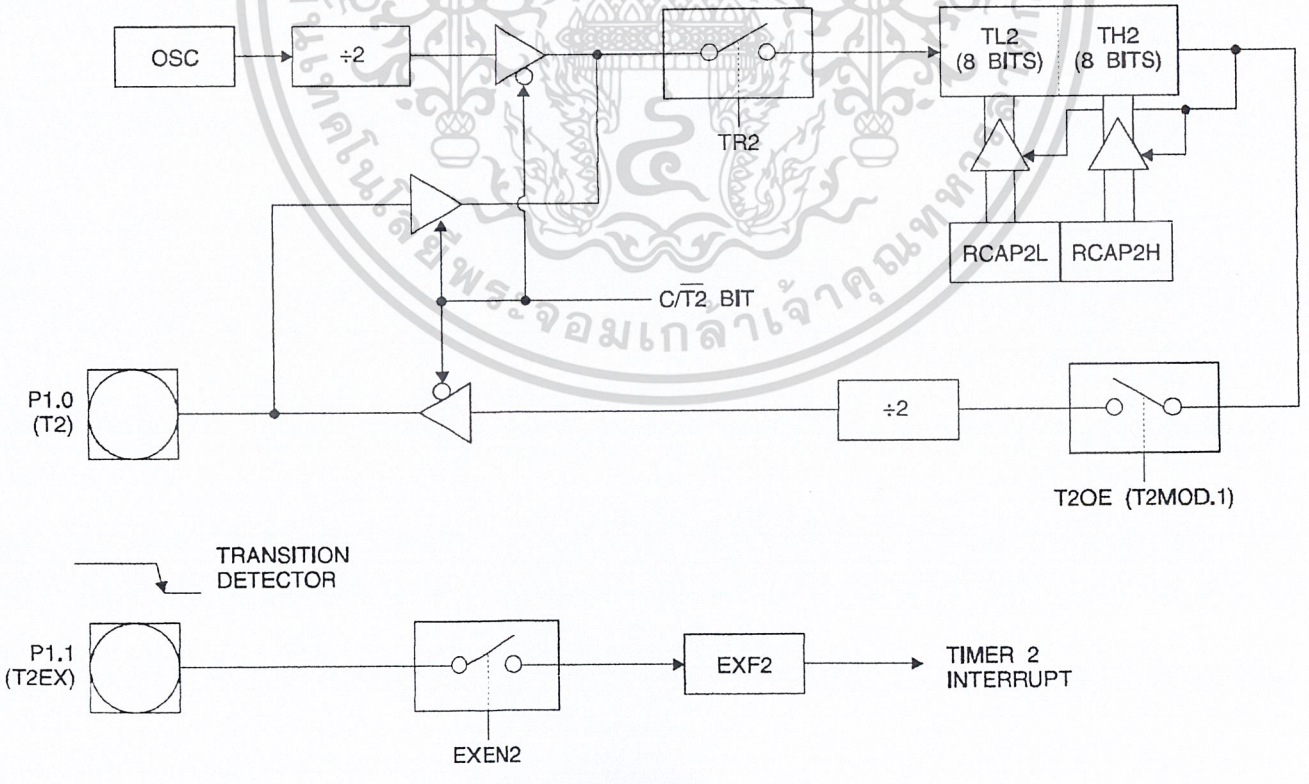
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

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.

Figure 5. Timer 2 in Clock-Out Mode



Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. This pin, besides being a regular I/0 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.

UART

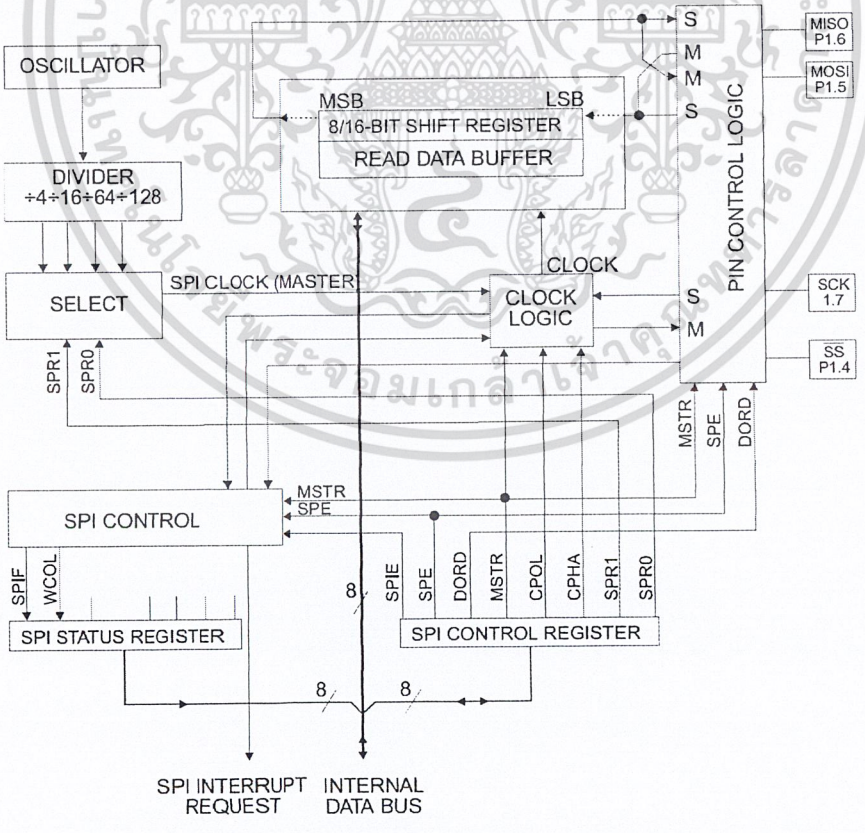
The UART in the AT89S8252 operates the same way as the UART in the AT89C51, AT89C52 and AT89C55. For further information, see the October 1995 Microcontroller Data Book, page 2-49, section titled, "Serial Interface."

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)

Figure 6. SPI Block Diagram



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 Figures 8 and 9.

Figure 7. SPI Master-Slave Interconnection

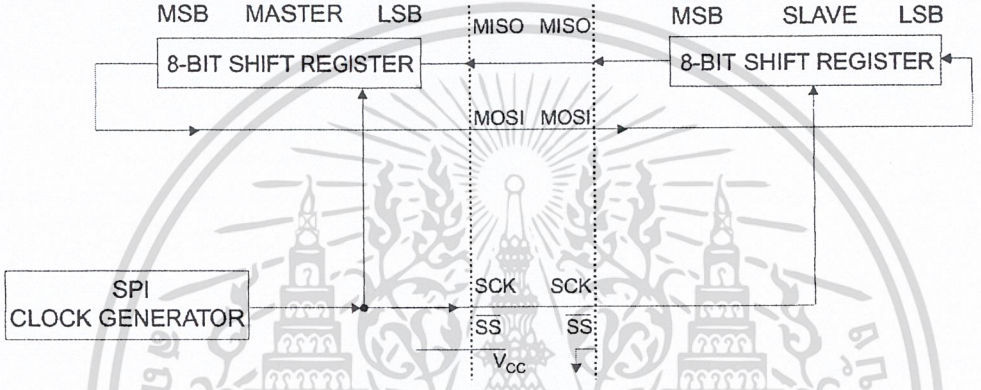
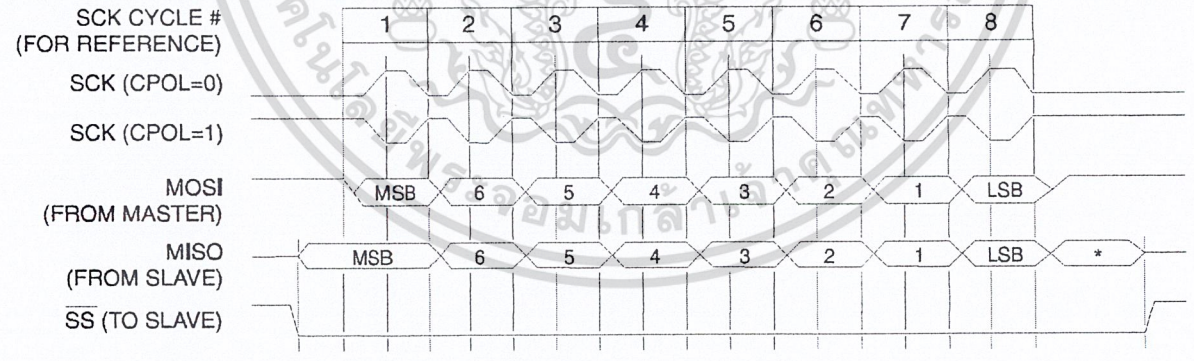
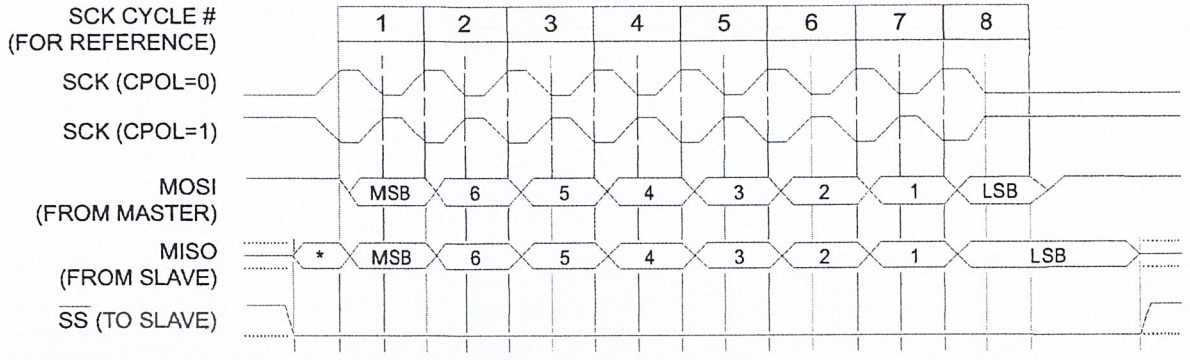


Figure 8. SPI transfer Format with CPHA = 0



*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

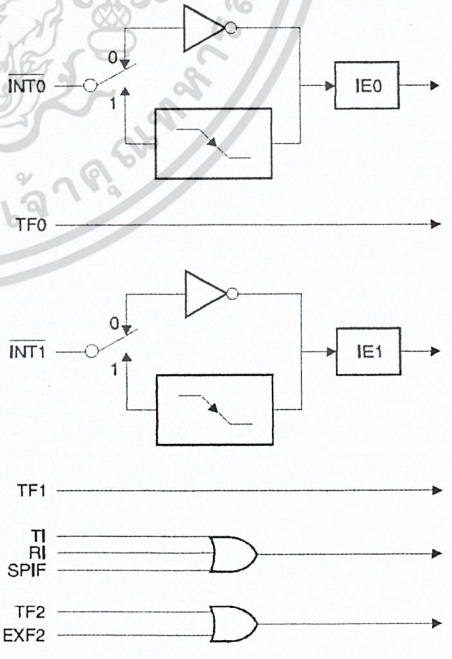


Figure 11. Oscillator Connections

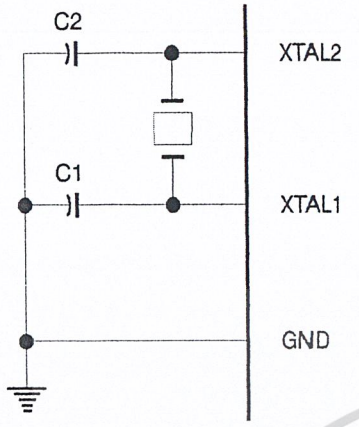
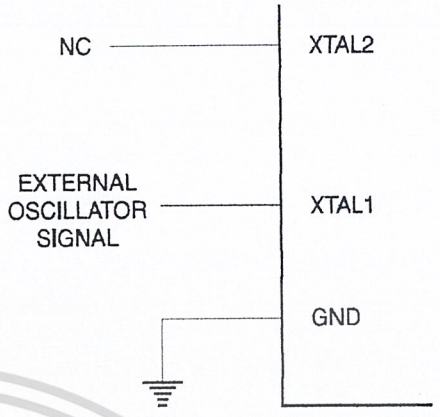


Figure 12. External Clock Drive Configuration



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

Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 11. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 12. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

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

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

Power Down Mode

In the power down mode, the oscillator is stopped and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. Exit from power down 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.

Status of External Pins During Idle and Power Down Modes

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

Program Memory Lock Bits

The 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				Protection Type
	LB1	LB2	LB3	
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.

$\overline{DATA}$ 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 V_{CC} 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	LB1 LB2 LB3 x x111	xxxx xxxx	Write lock bits. Set LB1, LB2 or LB3 = "0" to program lock bits.

Notes: 1. $\overline{\text{DATA}}$ 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 14. Programming the Flash/EEPROM Memory

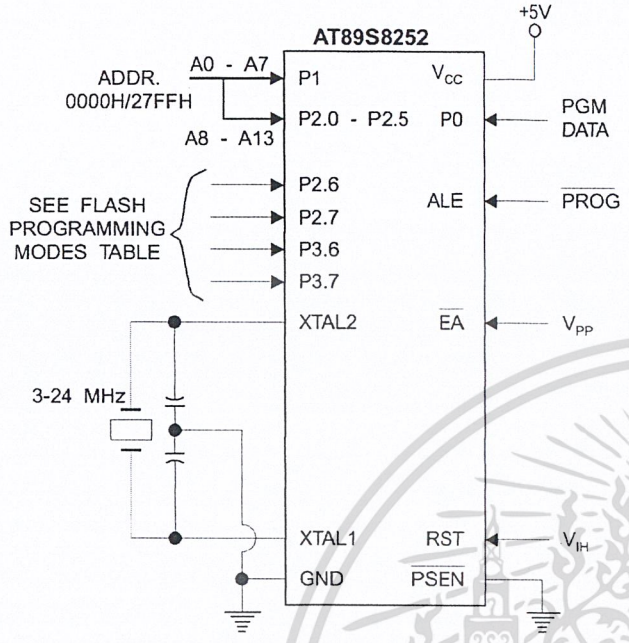


Figure 15. Flash/EEPROM Serial Downloading

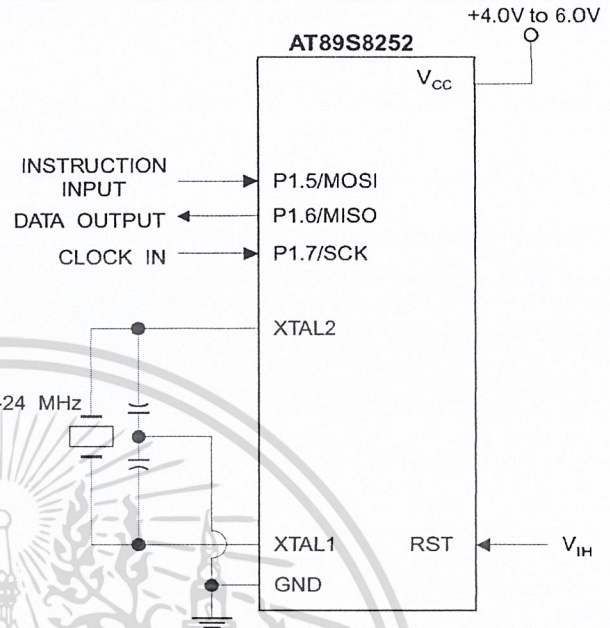
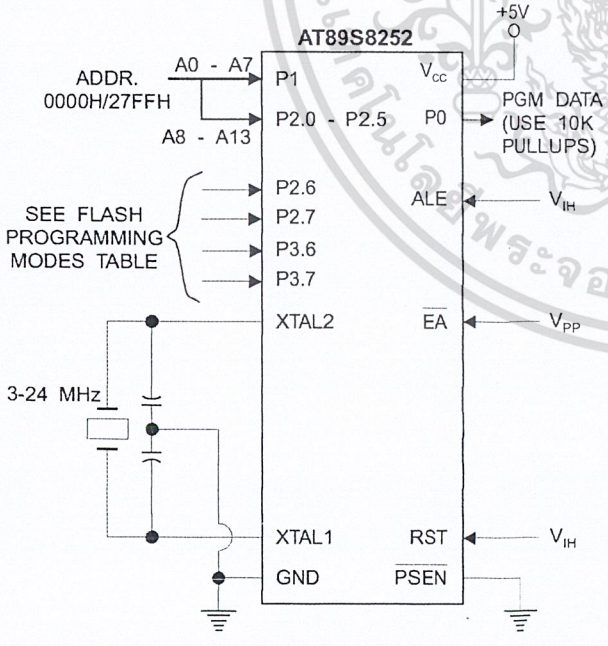


Figure 16. Verifying the Flash/EEPROM Memory

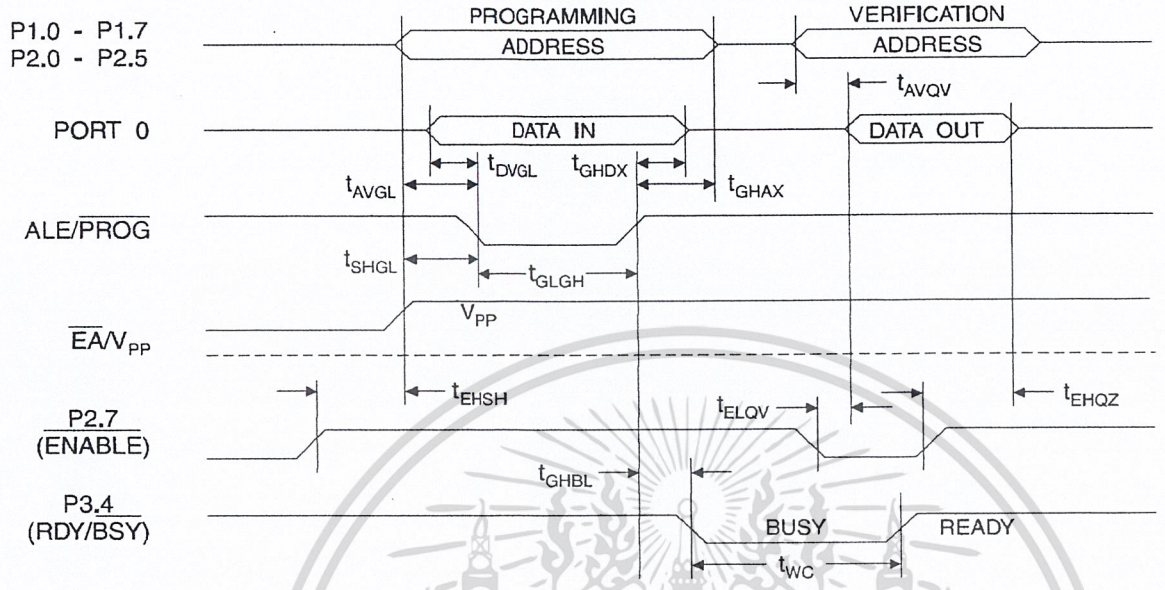


Flash Programming and Verification Characteristics-Parallel Mode

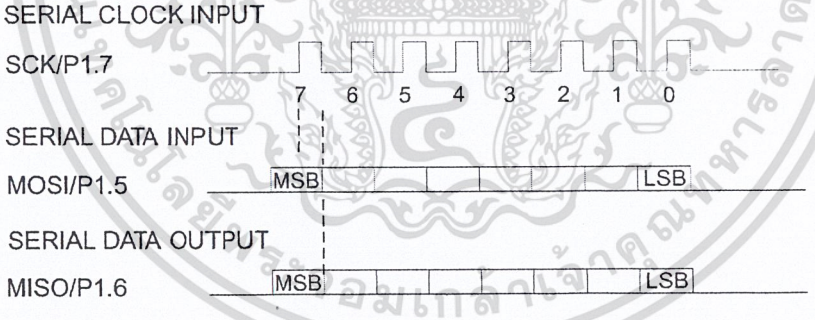
T_A = 0°C to 70°C, V_{CC} = 5.0V ± 10%

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 High Voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage (1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.5	V
V_{OL1}	Output Low Voltage (1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.5	V
V_{OH}	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		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

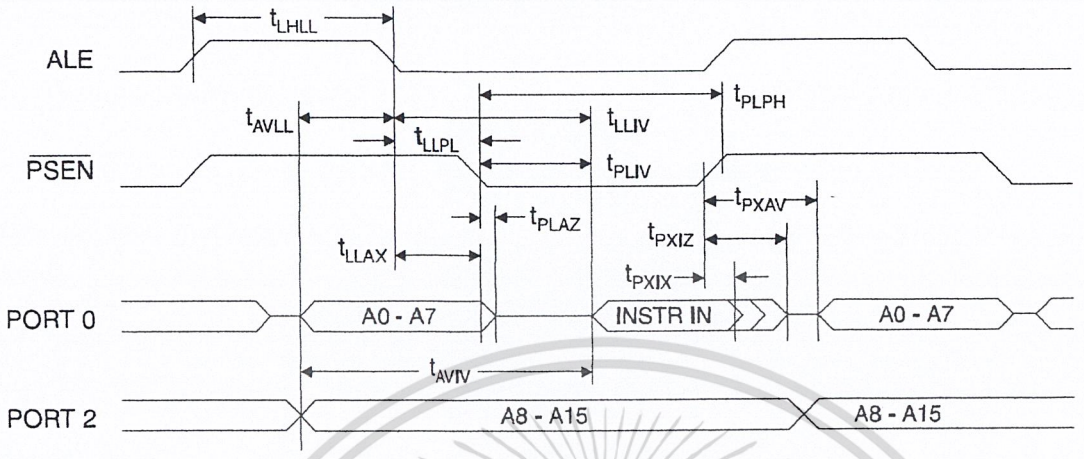
AC Characteristics

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

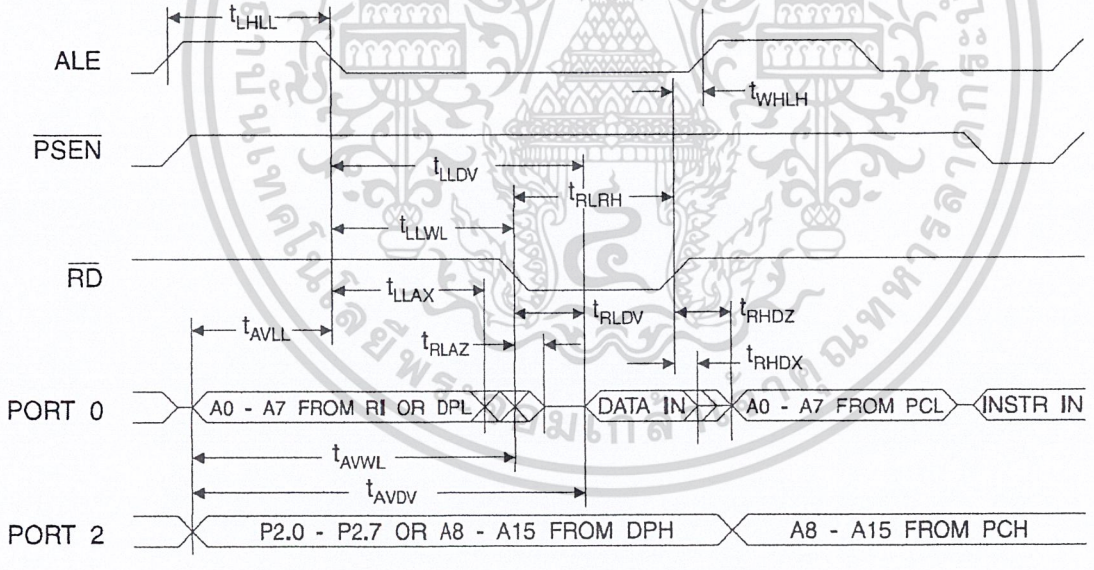
External Program and Data Memory Characteristics

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

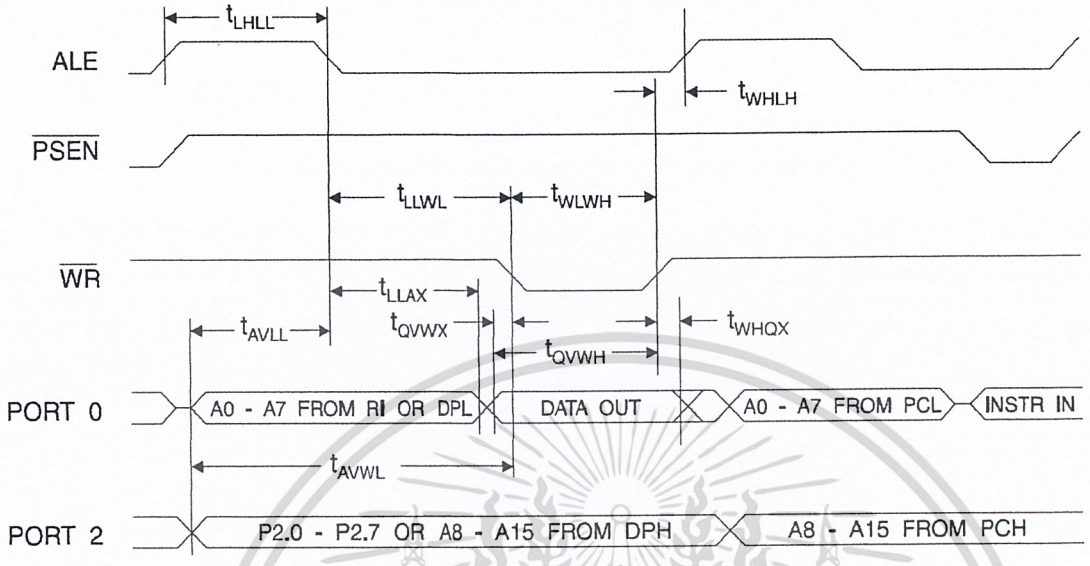
External Program Memory Read Cycle



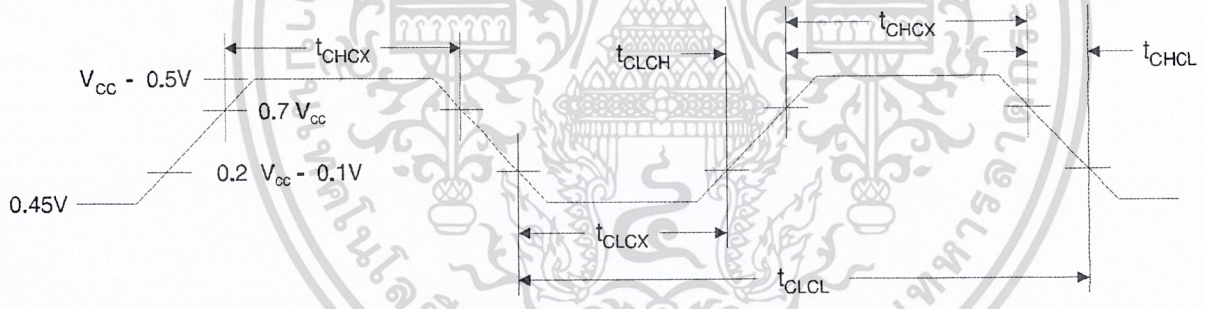
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

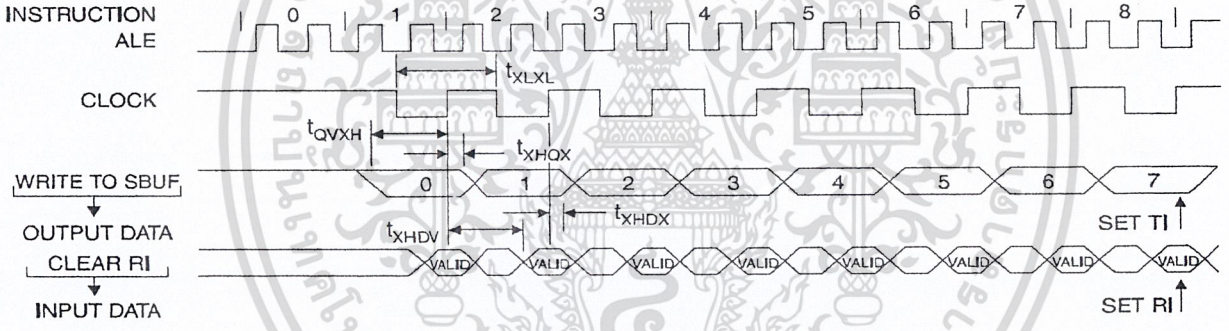
Symbol	Parameter	V_{CC} = 4.0V to 6.0V		Units
		Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

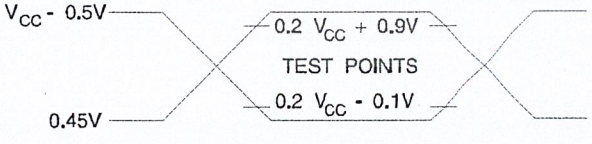
The values in this table are valid for V_{CC} = 4.0V to 6V and Load Capacitance = 80 pF.

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

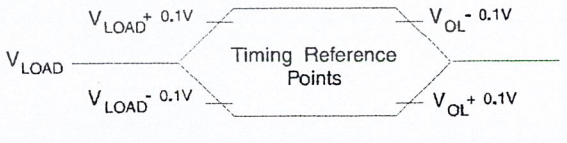


AC Testing Input/Output Waveforms(1)



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.

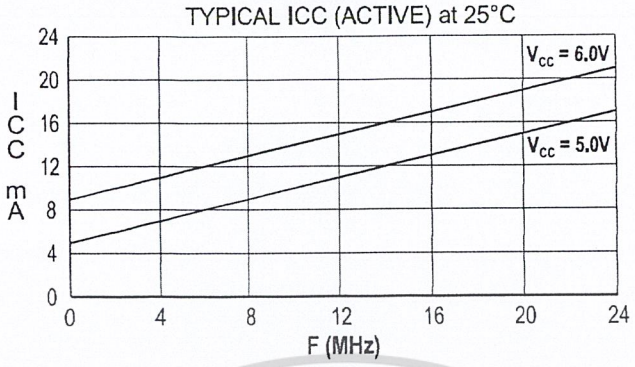
Float Waveforms(1)



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.

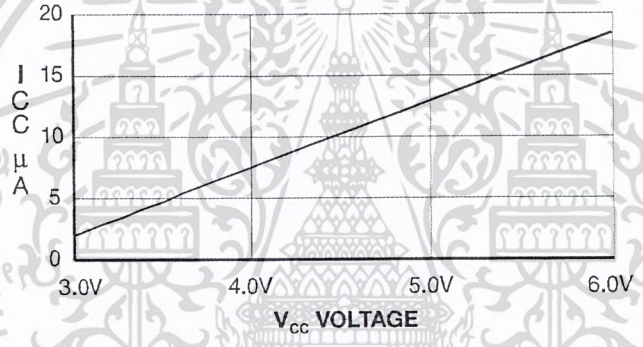
AT89S8252

TYPICAL ICC (ACTIVE) at 25°C



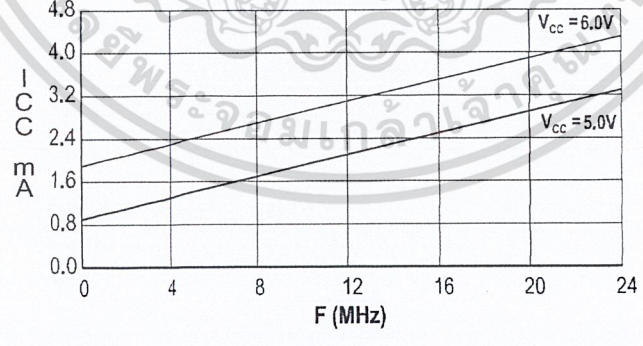
AT89S8252

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
16	4.0V to 6.0V	AT89S8252-16AA AT89S8252-16JA AT89S8252-16PA AT89S8252-16QA	44A 44J 40P6 44Q	Automotive (-40°C to 105°C)
24	4.0V to 6.0V	AT89S8252-24AC AT89S8252-24JC AT89S8252-24PC AT89S8252-24QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
	4.0V to 6.0V	AT89S8252-24AI AT89S8252-24JI AT89S8252-24PI AT89S8252-24QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
33	4.5V to 5.5V	AT89S8252-33AC AT89S8252-33JC AT89S8252-33PC AT89S8252-33QC	44A 44J 40P6 44Q	Commercial (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)

ภาคผนวก ข

โปรแกรมควบคุมการทำงานของรถสำรวจ

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include "Test4.h"

#include "SurVey4.h"
#include "About1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "Video"
#pragma resource "*.dfm"
TBaseForm *BaseForm;
//---------------------------------------------------------------------------
__fastcall TBaseForm::TBaseForm(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------------
void __fastcall TBaseForm::Button1Click(TObject *Sender)
{
 Video1->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TBaseForm::Button2Click(TObject *Sender)
{
 Video1->Enabled=false;
```

```
}
//---------------------------------------------------------------------------



void __fastcall TBaseForm::Close2Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Close1Click(TObject *Sender)
{
DataForm->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button8Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button3Click(TObject *Sender)
{
DataForm->Edit4->Text = IntToStr(1);
DataForm->Button1Click(Sender);
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button4Click(TObject *Sender)
{
```

```
DataForm->Edit4->Text = IntToStr(3);
DataForm->Button1Click(Sender);
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button5Click(TObject *Sender)
{
DataForm->Edit4->Text = IntToStr(4);
DataForm->Button1Click(Sender);
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button6Click(TObject *Sender)
{
DataForm->Edit4->Text = IntToStr(2);
DataForm->Button1Click(Sender);
}
//---------------------------------------------------------------------------


void __fastcall TBaseForm::Button9Click(TObject *Sender)
{
 DataForm->Button5Click(Sender);
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button3KeyPress(TObject *Sender, char &Key)
{
    switch (Key) {
      case '8' :
        Button3Click(Sender);
        break;
```

```
        case '4' :
          Button4Click(Sender);
          break;
        case '6' :
          Button5Click(Sender);
          break;
        case '2' :
          Button6Click(Sender);
        case '5' :
          Button9Click(Sender);
          break; }
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button9KeyPress(TObject *Sender, char &Key)
{
    switch (Key) {
        case '8' :
          Button3Click(Sender);
          break;
        case '4' :
          Button4Click(Sender);
          break;
        case '6' :
          Button5Click(Sender);
          break;
        case '2' :
          Button6Click(Sender);
        case '5' :
          Button9Click(Sender);
          break; }
}
```

```
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button6KeyPress(TObject *Sender, char &Key)
{
    switch (Key) {
      case '8' :
        Button3Click(Sender);
        break;
      case '4' :
        Button4Click(Sender);
        break;
      case '6' :
        Button5Click(Sender);
        break;
      case '2' :
        Button6Click(Sender);
      case '5' :
        Button9Click(Sender);
        break; }
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button5KeyPress(TObject *Sender, char &Key)
{
    switch (Key) {
      case '8' :
        Button3Click(Sender);
        break;
      case '4' :
        Button4Click(Sender);
        break;
      case '6' :
```

```
        Button5Click(Sender);
        break;
      case '2' :
        Button6Click(Sender);
      case '5' :
        Button9Click(Sender);
        break; }
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button4KeyPress(TObject *Sender, char &Key)
{
    switch (Key) {
      case '8' :
        Button3Click(Sender);
        break;
      case '4' :
        Button4Click(Sender);
        break;
      case '6' :
        Button5Click(Sender);
        break;
      case '2' :
        Button6Click(Sender);
      case '5' :
        Button9Click(Sender);
        break; }
}
//---------------------------------------------------------------------------

void __fastcall TBaseForm::Button1KeyPress(TObject *Sender, char &Key)
{
```

```
    switch (Key) {
      case '8' :
        Button3Click(Sender);
        break;
      case '4' :
        Button4Click(Sender);
        break;
      case '6' :
        Button5Click(Sender);
        break;
      case '2' :
        Button6Click(Sender);
      case '5' :
        Button9Click(Sender);
        break; }
}
//---------------------------------------------------------------------

void __fastcall TBaseForm::Button2KeyPress(TObject *Sender, char &Key)
{
    switch (Key) {
      case '8' :
        Button3Click(Sender);
        break;
      case '4' :
        Button4Click(Sender);
        break;
      case '6' :
        Button5Click(Sender);
        break;
      case '2' :
        Button6Click(Sender);
```

```
        case '5' :
          Button9Click(Sender);
          break; }
}
//---------------------------------------------------------------------------



void __fastcall TBaseForm::About1Click(TObject *Sender)
{
FormAbout->ShowModal();
}
//---------------------------------------------------------------------------
```