น้ำพุดนตรี

DANCING FOUNTAIN

โดย

นายอธิป ลิ้มเลิศเจริญวนิช

นายวรกานต์ สุขเจริญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

น้ำพุดนตรี

DANCING FOUNTAIN

โดย

นายอชิป ลิ้มเลิศเจริญวนิช รหัส 48015190

นายวรกานต์ สุขเจริญ รหัส 48015222

อาจารย์ที่ปรึกษา

รศ.ดร.ชูชาติ ปิณฑวิรุฬห์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

รายงานปีการศึกษา 2550

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง น้ำพุดนตรี

ผู้จัดทำ

1. นาย วรกานต์ สุขเจริญ รหัส 48015222

2. นาย อธิป ลิ้มเลิศเจริญวนิช รหัส 48015190

.....อาจารย์ที่ปรึกษา

(รศ.ดร.ชูชาติ ปีณฑวิรุจน์)

...../...../.....

น้ำพุดนตรี

นาย อธิป ลิ้มเลิศเจริญวนิช รหัส 48015190
นาย วรกานต์ สุขเจริญ รหัส 48015222
รศ.ดร.ชูชาติ ปิณฑวิรุจน์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2550

บทคัดย่อ

น้ำพุเป็นอุปกรณ์ที่ใช้ในงานตกแต่งสวนทั้งกลางแจ้งและภายในอาคารเพื่อความสวยงามอีกทั้งยังเป็นจุดสนใจของผู้ที่มาเยี่ยมชม และหากเป็นน้ำพุที่สามารถปรับระดับความสูงของน้ำได้ตามจังหวะของเสียงดนตรีได้ด้วยแล้วจะยิ่งกลายเป็นจุดสนใจมากขึ้นด้วยแต่ในการติดตั้งน้ำพุที่มีความสามารถดังกล่าวจำเป็นต้องมีค่าใช้จ่ายในการติดตั้งและการบำรุงรักษาสูงด้วยเช่นกัน

โครงงานนี้นำเสนอการประยุกต์ใช้งานไมโครคอนโทลเลอร์ในการควบคุมระดับของการปิดและเปิดวาล์วน้ำที่หัวของน้ำพุแต่ละหัวด้วยเซอร์โวมอเตอร์รูปแบบการทำงานของน้ำพุจะสามารถเลือกรูปแบบได้จากการโปรแกรมลงบนตัวของไมโครคอนโทลเลอร์เองหรือทำการกำหนดรูปแบบจากเครื่องคอมพิวเตอร์และในอีกรูปแบบหนึ่งเป็นการรับสัญญาณเสียงจากแหล่งกำเนิดเสียงเข้ามาเพื่อกำหนดระดับของน้ำให้สูงต่ำตามจังหวะเสียงดนตรี

ในโครงงานนี้ได้ใช้ไมโครคอนโทลเลอร์ PIC16F628 ของบริษัท Microchip ควบคุมวาล์วด้วยเซอร์โวมอเตอร์ Futaba S3003 วาล์วน้ำที่ให้เป็นวาล์วเป็นบอลวาล์วที่นำมาประกอบกับแท่นเพื่อให้สามารถติดตั้งเซอร์โวมอเตอร์เพื่อควบคุมวาล์วได้อีกทั้งยังมีการปรับความคล่องตัวของลูกบิดวาล์วให้เลื่อนไหลมากที่สุดเพื่อลดภาระของเซอร์โวมอเตอร์ขณะทำงาน

Dancing Fountain

Mr.A-Thip Limlertcharoenvanit ID.48015190
Mr.Warakan Sukcharoen ID.48015222
Assoc.Prof.Dr.Chuchart Pindhavirooj Advisor
Education Year 2007

Abstract

Fountain is one of the most popular decorations both indoor and outdoor of the buildings. If the height of the fountain can be adjusted according to the beat of music it will attract more attentions. On the other hand, the construction and maintenance cost are also high as well.

This project presents the application of Microcontroller in controlling the turning on / off of the fountain nozzles with servo motors. The operation of the fountain can be selected from the programming microcontrollers or by operating from the computer. Besides, it can be controlled by the sound signals to identify the height of the fountains according to the beat of the music.

In this project, the Microchip's PIC 16F628 microcontroller and Futaba S3003 servomotor are used. The ball valves are attached to the base in order to make they are able to install servomotor to control the valves. Further more, there is an adjustment in the movement ability of the valves to reduce the work power of servo motors.

กิตติกรรมประกาศ

โครงงานนี้จะไม่สามารถลุล่วงลงไปได้ ถ้าไม่ได้รับความร่วมมือและความช่วยเหลือจากหลาย ๆ ฝ่าย โดยเฉพาะอาจารย์ รศ.ดร.ชูชาติ ปิณฑวิรุจน์ (อาจารย์ที่ปรึกษา) และอาจารย์ในภาควิชาอิเล็กทรอนิกส์ทุกท่าน ที่ให้การอุปการะในการให้คำปรึกษาและแนะนำเกี่ยวกับโครงงานในครั้งนี้ และให้ยืมใช้เครื่องมือและอุปกรณ์อิเล็กทรอนิกส์ในการทดลอง และสั่งสอนให้ความรู้จนสามารถนำมาประยุกต์ใช้งานในการทำโครงงานครั้งนี้ อีกทั้ง พี่ๆห้อง CT LAB ทุกท่าน และเพื่อนทุกท่านที่ได้ร่วมทำงานและแลกเปลี่ยนความรู้ ความคิดเห็นซึ่งกันและกัน จนทำให้ผลงานที่ได้มีประสิทธิภาพและเป็นประโยชน์ต่อส่วนรวมอีกด้วย

นาย วรกานต์ สุขเจริญ
นาย อธิป ลิ้มเลิศเจริญวนิช
ผู้จัดทำโครงการ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

น้ำพุเกิดจากการเคลื่อนที่ของน้ำเนื่องจากแรงดันทำให้น้ำเคลื่อนที่ไปสู่ที่ที่มีแรงดันต่ำกว่า ดังจะเห็นได้จากการกำเนิดน้ำพุร้อนที่มีการพุ่งขึ้นมาจากภายใต้ผิวโลกสู่บนผิวโลกการที่น้ำเคลื่อนตัวผ่านชั้นหินและดินใต้ผิวโลกได้นั้นเกิดจากการที่ทางน้ำใต้ดินได้รับความร้อนจากชั้นหินเหลวที่มีความร้อนสูงน้ำที่ผ่านชั้นหิวเหลวนี้จะได้รับความร้อนทำให้น้ำเดือดและเกิดการขยายตัวของน้ำเป็นแรงดันอยู่ใต้ชั้นดินและหินเมื่อแรงดันสูงจนถึงจุดหนึ่งที่ชั้นของดินและหินไม่สามารถทนได้น้ำที่เดือดและมีแรงดันมากจะดันจนทะลุผ่านชั้นดินและหินที่อยู่ใต้ผิวขึ้นมาอยู่ที่บนพื้นผิวโลกด้วยแรงดันที่สูงมากพอเมื่อน้ำเดือดนี้พ้นจากผิวโลกแรงเฉื่อยที่มีอยู่จะทำให้น้ำเดือดพุ่งสูงขึ้นไปพ้นจากผิวโลกตามแรงดันของน้ำเดือดที่เหลืออยู่จึงเกิดเป็นน้ำพุร้อนส่วนน้ำเดือดที่เมื่อผ่านชั้นดินและหินที่อยู่ใต้ผิวโลกขึ้นมาแล้วแรงดันไม่เพียงพอก็จะเกิดเป็นบ่อน้ำร้อนทั้งสองจึงกลายแหล่งท่องเที่ยวทางธรรมชาติที่น่าสนใจของนักท่องเที่ยว

รูปที่ 1.1 น้ำพุตามธรรมชาติ

ด้วยหลักการของการเกิดน้ำพุร้อนข้างต้นทำให้มีการพัฒนาสร้างน้ำพุขึ้นโดยทำให้น้ำเกิดแรงดันขึ้นโดยใช้ปั๊มน้ำสร้างแรงดันสูงให้น้ำไปตามท่อที่มีปลายเปิดสิ่งที่เกิดขึ้นคือเมื่อน้ำไปถึงปลายของท่อน้ำจะมีแรงเฉื่อยที่มากพอที่จะเคลื่อนที่ต่อไปได้เช่นเดียวกันกับการกำเนิดน้ำพุร้อนส่วนของความสูงของน้ำจะถูกกำหนดโดยแรงดันที่ปั๊มทำได้ขนาดของท่อและขนาดของปลายท่อหากที่ค่าที่เหมาะสมจะทำให้ระบบของน้ำพุมีประสิทธิภาพสูงสุด ในปัจจุบันมีการปรับปรุงลักษณะ

ของน้ำพุให้มีความแปลกใหม่และสวยงามมากขึ้นโดยอาจจะมีการทำให้เป็นเส้นของสายน้ำที่มีขนาดเล็กแต่มีจำนวนสายที่มากและมีการปรับระดับของความสูงโดยปรับขนาดของรูที่สายน้ำต้องพุ่งออกไม่เท่ากันเท่าให้ในหนึ่งหัวของน้ำพุมีสายน้ำหลายสายที่มีความสูงไม่เท่ากันเกิดความสวยงานและแปลกตามากขึ้นแต่การพัฒนาไม่ได้หยุดเพียงแค่นั้นได้มีการพัฒนารูปแบบของน้ำพุที่แปลกออกไปโดยทำให้น้ำพุมีการพุ่งของสายน้ำที่ไม่พร้อมกันแต่ในการพุ่งจะมีรูปแบบจนสามารถใช้น้ำพุในการสร้างเป็นภาพต่างได้อีกทั้งยังมีการทำให้น้ำพุสามารถทำงานตามจังหวะของเสียงดนตรีดังเช่นในงาน “พืชสวนโลกเฉลิมพระเกียรติ” ที่ผ่านมาได้มีการใช้น้ำพุในการแสดงน้ำพุดนตรีโดยสายน้ำที่พุ่งออกมาจะเป็นไปตามจังหวะของเสียงดนตรีและยังมีการฉายภาพยนตร์ลงบนม่านน้ำโดยม่านน้ำนี้มีลักษณะการทำงานเหมือนน้ำพุ ทำให้เป็นที่สนใจของผู้ที่มาเยี่ยมชมอย่างมาก

รูปที่ 1.2 น้ำพุกับงานแสงสี

โครงงานนี้จะเป็นน้ำพุที่มีขนาดเล็กเหมาะสมกับการตกแต่งสวนขนาดเล็กที่มีขนาดพื้นที่ไม่มากนักในโครงงานนี้จะเป็นการประยุกต์ใช้งานไมโครคอนโทรเลอร์ในงานควบคุมการเปิดปิดวาล์วน้ำที่หัวของน้ำพุแต่ละหัวโดยใช้เซอร์โวมอเตอร์หนึ่งตัวต่อหนึ่งหัวน้ำพุแต่ที่การใช้โซลินอยด์วาล์วที่มีราคาสูงกว่าและยังได้มีการปรับแต่งให้วาล์วน้ำที่หัวของน้ำพุมีการหมุนเปิดปิดที่คลองตัวเพื่อลดกำลังของเซอร์โวมอเตอร์ที่ต้องขับ

1.1 วัตถุประสงค์

1. เพื่อศึกษาหลักการและประยุกต์ใช้งานเซอร์โวมอเตอร์และไมโครคอนโทรลเลอร์
2. เพื่อศึกษาหลักการและประยุกต์ใช้งานการสื่อสารแบบอนุกรมระหว่างอุปกรณ์อิเล็กทรอนิกส์กับคอมพิวเตอร์
3. เพื่อศึกษาวิธีการใช้งานโปรแกรมซี พลัส พลัส บิลเดอร์และโปรแกรมซีคอมไพล์เลอร์
4. เพื่อศึกษาและประยุกต์ให้งานไมโครคอนโทรเลอร์ PIC

1.2 ขอบเขตของโครงงาน

น้ำพุดนตรีจะใช้เซอร์โวมอเตอร์ในการควบคุมการเปิดและปิดของหัวน้ำพุจำนวน 8 หัว อีกทั้งยังควบคุมระดับความสูงของน้ำพุแต่ละหัวโดยการควบคุมจังหวะการเปิดและปิดจะถูกควบคุมโดยไมโครคอนโทรเลอร์PIC16F877A รูปแบบการเปิดและปิดจะถูกเก็บไว้ในEEpromซึ่งจะกำหนดได้โดยผู้ใช้งานควบคุมผ่านโปรแกรมบนเครื่องคอมพิวเตอร์

1.3 ประโยชน์ที่จะได้รับจากโครงงาน

ประโยชน์ที่จะได้รับจากโครงงานนี้คือการได้นำความรู้ที่ได้ศึกษามาประยุกต์ใช้งานในการออกแบบวงจรและใช้ทักษะในการเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครคอมโทรเลอร์ อีกทั้งยังสามารถเขียนโปรแกรม ซี พลัส พลัส บิลเดอร์ เพื่อติดต่อกับผู้ใช้งานนอกจากนี้ยังเสริมทักษะในการแก้ปัญหาซึ่งจะสามารถนำไปพัฒนาในงานต่างๆได้ในอนาคต

1.4 รายละเอียดโดยย่อของโครงงาน

น้ำพุดนตรีประกอบไปด้วยส่วนควบคุมหัวน้ำพุซึ่งจะใช้วาล์วน้ำแบบบอลวาล์วทำหน้าที่เป็นประตูน้ำและใช้เซอร์โวมอเตอร์เป็นตัวส่งกำลังหมุนวาล์วโดยที่ตัวของบอลวาล์วจะถูกปรับแต่งให้มีความคล่องตัวมากกว่าปรกติส่วนของตัวเก็บโปรแกรมและประมวลผลจะถูกเก็บข้อมูลของน้ำพุไว้ในPIC16F877A โดยในขั้นต้นนี้น้ำพุจะทำงานตามที่ได้โปรแกรมไว

1.5 รายละเอียดของปริญญานิพนธ์

โครงงานน้ำพุดนตรีประกอบไปด้วยระบบต่างๆที่ต้องประมวลความรู้ที่ได้เรียนมาโดยมีเนื้อหาต่างๆดังนี้

บทที่ 1 บทนำ กล่าวถึงขอบเขตและรายละเอียดโดยย่อของโครงงาน

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีที่เกี่ยวข้องหลักการพื้นฐานต่างๆที่ใช้ในการออกแบบ

บทที่ 3 การออกแบบ กล่าวถึงอุปกรณ์ที่ใช้ในการสร้างและออกแบบรวมถึงวิธีการในการออกแบบ

บทที่ 4 ผลการทดลอง กล่าวถึงผลที่ได้จากการทดลองใช้งานในรูปแบบต่างๆ

บทที่ 5 สรุปผลการทดลอง กล่าวถึงปัญหาที่เกิดขึ้นจากการทดลองแนวทางและวิธีการแก้ไขในอนาคต

บทที่ 2

ทฤษฎีและหลักการ

2.1 ไมโครคอนโทรลเลอร์ (Microcontroller)

มาจากคำสองคำ คำหนึ่งคือ ไมโคร (Micro) หมายถึงขนาดเล็ก และคำว่า คอนโทรลเลอร์ หมายถึง ตัวควบคุมหรืออุปกรณ์ควบคุม ดังนั้นไมโครคอนโทรลเลอร์ จึงหมายถึง อุปกรณ์ควบคุมขนาดเล็ก แต่ในตัวอุปกรณ์ควบคุมขนาดเล็กนี้ได้บรรจุความสามารถที่คล้ายคลึงกับคอมพิวเตอร์ที่คนส่วนใหญ่คุ้นเคย กล่าวคือ ภายในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู (CPU), หน่วยความจำ และพอร์ต (port) ซึ่งเป็นส่วนประกอบสำคัญของระบบคอมพิวเตอร์ไว้ด้วยกัน โดยรวมกันอยู่ภายในตัวถังเดียวกัน

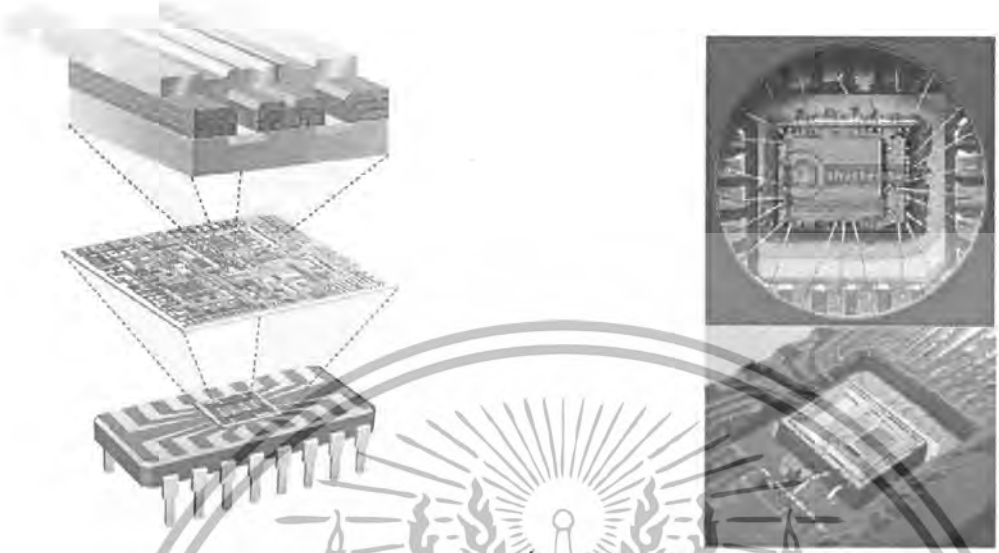
ซีพียูจะติดต่อกับหน่วยความจำโปรแกรมเพื่ออ่านคำสั่งที่ระบุไว้ โดยต้องทำการอ้างอิงตำแหน่งของหน่วยความจำผ่านสายสัญญาณที่เรียกว่า แอดเดรสบัส(address bus)แล้วทำการอ่านข้อมูลคำสั่งออกมาจากหน่วยความจำโปรแกรมในแอดเดรสนั้นๆจากนั้นทำการประมวลผลโดยมีหน่วยความจำข้อมูลแรมเป็นที่พักข้อมูลที่อยู่ในระหว่างการประมวลผลหรืออาจมองว่าหน่วยความจำข้อมูลแรมเป็นเหมือนกระดาษทดในการคำนวณก็ได้ ข้อมูลในการประมวลผลจะส่งผ่านสายสัญญาณที่เรียกว่า บัสข้อมูล(data bus) แล้วส่งต่อไปยังอุปกรณ์ภายนอกผ่านทางพอร์ตอินพุต(input)เอาท์พุต(output)

รูปที่ 2.1 ไมโครคอนโทรลเลอร์

2.1.1 หน่วยประมวลผลกลางหรือซีพียู (CPU: Central Processing Unit)

เป็นเหมือนมันสมองของไมโครคอนโทรลเลอร์โดนซีพียูนี้ทำหน้าที่ประมวลผลข้อมูลที่เข้ามาในระบบ แล้วทำการส่งต่อไปยังส่วนต่างๆเพื่อควบคุมการทำงานต่อไป หัวใจหลักของซีพียูคือหน่วยคำนวณทางคณิตศาสตร์และลอจิก(ALU : Arithmetic and logic unit) ซึ่งได้รับการกำหนดจังหวะการทำงานจากส่วนควบคุมลำดับการทำงาน โดยจังหวะการทำงานนั้นจะสัมพันธ์กับสัญญาณนาฬิกาเมื่อซีพียูทำการติดต่อกับหน่วยความจำสิ่งที่ปรากฏขึ้นบนบัสข้อมูลภายในซีพียูคือรหัสคำสั่ง(instruction code)ต้องผ่านการทำงานของส่วนถอดรหัสคำสั่ง(instruction decoder) เสียก่อนจะได้เป็นข้อมูลคำสั่งที่ซีพียูเข้าใจได้และสามารถดำเนินการต่อได้หลังจากที่หน่วยคำนวณทางคณิตศาสตร์และลอจิกประมวลผลแล้ก็จะส่งข้อมูลมายังส่วนเชื่อมต่อรีจิสเตอร์ภายในซีพียูเพื่อติดต่อกับส่วนอื่นๆต่อไป

การทำงานของซีพียูมีด้วยกันสองจังหวะคือ เฟตช์(fetch) และ เอ็กซิคิวต์(executed)โดยการทำงานจะเริ่มต้นจากการเฟตช์ ซึ่งก็คือการเรียกหรือการเข้าถึงคำสั่ง แล้วทำการถอดรหัสเป็นภาษาเครื่องเพื่อเตรียมประมวลผลจากนั้นจะเป็นจังหวะของการเอ็กซิคิวต์ซึ่งก็คือการกระทำตามคำสั่งที่กำหนดให้เสร็จสิ้นการที่จะระบุว่าไมโครคอนโทรลเลอร์มีขีดความสามารถในการประมวลผลเป็นอย่างไร จะพิจารณาที่ความสามารถในการประมวลผลของซีพียู หากซีพียูสามารถประมวลผลข้อมูลได้สูงสุด 8 บิต นั่นคือไมโครคอนโทรลเลอร์นี้เป็นแบบ 8 บิตแต่ในซีพียูในไมโครคอนโทรลเลอร์สมัยใหม่บางตัวมีขนาด 8 บิตแต่สามารถประมวลผลกับข้อมูล 16 บิตได้ ทำให้บางครั้งผู้ผลิตจึงระบุออกมาว่า ไมโครคอนโทรลเลอร์ตัวนี้ทำงานแบบ 16 บิตอาจกล่าวได้ว่าเป็นไมโครคอนโทรลเลอร์ 16 บิตเทียม เพราะถ้าหากเป็นแบบ 16 บิตแท้ ซีพียูต้องรองรับข้อมูลได้เต็ม 16 บิต หรือถ้าอ่านในดาต้าชีตของไมโครคอนโทรลเลอร์ตระกูลนั้นๆจะต้องระบุว่าเป็น 16–bit core ดังนั้นจึงต้องพิจารณารายละเอียดในส่วนนี้ด้วย



รูปที่ 2.2 ซีพียู

2.1.2 หน่วยความจำ

ในไมโครคอนโทรลเลอร์จะประกอบด้วยหน่วยความจำ 3 แบบคือ หน่วยความจำโปรแกรม (program memory), หน่วยความจำข้อมูลแรม (RAM data memory) และหน่วยความจำข้อมูลอีอีพรอม (EEPROM data memory)

2.1.3 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมเป็นที่สำหรับเก็บข้อมูลคำสั่งของโปรแกรมควบคุมที่ผู้พัฒนาเขียนขึ้นหรือเรียกว่า โปรแกรมมอนิเตอร์(monitor program)ซีพียูจะเข้ามาติดต่อเพื่ออ่านข้อมูลรหัสคำสั่งจากหน่วยความจำในส่วนนี้แล้วไปประมวลผลเพื่อควบคุมการทำงานของระบบทั้งหมดต่อไปเรียกได้ว่า มีความสำคัญเท่าๆกับซีพียูเลยทีเดียวหน่วยความจำโปรแกรมนี้มักมีขนาดใหญ่และถ้ายิ่งมีขนาดใหญ่มากเท่าใด ก็จะสามารถบรรจุโปรแกรมที่มีความซับซ้อนหรือสามารถเก็บตารางข้อมูลที่ใช้ในการประมวลผลได้มากตาม โดยทั่วไปมีความจุไม่น้อยกว่า 512 ไบต์ แต่จะให้ดีควรมีความจุ 1 กิโลไบต์ขึ้นไปจึงจะช่วยในการเขียนโปรแกรมควบคุมอิสระเพิ่มมากขึ้น ขนาดของหน่วยความจำโปรแกรมจะแปรตามความก้าวหน้าของเทคโนโลยีมีกรพัฒนาให้ไมโครคอนโทรลเลอร์มีความจุของหน่วยความจำโปรแกรมสูงขึ้นเรื่อยๆเป็น 4, 8, 16, 32 และ 64 กิโลไบต์และยังไม่สิ้นสุดเท่านี้ เชื่อแน่ว่าต้องมีการพัฒนาไมโครคอนโทรลเลอร์ให้มีความจุของหน่วยความจำโปรแกรมสูงเป็นหลักร้อยกิโลไบต์หรือหลักเมกะไบต์ชนิดของหน่วยความจำโปรแกรมที่ใช้ในไมโครคอนโทรลเลอร์มีอยู่ 3 แบบที่นิยมกันคือ แบบอีพรอม (EPROM: Erasable Programmable

Read-Only Memory), แบบอีอีพรอม (Electrically Erasable Read-Only Memory) และแบบแฟลช (Flash) ความแตกต่างอยู่ที่จำนวนครั้งในการลบและเขียนข้อมูลทับลงไปใหม่โดยสามารถสรุปได้ดังนี้

2.1.3.1 แบบอีพรอม ยังแบ่งเป็น 2 แบบคือ แบบโปรแกรมได้หลายครั้งและแบบโปรแกรมได้ครั้งเดียวถ้าหากเป็นแบบโปรแกรมได้หลายครั้งบนตัวถังของไมโครคอนโทรลเลอร์จะมีหน้าต่างกระจกติออยู่สมารถมองเห็นชิปภายในได้เวลาลบต้องลบด้วยแสงอัลตร้าไวโอเลตจำนวนในการโปรแกรมใหม่อยู่ระหว่าง 10-100 ครั้งแต่ถ้าเป็นแบบโปรแกรมได้ครั้งเดียวหรือ OTP (One-time programmable) จะไม่สามารถลบได้ตัวถังจะถูกปิดมิดชิดเหมือไอซีธรรมดา

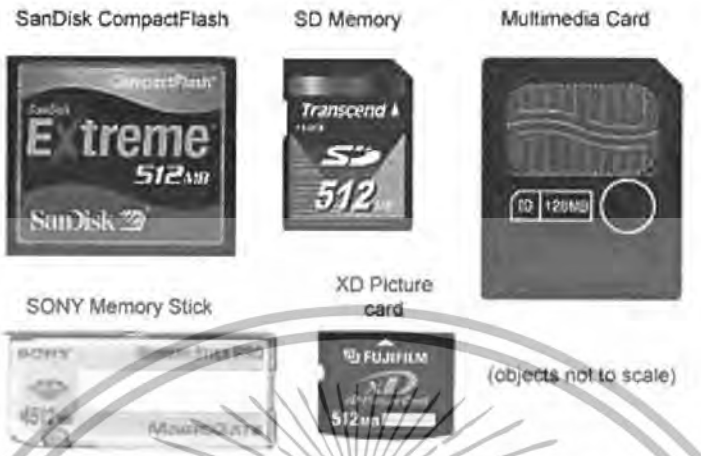
รูปที่ 2.3 หน่วยความจำแบบอีพรอม

2.1.3.2 แบบอีอีพรอม หน่วยความจำแบบนี้จะลบและเขียนใหม่ได้สนสัญญาณไฟฟ้า ในอดีตเป็นที่นิยมมากเนื่องจากสามารถลบและเขียนได้เป็นร้อยรอบขึ้นไปในบางตระกูลถึง 1 ล้านครั้ง แต่ในปัจจุบันแบบนี้ไม่เป็นที่นิยมใช้ในไมโครคอนโทรลเลอร์เนื่องจากต้นทุนสูง

รูปที่ 2.4 หน่วยความจำแบบอีอีพรอม

2.1.3.3 แบบแฟลช หน่วยความจำโปรแกรมชนิดนี้สามารถลบและเขียนได้ด้วยสัญญาณไฟฟ้าแตกต่างกับแบบอีอีพรอมในเชิงการใช้งานตรงที่กระบวนการลบข้อมูลหน่วยความจำโปรแกรมแบบแฟลชจะไม่สามารถเลือกลบได้เฉพาะเจาะจงบางแอดเดรสบางตำแหน่งได้เมื่อทำการลบข้อมูลจะต้องลบทั้งหมดหน่วยความจำโปรแกรมแบบนี้ได้รับความนิยมมากเนื่องจากราคาไม่สูงและสามารถโปรแกรมได้เป็นร้อยครั้งขึ้นไป แต่โดยปกติมักเริ่มที่1,000ครั้งในบางรุ่นสูงเป็นหมื่นครั้งและเป็นแสนครั้งขึ้นอยู่กับแรงดันที่ใช้ในการโปรแกรม

ขนาดข้อมูลของหน่วยความจำโปรแกรมขึ้นอยู่กับผู้ผลิตไมโครคอนโทรลเลอร์ยกตัวอย่างเช่นไมโครคอนโทรลเลอร์ตระกูล MCS-51 ขนาดของหน่วยความจำโปรแกรมเป็น 8 บิต ถ้าเป็นขนาด 16 บิต แต่ขนาดของหน่วยความจำโปรแกรมไม่ได้เป็นตัวระบุความสามารถในการประมวลผลของไมโครคอนโทรลเลอร์ตัวอย่างเช่น PIC กับ AVR ต่างมีขนาดของหน่วยความจำโปรแกรมสูงกว่า 8 บิต แต่ทั้งคู่ต่างเป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต ทั้งนี้เพราะซีพียูเป็นแบบ 8 บิต ขนาดของหน่วยความจำจะส่งผลต่อการระบุความจุของหน่วยความจำโปรแกรมภายในไมโครคอนโทรลเลอร์เบอร์นั้นๆตัวอย่างเช่น PIC16F628 มีหน่วยความจำโปรแกรมความจุ 2K×14 บิต หมายความว่า PIC16F628 มีหน่วยความจำขนาด 14 บิตอยู่ทั้งสิ้น 2,048 ตำแหน่งหรือเรียกว่ามีความจุ 2 กิโลเวิร์ด ทั้งนี้เนื่องจากขนาดหน่วยความจำโปรแกรมสูงกว่า 8 บิต จึงเรียกว่า ไบต์ไม่ได้ ขนาดของข้อมูลที่มากกว่า 8 บิต มักถูกเรียกว่าเวิร์ดสำหรับไมโครคอนโทรลเลอร์ PIC16F628 ขนาดของข้อมูล 1 เวิร์ดคือ 14 บิตแต่ถ้าเป็นในตระกูล MCS-51 จะสามารถระบุเป็นหน่วยไบต์ตรงๆ AVR จะเหมือนกับ PIC คือต้องระบุเป็นเวิร์ด



รูปที่ 2.5 หน่วยความจำแบบแฟลช

2.1.4 หน่วยความจำข้อมูลแรม

เป็นหน่วยความจำที่ต้องมีในไมโครคอนโทรลเลอร์ทุกตัวเพราะใช้เป็นพื้นที่สำหรับเก็บข้อมูลทั้งในระหว่างและหลังจากการประมวลผล ยิ่งมีมากยิ่งช่วยให้การทำงานสะดวกเพราะหน่วยความจำแรมมีอัตราเร็วในการอ่านเขียนสูงมากและไม่จำกัดจำนวนรอบในการอ่านเขียนในพื้นที่ของหน่วยความจำข้อมูลแรมจะแบ่งออกเป็น 2 ส่วนคือ ส่วนของข้อมูลทั่วไปสำหรับเก็บค่าตัวแปรและส่วนของรีจิสเตอร์

โดยปกติแล้วหน่วยความจำข้อมูลแรมจะมีความจุไม่มากเมื่อเทียบกับหน่วยความจำโปรแกรมในบางตัวอยู่ระดับสิบไบต์ แต่ถ้าไมโครคอนโทรลเลอร์มีความสามารถสูงขึ้นความจุของหน่วยความจำข้อมูลแรมจะเพิ่มมากขึ้นตามไปด้วยทั้งนี้เพราะต้องเพิ่มในส่วนของรีจิสเตอร์ตามความสามารถที่สูงขึ้นของไมโครคอนโทรลเลอร์

ทางด้านขนาดของหน่วยความจำข้อมูลแรม โดยส่วนใหญ่จะมีขนาด 8 บิต แต่สามารถต่อร่วมกันเป็น 16 บิตได้ ส่วนการจัดสรรตำแหน่งแอดเดรสของหน่วยความจำข้อมูลจะขึ้นอยู่กับสถาปัตยกรรมของไมโครคอนโทรลเลอร์หากเป็นแบบฮาร์วาร์ด (Harvard) จะได้รับการจัดสรรให้อยู่แยกจากหน่วยความจำโปรแกรมจึงทำให้มีแอดเดรสที่เหมือนกันได้นั่นคือผู้ใช้งานจะพบแอดเดรสเดียวกันทั้งหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล แต่จริงๆแล้วอยู่ต่างที่กันจะพบในไมโครคอนโทรลเลอร์ MCS-51, PIC, AVR เป็นต้น แต่ถ้าแบบพรินซ์ตัน (Princeton) จะจัดสรรให้อยู่ในบริเวณเดียวกันดังนั้นค่าแอดเดรสจะไม่มีทางตรงกันจะพบในไมโครคอนโทรลเลอร์ 68HCxxx ของ Motorola

รูปที่ 2.6 หน่วยความจำแบบแรม

2.1.5 หน่วยความจำข้อมูลอีอีพรอม

เป็นหน่วยความจำข้อมูลพิเศษที่ไมโครคอนโทรลเลอร์บางเบอร์ บางรุ่น บางตระกูลไม่มีใช้ สำหรับเก็บข้อมูลที่ต้องการรักษาไว้เมื่อไม่มีการจ่ายไฟเลี้ยงให้แก่ไมโครคอนโทรลเลอร์การติดต่อเพื่ออ่านเขียนจะมีลักษณะเป็นพิเศษขึ้นอยู่กับไมโครคอนโทรลเลอร์แต่ละเบอร์ขนาดของหน่วยความจำแบบนี้มักเท่ากับ 8 บิต ส่วนความจุที่แตกต่างกันไปมีตั้งแต่ไม่กี่สิบไบต์จนถึงเป็นกิโลไบต์

การอ่านเขียนหน่วยความจำแบบนี้จะใช้สัญญาณไฟฟ้าทั้งหมดและสามารถรักษาข้อมูลล่าสุดไว้แม้ว่าจะไม่มีการจ่ายไฟเลี้ยงให้แก่ไมโครคอนโทรลเลอร์แล้วก็ตามสำหรับจำนวนรอบในการเขียนโดยปกติอยู่ในหลักล้านครั้งขึ้นไป

2.1.6 รีจิสเตอร์ (Register)

เป็นหน่วยความจำพิเศษที่มีบทบาทสูงมากในการทำงานของไมโครคอนโทรลเลอร์ สามารถที่จะอ่านและเขียนข้อมูลได้ตลอดเวลาจนกว่าจะหยุดจ่ายไฟเลี้ยงให้แก่ไมโครคอนโทรลเลอร์หน้าที่หลักคือใช้เก็บข้อมูลในการทำงานของไมโครคอนโทรลเลอร์โดยข้อมูลที่เก็บนี้มีทั้งข้อมูลแสดงสถานะการทำงานข้อมูลสำหรับควบคุมการทำงานโมดูลย่อยต่างๆ ภายในไมโครคอนโทรลเลอร์ข้อมูลที่รับเข้ามาจากพอร์ตอินพุตข้อมูลที่ต้องการส่งออกไปยังอุปกรณ์ภายนอกผ่านพอร์ตเอาต์พุตโดยข้อมูลแต่ละประเภทก็จะถูกจัดเก็บลงในรีจิสเตอร์ที่แตกต่างกันตามหน้าที่การทำงาน

หน่วยความจำที่นำมาใช้เป็นรีจิสเตอร์มีด้วยกัน 2 ลักษณะขึ้นอยู่กับสถาปัตยกรรมของไมโครคอนโทรลเลอร์หากเป็นแบบพรินซ์ตันรีจิสเตอร์จะมีอยู่ด้วยกัน 2 ส่วน ส่วนแรกจะอยู่ร่วมกับซีพียูหรือเรียกว่า รีจิสเตอร์ซีพียู ส่วนที่สองจะอยู่แยกต่างหากซึ่งมักเป็นรีจิสเตอร์ควบคุมพอร์ตอินพุตเอาท์พุตและรีจิสเตอร์แสดงสถานะแต่ในสถาปัตยกรรมแบบฮาร์วาร์ดจะใช้บางส่วนในหน่วยความจำข้อมูลแรมภายในไมโครคอนโทรลเลอร์อย่างไรก็ตามข้อมูลในรีจิสเตอร์จะคงอยู่ตราบเท่าที่ยังจ่ายไฟเลี้ยงให้แก่ไมโครคอนโทรลเลอร์ซีพียูสามารถอ่านเขียนรีจิสเตอร์ได้ตลอดเวลาเท่ากับอายุการใช้งานของไมโครคอนโทรลเลอร์อาจกล่าวได้ว่ารีจิสเตอร์คือหน่วยความจำข้อมูลที่มีการระบุชื่อชัดเจนมีแอดเดรสและฟังก์ชั่นการทำงานที่เฉพาะเจาะจงตามที่กำหนดโดยผู้ผลิตไมโครคอนโทรลเลอร์

2.1.6.1 รีจิสเตอร์ตัวนับโปรแกรมหรือโปรแกรมเคาน์เตอร์ (PC)

การที่ซีพียูสามารถติดต่อกับหน่วยความจำโปรแกรมเพื่ออ่านข้อมูลคำสั่งได้อย่างถูกต้องเป็นผลมาจากรีจิสเตอร์หน้าที่พิเศษตัวหนึ่งคือ รีจิสเตอร์ตัวนับโปรแกรม หรือ PC (Program Counter) โดย PC จะเป็นตัวชี้ตำแหน่งแอดเดรสของหน่วยความจำโปรแกรมที่ซีพียูต้องไปกระทำในลำดับถัดไป โดยปกติแล้วค่าของ PC จะเปลี่ยนแปลงโดยอัตโนมัติขึ้นอยู่กับผลการทำงานที่เกิดขึ้นในไมโครคอนโทรลเลอร์บางตระกูลสามารถเข้าถึงรีจิสเตอร์ PC เพื่อทำการอ่านเขียนได้ในบางตระกูลก็ไม่สามารถทำได้

ขนาดของรีจิสเตอร์ PC ขึ้นอยู่กับความจุของหน่วยความจำโปรแกรมภายในไมโครคอนโทรลเลอร์นั้นเช่น MCS-51 สามารถมีหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์หรือ 65,536 ตำแหน่ง ขนาดของรีจิสเตอร์ PC จึงมีได้เท่ากับ 16 บิต ส่วนในไมโครคอนโทรลเลอร์ PIC อนุกรม 14 บิต มีหน่วยความจำโปรแกรมได้สูงสุด 8 กิโลเวิร์ด หรือ 8,192 ตำแหน่ง รีจิสเตอร์ PC จึงมีขนาด 13 บิต เป็นต้น

2.1.7 สแต็กในไมโครคอนโทรลเลอร์

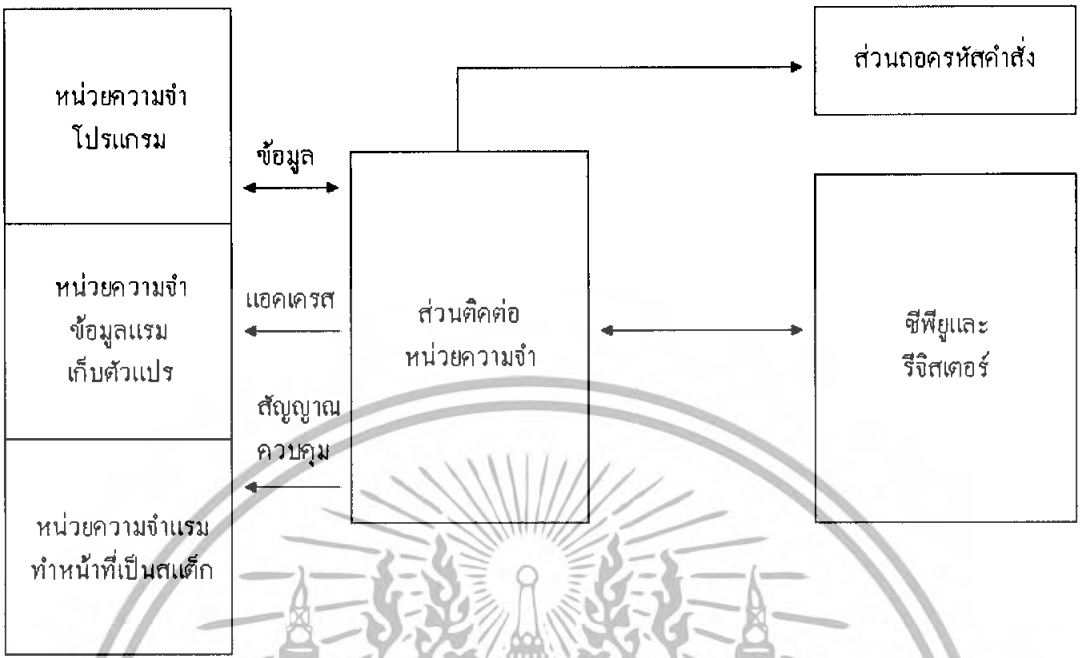
สแต็ก(stack)เป็นหน่วยความจำส่วนพิเศษที่ไมโครคอนโทรลเลอร์ทุกตัวต้องมีโดยหน้าที่ของมันคือ เก็บข้อมูลที่ยังต้องการอยู่ของรีจิสเตอร์และเมื่อข้อมูลนั้นถูกนำมาเก็บไว้ในสแต็กแล้วก็สามารถที่จะเปลี่ยนข้อมูลในรีจิสเตอร์ตัวนั้นๆได้ทันทีหลังจากที่ทำงานเรียบร้อยจึงกลับมาอ่านข้อมูลเดิมกลับจากสแต็กการเก็บข้อมูลของสแต็กจะมีลักษณะเป็นระดับหรือเป็นชั้น ข้อมูลที่เก็บเข้ามาก่อนจะต้องอ่านออกทีหลังเป็นแบบ FILO(First In Last Out) และจำนวนระดับหรือจำนวนชั้นของสแต็กก็มีจำกัด

ในไมโครคอนโทรลเลอร์ส่วนใหญ่จะมีความจุของสแต็กไม่น้อยกว่า 8 ระดับ การที่ยิ่งมีขนาดของสแต็กมากหรือมีจำนวนระดับมาก ก็จะยิ่งช่วยให้การทำงานสะดวกขึ้น เพราะในการประมวลผลมีโอกาสที่ต้องพักข้อมูลในรีจิสเตอร์หลักที่สำคัญเพื่อนไปทำงานอื่นก่อน หลังจากนั้นจึงจะกลับมาทำงานต่อ โดยเฉพาะอย่างยิ่งกับงานที่มีการอินเตอร์รัปต์หรือขัดจังหวะซีพียูบ่อยๆ รวมถึงงานที่มีการกระโดไปทำงานที่โปรแกรมย่อยจำนวนมาก เพราะเมื่อต้องกระโดออกจากโปรแกรมหลักไปทำงานที่โปรแกรมย่อย ก็ต้องเก็บข้อมูลของรีจิสเตอร์หลักที่ทำงานค้างอยู่ลงในสแต็ก หลังจากนั้นกระโดดไปที่โปรแกรมย่อยที่มีความต้องการเขียนข้อมูลในรีจิสเตอร์ตัวเดียวกันนี้ หลังจากทำงานแล้วจึงกลับมาที่โปรแกรมหลัก แล้วอ่านค่าเดิมก่อนหน้านี้กลับมาทำงานต่อ ทว่าในงานบางลักษณะมีการกระโดดไปทำงานยังโปรแกรมย่อยซ้อนกัน 2-3 ชั้น ทำให้ต้องมีการเก็บข้อมูลไว้ในสแต็กมากขึ้น หากความจุของสแต็กมีน้อยก็จะไม่สามารถรองรับการทำงานในลักษณะนี้ได้

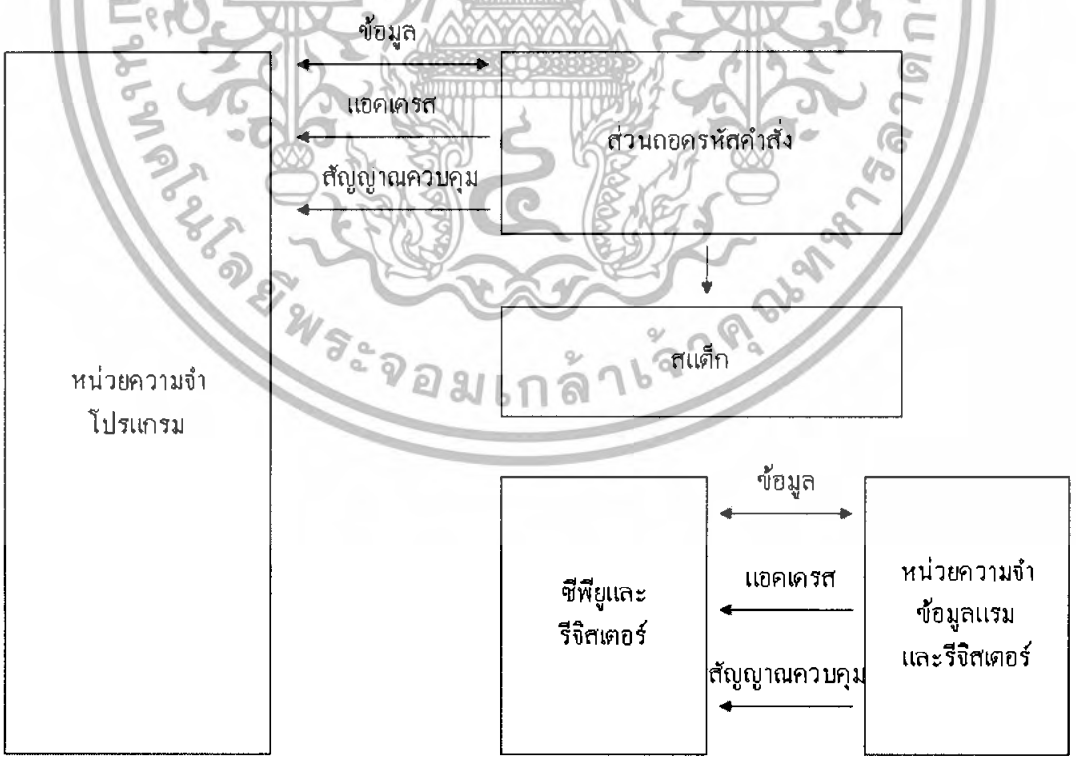
ขนาดของสแต็กโดยปกติจะต้องเท่ากับขนาดของรีจิสเตอร์ตัวนับโปรแกรมหรือ PC เพราะมีโอกาสที่จะต้องเก็บค่าของ PC ไว้ในสแต็กด้วย

2.2 สถาปัตยกรรมของไมโครคอนโทรลเลอร์

เป็นที่ยอมรับกันว่าสถาปัตยกรรมของไมโครคอนโทรลเลอร์มีด้วยกัน 2 แบบคือ **พรินซ์ตัน (Princeton)** หรือ **ฟอนนิวแมน (Von Neumann)** และ **ฮาร์วาร์ด (Harvard)** ในรูปที่ 4 และ 5 แสดงการจัดสรรหน่วยความจำและรีจิสเตอร์ในสถาปัตยกรรมของไมโครคอนโทรลเลอร์ทั้งสองแบบ พิจารณาในรูปที่ 4 ก่อนเป็นการจัดสรรในสถาปัตยกรรมแบบพรินซ์ตัน จะเห็นได้ว่า มีโครงสร้างที่เรียบง่ายไม่ซับซ้อน ส่วนของหน่วยความจำโปรแกรมกับหน่วยความจำข้อมูลจะได้รับการจัดสรรให้อยู่รวมกัน ติดติดต่อกับซีพียูส่วนจัดการเชื่อมต่อหน่วยความจำ และภายในซีพียูจะมีรีจิสเตอร์บรรจุอยู่ ข้อดีของสถาปัตยกรรมแบบนี้คือ ออกแบบง่าย เพราะหน่วยความจำทั้งหมดอยู่รวมกัน สามารถเข้าถึงได้ง่าย หน่วยความจำแรมหากมีขนาดใหญ่เพียงพอก็จะสามารถเก็บได้ทั้งโปรแกรมควบคุมการทำงานและข้อมูลของตัวแปรในการประมวลผล ข้อด้อยของสถาปัตยกรรมแบบนี้ก็คือ ความเร็วในการประมวลผล เนื่องจากหน่วยความจำอยู่รวมกัน จึงต้องติดต่อหน่วยความจำโปรแกรมสลับกับหน่วยความจำข้อมูล ส่งผลซีพียูต้องใช้จำนวนไซเกิลในการทำงานมาก แต่ข้อด้อยนี้สามารถชดเชยได้หากไมโครคอนโทรลเลอร์สามารถทำงานกับสัญญาณนาฬิกาที่มีความถี่สูงมากได้



รูปที่ 2.7 สถาปัตยกรรมแบบพรินซ์ตันหรือฟอนนิวแมน



รูปที่ 2.8 สถาปัตยกรรมแบบฮาร์วาร์ด

ในขณะที่สถาปัตยกรรมแบบฮาร์วาร์ด ซึ่งแสดงในรูปที่ 5 จะแยกส่วนของหน่วยความจำข้อมูลและรีจิสเตอร์ออกจากหน่วยความจำโปรแกรม ทำให้ไซเกิลการทำงานลดลง เนื่องจากสามารถติดต่อหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลได้เร็วกว่า นั่นคือทำงานได้เร็วกว่าแบบพรินซ์ตัน นอกจากนั้นในสถาปัตยกรรมแบบนี้ในขณะที่ซีพียูกำลังเอ็กซิคิวต์คำสั่งในปัจจุบันอยู่สามารถที่จะเฟตช์คำสั่งถัดไปได้ ยิ่งเป็นการเพิ่มความเร็วในการทำงานให้กับไมโครคอนโทรลเลอร์

2.3 การทำงานของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์จะสามารถทำงานได้เมื่อจ่ายไฟเลี้ยงและต่อวงจรกำเนิดสัญญาณนาฬิกาให้แก่มัน จากนั้นซีพียูภายในไมโครคอนโทรลเลอร์จะติดต่อกับหน่วยความจำโปรแกรมเพื่ออ่านข้อมูลคำสั่งแล้วทำงานตามคำสั่งที่บรรจุอยู่ในหน่วยความจำโปรแกรม

นั่นหมายความว่า ต้องทีการเขียนข้อมูลในหน่วยความจำโปรแกรมก่อน โดยไมโครคอนโทรลเลอร์แต่ละเบอร์จะมีรูปแบบของข้อมูลคำสั่งที่แตกต่างกัน ซึ่งจะต้องอาศัยกระบวนการเขียนโปรแกรม (programming) ภาษาที่ใช้ในการเขียนโปรแกรมสามารถแบ่งได้ 2 ระดับคือ ภาษาสูง (high language) และภาษาแอสเซมบลี (assembly language) โดยปกติไมโครคอนโทรลเลอร์ต้องการโปรแกรมที่เขียนด้วยภาษาแอสเซมบลี เนื่องจากสามารถทำงานได้รวดเร็วผ่านกระบวนการแปลงข้อมูลคำสั่งเป็นข้อมูลเลขฐานสิบหกเพื่อทำงานตามคำสั่งเพียง 1 ขั้นตอนคือ แปลงจากภาษาแอสเซมบลีเป็นข้อมูลเลขฐานสิบหก หรือที่เรียกว่า ออปโค้ด (Opcode) แต่ข้อเสียของการเขียนโปรแกรมด้วยภาษาแอสเซมบลีก็คือ ผู้เขียนต้องทำความเข้าใจในชุดคำสั่งของไมโครคอนโทรลเลอร์เบอร์นั้นๆ อย่างถ่องแท้ และเมื่อเปลี่ยนเบอร์ไมโครคอนโทรลเลอร์ก็ต้องทำหารเรียนรู้และทำความเข้าใจชุดคำสั่งใหม่ ซึ่งอาจทำให้เสียเวลามาก และการเขียนโปรแกรมด้วยภาษาแอสเซมบลี ผู้เขียนต้องมีทักษะในการเขียนโปรแกรมสูงพอสมควร และเข้าใจถึงสถาปัตยกรรมของไมโครคอนโทรลเลอร์เป็นอย่างดี

ในขณะที่การเขียนโปรแกรมด้วยภาษาสูง อาทิ ภาษาซี ภาษาเบสิก ต้องผ่านกระบวนการที่เรียกว่า คอมไพล์ (compile) เพื่อแปลงภาษาระดับสูงเหล่านั้นเป็นภาษาเครื่องหรือออปโค้ดของไมโครคอนโทรลเลอร์เบอร์นั้นๆ เสียก่อน และโปรแกรมที่ใช้ในการคอมไพล์นั้นเรียกว่า คอมไพเลอร์ (compiler) มักจะมีราคาแพง เมื่อใช้เครื่องมือทางซอฟต์แวร์ตัวนี้ ทำให้ผู้เขียนโปรแกรมอาจไม่จำเป็นต้องศึกษาสถาปัตยกรรมชุดคำสั่งของไมโครคอนโทรลเลอร์เบอร์นั้นๆ อย่างลึกซึ้งเท่ากับการเขียนโปรแกรมด้วยภาษาแอสเซมบลี ทั้งนี้เพราะคอมไพลเลอร์จะทำหน้าที่ในส่วนนี้แทน ดังนั้นเมื่อผู้ใช้งานเปลี่ยนเบอร์ไมโครคอนโทรลเลอร์ก็เพียงแต่จัดหาโปรแกรม

คอมไพเลอร์ที่เหมาะสมมาใช้งาน และศึกษาสถาปัตยกรรมของไมโครคอนโทรลเลอร์เบอร์ใหม่อีกเพียงเล็กน้อยสามารถใช้งานได้ แต่ข้อเสียของการใช้คอมไพเลอร์คือ ราคาแพงมาก

2.4 ภาษา C กับไมโครคอนโทรลเลอร์ PIC

ทำไมต้องเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ด้วยภาษา C ทั้งๆที่ภาษาแอสเซมบลีก็มีความสามารถสูงและใช้งานได้ดีกับไมโครคอนโทรลเลอร์อยู่แล้ว ถ้าเป็นอดีตคำตอบนี้จะเป็นคำตอบที่ถูกต้องที่สุดแต่ปัจจุบันมีทางเลือกที่ดีกว่าในการพัฒนาโปรแกรมสำหรับไมโครคอนโทรลเลอร์ด้วยภาษาแอสเซมบลีนั่นคือภาษาC ด้วยเหตุผลดังนี้

1. ภาษา C เป็นภาษามาตรฐานไม่ขึ้นกับฮาร์ดแวร์
2. ภาษา C ใช้หลักการเขียนโปรแกรมแบบสมัยใหม่มีทั้งเป็นโครงสร้างและออบเจ็กต์
3. ภาษา C มีความง่ายในการศึกษาและเรียนรู้และพัฒนาโปรแกรมมากกว่าภาษาแอสเซมบลี
4. ภาษา C มีความสามารถเทียบเท่าหรือใกล้เคียงภาษาแอสเซมบลีและอาจเหนือกว่าในบางด้าน
5. ภาษา C มีความยืดหยุ่นในการโยกย้ายไปใช้งานกับไมโครคอนโทรลเลอร์ตระกูลอื่น
6. ภาษา C มีความเข้ากันได้กับภาษาแอสเซมบลีในระดับซอร์สโค้ด
7. ภาษา C มีเครื่องมือในการพัฒนาโปรแกรมที่ก้าวหน้ากว่ามากด้วย IDE
8. ตัวแปลภาษาหรือคอมไพล์เลอร์และตัวเชื่อมโยงของภาษา C มีการพัฒนาอยู่ตลอดเวลาจนมีความสามารถสูง
9. สามารถพัฒนางานที่มีความซับซ้อนและมีรูปแบบเป็นผังงานง่ายกว่า

ทั้งหมดคือเหตุผลส่วนหนึ่งจากอีกหลายเหตุผลที่ทำให้ภาษา C เป็นตัวเลือกในการพัฒนาโปรแกรมสำหรับไมโครคอนโทรลเลอร์ในปัจจุบันการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ PIC ด้วยภาษา C จะมีรูปแบบและโครงสร้างทางภาษาเช่นเดียวกับภาษา C มาตรฐานและยังมีส่วนเพิ่มเติมพิเศษเฉพาะสำหรับใช้งานกับไมโครคอนโทรลเลอร์ PIC ซึ่งได้แก่ชุดคำสั่งพิเศษแลฟังก์ชั่นพร้อมใช้งานที่มีมากับตัวคอมไพล์เลอร์สำหรับการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ด้วยภาษา C

2.4.1โปรแกรมภาษา C

โปรแกรมภาษา C จะประกอบไปด้วย 3 ส่วนหลักๆคือ

1. ฟังก์ชั่น
2. ตัวแปร
3. ชุดคำสั่งการทำงาน

โดยฟังก์ชั่นจะมีอยู่ 2 ส่วนคือ ฟังก์ชั่นที่สร้างขึ้นมากับฟังก์ชั่นที่มีมาพร้อมกับตัวคอมไพล์เลอร์โดย ฟังก์ชั่นที่มากับตัวคอมไพล์เลอร์จะเรียกว่า Built-in function นอกจากนี้ใน CCS C คอมไพล์เลอร์ยังได้เพิ่มเติมฟังก์ชั่นสำเร็จรูปในรูปแบบโมดูลฟังก์ชั่นที่ CCS เขียนขึ้นมาเพื่ออำนวยความสะดวกให้กับผู้ใช้งานสามารถนำโค้ดไปใช้งานได้ทันที เช่น LCD, ระบบบัส 1 สาย, ระบบI2Cและอ่านเขียนหน่วยความจำข้อมูลอีอีพรอม รวมทั้งโค้ดโปรแกรมตัวอย่างใช้งานที่มีมากับ CCS C คอมไพล์เลอร์ที่ครอบคลุมการทำงานติดต่อกับอุปกรณ์ต่างๆ

2.4.2 CCS C คอมไพล์เลอร์

เป็นซอฟต์แวร์สำหรับแปลโปรแกรมภาษา C ของไมโครคอนโทรลเลอร์ PIC เป็นรหัสเครื่องหรือแมชชีนโค้ดผลิตโดย Custom Computer Service สหรัฐอเมริกา สามารถดูรายละเดียดผ่านทางเวปไซต์

CCS C คอมไพเลอร์มีหลายรุ่นเพื่อรองรับความต้องการของผู้ใช้งานการติดตั้งโปรแกรมทำได้ง่ายมากด้วยขั้นตอนที่เหมือนกับซอร์ฟแวร์ประยุกต์ทั่วไปที่ทำงานบนระบบวินโดวส์

CCS C คอมไพล์เลอร์แบ่งออกเป็น 2 รุ่นใหญ่ๆ คือ

1. รุ่นทำงานบนวินโดวส์
2. รุ่นบรรทัดคำสั่ง

ถ้าเป็นการใช้งานรุ่น IDE สามารถที่จะเขียนโค้ดโปรแกรมผ่าน Windows IDE ได้โดยตรงและสมารถที่จะคอมไพล์โปรแกรมได้ทันทีรวมทั้งมีคอมไพล์เลอร์แบบบรรทัดคำสั่งหรือ Command Line ให้ใช้งานด้วยแต่ถ้าเป็นรุ่นบรรทัดคำสั่งจะไม่มี Windows IDE ต้องจัดหาโปรแกรมเท็กซ์เอดิเตอร์มาใช้งานในการเขียนโค้ดโปรแกรมเอง ไม่ว่าจะเป็น Notepad ของวินโดวส์, Editor ของ MPLAB หรือจะเป็น Edit Plus ที่เป็นที่นิยมในบ้านเราซึ่งก็แล้วแต่ว่าความถนัดของผู้ใช้งานเป็นหลัก

2.4.3 พื้นฐานภาษา C กับ CCS C คอมไพล์เลอร์

โครงสร้างของภาษา C ในรูปแบบมาตรฐาน (ANSI Standard C) จะประกอบไปด้วยรายละเอียดดังนี้

- **พรีโปรเซสเซอร์ไดเร็กตีฟ (Preprocessor directives)**

ชุดคำสั่งที่ใช้ในการจัดการเตรียมข้อมูลสำหรับการประมวลผล โดยก่อนที่จะมีการคอมไพล์หรือแปลตัวโค้ดโปรแกรมให้เป็นภาษาเครื่องนั้นจะมาทำงานในส่วนนี้ก่อนจึงเรียกว่า พรีโปรเซสเซอร์สำหรับ CCS C คอมไพล์เลอร์ จะมีไดเร็กตีฟทั้งที่เป็นมาตรฐานเดียวกับ ANSI C และที่เพิ่มเติมเฉพาะสำหรับ CCS C คอมไพล์เลอร์

- **การประกาศ (Declarations)**

ก่อนใช้งานตัวแปรหรือฟังก์ชั่นต้องมีการประกาศและสร้างตัวแปรหรือฟังก์ชั่นขึ้นมาก่อน

- **การกำหนดค่า (Definitions)**

การประกาศและจองหน่วยความจำหรือกำหนดค่าให้กับตัวแปรหรือฟังก์ชั่น

- **นิพจน์ (Expressions)**

นิพจน์คือการกระทำระหว่างตัวดำเนินการ(Operators) กับตัวถูกดำเนินการหรือโอเปอร์แรนด์ (operands) เพื่อให้เกิดค่าใดค่าหนึ่ง

- **สเตตเมนต์ (Statements)**

คำสั่งการทำงานคือคำสั่งที่ใช้ในการทำงานตามความต้องการของผู้เขียนโปรแกรม

- **ฟังก์ชั่น (Functions)**

ฟังก์ชั่นคือส่วนประกอบของโปรแกรมที่กำหนดให้ทำงานอย่างใดอย่างหนึ่งจนเสร็จสิ้น โดยที่ฟังก์ชั่นจะประกอบไปด้วยการประกาศใช้งานตัวแปรการกำหนดค่าให้กับตัวแปร นิพจน์ และคำสั่งการทำงาน

- **ฟังก์ชั่น main() (Main function)**

เป็นฟังก์ชั่นที่จะต้องมีการประกาศทุกครั้งเมื่อมีการเขียนโปรแกรมด้วยภาษา C เพราะการทำงานของโปรแกรมจะเริ่มต้นที่ฟังก์ชั่นนี้และเป็นฟังก์ชั่นที่ใช้ในการเรียกฟังก์ชั่นอื่นๆในการทำงาน

2.4.4 ตัวแปรและชนิดข้อมูลในโปรแกรมภาษา C ของไมโครคอนโทรลเลอร์ PIC

นอกจากรายละเอียดโครงสร้างของภาษา C ที่ต้องทำความเข้าใจแล้วตัวแปรและชนิดข้อมูลที่ใช้ในการสร้างตัวแปร ก็เป็นส่วนหนึ่งที่สำคัญที่ต้องทำความเข้าใจ ก่อนที่จะเริ่มต้นเขียนโปรแกรมด้วยภาษา C โดยเฉพาะการเลือกใช้ชนิดของข้อมูล เนื่องจากการใช้งานชนิดของข้อมูลก็คือการเลือกใช้งานหน่วยความจำแรมของไมโครคอนโทรลเลอร์เช่นเดียวกันจึงต้องเลือกใช้ให้เหมาะสมเพื่อนำมาสร้างตัวแปรไม่ว่าจะเป็นทศนิยม, จำนวนเต็ม, ตัวอักขระ เพราะหน่วยความจำข้อมูลแรมของไมโครคอนโทรลเลอร์มีขีดจำกัด

2.4.5 ชนิดของข้อมูล

ในการสร้างตัวแปรขึ้นมาใช้งานจะต้องมีการระบุชนิดของข้อมูลให้กับตัวแปร โดยชนิดของข้อมูลก็จะแบ่งออกเป็น 2 ชนิดใหญ่คือ ชนิดข้อมูลจำนวนตัวเลขและชนิดข้อมูลตัวอักษร

ชนิดข้อมูล	ขนาด	ค่าของข้อมูล
Int1	ตัวเลข 1 บิต	0 หรือ 1
Int8	ตัวเลข 8 บิต	0 ถึง 255
Int16	ตัวเลข 16 บิต	0 ถึง 65,536
Int32	ตัวเลข 32 บิต	0 ถึง 4,294,967,295
char	ตัวอักขระ 8 บิต	ตัวอักขระรหัสแอสกี้
float	ตัวเลขทศนิยม 32 บิต	3.4×10^{-38} ถึง 3.4×10^{38}
short	ตัวเลข 1 บิต	0 หรือ 1
int	ตัวเลขจำนวนเต็ม 8 บิต	0 ถึง 255
long	ตัวเลข 16 บิต	0 ถึง 65,525
void	ไม่กำหนด	-

ตารางที่ 2.1 การกำหนดตัวแปร

2.4.6 ประเภทของการเก็บข้อมูลตัวแปร (Variable Storage Class)

ตัวแปรและฟังก์ชั่นที่ใช้งานในภาษา C จะมีรายละเอียดอยู่ 2 ส่วนคือ ชนิดของข้อมูล (Data type) และการเก็บข้อมูล (Storage class)

โดยชนิดของข้อมูลจะเป็นการระบุว่าตัวแปรที่สร้างขึ้นมานั้น มีชนิดข้อมูลเป็นอย่างไร เช่น Char, int, float เป็นต้น และอีกส่วนซึ่งโดยปกติแล้วจะมีค่าเริ่มต้นเป็น auto (นั่นคือสามารถปรับเปลี่ยนได้อัตโนมัติ) จึงไม่เขียนระบุไว้ตอนเริ่มต้นสร้างตัวแปร แต่เมื่อต้องการระบุการจัดเก็บและใช้งานหน่วยความจำว่าต้องการใช้งานบนหน่วยความจำโปรแกรมหรือบนรีจิส จึงต้องมีการระบุลงไปในตอนสร้างตัวแปรใช้งาน

คุณสมบัติ	รายละเอียด
Auto(automatic)	เป็นการระบุให้ตัวแปรเป็นประเภทหน่วยความจำอัตโนมัติและสามารถใช้ได้ภายในฟังก์ชั่น
Static	ตัวแปรที่ประกาศจะเป็นแบบทั่วไป(โกลบอล : global) มีค่าเริ่มต้นเท่ากับ 0 จะใช้ภายในหรือภายนอกฟังก์ชั่นก็ได้ เป็นตัวแปรแบบสถิต
Extern(external)	เป็นการบอกให้รู้ว่าตัวแปรที่ถูกประกาศมีการใช้งานแล้วในฟังก์ชั่นอื่น(นิยมใช้ในการประกาศข้ามไฟล์ของซอร์สโปรแกรม)
register	จะเหมือนกับการประกาศแบบ auto แต่จะใช้กับตัวแปรที่มีการใช้งานบ่อยๆ นิยมใช้กับตัวแปรที่เกี่ยวข้องกับการวนลูปเนื่องจากรียกใช้งานได้อย่างรวดเร็ว เพราะเป็นการประกาศในหน่วยความจำความเร็วสูง

ตารางที่ 2.2 คุณสมบัติของตัวแปร

2.4.6.1 ตัวแปร (Variable)

ตัวแปรคือชื่อที่ผู้พัฒนาโปรแกรมประกาศขึ้นมาเพื่อใช้เก็บข้อมูลชั่วคราว การประกาศใช้งานตัวแปรก็คือการประกาศใช้งานหน่วยความจำ การประกาศใช้งานตัวแปรต้องเลือกชนิดของข้อมูลให้เหมาะสมกับข้อมูลที่จะใช้และไม่ควรจะประกาศใช้งานตัวแปรมากเกินความจำเป็น เนื่องจากเมื่อมีการประกาศใช้งานตัวแปรหน่วยความจำแรมจะถูกจองไว้ใช้งานสำหรับตัวแปรนั้นทันที

ในการประกาศใช้งานตัวแปรในภาษา C จะต้องกำหนดชนิดของตัวแปรนั้นว่าเป็นชนิดใด เช่น ตัวแปรชนิด Integer, Character หรือ Floating เป็นต้น นอกจากนี้ยังใช้คีย์เวิร์ด typedef (ตั้งชื่อใหม่ใช้อ้างอิงแทนชื่อเก่า) เพื่อใช้ในการอ้างอิงถึงชนิดของข้อมูลที่กำหนดมา เช่น ถ้าต้องการนำค่าตัวเลข 2 ตัวมาบวกกับแล้วให้ผลลัพธ์ไปเก็บไว้จะต้องกระทำตามขั้นตอนดังนี้

1. สร้างตัวแปรขึ้นมารองรับ 3 ตัวสำหรับเก็บค่า 3 ค่า
2. นำค่าที่ต้องการบอกกันมาเก็บไว้ในตัวแปรทั้งสองตัวที่สร้างขึ้น
3. นำค่าตัวแปรทั้งสองตัวที่ได้เก็บค่าไว้มากระทำกันด้วยเครื่องหมายบวกแล้วเก็บไว้ในอีกตัวแปร ก็จะได้ค่า 2 ค่าบวกกันและสามารถนำไปใช้งานได้

2.4.6.2 กฎในการตั้งชื่อตัวแปร

ในการสร้างตัวแปรขึ้นมาใช้งานจะต้องมีกฎที่เกี่ยวข้องกับการตั้งชื่อ เพื่อป้องกันไม่ให้ไปซ้ำกับค่าที่ถูกกำหนดไว้ในตัวคอมไพเลอร์แล้วหรือสร้างชื่อตัวแปรที่คอมไพเลอร์ไม่สามารถตีความหมายได้กฎของการตั้งชื่อตัวแปรมีดังนี้

1. ไม่มีตัวอักษรพิเศษประกอบอยู่ในตัวแปร เช่น @, !, #, $, % เป็นต้น
2. ชื่อตัวแปรต้องไม่ขึ้นต้นด้วยตัวเลข
3. ไม่มีช่องว่างระหว่างชื่อของตัวแปร
4. ใช้อักษรตัวเล็กใหญ่ได้ แต่จะให้ความหมายเป็นคนละตัวแปร เช่น ตัวแปร Abc กับ abc เป็นคนละตัวแปรกันเนื่องจากภาษา C จะคำนึงถึง case sensitive
5. ห้ามตั้งชื่อตัวแปรซ้ำกับคำสงวนหรือคำสั่งที่มีในภาษา C และ Build-in function เช่น ไม่ตั้งชื่อตัวแปรดังนี้ Char, float, int, output_b, printf เป็นต้น

2.4.7 นิพจน์และตัวดำเนินการในโปรแกรมภาษา C ของไมโครคอนโทรลเลอร์ PIC

การเขียนโปรแกรมไม่ว่าจะเป็นภาษา C, BASIC, หรือPASCALจะต้องเกี่ยวข้องกับนิพจน์และตัวดำเนินการซึ่งจะอยู่ในรูปแบบของชุดคำสั่งการเขียนโปรแกรมจึงหนีไม่พ้นที่จะต้องทำความเข้าใจกับนิพจน์และตัวดำเนินการ โดยเฉพาะกับการเขียนโปรแกรมด้วยภาษา C แล้วการใช้งานนิพจน์และตัวดำเนินการใน CCS C คอมไพล์เลอร์ จะมีรูปแบบและลักษณะการใช้งานเช่นเดียวกับ ANSI C จึงสามารถศึกษารายละเอียดเพิ่มเติมได้จากหนังสือภาษา C ที่เป็น ANSI Standard C

2.4.7.1 นิพจน์ (Expression)

นิพจน์คือการนำตัวแปร ค่าคงที่ หรือตัวเลข ที่มากระทำกันด้วยเครื่องหมายดำเนินการ ด้วยเงื่อนไขใดเงื่อนไขหนึ่ง เช่น

```
A + B;                  // นำค่าในตัวแปร A และ B มาบวกกัน
A = =B;                 // นำค่าในตัวแปร A และ B มาเปรียบเทียบกันด้วยเครื่องหมายเท่ากับ
A + B / (A*B);          // นำค่าในตัวแปร B หารด้วย A คูณกับ B และนำค่าที่ได้ไปบวกกับ A
```

โดยการเขียนนิพจน์จะต้องมีเครื่องหมายตัวดำเนินการเข้ามาเกี่ยวข้องด้วยและมีระดับความสำคัญของการกระทำในตัวดำเนินการตามหลักระดับความสำคัญของตัวดำเนินการ

2.4.7.2 ตัวดำเนินการ (Operators)

เครื่องหมายดำเนินการคือ ตัวกระทำหรือเครื่องหมายการทำที่มีผลต่อข้อมูลหรือตัวแปร ในภาษา C จะแบ่งออกเป็น 5 ประเภทใหญ่คือ

1. เครื่องหมายดำเนินการทางคณิตศาสตร์ (Arithmetic & Unary Operators)
2. เครื่องหมายดำเนินการสัมพันธ์ทางลอจิก (Relational and Logical Operators)
3. เครื่องหมายดำเนินการกำหนดค่าหรือให้ค่า (Assignment Operators)
4. เครื่องหมายดำเนินการแก้ไข (Conditional Operator)
5. เครื่องหมายดำเนินการทางบิต (Bitwise Operator)

2.4.7.3 เครื่องหมายดำเนินการทางคณิตศาสตร์

ในภาษา C จะมีเครื่องหมายที่ใช้ดำเนินการเกี่ยวกับคณิตศาสตร์ดังนี้

เครื่องหมายดำเนินการ	การกระทำ	ตัวอย่าง
+	การบวกค่า	A+B
-	การลบค่า	A-B
*	การคูณค่า	A*B
/	การหารค่า	A/B
%	การหารค่าเอาเฉพาะเศษ	A%B
-	จำนวนลบ	-A
--	ลดค่า	--A,A--
++	เพิ่มค่า	++A,A++

ตารางที่ 2.3 เครื่องหมายดำเนินการทางคณิตศาสตร์

2.4.7.4 เครื่องหมายดำเนินการสัมพันธ์และลอจิก

เครื่องหมายดำเนินการในลักษณะนี้จะเกี่ยวข้องกับเรื่องของการเปรียบเทียบค่าเพื่อแสดงผลลัพธ์ของการเปรียบเทียบค่าว่าเป็นจริงหรือเป็นเท็จโดยที่เครื่องหมาย && , || และ ! จะเกี่ยวข้องกับตัวดำเนินการทางลอจิกดังมีรายการต่อไปนี้

เครื่องหมายดำเนินการ	การกระทำ	ตัวอย่าง
<	น้อยกว่า	A < B
<=	น้อยกว่าหรือเท่ากับ	A <= B

>	มากกว่า	A > B
>=	มากกว่าหรือท่ากับ	A >= B
= =	เท่ากับ	A = = B
!=	ไม่เท่ากับ	A != B
&&	และ	A && B
||	หรือ	A || B
!	ไม่ใช่	! A

ตารางที่ 2.4 เครื่องหมายดำเนินการสัมพันธ์และลอจิก

2.4.7.5 เครื่องหมายดำเนินการกำหนดค่าหรือให้ค่า

เครื่องหมายดำเนินการที่ใช้ในการกำหนดค่าหรือให้ค่าของข้อมูลหรือตัวแปรหรือการกระทำด้วยนิพจน์จากทางขวาของเครื่องหมาย = ให้กับตัวแปรที่อยู่ทางซ้ายเช่น

```
A = 20;          // กำหนดค่า 20 ให้กับตัวแปร A
A = A + B;       // นำค่าใน A บวกกับค่าใน B แล้วกำหนดให้กับ A
```

2.4.7.6 เครื่องหมายดำเนินการเงื่อนไข

เครื่องหมายดำเนินการเงื่อนไข ? : นี้เป็นเครื่องหมายดำเนินการเลือกอย่างใดอย่างหนึ่งโดยดูจากเงื่อนไขของการดำเนินการลอจิกตามตัวอย่าง

```
X = (A > B ) ? 0 : 150;    // ค่าของ X จะเท่ากับ 150
Y = (A < B ) ? 0 : 150;    // ค่าของ Y จะเท่ากับ 0
```

2.4.7.7 ความสัมพันธ์ของเครื่องหมายดำเนินการ

เครื่องหมายดำเนินการทั้งหมดมีการกำหนดระดับความสำคัญทั้งนี้เพื่อประโยชน์ในการลำดับความสำคัญในการทำงานในกรณีที่มีเครื่องหมายดำเนินการหลายตัวในเงื่อนไข ดังนี้

เครื่องหมายดำเนินการ	ระดับความสำคัญในการกระทำ
- , ++ , -- , ! , sizeof , (type)	ขวา ไป ซ้าย
*, / , %	ซ้าย ไป ขวา
+ , -	ซ้าย ไป ขวา
< , <= , > , >=	ซ้าย ไป ขวา
= = , !=	ซ้าย ไป ขวา

&&	ซ้าย ไป ขวา
||	ซ้าย ไป ขวา
? :	ขวา ไป ซ้าย
= , + = , - = , * = , / = , % =	ขวา ไป ซ้าย

ตารางที่ 2.5 ความสัมพันธ์ของเครื่องหมายดำเนินการ

2.4.7.8 เครื่องหมายดำเนินการทางบิต

เครื่องหมายดำเนินการทางบิตข้อมูลจัดเป็นเครื่องหมายที่มีความสำคัญมากโดยเฉพาะกับการเขียนโปรแกรมด้วยภาษา C กับงานไมโครคอนโทรลเลอร์ซึ่งหนีไม่พ้นเรื่องของบิตและไบต์ข้อมูลรายละเอียดของตัวดำเนินการมีดังนี้

เครื่องหมาย	การกระทำ	ตัวอย่าง
&	การแอนด์ข้อมูล	A & B
|	การออร์ข้อมูล	A | B
^	การเอ็กคลูซีพออร์ข้อมูล	A ^ B
<<	เลื่อนบิตไปทางซ้าย	A << B
>>	เลื่อนบิตไปทางขวา	A >> B
~	การกลับค่าบิต	~A

ตารางที่ 2.6 เครื่องหมายดำเนินการทางบิต

2.4.8 การใช้งานพอร์ตอนุกรม USARTในไมโครคอนโทรลเลอร์PICด้วยโปรแกรมภาษา C

พอร์ตอนุกรมจัดเป็นพอร์ตสื่อสารข้อมูลที่นิยมใช้งานทั้งบนเครื่องคอมพิวเตอร์และไมโครคอนโทรลเลอร์โดยเฉพาะนำมาใช้งานในการสื่อสารระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์ไมโครคอนโทรลเลอร์สำหรับไมโครคอนโทรลเลอร์ PIC หลายๆเบอร์ได้เตรียมโมดูลการเชื่อมต่อพอร์ตอนุกรมไว้ให้แล้วที่เรียกว่า USART (Universal Synchronous Asynchronous Receiver Transmitter) ที่มีคุณสมบัติในการทำงานเป็นตัวรับและตัวส่งข้อมูลในแบบอะซิงโครนัสหรือซิงโครนัสก็ได้ นอกจากนี้ยังสามารถกำหนดความเร็วในการรับส่งข้อมูลได้ด้วย

ใน CCS C คอมไพล์เลอร์ ได้เตรียมส่วนติดต่อและใช้งานพอร์ตอนุกรมไว้ให้แล้วทั้งในส่วนของการประกาศไดเร็กตีฟเพื่อเรียกใช้งานพอร์ตอนุกรมและฟังก์ชั่นพร้อมใช้งานเพื่อใช้ในการ

จัดการข้อมูลอนุกรมซึ่งมีการทำงานในลักษณะโพลลิ่ง คือ การเขียนโปรแกรมดักจับข้อมูลที่ผ่านเข้ามาทางพอร์ตอนุกรมตลอดเวลาในลูปของโปรแกรมกลับและแบบที่สองคือการอินเตอร์รัปต์ของพอร์ตอนุกรมโดยอาศัยฟังก์ชั่นที่เกี่ยวข้องกับการอินเตอร์รัปต์วิธีที่สองนี้เป็นการใช้อินเตอร์รัปต์ในการดักจับสัญญาณข้อมูลที่ผ่านเข้ามาทางพอร์ตอนุกรมแทนทำให้ไม่ต้องเขียนโปรแกรมจับข้อมูลทางพอร์ตตลอดเวลาเหมือนวิธีแรก

2.4.9 ไมโครคอนโทรลเลอร์ตระกูล PIC

PIC คือไมโครคอนโทรลเลอร์ตระกูลหนึ่ง ผลิตโดยบริษัทไมโครชิพโดย PIC ย่อมาจากคำว่า Peripheral Interface Controller ซึ่งภายใน PIC ประกอบด้วยหน่วยความจำโปรแกรม หน่วยความจำข้อมูล พอร์ตอินพุต พอร์ตเอาต์พุตทำให้ PIC เหมือนไมโครคอมพิวเตอร์ตัวหนึ่ง นอกจากนี้ภายใน PIC ยังมี I^2C, PWM, A/D ซึ่งถือว่าเป็นคุณสมบัติพิเศษของ PIC ที่แตกต่างจากไมโครคอนโทรลเลอร์ตัวอื่นๆการรวมทุกสิ่งทุกอย่างไว้ในตัวPICทำให้นำมาใช้งานได้ง่ายและสะดวก เพียงต่อไฟเลี้ยงป้อนสัญญาณนาฬิกาและเขียนโปรแกรมควบคุม PIC ก็สามารถควบคุมอุปกรณ์ภายนอกผ่านพอร์ตอินพุตและพอร์ตเอาต์พุตได้

2.4.9.1 ชนิดของไมโครคอนโทรลเลอร์ตระกูล PIC

ไมโครคอนโทรลเลอร์ตระกูล PIC ที่นิยมใช้กันในปัจจุบันมีด้วยกันหลายเบอร์โดยแยกเป็นกลุ่มได้ดังนี้ PIC10, PIC12, PIC14, PIC16, PIC17, PIC18 ซึ่งแต่ละกลุ่มยังแยกเป็นเบอร์ต่างอีกหลายเบอร์แต่กลุ่มที่ได้รับความนิยมมี 3 กลุ่มคือ PIC16, PIC17, PIC18 ทั้ง 3 กลุ่มนี้จะมีคุณสมบัติต่างกันออกไปเช่น ขนาดของหน่วยความจำ จำนวนพอร์ต สถาปัตยกรรมอาจแบ่งตามชนิดของหน่วยความจำโปรแกรมได้ 3 กลุ่ม คือ

1. มีหน่วยความจำโปรแกรมได้ครั้งเดียว
   หน่วยความจำโปรแกรมกลุ่มนี้เรียกว่า OTP (One Time Programmable) เป็นชิพที่ราคาถูกที่สุดเพราะสามารถโปรแกรมได้ครั้งเดียวไม่สามารถแก้ไขได้อีกจึงเหมาะกับงานที่ตรวจสอบว่าไม่มีข้อผิดพลาดใดๆแล้วชิปแบบOTP จะมีตัวอักษร C แสดงบนตัวชิป
2. มีหน่วยความจำโปรแกรมได้หลายครั้งแบบอีพรอม
   หน่วยความจำโปรแกรมกลุ่มนี้ เรียกว่า EPROM (Erasable Programmable ROM) เป็นชิปที่หน่วยความจำโปรแกรมสามารถเขียนและลบได้โดยใช้แสงอัลตราไวโอเลตหรือยูวี ซึ่งด้านบนของชิปจะมีกรอบกระจกเพื่อให้แสงยูวีส่องผ่านเข้าไปในตัวชิปโดยใช้เวลาประมาณ 5 – 10 นาทีการลบโปรแกรมบ่อยจะทำให้เกิดอาการด้านไม่สามารถโปรแกรมได้ ชิปแบบนี้จะมีอักษร JW แสดงบนตัวชิป

3. ที่มีหน่วยความจำโปรแกรมได้หลายครั้งแบบแฟลชหรืออีอีพรอม
หน่วยความจำโปรแกรมนี้เรียกว่า Flash หรือ EEPROMเป็นแบบที่นิยมมากที่สุดเนื่องจากหน่วยความจำโปรแกรมสามารถอ่านและลบได้หลายพันครั้งทำให้สะดวกในการแก้ไขปรับปรุงชิปแบบแฟลชจะมีอักษร F แสดงบนชิป

2.4.9.2 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล PIC ในแต่ละกลุ่ม

การแบ่งกลุ่มของ PIC จะแย่งตามตัวเลขที่ขึ้นต้นโดยแต่ละกลุ่มจะมีเบอร์ต่างๆแยกออกไปอีกหลายเบอร์ซึ่งแบ่งได้เป็น 6 กลุ่ม คือ

1. ไมโครคอนโทรลเลอร์ตระกูล PIC16c5X
เป็นชิปที่ผลิตออกมาในยุคแรกๆเหมาะกับงานที่ไม่ซับซ้อน มีคำสั่งใช้งานที่เป็นภาษาแอสเซมบลี 33คำสั่ง มี I/O, Timer, Watchdog ไม่มี I^2C หรือ Serial ซึ่งต้องเขียนโปรแกรมขึ้นเองชิปในกลุ่มนี้มีหลายเบอร์แต่ละเบอร์จำนวนขาและขนาดของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลต่างกัน
2. ไมโครคอนโทรลเลอร์ตระกูล PIC16CXXX
บริษัทไมโครชิปได้พัฒนาและปรับปรุงจนเป็นตระกูล PIC16CXXX โดยเป็นชิปแบบ 18 ขามีคำสั่งภาษาแอสเซมบลี 35 คำสั่ง มี Timer เพิ่มขึ้นบางเบอร์มีมากกว่า 1 ตัวมีพอร์ต I2C และ USART ทำให้สามารถติดต่อกับอุปกรณ์ภายนอกได้สะดวกขึ้นเขียนโปรแกรมควบคุมง่ายขึ้นนอกจากนี้ยังได้ทำการเพิ่มขนาดของหน่วยความจำให้มีขนาดใหญ่ขึ้นด้วย
3. ไมโครคอนโทรลเลอร์ตระกูล PIC12CXXX และ PIC12FXXX
เป็นชิปขนาดเล็กที่มีเพียง 8 ขาเท่านั้นเหมาะกับงานเล็กๆมีคำสั่งภาษาแอสเซมบลี 33 และ 35 คำสั่งสำหรับจุดเด่นของ PIC กลุ่มนี้ คือ มีสัญญาณนาฬิกาขนาด 4MHz อยู่ภายในชิปทำให้ไม่ต้องต่อภายนอกและมีหน่วยความจำข้อมูลภายในเป็นแบบ EEPROM แต่ส่วนของหน่วยความจำโปรแกรมยังเป็นแบบ OTP และ EPROM หลังจากนั้นได้ผลิต PIC16FXXX แล้วทางบริษัทก็ได้ผลิต PIC12FXXXขึ้นมาซึ่งเป็นแบบแฟลช ทำให้สามารถเขียนและลบได้หลายครั้ง
4. ไมโครคอนโทรลเลอร์ตระกูล PIC17CXXX
เป็นชิปที่ผลิตออกมาพร้อมกับ PIC16CXXX แต่ต่างกันที่ PIC17CXXX จะมีความสามารถสูงกว่ามีจำนวนขามากกว่ามีคำสั่งภาษาแอสเซมบลี 58 คำสั่ง มีคำสั่งการคูณ หาร รวมทั้งขนาดของหน่วยความจำโปรแกรมหน่วยความจำข้อมูลที่มีขนาดใหญ่ขึ้นและยังสามารถต่อกับหน่วยความจำภายนอกได้อีกด้วย

5. ไมโครคอนโทรลเลอร์ตระกูล PIC16FXXX
   เป็นชิปที่ได้รับความนิยมมากเพราะเป็นชิปที่ความจำโปรแกรมเป็นแบบแฟลชและมีหน่วยความจำข้อมูลแบบ EEPROM ทำให้สามารถพัฒนาโปรแกรมได้ง่าย ซึ่ง PIC16FXXX สนับสนุนการทำงานแบบอินเซอร์กิตดีบักเกอร์ทำให้ไม่ต้องซื้ออีพรอมอีมูเลเตอร์ที่มีราคาแพงมีคำสั่ง 35 คำสั่ง และมีวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลขนาด 10 บิตอยู่ภายใน
6. ไมโครคอนโทรลเลอร์ตระกูล PIC18CXXX และ 18FXXX
   เป็นอีกกลุ่มที่ได้รับความนิยมเนื่องจากมีความสามารถใหญ่กว่า PIC16FXXX ไม่ว่าจะเป็นคำสั่งภาษาแอสเซมบลีที่มีถึง 77 คำสั่ง หรือหน่วยความจำโปรแกรมที่มีขนาดใหญ่ทำสามารถรองรับการเขียนโปรแกรมภาษาซีได้

2.5 การใช้งานไมโครคอนโทรลเลอร์กับพอร์ตอนุกรม

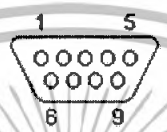
การติดต่อกับพอร์ตอนุกรมนั้นจะยากกว่าการ ติดต่อกับกับพอร์ตขนาน การสื่อสารของอุปกรณ์ที่ต่อกับพอร์ตอนุกรมจะถูกเปลี่ยน (Convert) เป็นสัญญาณแบบขนาน แล้วจึงนำไปประมวลผลต่อ ซึ่งจะใช้ Universal Asynchronous Receive / Transmitter (UART) เป็นตัวทำหน้าที่ส่วนทางด้านโปรแกรม ก็มีรีจิสเตอร์ที่ต้องจัดการมากกว่า Standard Parallel Port (SPP) อีกหลายตัว

ข้อดีของพอร์ตอนุกรม

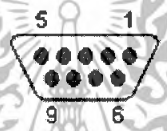
1. ระยะของสายอนุกรมสามารถมีความยาวได้มากกว่าสายของขนานมาก ทั้งนี้เพราะสัญญาณของพอร์ตอนุกรม ซึ่งส่วนใหญ่ใช้มาตรฐาน RS-232C จะมีค่า –3 Volt ถึง –15 Volt สำหรับ Logic “1” หรือ “Mark” และมีค่า +3 Volt ถึง +15 Volt สำหรับ Logic “0” หรือ “Space” (สำหรับช่วง +3 Volt ถึงถึง -3 Volt เป็นช่วง Undefined) ส่วนสัญญาณของสายขนานนั้น Logic “1” จะมีค่า +5 Volt และ logic “0” จะมีค่า 0 Volt ทำให้สัญญาณของสายอนุกรมสามารถรับการสูญเสียของสาย (Cable loss) ได้มากกว่าสัญญาณของสายขนานปรกติสายขนานจะไปได้เพียง 5 ฟุต ส่วนสาย RS-232 จะไปได้ถึง 50 ฟุตที่ความเร็วสูงสุดของมัน
2. สายอนุกรมจะใช้จำนวนสายไฟน้อยกว่าสายขนานถ้าต่อในลักษณะ Null Modemจะใช้สายเพียง 3 เส้น ขณะที่แบบขนาน จะต้องใช้สาย 19 ถึง 25 เส้น
3. การสื่อสารแบบไร้สายเช่นการใช้ Infra Red การส่งพร้อมกันทีละ 8 บิต แบบขนาน จะทำให้ไม่สามารถแยกแยะได้ว่า Bit ใดเป็น Bit0 หรือ Bit1 … เป็นต้น ปัจจุบันอุปกรณ์ IrDA มี

ความเร็วไม่ต่ำกว่า 115.2K Baud แต่มี Pulse length เพียง 3/16 ของ RS-232 เพื่อประหยัดพลังงาน เพราะส่วนมากใช้ในอุปกรณ์แบบพกพาเช่น Laptop หรือ Palmtop

4. ปัจจุบันไมโครคอนโทรลเลอร์ มักมีการผนวกพอร์ตการสื่อสารแบบอนุกรมไว้ด้วย เพราะใช้จำนวนขาน้อยกว่าแบบขนาน RS-232C กำหนด Baud rate ไว้ไม่เกิน 20K Baud ปัจจุบันได้แก้ไขให้รองรับกับเทคโนโลยีใหม่ได้ จึงมีการปรับปรุงถึง RS-232E ซึ่งมีรายละเอียดอีกหลายอย่าง



(Male at the computer side)



(Female at the cable side)

รูปที่ 2.9 พอร์ตอนุกรม

Pin	Name	RS232	V.24	Dir	Description
1	CD	CF	109	←	Carrier Detect
2	RXD	BB	104	←	Receive Data
3	TXD	BA	103	→	Transmit Data
4	DTR	CD	108.2	→	Data Terminal Ready
5	GND	AB	102	—	Signal Ground
6	DSR	CC	107	←	Data Set Ready
7	RTS	CA	105	→	Request to Send
8	CTS	CB	106	←	Clear to Send
9	RI	CE	125	←	Ring Indicator

ตารางที่ 2.7 แสดงถึงหน้าที่ของขาต่าง ๆ

Abbreviation	Full Name	Originator	Function
TD	Transmit Data	DTE	Serial data output (TXD) from DTE.
RD	Receive Data	DCE	Serial data input (RXD) to DTE.
CTS	Clear To Send	DCE	Tell DTE that DCE is ready to exchange data.
(D)CD	(Data) Carrier Detect	DCE	Carrier from remote DCE is detected.
DSR	Data Set Ready	DCE	Tell DTE that DCE is ready to establish a link.
DTR	Data Terminal Ready	DTE	Tell DCE that DTE is ready to establish a link.
RTS	Ready To Send	DTE	Tell DCE that DTE is ready to exchange data.
RI	Ring Indicator	DCE	Ringing signal from the phone line is detected.

ตารางที่ 2.8 แสดงสัญญาณต่างๆที่ส่งในรูปแบบอนุกรม

2.5.1 หน้าที่ของสัญญาณต่าง ๆ มีดังนี้

Protective ground เป็นจุดที่ต่อกับตัวเปลือกของอุปกรณ์ และไม่ต่อกับสัญญาณใด ๆ ในระบบเพื่อใช้ต่อลงดิน เป็นการป้องกันอันตรายจากไฟลัดวงจร และใช้เป็น Shield ป้องกันการรบกวน

Signal ground หรือ Common return เป็นจุดสำคัญที่สุดที่ต้องมีในระบบ RS-232 เพราะเป็นจุดอ้างอิงของทุกสัญญาณ ยกเว้น Protective ground

Transmit data เป็นสัญญาณข้อมูลที่ส่งจาก DTE ไปยัง DCE ในขณะที่ไม่ได้ส่งข้อมูลมันจะมีสถานะเป็นLogic "1" หรือ "Mark" หรือ "Off" หรือ -5 V ถึง -15 V ที่ด้านส่ง (DTE) หรือ -3 V ถึง -15 V ที่ด้านรับ (DCE) การส่งข้อมูลจะเกิดขึ้นได้ต้องมีสัญญาณควบคุมที่เกี่ยวข้อง "On" ก่อน คือสัญญาณ RTS, CTS, (D)CD, DTR และ DSR

Receive data เป็นสัญญาณข้อมูลจาก DCE มายัง DTE ในขณะที่ไม่ได้ส่งข้อมูลมันจะมีสถานะเป็น Logic "1" หรือ "Mark" หรือ "Off" หรือ -5 V ถึง -15 V ที่ด้านส่งหรือ -3 V ถึง -15 V ที่ด้านรับ ในขณะที่ไม่ได้ส่งข้อมูลมันจะมีสถานะเป็น Logic "1" หรือ "Mark" หรือ "Off" หรือ -5 V ถึง -15 V ที่ด้านส่ง (DCE) หรือ -3 V ถึง -15 V ที่ด้านรับ (DTE) กรณีที่เป็นการสื่อสารแบบ Half-duplex สัญญาณ RD จะอยู่สถานะ "Off" ขณะที่สัญญาณ RTS อยู่ในสถานะ "On" และสัญญาณ RD จะยังคงอยู่ในสถานะ "Off" อีกชั่วระยะหนึ่งหลังจากที่สัญญาณ RTS เปลี่ยนจากสถานะ "On" มาเป็น "Off" แล้วเพื่อให้การรับ-ส่งข้อมูลเสร็จสมบูรณ์

Request to send เป็นสัญญาณจากDTE ส่งไปให้DCEเพื่อขอส่งข้อมูล ไปปรกติจะอยู่ในสถานะ "Off" เมื่อต้องการส่งข้อมูลจะเปลี่ยนเป็นสถานะ "On" จนกว่าการส่งเสร็จสิ้นจึงเปลี่ยนกลับมาที่สถานะ "Off" ตามเดิม ทั้งนี้ทาง DTE ต้องได้รับสัญญาณ CTS จึงจะสามารถส่งข้อมูล TD ไปยัง DCE ได้ และสัญญาณ RTS ที่กลับสู่สถานะ "Off" จะไม่สามารถเปลี่ยนเป็น "On" ใหม่ ขณะที่สัญญาณ CTS อยู่ในสถานะ "On" ต้องรอจนกว่าสัญญาณ CTS เปลี่ยนมาอยู่ในสถานะ "Off" ก่อน เพื่อป้องกันการเกิด Overrun

Clear to send เป็นสัญญาณที่ DCE ส่งให้ DTE เพื่อแจ้งว่าพร้อมรับการส่งข้อมูลจาก DTE

Data set ready เป็นสัญญาณที่ DCE ส่งให้ DTE เพื่อแจ้งว่า DCE สามารถเชื่อมโยงไปยังปลาย ทางและพร้อมที่จะติดต่อแล้ว

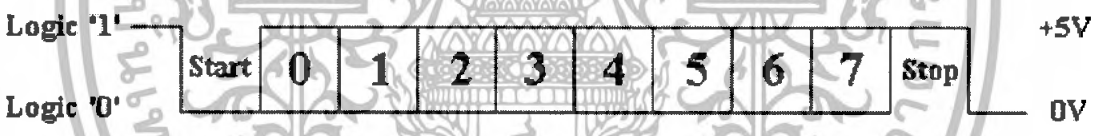
Data Terminal ready เป็นสัญญาณที่ DTE ส่งให้ DCE เพื่อแจ้งว่า DTE พร้อมหรือต้องการจะติดต่อสื่อสาร ซึ่งสัญญาณ DTR นี้ต้องเกิดก่อนทาง DCE จึงจะทำการติดต่อไปยังปลายทางและเมื่อติดต่อได้แล้วจึงส่งสัญญาณ DSR มายัง DTE เพื่อแจ้งให้รู้ว่าพร้อมรับการสื่อสารแล้ว และถ้า DTR เปลี่ยนเป็น Off แปลว่า DTE ไม่ต้องการติดต่อสื่อสารแล้วทาง DCE ก็จะปิดช่องสื่อ-สารและเปลี่ยนสัญญาณ DSR เป็น Off ทันที คู่สัญญาณระหว่าง RTS กับ CTS เป็นเรื่องของความพร้อมเกี่ยวกับช่องสื่อสารระหว่าง DTE กับ DCE ส่วนคู่สัญญาณ DTR กับ DSR เป็นเรื่องของความพร้อมเกี่ยวกับตัวอุปกรณ์ DTE กับ DCE

Data carrier detect เป็นสัญญาณที่ DCE ส่งให้ DTE เพื่อแจ้งว่าได้รับสัญญาณพาหะจาก DCE ที่อยู่อีกด้านหนึ่งของการสื่อสาร ซึ่งหมายความว่า ช่องการสื่อสารระหว่าง DCE ทั้ง 2 ไม่ขาดตอน พร้อมที่จะทำการสื่อสารได้ ซึ่งอุปกรณ์ DTE หรือโปรแกรมที่ควบคุมการสื่อสารมักจะตรวจ สัญญาณนี้ ถ้าไม่อยู่ที่ on แสดงว่าช่องการสื่อสารขาด ก็จะไม่ทำการรับหรือส่งข้อมูล

Ring Indicator เป็นสัญญาณจาก DCE แจ้งให้ DTE รู้ว่ามีการเรียกจาก DCE ที่อยู่อีกด้านหนึ่งของการสื่อสาร ซึ่งมักจะใช้ในระบบ automatic answering

2.5.2 รูปคลื่น สัญญาณ RS-232

การสื่อสารโดย RS-232 เป็นการสื่อสารแบบ asynchronous หมายความว่าสัญญาณ clock ที่ใช้ควบคุมจังหวะไม่ได้ส่งไปพร้อมกับ Data แต่จะใช้ start bit เป็นตัว sync.ในแต่ละ word ของการสื่อสารและใช้สัญญาณ clock ภายในของแต่ละด้านเป็นตัวให้จังหวะเอง



รูปที่ 2.10 รูปคลื่นของสัญญาณที่ส่ง

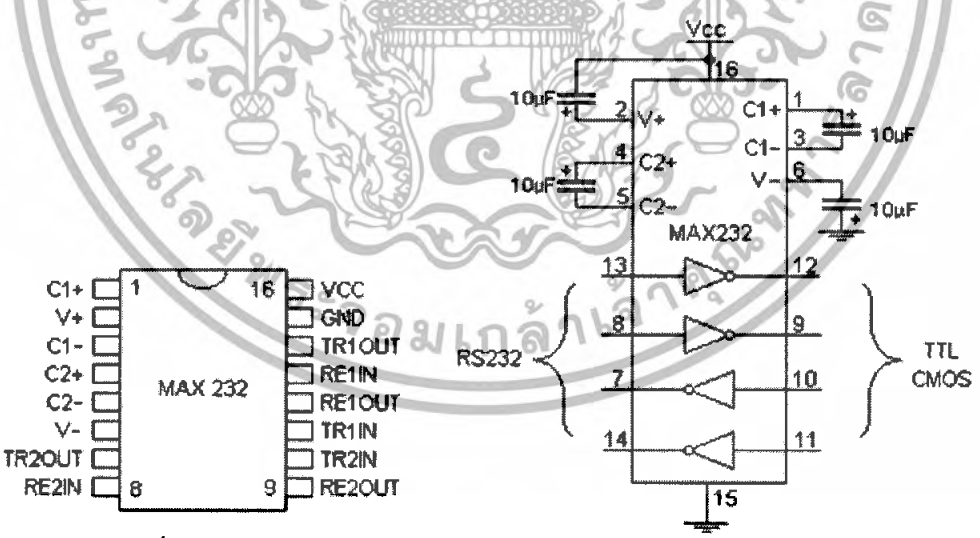
แสดงลักษณะของสัญญาณจาก UART เมื่อใช้ format แบบ 8N1 คือ 8 data bits ไม่มี parity bit และมี 1 stop bit ขณะที่ idle จะอยู่ในสถานะ “Mark” หรือ logic “1” การส่งจะเริ่มจากการส่ง start bit คือ logic “0” และตามด้วย LSB bit จนหมด data bits และถ้ามี parity bit ก็จะส่งที่จุดนี้แล้วลงท้ายด้วย stop bit ซึ่งมีค่าเป็น logic “1” ในรูปได้แสดง bit ที่ต่อถัดจาก stop bit ซึ่งมีค่าเป็น logic “0” หมายความว่าเป็น start bit ของ การส่ง word ถัดไป แต่ถ้ายังไม่มีการส่ง word ถัดไปก็ต้องอยู่ในสถานะของ logic “1” ซึ่งเป็นสถานะของ idle และถ้าสายอยู่ในสถานะของ logic “0” นานกว่าเวลาของการส่ง 1 full word ระบบจะถือว่าเป็นสัญญาณ “Break” เพื่อหยุดการสื่อสาร ดังนั้นต้องไม่ลืมที่จะสั่งในสายกลับสู่สถานะ idle เมื่อสิ้นสุดการส่ง การรับ-ส่งข้อมูลในลักษณะนี้เรียกว่าแบบ frame คือมีกรอบปิดล้อมข้อมูลไว้ด้วย start bit และ stop bit

2.5.3 ตัวแปลงสัญญาณ RS-232

สัญญาณ RS-232 มีค่าแรงไฟต่างจากที่ใช้ใน UART ดังนั้นจึงต้องมีตัวแปลงสัญญาณเพื่อแปลงระดับสัญญาณให้เหมาะสมก่อนที่ จะเชื่อมต่อกับพอร์ตอนุกรม หรือ RS-232 port ของคอมพิวเตอร์

สัญญาณ RS-232 นั้น logic "0" จะมีค่า +3 V ถึง +25 V และ logic "1" จะมีค่า –3 V ถึง –25 V ส่วนค่าระหว่าง –3 V ถึง +3 V เป็นค่า undefined ระดับสัญญาณนี้ใช้กับทุกสัญญาณไม่ใช่เฉพาะสัญญาณรับ-ส่งข้อมูลเท่านั้นแต่ยังรวมถึงสัญญาณควบคุมต่าง ๆ เช่น DTR, RTS, CTS เป็นต้น

IC ที่ใช้มักจะเป็นเบอร์ 1488 (RS-232 Driver) และ 1489 (RS-232 Receiver) โดยภายในแต่ละตัวจะประกอบด้วย inverter 4 ตัวและต้องการไฟเลี้ยง 2 ชุดคือ +7.5 V ถึง +15 V และ –7.5 V ถึง –15 V ซึ่งอาจจะมีปัญหาในเครื่องที่มีไฟเลี้ยง +5 V เพียงชุดเดียว แต่ก็ยังมี IC อีกตัวหนึ่งคือเบอร์ MAX-232 ซึ่งมีวงจร charge pump สามารถสร้างไฟ +10 V และ –10 V จากไฟ +5 V ได้พร้อมทั้งมี 2 Tx และ 2 Rx อยู่ใน package เดียวกัน และรองรับ baud rate ได้ถึง 120 Kbps จึงสะดวกมากเพราะใช้ IC เพียงตัวเดียว รูปข้างล่างคือ MAX-232



รูปที่ 2.11 แสดงโครสร้างภายในและตำแหน่งขาต่างๆของ Max232

ส่วนการที่เราจะนำข้อมูลมาใช้งานก็ต้องแปลงเป็น Parallel ก่อนซึ่งเป็นหน้าที่ของ UART ซึ่งปัจจุบันไมโครคอนโทรลเลอร์มักจะมี serial communication interface (SCI) อยู่ในตัว แต่อาจจะมีงานบางอย่างที่ไม่ได้ใช้ไมโครคอนโทรลเลอร์และต้องการประมวลผลข้อมูลกับพอร์ตอนุกรม เช่น ต่อ ADC เข้ากับ UART หรือต่อ LCD display เข้ากับ Serial comm. ก็ต้องใช้ UART ช่วย เช่น

เบอร์ 8250 หรือ 16550A หรือเบอร์อื่น ๆ ที่ได้กล่าวมาแล้ว แต่มี UART อีกพวกหนึ่งที่แยก Tx bus กับ Rx bus ออกจากกันทำให้มีความคล่องตัวมากขึ้น

2.6 การสื่อสารข้อมูล

การสื่อสารข้อมูลมีจุดมุ่งหมายในการส่ง หรือการถ่ายทอดข่าวสารจากจุดหนึ่งไปยังอีกจุดหนึ่งอย่างถูกต้องเหมือนกับการที่ด้านส่งออกมา ในการสื่อสารรูปแบบใดๆ จะต้องประกอบไปด้วยส่วน ประกอบหลักเบื้องต้น 3 ส่วนดังนี้

1. แหล่งต้นกำเนิดข่าวสาร(Source)
2. ตัวกลางในการสื่อสาร(Media)
3. แหล่งรับข่าวสาร(Receiver)

2.6.1 ประเภทของการสื่อสารข้อมูล

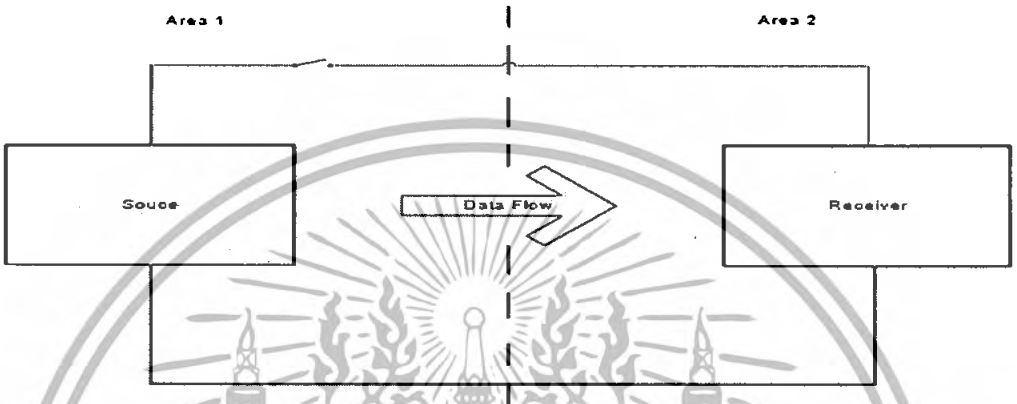
ข้อมูลในระบบของการสื่อสารเขียนแสดงได้ด้วยค่าในระบบเลขฐานสอง โดยใช้ค่าตัวเลข 0 หรือ 1 มาประกอบกันเป็นรหัส แต่ในการส่งเราอาศัยการส่งทางไฟฟ้า ข้อมูลจะถูกเปลี่ยน ให้อยู่ในรูปแบบทางไฟฟ้าโดยใช้ค่าสัญญาณไฟฟ้า 2 ระดับคือสูงและต่ำ ในการวัดอัตราเร็วในการส่งได้จากจำนวนบิตที่ส่งไปในหน่วยเวลาโดยทั่วไปใช้หน่วย Bit per second (bps) ซึ่งในระบบการสื่อสาร ข้อมูลนั้นอาจจะมีขนาดใดก็ได้แต่ในการส่งนั้นจะต้องมีการกำหนดจุดเริ่มต้น และจุดสิ้นสุดของการแต่ละส่ง ตามปกติจะจัดแบ่งข่าวสารที่จะส่งเป็นบล็อกๆ ซึ่งหนึ่งบล็อกคือกลุ่มของบิตจำนวนหนึ่งที่ถูกส่งออกไปเป็นหน่วยเดียวกัน โดยมีการนำกลุ่มบิตนั้นไปผ่านกระบวนการบางอย่าง เพื่อใช้ในการควบ คุมข้อผิดพลาดที่อาจจะเกิดขึ้นเราสามารถจำแนกวิธีการส่งข้อมูลได้หลายแบบตามคุณสมบัติต่างๆดังนี้

2.6.1.1 การจำแนกตามทิศทางในการส่งข้อมูล

สามารถแบ่งการส่งข้อมูลออกได้เป็น 3 ชนิด ดังนี้

2.6.1.1.1 การรับส่งข้อมูลทางเดียว (Simplex)

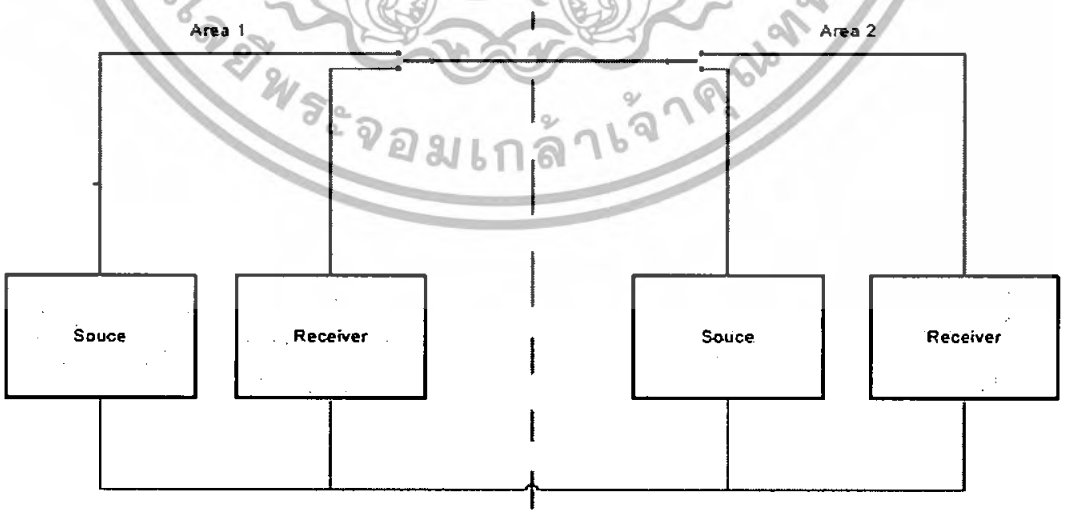
เป็นการสื่อสารทางเดียว ที่เห็นได้ชัดก็คือ การรับส่งโทรทัศน์ และวิทยุกระจายเสียงนั่นเอง สถานีโทรทัศน์จะเป็นตัวส่งและเครื่องรับทำหน้าที่รับเพียงอย่างเดียว จะส่งข่าวสารมายังสถานีส่งไม่ได้



รูปที่ 2.12 แสดงการรับส่งข้อมูลแบบทางเดียว (Simplex)

2.6.1.1.2 การรับส่งแบบผลัดกันส่ง (Half Duplex)

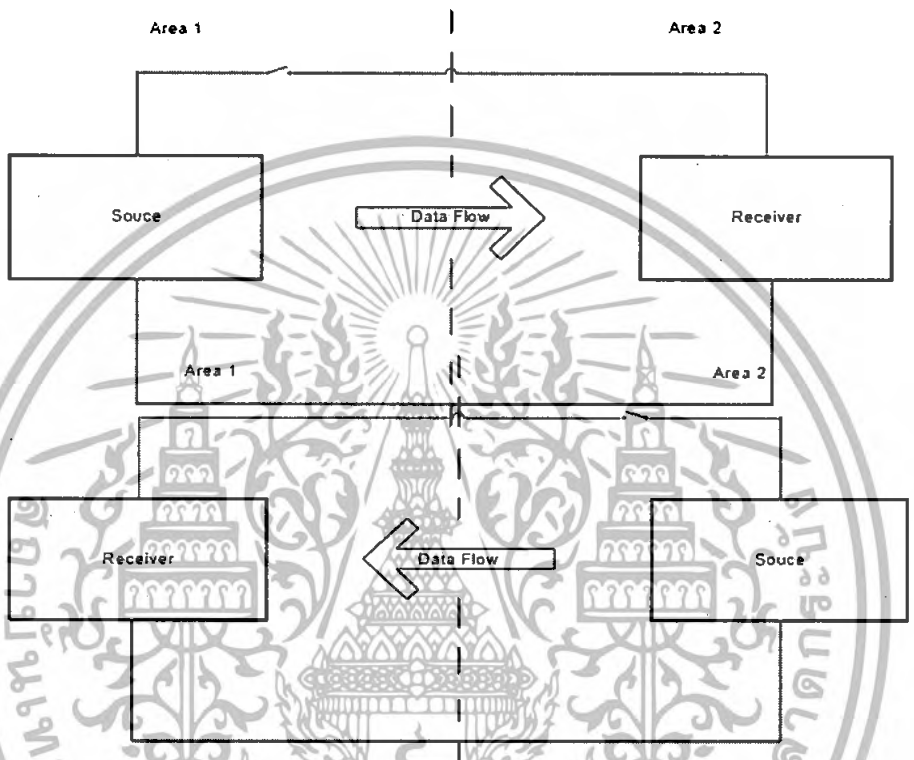
มีคุณสมบัติสามารถรับและส่งข้อมูลได้ แต่ว่าต้องสลับการส่งโดยจะส่งพร้อมกันทั้งสองด้านไม่ได้ อุปกรณ์ที่ใช้ในการติดต่อในแบบนี้ ได้แก่ วิทยุสื่อสาร และอินเตอร์คอม



รูปที่ 2.13 แสดงการรับส่งข้อมูลสวนทางกันได้แบบผลัดการส่ง(Half Diplex)

2.6.1.1.3 การรับส่งสวนทางได้พร้อมกัน (Full Duplex)

การรับสุ่งแบบนี้ผู้รับและผู้ส่งสามารถรับและส่งพร้อมๆกันได้ ในเวลาเดียวกันโดยไม่จำเป็นต้องรอให้อีกฝ่ายหนึ่งส่งจบเสียก่อน เช่น การพูดโทรศัพท์ของเรา



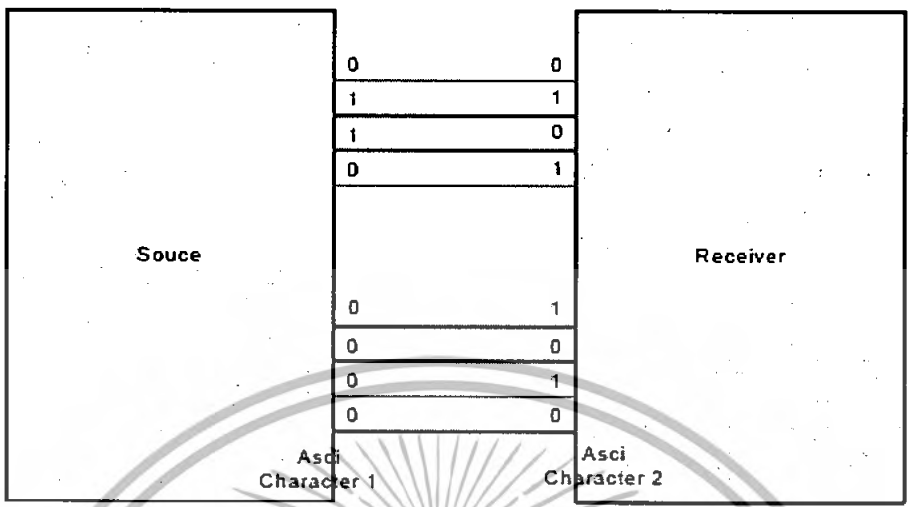
รูปที่ 2.14 แสดงการรับส่งข้อมูลแบบสวนทางกันได้พร้อมกัน (Full Duplex)

2.6.1.2 การจำแนกตามลักษณะการจัดข้อมูล

สามารถแบ่งการส่งข้อมูลออกได้เป็น 2 ชนิดดังนี้

2.6.1.2.1 การส่งข้อมูลแบบขนาน (Parallel Transmission)

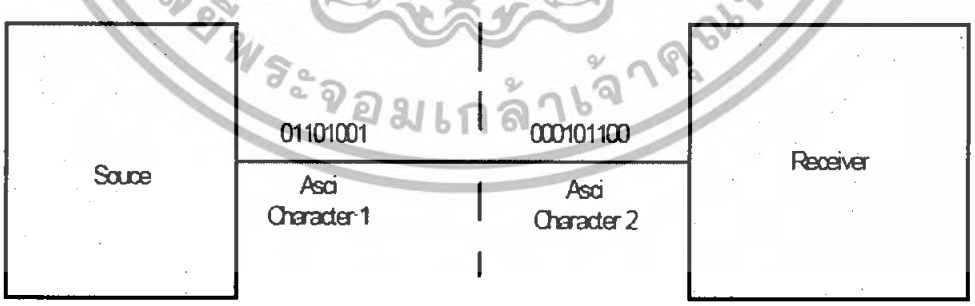
เป็นการส่งข้อมูลทีละไบท์ คือส่งทุกบิตของรหัสที่ประกอบขึ้นเป็นอักขระไปพร้อมๆ กันในเวลาเดียวกัน มีข้อดีคือ ความสามารถในการส่งข้อมูลที่สูงขึ้น แต่จำนวนช่องทางการสื่อสารที่จะต้องมีจำนวนเท่ากับจำนวนที่บิตที่ประกอบขึ้นเป็นอักขระ ซึ่งต้อง ใช้การมัลติเพล็กข้อมูลต่างๆ



รูปที่ 2.15 แสดงการส่งข้อมูลแบบขนาน

2.6.1.2.2 การส่งข้อมูลแบบอนุกรม (Series Transmission)

การส่งข้อมูลแบบนี้จะกระทำทีละบิตด้วยแชนแนลเพียงแชนแนลเดียว ทางด้านรับจะมีอุปกรณ์สำหรับจัดการจัดข้อมูลดังกล่าวเป็นชุดอักขระ ตามข้อตกลงระหว่างอุปกรณ์ปลายทั้งสองที่สื่อสารกัน



รูปที่ 2.16 แสดงการส่งข้อมูลแบบอนุกรม

2.6.1.3 การจำแนกตามความสัมพันธ์ของข้อมูล

ในการสื่อสารข้อมูลนั้นจะต้องพิจารณาถึงความสัมพันธ์ระหว่างข้อมูล 2 ชนิด คือ

- ความสัมพันธ์ระหว่างบิต (Bit Synchronization)

- ความสัมพันธ์ของอักขระ (Character Synchronization)

ในการสื่อสารข้อมูลเราจำแนกวิธีการส่งข้อมูลตามการกำหนดความสัมพันธ์ของข้อมูลได้ 2 แบบ คือ

2.6.1.3.1 การส่งแบบสัมพันธ์ (Synchronous Transmission)

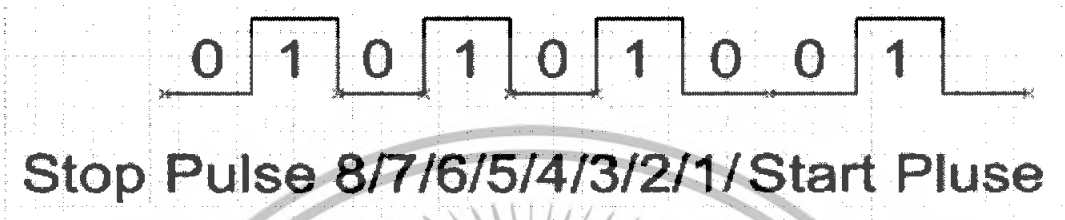
การส่งแบบนี้ใช้สำหรับการส่งข้อมูลทั้งชุดไปในครั้งเดียวดังรูป 2.17 เทคนิคการส่งแบบนี้ช่วงเวลาระหว่างบิตต่อบิตจะมีค่าเท่ากัน การส่งมีลักษณะคล้ายกับการส่งข่าวสารในรูปของเลขฐานสองที่มีจำนวนติดต่อกันไป โดยไม่ได้แยกว่าความยาวใดเป็นช่วงอักษรใดในระบบเช่นนี้บิดแต่ละบิตจะมีความยาวเท่ากัน ตัวอักษรแต่ละตัวมีช่วงเวลาห่างกันเท่ากับศูนย์ ทางด้านรับนั้นเพียงหาว่าบิตแรกของตัวอักษรตัวแรกคือบิตใด และทราบขนาดหรือจำนวนบิตในหนึ่งตัวอักษร พร้อมทั้งความเร็วในการส่งก็สามารถแยกข่าวสารของแต่ละอักษรออกมาได้

รูปที่ 2.17 แสดงการส่งข้อมูลแบบสัมพันธ์

2.6.1.3.2 การส่งแบบไม่สัมพันธ์ (Asynchronous Transmission)

การส่งแบบนี้ตัวอักขระจะถูกส่งออกไปที่เวลาใดๆ ก็ได้โดยไม่จำเป็นต้องมีความสัมพันธ์ระหว่างตัวอักษรว่าต้องมีเวลาที่แน่นอนอย่างไร ทางด้านรับจะต้องทราบถึงบิตเริ่มต้นของรหัสแต่ละตัวว่าเริ่มต้นเมื่อใดซึ่งเมื่อใดซึ่งสามารถกระทำได้โดยการเพิ่มบิดที่เรียกว่า พัลส์เริ่มต้น (Start Pulse) โดยเดิมเข้าไปที่ข้างหน้าชุดของทุกอักขระ และเมื่อการส่งส่งครบทุกบิตของตัวอักษรแล้วจะต้อง มีบิตสำหรับบอกถึงการสิ้นสุดที่เรียกว่า พัลส์การสิ้นสุด (Stop Pulse) ส่งมาให้ทางด้านรับมี

เวลาสำหรับ การเตรียมข้อมูลของตัวอักษรตัวต่อไปดังรูป 2.20 บางครั้งจึงเรียกระบบการส่งแบบนี้ว่า ระบบการส่งแบบเริ่มหยุด (Start-stop Transmission)



รูปที่ 2.18 แสดงการส่งข้อมูลแบบ ไม่สัมพันธ์

2.7 โปรแกรมซี-พลัส-พลัส บิลเดอร์ (C++ Builder)

ในโครงงานนี้ได้มีการใช้งานโปรแกรมซี-พลัส-พลัส บิลเดอร์เพื่อทำการติดต่อกับผู้ใช้งานให้สามารถควบคุมรูปแบบการเปิดและปิดของหัวของน้ำพุตามที่ผู้ใช้งานต้องการได้สะดวกและรวดเร็วเพื่อให้ผู้ใช้งานระบบน้ำพุที่มีความรู้ทางด้านอิเล็กทรอนิกส์หรือไม่ก็จะสามารถใช้งานระบบได้โดยศึกษาวิธีการใช้งานเพียงเล็กน้อยเท่านั้นการเขียนโปรแกรมโดยโปรแกรมซี-พลัส-พลัส บิลเดอร์มีรูปแบบของโครงสร้างที่สำคัญดังต่อไปนี้

2.7.1 การเขียนโปรแกรมแบบวิชวล

วิชวล หมายถึง ภาพหรือสิ่งที่เรามองเห็นการเขียนโปรแกรมแบบวิชวลหรือวิชวลโปรแกรมมิ่ง จึงหมายถึงการเขียนโปรแกรมด้วยภาพหรือการเขียนโปรแกรมด้วยสิ่งที่เรามองเห็น ส่วนประกอบเบื้องต้นสำหรับใช้เขียนโปรแกรมแบบวิชวลมี 4 รายการดังต่อไปนี้

1. ฟอร์ม (form) เป็นหน้าต่างว่างใช้สำหรับออกแบบโปรแกรม การทำงานต่างๆของโปรแกรมจะปรากฏอยู่บนฟอร์ม
2. คอมโพเนนต์ (component) เป็นส่วนประกอบต่างๆที่จะต้องใช้ในโปรแกรม เช่น เมนู (menu) ปุ่มหรือ บัตทอน (button – ปุ่มสำหรับให้โปรแกรมทำงานอย่างหนึ่ง) เมสเสจบ็อกซ์ (message box - กรอบแสดงข้อความ) ไดละล็อกบ็อกซ์ (dialog box – กรอบสำหรับโต้ตอบกับโปรแกรม)
3. คำสั่งสำหรับจัดลักษณะและการทำงานของคอมโพเนนต์

4. คำสั่งสำหรับควบคุมการทำงานของโปรแกรม

2.7.2 C++ และ C++บิลเดอร์

C++ (ซี-พลัส-พลัส) และ C++บิลเดอร์(C++ Builder - ซี-พลัส-พลัส บิลเดอร์) ต่างเป็นภาษาคอมพิวเตอร์ซึ่งใช้สำหรับเขียนโปรแกรมเพื่อสั่งให้คอมพิวเตอร์ทำงานต่างๆตามที่เราต้องการ C++ และ C++ บิลเดอร์มีคำและกฎเกณฑ์ในการเขียนโปรแกรมเหมือนกันทุกประการแต่วิธีเขียนโปรแกรมต่างกันกล่าวคือC++ ใช้วิธีเขียนโปรแกรมด้วยอักษรและเครื่องหมายแต่ C++บิลเดอร์ใช้วิธีเขียนโปรแกรมแบบวิชวลดังนั้นผู้เขียนโปรแกรมด้วย C++ จึงสามารถเปลี่ยนมาเขียนโปรแกรมด้วย C++บิลเดอร์ได้อย่างรวดเร็วเพราะเปลี่ยนแปลงเฉพาะวิธีการเท่านั้นซึ่งวิธีการของ C++บิลเดอร์จะทำให้เราสามารถเขียนโปรแกรมได้ง่ายและรวดเร็วกว่าวิธีการของ C++

C++บิลเดอร์เป็น ANSI C++ (แอนซี ซี-พลัส-พลัส) ซึ่งหมายความว่า C++บิลเดอร์เป็นภาษาC++ที่เป็นไปตามมาตรฐาน ANSI (American National Standards Institue - สถาบันมาตรฐานสหรัฐอเมริกาทำหน้าที่กำหนดมาตรฐานต่างๆรวมทั้งภาษาC++ด้วย) ดังนั้นผู้ที่เคยเขียนโปรแกรมด้วย ANSI C++ มาก่อนไม่ว่าจะเป็นผลิตภัณฑ์ของบริษัทใดก็ตามจะสามารถเขียนโปรแกรมด้วย C++บิลเดอร์ได้ทันที

C++บิลเดอร์ผลิตโดยบริษัทอินไพรซ์ (Inprise Corporation) ซึ่งเปลี่ยนชื่อมาจากบริษัทบอร์แลนด์อินเตอร์เนชันนัล (Borland International, Inc.) ประเทศสหรัฐอเมริกา C++บิลเดอร์มีชื่อเต็มว่าบอร์แลนด์ C++บิลเดอร์ (Borland C++ Builder) และนิยมเรียกว่า C++บิลเดอร์ C++บิลเดอร์เวอร์ชั่นแรกมีจำหน่ายในตอนต้นปี 1997 สำหรับ C++บิลเดอร์เวอร์ชั่น 5 มีจำหน่ายในตอนต้นปี 2000 โดยแบ่งการจำหน่ายเป็น 3 ชุดคือ 1) Borland C++ Builder 5 Standard 2) Borland C++ Builder 5 Professional 3) Borland C++Builder 5 Enterprise ซึ่งแต่ละชุดจะมีส่วนประกอบไม่เท่ากันคือชุด Standard จะมีส่วนประกอบน้อยที่สุดและชุด Enterprise จะมีส่วนประกอบมากที่สุด

วิธีเขียนโปรแกรมด้วย C++บิลเดอร์ทำได้โดยการนำคอมโพเนนต์ที่เรามองเห็นมาวางในฟอร์มแล้วกำหนดลักษณะการทำงานให้กับคอมโพเนนต์นั้น ในบางคอมโพเนนต์ในบางโปรแกรมอาจจะต้องเขียนคำสั่งควบคุมการทำงานของโปรแกรมเพิ่มเติมต่อจากนั้นจึงคอมไพล์ (compile) และรัน (run) โปรแกรมนั้นได้ผลจากการคอมไพล์โปรแกรมจะได้ไฟล์ชนิด .EXE ซึ่งสามารถนำไปรันในคอมพิวเตอร์เครื่องอื่นได้อย่างอิสระโดยไม่ต้องเกี่ยวข้องกับ C++บิลเดอร์อีกเลย

C++บิลเดอร์เป็นภาษา C++ ประเภทที่สมารถสร้างโปรแกรมใช้งานได้อย่างรวดเร็ว ภาษาคอมพิวเตอร์ประเภทนี้มีชื่อย่อว่า (RAD – Rapid Application Development) ซึ่งหมายความว่า

ด้วย C++บิลเดอร์เราสามารถเขียนโปรแกรมภาษา C++ เพื่อรันในวินโดวส์ได้เร็วกว่าการเขียนโปรแกรมแบบธรรมดา

2.7.2.1 การติดต่อระหว่างวินโดวส์กับแอปพลิเคชั่น

มีคำ 2 คำที่เกี่ยวข้องกับการอธิบายเรื่องการติดต่อระหว่างวินโดวส์กับแอปพลิเคชั่นคือ อีเวนต์(event) และ เมสเสจ(message)

อีเวนต์

อีเวนต์หมายถึงทุกสิ่งทุกอย่างที่เกิดขึ้นบนระบบคอมพิวเตอร์ เช่น การเลื่อนเมาส์ การคลิก การกดคีย์ การเลื่อนวินโดว์ การเปลี่ยนขนาดของวินโดว์ เมื่อเกิดอีเวนต์ขึ้นวินโดวส์จะเก็บอีเวนต์ทั้งหมดไว้ในคิว(queue) ที่มีชื่อเรียกว่า โกลบอลอีเวนต์คิว (global event queue – คิวของทั้งหมด ซึ่งบางทีเรียกว่า system-wide queue หรือ hardware event queue) โกลบอลอีเวนต์คิวใช้สำหรับเก็บอีเวนต์ของทุกแอปพริเคชั่นที่รันในวินโดวส์

เมสเสจ

เมสเสจหมายถึงชุดของข่าวสารที่เป็นรายละเอียดของแต่ละอีเวนต์ วินโดวส์จะสร้างเมสเสจของอีเวนต์โดยการส่งอีเวนต์ที่เกิดขึ้นให้แก่ไดรเวอร์(driver – โปรแกรมที่คอมพิวเตอร์ใช้ติดต่อกับอุปกรณ์ชนิดนั้น) ที่สอดคล้องกับอีเวนต์นั้น เช่น เมื่อเราคลิกจะเกิดอีเวนต์คลิก วินโดวส์จะส่งอีเวนต์นี้ให้แก่ไดรเวอร์ของเมาส์และไดรเวอร์ของเมาส์จะสร้างเมสเสจคลิกให้แก่วินโดวส์โดยไดรเวอร์จึงส่งเมสเสจนี้ไว้ในซิสเต็มคิว(system queue-คิวของระบบ) ต่อจากนั้นวินโดวส์จึงส่งเมสเสจจากซิสเต็มคิวไปที่แอปพลิเคชั่นคิว(application queue-คิวเก็บเมสเสจของแอปพลิเคชั่น)ของแอปพลิเคชั่นต่างๆที่กำลังทำงานอยู่ในขณะนั้นซึ่งเป็นผลให้แอปพลิเคชั่นได้รับเมสเสจจากวินโดวส์เนื่องจากวินโดวส์สามารถรันแอปพลิเคชั่นได้พร้อมกันครั้งละหลายแอปพลิเคชั่น เพราะฉะนั้นทุกแอปพลิเคชั่นจึงมีแอปพลิเคชั่นคิวสำหรับแอปพลิเคชั่นนั้นโดยเฉพาะความสัมพันธ์ระหว่างอีเวนต์กับเมสเสจ

หลักการทำงานของแอปพลิเคชั่นที่รันบนวินโดวส์ก็คือ การดำเนินการกับเมสเสจการรับส่งเมสเสจกับวินโดวส์หรือกับแอปพลิเคชั่นอื่นเมสเสจทั้งหมดของวินโดวส์กำหนดไว้เป็นคอนสแตนต์(constant – ค่าคงที่ไม่เปลี่ยนแปลง) เช่น จากเมนูที่ปรากฏบนจอภาพเมื่อเราเลือกรายการหนึ่งจะมีเมสเสจ WM_COMMAND เกิดขึ้นในบรรดาเมสเสจเหล่านี้มีหลายเมสเสจที่เกิดขึ้นพร้อมกับแวลิเอเบิล(variable – ค่าที่เปลี่ยนแปลงได้) จำนวนหนึ่งซึ่งนำมาใช้ในแอปพลิเคชั่นได้ เช่น เมื่อเรากดปุ่มซ้ายของเมาส์ วินโดวส์จะสร้างเมสเสจ WM_LBUTTONDOWN ขึ้นมาเมสเสจนี้ประกอบด้วยแวริเอเบิลซึ่งเป็นโคออร์ดิเนต (coordinate – ตำแหน่ง) ของเมาส์พอยน์เตอร์เรา

สามารถนำค่าดังกล่าวนี้มาใช้ได้ เช่น ใช้สำหรับแสดงผลบนจอภาพ ใช้ในการตรวจสอบตำแหน่งของเมาส์พอยน์เตอร์ เมสเสจในวินโดวส์มีประมาณ 200 เมสเสจแต่เมสเสจที่ใช้บ่อยมีประมาณ 20 เมสเสจเท่านั้น

2.7.3 โปรแกรมที่ทำงานด้วยอีเวนต์

แอปพลิเคชั่นที่รันในวินโดวส์จะใช้เวลาส่วนใหญ่เพื่อการรับและส่งเมสเสจชนิดต่างๆจึงนิยมเรียกแอปพลิเคชั่น(หรือโปรแกรม)ที่ทำงานในลักษณะนี้ว่าโปรแกรมที่ทำงานด้วยอีเวนต์(event-drive program)ซึ่งเป็นโปรแกรมที่มีหลักการทำงานดังต่อไปนี้คือในโปรแกรมที่ทำงานด้วยอีเวนต์ประกอบด้วยฟังก์ชั่น(function-สเตตเมนต์ชุดหนึ่งซึ่งทำงานอย่างหนึ่งและมีชื่อเฉพาะสำหรับให้ส่วนอื่นของโปรแกรมเรียกใช้งานได้)จำนวนหนึ่งซึ่งเรียกว่าฟังก์ชั่นตอบสนองอีเวนต์หรือเมตธอด(method)ซึ่งแต่ละฟังก์ชั่นจะมีหน้าที่ตอบสนองกับแต่ละอีเวนต์โดยเฉพาะและเป็นอิสระต่อกันทั้งนี้เนื่องจากในวินโดวส์จะมีอีเวนต์เกิดขึ้นเมื่อใดก็ได้เราจึงไม่ทราบว่าจะเรียกใช้ฟังก์ชั่นอะไรเมื่อใด

สมมติว่าเราจะเขียนโปรแกรมให้ทำงานพร้อมกัน2งานคือ 1) แสดงเวลาปัจจุบันซึ่งตัวเลขเปลี่ยนแปลงเวลาทุกวินาทีที่มุมล่างขวาของจอภาพ 2) แสดงเมนูซึ่งมีหลายรายการเมนูให้เลือกที่กลางจอภาพขณะที่โปรแกรมทำงานในบริเวณแสดงเวลาเราจะเห็นเวลาเปลี่ยนแปลงไปเรื่อยๆและในบริเวณที่แสดงเมนูจะเห็นรายการเมนูซึ่งเราสามารถเลือกรายการหนึ่งได้ทันทีโปรแกรมจะมีส่วนประกอบและหลักการทำงานดังนี้ 1) ในส่วนแสดงเวลาจะมีสเตตเมนต์เกี่ยวกับการแสดงเวลา 2) ในส่วนเมนูจะมีสเตตเมนต์แสดงเมนูและมีฟังก์ชั่นตอบสนองอีเวนต์การเลือกรายการเมนูแต่ละรายการโดยแต่ละส่วนของโปรแกรมและแต่ละฟังก์ชั่นที่มีอยู่ต่างทำงานแยกเป็นอิสระไม่เกี่ยวข้องกันในการทำงานเมื่อแอปพลิเคชั่นได้รับเมสเสจมาจากวินโดวส์แอปพลิเคชั่นจะดำเนินการเลือกฟังก์ชั่นที่สอดคล้องกันมาทำงาน เช่น ถ้ามีการคลิกเลือกรายการหนึ่งจากเมนูฟังก์ชั่นตอบสนองอีเวนต์การเลือกเมนูรายการนั้นจะทำงาน ด้วยกลไกการทำงานดังกล่าวนี้เราจึงสามารถสร้างแอปพลิเคชั่นที่มีความซับซ้อนได้ง่ายขึ้นเพราะเราเพียงแต่สร้างฟังก์ชั่นตอบสนองอีเวนต์ซึ่งได้รับมาจากวินโดวส์เท่านั้นโดยไม่จำเป็นต้องวนใจว่าเมสเสจนั้นจะเกิดขึ้นเมื่อไรเราสนใจแต่เพียงว่าเมื่อเมสเสจนั้นเกิดขึ้นโปรแกรมจะต้องทำอะไร

2.7.4 หลักการเขียนโปรแกรม

C++บิลเดอร์เป็นโปรแกรมภาษาคอมพิวเตอร์ชนิดคอมไพเลอร์(compiler-โปรแกรมแปลภาษา)ทำหน้าที่ คอมไพล์(compile-แปล)ซอร์สโค้ด(source code-ภาษาคอมพิวเตอร์ที่คน

เข้าใจ)ให้เป็น ออปเจ็กต์โค้ด(object code-รหัสที่สั่งให้คอมพิวเตอร์ทำงาน)โดยมีการลิงค์(link-รวมส่วนประกอบต่างๆของโปรแกรมเข้าด้วยกัน)แบบอัตโนมัติ
วิธีเขียนโปรแกรมมีลำดับขั้นดังนี้

1. สร้างซอร์สโค้ด ซอร์สโค้ดเป็นภาษาอังกฤษที่เขียนขึ้นตากกฎเกณฑ์ของภาษาC++ผู้ที่รู้ภาษาC++จะเข้าใจความหมายของซอร์สโค้ดว่าเป็นการสั่งให้คอมพิวเตอร์ทำอะไรทำอย่างไรและทำเมื่อไรการสร้างซอร์สโค้ดใน C++บิลเดอร์สามารถทำได้ง่ายและรวดเร็ว เพราะ C++บิลเดอร์จะสร้างซอร์สโค้ดส่วนหนึ่งอัตโนมัติซึ่งส่วนหนึ่งที่ทำให้โปรแกรมสามารถทำงานได้ในระดับหนึ่งส่วนของซอร์สโค้ดที่จะทำให้โปรแกรมทำงานได้สมบูรณ์เราอาจป้อนเพิ่มเติมเข้าทางคีย์บอร์ดหรือโดยการอ่านจากดิสก์ก็ได้
2. คอมไพล์ เมื่อมีซอร์สโค้ดอยู่ในคอมพิวเตอร์เรียบร้อยแล้วขั้นต่อไปก็คือการคอมไพล์ซึ่งเป็นการตรวจสอบว่าโปรแกรมเขียนถูกต้องตามกฎหรือไม่ถ้าผิดคอมไพล์เลอร์จะแสดงข้อความสาเหตุที่ผิดเราจะต้องแก้ไขและคอมไพล์ใหม่จนถูกต้องเมื่อไม่พบความผิดพลาดแล้วก็ถือว่าโปรแกรมนั้นผ่านการคอมไพล์
3. ลิงค์ สำหรับโปรแกรมที่ผ่านการคอมไพล์แล้ว C++บิลเดอร์จะนำส่วนประกอบอื่นๆเช่น โปรแกรมย่อย ข้อมูล ซึ่งโปรแกรมของเราจะต้องใช้เข้ามารวมเพื่อให้โปรแกรมสามารถทำงานได้ตามต้องการขั้นตอนนี้เรียกว่าการลิงค์ซึ่งC++บลิเดอร์จะดำเนินการให้แบบอัตโนมัติทันทีที่คอมไพล์ผ่านเมื่อผ่านการลิงค์แล้วเราจะได้ออปเจ็กต์โค้ดอยู่ในไฟล์ชนิด .EXE
4. รัน เมื่อเป็นออปเจ็กต์โค้ดแล้วโปรแกรมนั้นก็สามารถทำงานได้การสั่งให้โปรแกรมทำงานเรียกว่าการ รัน(run)ขณะรันโปรแกรมอาจจะมีความผิดพลาดเกิดขึ้นได้อีกเราเรียกความผิดพลาดกลุ่มนี้ว่ารันไทม์เออเรอร์(runtime error)ซึ่งโปรแกรมจะหยุดทำงานทันทีและแสดงสาเหตุของความผิดพลาดนั้น

2.7.5 C++บิลเดอร์IDE

C++บิลเดอร์ได้รวมส่วนประกอบต่างๆที่เกี่ยวข้องกับการเขียนโปรแกรมไว้ด้วยกันเพื่อเป็นการอำนวยความสะดวกในการเขียนโปรแกรมให้แก่ผู้เขียนโปรแกรมส่วนประกอบเหล่านี้มีชื่อย่อว่า IDE(Integrated Development Environment-ไอดีอี)ความสามารถของIDEมีตัวอย่างดังนี้

1. มีส่วนประกอบต่างๆคือ ฟอร์ม คอมโพเนนต์ คำสั่งดำเนินการกับคอมโพเนนต์เพื่อสำหรับใช้ออกแบบโปรแกรมแบบวิชวล

2. มีโค้ดเอดิเตอร์(code editor)ซึ่งช่วยอำนวยความสะดวกในการป้อนซอร์สโค้ดและปรับปรุงซอร์สโค้ด
3. ถ้าพบความผิดพลาดในการคอมไพล์ IDE จะแสดงสาเหตุความผิดพลาดนั้นและเลื่อนเคอร์เซอร์ไปอยู่ ณ บริเวณที่มีความผิดพลาดซึ่งช่วยให้เราทราบสาเหตุของความผิดพลาดและสามารถหาตำแหน่งที่ผิดพลาดได้เร็วขึ้น
4. อำนวยความสะดวกในการรันโปรแกรมโดยไม่ต้องออกจาก IDE ทำให้เราสามารถทดสอบโปรแกรมได้อย่างสะดวกและรวดเร็ว
5. มีระบบ Help ซึ่งสามารถแสดงคำอธิบายต่างเกี่ยวกับ C++บิลเดอร์ได้ทันทีและครบถ้วน

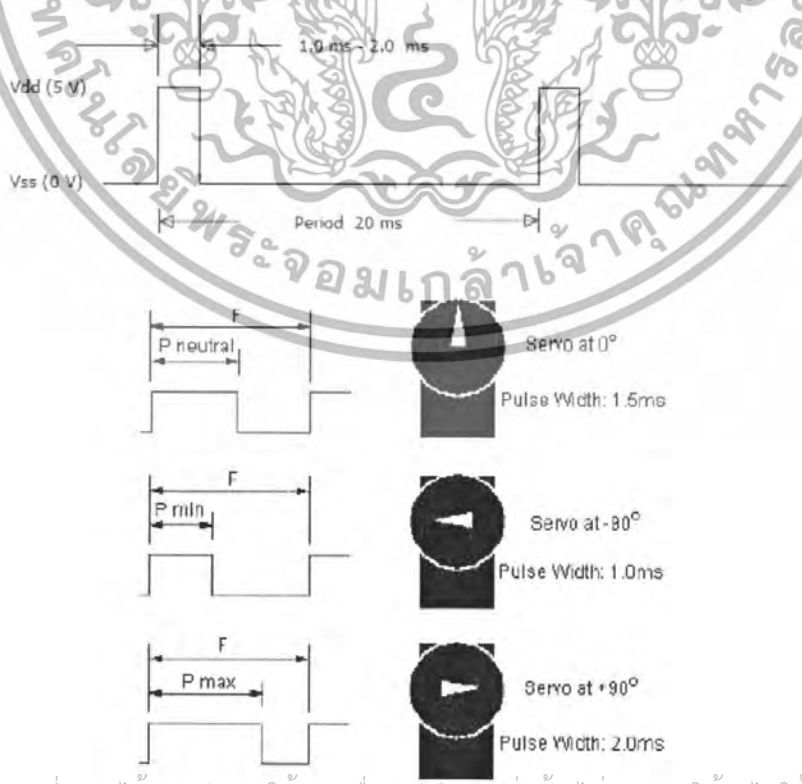
2.8 เซอร์โวมอเตอร์(Servo motor)

เซอร์โวมอเตอร์คือมอเตอร์ไฟฟ้ากระแสตรง (DC motor) ที่ถูกประกอบรวมกับ ชุดเกียร์ และ ส่วนควบคุม ต่างๆ ไว้ในโมดูลเดียวกัน หรือ ภายในกล่องพาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อใช้งานเพียง 3 เส้นเท่านั้น คือ VCC,GNDและ สายสัญญาณควบคุม(Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้าย หรือ ขวาได้จากสายสัญญาณเพียงเส้นเดียว โดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณ พัลส์วิดมอด (PWM) แบบ TTL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 โวลต์ขึ้นอยู่กับคุณสมบัติของมอเตอร์แต่ละตัวข้อดีของมอเตอร์ชนิดนี้ก็คือจะมีขนาดเล็กน้ำหนักเบา,ให้แรงบิดสูง,กินพลังงานน้อยและสามารถควบคุมด้วยแรงดันลอจิกที่เป็น TTL ได้โดยตรงไม่จำเป็นต้องต่อวงจรขับ(Driver)อื่นๆเพราะมอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้วซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่งหรือทิศทางองศาที่ต้องการได้โดยอาศัยสัญญาณความกว้างพัลส์ ที่ป้อนให้มอเตอร์แต่เซอร์โวมอเตอร์นี้จะหมุนได้แค่เพียงในช่วงประมาณ 180° หรือ ครึ่งรอบเท่านั้น หรือ บางรุ่นอาจหมุนได้ถึง 210° แต่จะไม่สามารถหมุนเป็นวงรอบได้เนื่องจากโครงสร้างภายในจะประกอบด้วย ตัวต้านทานชนิดปรับค่าได้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และ ตัวต้านทานนี้จะถูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้น เซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้เพียงแค่ประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้น เพื่อป้องกันความเสียหายที่จะเกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360°) นั้นก็สามารถทำได้โดยจะต้องทำการปรับแต่ง (Modify) ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์

รูปที่ 2.19 การแยกชิ้นส่วนของเซอร์โว

2.8.1 หลักการทำงานของ Servo motor

การควบคุมการทำงานของเซอร์โวมอเตอร์ทำได้โดยการป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้นๆ โดยทั่วไปแล้วความกว้างของสัญญาณพัลส์จะมีจุดให้อ้างอิง 3 จุด ดังรูป คือ



รูปที่ 2.20 การทำงานของเซอร์โวมอเตอร์

สัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือ จุดกึ่งกลางของมอเตอร์

สัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม - 90 องศา หรือในทิศทางทวนเข็มนาฬิกา

สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม + 90 องศา หรือในทิศทางตามเข็มนาฬิกา

ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆนั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่างๆ โดยอ้างอิงจากจุดทั้ง 3 จุดที่กล่าวมานี้ตัวอย่างเช่นถ้าต้องการให้มอเตอร์หมุนไปที่มุม - 45 องศา เราก็จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 ms เป็นต้นและสัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุกๆ 20 ms (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้ โดยหลักการก็คือจะอาศัยการเปรียบเทียบช่วงเวลาของความกว้างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงตามการหมุนของมอเตอร์เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่า (VR) เปลี่ยนแปลงไปเป็นผลทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงตามไปด้วยโดยในขณะที่เราป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ทางขาสัญญาณควบคุม สัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้ง 2 ไม่เท่ากันมอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั่งค่าเวลาความกว้างพัลส์ของวงจร RC เปลี่ยนแปลงจนเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

2.8.2 การปรับแต่งเซอร์โวมอเตอร์

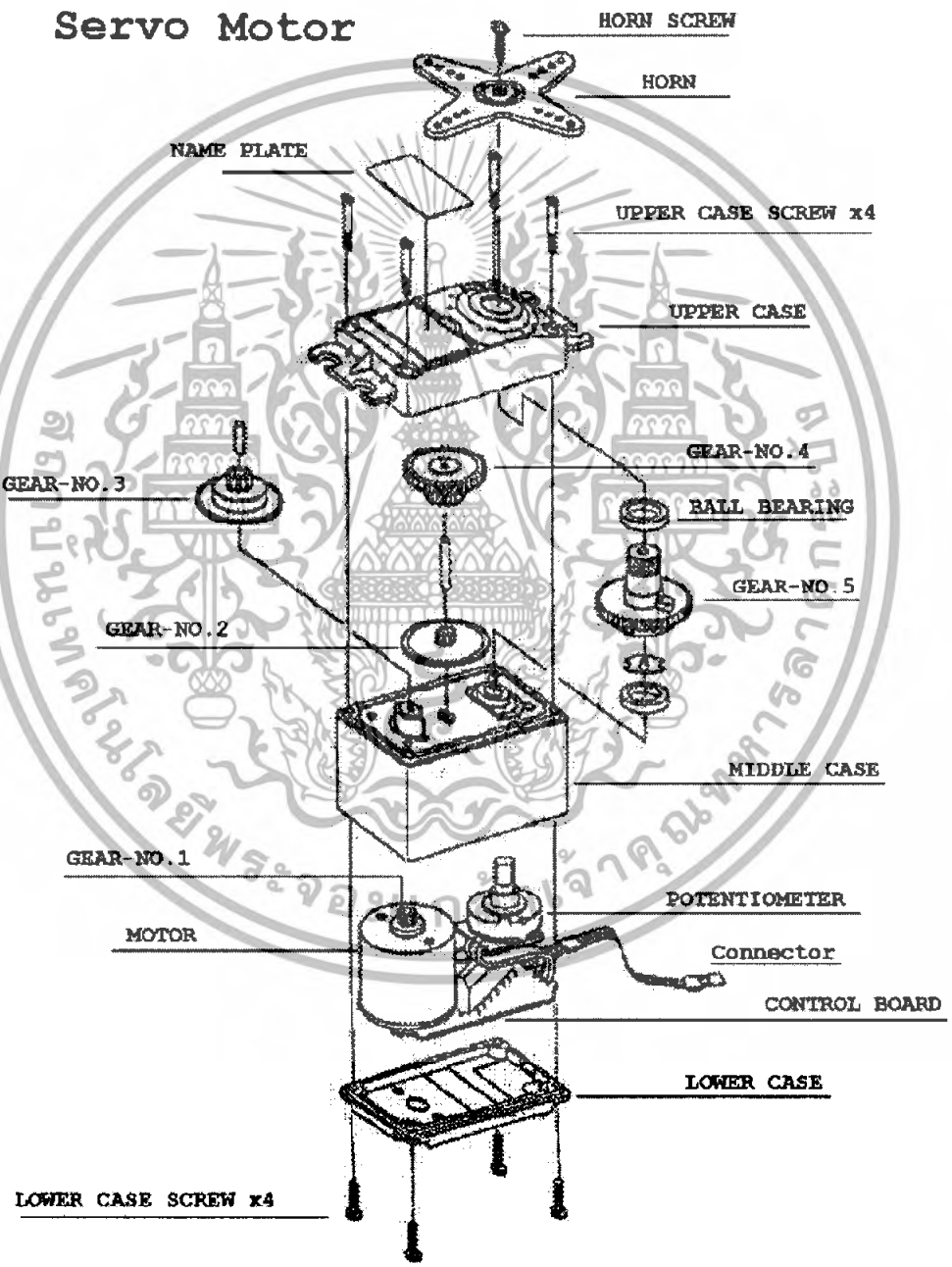
จากคุณสมบัติของ Servo motor ที่ผลิตออกมาจากโรงงานจะสามารถหมุนได้แค่พียงประมาณ 180องศาหรือประมาณครึ่งรอบเท่านั้น หากเราต้องการนำเอา Servo motor ไปใช้งานในลักษณะที่หมุนเป็นวงรอบนั้นก็สามารถทำได้แต่ก็จะสูญเสียการควบคุมในเรื่องของการสั่งให้มอเตอร์หมุนไปในตำแหน่ง หรือ มุมที่ต้องการไปด้วย จะทำได้ก็เพียงในเรื่องของการสั่งให้หมุนซ้าย, ขวา และหยุด เท่านั้นโดยการทำให้มอเตอร์สามารถหมุนเป็นวงรอบได้นั้นจะต้องทำการปรับแต่งหรือแก้ไขโครงสร้างภายในบางส่วนของมอเตอร์ ซึ่งได้แก่

- การต่อตัวต้านทานคงที่ 2 ตัวอนุกรม แทนตัวต้านทานปรับค่าได้

- ตัดชิ้นส่วนของแกนเฟืองที่ทำหน้าที่หยุดมอเตอร์ (TAB STOP) ออก
- การดัดแปลงตัวต้านทานปรับค่าได้ (VR) ให้สามารถหมุนได้รอบทิศทาง (360°)

มีขั้นตอนดังต่อไปนี้

1. ถอดชิ้นส่วนของ Servo motor ออกเป็นส่วนๆ

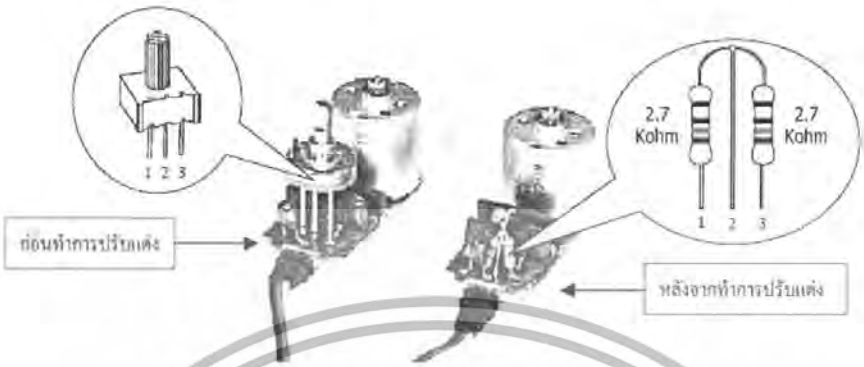


รูปที่ 2.21 ส่วนประกอบของเซอร์โว

2. ตัดแกนที่ติดกับเฟือง (TAB STOP) ออกโดยแกนนี้มีหน้าที่ป้องกันไม่ให้มอเตอร์หมุนเกินมุม 180 องศาทั้งนี้เพื่อ ป้องกันความเสียหายที่จะเกิดขึ้นกับตัวต้านทานปรับค่าได้เนื่องจาก ตัวต้านทานชนิดปรับค่าได้ ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้นเพื่อให้มอเตอร์หมุนเป็นวงรอบได้จึงต้องตัด TAB STOP ในส่วนนี้ออกดังรูป

รูปที่ 2.22 แกนที่ติดกับเฟือง (TAB STOP)

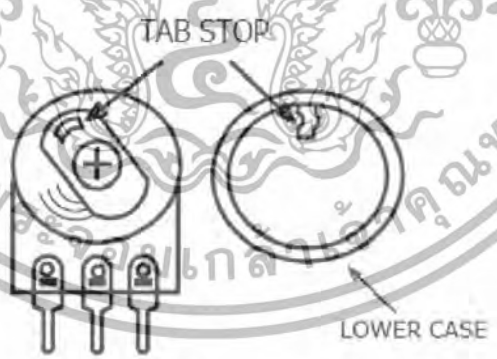
3. ถอดตัวต้านทานปรับค่าได้ (VR) ออก แล้วใส่ตัวต้านทานชนิดค่าคงที่ 2 ตัวที่ต่ออนุกรมกันเข้าไปแทนในตำแหน่งของตัวต้านทานปรับค่าได้ โดยตัวต้านทานชนิดค่าคงที่ที่นำมาต่อนี้จะต้องมีค่าอยู่ในช่วง 2.2 k ถึง 3.3 k ทั้งนี้เนื่องจากตัวต้านทานชนิดปรับค่าได้ที่อยู่ในบอร์ดควบคุมของ Servo motor นั้นจะมีค่าความต้านทาน 5 k ดังนั้น จึงต้องนำตัวต้านทานค่าคงที่มาต่ออนุกรมกันเพื่อให้ได้ค่าความต้านทานใกล้เคียงกับของเดิม ดังรูปต่อไปนี้



รูปที่ 2.23 ตัวต้านทานปรับค่าได้ (VR)

4. ถึงแม้ว่าเราจะถอดตัวต้านทานปรับค่า(VR) ออกจากวงจรแล้วก็ตามแต่เนื่องจากเรายังคงต้องใช้ตัวต้านทานปรับค่าได้นี้ไปเป็นแกนหมุนของมอเตอร์อยู่ ซึ่งตัวต้านทานปรับค่านี้จะไม่สามารถหมุนเป็นวงรอบได้ทำให้เราต้องแก้ไขเปลี่ยนแปลงบางส่วนของตัวต้านทานเพื่อให้ตัวต้านทานสามารถหมุนรอบตัวเองได้เพื่อที่จะได้ไม่ไปขัดขวางการหมุนของมอเตอร์ซึ่งทำได้โดย

-ถอดชิ้นส่วนของตัวต้านทานปรับค่าออก



รูปที่ 2.24 ภายในตัวต้านทานปรับค่า(VR)

- ตัวต้านทานปรับค่าในมอเตอร์แต่ละรุ่นนั้นอาจจะใช้ไม่เหมือนกันแต่จะมีหลักการเดียวกันโดยจะมี แท็ป ที่ทำหน้าที่หยุดการหมุนของตัวต้านทานอยู่ให้เราทำการตัดส่วนนี้ออกแล้วทดลองหมุนแกนของตัวต้านทานปรับค่าถ้าสามารถหมุนรอบตัวเองได้ ก็ทำการประกอบตัวต้านทานเข้าไว้เหมือนเดิมแต่ถ้ายังหมุนเป็นวงรอบไม่ได้ก็ให้พิจารณาดูว่ามีชิ้นส่วนใดที่ยังขัดขวางการหมุนของตัวต้านทานอยู่ เมื่อพบก็ให้เอาออก หรือ ทำลายได้เลยโดยไม่ต้องสนใจว่าจะทำให้ตัวต้านทานนี้พัง

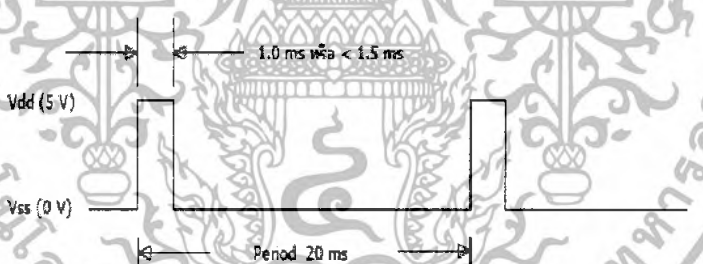
เพราะเราไม่ได้ใช้ประโยชน์จากการเปลี่ยนแปลงค่าความต้านทานนี้อีกแล้วนอกจากใช้เป็นแกนหมุนของเฟืองเท่านั้น

- จากนั้นตัดหรือพับขาของตัวต้านทานปรับค่า (VR) เพื่อป้องกันไม่ให้ขาของตัวต้านทานดังกล่าวไปช็อตกับแผงวงจรควบคุม

5. ประกอบชิ้นส่วนต่างๆ เข้าที่เดิม และ เพื่อความปลอดภัยในการประกอบตัวต้านทานปรับค่า (VR) ลงในกล่องของ Servo motor ควรหาฉนวนรองตรงส่วนของขาที่เป็นโลหะของตัวต้านทานด้วยเพื่อไม่ให้ไปช็อตกับส่วนอื่นๆในแผงวงจรควบคุม เพียงเท่านี้มอเตอร์ของเราก็จะสามารถหมุนเป็นวงรอบ 360 องศาได้แล้วและในการนำไปใช้งานจะต้องระวังเรื่องของโหลดที่นำมาต่อกับมอเตอร์ เพราะหากนำมอเตอร์ไปขับหรือยกโหลดที่มีน้ำหนักมากเกินไปอาจจะทำให้เกิดความเสียหายกับเฟืองหรือเกียร์ต่างๆ ของมอเตอร์ได้

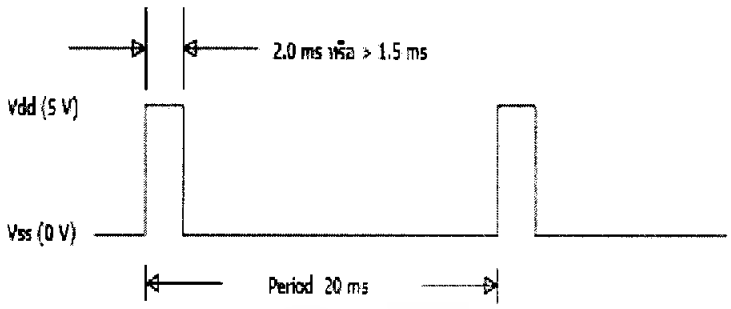
หลังจากเราได้ทำการปรับแต่งการทำงานของเซอร์โวมอเตอร์ให้สามารถหมุนเป็นวงรอบได้แล้ว วิธีในการควบคุมให้มอเตอร์หมุน จะมีลักษณะดังนี้

- การควบคุมให้มอเตอร์หมุนทางด้านซ้ายจะต้องป้อนสัญญาณพัลส์ที่มีขนาดความกว้างพัลส์ 1 ms หรือ ให้น้อยกว่า 1.5 ms โดยจะต้องป้อนสัญญาณพัลส์นี้ทุกๆ 20 ms (หรือในช่วงประมาณ 20ms – 30ms)



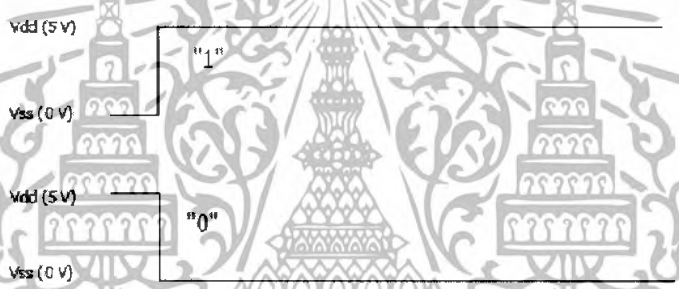
รูปที่ 2.25 การควบคุมให้มอเตอร์หมุนทางด้านซ้าย

- การควบคุมให้มอเตอร์หมุนทางด้านขวาจะต้องป้อนสัญญาณพัลส์ที่มีขนาดความกว้างพัลส์ 2 ms หรือ ไม่ต่ำกว่า 1.5 ms และจะต้องป้อนสัญญาณพัลส์นี้ทุกๆ 20 ms (หรือในช่วงประมาณ 20ms – 30ms) เช่นกัน



รูปที่ 2.26 การควบคุมให้มอเตอร์หมุนทางด้านขวา

การควบคุมให้มอเตอร์หยุดหมุน ทำได้โดยการส่งลอจิก “0” หรือ “1” ให้กับมอเตอร์ หรือ ก็คือการไม่จ่ายสัญญาณพัลส์ให้กับมอเตอร์นั่นเอง

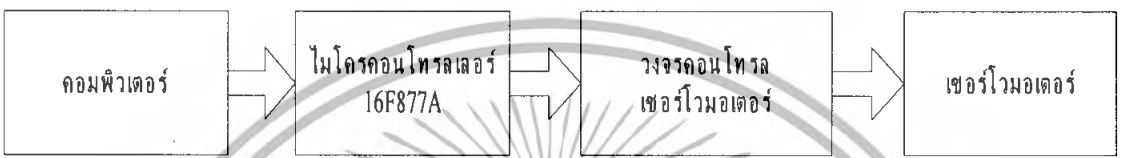


รูปที่ 2.27 การควบคุมให้มอเตอร์หยุดหมุน

บทที่ 3
การออกแบบและการสร้าง

3.1 โครงสร้างการทำงานของน้ำพุ

โครงสร้างการทำงานของน้ำพุ สามารถแบ่งได้ 4 ส่วนใหญ่ ดังรูป

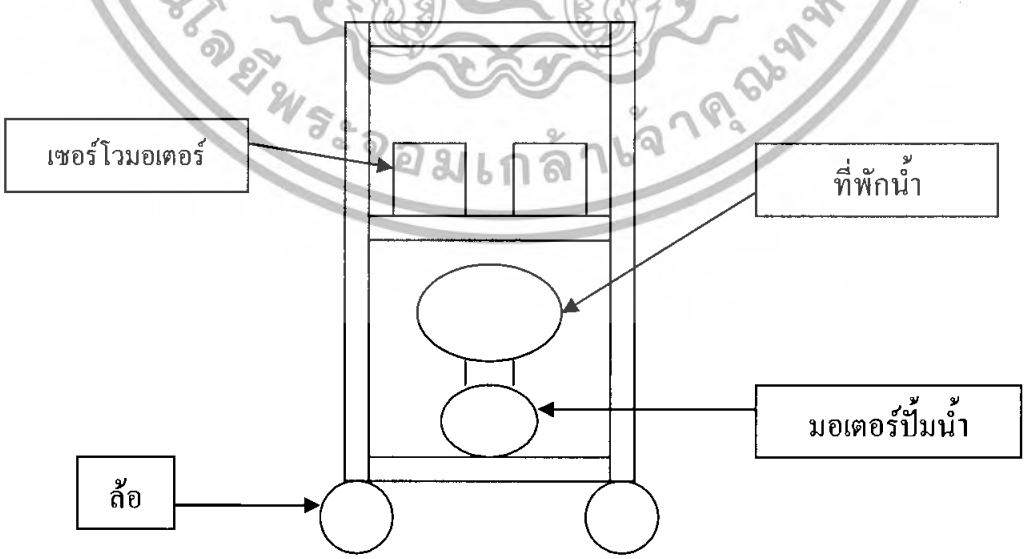


รูปที่ 3.1 แสดงโครงสร้างการทำงาน ของหุ่นยนต์

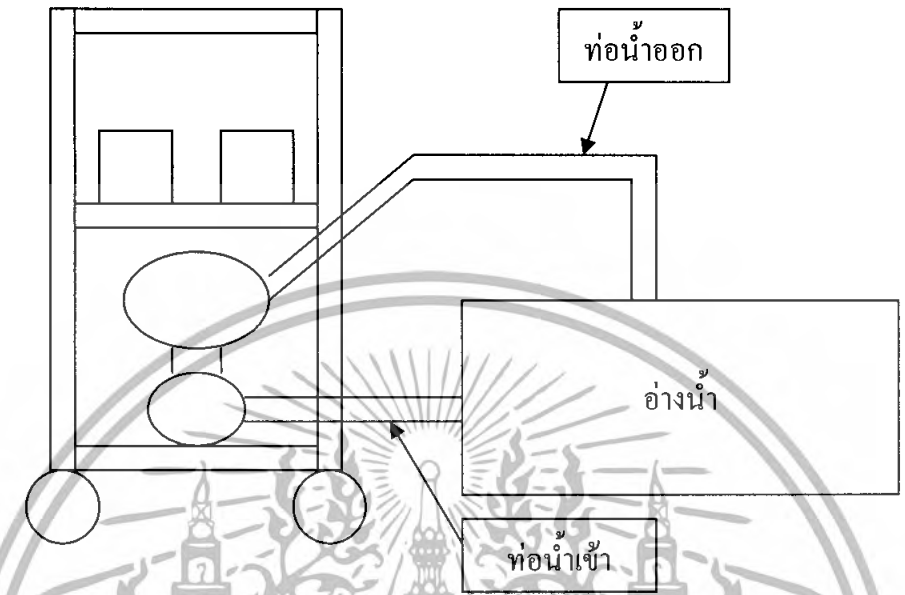
จากรูปที่ 3.1 คอมพิวเตอร์จะทำหน้าที่ป้อนข้อมูลลงบนไมโครคอนโทรลเลอร์ 16F877A และส่งข้อมูลจากคอมพิวเตอร์ โดยใช้พอร์ทอนุกรม ไปยังไมโครคอนโทรลเลอร์ และประมวลผลข้อมูลเพื่อส่งไปที่วงจรคอนโทรลเซอร์โวมอเตอร์

3.2 การออกแบบน้ำพุและโครงสร้าง

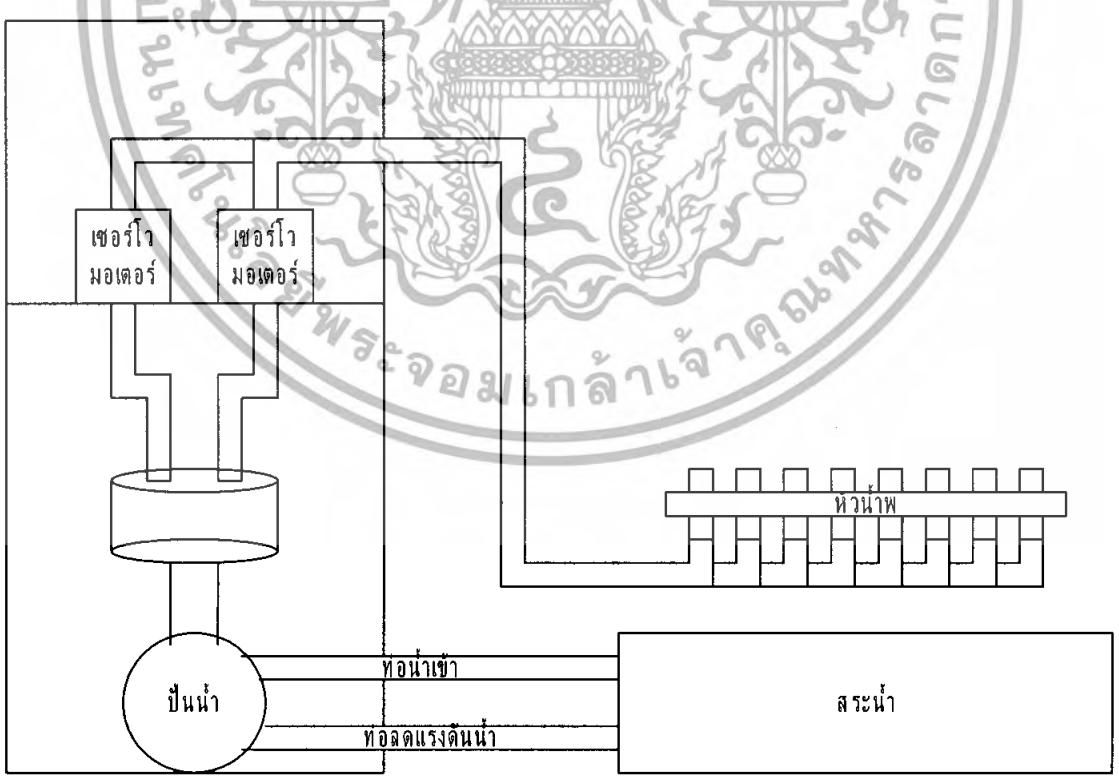
3.2.1 โครงสร้างของตัวแท่นน้ำพุ



รูปที่ 3.2 มุมมองด้านหน้า



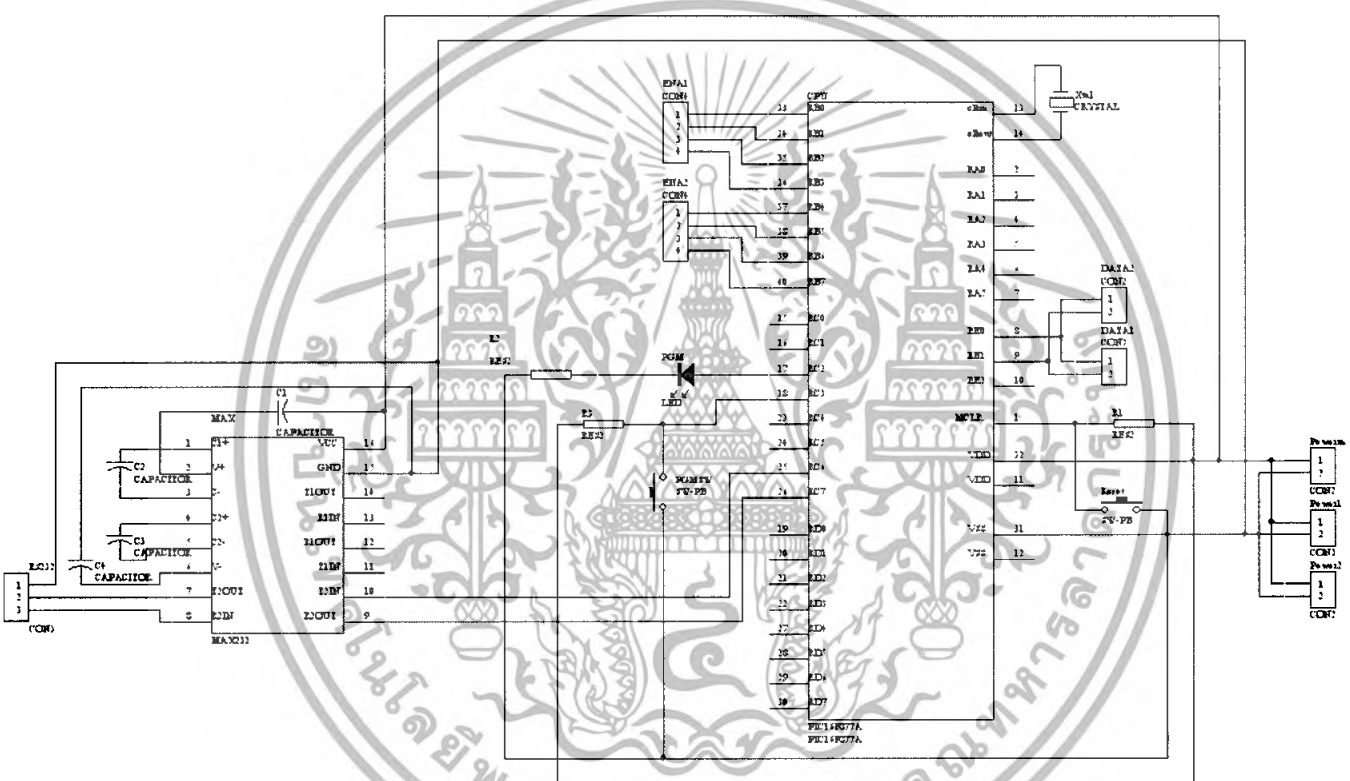
รูปที่ 3.3 การส่งน้ำ



รูปที่ 3.4 การเปิด-ปิดของเซอร์โวมอเตอร์เพื่อส่งน้ำไปยังหัวน้ำพุ

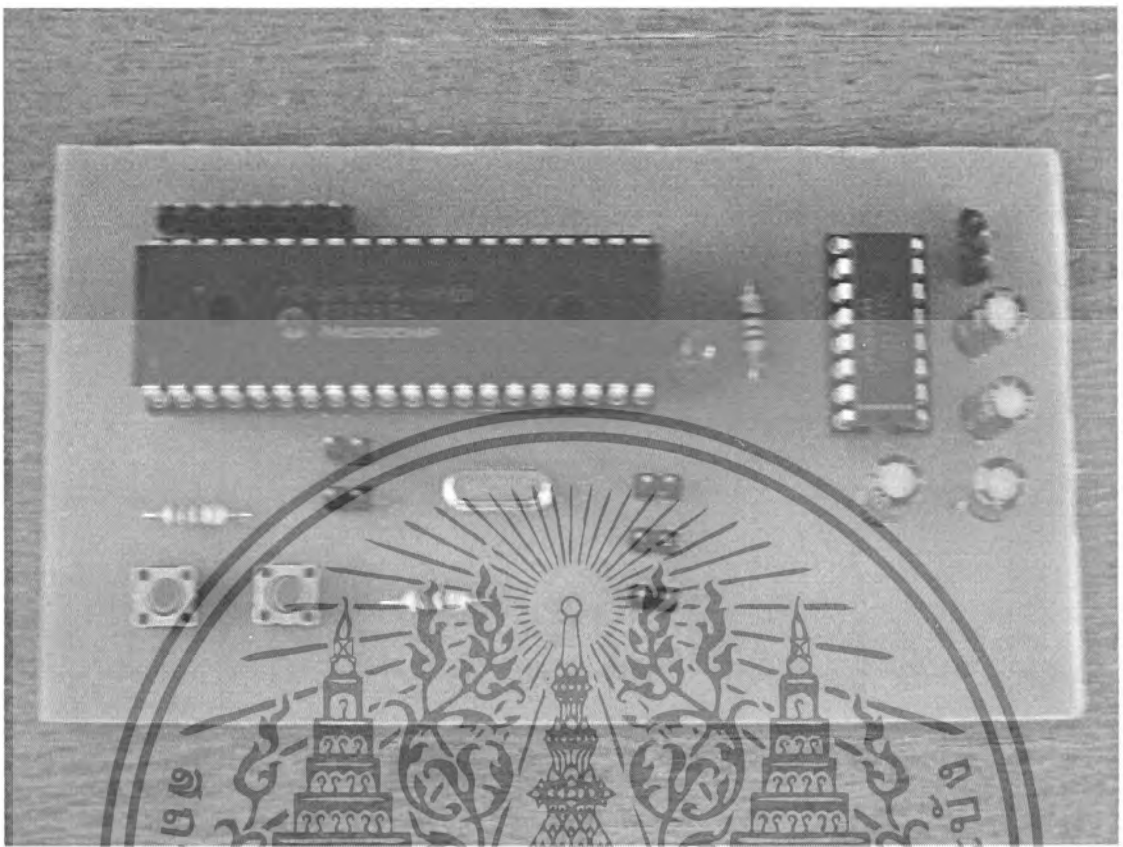
การทำงานเมื่อเปิดปั้มน้ำเพื่อสูบน้ำเข้าปั้ม จะอัดแรงดันน้ำขึ้นไปที่พักน้ำจะมีท่อระบายน้ำจากที่พักน้ำมายังอ่างน้ำ เพื่อที่จะระบายน้ำเมื่อวาล์วทุกวาล์วปิดหมด แรงดันน้ำจากปั้มจะส่งไปที่วาล์วและเซอร์โวมอเตอร์จะควบคุมการเปิด-ปิดวาล์วน้ำ ที่จะออกทางหัวน้ำพุมีอยู่ 4 ระดับ คือ ระดับ 1 คือ ปิดวาล์ว , ระดับ 2 คือ เปิดวาล์ว 25% , ระดับ 3 คือ เปิดวาล์ว 75% , ระดับ 4 คือ เปิดวาล์ว 100%

3.3 การออกแบบภาคควบคุมน้ำพุ (PIC16F877A)

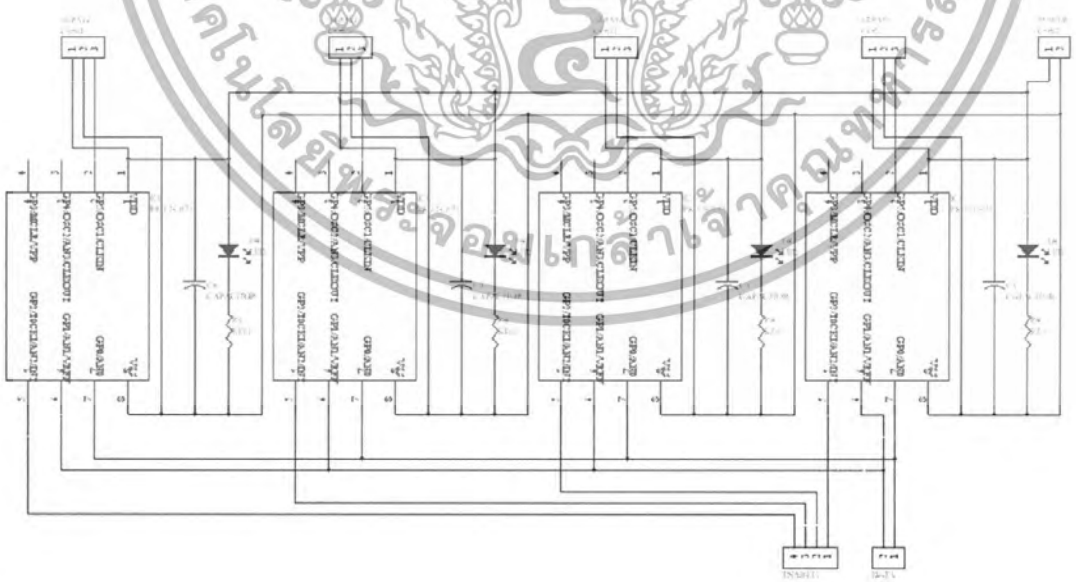


รูปที่ 3.5 ไมโครคอนโทรลเลอร์ 16F877A

ไมโครคอนโทรลเลอร์ 16F877A จะเป็นตัวประมวลผลข้อมูลจาก EEPROM เพื่อออกไปยังวงจรควบคุมเวอร์โวมอเตอร์ ข้อมูลจะส่งแบบขนาน ซึ่งจะมี ข้อมูล(DATA) และ การเปิดวงจรควบคุม (ENABLED)



รูปที่ 3.6 บอร์ดไมโครคอนโทรลเลอร์ 16F877A

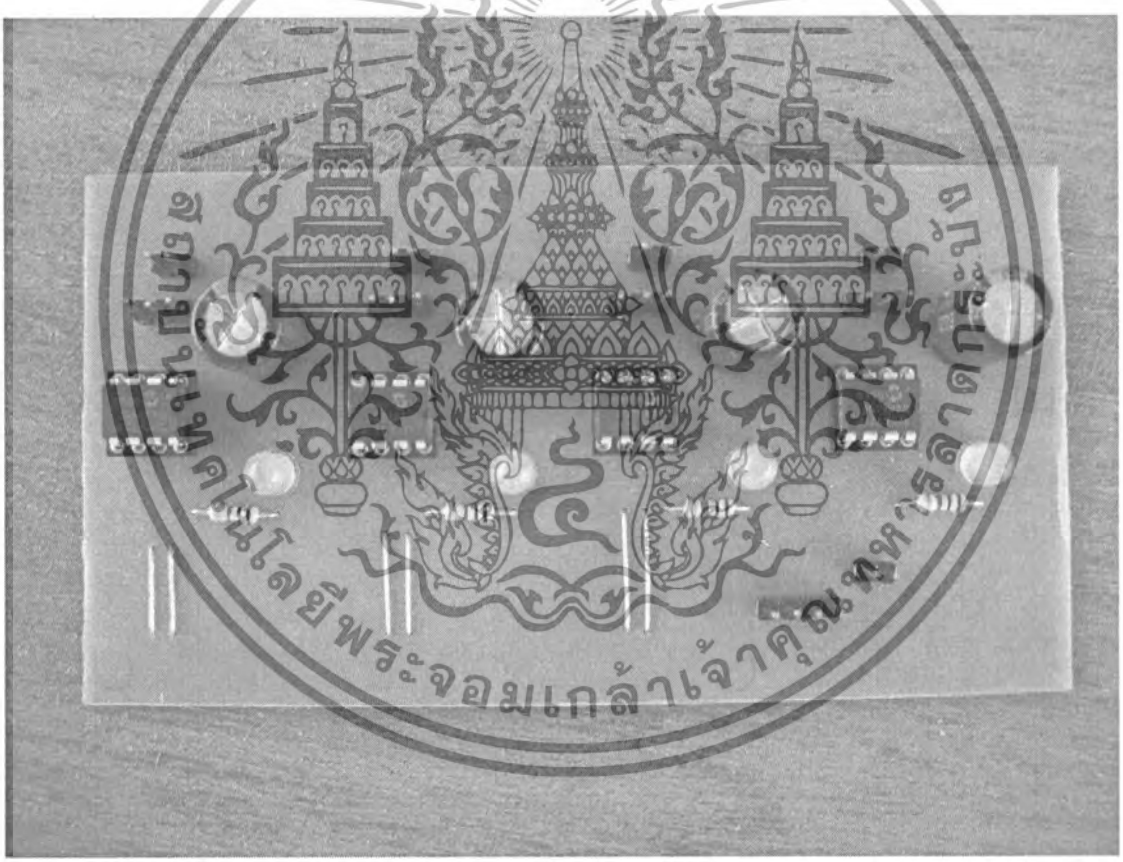


รูปที่ 3.7 วงจรควบคุมเซอร์โวมอเตอร์

รับข้อมูลจากไมโครคอนโทรลเลอร์ 16F877A เพื่อนำมาสร้างพัลส์วิดมอดไปจ่ายสัญญาณให้กับเซอร์โวมอเตอร์

การทำงานของข้อมูล(DATA) จะมีความสัมพันธ์กันกับการสร้างพัลส์ ดังนี้

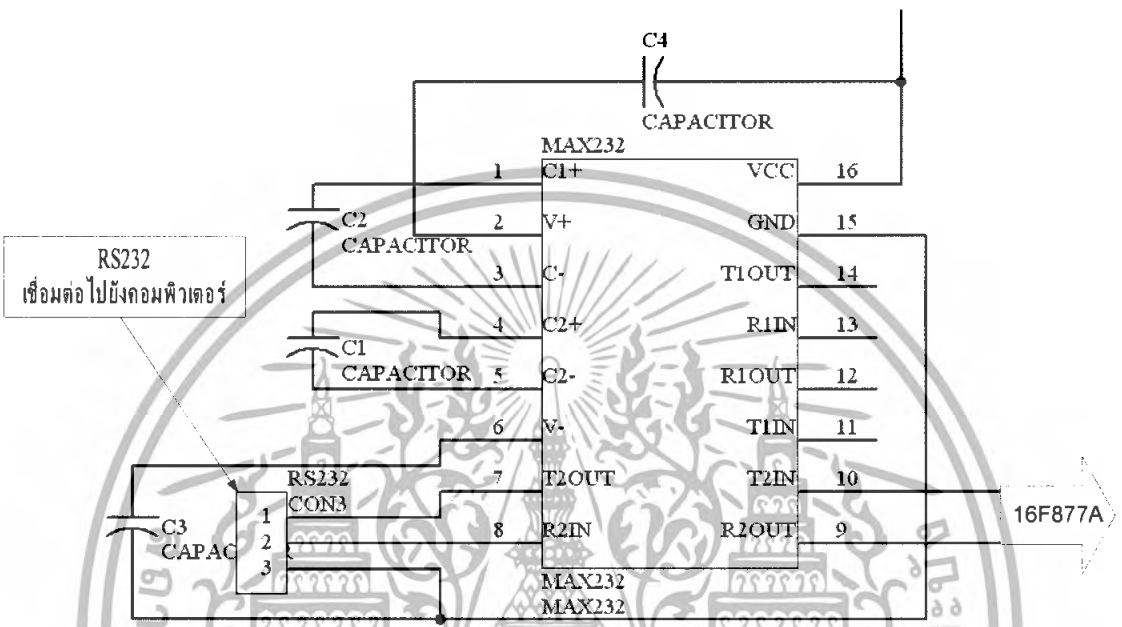
1. เมื่อข้อมูล(DATA) 100 จะสร้างพัลส์ขนาด 0.45 ms
2. เมื่อข้อมูล(DATA) 101 จะสร้างพัลส์ขนาด 0.8 ms
3. เมื่อข้อมูล(DATA) 110 จะสร้างพัลส์ขนาด 1.1 ms
4. เมื่อข้อมูล(DATA) 111 จะสร้างพัลส์ขนาด 1.4 ms



รูปที่ 3.8 บอร์ดวงจรควบคุมเซอร์โวมอเตอร์

3.4 การออกแบบการส่งข้อมูลจากคอมพิวเตอร์

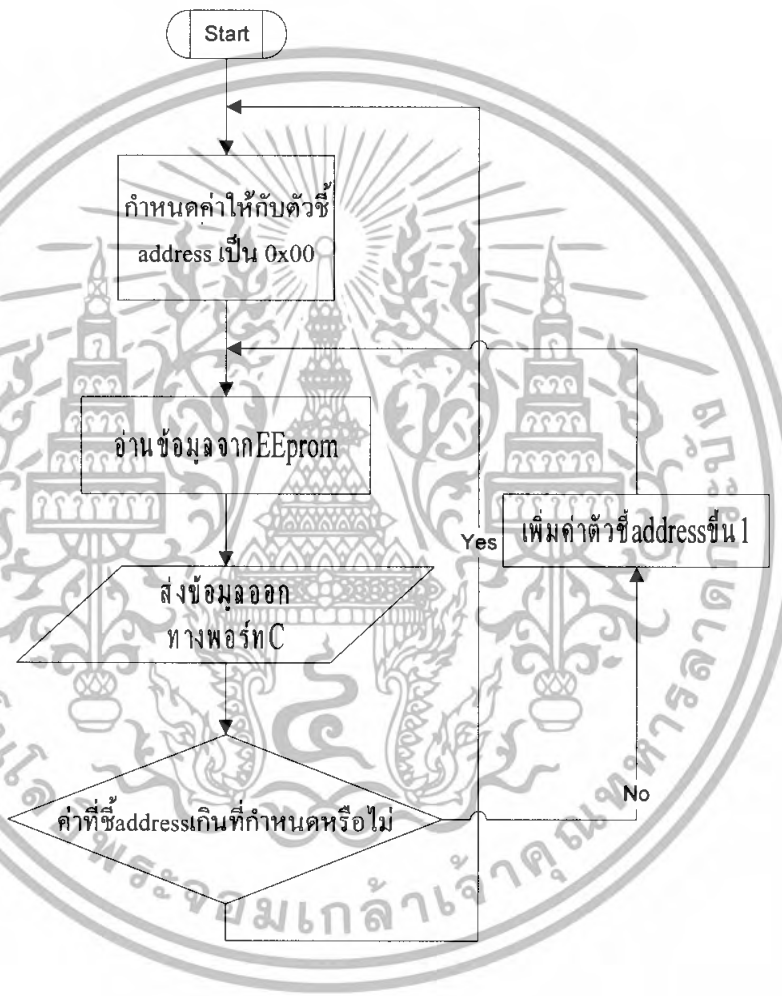
การส่งข้อมูลจากคอมพิวเตอร์จะส่งผ่าน COM1 เชื่อมต่อกับ RS-232 แบบอนุกรม โดยการโปรแกรมจากคอมพิวเตอร์ส่งข้อมูลมาเป็น ตัวอักษร



รูปที่ 3.7 วงจรการส่งข้อมูลจาก COM1 ไปยัง RS-232

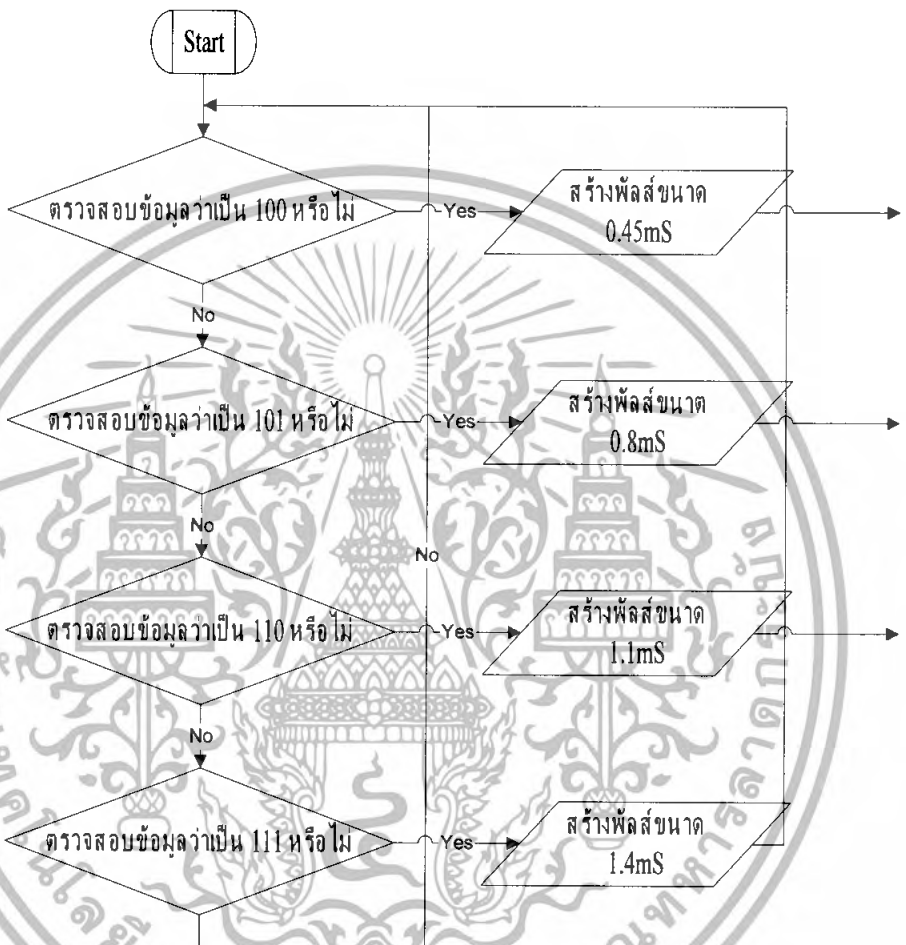
3.5 การทำงานของโปรแกรม

เมื่อเริ่มต้นไมโครคอนโทรลเลอร์ จะอ่านข้อมูลจาก EEPROM โดยมีการชี้ตัวแปรผ่านทาง ADDRESS แล้วจึงส่งข้อมูลออกทาง PORTC ส่งข้อมูลไปเรื่อยๆจนกว่าข้อมูลที่ถูกโปรแกรมภายในตัวไมโครคอนโทรลเลอร์ จะหมด ถ้าข้อมูลใน EEPROM หมดตัวไมโครคอนโทรลเลอร์ก็จะเริ่มวนรอบการส่งข้อใหม่ไปเรื่อยๆ



รูปที่ 3.6 โฟลว์ชาร์ตแสดงการควบคุมไมโครคอนโทรลเลอร์ PIC16F877A

การรับข้อมูลจากตัวไมโครคอนโทรลเลอร์ตรวจสอบข้อมูล(DATA) แล้วทำการสร้างพัลส์วิดมอดแบบต่างๆที่เซอร์โวจะเปิด-ปิด



รูปที่ 3.6 โฟลว์ชาร์ตแสดงวงจรการควบคุมเซอร์โว

บทที่ 4
ผลการทดลอง

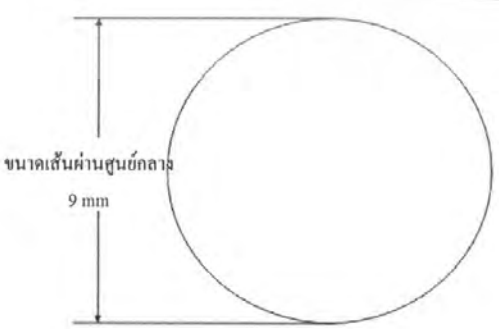
4.1 ทดสอบการใช้เซอร์โวมอเตอร์ควบคุมการเปิดปิดวาล์ว

4.1.1 การคำนวณหาพื้นที่หน้าวาล์ว

จากการคำนวณหาพื้นที่หน้าวาล์วค้าที่ได้จะเป็นค่าโดยประมาณซึ่งก็เป็นค่าที่ใกล้เคียงกับความเป็นจริงโดยใช้วิธีแทนพื้นที่ว่าที่ไม่เป็นเรขาคณิตด้วยรูปสามเหลี่ยมและหาพื้นที่ของรูปสามเหลี่ยม

- ขณะเปิดวาล์ว 100%

รูปที่ 4.1 วาล์วเปิด 100%

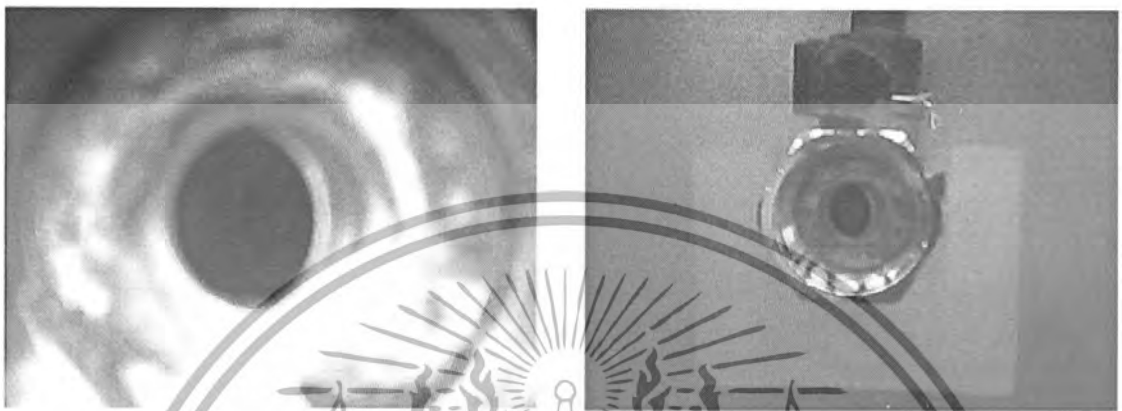


พื้นที่หน้าวาล์ว $= \pi r^2$

$r = 4.5mm$

พื้นที่หน้าวาล์ว $= 63.617mm^2$

- ขณะเปิดวาล์ว 75%



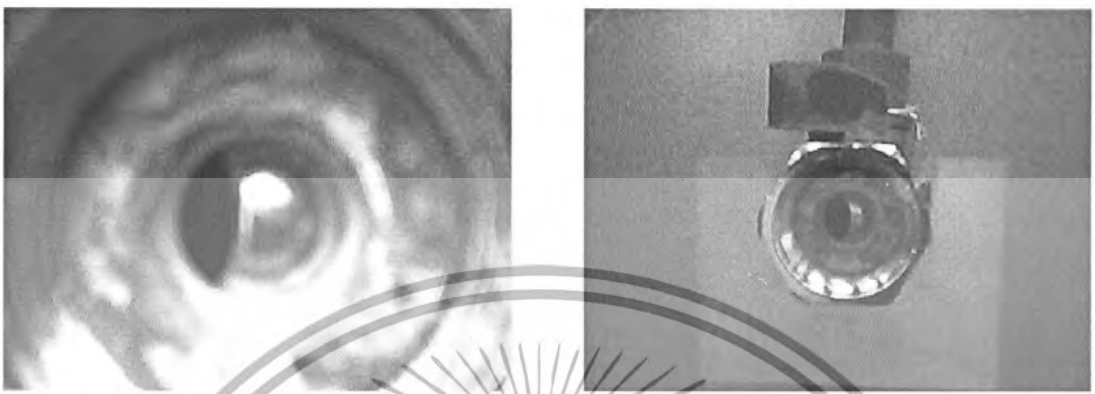
รูปที่ 4.2 วาล์วเปิด 75%

$$\approx \frac{\pi r^2}{2} + \left(\frac{1}{2} \times Base \times High\right) + 2\left(\frac{1}{6} \times \left(\pi \times r^2\right) - \left(\frac{1}{2} \times Base \times High\right)\right)$$

$$\approx \frac{\pi \times 4.5^2}{2} + \left(\frac{1}{2} \times 9mm \times 2mm\right) + 2\left(\frac{1}{6} \times \left(\pi \times 4.5^2\right) - \left(\frac{1}{2} \times 9mm \times 2mm\right)\right)$$

$$\approx 44.014mm^2$$

- ขณะเปิดวาล์ว 25%



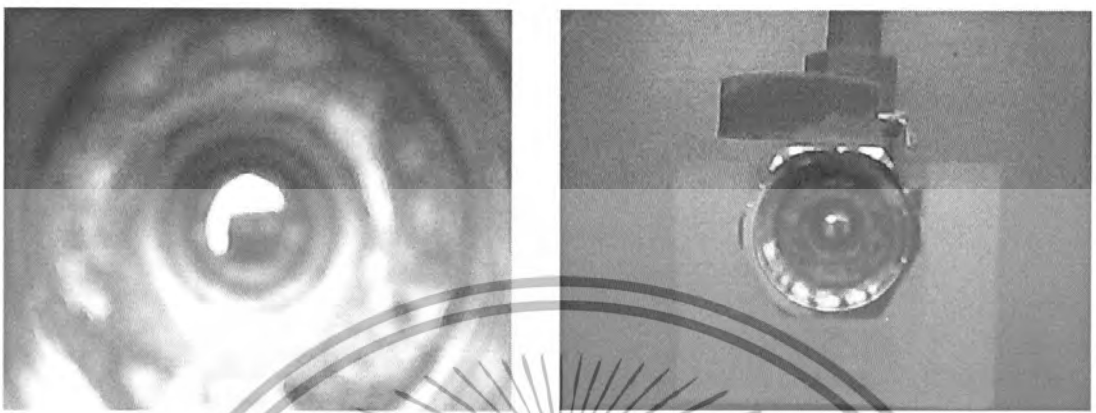
รูปที่ 4.3 วาล์วเปิด 25%

$$\text{พื้นที่หน้าวาล์ว} \approx \left(\frac{1}{2}\times Base_1\times High_1\right)+2\left(\frac{1}{6}\times\left(\pi\times r^2\right)-\left(\frac{1}{2}\times Base_1\times High_1\right)\right)$$
$$+\left(\frac{1}{2}\times Base_2\times High_2\right)+2\left(\frac{1}{6}\times\left(\pi\times r^2\right)-\left(\frac{1}{2}\times Base_2\times High_2\right)\right)$$

$$\approx\left(\frac{1}{2}\times 7.29mm\times 1.7mm\right)+2\left(\frac{1}{6}\times\left(\pi\times 4.5^2\right)-\left(\frac{1}{2}\times 7.29mm\times 1.7mm\right)\right)$$
$$+\left(\frac{1}{2}\times 7.29mm\times 1.216mm\right)+2\left(\frac{1}{6}\times\left(\pi\times 4.5^2\right)-\left(\frac{1}{2}\times 7.29mm\times 1.216mm\right)\right)$$

$$\approx 33.914mm^2$$

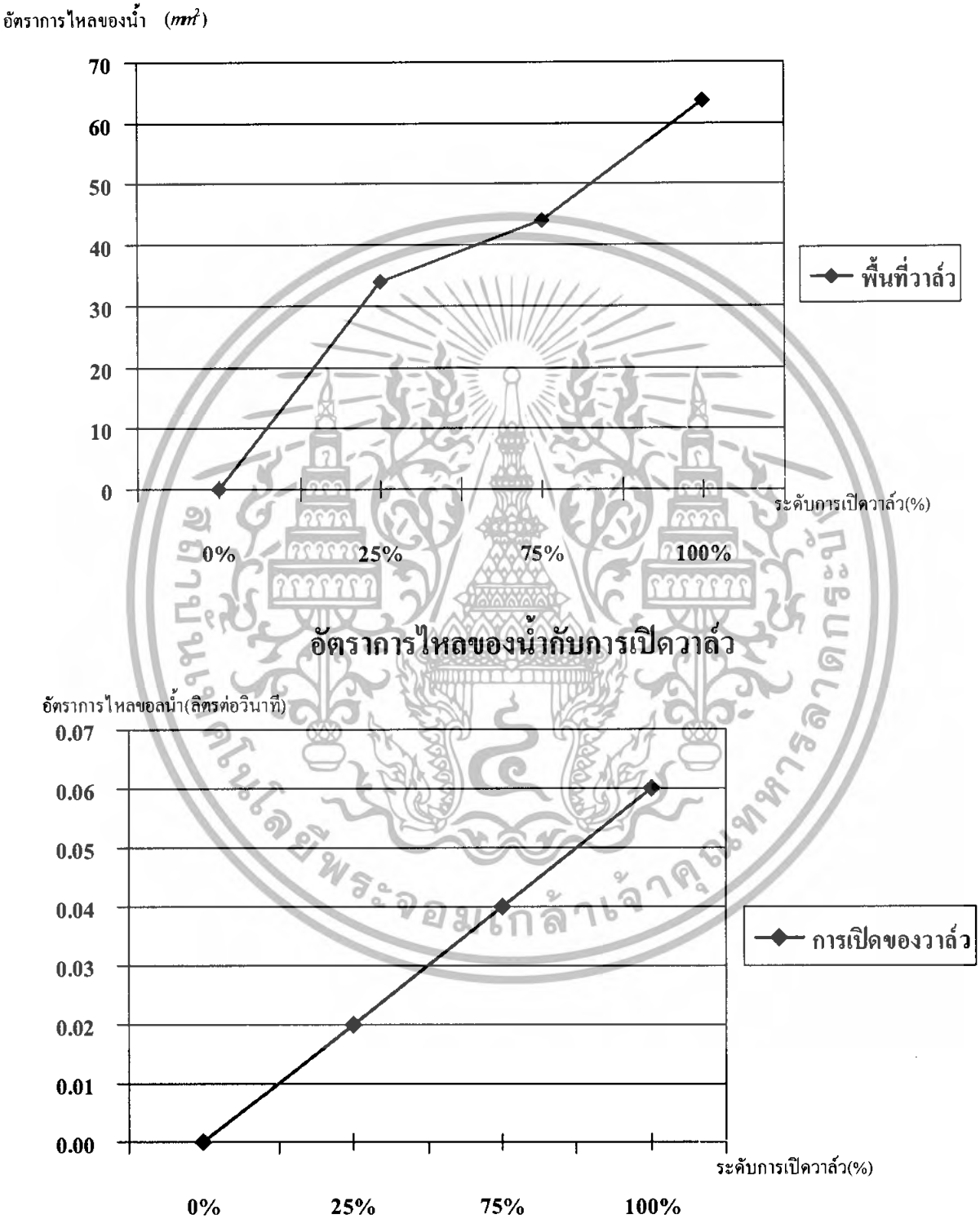
- ขณะปิดวาล์ว



รูปที่ 4.4 วาล์วปิด

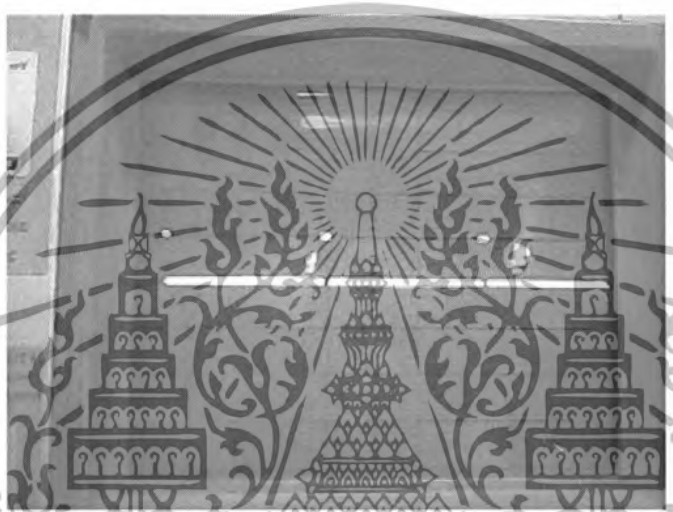
พื้นที่หน้าวาล์ว $= 0 mm^2$

พื้นที่หน้าวาล์วกับการเปิดวาล์ว



4.1.2 ทดสอบการควบคุมเซอร์โวมอเตอร์ให้หมุนตามที่ต้องการ

จากการทดสอบพบว่าในการควบคุมให้เซอร์โวมอเตอร์หมุนซ้ายหรือขวาจะเป็นการควบคุมช่วงของพัลส์ซีกบวกโดยเริ่มต้นที่ 0 องศา จะต้องสร้างสัญญาณที่มีช่วงของพัลส์บวกขนาด 0.5 mS และเมื่อเพิ่มความกว้างของพัลส์ไปเรื่อยๆจนถึง 1.5 mS เซอร์โวมอเตอร์ก็จะหมุนไปที่ตำแหน่ง 90 องศา พอดี



รูปที่ 4.5 สัญญาณที่มีความกว้างของช่วงบวก 0.5 mS

รูปที่ 4.6 สัญญาณที่มีความกว้างของช่วงบวก 0.8 mS

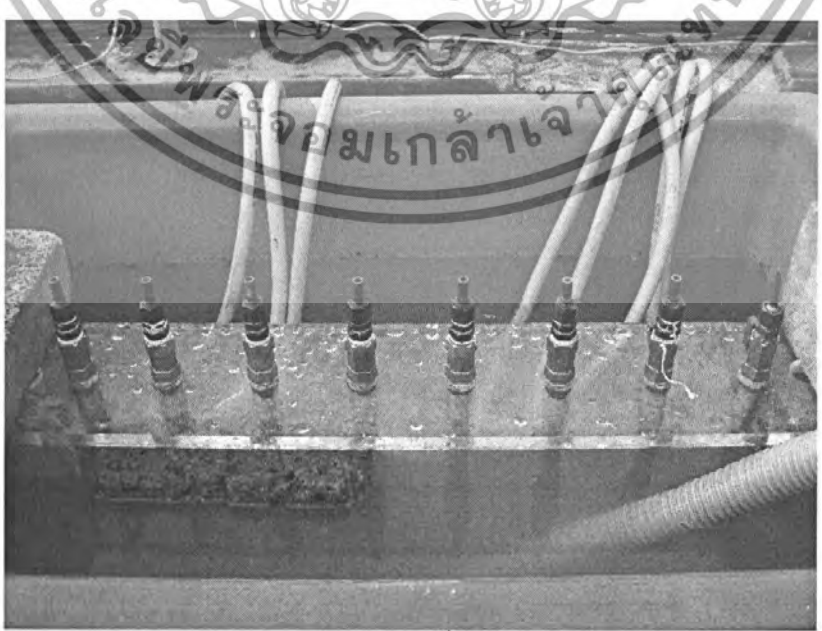
รูปที่ 4.7 สัญญาณที่มีความกว้างของช่วงบวก 1.1 mS

รูปที่ 4.8 สัญญาณที่มีความกว้างของช่วงบวก 1.4 mS

4.2 ทดสอบการทำงานโดยมีการจ่ายน้ำ

จากการทดสอบเดินปั้มน้ำเพื่อจ่ายน้ำให้กับระบบส่วนของตัวควบคุมยังทำงานได้ดีโดยสามารถปรับระดับของความสูงของน้ำพุได้ด้วยการปรับระดับการเปิดปิดหน้าวาล์วน้ำ

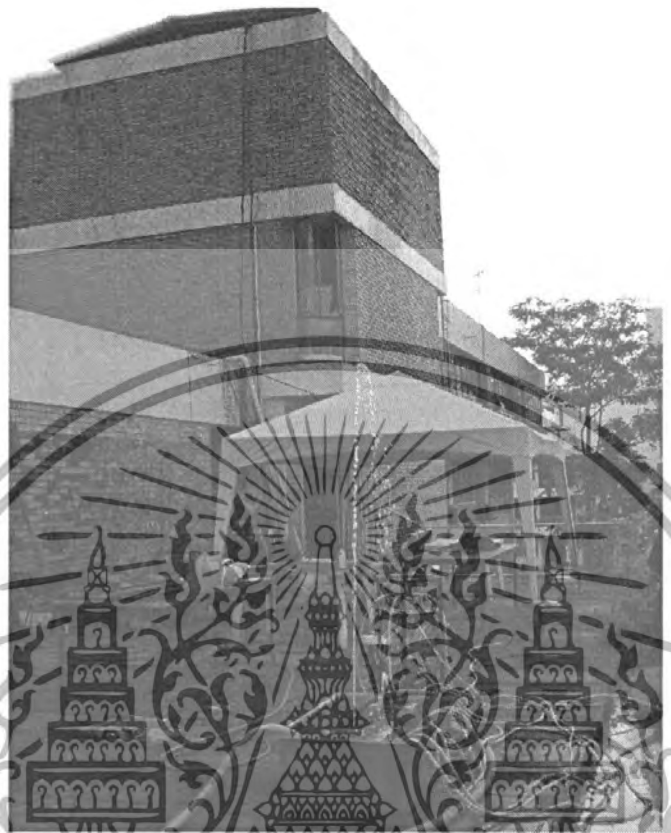
รูปที่ 4.9 ชุดเซอร์โวควบคุมน้ำพุ



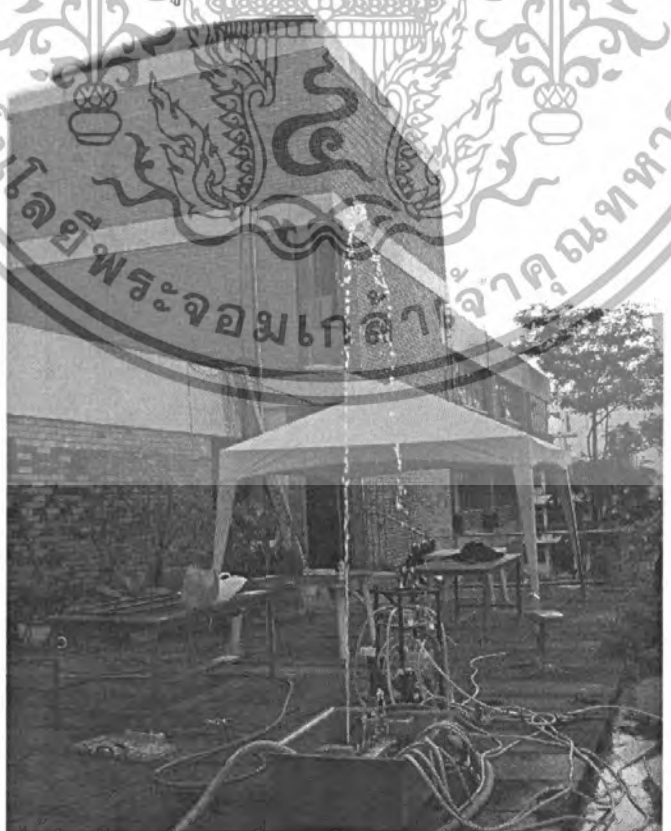
รูปที่ 4.10 หัวน้ำพุ 8 หัว

รูปที่ 4.11 หัวน้ำพุ

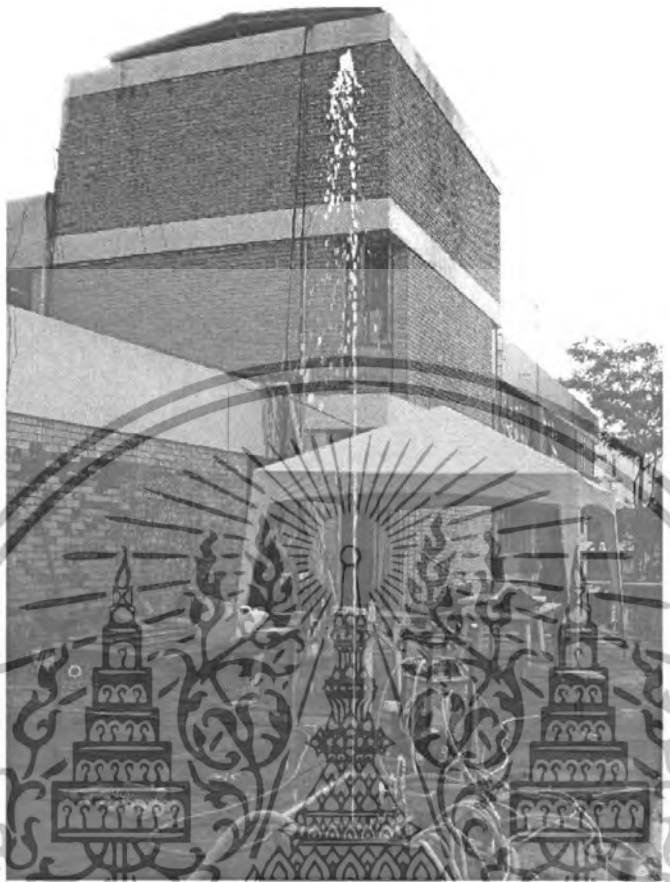
รูปที่ 4.12 ขณะปิดวาล์ว



รูปที่ 4.13 ขณะวาล์วเปิด 25%

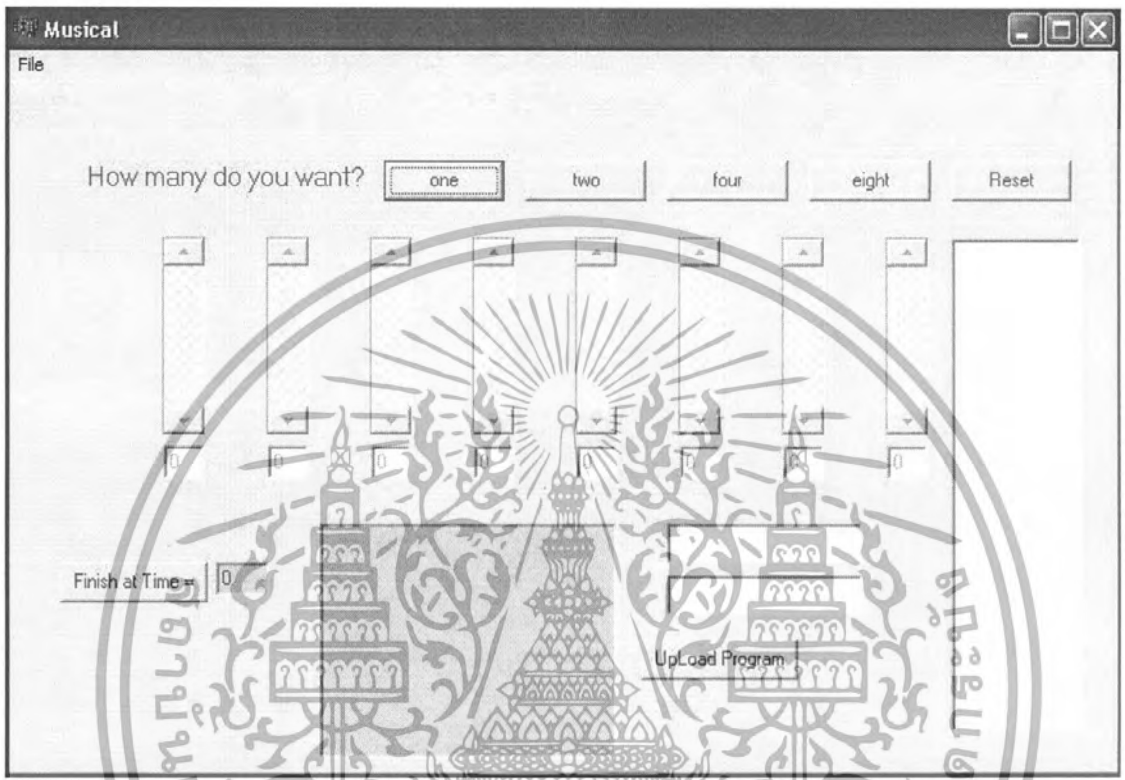


รูปที่ 4.14 ขณะวาล์วเปิด 75%



รูปที่ 4.15 ขณะวาล์วเปิด 100%

4.3 รูปโปรแกรมบนคอมพิวเตอร์



รูปที่ 4.16 โปรแกรมป้อนข้อมูล

บทที่ 5
บทสรุป

โครงงานนี้ทำขึ้นเพื่อศึกษาและประยุกต์ใช้งานไมโครคอนโทรลเลอร์ในงานควบคุมโดยมีโปรแกรมที่ใช้ในการติดต่อกับผู้ใช้งานเป็น C++ Builder เพื่อให้ผู้ที่ใช้ได้รับความสะดวกในการใช้งานโดยผู้ใช้จะทำการโปรแกรมจังหวะการขึ้นลงของน้ำพุได้ตามต้องการแล้วส่งรูปแบบที่ได้ไปเก็บไว้ในEEPROMเพื่อไม่ต้องทำการโปรแกรมใหม่ทุกครั้งที่เริ่มต้นใช้งานระบบแต่หากต้องการเปลี่ยนแปลงรูปแบบก็สามารถทำการโปรแกรมข้อมูลของจังหวะลงไปใหม่ได้

จากการทำโครงงานสามารถสรุปโครงงานปัญหาที่ได้พบและวิธีแก้ไขในการทำโครงงานได้เป็นส่วนๆดังต่อไปนี้

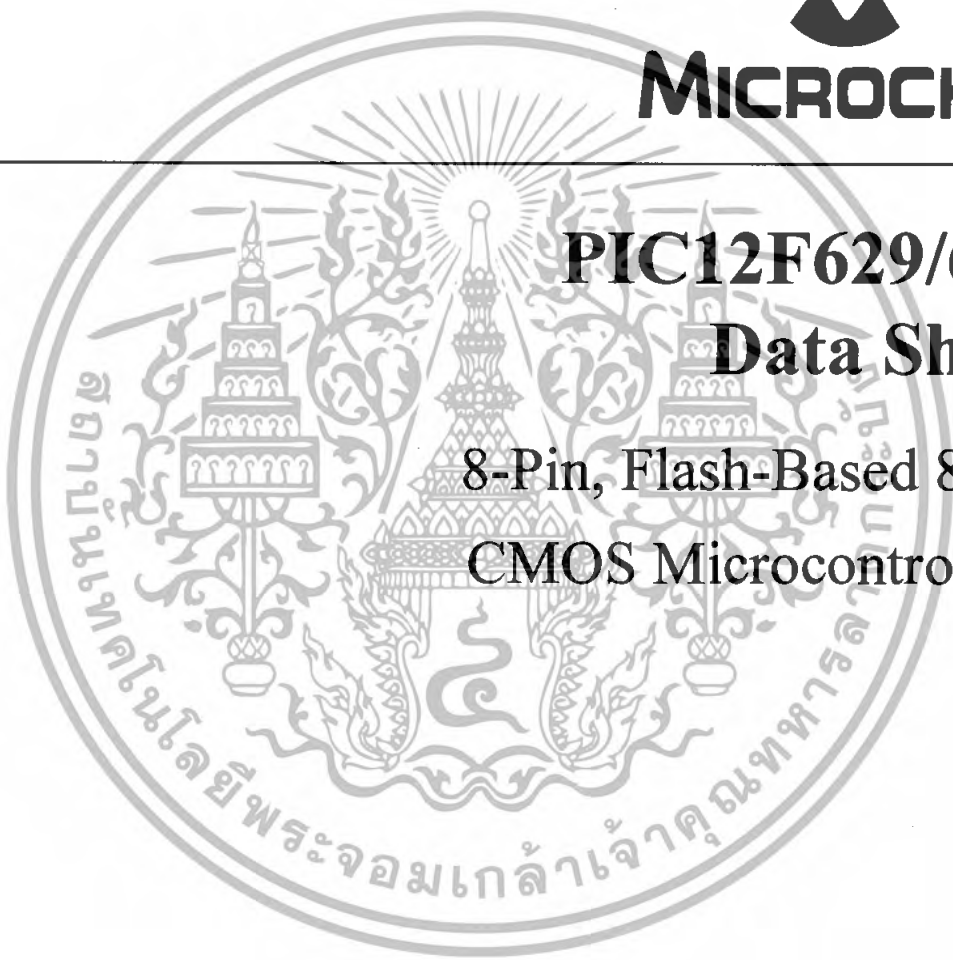
การทำงานส่วนของโปรแกรมที่ใช้ในการติดต่อกับผู้ใช้ได้ถูกออกแบบมาให้ใช้งานได้โดยง่ายไม่ซับซ้อนผู้ใช้สามารถเข้าถึงได้โดยง่ายแต่อาจจะมีรูปแบบไม่สวยงามมากนักจะต้องได้รับการพัฒนารูปแบบให้สวยงามมากขึ้น

การทำงานส่วนของโปรแกรมที่ใช้ในการเก็บข้อมูลจังหวะของน้ำพุเป็นโปรแกรมที่ค่อนข้างซับซ้อนหากต้องการแก้ไขส่วนใดส่วนหนึ่งจะต้องตรวจสอบใหม่ทั้งหมดเพราะแต่ละส่วนมีความสัมพันธ์กันจึงต้องใช้เวลามากในการพัฒนาโปรแกรม

ถึงแม้โครงงานนี้จะไม่สามารถทำงานตามจังหวะของดนตรีได้ตามที่ตั้งใจไว้แต่ก็สามารถที่จะพัฒนาต่อให้ระบบมีความสามารถสูงขึ้นได้โดยอาจจำเป็นต้องปรับเปลี่ยนอุปกรณ์บางตัวเพื่อให้รองรับกับการประมวลผลที่มีความเร็วสูงได้

เอกสารอ้างอิง

1. คอนสัน ปงผาบ, ทิพย์วัลย์ คำน้ำนอง, ไมโครคอนโทรลเลอร์ PIC และการประยุกต์ใช้งาน, กรุงเทพ, สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2550
2. ประจิน พลังสันติสกุล, เรียนรู้และใช้งาน CCS C คอมไพล์เลอร์เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์ PIC , กรุงเทพฯ:บริษัทอินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด
3. นุกูล กระจาย, การเขียนโปรแกรมแบบวิชวลด้วย C++ Builder 5, สุวีริยาสาสน์, กรุงเทพฯ, 2544



PIC12F629/675 Data Sheet

8-Pin, Flash-Based 8-Bit CMOS Microcontrollers

***8-bit, 8-pin Devices Protected by Microchip's Low Pin Count Patent: U.S. Patent No. 5,847,450. Additional U.S. and foreign patents and applications may be issued or pending.**

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV

ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

PIC12F629/675

8-Pin FLASH-Based 8-Bit CMOS Microcontroller

High Performance RISC CPU:

- Only 35 instructions to learn
  - All single cycle instructions except branches
- Operating speed:
  - DC - 20 MHz oscillator/clock input
  - DC - 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect, and Relative Addressing modes

Special Microcontroller Features:

- Internal and external oscillator options
  - Precision Internal 4 MHz oscillator factory calibrated to ±1%
  - External Oscillator support for crystals and resonators
  - 5 µs wake-up from SLEEP, 3.0V, typical
- Power saving SLEEP mode
- Wide operating voltage range - 2.0V to 5.5V
- Industrial and Extended temperature range
- Low power Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Detect (BOD)
- Watchdog Timer (WDT) with independent oscillator for reliable operation
- Multiplexed MCLR/Input-pin
- Interrupt-on-pin change
- Individual programmable weak pull-ups
- Programmable code protection
- High Endurance FLASH/EEPROM Cell
  - 100,000 write FLASH endurance
  - 1,000,000 write EEPROM endurance
  - FLASH/Data EEPROM Retention: > 40 years

Low Power Features:

- Standby Current:
  - 1 nA @ 2.0V, typical
- Operating Current:
  - 8.5 µA @ 32 kHz, 2.0V, typical
  - 100 µA @ 1 MHz, 2.0V, typical
- Watchdog Timer Current
  - 300 nA @ 2.0V, typical
- Timer1 oscillator current:
  - 4 µA @ 32 kHz, 2.0V, typical

Peripheral Features:

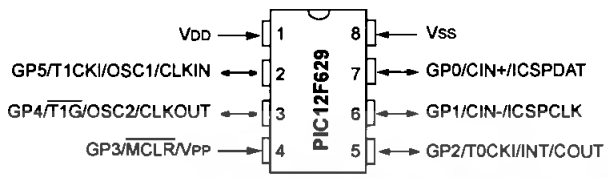
- 6 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - One analog comparator
  - Programmable on-chip comparator voltage reference (CV_{REF}) module
  - Programmable input multiplexing from device inputs
  - Comparator output is externally accessible
- Analog-to-Digital Converter module (PIC12F675):
  - 10-bit resolution
  - Programmable 4-channel input
  - Voltage reference input
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator, if INTOSC mode selected
- In-Circuit Serial Programming™ (ICSP™) via two pins

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)				
PIC12F629	1024	64	128	6	–	1	1/1
PIC12F675	1024	64	128	6	4	1	1/1

* 8-bit, 8-pin devices protected by Microchip's Low Pin Count Patent: U.S. Patent No. 5,847,450. Additional U.S. and foreign patents and applications may be issued or pending.

Pin Diagrams

8-pin PDIP, SOIC, DFN-S



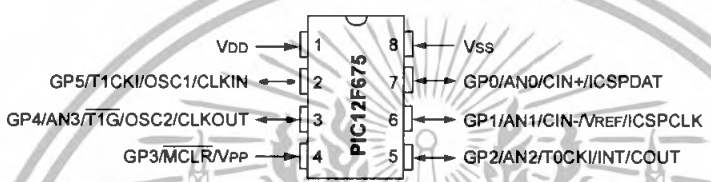


Table of Contents

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com** to receive the most current information on all of our products.

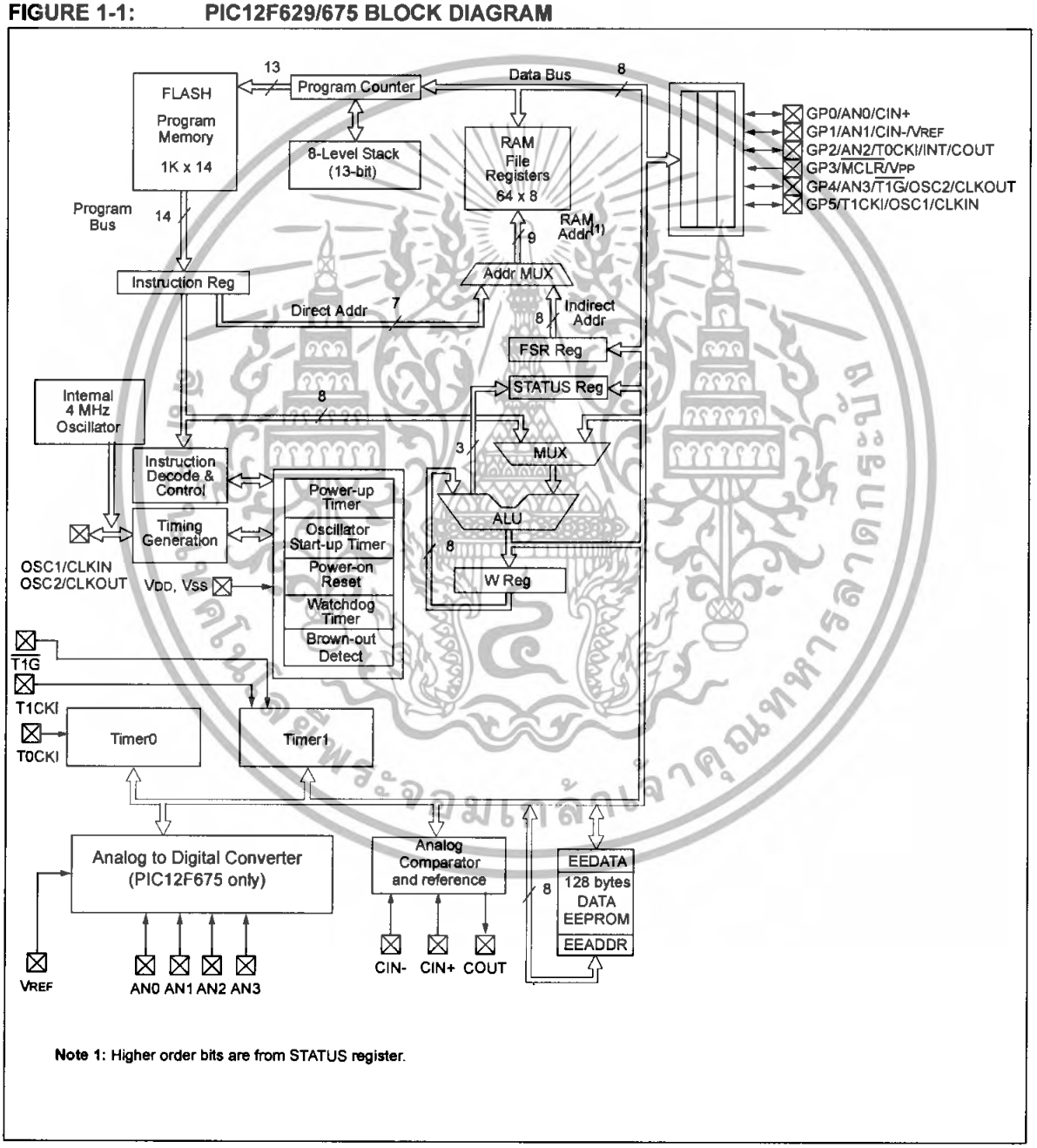
NOTES:

1.0 DEVICE OVERVIEW

This document contains device specific information for the PIC12F629/675. Additional information may be found in the PIC® Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The Reference Manual should be considered a complementary document to this Data Sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC12F629 and PIC12F675 devices are covered by this Data Sheet. They are identical, except the PIC12F675 has a 10-bit A/D converter. They come in 8-pin PDIP, SOIC, and MLF-S packages. Figure 1-1 shows a block diagram of the PIC12F629/675 devices. Table 1-1 shows the Pinout Description.

FIGURE 1-1: PIC12F629/675 BLOCK DIAGRAM



Note 1: Higher order bits are from STATUS register.

TABLE 1-1: PIC12F629/675 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
GP0/AN0/CIN+/ICSPDAT	GP0	TTL	CMOS	Bi-directional I/O w/ programmable pull-up and interrupt-on-change
	AN0	AN		A/D Channel 0 input
	CIN+	AN		Comparator input
	ICSPDAT	TTL	CMOS	Serial programming I/O
GP1/AN1/CIN-/VREF/ ICSPCLK	GP1	TTL	CMOS	Bi-directional I/O w/ programmable pull-up and interrupt-on-change
	AN1	AN		A/D Channel 1 input
	CIN-	AN		Comparator input
	VREF	AN		External voltage reference
	ICSPCLK	ST		Serial programming clock
GP2/AN2/T0CKI/INT/COUT	GP2	ST	CMOS	Bi-directional I/O w/ programmable pull-up and interrupt-on-change
	AN2	AN		A/D Channel 2 input
	T0CKI	ST		TMR0 clock input
	INT	ST		External interrupt
	COUT		CMOS	Comparator output
GP3/$\overline{\text{MCLR}}$/VPP	GP3	TTL		Input port w/ interrupt-on-change
	$\overline{\text{MCLR}}$	ST		Master Clear
	VPP	HV		Programming voltage
GP4/AN3/$\overline{\text{T1G}}$/OSC2/ CLKOUT	GP4	TTL	CMOS	Bi-directional I/O w/ programmable pull-up and interrupt-on-change
	AN3	AN		A/D Channel 3 input
	$\overline{\text{T1G}}$	ST		TMR1 gate
	OSC2		XTAL	Crystal/resonator
	CLKOUT		CMOS	Fosc/4 output
GP5/T1CKI/OSC1/CLKIN	GP5	TTL	CMOS	Bi-directional I/O w/ programmable pull-up and interrupt-on-change
	T1CKI	ST		TMR1 clock
	OSC1	XTAL		Crystal/resonator
	CLKIN	ST		External clock input/RC oscillator connection
VSS	VSS	Power		Ground reference
VDD	VDD	Power		Positive supply

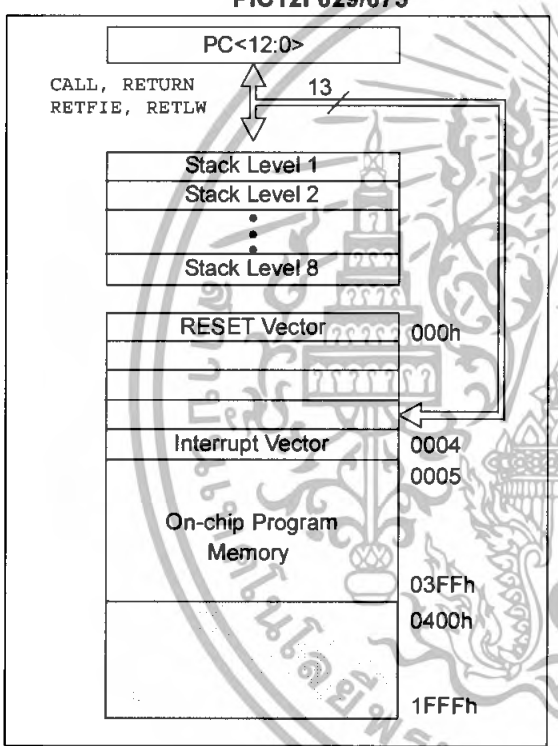
Legend: Shade = PIC12F675 only
TTL = TTL input buffer, ST = Schmitt Trigger input buffer

2.0 MEMORY ORGANIZATION

2.1 Program Memory Organization

The PIC12F629/675 devices have a 13-bit program counter capable of addressing an 8K x 14 program memory space. Only the first 1K x 14 (0000h - 03FFh) for the PIC12F629/675 devices is physically implemented. Accessing a location above these boundaries will cause a wrap around within the first 1K x 14 space. The RESET vector is at 0000h and the interrupt vector is at 0004h (see Figure 2-1).

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK FOR THE PIC12F629/675



2.2 Data Memory Organization

The data memory (see Figure 2-2) is partitioned into two banks, which contain the General Purpose registers and the Special Function registers. The Special Function registers are located in the first 32 locations of each bank. Register locations 20h-5Fh are General Purpose registers, implemented as static RAM and are mapped across both banks. All other RAM is unimplemented and returns '0' when read. RP0 (STATUS<5>) is the bank select bit.

- RP0 = 0 Bank 0 is selected
- RP0 = 1 Bank 1 is selected

Note: The IRP and RP1 bits STATUS<7:6> are reserved and should always be maintained as '0's.

2.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 64 x 8 in the PIC12F629/675 devices. Each register is accessed, either directly or indirectly, through the File Select Register FSR (see Section 2.4).

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function registers are registers used by the CPU and peripheral functions for controlling the desired operation of the device (see Table 2-1). These registers are static RAM.

The special registers can be classified into two sets: core and peripheral. The Special Function registers associated with the "core" are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

FIGURE 2-2: DATA MEMORY MAP OF THE PIC12F629/675

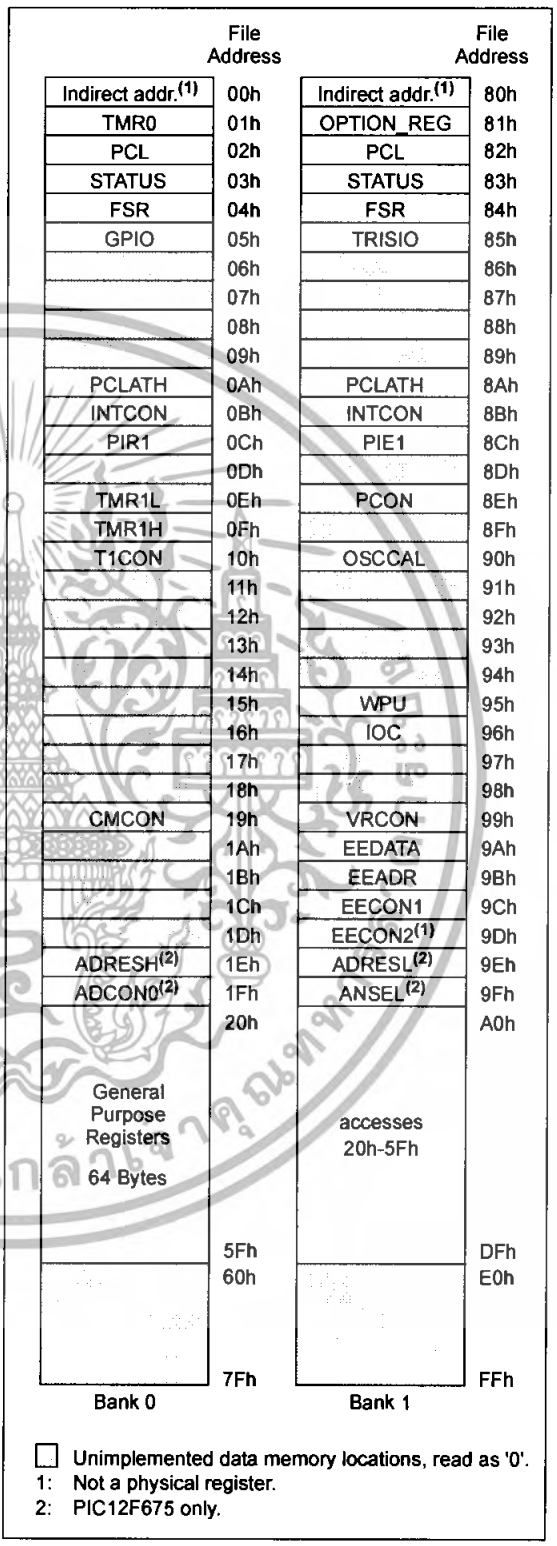


TABLE 2-1: SPECIAL FUNCTION REGISTERS SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOD	Page
Bank 0											
00h	INDF(1)	Addressing this Location uses Contents of FSR to Address Data Memory								0000 0000	18,59
01h	TMR0	Timer0 Module's Register								xxxx xxxx	27
02h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	17
03h	STATUS	IRP(2)	RP1(2)	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	11
04h	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	18
05h	GPIO	—	—	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	--xx xxxx	19
06h	—	Unimplemented								—	—
07h	—	Unimplemented								—	—
08h	—	Unimplemented								—	—
09h	—	Unimplemented								—	—
0Ah	PCLATH	—	—	—	Write Buffer for Upper 5 bits of Program Counter					---0 0000	17
0Bh	INTCON	GIE	PEIE	T0IE	INTE	GPIE	T0IF	INTF	GPIF	0000 0000	13
0Ch	PIR1	EEIF	ADIF	—	—	CMIF	—	—	TMR1IF	00-- 0--0	15
0Dh	—	Unimplemented								—	—
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit Timer1								xxxx xxxx	30
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit Timer1								xxxx xxxx	30
10h	T1CON	—	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	-000 0000	32
11h	—	Unimplemented								—	—
12h	—	Unimplemented								—	—
13h	—	Unimplemented								—	—
14h	—	Unimplemented								—	—
15h	—	Unimplemented								—	—
16h	—	Unimplemented								—	—
17h	—	Unimplemented								—	—
18h	—	Unimplemented								—	—
19h	CMCON	—	COUT	—	CINV	CIS	CM2	CM1	CM0	-0-0 0000	35
1Ah	—	Unimplemented								—	—
1Bh	—	Unimplemented								—	—
1Ch	—	Unimplemented								—	—
1Dh	—	Unimplemented								—	—
1Eh	ADRESH(3)	Most Significant 8 bits of the Left Shifted A/D Result or 2 bits of the Right Shifted Result								xxxx xxxx	42
1Fh	ADCON0(3)	ADFM	VCFG	—	—	CHS1	CHS0	GO/$\overline{DONE}$	ADON	00-- 0000	43,59

Legend: — = unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: This is not a physical register.
2: These bits are reserved and should always be maintained as '0'.
3: PIC12F675 only.

TABLE 2-1: SPECIAL FUNCTION REGISTERS SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOD	Page
Bank 1											
80h	INDF(1)	Addressing this Location uses Contents of FSR to Address Data Memory								0000 0000	18,59
81h	OPTION_REG	$\overline{GPPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	12,28
82h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	17
83h	STATUS	IRP(2)	RP1(2)	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	11
84h	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	18
85h	TRISIO	—	—	TRISIO5	TRISIO4	TRISIO3	TRISIO2	TRISIO1	TRISIO0	--11 1111	19
86h	—	Unimplemented								—	—
87h	—	Unimplemented								—	—
88h	—	Unimplemented								—	—
89h	—	Unimplemented								—	—
8Ah	PCLATH	—	—	—	Write Buffer for Upper 5 bits of Program Counter					---0 0000	17
8Bh	INTCON	GIE	PEIE	T0IE	INTE	GPIE	T0IF	INTF	GPIF	0000 0000	13
8Ch	PIE1	EEIE	ADIE	—	—	CMIE	—	—	TMR1IE	00-- 0--0	14
8Dh	—	Unimplemented								—	—
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOD}$	---- --0x	16
8Fh	—	Unimplemented								—	—
90h	OSCCAL	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	—	—	1000 00--	16
91h	—	Unimplemented								—	—
92h	—	Unimplemented								—	—
93h	—	Unimplemented								—	—
94h	—	Unimplemented								—	—
95h	WPU	—	—	WPU5	WPU4	—	WPU2	WPU1	WPU0	--11 -111	20
96h	IOC	—	—	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0	--00 0000	21
97h	—	Unimplemented								—	—
98h	—	Unimplemented								—	—
99h	VRCON	VREN	—	VRR	—	VR3	VR2	VR1	VR0	0-0- 0000	40
9Ah	EEDATA	Data EEPROM Data Register								0000 0000	47
9Bh	EEADR	—	Data EEPROM Address Register							-000 0000	47
9Ch	EECON1	—	—	—	—	WRERR	WREN	WR	RD	---- x000	48
9Dh	EECON2(1)	EEPROM Control Register 2								---- ----	48
9Eh	ADRESL(3)	Least Significant 2 bits of the Left Shifted A/D Result of 8 bits or the Right Shifted Result								xxxx xxxx	42
9Fh	ANSEL(3)	—	ADCS2	ADCS1	ADCS0	ANS3	ANS2	ANS1	ANS0	-000 1111	44,59

Legend: — = unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: This is not a physical register.
2: These bits are reserved and should always be maintained as '0'.
3: PIC12F675 only.

2.2.2.1 STATUS Register

The STATUS register, shown in Register 2-1, contains:

- the arithmetic status of the ALU
- the RESET status
- the bank select bits for data memory (SRAM)

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect any STATUS bits. For other instructions not affecting any STATUS bits, see the "Instruction Set Summary".

Note 1: Bits IRP and RP1 (STATUS<7:6>) are not used by the PIC12F629/675 and should be maintained as clear. Use of these bits is not recommended, since this may affect upward compatibility with future products.

2: The C and DC bits operate as a $\overline{Borrow}$ and $\overline{Digit\ Borrow}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 2-1: STATUS — STATUS REGISTER (ADDRESS: 03h OR 83h)

Reserved	Reserved	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** This bit is reserved and should be maintained as '0'

bit 6 **RP1:** This bit is reserved and should be maintained as '0'

bit 5 **RP0:** Register Bank Select bit (used for direct addressing)
0 = Bank 0 (00h - 7Fh)
1 = Bank 1 (80h - FFh)

bit 4 $\overline{TO}$: Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3 $\overline{PD}$: Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: $\overline{Digit\ carry}$/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
For $\overline{borrow}$, the polarity is reversed.
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.2 OPTION Register

The OPTION register is a readable and writable register, which contains various control bits to configure:

- TMR0/WDT prescaler
- External GP2/INT interrupt
- TMR0
- Weak pull-ups on GPIO

Note: To achieve a 1:1 prescaler assignment for TMR0, assign the prescaler to the WDT by setting PSA bit to '1' (OPTION<3>). See Section 4.4.

REGISTER 2-2: OPTION_REG — OPTION REGISTER (ADDRESS: 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{GPPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{GPPU}}$:** GPIO Pull-up Enable bit
1 = GPIO pull-ups are disabled
0 = GPIO pull-ups are enabled by individual port latch values

bit 6 **INTEDG:** Interrupt Edge Select bit
1 = Interrupt on rising edge of GP2/INT pin
0 = Interrupt on falling edge of GP2/INT pin

bit 5 **T0CS:** TMR0 Clock Source Select bit
1 = Transition on GP2/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE:** TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on GP2/T0CKI pin
0 = Increment on low-to-high transition on GP2/T0CKI pin

bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the TIMER0 module

bit 2-0 **PS2:PS0:** Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains the various enable and flag bits for TMR0 register overflow, GPIO port change and external GP2/INT pin interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-3: INTCON — INTERRUPT CONTROL REGISTER (ADDRESS: 0Bh OR 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE	PEIE	T0IE	INTE	GPIE	T0IF	INTF	GPIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE:** GP2/INT External Interrupt Enable bit
1 = Enables the GP2/INT external interrupt
0 = Disables the GP2/INT external interrupt

bit 3 **GPIE:** Port Change Interrupt Enable bit(1)
1 = Enables the GPIO port change interrupt
0 = Disables the GPIO port change interrupt

bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit(2)
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** GP2/INT External Interrupt Flag bit
1 = The GP2/INT external interrupt occurred (must be cleared in software)
0 = The GP2/INT external interrupt did not occur

bit 0 **GPIF:** Port Change Interrupt Flag bit
1 = When at least one of the GP5:GP0 pins changed state (must be cleared in software)
0 = None of the GP5:GP0 pins have changed state

Note 1: IOC register must also be enabled to enable an interrupt-on-change.

2: T0IF bit is set when TIMER0 rolls over. TIMER0 is unchanged on RESET and should be initialized before clearing T0IF bit.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.4 PIE1 Register

The PIE1 register contains the interrupt enable bits, as shown in Register 2-4.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1 — PERIPHERAL INTERRUPT ENABLE REGISTER 1 (ADDRESS: 8Ch)

R/W-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0	R/W-0
EEIE	ADIE	—	—	CMIE	—	—	TMR1IE
bit 7							bit 0

bit 7 **EEIE:** EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 6 **ADIE:** A/D Converter Interrupt Enable bit (PIC12F675 only)
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5-4 **Unimplemented:** Read as '0'

bit 3 **CMIE:** Comparator Interrupt Enable bit
1 = Enables the comparator interrupt
0 = Disables the comparator interrupt

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.5 PIR1 Register

The PIR1 register contains the interrupt flag bits, as shown in Register 2-5.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1 — PERIPHERAL INTERRUPT REGISTER 1 (ADDRESS: 0Ch)

R/W-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0	R/W-0
EEIF	ADIF	—	—	CMIF	—	—	TMR1IF
bit 7							bit 0

bit 7 **EEIF:** EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation has not completed or has not been started

bit 6 **ADIF:** A/D Converter Interrupt Flag bit (PIC12F675 only)
1 = The A/D conversion is complete (must be cleared in software)
0 = The A/D conversion is not complete

bit 5-4 **Unimplemented**: Read as '0'

bit 3 **CMIF**: Comparator Interrupt Flag bit
1 = Comparator input has changed (must be cleared in software)
0 = Comparator input has not changed

bit 2-1 **Unimplemented**: Read as '0'

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.6 PCON Register

The Power Control (PCON) register contains flag bits to differentiate between a:

- Power-on Reset (POR)
- Brown-out Detect (BOD)
- Watchdog Timer Reset (WDT)
- External $\overline{MCLR}$ Reset

The PCON Register bits are shown in Register 2-6.

REGISTER 2-6: PCON — POWER CONTROL REGISTER (ADDRESS: 8Eh)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-x
—	—	—	—	—	—	$\overline{POR}$	$\overline{BOD}$
bit 7							bit 0

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **$\overline{POR}$:** Power-on Reset STATUS bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOD}$:** Brown-out Detect STATUS bit
1 = No Brown-out Detect occurred
0 = A Brown-out Detect occurred (must be set in software after a Brown-out Detect occurs)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.7 OSCCAL Register

The Oscillator Calibration register (OSCCAL) is used to calibrate the internal 4 MHz oscillator. It contains 6 bits to adjust the frequency up or down to achieve 4 MHz.

The OSCCAL register bits are shown in Register 2-7.

REGISTER 2-7: OSCCAL — OSCILLATOR CALIBRATION REGISTER (ADDRESS: 90h)

R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	—	—
bit 7							bit 0

bit 7-2 **CAL5:CAL0:** 6-bit Signed Oscillator Calibration bits
111111 = Maximum frequency
100000 = Center frequency
000000 = Minimum frequency

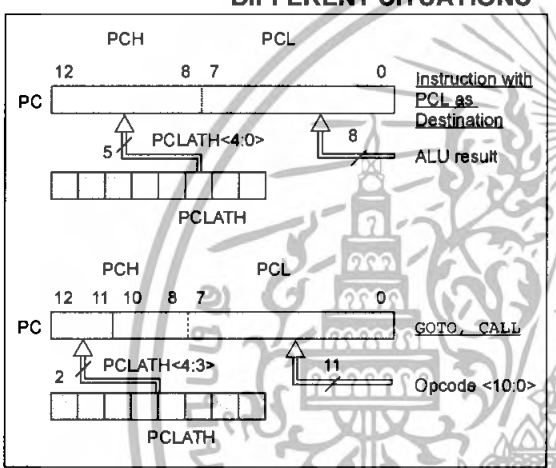
bit 1-0 **Unimplemented:** Read as '0'

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.3 PCL and PCLATH

The program counter (PC) is 13-bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<12:8>) is not directly readable or writable and comes from PCLATH. On any RESET, the PC is cleared. Figure 2-3 shows the two situations for the loading of the PC. The upper example in Figure 2-3 shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in Figure 2-3 shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

FIGURE 2-3: LOADING OF PC IN DIFFERENT SITUATIONS



2.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When performing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to the Application Note *"Implementing a Table Read"* (AN556).

2.3.2 STACK

The PIC12F629/675 family has an 8-level deep x 13-bit wide hardware stack (see Figure 2-1). The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed, or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

Note 1: There are no STATUS bits to indicate stack overflow or stack underflow conditions.

2: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW and RETFIE instructions, or the vectoring to an interrupt address.

2.4 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

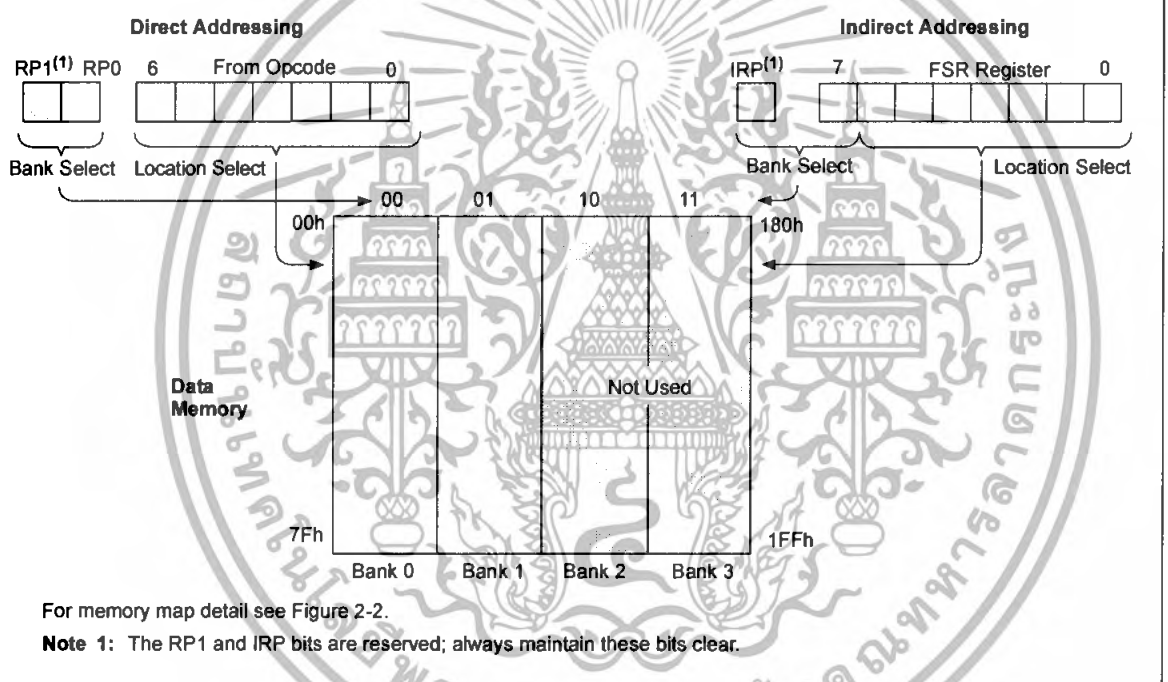
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses data pointed to by the File Select register (FSR). Reading INDF itself indirectly will produce 00h. Writing to the INDF register indirectly results in a no operation (although STATUS bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-4.

A simple program to clear RAM location 20h-2Fh using indirect addressing is shown in Example 2-1.

EXAMPLE 2-1: INDIRECT ADDRESSING

```
        movlw   0x20    ;initialize pointer
        movwf   FSR     ;to RAM
NEXT    clrf    INDF    ;clear INDF register
        incf    FSR     ;inc pointer
        btfss   FSR,4   ;all done?
        goto    NEXT    ;no clear next
CONTINUE                ;yes continue
```

FIGURE 2-4: DIRECT/INDIRECT ADDRESSING PIC12F629/675



For memory map detail see Figure 2-2.

Note 1: The RP1 and IRP bits are reserved; always maintain these bits clear.

3.0 GPIO PORT

There are as many as six general purpose I/O pins available. Depending on which peripherals are enabled, some or all of the pins may not be available as general purpose I/O. In general, when a peripheral is enabled, the associated pin may not be used as a general purpose I/O pin.

Note:	Additional information on I/O ports may be found in the PIC® Mid-Range Reference Manual, (DS33023).

3.1 GPIO and the TRISIO Registers

GPIO is an 6-bit wide, bi-directional port. The corresponding data direction register is TRISIO. Setting a TRISIO bit (= 1) will make the corresponding GPIO pin an input (i.e., put the corresponding output driver in a Hi-impedance mode). Clearing a TRISIO bit (= 0) will make the corresponding GPIO pin an output (i.e., put the contents of the output latch on the selected pin). The exception is GP3, which is input only and its TRISIO bit will always read as '1'. Example 3-1 shows how to initialize GPIO.

Reading the GPIO register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified, and then written to the port data latch. GP3 reads '0' when MCLREN = 1.

The TRISIO register controls the direction of the GP pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISIO register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

Note:	The ANSEL (9Fh) and CMCON (19h) registers (9Fh) must be initialized to configure an analog channel as a digital input. Pins configured as analog inputs will read '0'. The ANSEL register is defined for the PIC12F675.

EXAMPLE 3-1: INITIALIZING GPIO

```
BCF     STATUS,RP0      ;Bank 0
CLRF    GPIO            ;Init GPIO
MOVLW   07h             ;Set GP<2:0> to
MOVWF   CMCON           ;digital IO
BSF     STATUS,RP0      ;Bank 1
CLRF    ANSEL           ;Digital I/O
MOVLW   0Ch             ;Set GP<3:2> as inputs
MOVWF   TRISIO          ;and set GP<5:4,1:0>
                        ;as outputs
```

3.2 Additional Pin Functions

Every GPIO pin on the PIC12F629/675 has an interrupt-on-change option and every GPIO pin, except GP3, has a weak pull-up option. The next two sections describe these functions.

3.2.1 WEAK PULL-UP

Each of the GPIO pins, except GP3, has an individually configurable weak internal pull-up. Control bits WPUx enable or disable each pull-up. Refer to Register 3-3. Each weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset by the $\overline{\text{GPPU}}$ bit (OPTION<7>).

REGISTER 3-1: GPIO — GPIO REGISTER (ADDRESS: 05h)

U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
bit 7							bit 0

bit 7-6: **Unimplemented**: Read as '0'

bit 5-0: **GPIO<5:0>**: General Purpose I/O pin.
1 = Port pin is >VIH
0 = Port pin is <VIL

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 3-2: TRISIO — GPIO TRI-STATE REGISTER (ADDRESS: 85h)

U-0	U-0	R/W-x	R/W-x	R-1	R/W-x	R/W-x	R/W-x
—	—	TRISIO5	TRISIO4	TRISIO3	TRISIO2	TRISIO1	TRISIO0
bit 7							bit 0

bit 7-6: **Unimplemented**: Read as '0'

bit 5-0: **TRISIO<5:0>**: General Purpose I/O Tri-State Control bit
1 = GPIO pin configured as an input (tri-stated)
0 = GPIO pin configured as an output.

Note: TRISIO<3> always reads 1.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 3-3: WPU — WEAK PULL-UP REGISTER (ADDRESS: 95h)

U-0	U-0	R/W-1	R/W-1	U-0	R/W-1	R/W-1	R/W-1
—	—	WPU5	WPU4	—	WPU2	WPU1	WPU0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **WPU<5:4>:** Weak Pull-up Register bit
1 = Pull-up enabled
0 = Pull-up disabled

bit 3 **Unimplemented:** Read as '0'

bit 2-0 **WPU<2:0>:** Weak Pull-up Register bit
1 = Pull-up enabled
0 = Pull-up disabled

Note 1: Global $\overline{\text{GPPU}}$ must be enabled for individual pull-ups to be enabled.
2: The weak pull-up device is automatically disabled if the pin is in Output mode (TRISIO = 0).

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

3.2.2 INTERRUPT-ON-CHANGE

Each of the GPIO pins is individually configurable as an interrupt-on-change pin. Control bits IOC enable or disable the interrupt function for each pin. Refer to Register 3-4. The interrupt-on-change is disabled on a Power-on Reset.

For enabled interrupt-on-change pins, the values are compared with the old value latched on the last read of GPIO. The 'mismatch' outputs of the last read are OR'd together to set, the GP Port Change Interrupt flag bit (GPIF) in the INTCON register.

This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of GPIO. This will end the mismatch condition.

b) Clear the flag bit GPIF.

A mismatch condition will continue to set flag bit GPIF. Reading GPIO will end the mismatch condition and allow flag bit GPIF to be cleared.

Note:	If a change on the I/O pin should occur when the read operation is being executed (start of the Q2 cycle), then the GPIF interrupt flag may not get set.

REGISTER 3-4: IOC — INTERRUPT-ON-CHANGE GPIO REGISTER (ADDRESS: 96h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **IOC<5:0>:** Interrupt-on-Change GPIO Control bit
1 = Interrupt-on-change enabled
0 = Interrupt-on-change disabled

Note 1: Global interrupt enable (GIE) must be enabled for individual interrupts to be recognized.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

3.3 Pin Descriptions and Diagrams

Each GPIO pin is multiplexed with other functions. The pins and their combined functions are briefly described here. For specific information about individual functions such as the comparator or the A/D, refer to the appropriate section in this Data Sheet.

3.3.1 GP0/AN0/CIN+

Figure 3-1 shows the diagram for this pin. The GP0 pin is configurable to function as one of the following:

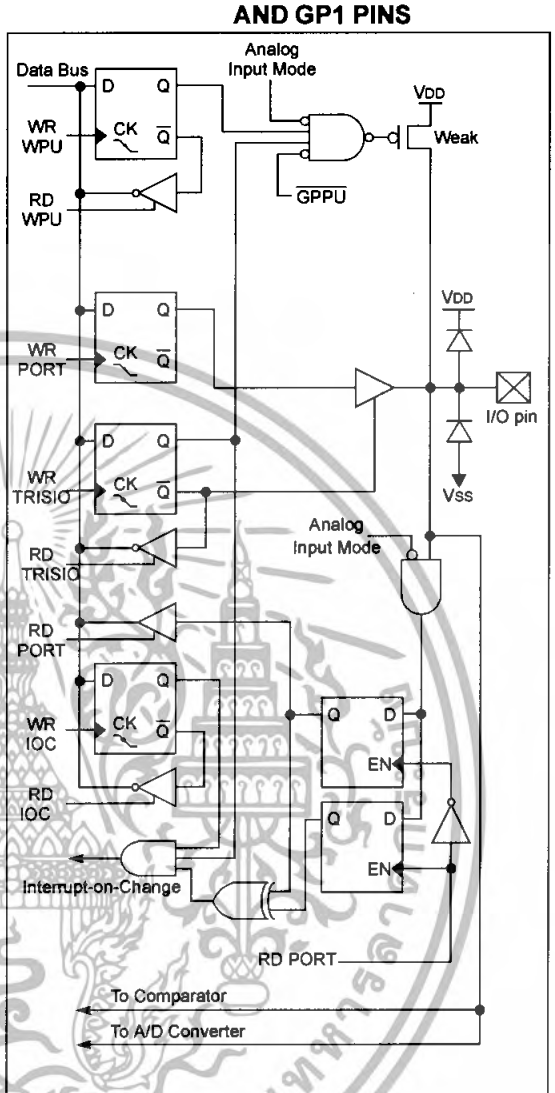
- a general purpose I/O
- an analog input for the A/D (PIC12F675 only)
- an analog input to the comparator

3.3.2 GP1/AN1/CIN-/VREF

Figure 3-1 shows the diagram for this pin. The GP1 pin is configurable to function as one of the following:

- as a general purpose I/O
- an analog input for the A/D (PIC12F675 only)
- an analog input to the comparator
- a voltage reference input for the A/D (PIC12F675 only)

FIGURE 3-1: BLOCK DIAGRAM OF GP0 AND GP1 PINS

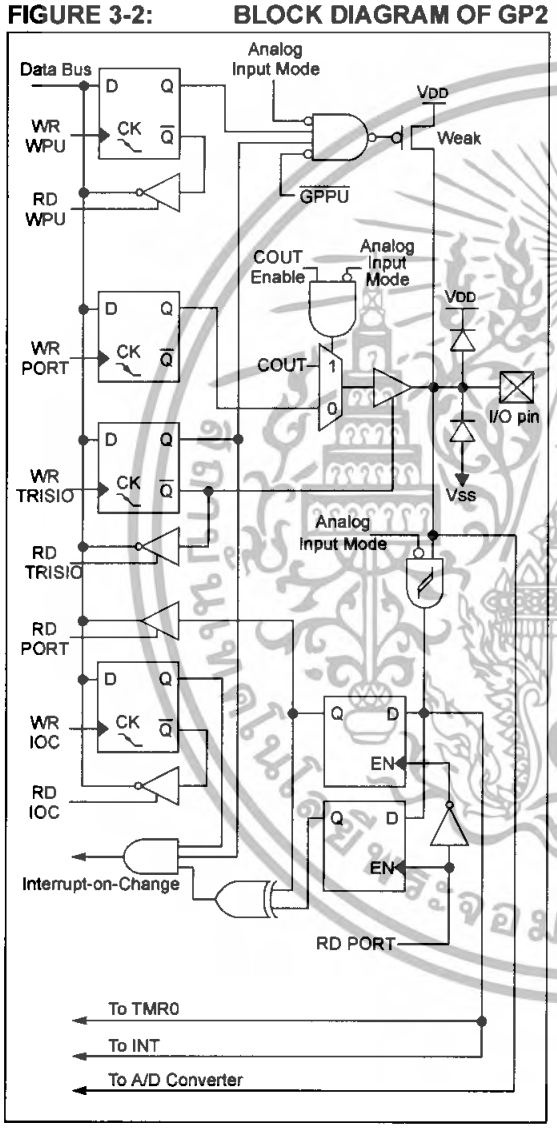


3.3.3 GP2/AN2/T0CKI/INT/COUT

Figure 3-2 shows the diagram for this pin. The GP2 pin is configurable to function as one of the following:

- a general purpose I/O
- an analog input for the A/D (PIC12F675 only)
- the clock input for TMR0
- an external edge triggered interrupt
- a digital output from the comparator

FIGURE 3-2: BLOCK DIAGRAM OF GP2

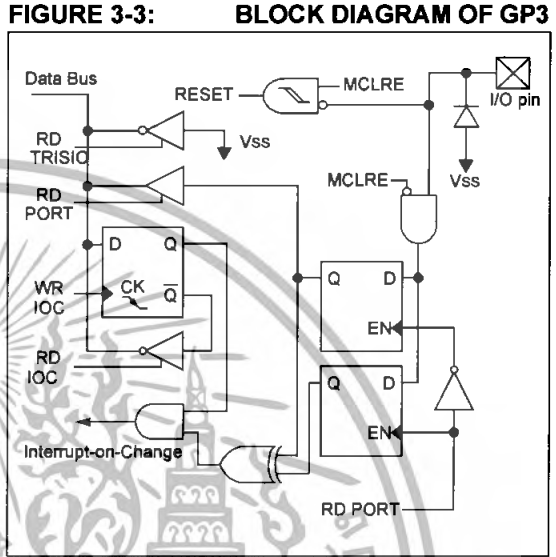


3.3.4 GP3/$\overline{MCLR}$/VPP

Figure 3-3 shows the diagram for this pin. The GP3 pin is configurable to function as one of the following:

- a general purpose input
- as Master Clear Reset

FIGURE 3-3: BLOCK DIAGRAM OF GP3

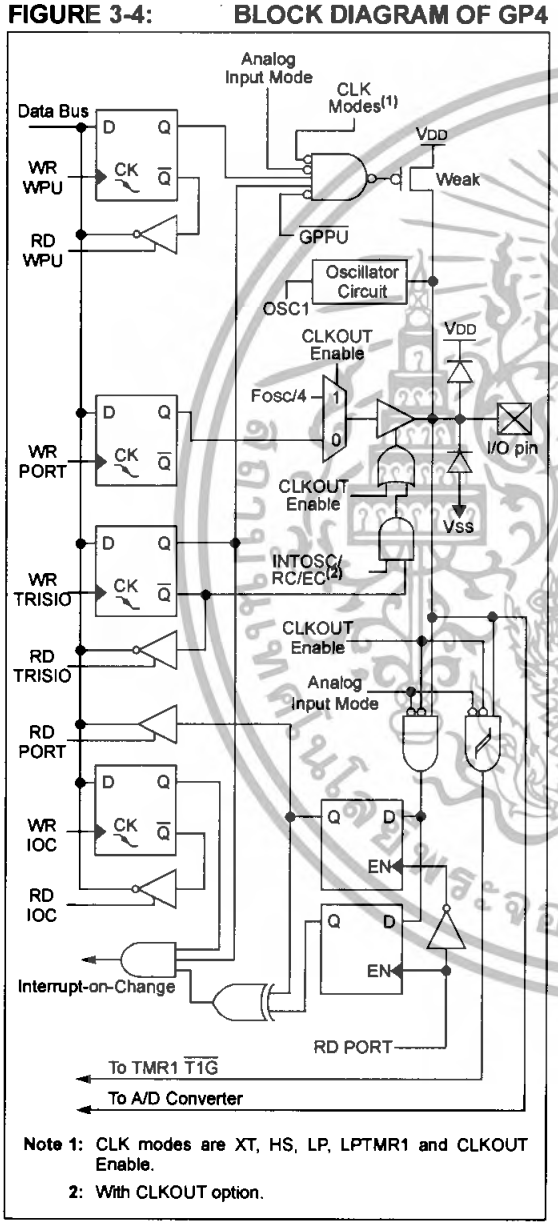


3.3.5 GP4/AN3/T1G/OSC2/CLKOUT

Figure 3-4 shows the diagram for this pin. The GP4 pin is configurable to function as one of the following:

- a general purpose I/O
- an analog input for the A/D (PIC12F675 only)
- a TMR1 gate input
- a crystal/resonator connection
- a clock output

FIGURE 3-4: BLOCK DIAGRAM OF GP4



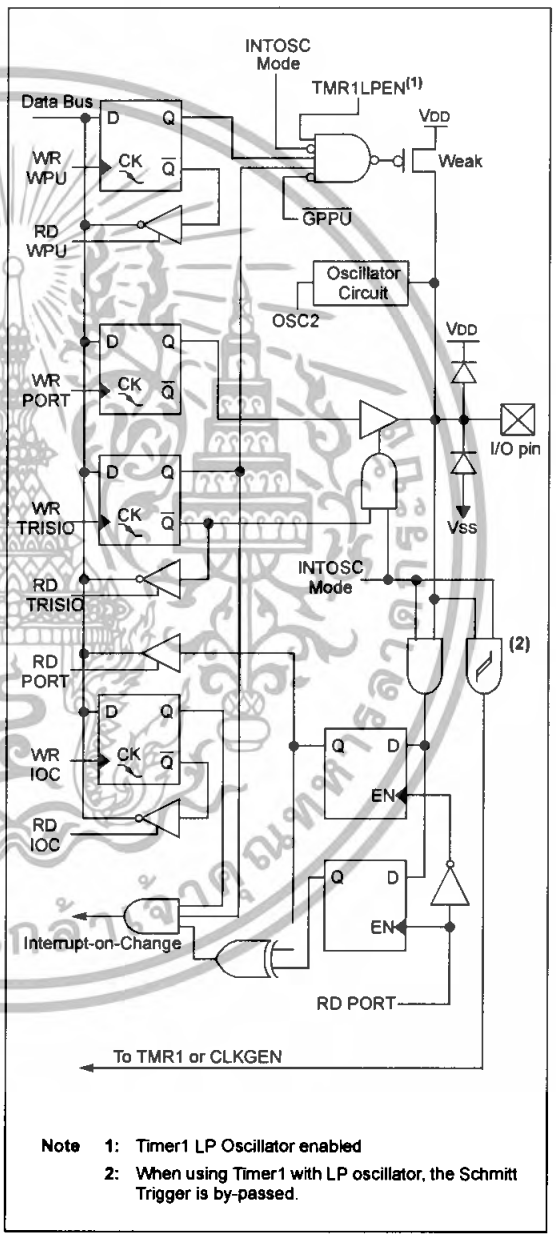
Note 1: CLK modes are XT, HS, LP, LPTMR1 and CLKOUT Enable.
2: With CLKOUT option.

3.3.6 GP5/T1CKI/OSC1/CLKIN

Figure 3-5 shows the diagram for this pin. The GP5 pin is configurable to function as one of the following:

- a general purpose I/O
- a TMR1 clock input
- a crystal/resonator connection
- a clock input

FIGURE 3-5: BLOCK DIAGRAM OF GP5



Note 1: Timer1 LP Oscillator enabled
2: When using Timer1 with LP oscillator, the Schmitt Trigger is by-passed.

TABLE 3-1: SUMMARY OF REGISTERS ASSOCIATED WITH GPIO

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOD	Value on all other RESETS
05h	GPIO	—	—	GP5	GP4	GP3	GP2	GP1	GP0	--xx xxxx	--uu uuuu
0Bh/8Bh	INTCON	GIE	PEIE	T0IE	INTE	GPIE	T0IF	INTF	GPIF	0000 0000	0000 000u
19h	CMCON	—	COUT	—	CINV	CIS	CM2	CM1	CM0	-0-0 0000	-0-0 0000
81h	OPTION_REG	$\overline{GPPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISIO	—	—	TRISIO5	TRISIO4	TRISIO3	TRISIO2	TRISIO1	TRISIO0	--11 1111	--11 1111
95h	WPU	—	—	WPU5	WPU4	—	WPU2	WPU1	WPU0	--11 -111	--11 -111
96h	IOC	—	—	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0	--00 0000	--00 0000
9Fh	ANSEL	—	ADCS2	ADCS1	ADCS0	ANS3	ANS2	ANS1	ANS0	-000 1111	-000 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by GPIO.

NOTES:

PIC16F87X
Data Sheet

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Trademarks

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV — DNV MSC The Netherlands Accredited by the RvA — ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

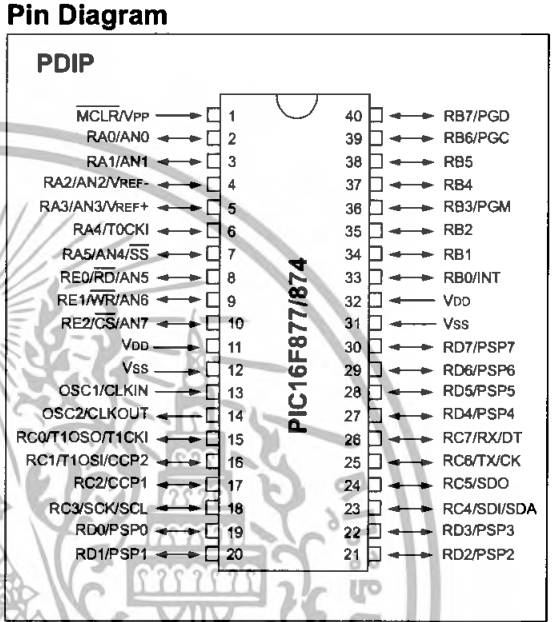
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

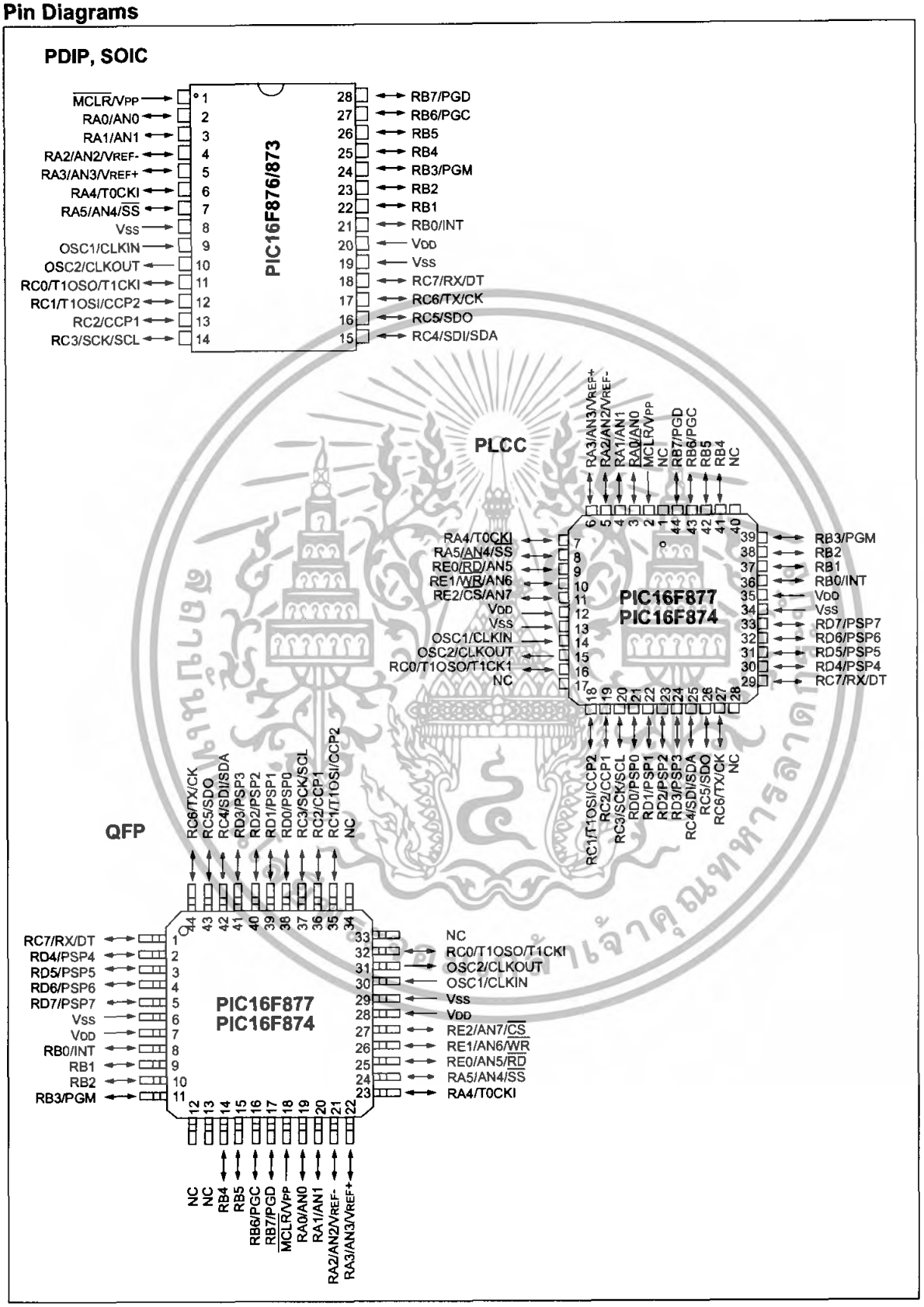
PDIP



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Pin Diagrams



Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

Table of Contents

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com/cn** to receive the most current information on all of our products.

1.0 DEVICE OVERVIEW

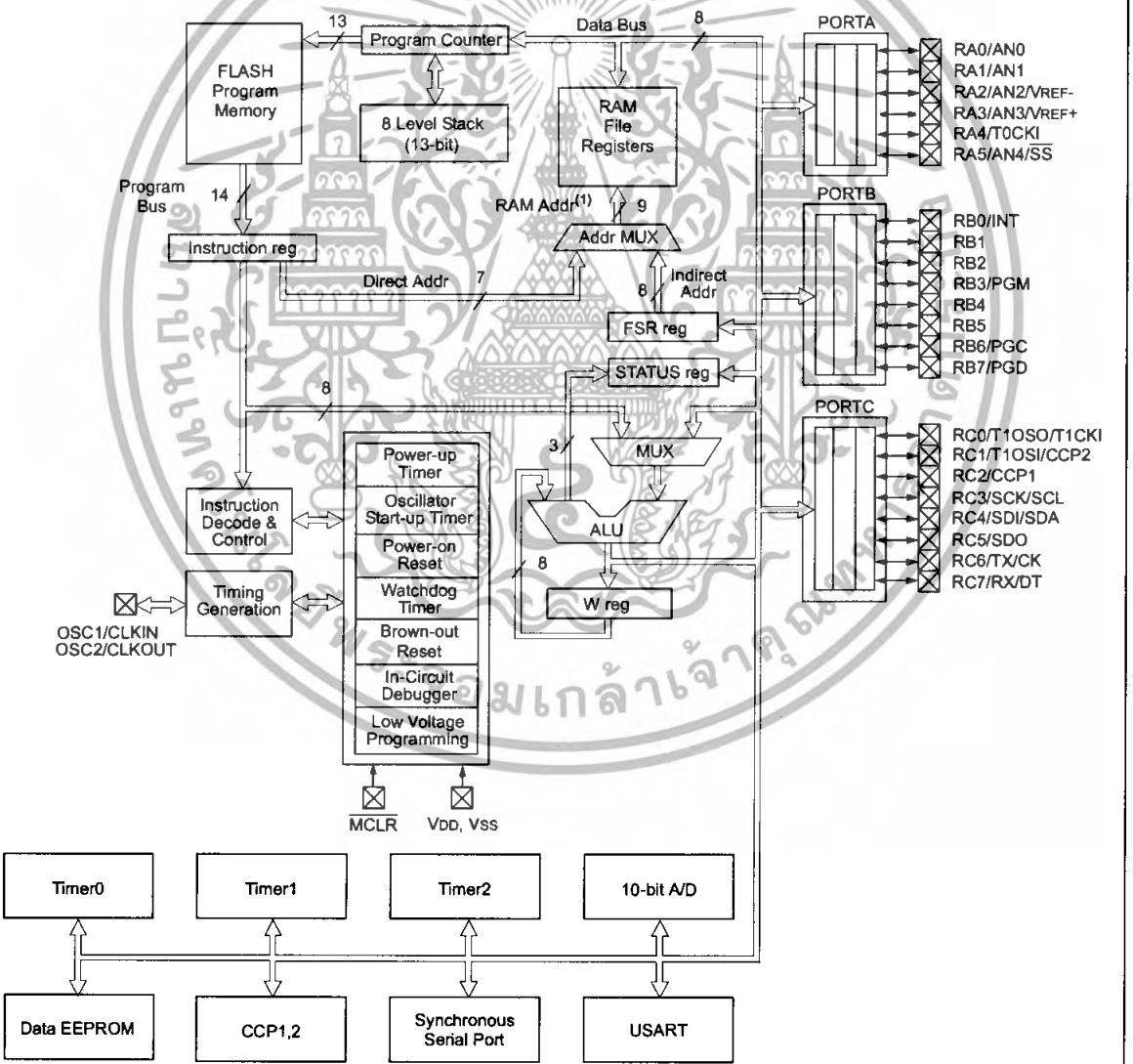
This document contains device specific information. Additional information may be found in the PICmicro™ Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

There are four devices (PIC16F873, PIC16F874, PIC16F876 and PIC16F877) covered by this data sheet. The PIC16F876/873 devices come in 28-pin packages and the PIC16F877/874 devices come in 40-pin packages. The Parallel Slave Port is not implemented on the 28-pin devices.

The following device block diagrams are sorted by pin number; 28-pin for Figure 1-1 and 40-pin for Figure 1-2. The 28-pin and 40-pin pinouts are listed in Table 1-1 and Table 1-2, respectively.

FIGURE 1-1: PIC16F873 AND PIC16F876 BLOCK DIAGRAM

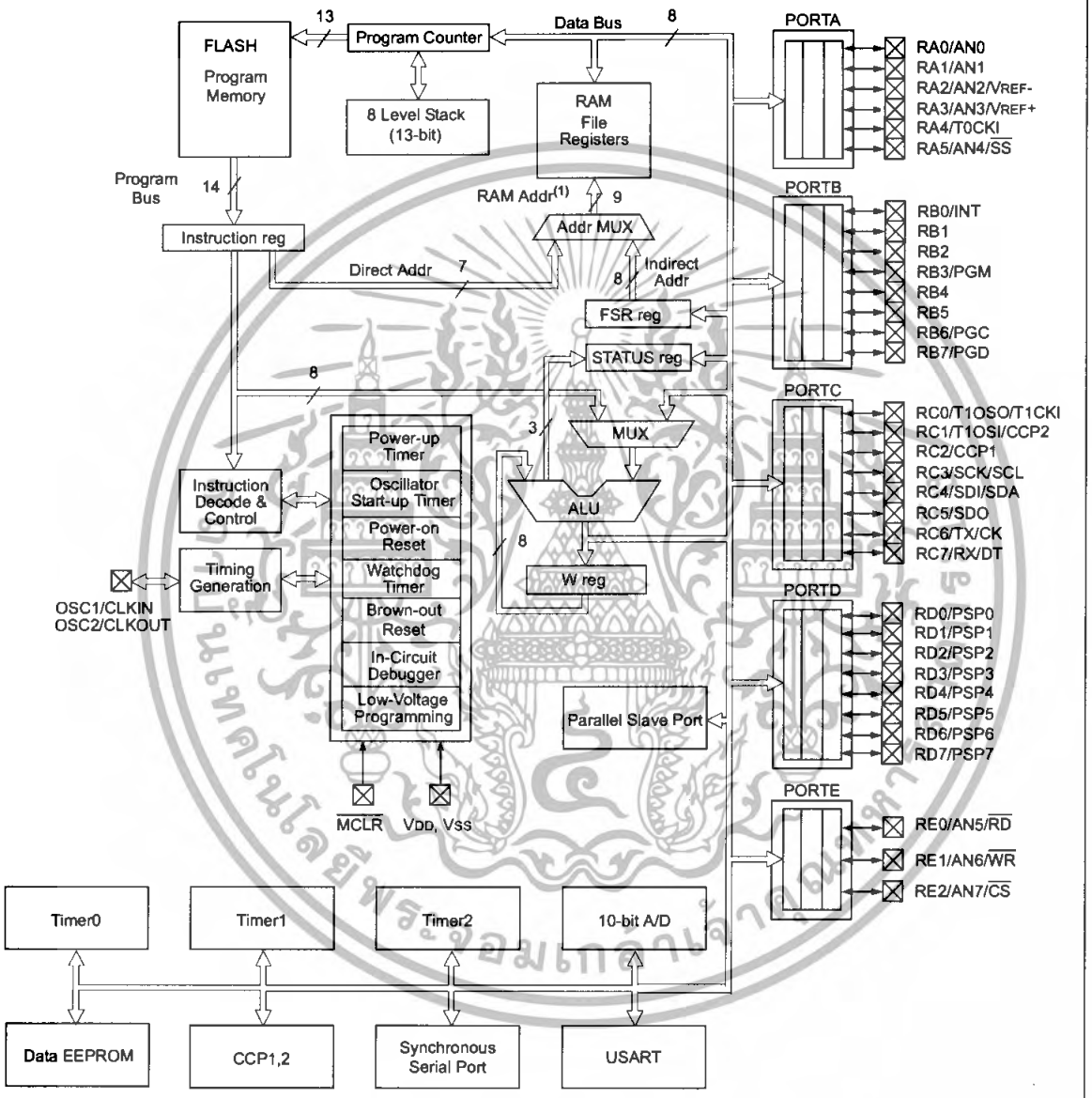
Device	Program FLASH	Data Memory	Data EEPROM
PIC16F873	4K	192 Bytes	128 Bytes
PIC16F876	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

TABLE 1-1: PIC16F873 AND PIC16F876 PINOUT DESCRIPTION

Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	6	I/O	ST	RA4 can also be the clock input to the Timer0 module. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	21	21	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	25	25	I/O	TTL	Interrupt-on-change pin.
RB5	26	26	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	27	27	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	28	28	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
					PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	11	I/O	ST	RC0 can also be the Timer1 oscillator output or Timer1 clock input.
RC1/T1OSI/CCP2	12	12	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	13	13	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	14	14	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	16	16	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	17	17	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	18	18	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
VSS	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	20	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

NOTES:

2.0 MEMORY ORGANIZATION

There are three memory blocks in each of the PIC16F87X MCUs. The Program Memory and Data Memory have separate buses so that concurrent access can occur and is detailed in this section. The EEPROM data memory block is detailed in Section 4.0.

Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

2.1 Program Memory Organization

The PIC16F87X devices have a 13-bit program counter capable of addressing an 8K x 14 program memory space. The PIC16F877/876 devices have 8K x 14 words of FLASH program memory, and the PIC16F873/874 devices have 4K x 14. Accessing a location above the physically implemented address will cause a wraparound.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PIC16F877/876 PROGRAM MEMORY MAP AND STACK

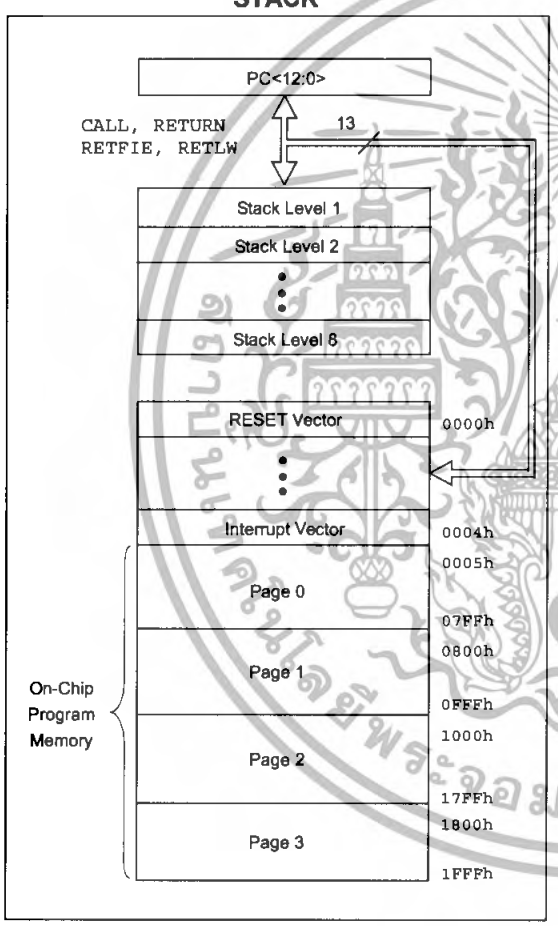
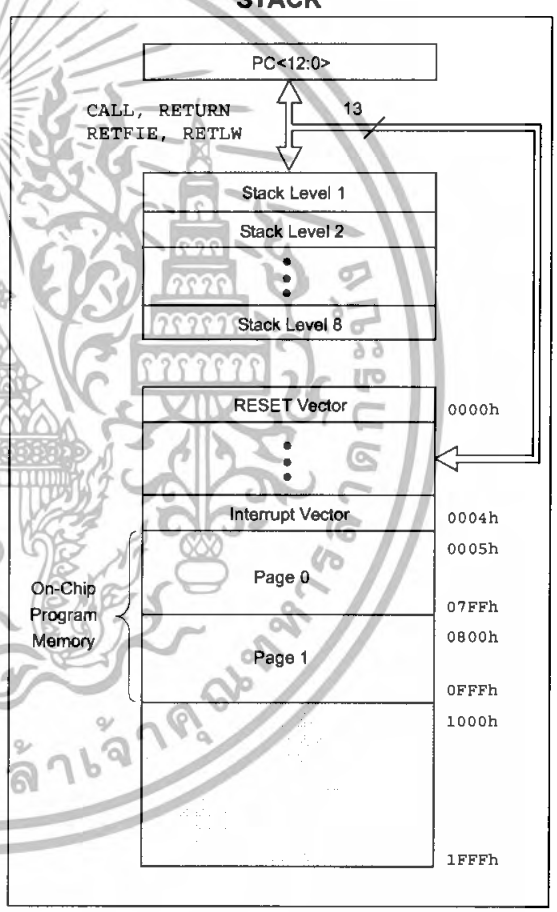


FIGURE 2-2: PIC16F874/873 PROGRAM MEMORY MAP AND STACK



2.2 Data Memory Organization

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 (STATUS<6>) and RP0 (STATUS<5>) are the bank select bits.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

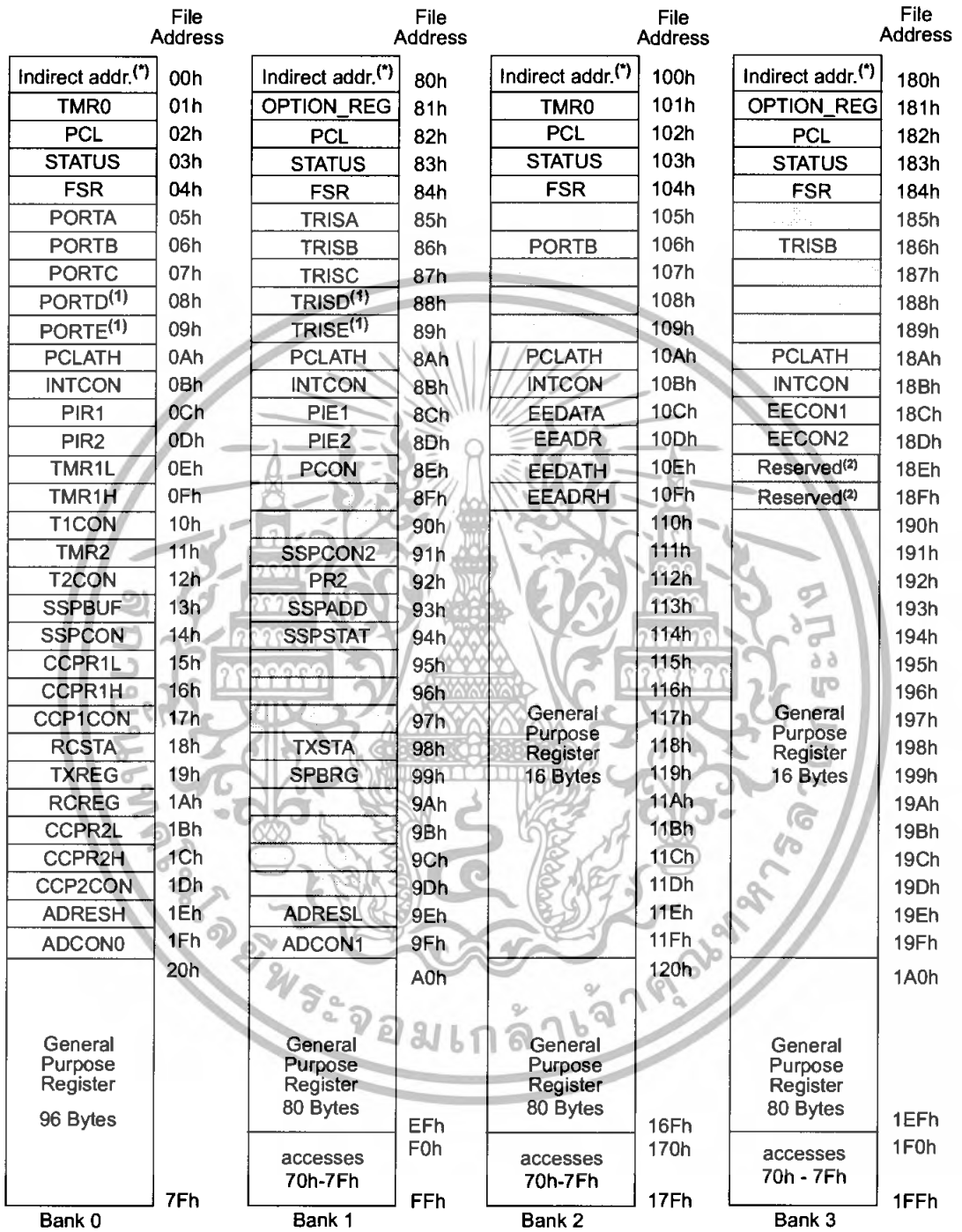
Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

Note: EEPROM Data Memory description can be found in Section 4.0 of this data sheet.

2.2.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly, or indirectly through the File Select Register (FSR).

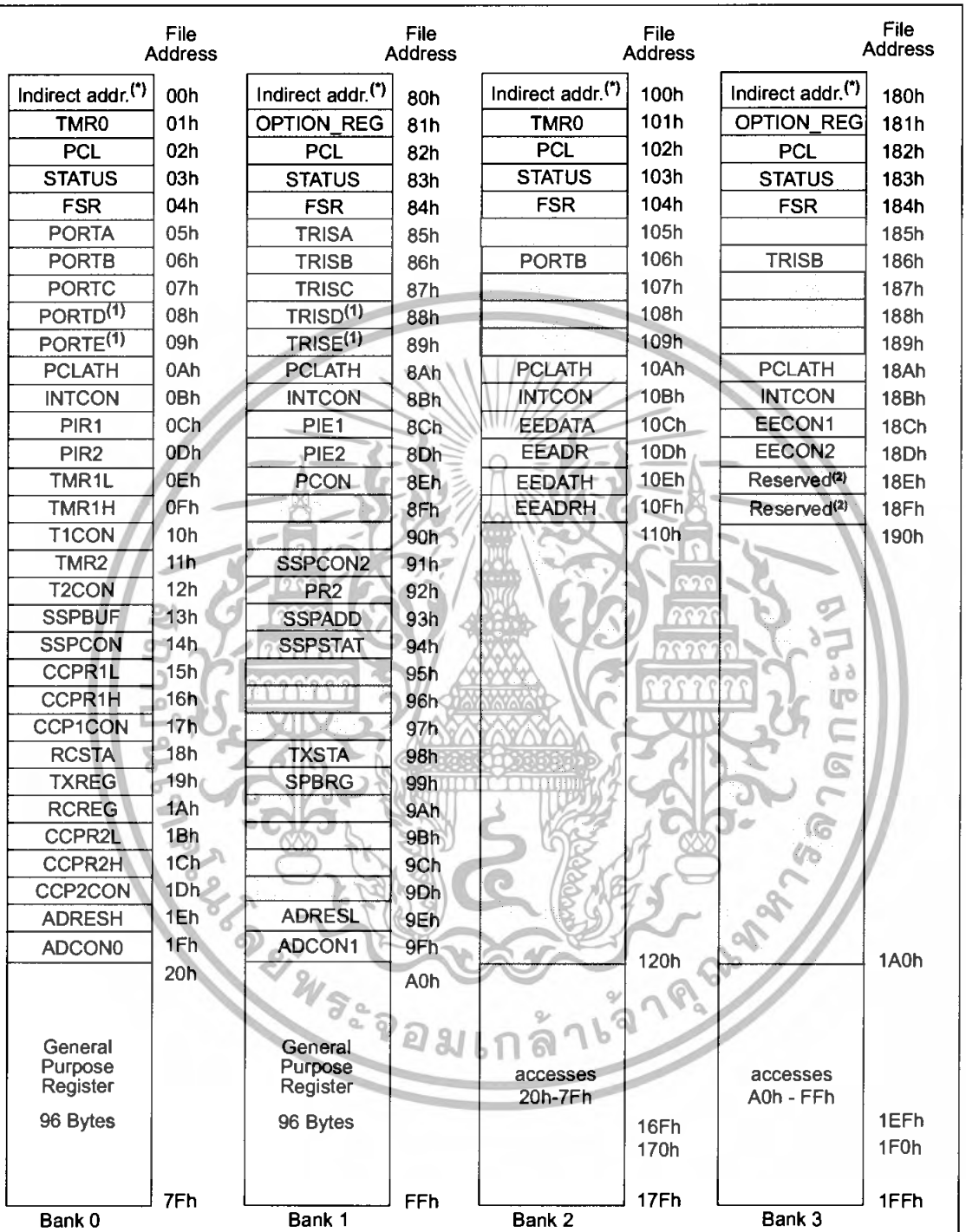
FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP



☐ Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.

FIGURE 2-4: PIC16F874/873 REGISTER FILE MAP



☐ Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873.
2: These registers are reserved, maintain these registers clear.

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
01h	TMR0	Timer0 Module Register								xxxx xxxx	47
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	29
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	33
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	35
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	36
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
0Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	22
0Dh	PIR2	—	(5)	—	EEIF	BCLIF	—	—	CCP2IF	-r-0 0--0	24
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	51
11h	TMR2	Timer2 Module Register								0000 0000	55
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	55
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	70, 73
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	67
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	57
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	57
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	58
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	96
19h	TXREG	USART Transmit Data Register								0000 0000	99
1Ah	RCREG	USART Receive Data Register								0000 0000	101
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	57
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	57
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	58
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	116
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	$\overline{GO/DONE}$	—	ADON	0000 00-0	111

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devicas; always maintain these bits clear.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	29
86h	TRISB	PORTB Data Direction Register								1111 1111	31
87h	TRISC	PORTC Data Direction Register								1111 1111	33
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	35
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	37
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
8Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	21
8Dh	PIE2	—	(5)	—	EEIE	BCLIE	—	—	CCP2IE	-r-0 0--0	23
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	25
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	68
92h	PR2	Timer2 Period Register								1111 1111	55
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	73, 74
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	66
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	95
99h	SPBRG	Baud Rate Generator Register								0000 0000	97
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	—	Unimplemented								—	—
9Dh	—	Unimplemented								—	—
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	116
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000	112

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
101h	TMR0	Timer0 Module Register								xxxx xxxx	47
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	26
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
10Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	41
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	41
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						xxxx xxxx	41
10Fh	EEADRH	—	—	—	EEPROM Address Register High Byte					xxxx xxxx	41
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	31
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
18Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	41, 42
18Dh	EECON2	EEPROM Control Register2 (not a physical register)								---- ----	41
18Eh	—	Reserved maintain clear								0000 0000	—
18Fh	—	Reserved maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

2.2.2.1 STATUS Register

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory.

The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable, therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C or DC bits from the STATUS register. For other instructions not affecting any status bits, see the "Instruction Set Summary."

Note:	The C and DC bits operate as a $\overline{borrow}$ and digit $\overline{borrow}$ bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP**: Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
(for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.2 OPTION_REG Register

The OPTION_REG Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the External INT Interrupt, TMR0 and the weak pull-ups on PORTB.

Note: To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: When using low voltage ICSP programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device

2.2.2.3 INTCON Register

The INTCON Register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB Port change and External RB0/INT pin interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.4 PIE1 Register

The PIE1 register contains the individual enable bits for the peripheral interrupts.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE(1):** Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6 **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE:** USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE:** USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE:** Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Note 1: PSPIE is reserved on PIC16F873/876 devices; always maintain this bit clear.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.5 PIR1 Register

The PIR1 register contains the individual flag bits for the peripheral interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF(1)**: Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag
1 = The SSP interrupt condition has occurred, and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI
  - A transmission/reception has taken place.
- I^2C Slave
  - A transmission/reception has taken place.
- I^2C Master
  - A transmission/reception has taken place.
  - The initiated START condition was completed by the SSP module.
  - The initiated STOP condition was completed by the SSP module.
  - The initiated Restart condition was completed by the SSP module.
  - The initiated Acknowledge condition was completed by the SSP module.
  - A START condition occurred while the SSP module was idle (Multi-Master system).
  - A STOP condition occurred while the SSP module was idle (Multi-Master system).

0 = No SSP interrupt condition has occurred.

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1: PSPIF is reserved on PIC16F873/876 devices; always maintain this bit clear.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.6 PIE2 Register

The PIE2 register contains the individual enable bits for the CCP2 peripheral interrupt, the SSP bus collision interrupt, and the EEPROM write operation interrupt.

REGISTER 2-6: PIE2 REGISTER (ADDRESS 8Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	Reserved	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **Reserved**: Always maintain this bit clear

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIE**: EEPROM Write Operation Interrupt Enable
1 = Enable EE Write Interrupt
0 = Disable EE Write Interrupt

bit 3 **BCLIE**: Bus Collision Interrupt Enable
1 = Enable Bus Collision Interrupt
0 = Disable Bus Collision Interrupt

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.7 PIR2 Register

The PIR2 register contains the flag bits for the CCP2 interrupt, the SSP bus collision interrupt and the EEPROM write operation interrupt.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-7: PIR2 REGISTER (ADDRESS 0Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	Reserved	—	EEIF	BCLIF	—	—	CCP2IF
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **Reserved**: Always maintain this bit clear

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIF**: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **BCLIF**: Bus Collision Interrupt Flag bit
1 = A bus collision has occurred in the SSP, when configured for I2C Master mode
0 = No bus collision has occurred

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IF**: CCP2 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.2.2.8 PCON Register

The Power Control (PCON) Register contains flag bits to allow differentiation between a Power-on Reset (POR), a Brown-out Reset (BOR), a Watchdog Reset (WDT), and an external MCLR Reset.

Note:	$\overline{BOR}$ is unknown on POR. It must be set by the user and checked on subsequent RESETS to see if BOR is clear, indicating a brown-out has occurred. The BOR status bit is a "don't care" and is not predictable if the brown-out circuit is disabled (by clearing the BODEN bit in the configuration word).

REGISTER 2-8: PCON REGISTER (ADDRESS 8Eh)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-1
—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7-2 **Unimplemented:** Read as '0'

bit 1 $\overline{POR}$: Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

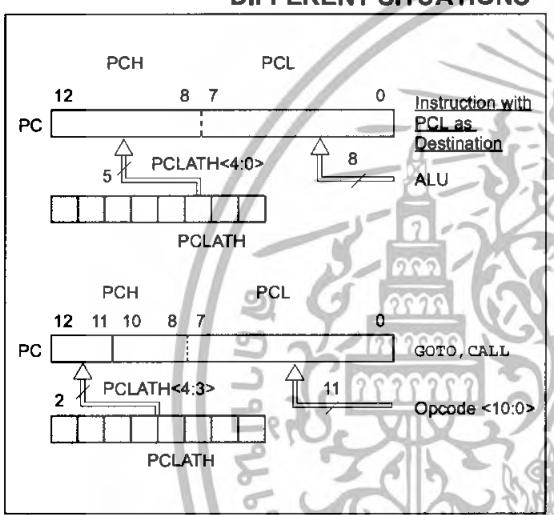
bit 0 $\overline{BOR}$: Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.3 PCL and PCLATH

The program counter (PC) is 13-bits wide. The low byte comes from the PCL register, which is a readable and writable register. The upper bits (PC<12:8>) are not readable, but are indirectly writable through the PCLATH register. On any RESET, the upper bits of the PC will be cleared. Figure 2-5 shows the two situations for the loading of the PC. The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in the figure shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> → PCH).

FIGURE 2-5: LOADING OF PC IN DIFFERENT SITUATIONS



2.3.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256 byte block). Refer to the application note, *"Implementing a Table Read"* (AN556).

2.3.2 STACK

The PIC16F87X family has an 8-level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed, or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

Note 1: There are no status bits to indicate stack overflow or stack underflow conditions.

2: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW and RETFIE instructions, or the vectoring to an interrupt address.

2.4 Program Memory Paging

All PIC16F87X devices are capable of addressing a continuous 8K word block of program memory. The CALL and GOTO instructions provide only 11 bits of address to allow branching within any 2K program memory page. When doing a CALL or GOTO instruction, the upper 2 bits of the address are provided by PCLATH<4:3>. When doing a CALL or GOTO instruction, the user must ensure that the page select bits are programmed so that the desired program memory page is addressed. If a return from a CALL instruction (or interrupt) is executed, the entire 13-bit PC is popped off the stack. Therefore, manipulation of the PCLATH<4:3> bits is not required for the return instructions (which POPs the address from the stack).

Note: The contents of the PCLATH register are unchanged after a RETURN or RETFIE instruction is executed. The user must rewrite the contents of the PCLATH register for any subsequent subroutine calls or GOTO instructions.

Example 2-1 shows the calling of a subroutine in page 1 of the program memory. This example assumes that PCLATH is saved and restored by the Interrupt Service Routine (if interrupts are used).

EXAMPLE 2-1: CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0

```
        ORG 0x500
        BCF PCLATH,4
        BSF PCLATH,3    ;Select page 1
                        ;(800h-FFFh)
        CALL SUB1_P1    ;Call subroutine in
        :               ;page 1 (800h-FFFh)
        :
        ORG 0x900       ;page 1 (800h-FFFh)
SUB1_P1
        :               ;called subroutine
                        ;page 1 (800h-FFFh)
        :
        RETURN          ;return to
                        ;Call subroutine
                        ;in page 0
                        ;(000h-7FFh)
```

2.5 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

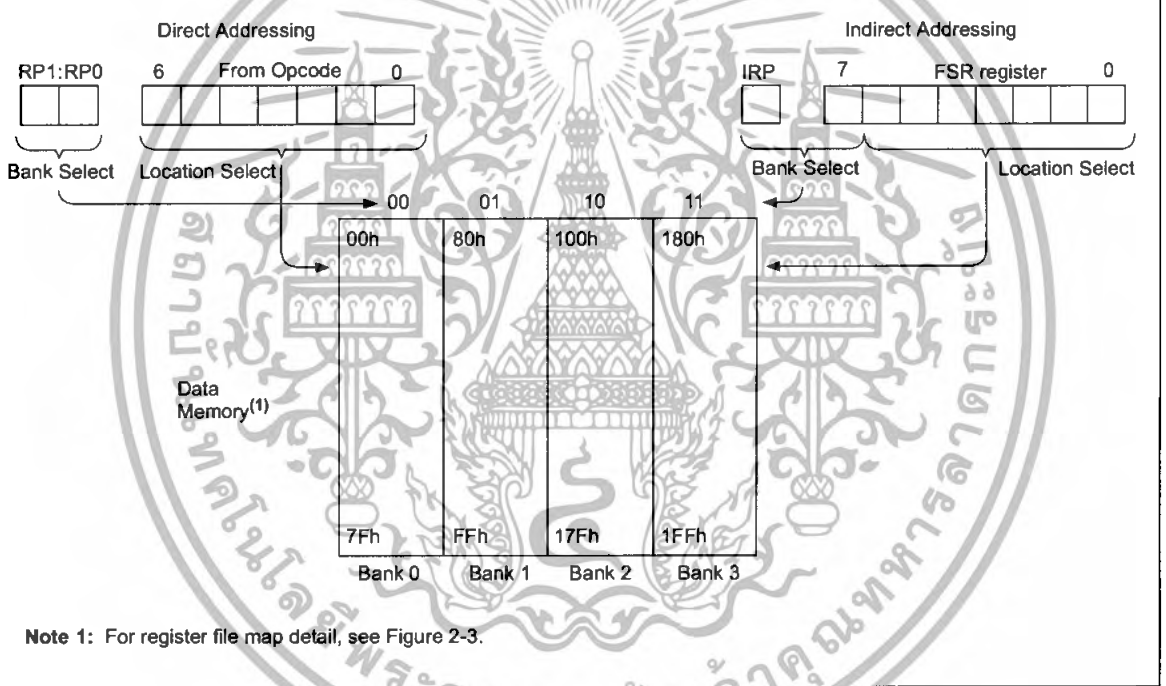
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = '0') will read 00h. Writing to the INDF register indirectly results in a no operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-6.

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: INDIRECT ADDRESSING

```
        MOVLW  0x20     ;initialize pointer
        MOVWF  FSR      ;to RAM
NEXT    CLRF   INDF     ;clear INDF register
        INCF   FSR,F    ;inc pointer
        BTFSS  FSR,4    ;all done?
        GOTO   NEXT     ;no clear next
CONTINUE
        :               ;yes continue
```

FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Note 1: For register file map detail, see Figure 2-3.

NOTES:

19-4323; Rev 9; 4/00

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multidrop RS-232 Networks

Features

Superior to Bipolar

- **Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)**
- **Low-Power Receive Mode in Shutdown (MAX223/MAX242)**
- **Meet All EIA/TIA-232E and V.28 Specifications**
- **Multiple Drivers and Receivers**
- **3-State Driver and Receiver Outputs**
- **Open-Line Detection (MAX243)**

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

*Contact factory for dice specifications.

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	4.7/10	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} - 0.3V)
R_{IN} (Except MAX220) ..... ±30V
R_{IN} (MAX220) ..... ±25V
T_{OUT} (Except MAX220) (Note 1) ..... ±15V
T_{OUT} (MAX220) ..... ±13.2V
Output Voltages
T_{OUT} ..... ±15V
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) ... 842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C) ... 889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) .. 440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ... 696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C) 800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) 800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C) 842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ ..... 0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ ..... -40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate	All except MAX220, normal operation			200	116	kb/s
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

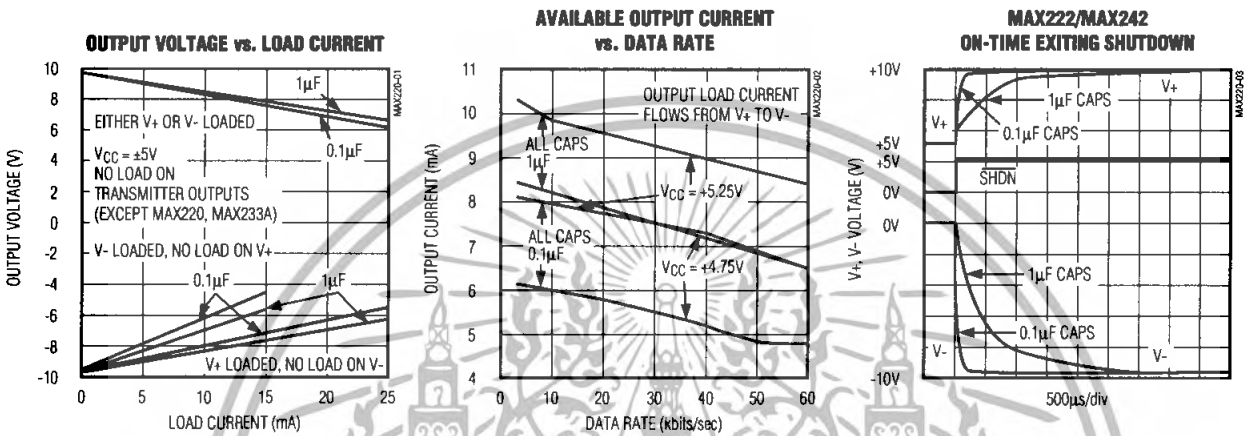
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (normal operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500.	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

Note 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243



ABSOLUTE MAXIMUM RATINGS—MAX223/MAX230–MAX241

V_{CC} ..... -0.3V to +6V
V+ ..... (V_{CC} - 0.3V) to +14V
V- ..... +0.3V to -14V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} + 0.3V)
R_{IN} ..... ±30V
Output Voltages
T_{OUT} ..... (V+ + 0.3V) to (V- - 0.3V)
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Short-Circuit Duration, T_{OUT} ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C) 800mW
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
20-Pin Plastic DIP (derate 11.11mW/°C above +70°C) 889mW
24-Pin Narrow Plastic DIP
(derate 13.33mW/°C above +70°C) 1.07W
24-Pin Plastic DIP (derate 9.09mW/°C above +70°C) 500mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Wide SO (derate 10 00mW/°C above +70°C) 800mW
24-Pin Wide SO (derate 11.76mW/°C above +70°C) 941mW
28-Pin Wide SO (derate 12.50mW/°C above +70°C) 1W
44-Pin Plastic FP (derate 11.11mW/°C above +70°C) 889mW
14-Pin CERDIP (derate 9.09mW/°C above +70°C) 727mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) 800mW
20-Pin CERDIP (derate 11.11mW/°C above +70°C) 889mW
24-Pin Narrow CERDIP
(derate 12.50mW/°C above +70°C) 1W
24-Pin Sidebraze (derate 20.0mW/°C above +70°C) 1.6W
28-Pin SSOP (derate 9.52mW/°C above +70°C) 762mW
Operating Temperature Ranges
MAX2 _ _ C _ _ ..... 0°C to +70°C
MAX2 _ _ E _ _ ..... -40°C to +85°C
MAX2 _ _ M _ _ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX223/MAX230–MAX241

(MAX223/230/232/234/236/237/238/240/241, V_{CC} = +5V ±10; MAX233/MAX235, V_{CC} = 5V ±5%, C1–C4 = 1.0µF; MAX231/MAX239, V_{CC} = 5V ±10%; V+ = 7.5V to 13.2V; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to ground		±5.0	±7.3		V
V_{CC} Power-Supply Current	No load, T_A = +25°C	MAX232/233		5	10	mA
		MAX223/230/234–238/240/241		7	15	
		MAX231/239		0.4	1	
V+ Power-Supply Current		MAX231		1.8	5	mA
		MAX239		5	15	
Shutdown Supply Current	T_A = +25°C	MAX223		15	50	µA
		MAX230/235/236/240/241		1	10	
Input Logic Threshold Low	T_{IN}; EN, $\overline{SHDN}$ (MAX233); $\overline{EN}$, SHDN (MAX230/235–241)				0.8	V
Input Logic Threshold High	T_{IN}		2.0			V
	EN, $\overline{SHDN}$ (MAX223); $\overline{EN}$, SHDN (MAX230/235/236/240/241)		2.4			
Logic Pull-Up Current	T_{IN} = 0V			1.5	200	µA
Receiver Input Voltage Operating Range			-30		30	V

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ELECTRICAL CHARACTERISTICS—MAX223/MAX230–MAX241 (continued)

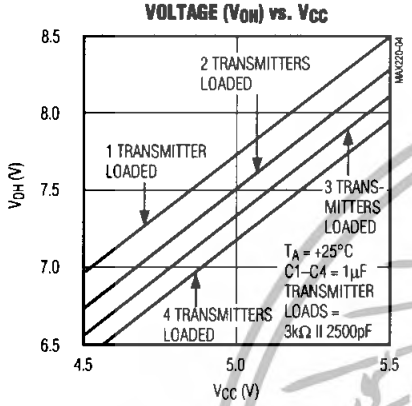
(MAX223/230/232/234/236/237/238/240/241, V_{CC} = +5V ±10; MAX233/MAX235, V_{CC} = 5V ±5%, C1–C4 = 1.0µF; MAX231/MAX239, V_{CC} = 5V ±10%; V+ = 7.5V to 13.2V; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

PARAMETER	CONDITIONS			MIN	TYP	MAX	UNITS
RS-232 Input Threshold Low	T_A = +25°C, V_{CC} = 5V	Normal operation $\overline{SHDN}$ = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)		0.8	1.2		V
		Shutdown (MAX223) $\overline{SHDN}$ = 0V, EN = 5V ($R4_{IN}$, $R5_{IN}$)		0.6	1.5		
RS-232 Input Threshold High	T_A = +25°C, V_{CC} = 5V	Normal operation $\overline{SHDN}$ = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)			1.7	2.4	V
		Shutdown (MAX223) $\overline{SHDN}$ = 0V, EN = 5V ($R4_{IN}$, $R5_{IN}$)			1.5	2.4	
RS-232 Input Hysteresis	V_{CC} = 5V, no hysteresis in shutdown			0.2	0.5	1.0	V
RS-232 Input Resistance	T_A = +25°C, V_{CC} = 5V			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 1.6mA (MAX231/232/233, I_{OUT} = 3.2mA)					0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1mA			3.5	V_{CC} - 0.4		V
TTL/CMOS Output Leakage Current	0V ≤ R_{OUT} ≤ V_{CC}; EN = 0V (MAX223); $\overline{EN}$ = V_{CC} (MAX235–241)				0.05	±10	µA
Receiver Output Enable Time	Normal operation	MAX223			600		ns
		MAX235/236/239/240/241			400		
Receiver Output Disable Time	Normal operation	MAX223			900		ns
		MAX235/236/239/240/241			250		
Propagation Delay	RS-232 IN to TTL/CMOS OUT, C_L = 150pF	Normal operation			0.5	10	µs
		$\overline{SHDN}$ = 0V (MAX223)	t_{PHLS}		4	40	
			t_{PLHS}		6	40	
Transition Region Slew Rate	MAX223/MAX230/MAX234–241, T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 2500pF, measured from +3V to -3V or -3V to +3V			3	5.1	30	V/µs
	MAX231/MAX232/MAX233, T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 2500pF, measured from +3V to -3V or -3V to +3V				4	30	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V			300			Ω
Transmitter Output Short-Circuit Current					±10		mA

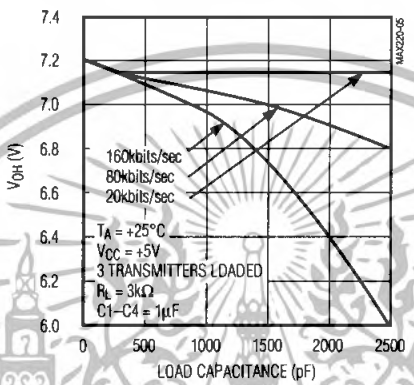
Typical Operating Characteristics

MAX223/MAX230–MAX241

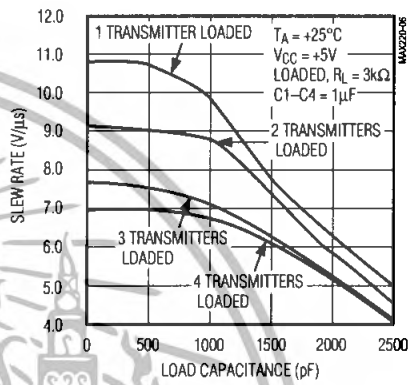
TRANSMITTER OUTPUT VOLTAGE (V_{OH}) vs. V_{CC}



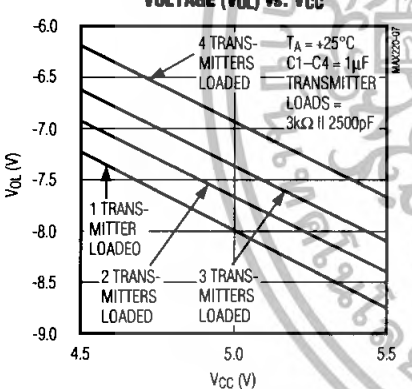
TRANSMITTER OUTPUT VOLTAGE (V_{OH}) vs. LOAD CAPACITANCE AT DIFFERENT DATA RATES



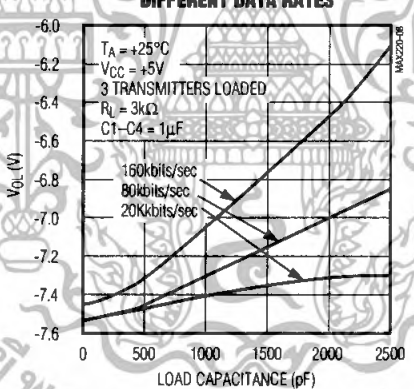
TRANSMITTER SLEW RATE vs. LOAD CAPACITANCE



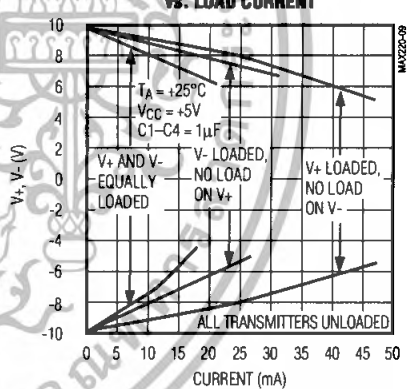
TRANSMITTER OUTPUT VOLTAGE (V_{OL}) vs. V_{CC}



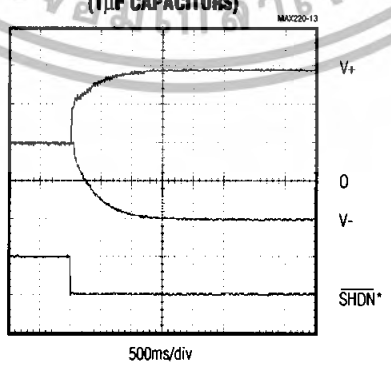
TRANSMITTER OUTPUT VOLTAGE (V_{OL}) vs. LOAD CAPACITANCE AT DIFFERENT DATA RATES



TRANSMITTER OUTPUT VOLTAGE (V+, V-) vs. LOAD CURRENT



V+, V- WHEN EXITING SHUTDOWN (1µF CAPACITORS)



*SHUTDOWN POLARITY IS REVERSED FOR NON MAX241 PARTS

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX225/MAX244–MAX249

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
T_{IN}, $\overline{ENA}$, $\overline{ENB}$, $\overline{ENR}$, $\overline{ENT}$, $\overline{ENRA}$, $\overline{ENRB}$, $\overline{ENTA}$, $\overline{ENTB}$ ..... -0.3V to (V_{CC} + 0.3V)
R_{IN} ..... ±25V
T_{OUT} (Note 3) ..... ±15V
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Short Circuit (one output at a time)
T_{OUT} to GND ..... Continuous
R_{OUT} to GND ..... Continuous

Continuous Power Dissipation (T_A = +70°C)
28-Pin Wide SO (derate 12.50mW/°C above +70°C) 1W
40-Pin Plastic DIP (derate 11.11mW/°C above +70°C) ... 611mW
44-Pin PLCC (derate 13.33mW/°C above +70°C) 1.07W
Operating Temperature Ranges
MAX225C_ _, MAX24_C_ _ ..... 0°C to +70°C
MAX225E_ _, MAX24_E_ _ ..... -40°C to +85°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Note 4: Input voltage measured with transmitter output in a high-impedance state, shutdown, or V_{CC} = 0V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX225/MAX244–MAX249

(MAX225, V_{CC} = 5.0V ±5%; MAX244–MAX249, V_{CC} = +5.0V ±10%, external capacitors C1–C4 = 1µF; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High			2	1.4		V
Logic Pull-Up/Input Current	Tables 1a–1d	Normal operation		10	50	µA
		Shutdown		±0.01	±1	
Data Rate	Tables 1a–1d, normal operation			120	64	kbits/sec
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±7.5		V
Output Leakage Current (shutdown)	Tables 1a–1d	$\overline{ENA}$, $\overline{ENB}$, $\overline{ENT}$, $\overline{ENTA}$, $\overline{ENTB}$ = V_{CC}, V_{OUT} = ±15V		±0.01	±25	µA
		V_{CC} = 0V, V_{OUT} = ±15V		±0.01	±25	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V (Note 4)		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±30		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±25	V
RS-232 Input Threshold Low	V_{CC} = 5V		0.8	1.3		V
RS-232 Input Threshold High	V_{CC} = 5V			1.8	2.4	V
RS-232 Input Hysteresis	V_{CC} = 5V		0.2	0.5	1.0	V
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		
TTL/CMOS Output Leakage Current	Normal operation, outputs disabled, Tables 1a–1d, 0V ≤ V_{OUT} ≤ V_{CC}, $\overline{ENR\_}$ = V_{CC}			±0.05	±0.10	µA

ELECTRICAL CHARACTERISTICS—MAX225/MAX244–MAX249 (continued)

(MAX225, V_{CC} = 5.0V ±5%; MAX244–MAX249, V_{CC} = +5.0V ±10%, external capacitors C1–C4 = 1µF; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

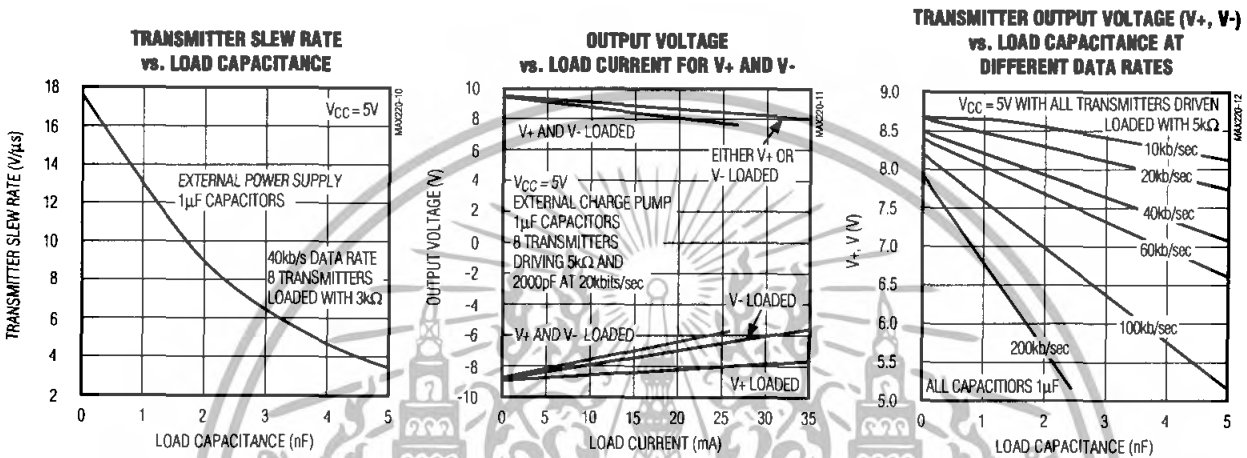
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
POWER SUPPLY AND CONTROL LOGIC						
Operating Supply Voltage		MAX225	4.75		5.25	V
		MAX244–MAX249	4.5		5.5	
V_{CC} Supply Current (normal operation)	No load	MAX225		10	20	mA
		MAX244–MAX249		11	30	
	3kΩ loads on all outputs	MAX225		40		
		MAX244–MAX249		57		
Shutdown Supply Current	T_A = +25°C			8	25	µA
	T_A = T_{MIN} to T_{MAX}				50	
Control Input	Leakage current				±1	µA
	Threshold low			1.4	0.8	V
	Threshold high		2.4	1.4		
AC CHARACTERISTICS						
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V		5	10	30	V/µs
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}			1.3	3.5	µs
	t_{PLHT}			1.5	3.5	
Receiver Propagation Delay TLL to RS-232 (normal operation), Figure 2	t_{PHLR}			0.6	1.5	µs
	t_{PLHR}			0.6	1.5	
Receiver Propagation Delay TLL to RS-232 (low-power mode), Figure 2	t_{PHLS}			0.6	10	µs
	t_{PLHS}			3.0	10	
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}			350		ns
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}			350		ns
Receiver-Output Enable Time, Figure 3	t_{ER}			100	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}			100	500	ns
Transmitter Enable Time	t_{ET}	MAX246–MAX249 (excludes charge-pump start-up)		5		µs
		MAX225/MAX245–MAX249 (includes charge-pump start-up)		10		ms
Transmitter Disable Time, Figure 4	t_{DT}			100		ns

Note 5: The 300Ω minimum specification complies with EIA/TIA-232E, but the actual resistance when in shutdown mode or V_{CC} = 0V is 10MΩ as is implied by the leakage specification.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX225/MAX244–MAX249



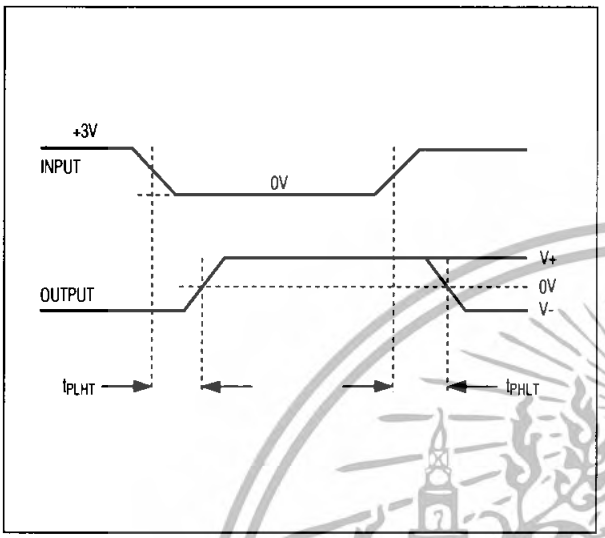


Figure 1. Transmitter Propagation-Delay Timing

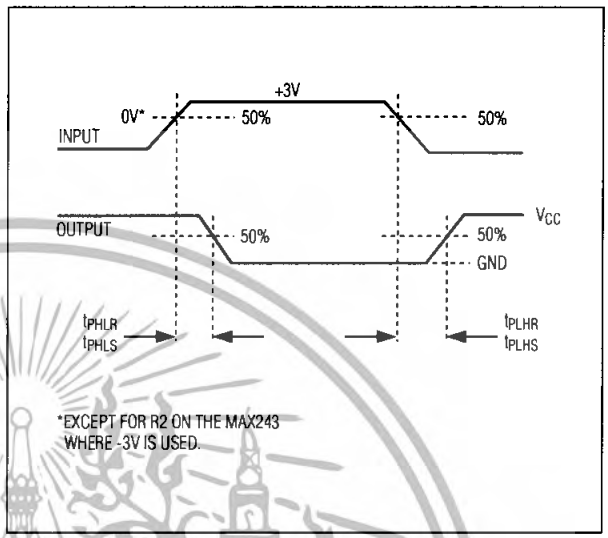


Figure 2. Receiver Propagation-Delay Timing

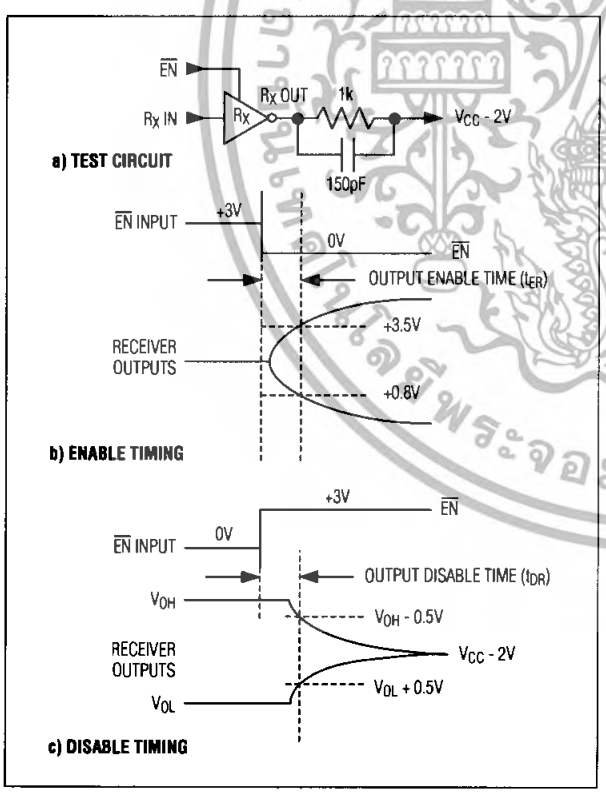


Figure 3. Receiver-Output Enable and Disable Timing

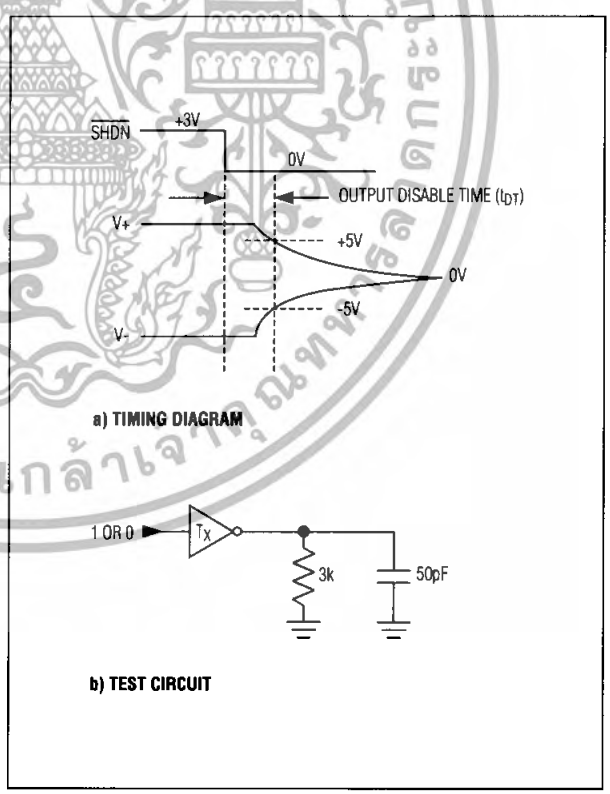


Figure 4. Transmitter-Output Disable Timing

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Table 1a. MAX245 Control Pin Configurations

$\overline{ENT}$	$\overline{ENR}$	OPERATION STATUS	TRANSMITTERS	RECEIVERS
0	0	Normal Operation	All Active	All Active
0	1	Normal Operation	All Active	All 3-State
1	0	Shutdown	All 3-State	All Low-Power Receive Mode
1	1	Shutdown	All 3-State	All 3-State

Table 1b. MAX245 Control Pin Configurations

$\overline{ENT}$	$\overline{ENR}$	OPERATION STATUS	TRANSMITTERS		RECEIVERS	
			TA1–TA4	TB1–TB4	RA1–RA5	RB1–RB5
0	0	Normal Operation	All Active	All Active	All Active	All Active
0	1	Normal Operation	All Active	All Active	RA1–RA4 3-State, RA5 Active	RB1–RB4 3-State, RB5 Active
1	0	Shutdown	All 3-State	All 3-State	All Low-Power Receive Mode	All Low-Power Receive Mode
1	1	Shutdown	All 3-State	All 3-State	RA1–RA4 3-State, RA5 Low-Power Receive Mode	RB1–RB4 3-State, RB5 Low-Power Receive Mode

Table 1c. MAX246 Control Pin Configurations

$\overline{ENA}$	$\overline{ENB}$	OPERATION STATUS	TRANSMITTERS		RECEIVERS	
			TA1–TA4	TB1–TB4	RA1–RA5	RB1–RB5
0	0	Normal Operation	All Active	All Active	All Active	All Active
0	1	Normal Operation	All Active	All 3-State	All Active	RB1–RB4 3-State, RB5 Active
1	0	Shutdown	All 3-State	All Active	RA1–RA4 3-State, RA5 Active	All Active
1	1	Shutdown	All 3-State	All 3-State	RA1–RA4 3-State, RA5 Low-Power Receive Mode	RB1–RB4 3-State, RA5 Low-Power Receive Mode

Table 1d. MAX247/MAX248/MAX249 Control Pin Configurations

$\overline{ENTA}$	$\overline{ENTB}$	$\overline{ENRA}$	$\overline{ENRB}$	OPERATION STATUS		TRANSMITTERS		RECEIVERS	
					MAX247	TA1–TA4	TB1–TB4	RA1–RA4	RB1–RB5
					MAX248	TA1–TA4	TB1–TB4	RA1–RA4	RB1–RB4
					MAX249	TA1–TA3	TB1–TB3	RA1–RA5	RB1–RB5
0	0	0	0	Normal Operation		All Active	All Active	All Active	All Active
0	0	0	1	Normal Operation		All Active	All Active	All Active	All 3-State, except RB5 stays active on MAX247
0	0	1	0	Normal Operation		All Active	All Active	All 3-State	All Active
0	0	1	1	Normal Operation		All Active	All Active	All 3-State	All 3-State, except RB5 stays active on MAX247
0	1	0	0	Normal Operation		All Active	All 3-State	All Active	All Active
0	1	0	1	Normal Operation		All Active	All 3-State	All Active	All 3-State, except RB5 stays active on MAX247
0	1	1	0	Normal Operation		All Active	All 3-State	All 3-State	All Active
0	1	1	1	Normal Operation		All Active	All 3-State	All 3-State	All 3-State, except RB5 stays active on MAX247
1	0	0	0	Normal Operation		All 3-State	All Active	All Active	All Active
1	0	0	1	Normal Operation		All 3-State	All Active	All Active	All 3-State, except RB5 stays active on MAX247
1	0	1	0	Normal Operation		All 3-State	All Active	All 3-State	All Active
1	0	1	1	Normal Operation		All 3-State	All Active	All 3-State	All 3-State, except RB5 stays active on MAX247
1	1	0	0	Shutdown		All 3-State	All 3-State	Low-Power Receive Mode	Low-Power Receive Mode
1	1	0	1	Shutdown		All 3-State	All 3-State	Low-Power Receive Mode	All 3-State, except RB5 stays active on MAX247
1	1	1	0	Shutdown		All 3-State	All 3-State	All 3-State	Low-Power Receive Mode
1	1	1	1	Shutdown		All 3-State	All 3-State	All 3-State	All 3-State, except RB5 stays active on MAX247

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Detailed Description

The MAX220–MAX249 contain four sections: dual charge-pump DC-DC voltage converters, RS-232 drivers, RS-232 receivers, and receiver and transmitter enable control inputs.

Dual Charge-Pump Voltage Converter

The MAX220–MAX249 have two internal charge-pumps that convert +5V to ±10V (unloaded) for RS-232 driver operation. The first converter uses capacitor C1 to double the +5V input to +10V on C3 at the V+ output. The second converter uses capacitor C2 to invert +10V to -10V on C4 at the V- output.

A small amount of power may be drawn from the +10V (V+) and -10V (V-) outputs to power external circuitry (see the *Typical Operating Characteristics* section), except on the MAX225 and MAX245–MAX247, where these pins are not available. V+ and V- are not regulated, so the output voltage drops with increasing load current. Do not load V+ and V- to a point that violates the minimum ±5V EIA/TIA-232E driver output voltage when sourcing current from V+ and V- to external circuitry.

When using the shutdown feature in the MAX222, MAX225, MAX230, MAX235, MAX236, MAX240, MAX241, and MAX245–MAX249, avoid using V+ and V- to power external circuitry. When these parts are shut down, V- falls to 0V, and V+ falls to +5V. For applications where a +10V external supply is applied to the V+ pin (instead of using the internal charge pump to generate +10V), the C1 capacitor must not be installed and the $\overline{SHDN}$ pin must be tied to V_{CC}. This is because V+ is internally connected to V_{CC} in shutdown mode.

RS-232 Drivers

The typical driver output voltage swing is ±8V when loaded with a nominal 5kΩ RS-232 receiver and V_{CC} = +5V. Output swing is guaranteed to meet the EIA/TIA-232E and V.28 specification, which calls for ±5V minimum driver output levels under worst-case conditions. These include a minimum 3kΩ load, V_{CC} = +4.5V, and maximum operating temperature. Unloaded driver output voltage ranges from (V+ -1.3V) to (V- +0.5V).

Input thresholds are both TTL and CMOS compatible. The inputs of unused drivers can be left unconnected since 400kΩ input pull-up resistors to V_{CC} are built in (except for the MAX220). The pull-up resistors force the outputs of unused drivers low because all drivers invert. The internal input pull-up resistors typically source 12µA, except in shutdown mode where the pull-ups are disabled. Driver outputs turn off and enter a high-impedance state—where leakage current is typically microamperes (maximum 25µA)—when in shutdown mode, in three-state mode, or when device power is removed. Outputs can be driven to ±15V. The power-supply current typically drops to 8µA in shutdown mode. The MAX220 does not have pull-up resistors to force the ouputs of the unused drivers low. Connect unused inputs to GND or V_{CC}.

The MAX239 has a receiver three-state control line, and the MAX223, MAX225, MAX235, MAX236, MAX240, and MAX241 have both a receiver three-state control line and a low-power shutdown control. Table 2 shows the effects of the shutdown control and receiver three-state control on the receiver outputs.

The receiver TTL/CMOS outputs are in a high-impedance, three-state mode whenever the three-state enable line is high (for the MAX225/MAX235/MAX236/MAX239–MAX241), and are also high-impedance whenever the shutdown control line is high.

When in low-power shutdown mode, the driver outputs are turned off and their leakage current is less than 1µA with the driver output pulled to ground. The driver output leakage remains less than 1µA, even if the transmitter output is backdriven between 0V and (V_{CC} + 6V). Below -0.5V, the transmitter is diode clamped to ground with 1kΩ series impedance. The transmitter is also zener clamped to approximately V_{CC} + 6V, with a series impedance of 1kΩ.

The driver output slew rate is limited to less than 30V/µs as required by the EIA/TIA-232E and V.28 specifications. Typical slew rates are 24V/µs unloaded and 10V/µs loaded with 3Ω and 2500pF.

RS-232 Receivers

EIA/TIA-232E and V.28 specifications define a voltage level greater than 3V as a logic 0, so all receivers invert. Input thresholds are set at 0.8V and 2.4V, so receivers respond to TTL level inputs as well as EIA/TIA-232E and V.28 levels.

The receiver inputs withstand an input overvoltage up to ±25V and provide input terminating resistors with

Table 2. Three-State Control of Receivers

PART	SHDN	$\overline{SHDN}$	EN	$\overline{EN(R)}$	RECEIVERS
MAX223	—	Low High High	X Low High	—	High Impedance Active High Impedance
MAX225	—	—	—	Low High	High Impedance Active
MAX235 MAX236 MAX240	Low Low High	—	—	Low High X	High Impedance Active High Impedance

nominal 5kΩ values. The receivers implement Type 1 interpretation of the fault conditions of V.28 and EIA/TIA-232E.

The receiver input hysteresis is typically 0.5V with a guaranteed minimum of 0.2V. This produces clear output transitions with slow-moving input signals, even with moderate amounts of noise and ringing. The receiver propagation delay is typically 600ns and is independent of input swing direction.

Low-Power Receive Mode

The low-power receive-mode feature of the MAX223, MAX242, and MAX245–MAX249 puts the IC into shutdown mode but still allows it to receive information. This is important for applications where systems are periodically awakened to look for activity. Using low-power receive mode, the system can still receive a signal that will activate it on command and prepare it for communication at faster data rates. This operation conserves system power.

Negative Threshold—MAX243

The MAX243 is pin compatible with the MAX232A, differing only in that RS-232 cable fault protection is removed on one of the two receiver inputs. This means that control lines such as CTS and RTS can either be driven or left floating without interrupting communication. Different cables are not needed to interface with different pieces of equipment.

The input threshold of the receiver without cable fault protection is -0.8V rather than +1.4V. Its output goes positive only if the input is connected to a control line that is actively driven negative. If not driven, it defaults to the 0 or "OK to send" state. Normally, the MAX243's other receiver (+1.4V threshold) is used for the data line (TD or RD), while the negative threshold receiver is connected to the control line (DTR, DTS, CTS, RTS, etc.).

Other members of the RS-232 family implement the optional cable fault protection as specified by EIA/TIA-232E specifications. This means a receiver output goes high whenever its input is driven negative, left floating, or shorted to ground. The high output tells the serial communications IC to stop sending data. To avoid this, the control lines must either be driven or connected with jumpers to an appropriate positive voltage level.

Shutdown—MAX222-MAX242

On the MAX222, MAX235, MAX236, MAX240, and MAX241, all receivers are disabled during shutdown. On the MAX223 and MAX242, two receivers continue to operate in a reduced power mode when the chip is in shutdown. Under these conditions, the propagation delay increases to about 2.5µs for a high-to-low input transition. When in shutdown, the receiver acts as a CMOS inverter with no hysteresis. The MAX223 and MAX242 also have a receiver output enable input ($\overline{EN}$ for the MAX242 and EN for the MAX223) that allows receiver output control independent of $\overline{SHDN}$ (SHDN for MAX241). With all other devices, $\overline{SHDN}$ (SHDN for MAX241) also disables the receiver outputs.

The MAX225 provides five transmitters and five receivers, while the MAX245 provides ten receivers and eight transmitters. Both devices have separate receiver and transmitter-enable controls. The charge pumps turn off and the devices shut down when a logic high is applied to the ENT input. In this state, the supply current drops to less than 25µA and the receivers continue to operate in a low-power receive mode. Driver outputs enter a high-impedance state (three-state mode). On the MAX225, all five receivers are controlled by the $\overline{ENR}$ input. On the MAX245, eight of the receiver outputs are controlled by the $\overline{ENR}$ input, while the remaining two receivers (RA5 and RB5) are always active. RA1–RA4 and RB1–RB4 are put in a three-state mode when $\overline{ENR}$ is a logic high.

Receiver and Transmitter Enable Control Inputs

The MAX225 and MAX245–MAX249 feature transmitter and receiver enable controls.

The receivers have three modes of operation: full-speed receive (normal active), three-state (disabled), and low-power receive (enabled receivers continue to function at lower data rates). The receiver enable inputs control the full-speed receive and three-state modes. The transmitters have two modes of operation: full-speed transmit (normal active) and three-state (disabled). The transmitter enable inputs also control the shutdown mode. The device enters shutdown mode when all transmitters are disabled. Enabled receivers function in the low-power receive mode when in shutdown.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Tables 1a–1d define the control states. The MAX244 has no control pins and is not included in these tables.

The MAX246 has ten receivers and eight drivers with two control pins, each controlling one side of the device. A logic high at the A-side control input ($\overline{ENA}$) causes the four A-side receivers and drivers to go into a three-state mode. Similarly, the B-side control input ($\overline{ENB}$) causes the four B-side drivers and receivers to go into a three-state mode. As in the MAX245, one A-side and one B-side receiver (RA5 and RB5) remain active at all times. The entire device is put into shutdown mode when both the A and B sides are disabled ($\overline{ENA}$ = $\overline{ENB}$ = +5V).

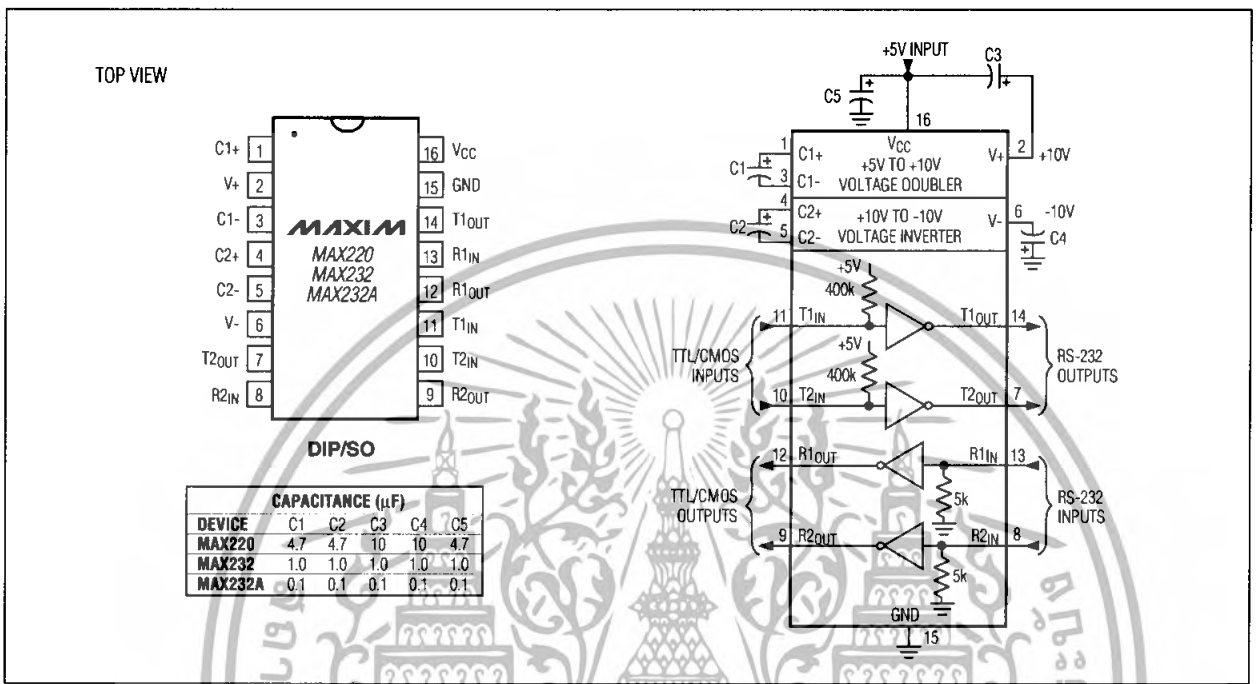
The MAX247 provides nine receivers and eight drivers with four control pins. The $\overline{ENRA}$ and $\overline{ENRB}$ receiver enable inputs each control four receiver outputs. The $\overline{ENTA}$ and $\overline{ENTB}$ transmitter enable inputs each control four drivers. The ninth receiver (RB5) is always active. The device enters shutdown mode with a logic high on both $\overline{ENTA}$ and $\overline{ENTB}$.

The MAX248 provides eight receivers and eight drivers with four control pins. The $\overline{ENRA}$ and $\overline{ENRB}$ receiver enable inputs each control four receiver outputs. The $\overline{ENTA}$ and $\overline{ENTB}$ transmitter enable inputs control four drivers each. This part does not have an always-active receiver. The device enters shutdown mode and transmitters go into a three-state mode with a logic high on both $\overline{ENTA}$ and $\overline{ENTB}$.

The MAX249 provides ten receivers and six drivers with four control pins. The $\overline{ENRA}$ and $\overline{ENRB}$ receiver enable inputs each control five receiver outputs. The $\overline{ENTA}$ and $\overline{ENTB}$ transmitter enable inputs control three drivers each. There is no always-active receiver. The device enters shutdown mode and transmitters go into a three-state mode with a logic high on both $\overline{ENTA}$ and $\overline{ENTB}$. In shutdown mode, active receivers operate in a low-power receive mode at data rates up to 20kbits/sec.

Applications Information

Figures 5 through 25 show pin configurations and typical operating circuits. In applications that are sensitive to power-supply noise, V_{CC} should be decoupled to ground with a capacitor of the same value as C1 and C2 connected as close as possible to the device.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

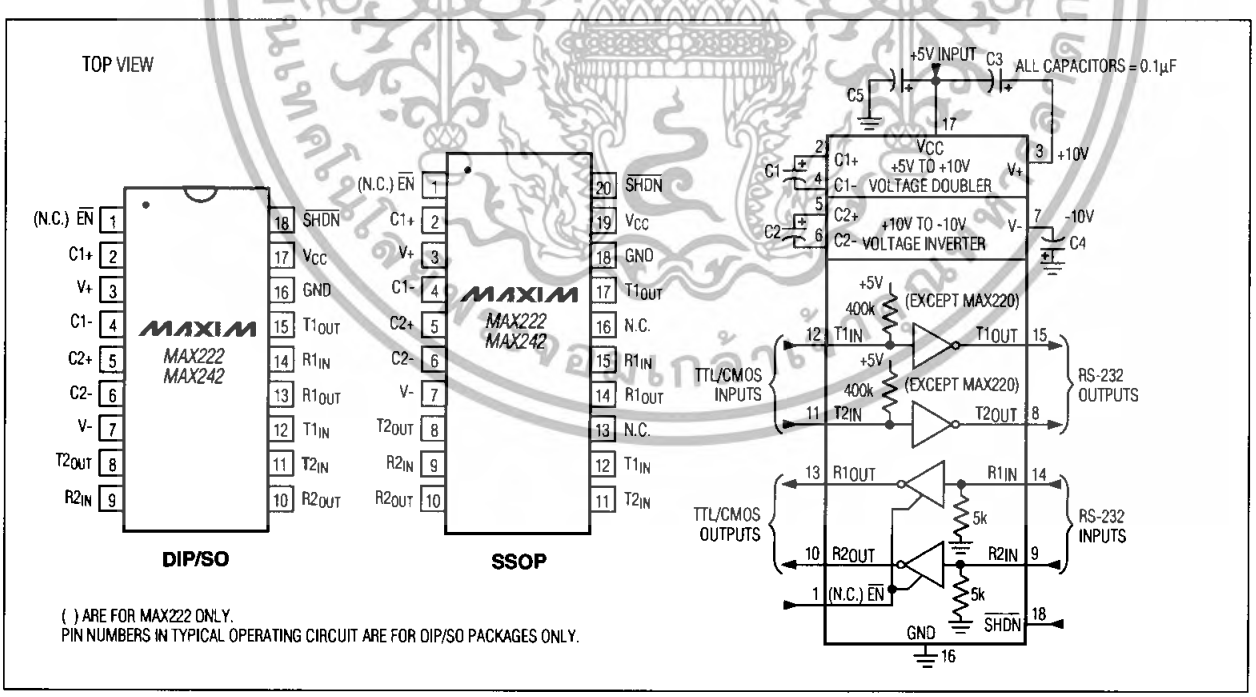


Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

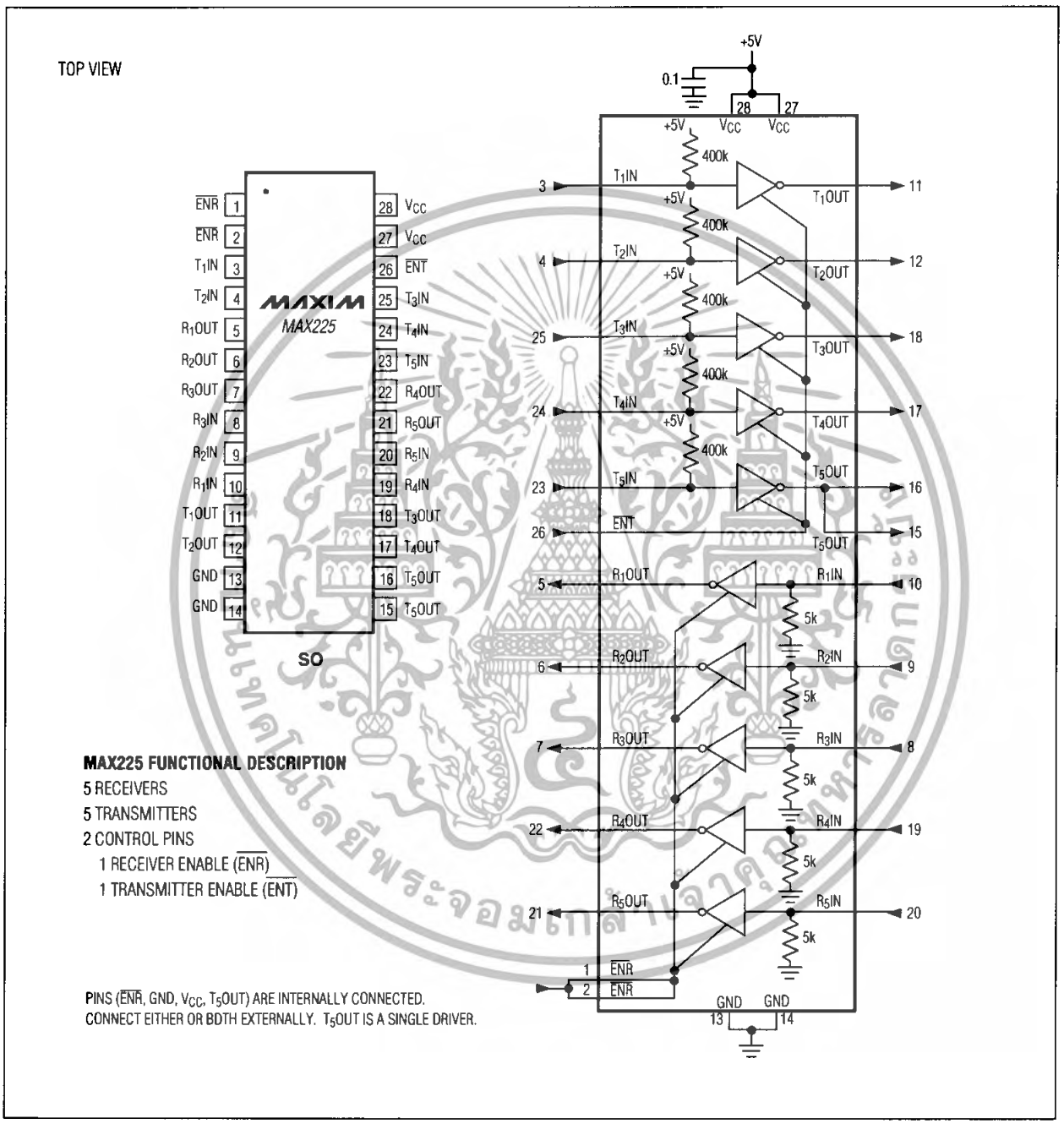


Figure 7. MAX225 Pin Configuration and Typical Operating Circuit

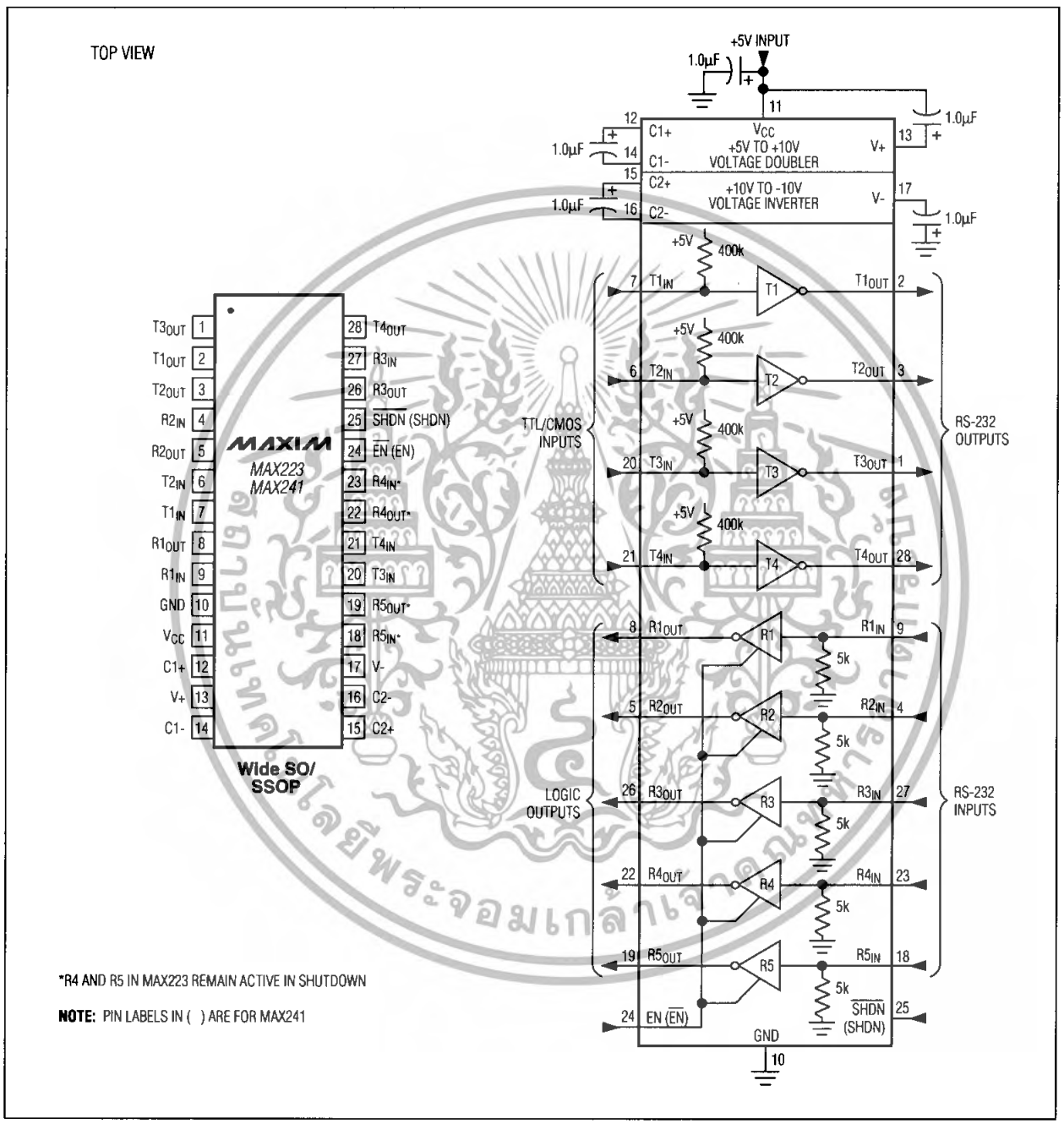


Figure 8. MAX223/MAX241 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

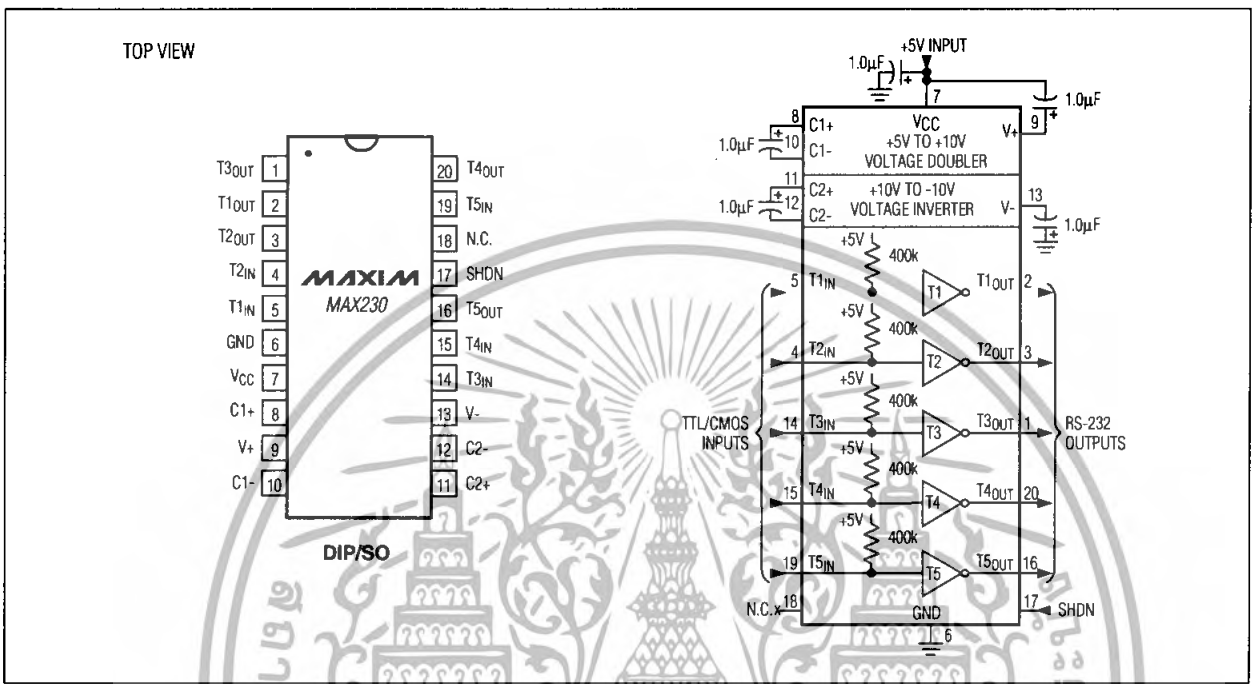


Figure 9. MAX230 Pin Configuration and Typical Operating Circuit

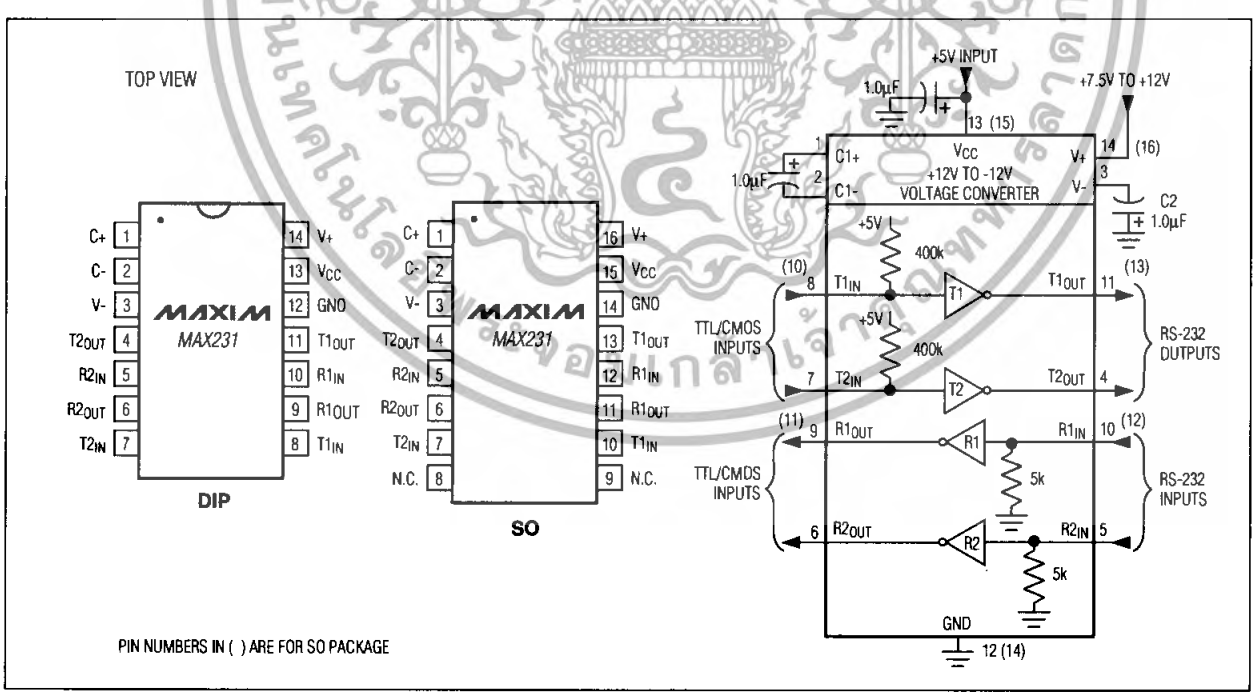


Figure 10. MAX231 Pin Configurations and Typical Operating Circuit

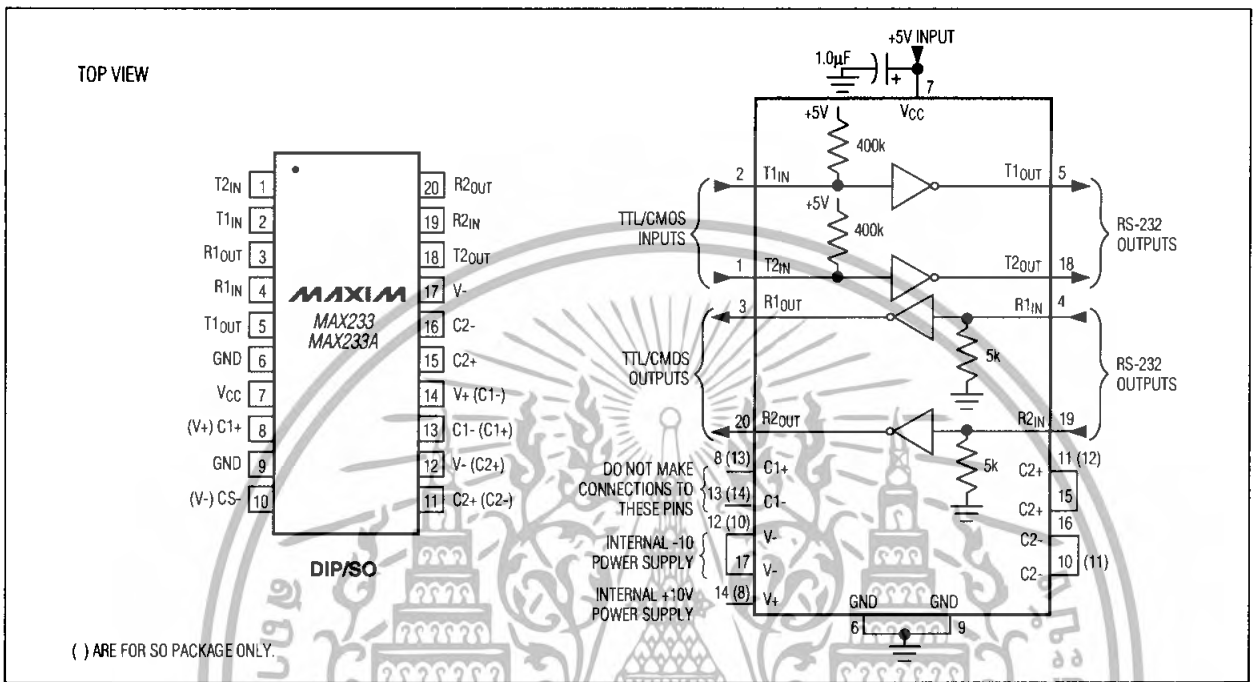


Figure 11. MAX233/MAX233A Pin Configuration and Typical Operating Circuit

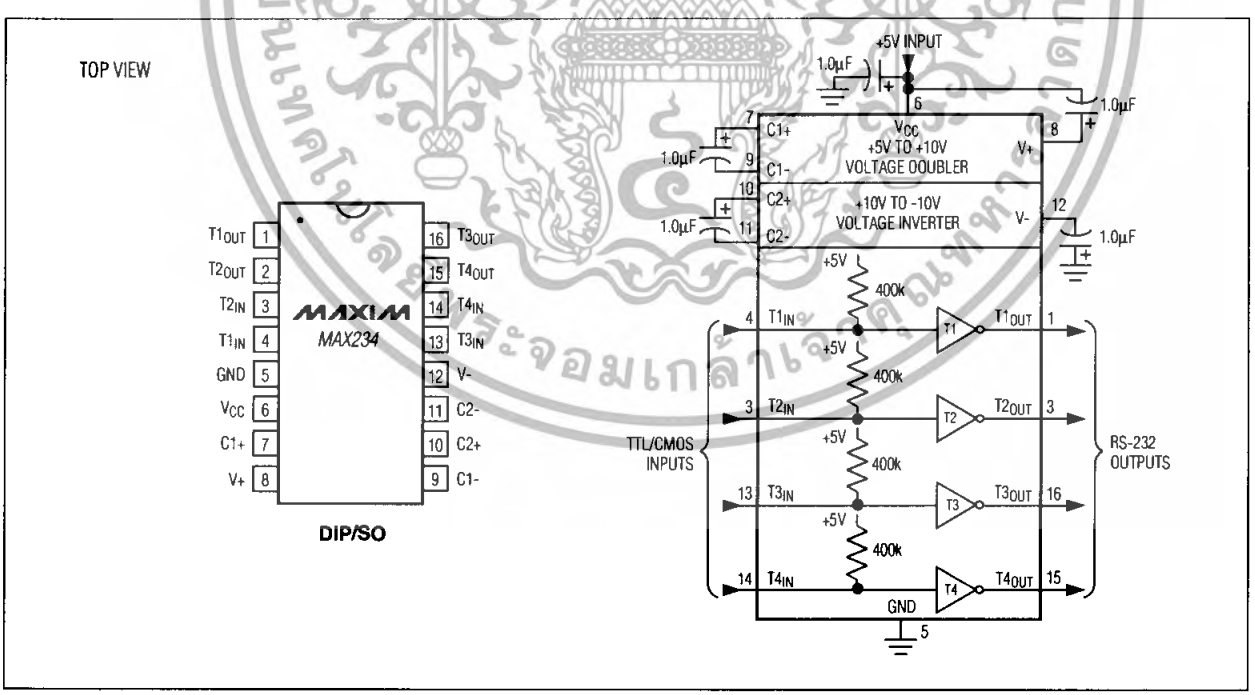


Figure 12. MAX234 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

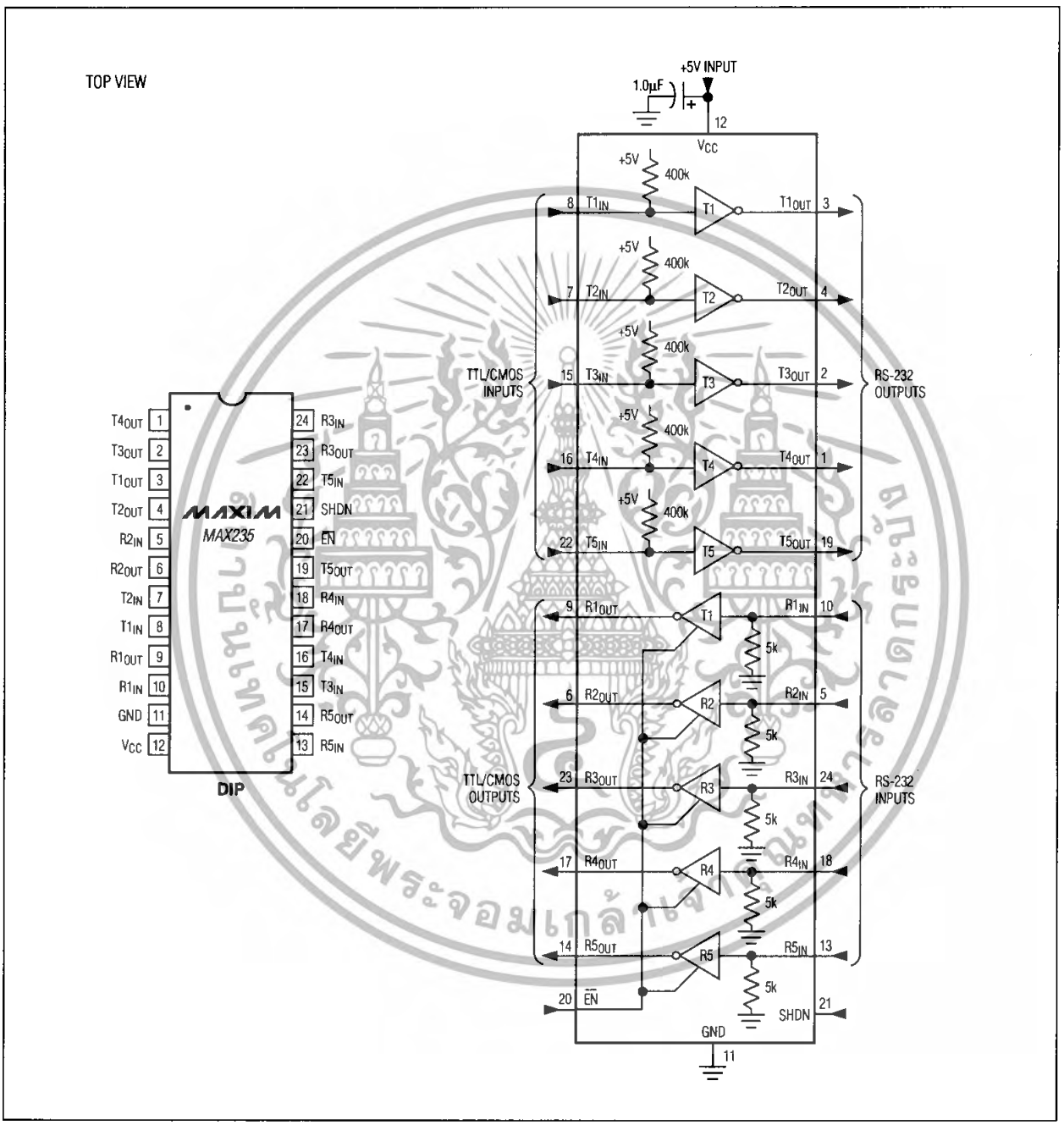


Figure 13. MAX235 Pin Configuration and Typical Operating Circuit

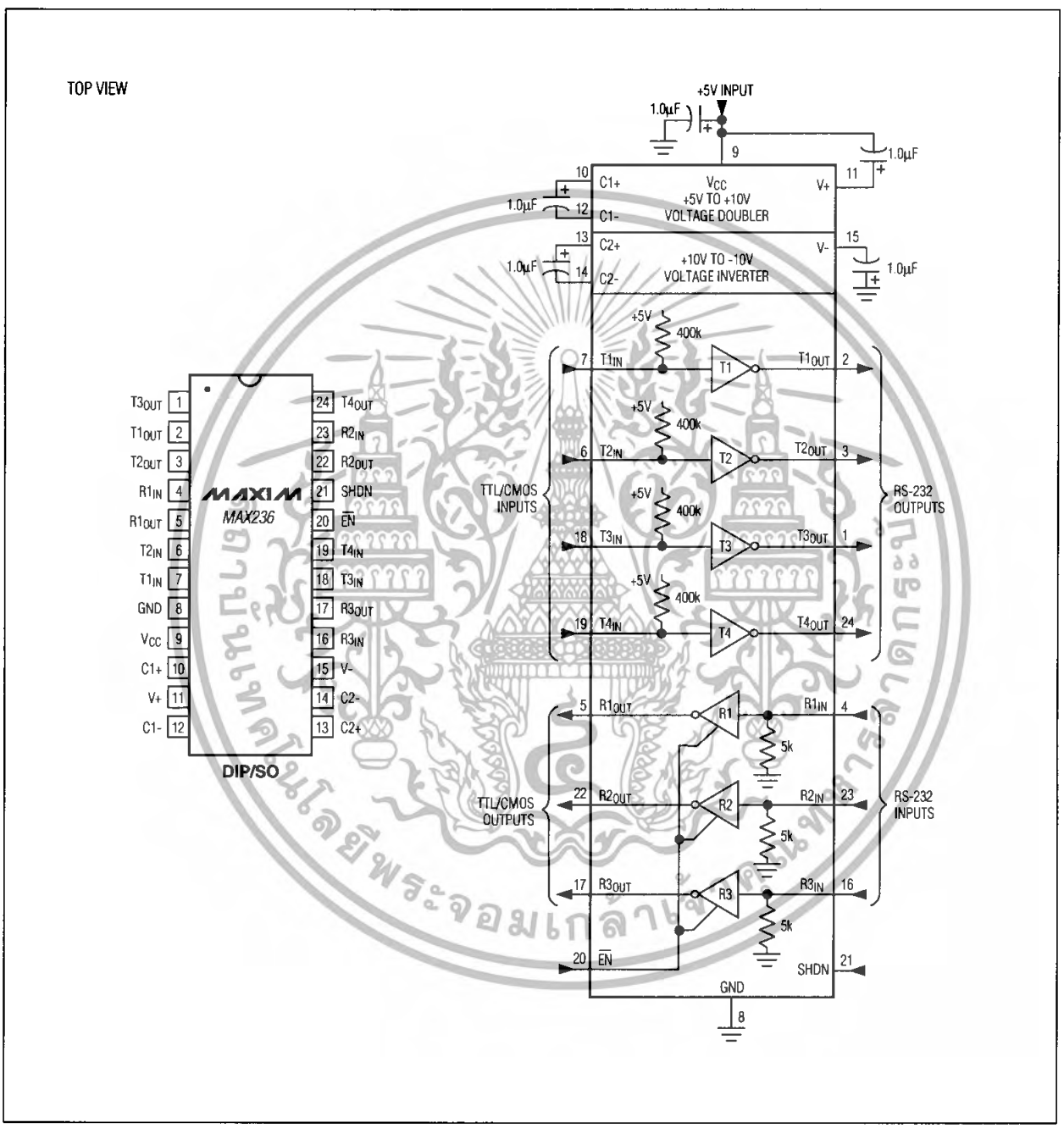


Figure 14. MAX236 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

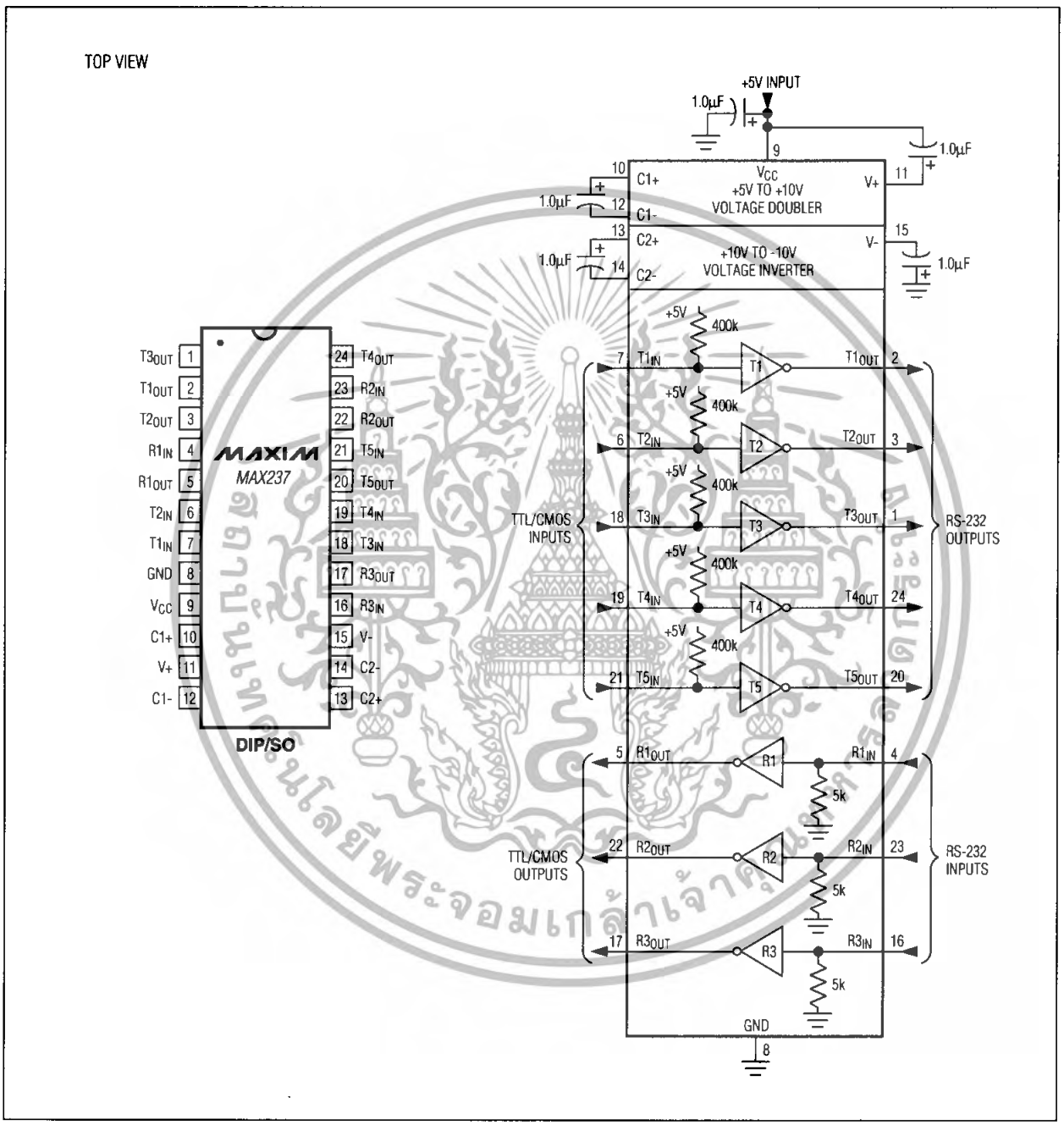


Figure 15. MAX237 Pin Configuration and Typical Operating Circuit

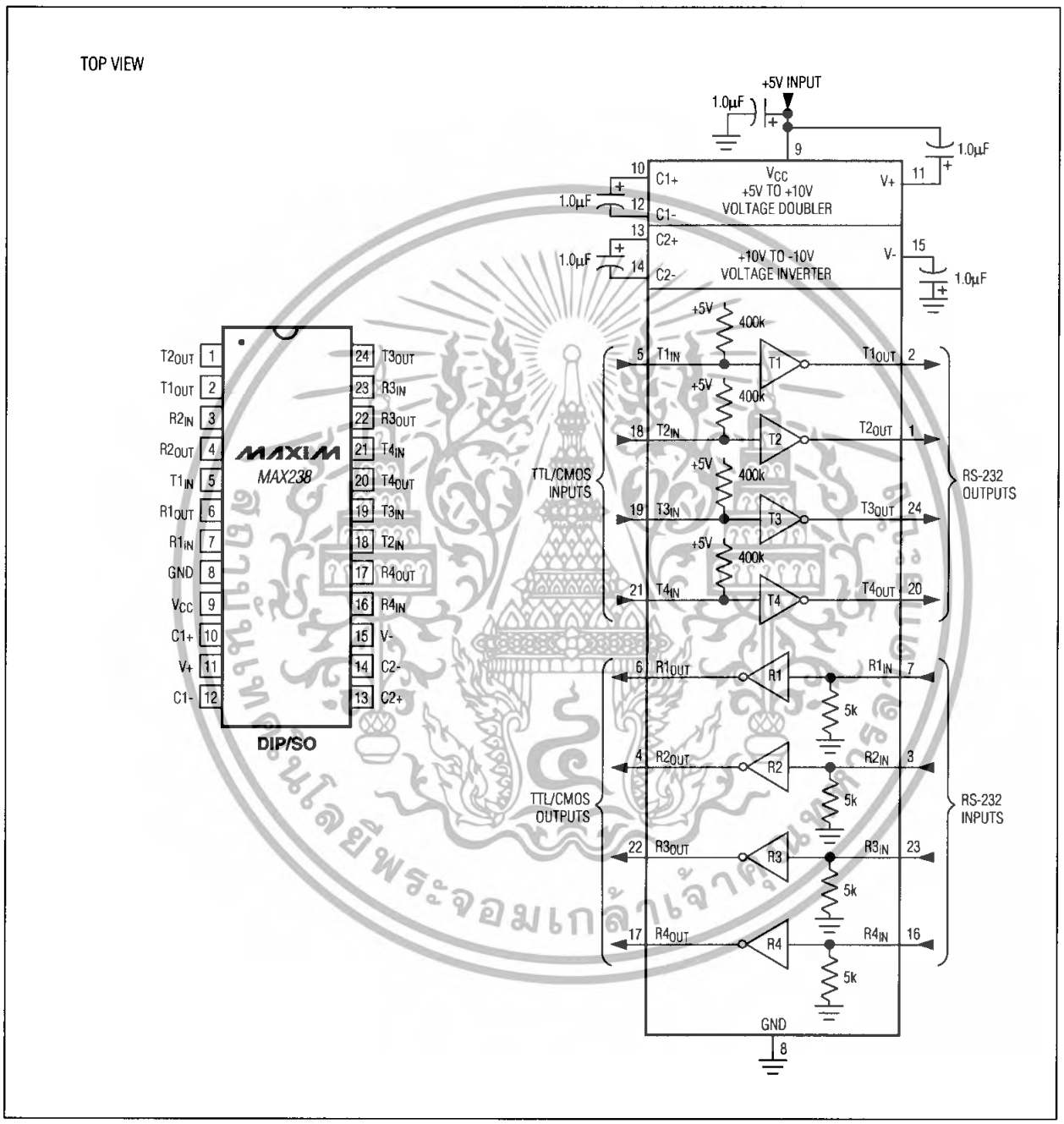


Figure 16. MAX238 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

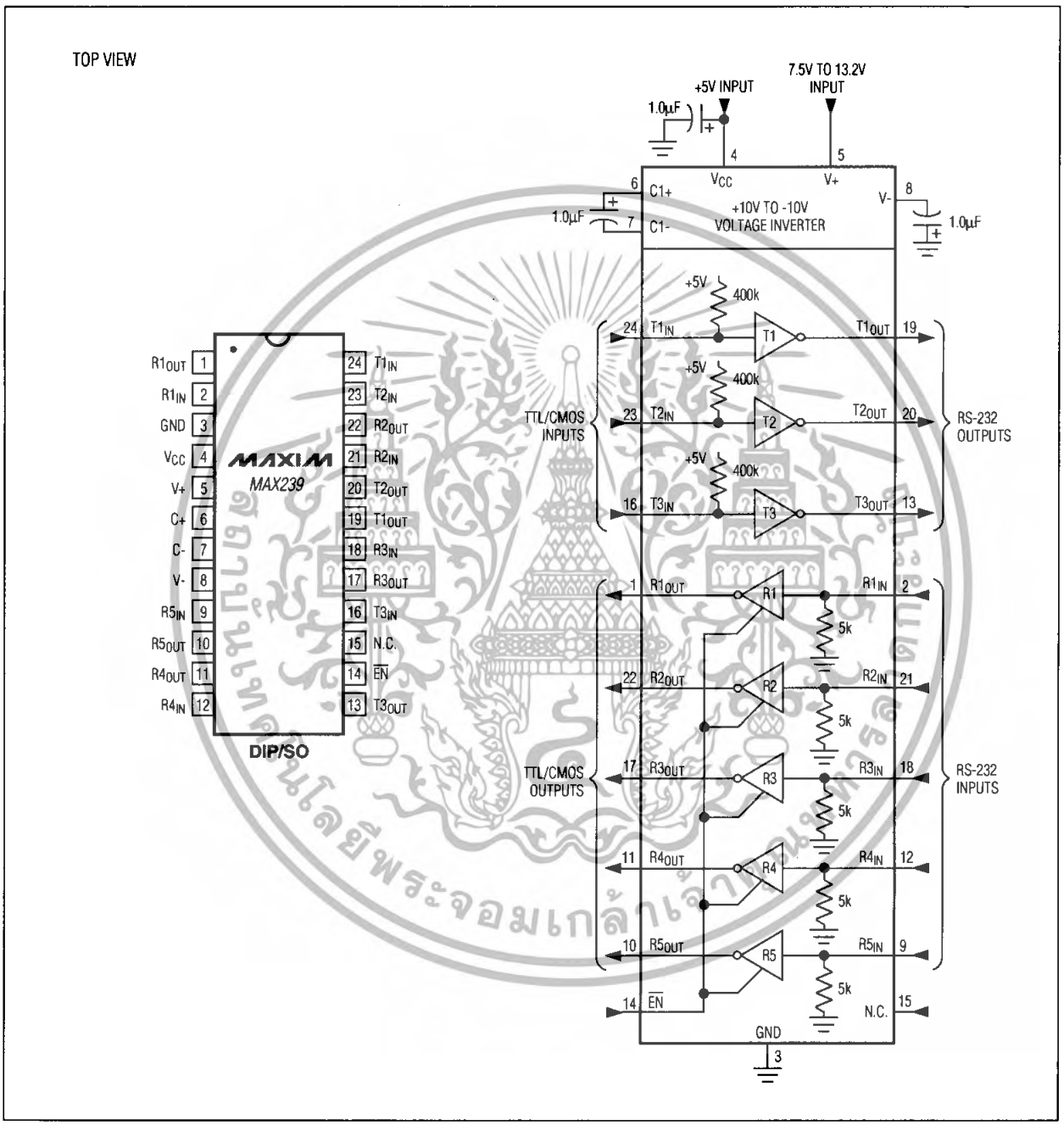


Figure 17. MAX239 Pin Configuration and Typical Operating Circuit

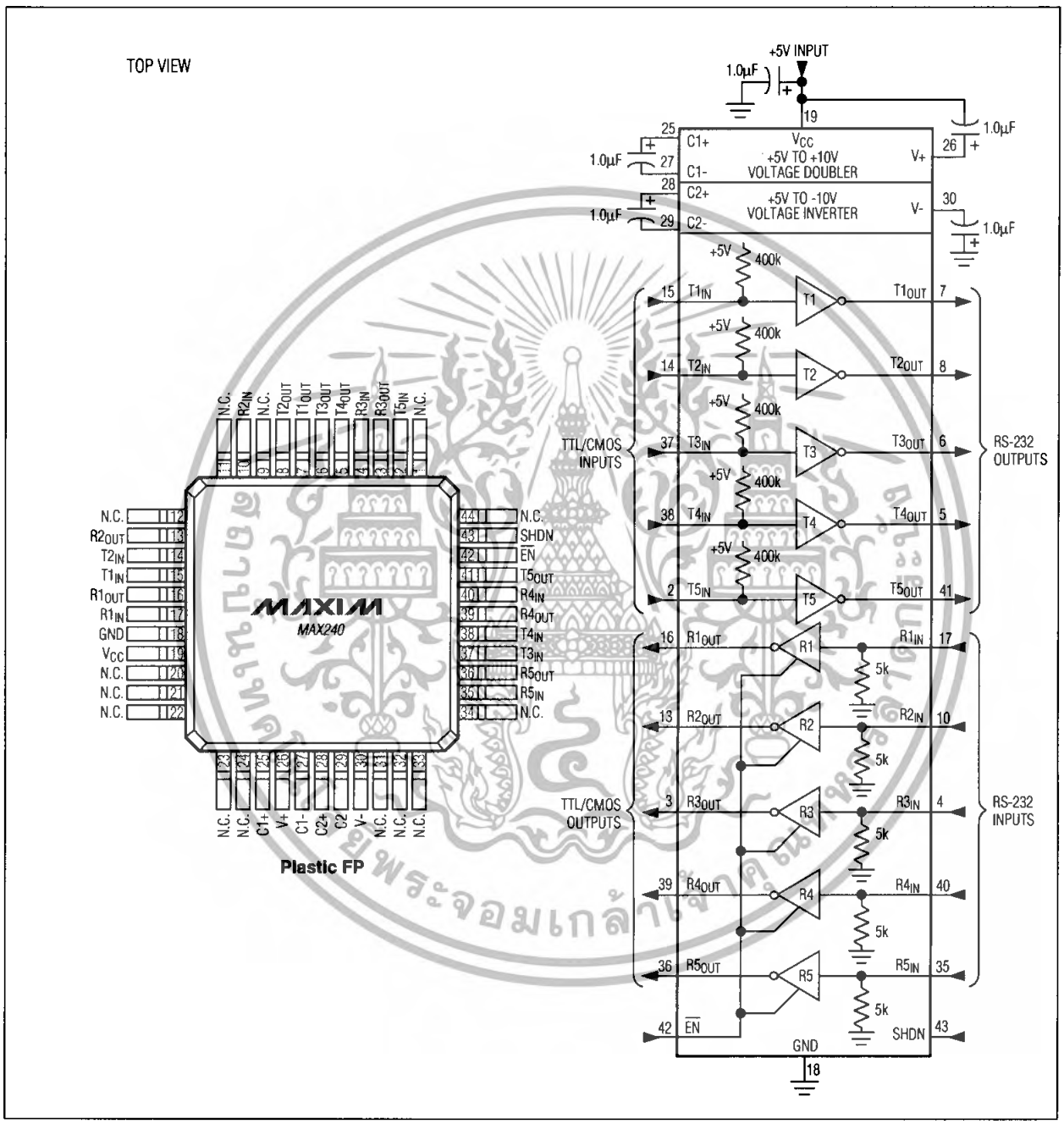


Figure 18. MAX240 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

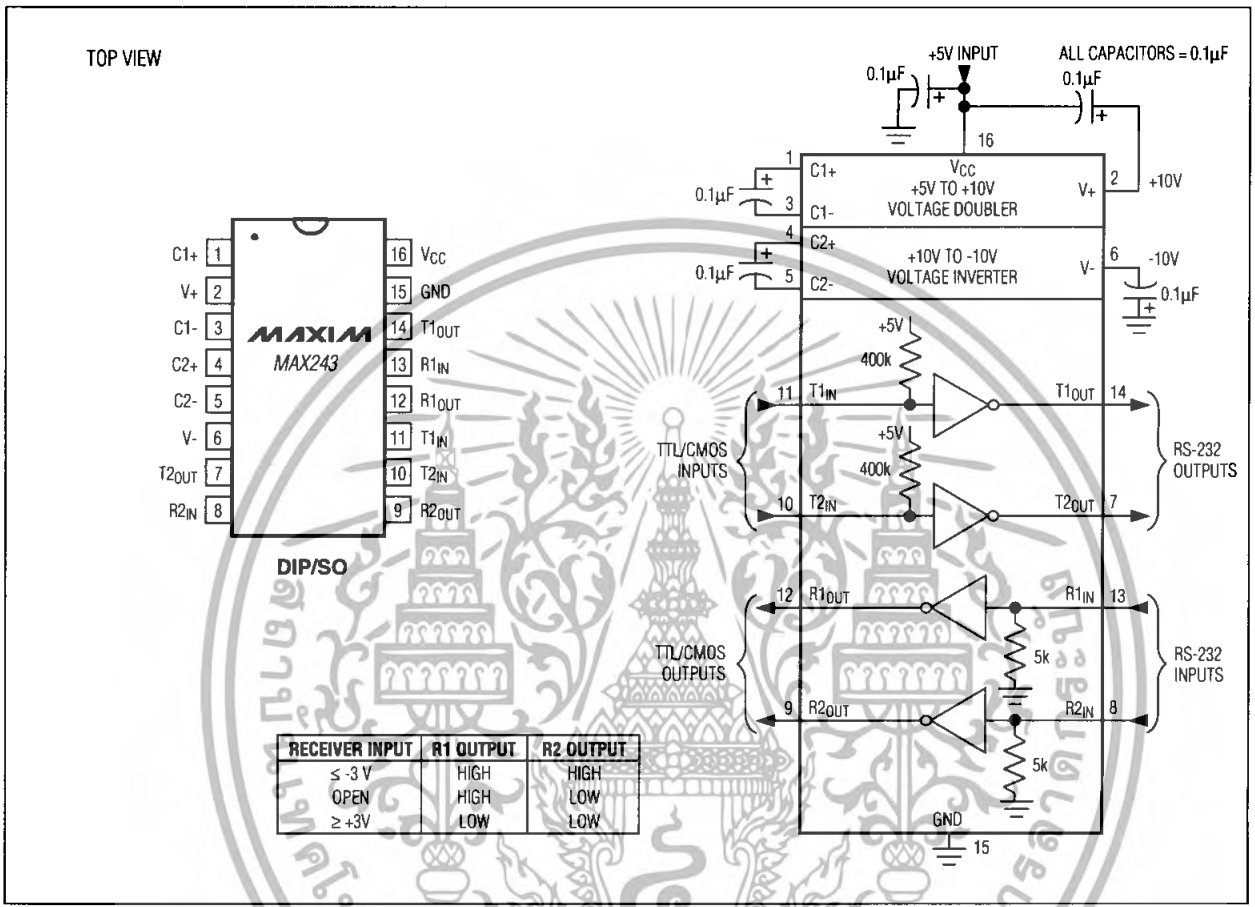


Figure 19. MAX243 Pin Configuration and Typical Operating Circuit

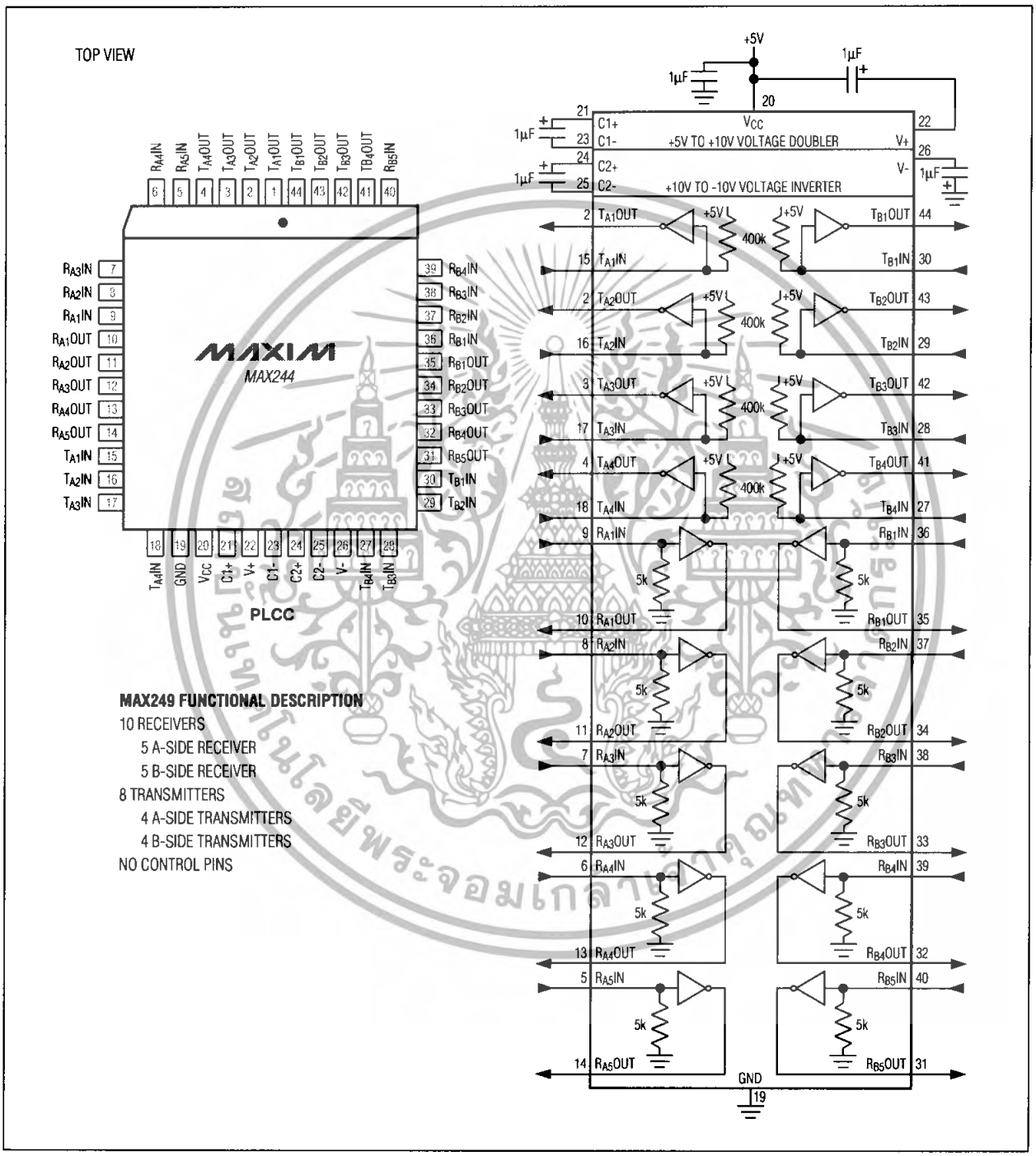
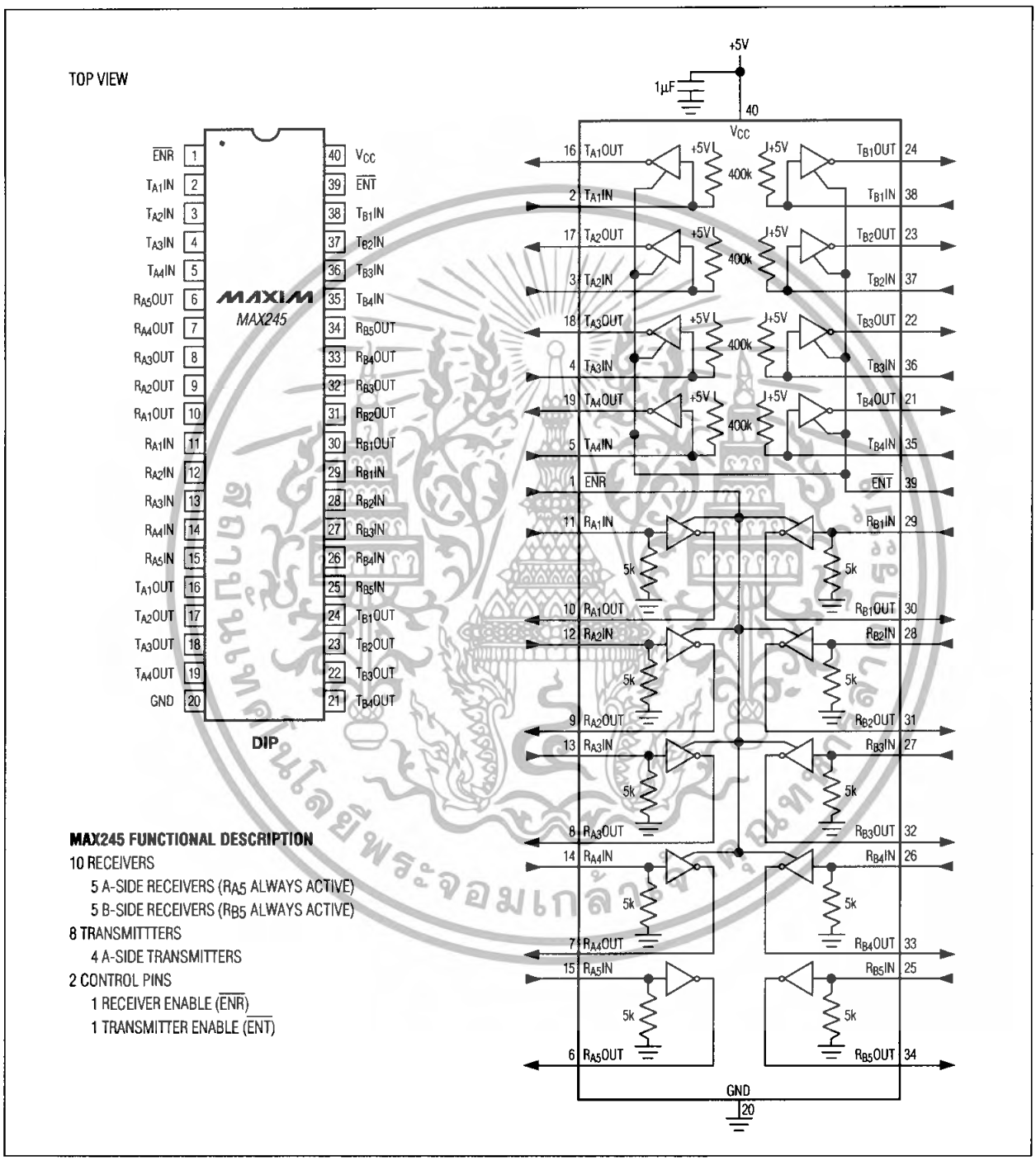


Figure 20. MAX244 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers



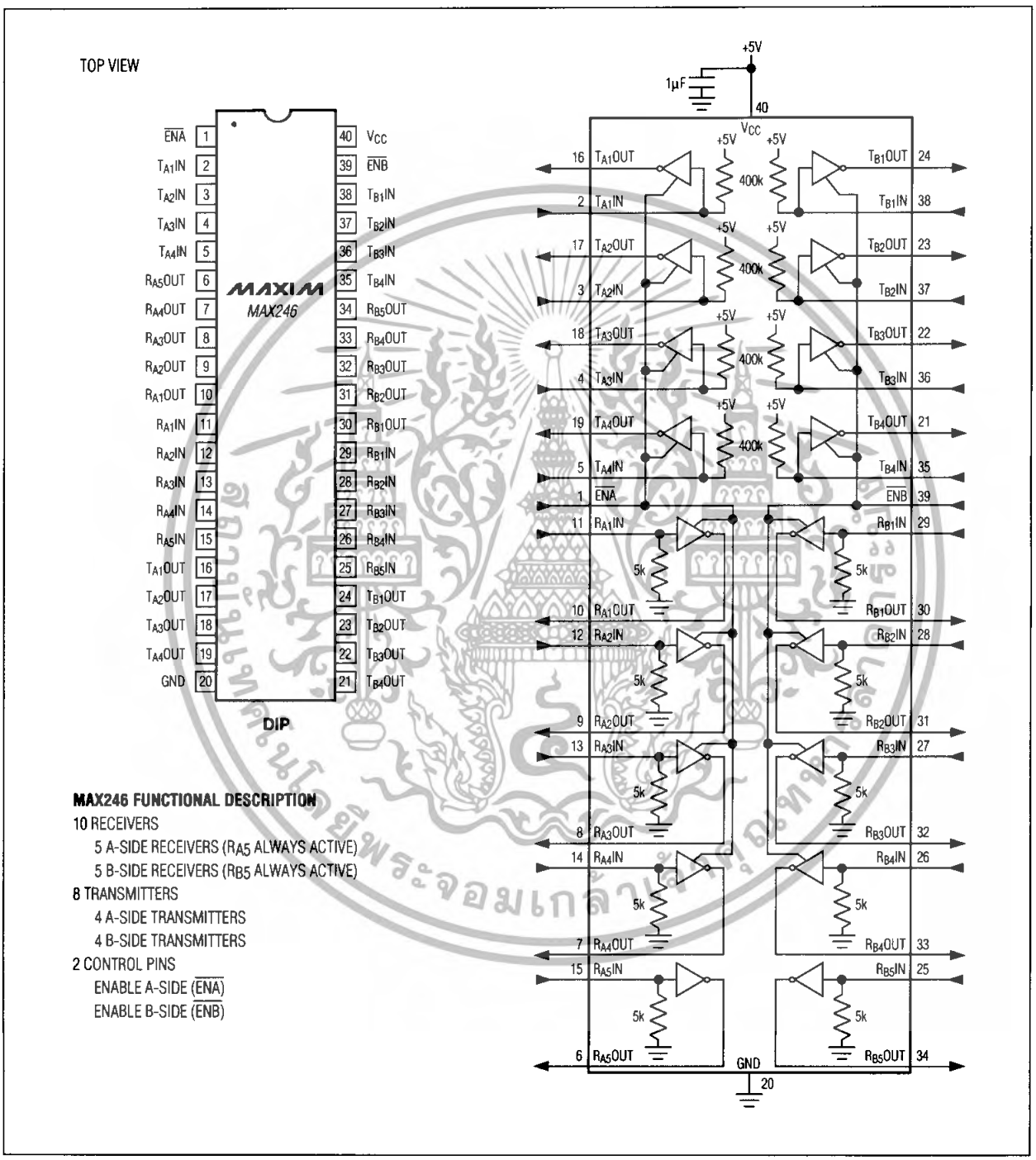
MAX245 FUNCTIONAL DESCRIPTION

10 RECEIVERS
- 5 A-SIDE RECEIVERS (R_{A5} ALWAYS ACTIVE)
- 5 B-SIDE RECEIVERS (R_{B5} ALWAYS ACTIVE)

8 TRANSMITTTERS
- 4 A-SIDE TRANSMITTERS

2 CONTROL PINS
- 1 RECEIVER ENABLE ($\overline{ENR}$)
- 1 TRANSMITTER ENABLE ($\overline{ENT}$)

Figure 21. MAX245 Pin Configuration and Typical Operating Circuit



MAX246 FUNCTIONAL DESCRIPTION

10 RECEIVERS
- 5 A-SIDE RECEIVERS (R_{A5} ALWAYS ACTIVE)
- 5 B-SIDE RECEIVERS (R_{B5} ALWAYS ACTIVE)

8 TRANSMITTERS
- 4 A-SIDE TRANSMITTERS
- 4 B-SIDE TRANSMITTERS

2 CONTROL PINS
- ENABLE A-SIDE ($\overline{ENA}$)
- ENABLE B-SIDE ($\overline{ENB}$)

Figure 22. MAX246 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

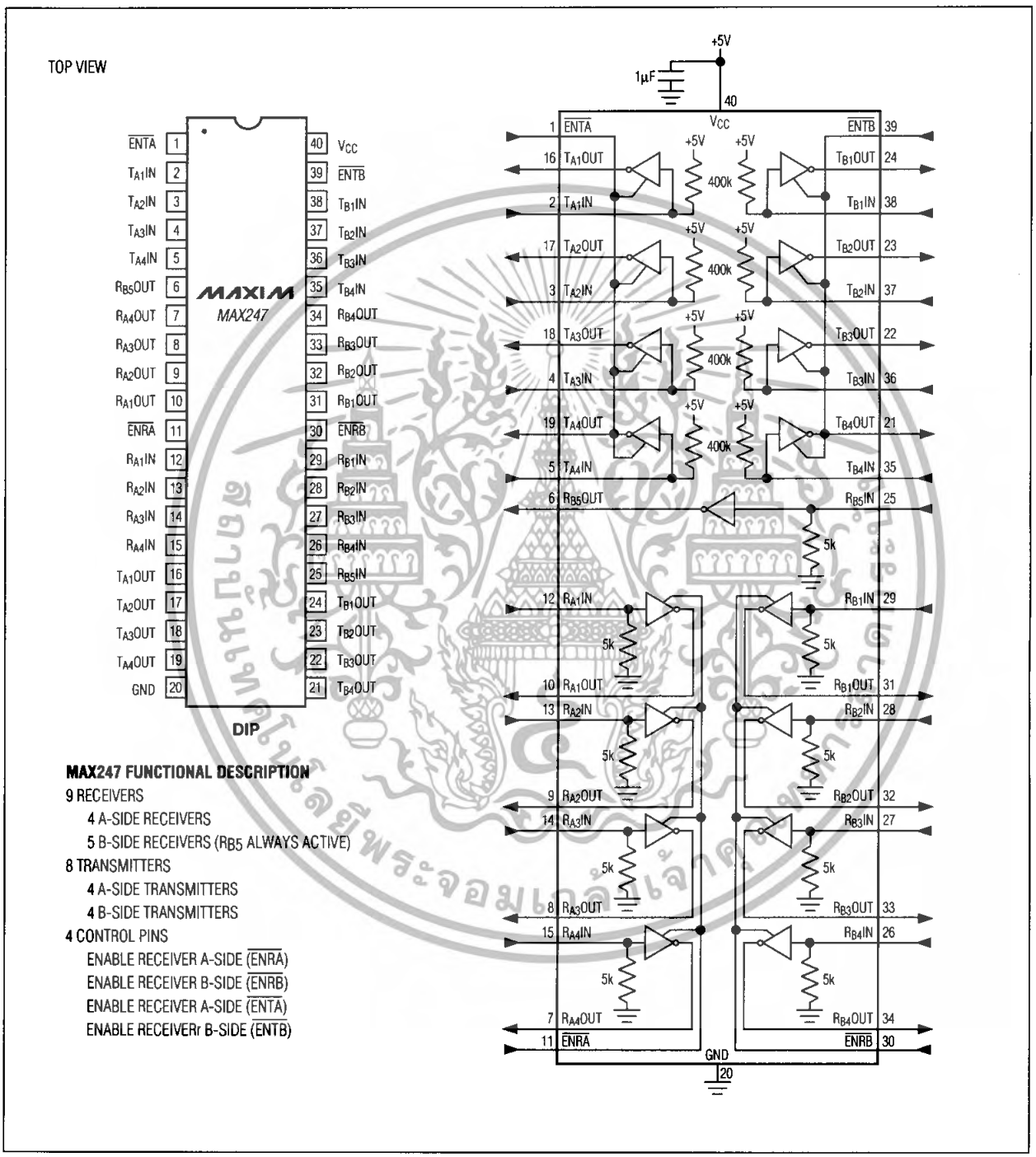


Figure 23. MAX247 Pin Configuration and Typical Operating Circuit

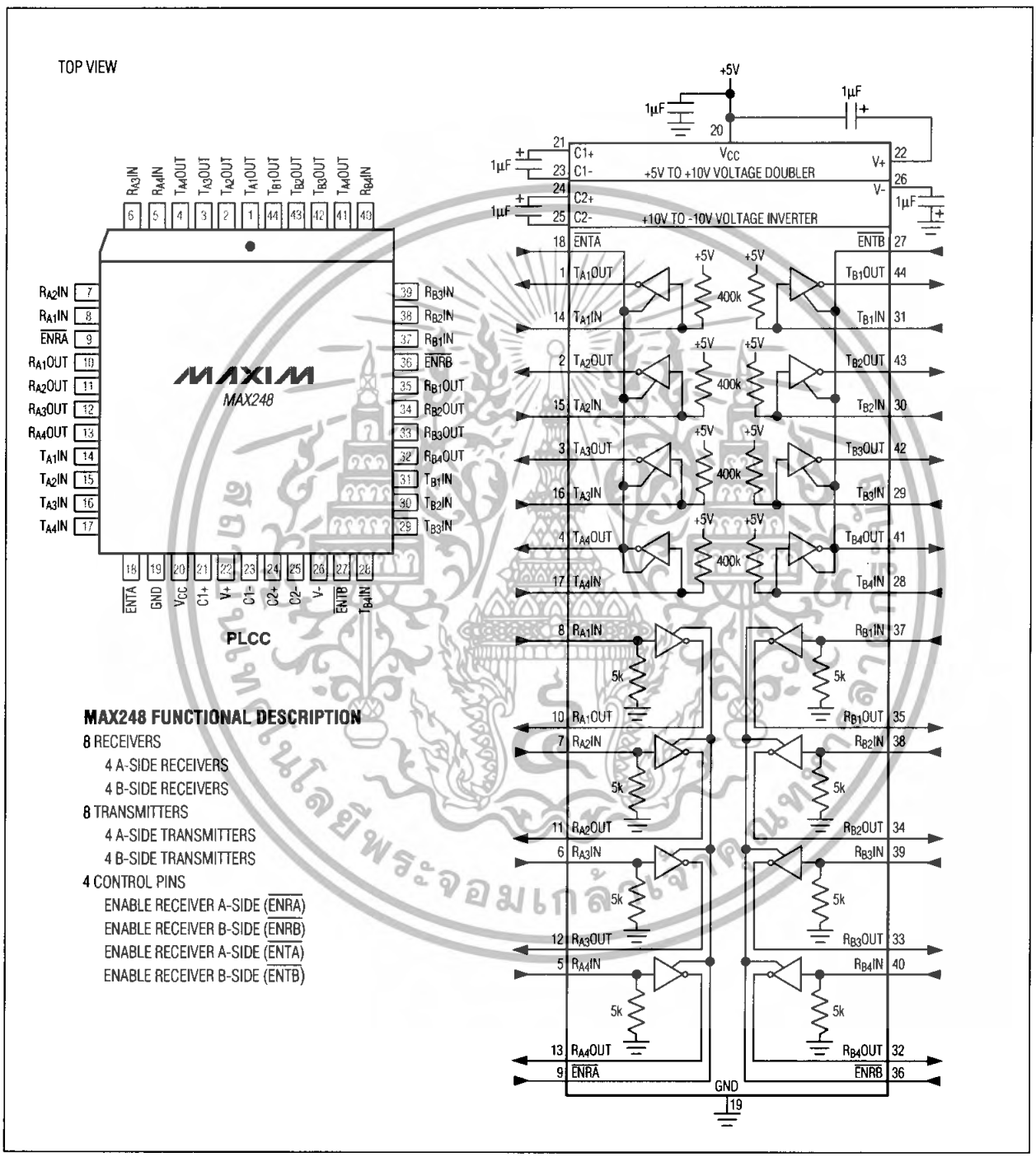


Figure 24. MAX248 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

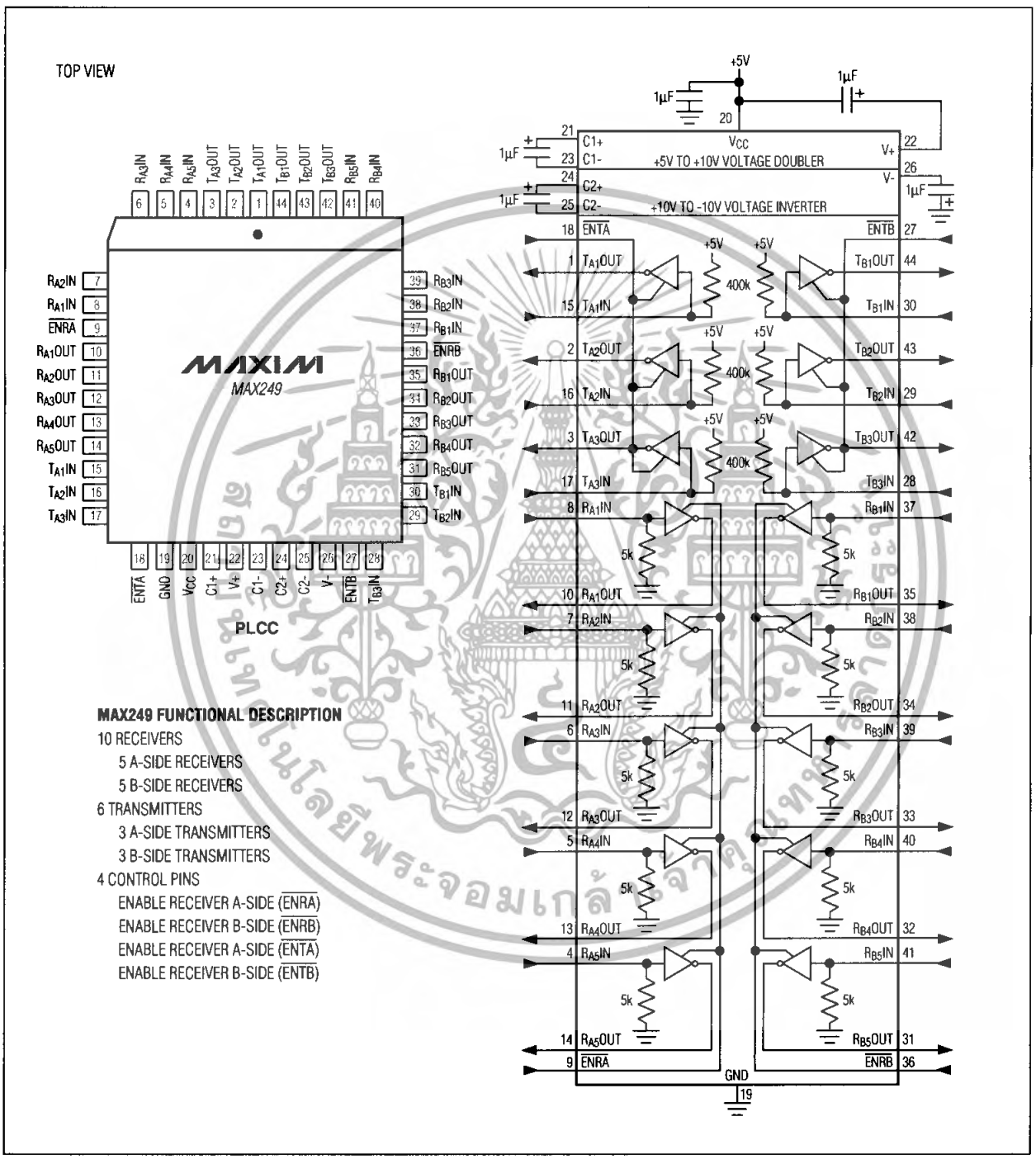


Figure 25. MAX249 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE
MAX222CPN	0°C to +70°C	18 Plastic DIP
MAX222CWN	0°C to +70°C	18 Wide SO
MAX222C/D	0°C to +70°C	Dice*
MAX222EPN	-40°C to +85°C	18 Plastic DIP
MAX222EWN	-40°C to +85°C	18 Wide SO
MAX222EJN	-40°C to +85°C	18 CERDIP
MAX222MJN	-55°C to +125°C	18 CERDIP
MAX223CAI	0°C to +70°C	28 SSOP
MAX223CWI	0°C to +70°C	28 Wide SO
MAX223C/D	0°C to +70°C	Dice*
MAX223EAI	-40°C to +85°C	28 SSOP
MAX223EWI	-40°C to +85°C	28 Wide SO
MAX225CWI	0°C to +70°C	28 Wide SO
MAX225EWI	-40°C to +85°C	28 Wide SO
MAX230CPP	0°C to +70°C	20 Plastic DIP
MAX230CWP	0°C to +70°C	20 Wide SO
MAX230C/D	0°C to +70°C	Dice*
MAX230EPP	-40°C to +85°C	20 Plastic DIP
MAX230EWP	-40°C to +85°C	20 Wide SO
MAX230EJP	-40°C to +85°C	20 CERDIP
MAX230MJP	-55°C to +125°C	20 CERDIP
MAX231CPD	0°C to +70°C	14 Plastic DIP
MAX231CWE	0°C to +70°C	16 Wide SO
MAX231CJD	0°C to +70°C	14 CERDIP
MAX231C/D	0°C to +70°C	Dice*
MAX231EPD	-40°C to +85°C	14 Plastic DIP
MAX231EWE	-40°C to +85°C	16 Wide SO
MAX231EJD	-40°C to +85°C	14 CERDIP
MAX231MJD	-55°C to +125°C	14 CERDIP
MAX232CPE	0°C to +70°C	16 Plastic DIP
MAX232CSE	0°C to +70°C	16 Narrow SO
MAX232CWE	0°C to +70°C	16 Wide SO
MAX232C/D	0°C to +70°C	Dice*
MAX232EPE	-40°C to +85°C	16 Plastic DIP
MAX232ESE	-40°C to +85°C	16 Narrow SO
MAX232EWE	-40°C to +85°C	16 Wide SO
MAX232EJE	-40°C to +85°C	16 CERDIP
MAX232MJE	-55°C to +125°C	16 CERDIP
MAX232MLP	-55°C to +125°C	20 LCC
MAX232ACPE	0°C to +70°C	16 Plastic DIP
MAX232ACSE	0°C to +70°C	16 Narrow SO
MAX232ACWE	0°C to +70°C	16 Wide SO
MAX232AC/D	0°C to +70°C	Dice*
MAX232AEPE	-40°C to +85°C	16 Plastic DIP
MAX232AESE	-40°C to +85°C	16 Narrow SO
MAX232AEWE	-40°C to +85°C	16 Wide SO
MAX232AEJE	-40°C to +85°C	16 CERDIP
MAX232AMJE	-55°C to +125°C	16 CERDIP
MAX232AMLP	-55°C to +125°C	20 LCC
MAX233CPP	0°C to +70°C	20 Plastic DIP
MAX233EPP	-40°C to +85°C	20 Plastic DIP
MAX233ACPP	0°C to +70°C	20 Plastic DIP
MAX233ACWP	0°C to +70°C	20 Wide SO
MAX233AEPP	-40°C to +85°C	20 Plastic DIP
MAX233AEWP	-40°C to +85°C	20 Wide SO
MAX234CPE	0°C to +70°C	16 Plastic DIP
MAX234CWE	0°C to +70°C	16 Wide SO
MAX234C/D	0°C to +70°C	Dice*
MAX234EPE	-40°C to +85°C	16 Plastic DIP
MAX234EWE	-40°C to +85°C	16 Wide SO
MAX234EJE	-40°C to +85°C	16 CERDIP
MAX234MJE	-55°C to +125°C	16 CERDIP
MAX235CPG	0°C to +70°C	24 Wide Plastic DIP
MAX235EPG	-40°C to +85°C	24 Wide Plastic DIP
MAX235EDG	-40°C to +85°C	24 Ceramic SB
MAX235MDG	-55°C to +125°C	24 Ceramic SB
MAX236CNG	0°C to +70°C	24 Narrow Plastic DIP
MAX236CWG	0°C to +70°C	24 Wide SO
MAX236C/D	0°C to +70°C	Dice*
MAX236ENG	-40°C to +85°C	24 Narrow Plastic DIP
MAX236EWG	-40°C to +85°C	24 Wide SO
MAX236ERG	-40°C to +85°C	24 Narrow CERDIP
MAX236MRG	-55°C to +125°C	24 Narrow CERDIP
MAX237CNG	0°C to +70°C	24 Narrow Plastic DIP
MAX237CWG	0°C to +70°C	24 Wide SO
MAX237C/D	0°C to +70°C	Dice*
MAX237ENG	-40°C to +85°C	24 Narrow Plastic DIP
MAX237EWG	-40°C to +85°C	24 Wide SO
MAX237ERG	-40°C to +85°C	24 Narrow CERDIP
MAX237MRG	-55°C to +125°C	24 Narrow CERDIP
MAX238CNG	0°C to +70°C	24 Narrow Plastic DIP
MAX238CWG	0°C to +70°C	24 Wide SO
MAX238C/D	0°C to +70°C	Dice*
MAX238ENG	-40°C to +85°C	24 Narrow Plastic DIP

** Contact factory for dice specifications.*

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE
MAX238EWG	-40°C to +85°C	24 Wide SO
MAX238ERG	-40°C to +85°C	24 Narrow CERDIP
MAX238MRG	-55°C to +125°C	24 Narrow CERDIP
MAX239CNG	0°C to +70°C	24 Narrow Plastic DIP
MAX239CWG	0°C to +70°C	24 Wide SO
MAX239C/D	0°C to +70°C	Dice*
MAX239ENG	-40°C to +85°C	24 Narrow Plastic DIP
MAX239EWG	-40°C to +85°C	24 Wide SO
MAX239ERG	-40°C to +85°C	24 Narrow CERDIP
MAX239MRG	-55°C to +125°C	24 Narrow CERDIP
MAX240CMH	0°C to +70°C	44 Plastic FP
MAX240C/D	0°C to +70°C	Dice*
MAX241CAI	0°C to +70°C	28 SSOP
MAX241CWI	0°C to +70°C	28 Wide SO
MAX241C/D	0°C to +70°C	Dice*
MAX241EAI	-40°C to +85°C	28 SSOP
MAX241EWI	-40°C to +85°C	28 Wide SO
MAX242CAP	0°C to +70°C	20 SSOP
MAX242CPN	0°C to +70°C	18 Plastic DIP
MAX242CWN	0°C to +70°C	18 Wide SO
MAX242C/D	0°C to +70°C	Dice*
MAX242EPN	-40°C to +85°C	18 Plastic DIP
MAX242EWN	-40°C to +85°C	18 Wide SO
MAX242EJN	-40°C to +85°C	18 CERDIP
MAX242MJN	-55°C to +125°C	18 CERDIP

MAX243CPE	0°C to +70°C	16 Plastic DIP
MAX243CSE	0°C to +70°C	16 Narrow SO
MAX243CWE	0°C to +70°C	16 Wide SO
MAX243C/D	0°C to +70°C	Dice*
MAX243EPE	-40°C to +85°C	16 Plastic DIP
MAX243ESE	-40°C to +85°C	16 Narrow SO
MAX243EWE	-40°C to +85°C	16 Wide SO
MAX243EJE	-40°C to +85°C	16 CERDIP
MAX243MJE	-55°C to +125°C	16 CERDIP
MAX244CQH	0°C to +70°C	44 PLCC
MAX244C/D	0°C to +70°C	Dice*
MAX244EQH	-40°C to +85°C	44 PLCC
MAX245CPL	0°C to +70°C	40 Plastic DIP
MAX245C/D	0°C to +70°C	Dice*
MAX245EPL	-40°C to +85°C	40 Plastic DIP
MAX246CPL	0°C to +70°C	40 Plastic DIP
MAX246C/D	0°C to +70°C	Dice*
MAX246EPL	-40°C to +85°C	40 Plastic DIP
MAX247CPL	0°C to +70°C	40 Plastic DIP
MAX247C/D	0°C to +70°C	Dice*
MAX247EPL	-40°C to +85°C	40 Plastic DIP
MAX248CQH	0°C to +70°C	44 PLCC
MAX248C/D	0°C to +70°C	Dice*
MAX248EQH	-40°C to +85°C	44 PLCC
MAX249CQH	0°C to +70°C	44 PLCC
MAX249EQH	-40°C to +85°C	44 PLCC

** Contact factory for dice specifications.*