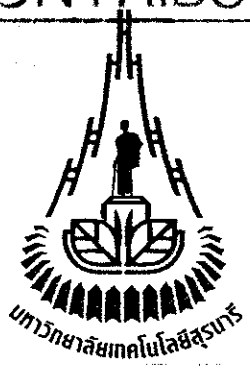
ระบบแสดงสัญญาณไฟจราจรอัตโนมัติ

นางสาวจันทร์วิภา	จูนละวงศ์	B4409827
นางสาวกันติดา	ไชยหนึ่งนิด	B4402893
นางสาวพัชณา	แสงขอน	B4404125

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2547



ระบบแสดงสัญญาณไฟจราจรอัตโนมัติ

นางสาวจันทร์จริยา	ลุนละวงค์	B4400837
นางสาวนันทิดา	ใจคำนึงนิด	B4402893
นางสาวพัฒนา	แสงซอน	B4404125

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ปีการศึกษา 2547

โครงงาน	ระบบแสดงสัญญาณไฟจราจรอัตโนมัติ
ผู้ดำเนินงาน	1. นางสาวจันทร์จริยา ลุนละวงค์
	2. นางสาวนันทิดา ใจคำนึงนิต
	3. นางสาวพัฒนา แสงซอน
อาจารย์ที่ปรึกษา	ดร.ชาญชัย ทองโสภา
สาขาวิชา	วิศวกรรมโทรคมนาคม
ภาคการศึกษาที่	2/2547

บทคัดย่อ

ระบบแสดงสัญญาณไฟจราจรอัตโนมัติด้วยไมโครคอนโทรเลอร์PIC16F84Aประกอบด้วยส่วนที่สำคัญ3ส่วนคือภาคตรวจรับสัญญาณจากไฟจราจรในส่วนนี้จะทำหน้าที่ในการรับสัญญาณจากไฟจราจร แล้วทำการส่งสัญญาณที่ได้นี้ให้กับตัวไมโครคอนโทรลเลอร์ PIC16F84A ในภาคควบคุมและประมวลผลเพื่อทำการประมวลผลต่อไปภาคของส่วนควบคุมในส่วนนี้จะใช้ไมโครคอนโทรลเลอร์ PIC16F84A ในการประมวลผลสัญญาณที่รับได้จากส่วนรับสัญญาณ และส่งผลที่ได้นั้นไปแสดงผลของตัวเลขในหลอดLEDโดยส่วนของตัวควบคุมการแสดงผลนี้จะทำงานในลักษณะของการวนรอบแล้วตรวจสอบค่าระหว่างสัญญาณ ที่รับได้จากการจับกระแสในสัญญาณไฟจราจรเพื่อเปรียบเทียบว่าผลที่ได้ตรงกันกับที่ผ่านมาหรือไม่หากว่าเมื่อใดที่ผลที่รับได้ไม่ตรงกันตัวควบคุมจะสั่งการให้การแสดงสัญญาณหยุดการทำงานโดยทันทีและเริ่มการประมวลผลใหม่อีกครั้ง และภาคที่ใช้ในการแสดงผลตัวเลขการนับเวลาโดยจะใช้หลอด LEDชนิดเลเซอร์ สีเขียวและสีแดงสลับกันตามสีที่ปรากฏ รวมถึงการขับกระแสให้หลอด LED ทุกหลอดสามารถติดพร้อมกันได้

กิตติกรรมประกาศ

ในการจัดทำโครงงานระบบแสดงสัญญาณไฟจราจรอัตโนมัติในครั้งนี้ สามารถเสร็จสมบูรณ์ ลงได้ก็เพราะด้วยความกรุณาของบุคคลหลายท่าน ซึ่งคอยให้ความช่วยเหลือและคำปรึกษา รวมทั้งข้อเสนอแนะที่เป็นประโยชน์ในการทำโครงงานครั้งนี้ซึ่งประกอบด้วยอาจารย์ ดร.ชาญชัย ทองโสภา อาจารย์ที่ปรึกษาโครงงานผู้เปิดโอกาสให้ผู้จัดทำได้สัมผัสและรู้จักกับการทำโครงงานนี้ เป็นผู้ประสิทธิ์ประสาทวิชาความรู้รวมทั้งให้คำแนะนำอันเป็นประโยชน์อย่างยิ่งเกี่ยวกับโครงงาน ผศ.ดร.รังสรรค์ วงศ์สรรค์ หัวหน้าสาขาวิชาวิศวกรรมโทรคมนาคม มหาวิทยาลัยเทคโนโลยีสุรนารี ผู้ซึ่งมีความเมตตาคอยให้ความช่วยเหลือในทุกเรื่อง รวมทั้งคอยให้คำแนะนำปรึกษาที่ดี

ขอขอบคุณคุณประพล จาระตะคุที่ให้ความช่วยเหลือในการดำเนินการเกี่ยวกับงบประมาณที่ใช้ในการดำเนินโครงการ คุณวิชัย ศรีสุรักษ์ และ คุณสมิง เติมพรหมราชที่ช่วยให้คำปรึกษาเกี่ยวกับวงจรอิเล็กทรอนิกส์ และ การเขียนโปรแกรมเจ้าหน้าที่ห้องปฏิบัติการวิศวกรรมโทรคมนาคมห้องปฏิบัติการไมโครโปรเซสเซอร์ห้องปฏิบัติการวงจรและอุปกรณ์ทุกท่านที่ให้ความช่วยเหลือเกี่ยวกับอุปกรณ์และเครื่องมือต่างๆ รวมทั้งเพื่อนๆทุกคนสำหรับความช่วยเหลือที่ดีและกำลังใจที่มอบให้ตลอดมาและที่สำคัญยิ่งผู้จัดทำขอกราบขอบพระคุณบิดาและมารดาของผู้จัดทำผู้ให้โอกาสทางการศึกษา และ คอยสนับสนุนด้วยดีตลอดมา รวมทั้งกำลังใจที่คอยมอบให้ผู้จัด ทำอย่างหาที่เปรียบมิได้

สุดท้ายนี้คุณงามความดีทั้งหลายที่ได้จากการทำโครงงานในครั้งนี้ ผู้จัดทำขออุทิศให้แด่นางสาวพัฒนา แสงซอน เพื่อนผู้ล่วงลับไปอย่างกะทันหัน ในขณะที่ได้ร่วมทำโครงงานชิ้นนี้สำเร็จแล้ว

นางสาวจันทร์จริยา ลุนละวงค์
นางสาวนันทิดา ใจคำนึงนิด
นางสาวพัฒนา แสงซอน

สารบัญ

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ(ต่อ)

หน้า

สารบัญภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ปัญหาที่มาของโครงงาน

ระบบจราจรในตัวเมืองใหญ่ๆ จะมีความหนาแน่นมาก โดยฉะเพราะตามบริเวณถนนสายหลักที่มีทางแยกต่างๆ จะมีการจราจรที่ติดขัด และเกิดอุบัติเหตุได้บ่อย ปัญหาจราจร ในช่วงที่ต้องหยุดรอสัญญาณไฟจราจรได้สร้างความทุกข์ยากให้กับประชาชนมานาน ทำลายเศรษฐกิจ เผาน้ำมันทิ้งเกินกว่าจำเป็น ทั้งทำลายสังคม ทำให้เกิดความเครียด ทำลายสุขภาพ เพราะอากาศเป็นพิษ และทำลายสิ่งแวดล้อม คิดเป็นความสูญเสียนับแสนล้านต่อปี และยังส่งผลกระทบต่อชีวิตของประชาชน

ในขณะนี้ทางรัฐบาลได้มีการติดตั้งสัญญาณไฟจราจรที่เป็น LED ควบคู่กับสัญญาณไฟจราจรปกติ การที่จะติดตั้งสัญญาณไฟจราจรที่เป็นแบบ LED ไม่ได้ช่วยให้การจราจรที่ติดขัด มีความคร่องตัวขึ้นแต่อย่างไร แต่เพื่อให้คนสามารถรู้ได้ว่าตนเองต้องรออีกนานเท่าไร หรือเหลือเวลาอีกเท่าไรที่จะต้องหยุดรถ ซึ่งจะทำให้คนที่ใช้ถนนมีการเตรียมพร้อม แต่ในการติดตั้งสัญญาณไฟจราจรแบบ LED ในขณะนี้มีราคาสูงมาก จึงทำให้เกิดความคิดที่จะทดลองสร้างเครื่องบอกเวลาสัญญาณไฟจราจรโดยใช้ความรู้ที่ได้เรียนมาเพื่อประยุกต์ใช้ในการทำงาน โครงงานชิ้นนี้มีหลักการทำงานเช่นเดียวกันกับอุปกรณ์บอกเวลาที่มีขายทั่วไป เหตุที่เราต้องการทำโครงงานชิ้นนี้เพียงเพื่อต้องการศึกษาและนำนำความรู้ไปปฏิบัติจริงว่าสิ่งที่เราได้เรียนมาจะสามารถนำไปใช้งานได้จริงหรือไม่

ระบบแสดงสัญญาณไฟจราจรอัตโนมัติได้แบ่งออกเป็น 3 ส่วนคือ ส่วนแรก คือส่วนของการตรวจรับสัญญาณ จากไฟจราจร ส่วนที่สอง คือส่วนของการควบคุมประมวลผล การตรวจจับเวลาที่สัญญาณไฟเขียวและสัญญาณไฟแดงที่แสดงในแต่ละครั้งแล้วนำไปเก็บไว้ในหน่วยความจำ และส่วนสุดท้ายคือ เป็นภาคแสดงผลโดยใช้เป็นแผงหลอด LED

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 ศึกษาลักษณะและการทำงานของไมโครคอนโทรลเลอร์ PIC16F84A

1.2.2 ศึกษาลักษณะและการทำงานของสัญญาณไฟจราจร

1.2.3 ศึกษาและออกแบบวงจร อุปกรณ์บอกเวลาสัญญาณไฟจราจร

1.2.4 ศึกษาการใช้งานของหลอด LED และการขยายพอร์ต

1.2.5 ออกแบบวงจรและ Print Circuit Board(PCB)ด้วยโปรแกรม Protel 99SE

1.2.6 เขียนโปรแกรมภาษาแอสเซมบลี้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ PIC16F84A

1.2.7 สร้างอุปกรณ์บอกสัญญาณไฟจราจร และสามารถนำไปใช้งานได้จริง

1.3 ขอบข่ายของโครงงาน

1.3.1 เครื่องสามารถที่จะกำหนดการแสดงเวลาของสัญญาณไฟเขียวและไฟแดงโดยเวลาที่แสดงจะต้องไม่เกิน 999 วินาที และไม่ต่ำกว่า 000 วินาที

1.3.2 เครื่องจะสามารถหยุดการทำงานได้เองเมื่อสีของสัญญาณไฟจราจรและสีของตัวแสดงเวลาไม่ตรงกัน

1.3.3 เครื่องจะทำงานเองโดยอัตโนมัติ ไม่มีการกำหนดค่าเริ่มต้นเครื่องจะทำการจับสีของสัญญาณไฟจราจรก่อน 1 รอบจากนั้นเครื่องจึงจะเริ่มทำงาน

1.3.4 เครื่องนี้จะสามารถนำไปใช้ในจุดที่ไม่มีการแสดงสัญญาณไฟเลี้ยว

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 ได้เรียนรู้หลักการทำงานและโครงสร้าง ไมโครคอนโทรลเลอร์ PIC16F84A

1.4.2 ได้เรียนรู้หลักการทำงานและโครงสร้างของ สัญญาณไฟจราจร

1.4.3 สามารถใช้โปรแกรม Protel 99SE ในการออกแบบวงจรและลายวงจรได้

1.4.4 สามารถเขียนโปรแกรมภาษา Assembly ควบคุมการทำงานของไมโครคอนโทรลเลอร์ได้

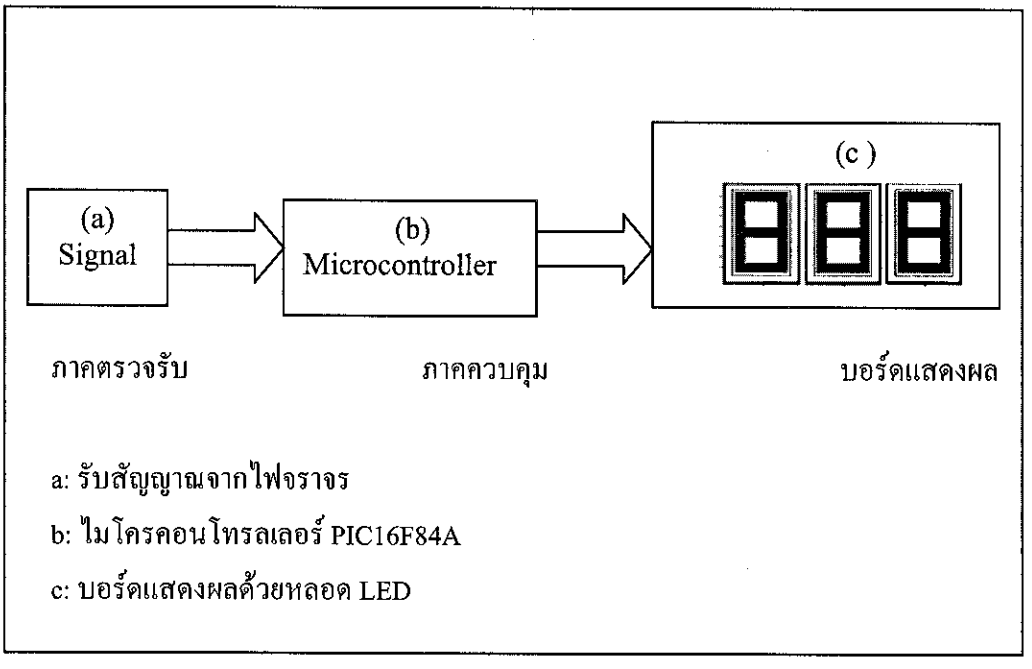
1.4.5 ได้เรียนรู้วิธีการขยายพอร์ต และการใช้งานของหลอด LED รวมถึงการขยายกระแส

1.4.6 สามารถที่จะนำอุปกรณ์บอกเวลาสัญญาณไฟจราจรไปใช้งานได้จริง โดยไม่เกิดปัญหา

บทที่ 2
การออกแบบระบบ

2.1 กล่าวนำ

ระบบโดยรวมมีองค์ประกอบสำคัญ 3 ส่วน คือ ภาคที่เป็นโหลดทางไฟสลับเพื่อตรวจรับสัญญาณเพื่อส่งให้กับไมโครคอนโทรลเลอร์ ภาคประมวลผลด้วยไมโครคอนโทรเลอร์PIC16F84A และภาคแสดงผลที่บอร์ด LED



ภาพที่ 2.1 โครงสร้างโดยรวมของระบบ

ภาคตรวจรับสัญญาณจะทำหน้าที่รับสัญญาณจากไฟจราจร ซึ่งเป็นโหลดที่มีแรงดันและกระแสไฟฟ้าสูงมาก เราจึงต้องใช้การขับโหลดโดยใช้ออปโต้คัปเปลอร์เพื่อให้เป็นโหลดทางไฟสลับ โดยใช้ ออปโต้ เบอร์ 4N35 ซึ่ง เป็นแบบเอาต์พุตเป็นโฟโต้ทรานซิสเตอร์ แล้วทำการส่งสัญญาณที่ได้นี้ไปให้ไมโครคอนโทรลเลอร์ PIC16F84A เพื่อทำการประมวลผลในลำดับต่อไป

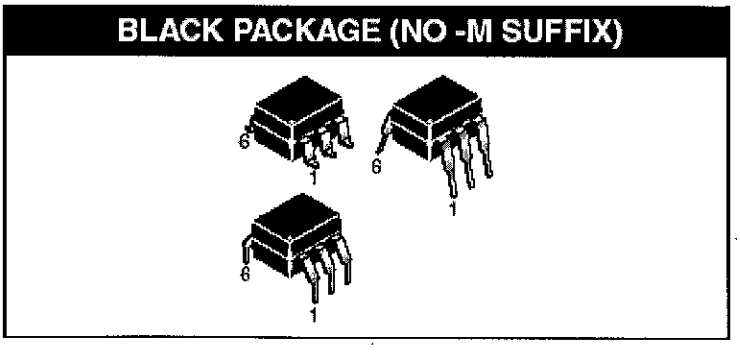
ภาคควบคุม มีส่วนสำคัญที่เป็นหลักใหญ่ๆ คือ ไมโครคอนโทรเลอร์ PIC16F84A ทำหน้าที่รับสัญญาณจากไฟจราจรมาเก็บไว้ในหน่วยความจำแล้วทำการประมวลผลและส่งให้บอร์ด LEDแสดงผล

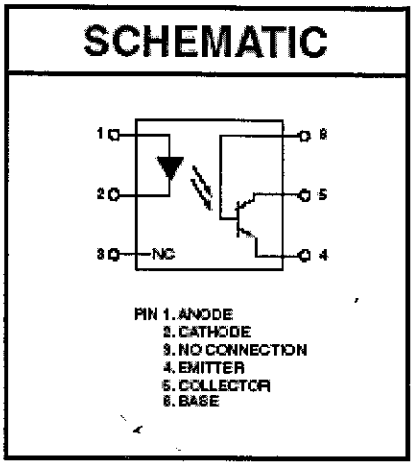
ในตัวของไมโครคอนโทรเลอร์ PIC16F84A จะทำหน้าที่รับสัญญาณไฟจราจร มาเก็บไว้ในหน่วยความจำตลอดเวลา คือ ครั้งแรกตัวไมโครคอนโทรเลอร์ PIC16F84A จะตรวจวัดเวลาของสัญญาณไฟแดง มาเก็บไว้ในหน่วยความจำ ครั้งที่สองที่ตรวจพบสัญญาณไฟแดง จะทำการตรวจวัดเวลาแล้วทำการเปรียบเทียบว่าเวลาที่ได้ตรงกันหรือไม่ ถ้าไม่ตรงกัน ให้เริ่มมีการตรวจวัดเวลาของสัญญาณใหม่ หากเวลาที่ได้ตรงกัน ก็ให้ไปแสดงผลที่บอร์ด LED ทำการตรวจวัดสัญญาณไฟเขียวเช่นเดียวกับสัญญาณไฟแดง ในขณะที่มีการแสดงผล ตัวไมโครคอนโทรเลอร์ก็ยังมีการตรวจสอบสัญญาณไฟจราจรอยู่ตลอดเวลา

สำหรับการออกแบบบอร์ดแสดงผล ได้นำเอาหลอด LED แบบเลเซอร์มาต่อเป็นเป็น 7-Secmen โดยใช้หลักต่อแบบ คอมมอนแคโทด ในแต่ละsecmenจะรับคำสั่งจากไมโครคอนโทรเลอร์ PIC16F84A โดยผ่าน ทรานสิเตอร์ MJE305 ใน 7-Secmen ได้ทำทั้งหมด 3 ชุด ในแต่ละชุดจะมีทรานสิเตอร์ 2N3055 ซึ่งทำหน้าที่ขยายกระแสให้กับวงจร7-Secmen

2.2 ส่วนที่ทำหน้าที่รับสัญญาณไฟจราจร

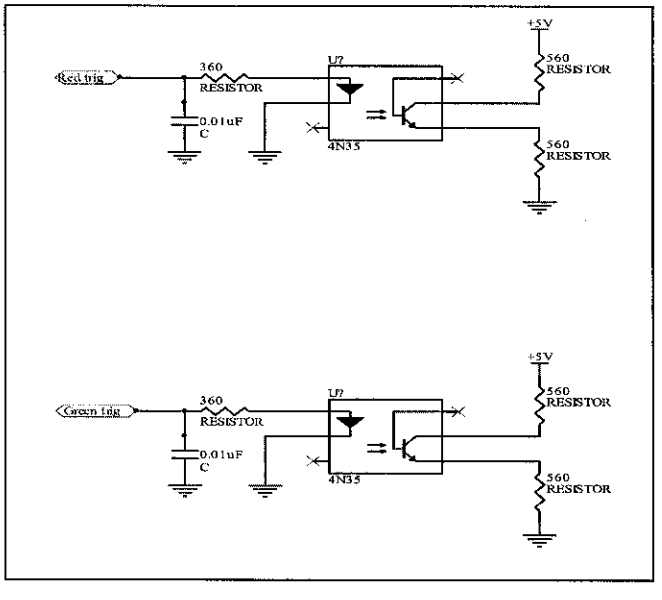
เนื่องจากส่วนนี้จะทำหน้าที่ในการรับสัญญาณ ที่มีแรงดันและกระแสสูง เราจึงต้องใช้วิธีการขับโหลดโดยใช้ ออปโต้คัปเปอร์ ที่เป็นโหลดทางไฟสลับมาใช้เพื่อรับสัญญาณที่ได้ เพื่อเป็นการป้องกันความเสียหาย ที่อาจเกิดจากการเปลี่ยนแปลงระดับของแรงดัน และกระแสไฟฟ้าอย่างมากมาย และ รวดเร็วแล้วจึงส่งสัญญาณที่ได้นั้น ให้กับไมโครคอนโทรลเลอร์เพื่อนำค่าที่ได้ไปประมวลผลในลำดับต่อไป รายระเอียดของออปโต้คัปเปลอร์แบบเอาพุตเป็นโฟโต้ทรานซิสเตอร์ เบอร์ 4N35 ดังแสดงในภาพที่ 2.2





ภาพที่ 2.2 ออปโต้คัปเปลอร์ เบอร์ 4N35 : GENERAL PURPOSE 6-PINPHOTOTRANSISTOR OPTOCOUPLERS

วงจรใช้งานจริงที่แสดงดังภาพที่ 2.3 เป็นการรับสัญญาณจากไฟจราจรและส่งสัญญาณส่งต่อไปยังไมโครคอนโทรลเลอร์ PIC16F84A โดยรายละเอียดของวงจรประกอบด้วย ขา 1 ของออปโต้คัปเปลอร์ เบอร์ 4N35 จะทำหน้าที่รับสัญญาณจากไฟจราจร แล้วขา 4 จะเป็นส่วนที่เชื่อมต่อกับไมโครคอนโทรลเลอร์ ซึ่งเป็นส่วนที่ทำหน้าที่ส่งสัญญาณที่ได้ ให้กับไมโครคอนโทรลเลอร์ PIC16F84A เพื่อทำการประมวลผล ส่วนขา 3 และ ขา 6 จะไม่นำมาใช้งาน แรงดันจะถูกจ่ายเข้าที่ขา Collector (ขา 5) แรงดันที่ใช้จะมีค่าเท่ากับ +5V



ภาพที่ 2.3 วงจรรับสัญญาณไฟจราจรโดยใช้ออปโต้คัปเปลอร์แบบเอาต์พุตเป็นโฟโต้ทรานซิสเตอร์

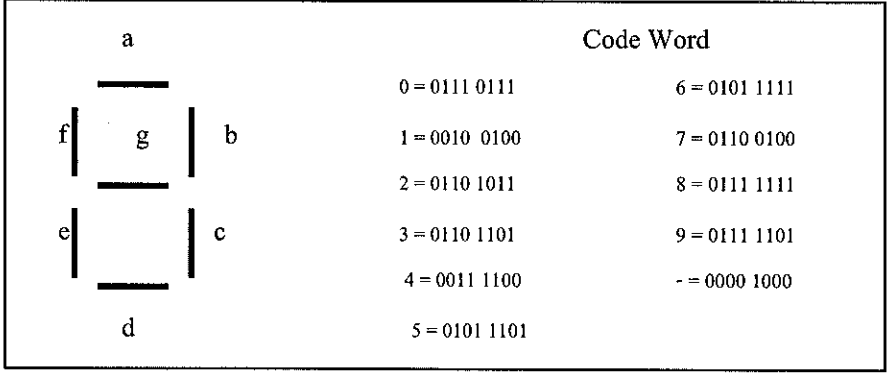
2.3 ส่วนที่ทำหน้าที่ประมวลผลและควบคุม

ในส่วนการประมวลผลและควบคุมนี้เราจะใช้ไมโครคอนโทรลเลอร์ PIC16F84A ในการรับสัญญาณจาก ออปโต้คัปเปลอร์ มาเก็บไว้ในหน่วยความจำ และควบคุมให้บอร์ด LED แสดงตามเวลาที่ตรวจวัดได้ ซึ่งรายละเอียดของส่วนนี้ประกอบด้วย ขาที่ 3 ของไมโครคอนโทรเลอร์ PIC16F84A จะทำรับสัญญาณจาก ออปโต้คัปเปลอร์ ที่ทำหน้าที่รับสัญญาณไฟแดง และขาที่ 2 ของไมโครคอนโทรเลอร์ PIC16F84A จะทำรับสัญญาณจาก ออปโต้คัปเปลอร์ ที่ทำหน้าที่รับสัญญาณไฟเขียว เวลาที่รับเข้ามาจะเก็บไว้ในหน่วยความจำ และจะใช้คริสตอลที่มีความถี่ 4MHz ต่อเข้าที่ขาระหว่างขา 15 และขา 16 โดยเป็นสัญญาณนาฬิกา ในการเทียบเวลาที่ได้ เหตุที่ใช้สัญญาณนาฬิกาเป็นแบบคริสตอลเพราะมีความเที่ยงตรงสูงมากส่งผลให้การคำนวณเกี่ยวกับการหน่วงเวลาในการเขียนโปรแกรมกระทำได้อย่างแม่นยำมากขึ้นเพราะชิ้นงานนี้ต้องการความถูกต้องของเวลาเป็นอย่างมาก

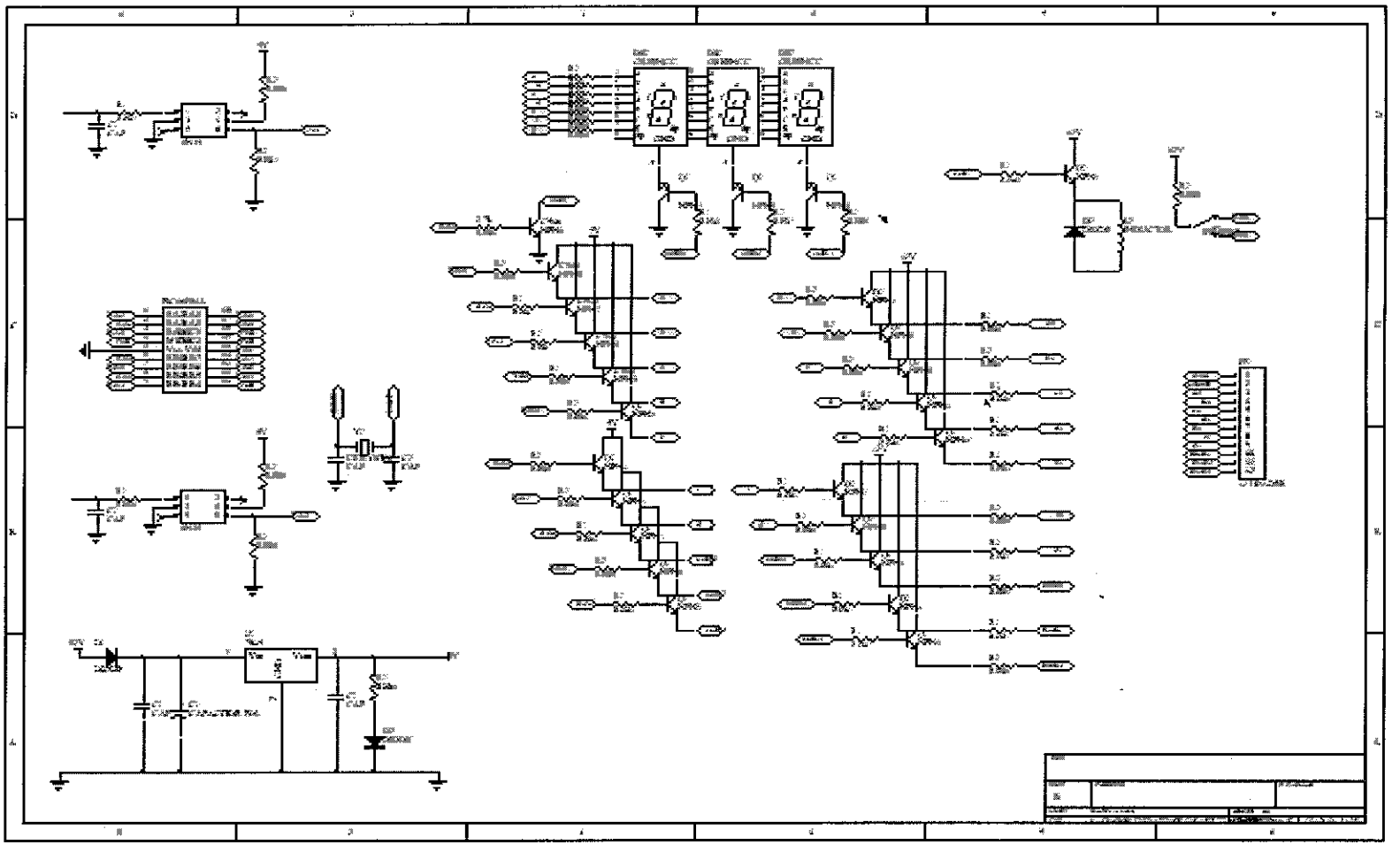
ในไมโครคอนโทรเลอร์ PIC16F84A ต้องการไฟเลี้ยงขนาด +5 V เข้าที่ขา 14 และขา 4 โดยขาที่ 4 จะต้องต่อต่อตัวต้านทานขนาด 4.7 kΩ ขา 5 ของไมโครคอนโทรลเลอร์ จะต่อลงกราวด์ ส่วนขาอื่น ๆ ที่เหลือจะทำหน้าที่ในการควบคุมสัญญาณไฟ ซึ่งจะแสดงให้เห็นดังภาพที่ 2.5

2.4 ส่วนที่ทำหน้าที่แสดงผล

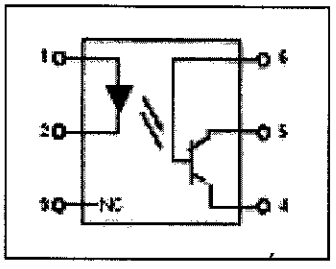
ในส่วนนี้เราจะใช้หลอด LED แบบเลเซอร์ มาต่อเป็น ตัวเลข 7 ส่วน ซึ่งจะใช้วิธีการต่อแบบคอมมอนแคโทด เพื่อให้สั่งการขา แคโทดได้จากการควบคุมของไมโครคอนโทรลเลอร์ PIC16F84A ซึ่งสัญญาณเอาท์พุตเป็นลอจิก 0 เมื่อมีการเลือกใช้งาน และใช้ตัวต้านทานเพื่อจำกัดกระแสที่ไหลผ่าน ตัวเลข 7 ส่วน เหตุที่เลือกใช้หลอด LED แบบเลเซอร์เพราะหลอดชนิดนี้จะให้แสดงสว่างมากกว่า หลอด LED แบบธรรมดา แต่ราคาก็จะเพิ่มขึ้นตามความสว่างของหลอดเช่นกัน ในส่วนของการแสดงผลนี้สิ่งสำคัญก็คือการทำให้หลอดทุกหลอดสว่างเท่ากัน และมีการขับกระแสสูงสุดเพื่อให้หลอดสว่างมากที่สุดด้วย



ภาพที่ 2.4 แสดง 7 – Segment การเข้ารหัส และรหัสแสดงผล

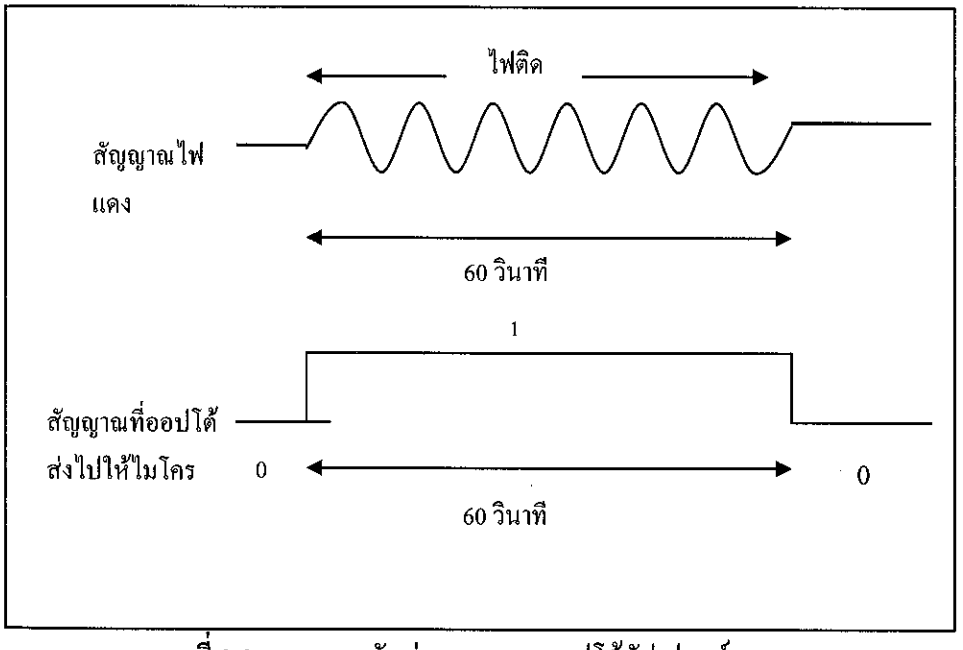


ภาพที่ 2.5 วงจรควบคุมการประมวลผล



ภาพที่ 2.7 แสดงการทำงานของออปโต้คัปเปลอร์

จากคุณสมบัติในการทำงานของออปโต้เราจึงนำมาเป็นตัว รับสัญญาณให้กับไมโครคอนโทรลเลอร์ โดยที่ออปโต้คัปเปลอร์จะทำการ สร้างสัญญาณที่เป็นลักษณะพัลส์เมื่อมีแรงดันมาตกคร่อม และหยุดทำงานเมื่อแรงดันหายไป เช่น เมื่อกำหนดสัญญาณไฟจราจร สีแดง ติดเป็นเวลา 60 วินาทีซึ่งแสดงดังภาพที่ 2.8



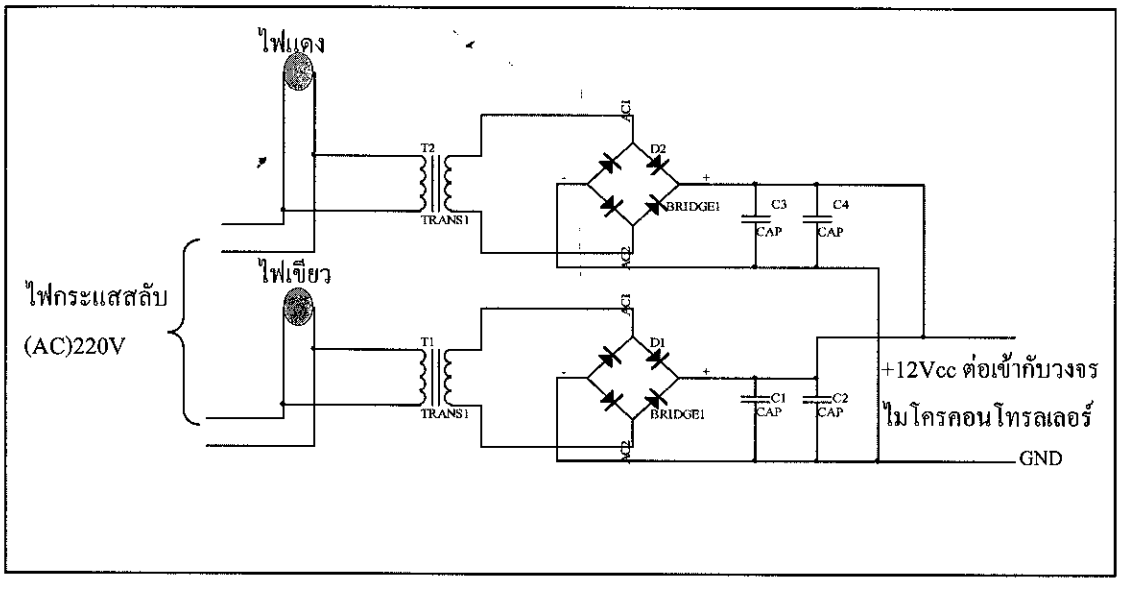
ภาพที่ 2.8 แสดงการนับช่วงเวลาของออปโต้คัปเปลอร์

จากภาพที่ 2.7 เมื่อสัญญาณไฟแดงติด จะมีแรงดันไปตกคร่อมที่ LED ในออปโต้คัปเปลอร์ ทำให้ LED แปล่งแสงออกมา ทำให้ทรานซิสเตอร์ทำงาน แล้วส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ PIC 16F84A เพื่อเก็บค่าของเวลาที่ออปโต้คัปเปลอร์ 4N35 ทำงาน เมื่อสัญญาณไฟแดงหยุดตัด ออปโต้คัปเปลอร์จะหยุดทำงานและไม่มีการส่งสัญญาณให้กับ

ไมโครคอนโทรลเลอร์ ซึ่งก็หมายความว่า จะเก็บช่วงเวลาที่ออปโต้คัปเปลอร์ทำงานเป็นเวลาที่จะทำให้แผงหลอด LED 7-segment แสดงผลต่อไป

2.5.3 ส่วนของแหล่งจ่ายไฟ (Power supply)

ในสภาพของการใช้งานจริง เราต้องการแหล่งจ่ายไฟที่อยู่บริเวณใกล้กับสัญญาณไฟจราจร จึงตัดสินใจที่จะแหล่งจ่ายไฟเดียวกับสัญญาณไฟจราจร ซึ่งจะแส่ดงวิธีการต่อดังภาพที่ 2.8



ภาพที่ 2.9 แสดงวิธีการต่อแหล่งจ่ายไฟ

จากภาพที่ 2.9 เหตุที่ต้องมีแหล่งจ่ายไฟ 2 แหล่ง เพราะหากว่ามีแหล่งจ่ายไฟเพียงแหล่งเดียวและต่อขนานเข้ากับสัญญาณไฟ ชุดใดเพียงชุดหนึ่ง สมมุติว่าเป็นไฟแดง หากว่าสัญญาณไฟแดงทำงานอยู่ ก็จะมีแรงดันไปเลี้ยงให้กับวงจรในส่วนควบคุม แต่ถ้าหากสัญญาณไฟแดงหยุดทำงาน แล้วเปลี่ยนมาเป็นสัญญาณไฟเขียว ก็จะไม่มีไฟไปเลี้ยงให้กับวงจรในภาคควบคุม เพราะฉะนั้นจึงควรที่จะมีแหล่งจ่ายไฟ 2 แหล่งต่อขนานเข้ากับสัญญาณไฟจราจรทั้งหลอดไฟสีเขียวและสีแดง เพื่อว่าสัญญาณไฟจราจรสีใดหยุดทำงานวงจรของเราก็จะมีไฟเลี้ยงภาคควบคุมตลอดเวลา

บทที่3
ไมโครคอนโทรลเลอร์ PIC16F84A

3.1 ความรู้เบื้องต้นเกี่ยวกับไมโครคอนโทรเลอร์ PIC16F84A

PIC 16F84A เป็นไมโครคอนโทรลลเลอร์ในตระกูล PIC (Peripheral Interface Controller) ของไมโครชิป เทคโนโลยี (Microchip Technology) ไมโครคอนโทรลเลอร์ในตระกูล PIC มีด้วยกันหลายเบอร์ แต่ละเบอร์ก็จะมีขีดความสามารถแตกต่างกันไป สำหรับในด้านการเรียนรู้เบื้องต้น เบอร์ที่มีความสามารถเหมาะสมมากที่สุดคือ PIC16F84A เนื่องจากภายใน PIC16F84A มีหน่วยความจำโปรแกรม (Program memory) เป็นแบบเฟลช (Flash) ซึ่งเป็นหน่วยความจำที่สามารถเขียนและลบได้ด้วยสัญญาณไฟฟ้านับพันครั้งเมื่อผู้เรียนหรือผู้ใช้งานต้องการพัฒนาโปรแกรมก็สามารถทำได้อย่างสะดวก

ไมโครคอนโทรลเลอร์ PIC16F84A เป็นไมโครคอนโทรลเลอร์สมัยใหม่ที่จัดอยู่ในกลุ่มของไมโครโปรเซสเซอร์แบบ RISC (Reduced Insturction Ser Computer) กล่าวคือ ไมโครคอนโทรลเลอร์ตระกูลนี้จะมีชุดคำสั่งน้อยมากเพียง 33 คำสั่งพื้นฐานเท่านั้น และทุกคำสั่งสามารถทำงานให้เสร็จสิ้นได้ด้วยการใช้สัญญาณนาฬิกาเพียงลูกเดียว ทั้งยังทำงานในลักษณะไปป์ไลน์ (Pipe line) เหมือนกับไมโครโปรเซสเซอร์ในสมัยใหม่ ความเร็วในการทำงานจึงสูงมากเมื่อเทียบกับไมโครคอนโทรลเลอร์เบอร์อื่นที่ความถี่ของสัญญาณนาฬิกาเท่ากัน

3.1.1 คุณสมบัติทางเทคนิคของ PIC16F84A

สามารถแบ่งออกเป็น 3 ส่วนคือ หน่วยประมวลผลกลาง(Central Processing Unit : CPU), ส่วนของเพอริเฟอรัล (Peripheral) และคุณสมบัติพิเศษอื่นๆ

1) หน่วยประมวลผลกลางภายใน PIC16F84A

. หน่วยประมวลกลางเป็นแบบ PISC

. มีคำสั่งเพียง 33 คำสั่ง ขนาด 14 บิต

. ทุกคำสั่งใช้เวลาในการประมวลผลเพียง 1 ไซเคิลของสัญญาณนาฬิกา หรือประมาณ 400 นาโนวินาที ที่สัญญาณนาฬิกาความถี่ 10 MHz ยกเว้น ชุดคำสั่งการกระโดดจะใช้เวลา 2 ไซเคิลของสัญญาณนาฬิกา

. ประมวลผลข้อมูลขนาด 8 บิต

. มีรีจีสเตอร์ฟังก์ชั่นพิเศษ 15 ตัว

. มีสแต็ก 8 ระดับ

. มีดหมดการอ้างอิงแอดเดรส 3 โหมดคือ แบบโดยตรง (direct), แบบโดยอ้อม (indirect) และแบบสัมพัทธ์ (relative)

. มีแหล่งกำเนิดอินเตอร์รัปต์ 4 แหล่งได้แก่

1. รับสัญญาณจากภายนอกโดยป้อนสัญญาณอินเตอร์รัปต์เข้าที่ขาอินพุต RB0/INT

2. จาก TMR0 ไทเมอร์โอเวอร์โฟลว์

3. เมื่อเกิดการเปลี่ยนแปลงที่พอร์ต B

4. เมื่อการเขียนข้อมูลลงในหน่วยความจำอีอี พร้อมเสร็จสิ้นสมบูรณ์

. หน่วยความจำข้อมูล (data memory) เป็นแบบอีอีพรอมสามารถลบและเขียนใหม่ได้ประมาณล้านครั้งและเก็บข้อมูลได้นาน 40ปี

. ขนาดหน่วยความจำโปรแกรมซึ่งเป็นแบบแฟลซมีขนาด 1 กิโลเวิร์ด (1 เวิร์ดของ PIC16F84A มีขนาด 14 บิต), หน่วยความจำอีอีพรอมภายใน 64 ไบต์ และหน่วยความจำแรม 68 ไบต์ซึ่งใช้เป็นรีจิสเตอร์

2) คุณสมบัติทางเทคนิคของเพอริเฟอรัลใน PIC16F84A

. มีขาอินพุตเอาต์พุต 13 ขา สามารถกำหนดเป็นขาอินพุตหรือเอาต์พุตได้อย่างอิสระ

. กระแสซิงก์/ซอร์สของแต่ละขาอินพุตเอาต์พุตสูงพอที่จะขับ LED ได้โดยตรง

. กระแสซิงก์สูงสุด 25 mA ต่อขา

. กระแสซอร์สสูงสุด 20 mA ต่อขา

. มีไทเมอร์/เคาน์เตอร์ขนาด 8 บิตคือ TMR0 พร้อมกับปรีสเกลเลอร์ขนาด 8 บิตที่สามารถโปรแกรมได้

3) คุณสมบัติอื่นๆ

. มีเพาเวอร์ออนรีเซตในตัว (POR: Power-On Reset)

. มีเพาเวอร์อัปไทเมอร์ในตัว (PWRT: Power-up Timer)

. มีออสซิลเลเตอร์สตาร์ตอัปไทเมอร์ (OST: Oscillator Start-up Timer)

. มีวอตช์ด็อกไทเมอร์ (WDT: Watch Dog Timer) พร้อมกับวงจรออสซิลเลเตอร์ RC ภายใน เพื่อช่วยให้การทำงานของไมโครคอนโทรลเลอร์มีความแน่นอนยิ่งขึ้น

. ป้องกันการคัดลอกข้อมูลในหน่วยความจำโปรแกรม

. มีโหมดประหยัดพลังงานหรือโหมดสลีป (Sleep mode)

. สามารถเลือกวงจรออสซิเลเตอร์ ใช้กำหนดการทำงานได้

. การเขียนข้อมูลเข้าสู่หน่วยความจำภายในไมโครคอนโทรลเลอร์เป็นแบบอนุกรมผ่านขา ใช้งานเพียง 2 ขา

. เป็นไมโครคอนโทรลเลอร์ที่ได้รับการพัฒนาภายใต้เทคโนโลยีซีมอสแฟลช/อีอีพรอม ความเร็วสูง พลังงานต่ำ

. ย่านไฟเลี้ยง 2.0 - 6.0 V

. ปริมาณการใช้กระแสไฟฟ้า

<2 mA ที่ไฟเลี้ยง +5 V สัญญาณนาฬิกาความถี่ 4MHz

15 uA ที่ไฟเลี้ยง +2 V สัญญาณนาฬิกาความถี่ 32kHz

<1 mA ที่ไฟเลี้ยง +2 V ขณะสแตนด์บาย

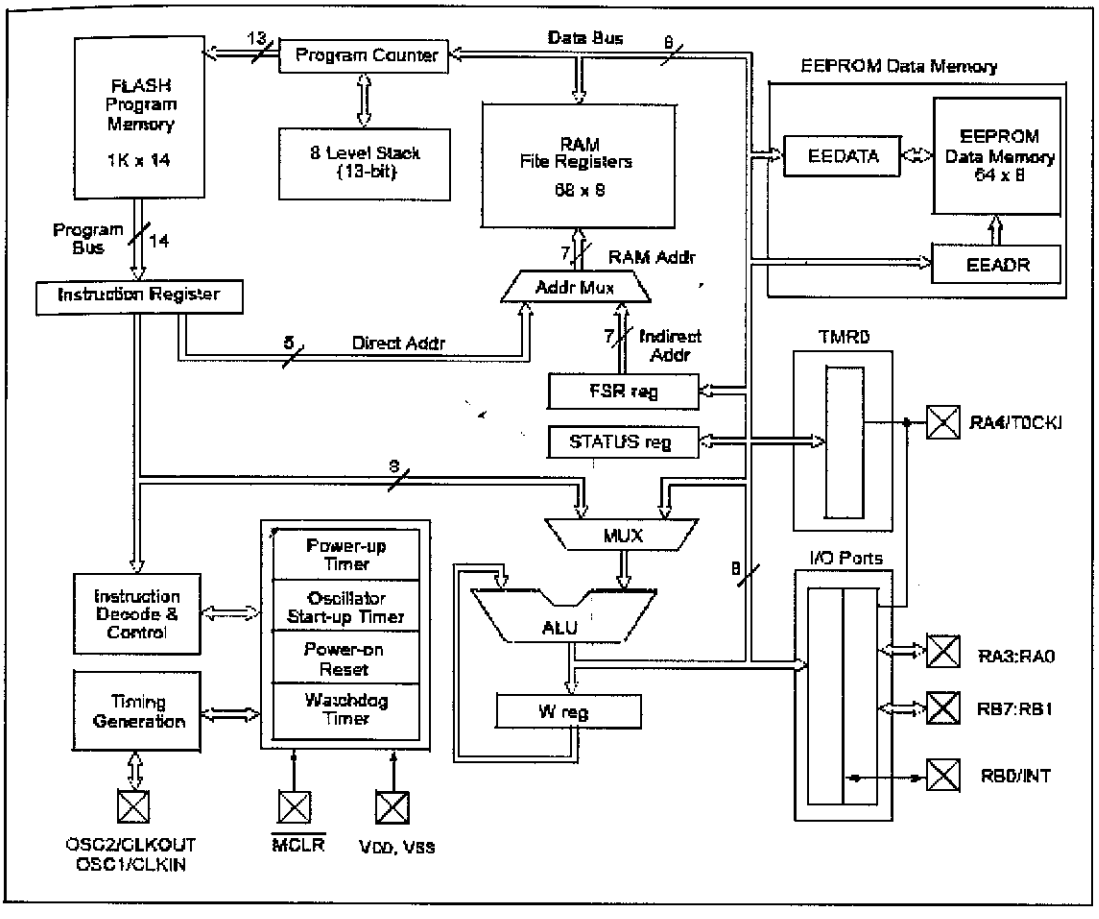
3.1.2 สถาปัตยกรรมของ PIC16F84A

ไมโครคอนโทรลเลอร์ PIC16F84A ได้รับการบรรจุหน่วยประมวลผล, หน่วยความจำ, หน่วยคำนวณทางคณิตศาสตร์และหน่วยอินพุตเอาต์พุตไว้พร้อมสรรพ ทั้งยังมีไทเมอร์และวอตช์ด็อกไทเมอร์ครบถ้วยสมบูรณ์ ดังแสดงสถาปัตยกรรมของไมโครคอนโทรลเลอร์ PIC16F84A ในรูปที่ 3.1-1

PIC16F84A มีการจัดสรรหน่วยความจำดังนี้

- หน่วยความจำโปรแกรมมีโครงสร้างเป็นหน่วยความจำแบบแฟลช มีขนาด 1 กิโลเวิร์ด โดยใน 1 เวิร์ดของ PIC16F84A มีขนาด 14 บิต
- หน่วยความจำข้อมูลเป็นหน่วยความจำแบบอีอีพรอมขนาด 64 ไบต์
- หน่วยความจำแรมได้รับการกำหนดให้ทำงานเป็นรีจิสเตอร์กำหนดแฟ้มข้อมูลหรือรีจิสเตอร์ไฟล์ขนาด 68 ไบต์

การเข้าถึงหน่วยความจำทั้งหมดของหน่วยประมวลผลกลางหรือซีพียูภายในไมโครคอนโทรลเลอร์นี้สามารถทำได้ทั้งในลักษณะโดยตรง, โดยอ้อม และแบบสัมพัทธ์ โดยมีรีจิสเตอร์ FSR (File Select Register) ทำหน้าที่ในการควบคุมการเข้าถึงหน่วยความจำ



ภาพที่ 3.1-1 สถาปัตยกรรมของไมโครคอนโทรลเลอร์ PIC16F84A

เมื่อไมโครคอนโทรลเลอร์ทำงานตามคำสั่งที่กำหนดให้ข้อมูลของชุดคำสั่งจะถูกนำไปเก็บไว้ในรีจิสเตอร์คำสั่ง (Instruction register) จากนั้นจะถูกส่งต่อไปยังวงจรถอดรหัส เพื่อทำการควบคุมไทเมอร์ทั้งหมดภายในตัวไมโครคอนโทรลเลอร์ นอกจากนั้นยังส่งไปควบคุมหน่วยคำนวณทางคณิตศาสตร์โดยผ่านวงจรมัลติเพล็กซ์ด้วย

ใน PIC16F84A มีไทเมอร์/เคาน์เตอร์ขนาด 8 บิต 1 ตัวคือ TMR0 ภายในไทเมอร์/เคาน์เตอร์ตัวนี้มีปรีสเกลเลอร์ขนาด 8 บิต ที่สามารถโปรแกรมได้ โดยมีขาต่อใช้งาน 1 ขาคือขา RA4/T0CK1

หน่วยคำนวณทางคณิตศาสตร์ (Arithmatic Logic Unit : ALU) มีขนาด 8 บิต สามารถทำการบวก, ลบ, เลื่อนข้อมูล และประมวบผลทางลอจิกโดยใช้ฟังก์ชั่นบูลีน ในการทำงาน ALU จะต้องมีรีจิสเตอร์ W ช่วย ซึ่งซีพียูจะไม่สามารถเข้าถึงรีจิสเตอร์ W นี้ได้โดยตรง เมื่อ ALU

ทำงานจะมีผลต่อบิตทด (carry bit) ,บิตหลักทด (digit carry) และบิตศูนย์ (zero bit) ในรีจีสเตอร์ STATRS

ในส่วนของพอร์ตอินพุตใน PIC16F84A มีด้วยกัน 2 พอร์ต คือ พอร์ต A และ B โดยในพอร์ต A มี 5 บิตคือ RA0 – RA4 สำหรับขา RA4 ยังใช้เป็นขาอินพุตสำหรับรับสัญญาณนาฬิกาจากภายนอกให้แก่ TMR0 ด้วยส่วนพอร์ต B มี 8 บิตคือ RB0 – RB7 สำหรับขา RB0 ยังใช้เป็นขาอินพุตสำหรับสัญญาณอินเตอร์รัปต์ด้วย

การกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์ PIC16F84A เป็นหน้าที่ของส่วนกำเนิดจังหวะการทำงาน (timing generation) ซึ่งต้องทำงานสัมพันธ์กับไทเมอร์ทั้ง 3 ตัวคือ เพาเวอร์อัปไทเมอร์ , ออสซิสเลเตอร์สตาร์ตอัปไทเมอร์ และวอตช์ด็อกไทเมอร์สำหรับ PIC16F84A สามารถใช้สัญญาณนาฬิกาจากภายนอกโดยต่อสัญญาณเข้าที่ขา OSC1 และ OSC2 หรือจะใช้สัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์ก็ได้

3.1.3 การจักการขาของ PIC16F84A

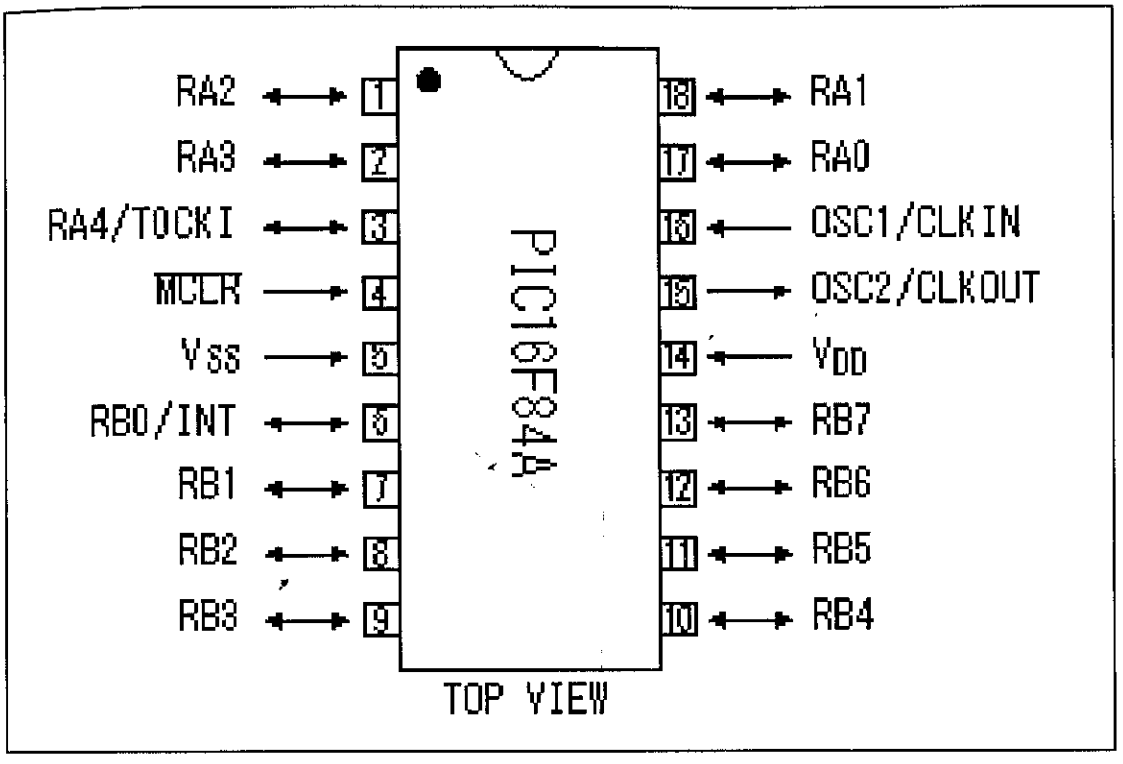
ไมโครคอนโทรลเลอร์ PIC16F84A บรรจุอยู่ในตัวถัง 2 แบบ คือ PDIP (Plastic Dual – In Line package) ซึ่งมีลักษณะเดียวกับไอซีแบบตีนตะขาบที่พบเห็นโดยทั่งไป และแบบ SOIC อันเป็นตัวถังแบบที่ใช้ติดตั้งบนผิวหน้าของแผ่นวงจรพิมพ์ ตัวถังทั้งสองแบบของ PIC16F84A มีขาต่อใช้งานทั้งสิ้น 18 ขา ดังแสดงในรูปที่ 3.1-2 ซึ่งสามารถจัดขาต่อใช้งานของ PIC16F84A เป็น 4 กลุ่มคือ

1. กลุ่มสัญญาณนาฬิกา มี 2 ขาคือ OSC1/CLKIN (ขา 16) และ OSC2/CLKOUT (ขา 15)

2. กลุ่มขาควบคุม มี 1 ขาคือ $\overline{\text{MCLR}}$ (ขา 4)

3. กลุ่มขาพอร์ตอินพุตเอาต์พุต มี 13 ขา แบ่งเป็นขา พอร์ต A 5 ขา ได้แก่ RA0 – RA4 (ขา 17,18,1,2 และ 3) และขาพอร์ต B ได้แก่ ขา RB0 – RB7 (ขา 6 ถึงขา 13)

4. กลุ่มขาไฟเลี้ยง มี 2 ขา คือ ขา Vss (ขา 5) หรือขาต่อ กราวด์ และขา VDD (ขา 14) หรือขาต่อไฟเลี้ยง ปกติใช้ +5 V



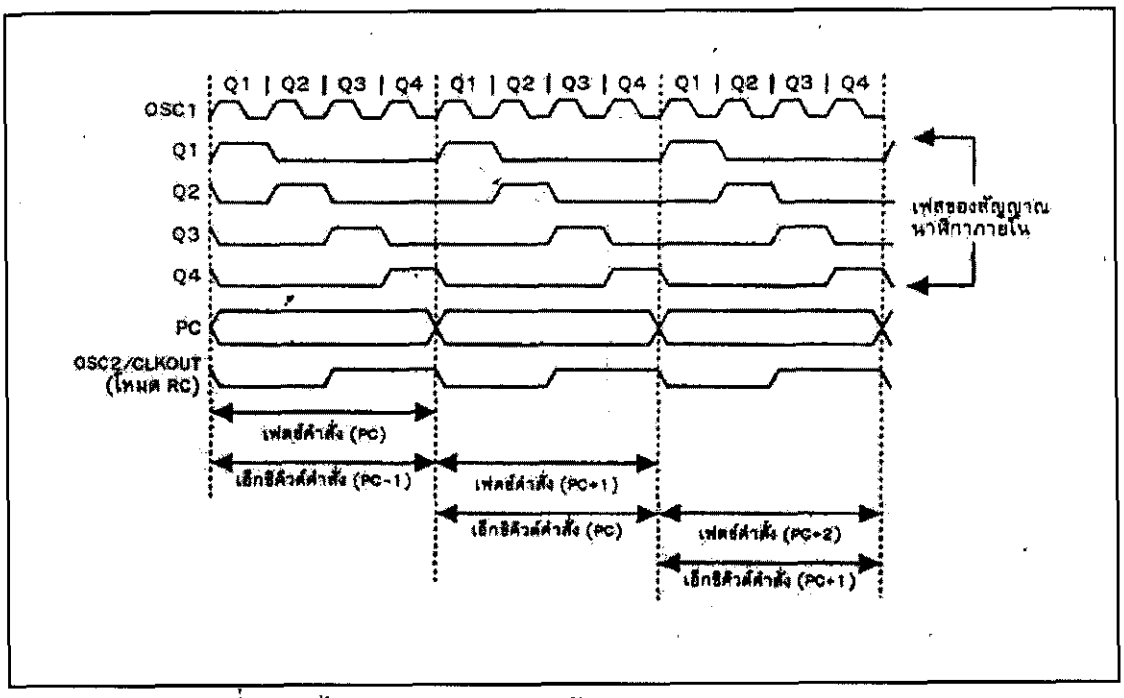
ภาพที่ 3.1-2 การจัดขาของไมโครคอนโทรลเลอร์ PIC16F84A

สำหรับรายละเอียดโดยสรุปของขาต่อใช้งานทั้งหมดแสดงในตารางที่ 3.1-1

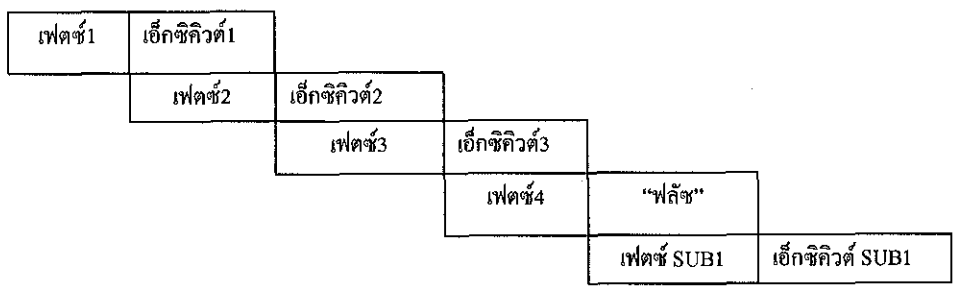
ชื่อขา	ขาที่	ชนิดของขา	ชนิดของบัฟเฟอร์ที่ต่ออยู่	รายละเอียด
OSC1/CLKIN	16	อินพุต	ชมิตต์ทริกเกอร์/ซีมอส[3]	-เป็นขาสำหรับรับสัญญาณนาฬิกาจากคริสตอลหรือจากแหล่งกำเนิดสัญญาณนาฬิกาจากภายนอก
OSC2/CLKOUT	15	เอาต์พุต		-เป็นขาสำหรับส่งสัญญาณนาฬิกาออก -หากทำงานในโหมดคริสตอลให้ต่อกับขาหนึ่งของคริสตอลหรือเซรามิกเรโซเนเตอร์ -หากทำงานในโหมด RC ขานี้ปล่อยลอยไว้ -สัญญาณนาฬิกาที่ออกจากขานี้จะมีความถี่เท่ากับ 1/4 ของความถี่ที่ขา OSC1
$\overline{MCLR}$	4	อินพุต	ชมิตต์ทริกเกอร์	-เป็นขาสำหรับรับสัญญาณรีเซต โดยทำงานที่ลอจิก '0' -เป็นขารับแรงดันสำหรับโปรแกรมหรือเขียนข้อมูลลงในตัวไมโครคอนโทรลเลอร์ด้วย
ขาสัญญาณพอร์ต A				
RA0	17	อินพุต/เอาต์พุต	ทีทีแอล	-เป็นขาอินพุตเอาต์พุต 2 ทิศทางทุกขา
RA1	18	อินพุต/เอาต์พุต	ทีทีแอล	
RA2	1	อินพุต/เอาต์พุต	ทีทีแอล	
RA3	2	อินพุต/เอาต์พุต	ทีทีแอล	
RA4/TOCKI	3	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	-เฉพาะขานี้ใช้เป็นขารับสัญญาณนาฬิกาให้แก่ TMR0 ด้วย
ขาสัญญาณพอร์ต B				
RB0/INT	6	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์[1]	-เป็นขาอินพุตเอาต์พุต 2 ทิศทางทุกขา
RB1	7	อินพุต/เอาต์พุต	ทีทีแอล	-ขา RB0/INT ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์ด้วย
RB2	8	อินพุต/เอาต์พุต	ทีทีแอล	
RB3	9	อินพุต/เอาต์พุต	ทีทีแอล	
RB4	10	อินพุต/เอาต์พุต	ทีทีแอล	-ขา RB4-RB7 ยังใช้เป็นขาที่ทำให้เกิดสัญญาณอินเตอร์รัปต์ได้ โดยการเปลี่ยนแปลงระดับลอจิกที่ขานี้
RB5	11	อินพุต/เอาต์พุต	ทีทีแอล	
RB6	12	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์[2]	-ขา RB6 ยังใช้เป็นขารับสัญญาณนาฬิกาของการโปรแกรมแบบอนุกรมด้วย ในขณะที่ขา RB7 ใช้เป็นขารับข้อมูลของการโปรแกรมแบบอนุกรม
RB7	13	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์[2]	
ขาไฟเลี้ยง				
Vss	5	ขาต่อไฟเลี้ยง	-	-ต่อกับกราวด์
VDD	14	ขาต่อไฟเลี้ยง	-	-ต่อกับไฟเลี้ยงบวก ตั้งแต่ 2-6 V

ตารางที่ 3.1-1 แสดงรายละเอียดของขาต่อใช้งานทั้งหมดของ PIC16F84A

ซีพียูเพื่อเตรียมประมวลผล) จากนั้นซีพียูทำการแลตช์ข้อมูลคำสั่งนั้นไว้ในรีจิสเตอร์คำสั่ง ที่ช่วงสัญญาณนาฬิกา Q4 คำสั่งจะถูกถอดรหัสและเอ็กซิคิวต์(executed : การกระทำตามคำสั่งที่กำหนดให้) จนเสร็จสิ้นภายในช่วงสัญญาณนาฬิกา Q1 – Q4 หรือภายใน 1 ไซเคิลของสัญญาณนาฬิกานั่นเอง



ภาพที่ 3.1-4 ไดอะแกรมเวลาแสดงจังหวะการทำงานของ PIC16F84A



1. movlw 0x55 2. movwf PORTB 3. call SUB1 4. bsf PORTA,3

ช่วงเวลา "ฟลัช" หมายถึง ช่วงเวบาที่ซีพียูทำงานคำสั่งค่อนหน้าซึ่งไม่เสร็จสิ้น เนื่องจากต้องทำการค้นหาตำแหน่งแอดเดรสของโปรแกรมย่อยที่ต้องกระโดดไปทำงาน ช่วงเวลาแบบนี้จะเกิดขึ้น เมื่อซีพียูกระทำคำสั่งเกี่ยวกับการกระโดด ในขณะเดียวกันซีพียูก็จะทำการเฟตซีคำสั่งต่อไปพร้อม ๆ กัน ทำให้ซีพียูสามารถทำงานได้เสมือนหนึ่งว่า 1 คำสั่งใช้เวลา 1 ไซเคิลของสัญญาณนาฬิกาเช่นเดิม เทคนิคลักษณะนี้เรียกว่า "ไปป์ไลน์" (pipeline) ซึ่งเป็นที่นิยมมากในไมโครโปรเซสเซอร์สมัยใหม่

ภาพที่ 3.1-5 แสดงลักษณะการทำงานแบบไปป์ไลน์ที่ใช้ใน PIC16F84A

ในภาพที่ 3.1-4 เป็นไดอะแกรม แสดงความสัมพันธ์ระหว่างจังหวะของสัญญาณนาฬิกากับกรประมวลผลคำสั่งของไมโครคอนโทรลเลอร์ PIC16F84A ถ้าพิจารณาให้ดีจะพบว่า ไมโครคอนโทรลเลอร์ PIC16F84A จะทำงาน โดยใช้สัญญาณนาฬิกาเพียง 1 ลูกต่อหนึ่งคำสั่งเป็นอย่างน้อย แต่สัญญาณนาฬิกาเพียง 1 ลูกจะถูกแบ่งย่อยเป็น 4 ช่วงเพื่อกำหนดจังหวะการทำงานของซีพียูให้มีความชัดเจนมากขึ้น

ไซเคิลการทำงาน (instruction cycle) ของ PIC16F84A แบ่งเป็น 2 ไซเคิลคือ เฟตช์และเอ็กซิคิวต์ ประกอบด้วย 4 คิวไซเคิล (Q-cycle: Quadrarue cycle) คือ Q1, Q2, Q3 และ Q4 การเฟตช์และเอ็กซิคิวต์คำสั่งของไมโครคอนโทรลเลอร์ จะมีลักษณะเป็นแบบไปป์ไลน์ (pipeline การทำงานที่เหลื่อมกันในแต่ละขั้นตอนทางคอมพิวเตอร์ เพื่อที่จะทำให้คอมพิวเตอร์ทำงานได้เร็วขึ้น) กล่าวคือ เมื่อซีพียูกระทำคำสั่งหนึ่ง โดยที่คำสั่งนั้นมี 2 ขั้นตอน เมื่อทำงานไปแล้ว 1 ขั้นตอน คือ ขั้นตอนเฟตช์คำสั่ง ซีพียูจะเริ่มเฟตช์คำสั่งต่อไปทันที พร้อมๆ กับการเอ็กซิคิวต์คำสั่งก่อนหน้านี้ ดังแสดงการบวนการไปป์ไลน์ในภาพที่ 3.1-5

ข้อเด่นของไมโครคอนโทรลเลอร์ตระกูล PIC นี้อยู่ตรงที่การกระทำคำสั่งแยบไปป์ไลน์ อันเป็นกระบวนการกระทำคำสั่งที่ทันสมัย และเป็นเทคโนโลยีที่ใช้ในไมโครโปรเซสเซอร์บนเครื่องคอมพิวเตอร์พีซี ตั้งแต่ระดับ 80286 เป็นต้นมา ทำให้การทำงานของ ไมโครคอนโทรลเลอร์ตระกูล PIC นี้มีความเร็วสูงกว่า ไมโครคอนโทรลเลอร์เบอร์อื่นที่ความถี่ของสัญญาณนาฬิกาเท่ากัน

ไซเคิลการเฟตช์คำสั่งเริ่มต้นในทุก ๆ คิวไซเคิลที่ 1 พร้อมๆ กับคำสั่ง ข้อมูลที่ได้รับการเฟตช์จะถูกนำเข้าไปเก็บไว้ในรีจิสเตอร์คำสั่งในคิวไซเคิลที่ 1 จากนั้นคำสั่งดังกล่าวจะได้รับการถอดรหัส และนำไปดำเนินการจนเสร็จสิ้นภายในคิวไซเคิลที่ 2-4 โดยกำหนดให้ในคิวไซเคิลที่ 2 หน่วยความจำข้อมูลที่ใช้เก็บค่าโอเปอร์แรนด์จะถูกอ่าน และถูกเขียนข้อมูลใหม่เข้าไปแทนที่ในคิวไซเคิลที่4

กลับไปพิจารณาในภาพที่ 3.1-5 ไมโครคอนโทรลเลอร์ PIC16F84A จะกระทำทุกคำสั่งให้เสร็จสิ้นภายใน สัญญาณนาฬิกาเพียง 1 ลูก (ตั้งแต่ Q1-Q4) ยกเว้นในกรณีที่เกิดการสั่งให้กระโดดไปกระทำคำสั่งในโปรแกรมย่อยอื่น ๆ ดังเช่นในไซเคิลการทำงานที่ 4 จะต้องกรำทำคำสั่ง bsf PORTA,3 ในไซเคิลการทำงานที่ 3 มีการสั่งให้ซีพียูกระโดดไปเฟตช์คำสั่งที่โปรแกรมย่อย SUB_1 ทำให้ซีพียูไม่สามารถกระทำคำสั่งในไซเคิลการทำงานที่ 4 ได้เสร็จสิ้นลง เพราะต้องกระโดดไปเฟตช์ คำสั่งที่โปรแกรมย่อย SUB_1 ซีพียูจะกระทำคำสั่ง bsf PORTA,3 ค้างไว้ จนกว่าจะเสร็จสิ้นการทำงานในโปรแกรมย่อย SUB_1 เมื่อเรียบร้อยก็จะกระโดดกลับมากระทำคำสั่งที่ค้างไวให้เสร็จสิ้น ช่วงที่เกิดการทำงานไม่เสร็จสิ้น เนื่องมาจากการกระโดดไปทำงานที่โปรแกรมย่อยเรียกว่า

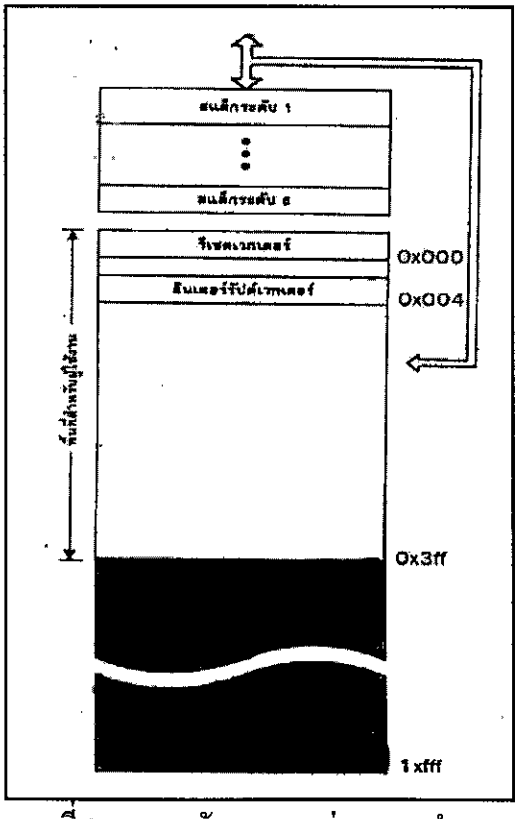
ฟลัช จะเกิดขึ้นเมื่อซีพียูกระทำคำสั่งที่เกี่ยวข้องกับการกระโดดเสมอเนื่องจากคำสั่งนี้จะต้องใช้สัญญาณนาฬิกา 2 ลูก ซึ่งแตกต่างจากคำสั่งอื่น ๆ ที่สามารถดำเนินการเสร็จสิ้นได้ภายในสัญญาณนาฬิกาเพียงลูกเดียว

3.2 การจัดสรรหน่วยความจำใน PIC16F84A รายละเอียดของรีจิสเตอร์ควบคุมและสแต็ก

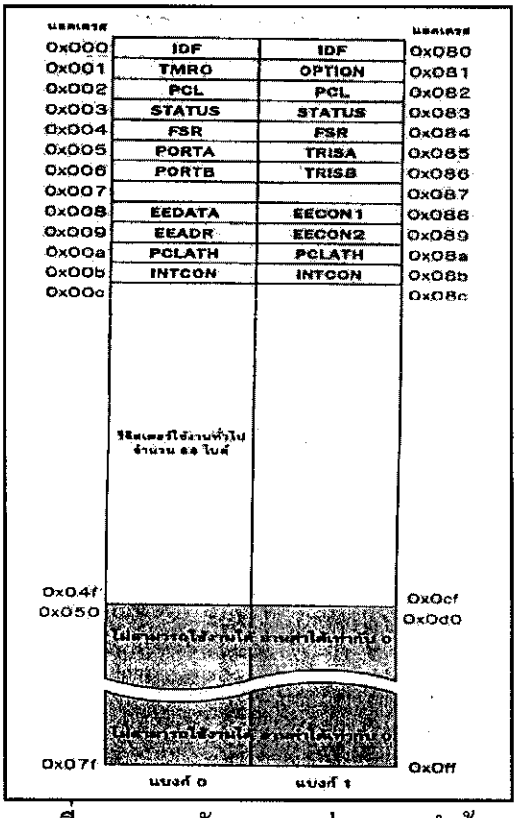
การจัดสรรหน่วยความจำภายใน PIC16F84A แบ่งเป็น 2 ส่วนคือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล

3.2.1 หน่วยความจำโปรแกรม

เป็นหน่วยความจำที่ใช้ในการโปรแกรมควบคุมการทำงานของระบบ หรือใช้เป็นที่เก็บโปรแกรมมอนิเตอร์นั่นเอง หน่วยความจำโปรแกรมใน PIC16F84A เป็นหน่วยความจำแบบแฟลช ในขณะที่ PIC16F84A ทำงานปกติ หน่วยความจำโปรแกรมนี้จะสามารถอ่านได้เพียงอย่างเดียวเท่านั้น และจะสามารถเขียนหรือแก้ไขได้ก็ต่อเมื่อ PIC16F84A อยู่ในโหมดของการโปรแกรมเท่านั้น



ภาพที่ 3.2-1 การจัดสรรพหน่วยความจำโปรแกรมใน PIC16F84



ภาพที่ 3.2-2 การจัดสรรพหน่วยความจำข้อมูลภายใน PIC16F84A

หน่วยความจำโปรแกรมมีขนาด 1 กิโลเวิร์ด (จำนวนบิตต่อ 1 เวิร์ดของหน่วยความจำโปรแกรมของ PIC16F84 นี้เท่ากับ 14 บิต) ได้รับการจัดสรรอยู่ในตำแหน่ง 0000H-03FFH สามารถเขียนข้อมูลเก็บลงในพื้นที่หน่วยความจำนี้ได้ทุกแอดเดรส เว้นแต่แอดเดรส 0000H ซึ่งถูกสงวนไว้สำหรับเก็บค่าเวกเตอร์ของการรีเซตหรือรีเซตเวกเตอร์ (Reset Vector) และแอดเดรส 0004F ซึ่งในการเก็บค่าเวกเตอร์ของการอินเตอร์รัปต์หรืออินเตอร์รัปต์เวกเตอร์ (Interrupt Vector) ในภาพที่ 3.2-1 แสดงการจัดสรรพื้นที่ของหน่วยความจำโปรแกรมในไมโครคอนโทลเลอร์ PIC16F84

3.2.2 หน่วยความจำข้อมูล

พื้นที่ของหน่วยความจำข้อมูลได้รับการจัดสรรเป็น 2 ส่วนคือ พื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษ (Special Function Registers: SFR) และพื้นที่ของรีจิสเตอร์ใช้งานทั่วไป (General Purpose Registers: GPR) การจัดสรรหน่วยความจำข้อมูลทั้ง SFRและ GPR จะจัดแบ่งเป็นแบงค์ (Bank) ในแต่ละแบงค์ของ SFR จะเป็นรีจีสเตอร์ที่ทำหน้าที่แตกต่างกันออกไป การติดต่อกับหน่วยความจำข้อมูลในแต่ละแบงก์ จะต้องกำหนดค่าของบิดควบคุม RP0 และ RP1 ในรีจีสเตอร์ STATUS ในภาพที่ 3.2-2 เป็นการจัดสรรหน่วยความจำข้อมูลของ PIC16F84

จะเห็นได้ว่าหน่วยความจำข้อมูลจะแบ่งออกเป็น 2 แบงก์เพื่อเก็บค่าของ SFR และ GPR หน่วยความจำในแบงก์ 0 จะถูกเลือกเมื่อทำการเคลียร์บิต RP0 (บิต 5 ของรีจีสเตอร์ Status) และถ้าหากบิตนี้เซต (เป็น “1”) จะเป็นการเลือกหน่วยความจำในแบงก์ 1 ในแต่ละแบงก์จะมีขนาด 128 ไบต์ (0×07f) และในทุกๆ 12 ตำแหน่งแรกของแต่ละแบงก์จะถูกสำรองไว้สำหรับเก็บค่าของ SFR

การเข้าถึงหน่วยความจำข้อมูลสามารถทำได้โดยตรง (direct) โดยผ่านรีจีสเตอร์เลือกแฟ้มข้อมูล (File Select Register : FSR)

บิต 7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
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	$\overline{T0}$	$\overline{PD}$	Z	DC	C

R: เป็นบิตที่สามารถอ่านค่าได้
W: เป็นบิตที่สามารถเขียนค่าได้
U:เป็นบิตที่ไม่ใช้งาน อ่านค่าได้เท่ากับ "0"
-n:ค่าที่เกิดขึ้นหลังจากเกิดเพาเวอร์ออนรีเซต

บิต 7: IRP บิตเลือกแบงค์ของรีจิสเตอร์ แต่ใน PIC16F84 ไม่ใช้บิตนี้ ต้องกำหนดให้เป็น "0" เท่านั้น

บิต 0-5: RP1 และ RP0 บิตเลือกแบงค์ของรีจิสเตอร์ สามารถเข้าถึงได้โดยการอ้างแอดเดรสโดยตรง (direct addressing) ใน PIC16F84 จะใช้เพียงบิต RP0 เท่านั้น ในขณะที่ RP1 ต้องกำหนดให้เป็น "0" การ เลือกแบงก์ของรีจิสเตอร์ ทำได้โดยการกำหนดบิต RP1 และ RP0 (Time out bit) ดังนี้

00= แบงก์ 0 (แอดเดรส 0000H-007FH)

01= แบงก์ 1 (แอดเดรส 0080H-00FFH)

บิต 4: $\overline{T0}$ (Time out bit) บิตขอบเขตเวลา บิตนี้สามารถอ่านค่าได้อย่างเดียวเท่านั้น

เป็น "1" เมื่อมีการจ่ายไฟเลี้ยงให้แก่ PIC16F84 หรือ เมื่อมีการกระทำคำสั่ง clrwdt หรือ sleep

เป็น "0" เมื่อวอตช์ด็อกไทเมอร์ (WDT) ภายใน PIC16F84 ทำงานครบเวลาที่กำหนดหรือเกิดไทม์เอาต์

บิต 3: PD (Power down bit) พาเวอร์กาวน์ บิตนี้สามารถอ่านค่าได้อย่างเดียวเท่านั้น เป็นบิตแสดงสถานะการทำงานของ PIC16F84 ในโหมดสลีป (Sleep)

เป็น "1" เมื่อมีการจ่ายไฟเลี้ยงให้แก่ PIC16F84 หรือ เมื่อมีการกระทำคำสั่ง clrwdt

เป็น "0" เมื่อมีกระทำสั่ง sleep

บิต 2: z (Zero bit) บิตศูนย์ เป็นบิตแสดงผลการกระทำทางคณิตศาสตร์

เป็น "1" เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว เกิดค่าศูนย์ (0) ขึ้น

เป็น "0"เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ค่าไม่เป็นศูนย์

บิต 1: DC (Digit carry/borrow bit) บิตทดหรือยืมระหว่างหลัก เป็นบิตแสดงผลทางคณิตศาสตร์

ในกรณีที่กระทำคำสั่ง addwf และ addlw บิต DC จะเกิดผลดังนี้

เป็น "1" เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว เกิดการทดจากบิตล่างที่ 4 ไปยังกลุ่มบิตบน (จากบิตที่ 4 ไปยังบิตที่ 5)

เป็น "0"เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ไม่เกิดการทดจากลิตล่างที่ 4 ไปยังกลุ่มบิตบน (จากบิตที่ 4 ไปยังบิตที่ 5)

ในกรณีที่กระทำคำสั่ง subwf และ sublw บิต DC จะเกิดผลดังนี้

เป็น "1" เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ไม่มีการยืมค่าจากบิตล่างที่ 4 ไปยังกลุ่มบิตบน (บิตที่ 5 ยืมค่าจากบิตที่ 4)

เป็น "0" เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว เกิดการยืมค่าจากบิตล่างที่ 4 ไปยังกลุ่มบิตบน (บิตที่ 5 ยืมค่าจากบิตที่ 4)

บิต 0: c(Carry bit / borrow bit)บิตทดหรือยืม เป็นบิตที่ใช้แสดงผลการทดและยืมค่าทางคณิตศาสตร์ มีลักษณะงานคล้ายกับบิต DC แต่จะเกิดการเปลี่ยนแปลงค่าก็ต่อเมื่อเกิดการทดหรือยืมค่าจากบิตนัยสำคัญสูงสุดหรือบิต MSB

เป็น "1" เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว เกิดการทดจากบิต MSB

เป็น "0" เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้วไม่ เกิดการทด

ในกรณีที่กระทำคำสั่ง subwf และ sublw บิต C จะเกิดผลดังนี้

เป็น "1" เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ไม่มีการยืมค่าจากบิต MSB

เป็น "0" เมื่อกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว เกิดการยืมค่า

ภาพที่ 3.2-3 รายละเอียดของบิตต่างๆ ในรีจีสเตอร์ STATUS

3.2.3 รีจีสเตอร์ควบคุมของ PIC16F84

ใน PIC16F84 มีรีจีสเตอร์ควบคุมที่มีบทบาทสำคัญอยู่ 6 ตัว คือ STATUS, OPTION, PCL, PCLATH และ W ซึ่งจะได้ทำการอธิบายตามลำดับดังนี้

บิต 7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	TOCS	TOSE	$\overline{\text{PSA}}$	PS2	PS1	PS0

R: เป็นบิตที่สามารถอ่านค่าได้
W: เป็นบิตที่สามารถเขียนค่าได้
U:เป็นบิตที่ไม่ใช้งาน อ่านค่าได้เท่ากับ "0"
-n:ค่าที่เกิดขึ้นหลังจากเกิดเพาเวอร์ออนรีเซต

บิต 7: $\overline{\text{RBPU}}$ (Port B pull-up enable bit) ใช้ในการอินาเบิลการพูลอัปที่ขาพอร์ต B
"1" – ดิสเอเบิลการพูลอัปที่พอร์ต B
"0" – อินาเบิลการพูลอัปที่พอร์ต B

บิต 6: INTEDG (Interrupt edge select bit) บิตเลือกขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอก ใช้เลือกขอบขาของสัญญาณที่ทำให้เกิดการอินเตอร์รัปต์ที่ขา RB0/INT
"1" –เลือกขอบขาขึ้นของสัญญาณ
"0" –เลือกขอบขาลงของสัญญาณ

บิต 5: TOCS (TMRO clock source select bit) บิตเลือกแหล่งจ่ายสัญญาณนาฬิกาให้แก่ไทเมอร์เคาน์เตอร์
"1" –เลือกจากการเปลี่ยนแปลงระดับสัญญาณที่ขา RA4/TOCKI
"1" –เลือกจากสัญญาณนาฬิกาที่ใช้กำหนดไซเกิลการทำงานภายใน PIC16F84

บิต 4: TOCS (TMRO source edge select bit) บิตเลือกของขาของสัญญาณนาฬิกาที่ป้อนให้แก่ไทเมอร์เคาน์เตอร์ใช้เลือกของขาของสัญญาณนาฬิกาที่จ่ายเข้ามายังขา RA4/TOCKI ในกรณีที่เลือกแหล่งจ่ายสัญญาณนาฬิกาให้ไทม์เมอร์เคาน์เตอร์ผ่านทางขา RA/TOCKI โดยการกำหนดบิต TOCS ให้เป็น "1"
"1" –กำหนดให้ TMRO เกิดการเพิ่มค่าเมื่อเกิดการเปลี่ยนแปลงระดับสัญญาณที่ขา RA4/TOCKI จากสูงมาต่ำ
"0" –กำหนดให้ TMRO เกิดการเพิ่มค่าเมื่อเกิดการเปลี่ยนแปลงระดับสัญญาณที่ขา RA4/TOCKI จากต่ำไปสูง

บิต 3: PSA(Prescaler assignment bit) บิตกำหนดการทำงานของรีสเกลเลอร์
"1" –กำหนดให้ปรีสเกลเลอร์ทำงานร่วมกับวอตช์ด็อกไทเมอร์ เมื่อทำงานกับวอตช์ด็อกไทเมอร์จะเรียกปรีสเกลเลอร์ว่า โพสต์สเกลเลอร์
"0" –กำหนดให้ ปรีสเกลเลอร์ทำงานร่วมกับ TMRO

บิต 2-0: PS2-PS0 (Prescaler rate select bit) บิตเลือกยัตราส่วนของปรีสเกลเลอร์ เป็นบิตที่ใช้ในการกำหนดอัตราส่วนในการทำงานของปรีสเกลเลอร์ เมื่อทำงานกับ TMRO และวอตช์ด็อกไทเมอร์ อัตราส่วนนี้จะมีค่าไม่เท่ากันการกำหนดอัตราส่วนของปรีสเกลเลอร์เป็นไปดังนี้

PS2	PS0	PS0	อัตราส่วนเมื่อทำงานกับ WDT	อัตราส่วนเมื่อทำงานกับ TMRO
0	0	0	1:1	1:2
0	0	1	1:2	1:4
0	1	0	1:4	1:8
0	1	1	1:8	1:16
1	0	0	1:16	1:32
1	0	1	1:32	1:64
1	1	0	1:64	1:128
1	1	1	1:128	1:256

ภาพที่ 3.2-4 รายละเอียดของบิตต่างๆ ในรีจีสเตอร์ OPTION

1) รีจีสเตอร์ STATUS

เป็นรีจีสเตอร์ที่ใช้แสดงสถานะทางคณิตศาสตร์ของ ALU, สถานะการทำงานของ PIC16F84 และให้เป็นตัวกำหนดการเลือกแบงก์ของหน่วยความจำข้อมูล การเข้าถึงรีจีสเตอร์ STATUS เพื่ออ่านและเขียนข้อมูลสามารถกระทำได้ด้วยวิธีการเดียวกับการอ่านและเขียนรีสิสเตอร์ตัวอื่นๆ และถ้าหากมีการกระทำคำสั่งเกี่ยวกับคณิตศาสตร์และลอจิก บิต Z, DC และ C จะเกิดการเปลี่ยนแปลงค่าตามผลการทำงานที่เกิดขึ้น โดยไม่คำนึงถึงค่าเดิมที่มีการเขียนหรือกำหนดมาก่อนหน้านี้

รีจีสเตอร์ STATUS มีแอดเดรสอยู่ที่ตำแหน่ง 0 ×003 และ 0 ×083 ในหน่วยความจำข้อมูล

สำหรับความหมายและการกำหนดค่าในแต่ละบิตของรีจีสเตอร์ STATUS มีรายละเอียดตามภาพที่ 3.2-3 อย่างไรก็ตามในบางคำสั่งที่จัดการข้อมูลระดับบิตเช่น btfss หรือ btfsc จะไม่มีผลต่อค่าของรีจีสเตอร์ STATUS เช่นเดียวกับคำสั่ง bcf, bsf, swapf และ movwf ซึ่งเมื่อกระทำด้วยตัวดังกล่าวแล้ว บิตที่ใช้แสดงสถานะจะไม่เกิดการเปลี่ยนแต่อย่างใด

2) รีจีสเตอร์ OPTION

เป็นรีจีสเตอร์ที่บรรจุบิตควบคุมการอินเตอร์รัปต์จากสัญญาณภายนอก ปรีสเกลเลอร์ของ TMRO/WDT, การพูลอัปที่ขาของพอร์ต B และควบคุมการทำงานของ TMRO มีแอดเดรสอยู่ที่ 0081H สำหรับรายละเอียดของรีจีสเตอร์ OPTION ในแต่ละบิตแสดงดังภาพที่ 3.2-4

รีจีสเตอร์ INTCON

เป็นรีจีสเตอร์ที่เก็บค่าบิตของการอีนาเบิลสัญญาณอินเตอร์รัปต์ มีแอดเดรสอยู่ที 000BH มีบทบาทสำคัญมากในเรื่องการอินเตอร์รัปต์ ซึ่งมีรายละเอียดเขียนเพิ่มเติมในบทที่ 8 สำหรับรายละเอียดของรีจีสเตอร์ INTCON ในแต่ละบิตแสดงในภาพที่ 3.2-5

รีจีสเตอร์โปรแกรมเคาน์เตอร์ PCL และPCLATH

โปรแกรมเคาน์เตอร์ (Program counter: PC) เป็นรีจีสเตอร์ที่มีหน้าที่ชี้ตำแหน่งแอดเดรสต่อไปของหน่วยความจำโปรแกรมที่ซีพียูจะต้องไปทำงาน

บิต 7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
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	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

R: เป็นบิตที่สามารถอ่านค่าได้
W: เป็นบิตที่สามารถเขียนค่าได้
U:เป็นบิตที่ไม่ใช้งาน อ่านค่าได้เท่ากับ "0"
-n:ค่าที่เกิดขึ้นหลังจากเกิดเพาเวอร์ออนรีเซต

บิต 7: GIE (Global interrupt enable bit) บิตอีนาเบิลการอินเตอร์รัปต์
"1" –อีนาเบิลการอินเตอร์รัปต์ทั้งหมด
"0" –ดีสเอเบิลการอินเตอร์รัปต์ทั้งหมด

บิต 6: EEIE (EEPROM write complete interrupt enable bit) บิตอีนาเบิลการอินเตอร์รัปต์เมื่อการเขียนข้อมูลลงในหน่วยความจำข้อมูลอีอีพรอมเสร็จสมบูรณ์
"1" –อีนาเบิลการอินเตอร์รัปต์แบบนี้
"0" -ดีสเอเบิลการอินเตอร์รัปต์แบบนี้

บิต 5: TOIE (TMRO overflow interrupt enable bit) บิตอีนาเบิลการอินเตอร์รัปต์เมื่อ TMRO เกิดการโอเวอร์โฟลว
"1" –อีนาเบิลการอินเตอร์รัปต์แบบนี้
"0" –ดีสเอเบิลการอินเตอร์รัปต์แบบนี้

บิต 4: INTE (RBO/INT interrupt enable bit) บิตอีนาเบิลการอินเตอร์รัปต์จากสัญญาณภายนอกที่ขา RBO/INT
"1" –อีนาเบิลการอินเตอร์รัปต์แบบนี้
"0" –ดีสเอเบิลการอินเตอร์รัปต์แบบนี้

บิต 3: RBIE (Port B change interrupt enable bit) บิตอีนาเบิลการอินเตอร์รัปต์เนื่องจากเกิดการเปลี่ยนแปลงระดับสัญญาณที่ขาพอร์ต B บิตที่ 4-7
"1" –อีนาเบิลการอินเตอร์รัปต์แบบนี้
"0" –ดีสเอเบิลการอินเตอร์รัปต์แบบนี้

บิต 2: TOIF (TMRO overflow interrupt flag) บิตแจ้งการเกิดอินเตอร์รัปต์อันเนื่องมาจาก TMRO เกิดโอเวอร์โฟลว
"1" –แจ้งให้ทราบว่าเกิดการอินเตอร์รัปต์อันเนื่องมาจาก TMRO เกิดโอเวอร์โฟลว
"0" –ไม่เกิดการอินเตอร์รัปต์อันเนื่องมาจาก TMRO เกิดโอเวอร์โฟลว

บิต 1: INTF (RBO/INT interrupt flag) บิตแจ้งการเกิดอันเตอร์รัปต์เนื่องสัญญาณภายนอกที่ขา RBO/INT
"1" –แจ้งให้ทราบว่าเกิดการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ขา RBO/INT
"0" –ไม่เกิดการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ขา RBO/INT

บิต 0: RBIF (Port B change interrupt flag) บิตแจ้งการเกิดอันเตอร์รัปต์อันเนื่องมาจากเกิดการเปลี่ยนแปลงระดับสัญญาณที่ขาพอร์ต B บิตที่ 4-7
"1" –แจ้งให้ทราบว่าเกิดการอินเตอร์รัปต์อันเนื่องมาจากเกิดการเปลี่ยนแปลงระดับสัญญาณที่ขาพอร์ต B บิตที่ 4-7
"0" –ไม่เกิดการอินเตอร์รัปต์อันเนื่องมาจากเกิดการเปลี่ยนแปลงระดับสัญญาณที่ขาพอร์ต B บิตที่ 4-7

ภาพที่ 3.2-5 รายละเอียดของบิตต่างๆ ในรีจีสเตอร์ INTCON

3.3 พอร์ตของ PIC16F84

ไมโครคอนโทรลเลอร์ PIC16F84มีพอร์ตสำหรับติดต่อกับอุปกรณ์ภายนอก 2 พอร์ต คือ พอร์ต A และพอร์ต B พอร์ต A มีด้วยกัน 5 บิตคือ RAO-RA4 ในขณะที่พอร์ต B มี 8 บิต คือ RB0-RB7

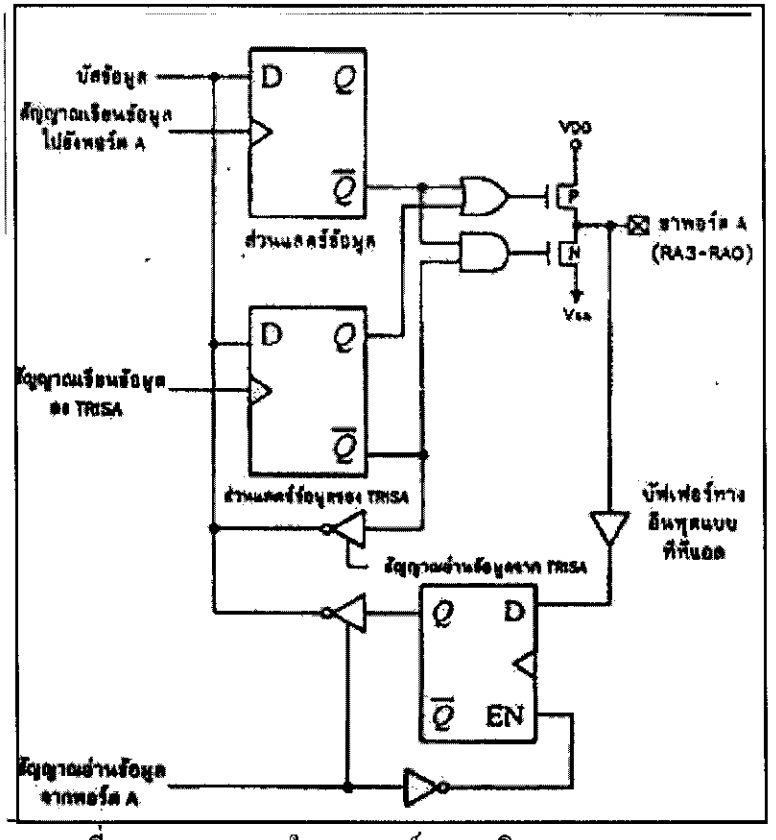
3.3.1 พอร์ต A

1) โครงสร้างทางฮาร์ดแวร์

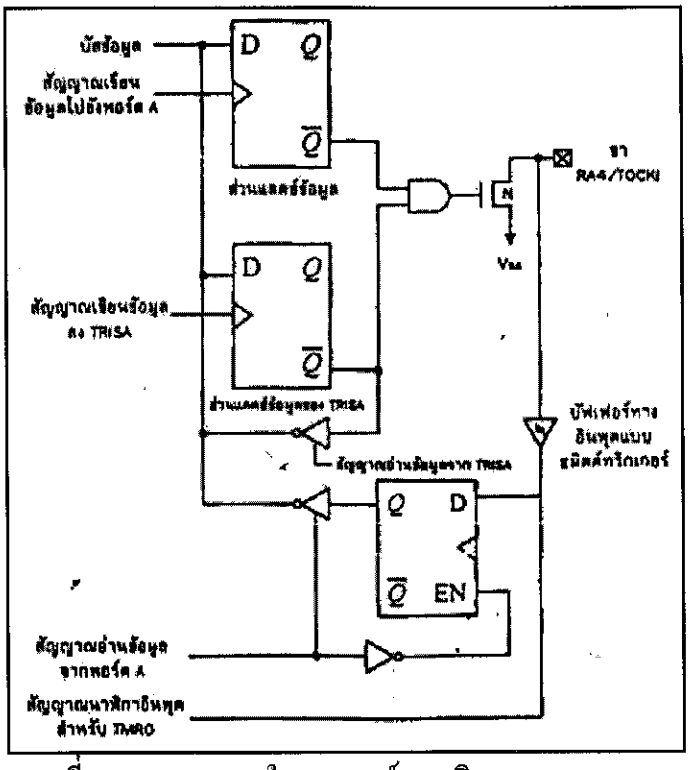
ในภาพที่ 3.3-1 แสดงวงจรภายในพอร์ต A 4 บิตแรกคือ RAO-RA3 สามารถแลตช์ข้อมูลได้พร้อมกันนอกจากนั้นยังมีไว้คอยป้องกันแรงดันย้อนกลับทั้งจากไฟเลี้ยงบวกและลบด้วย ทุกขาของพอร์ต A สามารถรับสัญญาณอินพุตในระดับทีทีแอล (0-5 โวลต์) เมื่อทำงานเป็นอินพุตและสามารถขับอุปกรณ์ซีมอสทางเอาต์พุตได้เมื่อกำหนดให้ทำงานเป็นขาเอาต์พุต

สำหรับขาพอร์ต A บิตที่ 4 หรือ RA4/TOCKI จะมีความแตกต่างจากขาอื่นๆ ของพอร์ต A พิจารณาภาพที่ 3.3-2 เห็นว่ามีวงจรชมิตต์ทริกเกอร์ต่อเข้าที่ขาอินพุต ทั้งนี้เพราะที่ขานี้จะใช้เป็นขาอินพุตรับสัญญาณนาฬิกาภายนอกสำหรับไทเมอร์เคาน์เตอร์ TMRO ภายใน PIC16F84 ทำให้สัญญาณอินพุตที่ต้องการจะต้องมีความเที่ยงตรงสูง วงจรทางเอาต์พุตของขา RA4 จะเป็นเอาต์พุตแบบเดรนเปิด (Open Drain Output)

ที่ 3.3-1เป็นตารางสรุป ในตารางการทำงานของทุกขาสัญญาณของพอร์ต A



ภาพที่ 3.3-1 วงจรภายในของพอร์ต A 4 บิตแรก (RA3-RAO)



ภาพที่ 3.3-2 วงจรภายในของพอร์ต A บิต 4 (RA4/TOCKI)

ชื่อขา	บิตที่	ชนิดของขา	ชนิดของบัฟเฟอร์ที่ต่ออยู่กับขา	รายละเอียด
RA0	0	อินพุต/เอาต์พุต	ทีทีแอล	-เป็นขาอินพุตเอาต์พุต 2 ทิศทางทุกขา
RA1	1	อินพุต/เอาต์พุต	ทีทีแอล	
RA2	2	อินพุต/เอาต์พุต	ทีทีแอล	
RA3	3	อินพุต/เอาต์พุต	ทีทีแอล	
RA4/TOCKI	4	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	-เฉพาะขานี้ใช้เป็นขารับสัญญาณนาฬิกาให้แก่ TMRO ด้วย

ตารางที่ 3.3-1 เป็นตารางสรุปการทำงานของพอร์ต A

2) รีจีสเตอร์ที่เกี่ยวข้องกับพอร์ต A

การทำงานของพอร์ต Aจะมีรีจีสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัวคือ รีจีสเตอร์สำหรับเก็บข้อมูลของพอร์ต A และรีจีสเตอร์กำหนดทิสทางการถ่ายทอดข้อมูลของพอร์ต A หรือ TRISA โดยรีจีสเตอร์พอร์ต A มีขนาด 8 บิต แต่ใช้เพียง 5 บิต โดย 3 บิตบนคือ PORT A5- PORT A7 จะต้องกำหนดให้เป็น “0” ทั้งหมดมีแอดเดรสอยู่ที่ 05H รีจีสเตอร์ PORT A บิต 0 ใช้เก็บข้อมูลของพอร์ต A บิต 0 หรือ RAO ไล่เรียงตามลำดับ จนถึงรีจีสเตอร์ PORT A บิต 4 หรือ PORT A4 ใช้เก็บข้อมูลของพอร์ต A บิต 4 หรือ RA4

สำหรับรีจีสเตอร์ TRISA มีแอดเดรสอยู่ที่ 85 H ใช้กำหนดทิศทางการถ่ายทอดข้อมูลให้ขาสัญญาณของพอร์ต A เป็นขาอินพุตหรือเอาต์พุต ถ้าต้องการให้เป็นอินพุตต้องเขียนข้อมูล "1" ไปยังบิตที่ต้องการกำหนดให้เป็นอินพุต และถ้าหากต้องการกำหนดให้เป็นเอาต์พุตให้เขียนข้อมูล "0" ไปยังบิตนั้น โดย TRISA บิต 0 (TRISA0) ใช้กำหนดการทำงานของ RA0 ไล่เรียงตามลำดับจนถึง TRISA4 ใช้กำหนดการทำงานของ RA4 สำหรับ 3 บิตบนของ TRISA ไม่มีการใช้งานเช่นเดียวกับรีจีสเตอร์ PORT A เมื่อทำการอ่านจะได้ค่าเป็น "0" ทั้งหมด ในตารางที่ 4-2 เป็นตารางสรุปการกำหนดค่าของรีจีสเตอร์ที่เกี่ยวข้องกับพอร์ต A ทั้งหมด

แอดเดรส	รีจีสเตอร์	บิต 7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0	ค่าที่เกิดขึ้นหากเกิดเพาเวอร์ออนรีเซต	ค่าที่เกิดขึ้นหากเกิดการรีเซตแบบอื่น
0 × 005	PORT A	-	-	-	RA4/TOCKI	RA3	RA2	RA1	RA0	ไม่ทราบค่า	ไม่เปลี่ยนแปลง
0 × 085	TRISA	-	-	-	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

หมายเหตุ: x หมายถึงไม่ทราบค่า, u หมายถึงไม่เกิดการเปลี่ยนแปลงข้อมูลที่บิตนั้น, q หมายถึง ค่าจะขึ้นอยู่กับเงื่อนไขของรีจีสเตอร์แต่ละตัว,
- หมายถึง ไม่สามารถเขียนได้ อ่านค่าเป็น 0 เสมอ ส่วนที่แรเงาไม่ใช้งาน ต้องกำหนดให้มีค่าเท่ากับ 0

ตารางที่ 3.3-2 ตารางสรุปการกำหนดค่าของรีจีสเตอร์ที่เกี่ยวข้องกับพอร์ต A ทั้งหมด

3) ตัวอย่างการกำหนดค่าเริ่มต้นของพอร์ต A

ในโปรแกรมที่ 4-1 เป็นโปรแกรมย่อย ตัวอย่างที่เขียนขึ้นเพื่อกำหนดค่าเดิมต้นหรืออินิเชียล (Initialzing)พอร์ต A โดยต้องการกำหนดให้ RA0-RA3 เป็นอินพุต และ RA4 เป็นเอาต์พุต

โปรแกรมที่ 3.3-1 โปรแกรมอินิเชียลพอร์ต A

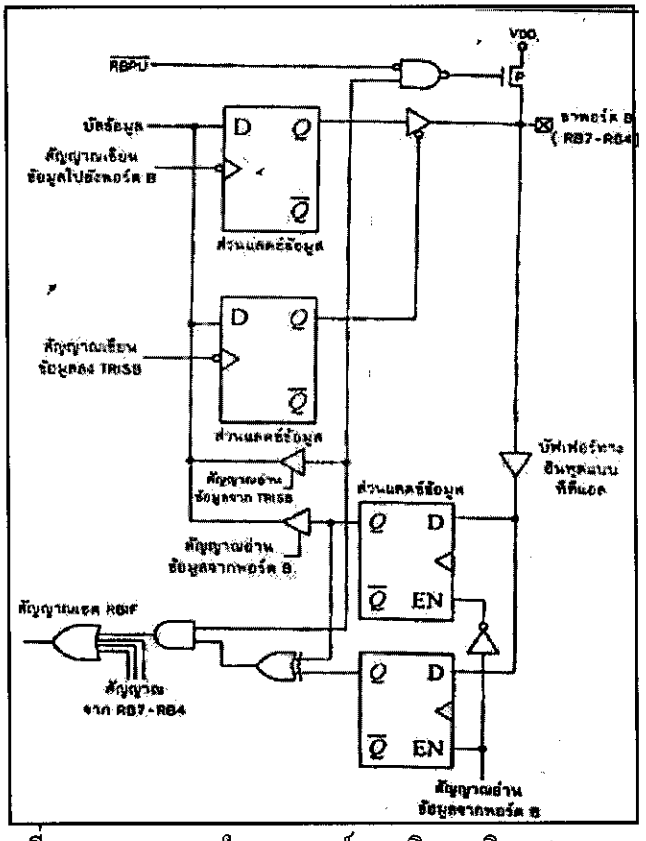
```
Clrf                    : เคลียร์ค่าของรีจีสเตอร์ PORT A
bsf     STATUS, RPO     : เลือกรีจีสเตอร์แบงก์ 1
movlw   0 × 0f          : กำหนดค่าเพื่อเตรียมกำหนดทิศทางการถ่ายทอดข้อมูลของพอร์ต A

movwf   TRISA   : เขียนคำสั่ง OFH ไปยังรีจีสเตอร์ TRISA เป็นการกำหนดให้ RA3- RA0 เป็นอินพุต
                  และ RA4 เป็นเอาต์พุต ส่วนบิต TRISA7- TRISA5 กำหนดให้เป็น "0" ทั้งหมด
```

3.3.2 พอร์ต B

1) โครงสร้างทางฮาร์ดแวร์

ในภาพที่ 3.3-3 และ 3.3-1 แสดงวงจรภายในของพอร์ต B โดยแบ่งเป็น 2 ส่วนคือ ในรูปที่ 3.3-3 เป็นวงจรภายในของพอร์ต B บิต 7- บิต 4 หรือ RB7-RB4 ส่วนในภาพที่ 3.3-4 เป็นวงจรภายในของ RB3-RB0



ภาพที่ 3.3-3 วงจรภายในของพอร์ต B บิต 7- บิต 4 (RB7-RB4)

ชื่อขา	บิตที่	ชนิดของขา	ชนิดของบัฟเฟอร์ที่ต่ออยู่กับขา	รายละเอียด
RB0/INT	0	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์	-เป็นขาอินพุตเอาต์พุต 2 ทิศทางทุกขา
RB1	1	อินพุต/เอาต์พุต	ทีทีแอล	-ขา RB0/INT ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์
RB2	2	อินพุต/เอาต์พุต	ทีทีแอล	รัปต์ด้วย
RB3	3	อินพุต/เอาต์พุต	ทีทีแอล	
RB4	4	อินพุต/เอาต์พุต	ทีทีแอล	
RB5	5	อินพุต/เอาต์พุต	ทีทีแอล	-ขา RB4-RB7ยังใช้เป็นขาอินพุตรับสัญญาณอินเตอร์
RB6	6	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์	รัปต์ได้โดยการเปลี่ยนแปลงระดับลอจิกที่ขานี้
RB7	7	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์	
				-ขา RB6 ยังใช้เป็นขารับสัญญาณนาฬิกาของการโปรแกรมแบบอนุกรมด้วย ในขณะที่ขา RB7 ใช้เป็นขารับข้อมูลของการโปรแกรมแบบอนุกรม

หมายเหตุ : (1) บัฟเฟอร์นี้จะมีอินพุตเป็นแบบชมิตต์ทริกเกอร์เมื่อมีการกำหนดให้เป็นขารับสัญญาณอินเตอร์รัปต์จากภายนอก

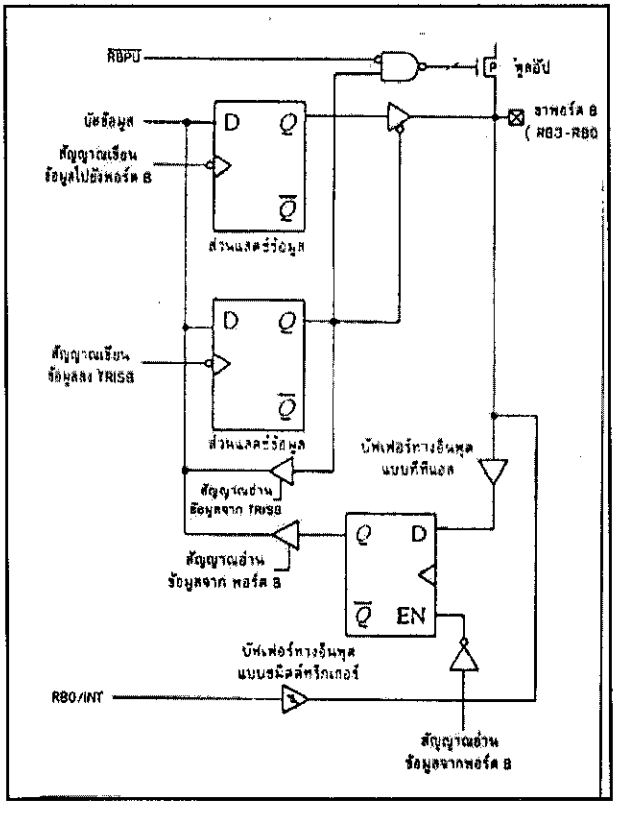
(2) บัฟเฟอร์นี้จะมีอินพุตเป็นแบบชมิตต์ทริกเกอร์เมื่อมใช้ในการโหลดของการโปรแกรมแบบอนุกรม

ตารางที่ 3.3-3 เป็นตารางสรุปการทำงานของพอร์ต B

ทุกสัญญาณของพอร์ต B จะมีการพูลอัปด้วยตัวต้านทานค่าน้อยๆ เมื่อกำหนดที่ขาพอร์ต B เป็นอินพุต โดยต้องมีการเคลียร์บิต RBPU ในรีจีสเตอร์ OPTION เพื่อแจ้งให้ทราบว่าขณะนี้พอร์ต B ถูกกำหนดให้เป็นอินพุตแล้ว และเมื่อกำหนดให้พอร์ต B เป็นเอาต์พุตการพูลอัปจะถูกยกเลิกโดยอัตโนมัติ นอกจากนั้นการพูลอัปจะถูกยกเลิกเมื่อเกิดเพาเวอร์ออนรีเซต

บิต RB7-RB4 นอกจากจะใช้ประโยชน์ในการเป็นขาอินพุตและเอาต์พุตเพื่อถ่ายข้อมูลแล้ว ยังใช้เป็นขาสัญญาณเพื่อทำการตรวจสอบการเปลี่ยนแปลงข้อมูลหรือระดับสัญญาณของพอร์ต B เพื่อกระตุ้นให้เกิดการอินเตอร์รัปต์ โดยแจ้งให้ซีพียูทราบด้วยการเซตบิต RBIF

ส่วนขา RB3-RB0 เป็นขาสัญญาณอินพุตเอาต์พุต ซึ่งได้รับการกำหนดทิศทางจากรีจีสเตอร์ TRISB ในตารางที่ 3.3-3 เป็นตารางสรุปการทำงานของพอร์ต B



ภาพที่ 3.3-4 วงจรภายในของพอร์ต B บิต 3- บิต 0 (RB3-RB0)

2) รีจีสเตอร์ที่เกี่ยวข้องกับ พอร์ต B

เช่นเดียวกับพอร์ต A พอร์ต B จะมีรีจีสเตอร์ที่เกี่ยวข้อง 3 ตัวคือ PORT B, TRISB และ OPTION โดยรีจีสเตอร์ PORT B มีแอดเดรสอยู่ที่ 06H, TRISB มีแอดเดรสที่ 86 H และ OPTION มีแอดเดรสที่ 81 H การกำหนดลักษณะการทำงานของรีจีสเตอร์เป็นไปในลักษณะเดียวกับรีจีสเตอร์ PORT A และ TRISA ที่พิเศษคือต้องมีการกำหนดสถานะของบิต RBPU ของรีจีสเตอร์ OPTION เพื่ออีนาเบิลกาพูลอัปในกรณีที่กำหนดให้ขาของของพอร์ต B เป็นอินพุต

โปรแกรมที่ 3.3-2 โปรแกรมอินิเชียลพอร์ต B

```
Clrf            : เคลียร์ค่าของรีจีสเตอร์ PORT B
bsf   STATUS, RPO : เลือกรีจีสเตอร์แบงก์ 1
movlw 0 × 0f      : กำหนดค่าเพื่อเตรียมกำหนดทิศทางการถ่ายทอดข้อมูลของพอร์ต B

movwf TRISB       : เขียนค่า 0 × CFH ไปยังTRISB เพื่อกำหนดให้ RB3-RB0เป็นอินพุต
                    และ RB4,RB5เป็นเอาต์พุต ส่วน RB6 กับ RB7 เป็นอินพุต
```

แอดเดรส	รีจีสเตอร์	บิต 7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0	ค่าที่เกิดขึ้นหากเกิดเพาเวอร์ออนรีเซต	ค่าที่เกิดขึ้นหากเกิดการรีเซตแบบอื่น
0 × 006	PORT B	RB7	RB6	RB7	RB7	RB7	RB7	RB7	RB7	ไม่ทราบค่า	ไม่เปลี่ยนแปลง
0 × 086	TRISB	TRISB	TRISB	TRISB	TRISB	TRISB	TRISB	TRISB	TRISB	111 1 1111	111 1 1111
0 × 081	OPTION	$\overline{\text{RBPU}}$	$\overline{\text{INTEDG}}$	TOCS	TOSE	PSA	PS2	PS1	PS0	111 1 1111	111 1 1111

หมายเหตุ: x หมายถึงไม่ทราบค่า, u หมายถึงไม่เกิดการเปลี่ยนแปลงข้อมูลที่บิตนั้น, q หมายถึง ค่าจะขึ้นอยู่กับเงื่อนไขของรีจีสเตอร์แต่ละตัว

ส่วนที่แรเงาไม่เกี่ยวข้องกับพอร์ต B

ตารางที่ 3.3-4 ตารางสรุปการกำหนดค่าของรีจีสเตอร์ที่เกี่ยวข้องกับพอร์ต Bทั้งหมด

ตารางที่ 3.3-4 เป็นการสรุปรายละเอียดของรีจีสเตอร์ที่เกี่ยวข้องกับพอร์ต B และในโปรแกรมย่อยตัวอย่างของการอินิเชียลพอร์ต B โดยการกำหนด RB0-RB3 ,RB6 และ RB7 เป็นอินพุต ในขณะที่ RB4 และ RB5 เป็นเอาต์พุต

พอร์ตอินพุตและเอาต์พุตของไมโครคอนโทรเลอร์เป็นส่วนที่มีความสำคัญมาก เพราะเป็นส่วนที่ใช้ติดต่อกับอุปกรณ์ภายนอก เพื่อทำการรับสัญญาณเมื่อเป็นอินพุต และใช้ส่งสัญญาณออกไปควบคุมการทำงานเมื่อทำงานเป็นเอาต์พุต ถ้าจะเปรียบเทียบกับมนุษย์ พอร์ตอินพุตเอาต์พุตก็เหมือนกับแขน, ขา, ตา, จมูก, ปาก นั่นเอง ดังนั้นการเรียนรู้เพื่อใช้งานไมโครคอนโทรลเลอร์จำเป็นอย่างยิ่งที่ต้องเข้าใจเรื่องของการควบคุมและใช้งานพอร์ตอินพุตเอาต์พุตซึ่งสามารถศึกษาในรายละเอียดจากการทดลองทางฮาร์ดแวร์และการประยุกต์ใช้งานในหนังสือที่มีเนื้อหาเกี่ยวข้องได้

3.4 ไทเมอร์เคาน์เตอร์ภายใน PIC16F84

ในไมโครคอนโทรลเลอร์ PIC16F84 มีโมดูลเคาน์เตอร์ขนาด 8 บิต 1 ตัวคือ โมดูลไทเมอร์ 0 ซึ่งมีคุณสมบัติดังนี้

- เป็นไทเมอร์เคาน์เตอร์ขนาด 8 บิต
- สามารถเขียนและอ่านได้
- มีปรีเกลเลอร์ขนาด 8 บิตที่สามารถโปรแกรมได้โดยใช้ซอฟต์แวร์
- สามารถเลือกสัญญาณนาฬิกาจากภายในและภายนอกได้
- กำเนิดสัญญาณอินเตอร์รัปต์เมื่อเกิดโอเวอร์โฟลว์เนื่องจากการเปลี่ยนค่าของการจับจาก $0 \times ff$ เป็น 0×00
- สามารถเลือกขอบขาของสัญญาณจากภายนอกเพื่อกระตุ้นการทำงานได้ทั้งของขาลงและขอบขาขึ้น

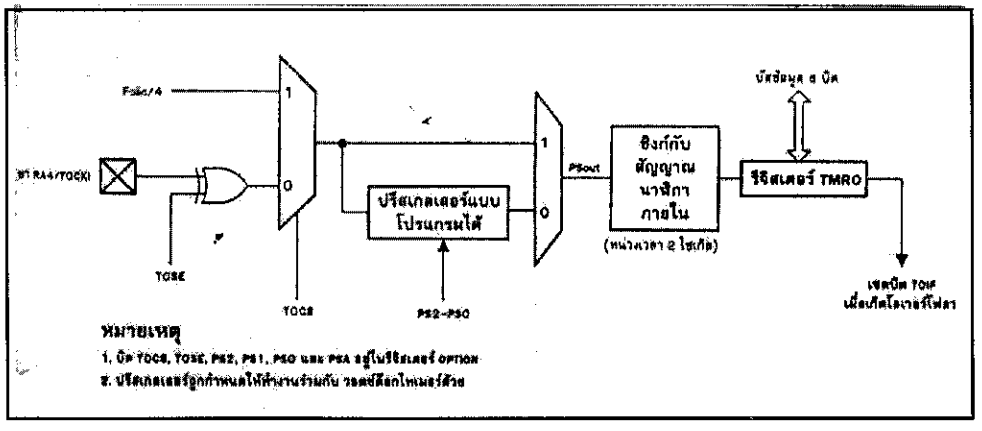
3.4.1 การเลือกโหมดไทเมอร์หรือเคาน์เตอร์

การกำหนดให้โมดูลไทเมอร์ 0 ทำงานในลักษณะเป็นไทเมอร์หรือตัวตั้งเวลา กับเคาน์เตอร์หรือวงจรนับ จะต้องทำการกำหนดบิต TOCS เป็น "0" โมดูลไทเมอร์ 0 จะทำงานในโหมดไทเมอร์ ซึ่งจะรับสัญญาณนาฬิกาจากภายในตัวของ PIC16F84 และถ้าหากกำหนดให้บิต TOCS เป็น "1" โมดูลไทเมอร์ 0 จะทำงานในโหมดเคาน์เตอร์ ซึ่งจะรับสัญญาณนาฬิกาจากภายนอกโดยผ่านมาทางขา RA4/TOCKI

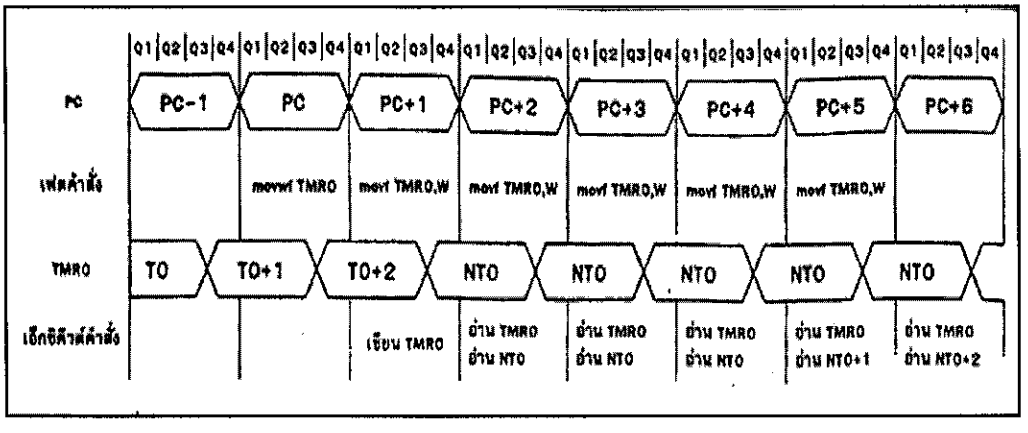
ในภาพที่ 3.4-1 เป็นไดอะแกรมการทำงานของโมดูลไทเมอร์ 0 ภายในโมดูลไทเมอร์มีรีจีสเตอร์ขนาด 8 บิตที่ชื่อว่า TMRO เป็นรีจีสเตอร์ที่ทำหน้าที่เก็บค่าของการนับที่เกิดขึ้นในโมดูลไทเมอร์ 0 เมื่อโมดูลไทเมอร์ 0 ทำงาน ค่าของรีจิสเตอร์ TMRO จะเพิ่มขึ้นในทุกๆไซเกิลของการกระทำคำสั่ง (Instruction Cycle) ถ้าหากมีการเขียนข้อมูลลงในรีจีสเตอร์ TMRO การเพิ่มค่าขึ้นของ TMRO จะเกิดขึ้นหลังจากที่เขียนค่า 2 ไซเกิล ดังแสดงในภาพที่ 3.4-2 ดังนั้นผู้เขียนโปรแกรมจะ

ต้องระมัดระวังในเรื่องนี้ เมื่อทำการเขียนข้อมูลใน TMRO ก่อนจะทำการอ่านค่าของ TMRO มาใช้ ต้องรอให้ผ่านไป 2 ไซเกิลเสียก่อน หรือจะต้องมีการปรับค่าของไทเมอร์เคาน์เตอร์ที่อ่านได้เสมอ

เมื่อโมดูลไทเมอร์ 0 ทำงานในโหมดเคาน์เตอร์ TMRO จะเพิ่มค่าขึ้น เมื่อมีสัญญาณนาฬิกาเข้ามาทาง RA4/TOCKI โดยจะทำงานที่ขอบขาลงหรือขึ้นของสัญญาณนาฬิกานั้นขึ้นอยู่กับการกำหนดที่บิต TOSE อันเป็นบิตที่ 4 ของรีจิสเตอร์ OPTION



ภาพที่ 3.4-1 ไดอะแกรมการทำงานของไทเมอร์เคาน์เตอร์ภายใน PIC16F84



ภาพที่ 3.4-2 เป็นไดอะแกรมเวลาแสดงจังหวะของการเขียนและข้อมูลของ TMRO จะเห็นได้ว่าหลังจากการเขียนข้อมูลแล้วต้องรออย่างน้อย 2 ไซเกิลจึงจะสามารถอ่านค่าของ TMRO ได้

3.4.2 การอินเตอร์รัปต์ของ TMRO

จะเกิดขึ้นเมื่อค่าของรีจิสเตอร์ TMRO เกิดการเปลี่ยนแปลงจาก 0 × ff มาเป็น 0 × 00 หรือเรียกว่า เกิดโอเวอร์โฟลว เมื่อเกิดโอเวอร์โฟลว บิต TOIF ซึ่งเป็นบิต 2 ของ รีจิสเตอร์INCTION จะ

เซตเป็น "1" และถ้าหากไม่ต้องการให้เกิดอินเตอร์รัปต์อันเนื่องมาจากการเกิดโอเวอร์โฟลวของ TMRO ให้ทำการเคลียร์บิต TOIE อันเป็นบิต 5 ของรีจิสเตอร์ INCTION

ถ้าหากเกิดโอเวอร์โฟลวแล้วต้องการเคลียร์บิต TOIF จะต้องใช้วิธีการทางซอฟต์แวร์เท่านั้น ซึ่งทำได้โดยการเขียนโปรแกรมเพื่อเคลียร์บิต TOIF รวมเข้าไปในโปรแกรมย่อยบริการอินเตอร์รัปต์ของโมดูลไทเมอร์ 0 ในภาพที่ 3.4-3 แสดงไดอะแกรมของการเกิดอินเตอร์รัปต์อันเนื่องมาจาก TMRO เกิดโอเวอร์โฟลว

PIC16F84 จะไม่สามารถบริการอินเตอร์รัปต์แบบนี้ได้ หากก่อนหน้านี้ PIC16F84 ถูกกำหนดให้อยู่ในสภาวะสลีป(Sleep) ทั้งนี้เพราะเมื่อ PIC16F84 ได้รับการกำหนดให้ทำงานในโหมดสลีป ไทเมอร์จะถูกควบคุมให้หยุดทำงานโดยอัตโนมัติ

3.4.3 การใช้ TMRO สัญญาณนาฬิกาภายนอก

เมื่อต้องการให้ TMRO ทำงานด้วยสัญญาณนาฬิกาจากภายนอก สิ่งที่จะต้องให้ความสำคัญมากคือ จังหวะเวลาของสัญญาณจะต้องสอดคล้องกับจังหวะการทำงานภายในของ PIC16F84 หรือเกิดการซิงโครไนเซชั่น เมื่อมีการป้อนสัญญาณนาฬิกาจากภายนอกให้แก่ TMRO ผ่านทางขา RA/TOCKI และไม่มีการใช้ปรีสเกลเลอร์ภายใน PIC16F84 การซิงโครไนเซชั่นระหว่างสัญญาณนาฬิกาจากภายนอกที่ขา RA/TOCKI กับสัญญาณนาฬิกาภายในจะเกิดขึ้นจากการสุ่มที่ขาเอาต์พุตปรีสเกลเลอร์ของไซเกิล Q2 และ Q4 ของสัญญาณนาฬิกาภายใน สังเกตจากภาพที่ 3.4-4 สัญญาณที่ขา TOCKI จะต้องมีคาบเวลามากกว่าสัญญาณนาฬิกาภายในหรือ Tose อย่างน้อย 2 เท่า

ในกรณีที่มีการใช้ปรีสเกลเลอร์ สัญญาณนาฬิกาภายนอกจะถูกหารด้วยปรีสเกลเลอร์แบบวงจรนับอะซิงโครนัส (Asynchronous ripple counter type presealer) ทำให้สัญญาณที่เอาต์พุตปรีสเกลเลอร์จะมีความสมมาตรกัน สัญญาณนาฬิกาภายนอกทีจะป้อนเข้าสู่ TMRO ต้องมีคาบเวลาอย่างน้อย 4 เท่าของ Tose และความกว้างของสัญญาณจะต้องไม่น้อยกว่า 10 นาโนวินาที

แอดเดรส	รีจีสเตอร์	บิต 7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0	ค่าที่เกิดขึ้นหากเกิดเพาเวอร์ออนรีเซต	ค่าที่เกิดขึ้นหากเกิดการรีเซตแบบอื่น
0×001	TMRO	เรียวไทม์คล็อกและเคาน์เตอร์ขนาด 8 บิต								ไม่ทราบค่า	ไม่เปลี่ยนแปลง
0×00b	INCTION	GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 0000	0000 0000
0×081	OPTION	RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	111 1 1111	111 1 1111
0×085	TRISA	-	-	-	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

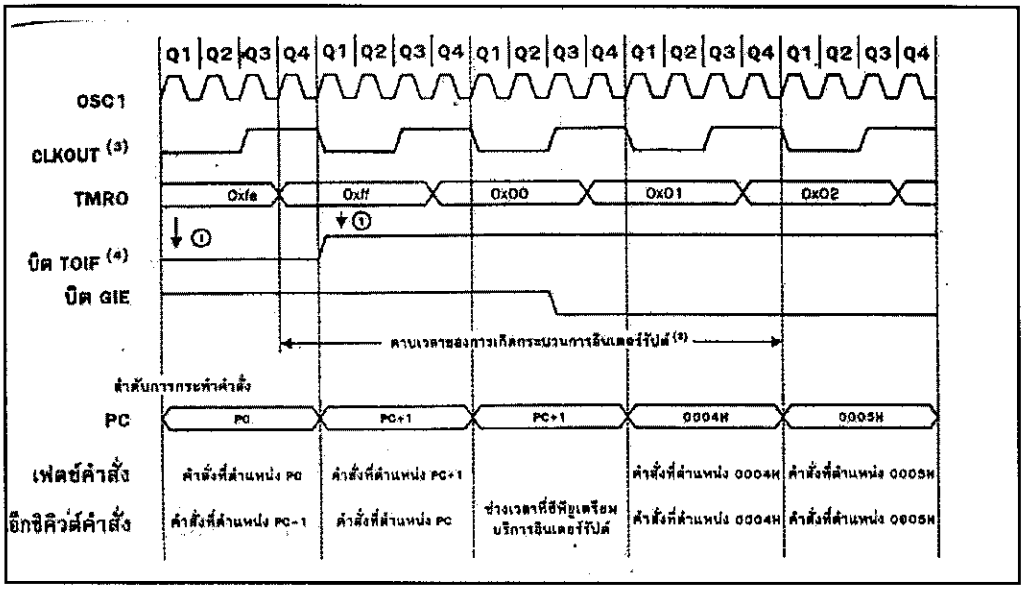
หมายเหตุ: x หมายถึงไม่ทราบค่า, u หมายถึงไม่เกิดการเปลี่ยนแปลงข้อมูลที่บิตนั้น, q หมายถึง ค่าจะขึ้นอยู่กับเงื่อนไขของรีจีสเตอร์แต่ละตัว

ส่วนที่แรเงาเป็นบิตที่ TMRO ไม่ได้ใช้

ตารางที่ 3.4-1 ตารางสรุปรีจิสเตอร์ทั้งหมดที่เกี่ยวข้องกับการทำงานของไทเมอร์เคาน์เตอร์

หมายเหตุ

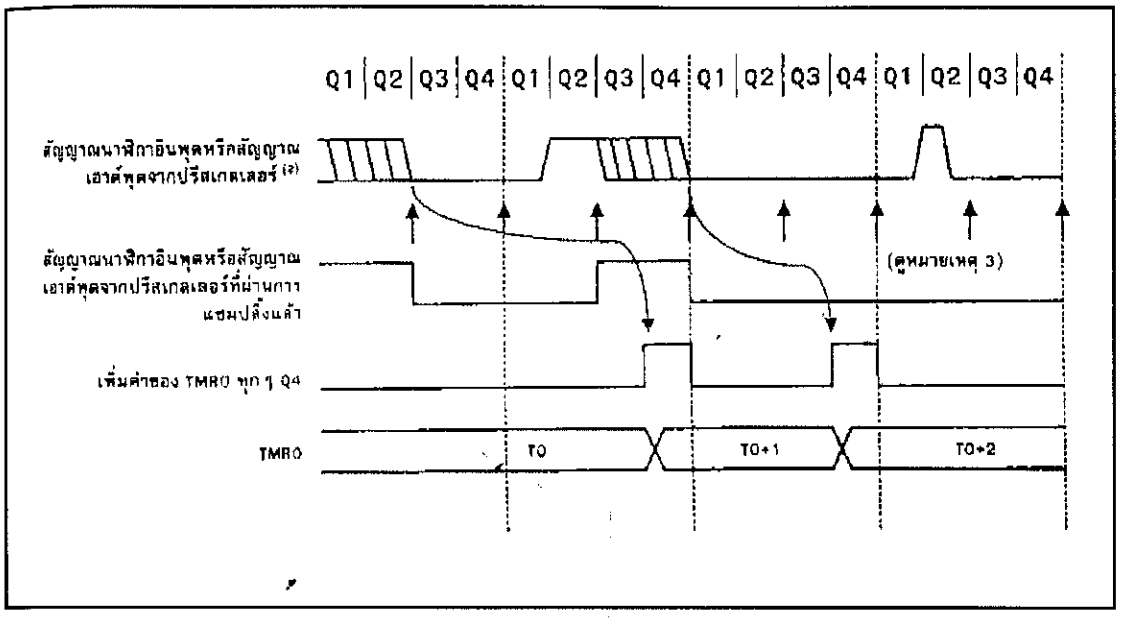
1. แฟลกอินเตอร์รัปต์ TDIF จะถูกสุ่มที่ตำแหน่งนี้ในทุกไซเกิล Q1
2. คาบเวลาทั้งหมดของการอินเตอร์รัปต์เท่ากับ 3.25 Tcy โดยที่ Tcy = เวลาของไซเกิลการทำงาน
3. สัญญาณ CLKOUT จะเกิดขึ้นเมื่อต่อวงจรสัญญาณนาฬิกาแบบ RC
4. ไทเมอร์จะเพิ่มค่าขึ้นจาก FFH เป็น 00H พร้อมๆ กับเซตบิต TDIFในขณะที่รีจิสเตอร์TMRO จะพร้อมทำงานในอีก3 ไซเกิลของ Tcsc ต่อไป



หมายเหตุ

1. แฟลกอินเตอร์รัปต์ TOIFจะถูกสุ่มที่ตำแหน่งนี้ได้ทุกไซเกิล Q1
2. คาบเวลาทั้งหมดของการอินเตอร์รัปต์ เท่ากับ 3.25 Tcy โดยที่ Tcy =เวลาของไซเคิลการทำงาน
3. สัญญาณ CLKOUT จะเกิดขึ้น เมื่อต่อวงจรสัญญาณนาฬิกาแบบ RC
4. ไทเมอร์จะเพิ่มค่าขึ้นจาก FFH เป็น 00H พร้อม ๆ กับเซตบิต TOIF ในขณะที่รีจิสเตอร์ TMRO จะพร้อมทำงานในอีก 3 ไซเกิลของ Tosc ต่อไป

ภาพที่ 3.4-3 เป็นไดอะแกรมเวลาแสดงการเกิดอินเตอร์รัปต์เนื่องจาก TMRO เกิดโอเวอร์โฟลว์

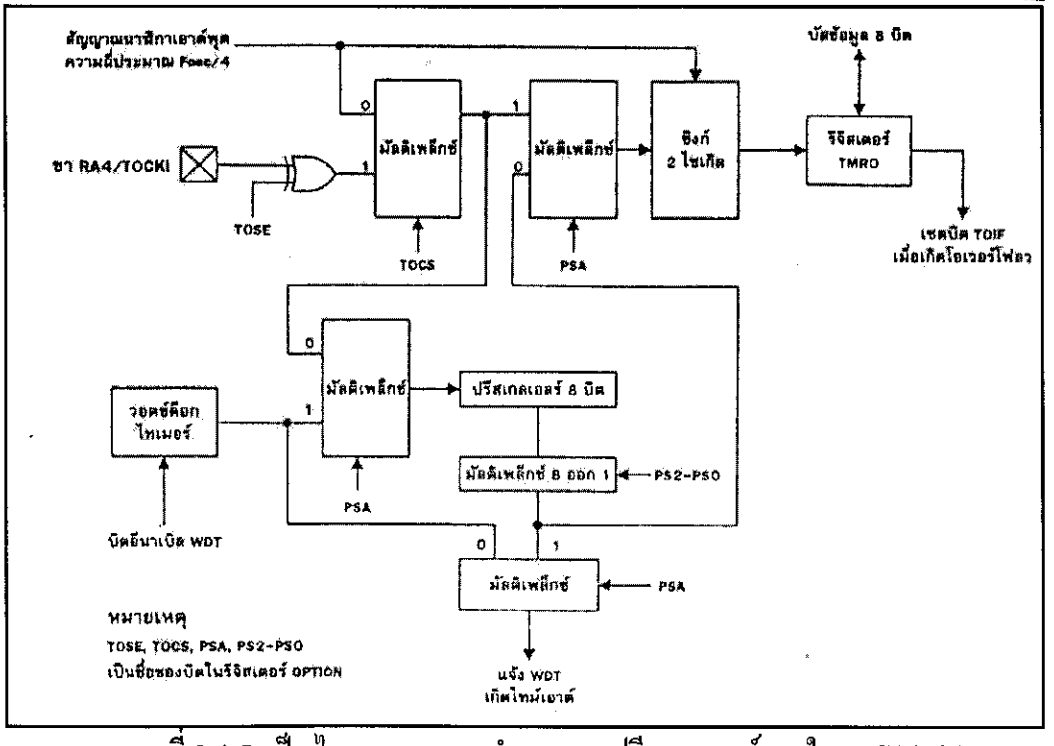


หมายเหตุ 1. ช่วงเวลาหน่วงที่เกิดขึ้นของสัญญาณนาฬิกาอินพุตในขณะที่ TMR0 เพื่อค่าขึ้น มีค่าอยู่ระหว่าง 3Tosc-7Tosc(Tosc คือคาบเวลาของ Q)

2.ถ้าหากปรีสเกลเลอร์ไม่ถูกเลือกให้ทำงาน จะเป็นสัญญาณนาฬิกาจากภายนอกที่ป้อนเข้าที่อินพุต RA4/TOCKI

3. ลูกศรแสดงตำแหน่งที่เกิดการแซมปลิ้ง ถ้าหากสัญญาณที่เข้ามาเป็นพัลส์ที่มีความแคบมาก ๆ สัญญาณนั้นอาจไม่ได้รับการแซมปลิ้ง

ภาพที่ 3.4-4 เป็นไดอะแกรมเวลาแสดงการซิงโครไนเซชั่นระหว่างสัญญาณนาฬิกาจากภายนอกกับสัญญาณนาฬิกาภายในของ TMRO



ภาพที่ 3.4-5 เป็นไดอะแกรมการทำงานของปรีสเกลเลอร์ภายใน PIC16F84

3.4.4 ผลทางเอาต์พุตของไทเมอร์เคาน์เตอร์

ไทเมอร์เคาน์เตอร์จะให้ผลทางเอาต์พุต 2 ลักษณะคือ อ่านค่าของโมดูลไทเมอร์เคาน์เตอร์ที่รีจิสเตอร์ TMRO ที่แอดเดรส 0 × 01 และกำเนิดสัญญาณอินเตอร์รัปต์เมื่อเกิดโอเวอร์โฟลวจากการเปลี่ยนแปลงข้อมูล 0 × ff เป็น 0 × 00

3.4.5 ปรีสเกลเลอร์

คือตัวนับขนาด 8 บิตที่ทำหน้าที่ 2 อย่างคือเป็น ปรีสเกลเลอร์เมื่อทำงานร่วมกับไทเมอร์เคาน์เตอร์ และเป็นโพสต์สเกลเลอร์(post scaler) เมื่อทำงานร่วมกับวอตช์ด็อกไทเมอร์ดังแสดงบล็อกไดอะแกรมของปรีสเกลเลอร์ในภาพที่ 5-5

ปรีสเกลเลอร์จะทำการหารสัญญาณนาฬิกาอินพุตด้วยค่าต่างๆ 1 ใน 8 ค่าซึ่งผู้ใช้งานสามารถกำหนดได้ ส่งผลให้ความถี่ของสัญญาณนาฬิกาลดลง โดยค่าของปรีสเกลเลอร์ที่สามารถกำหนดได้ประกอบด้วย 2, 4, 8, 16, 32, 64, 128 และ 256

หากกำหนดค่าของปรีสเกลเลอร์เท่ากับ 1 หมายความว่าสัญญาณนาฬิกาอินพุตจะผ่านปรีสเกลเลอร์ออกไปโดยไม่มีการลดทอนความถี่ลง ซึ่งก็คือเป็นการกำหนดให้ปรีสเกลเลอร์ทำงานเป็นโพสต์สเกลเลอร์ และเป็นการอีนาเบิลให้วอตช์ด็อกไทเมอร์ทำงาน

โปรแกรมที่ 3.4-1 โปรแกรมการเปลี่ยนการทำงานของปรีสเกลเลอร์ที่ทำงานร่วมกับ TMRO มาเป็น WDT (วอตช์ด็อกไทเมอร์)

```
fbcf    STATUS, RPO     : เลือกรีจิสเตอร์
clrf    TMRO            : เครียร์ค่าของไทเมอร์เคาน์เตอร์และปรีสเกลเลอร์
bsf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 1
clrwdt                  : เครียร์ค่าของวอตช์ด็อกไทเมอร์ (WDT)
movlw   xxxx 1xxx       : เลือกค่าของปรีสเกลเลอร์ใหม่
movwf   OPTION          : เขียนค่าลงในรีจิสเตอร์ OPTION
bcf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 0
```

โปรแกรมที่ 3.4-2 โปรแกรมย่อยการเปลี่ยนการทำงานของปรีสเกลเลอร์ที่ทำงานร่วมกับ WDT มาเป็น TMRO

```
clrf                    : เครียร์ค่าของ WDT และปรีสเกลเลอร์
bsf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 1
movlw   xxxx 0xxx       :กำหนดค่าของปรีสเกลเลอร์ใหม่ เลือกทำงานกับ TMRO และเลือก
                         แหล่งจ่ายสัญญาณนาฬิกา
movwf   OPTION          : เขียนค่าลงในรีจิสเตอร์ OPTION
bcf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 0
```

การกำหนดค่าของปรีสเกลเลอร์ เพื่อทำการลดทอนสัญญาณนาฬิกาสามารถทำได้โดยกำหนดที่รีจิสเตอร์ OPTION 4 บิต คือ บิต PSA, PS2-PS0 (ดูรายละเอียดในหัวข้อรีจิสเตอร์ OPTION)

ค่าของปรีสเกลเลอร์จะถูกเคลียร์เมื่อ PIC16F84 กระทำคำสั่งที่เกี่ยวข้องกับการเขียนข้อมูลมายัง TMRO เช่น clrf, movwf, bsf เป็นต้น

3.4.6 การกำหนดและเปลี่ยนการทำงานของปรีสเกลเลอร์

สามารถกระทำได้ด้วยกระบวนการทางซอฟต์แวร์ดังมีตัวอย่างโปรแกรมที่เขียนขึ้นเพื่อกำหนดและเปลี่ยนการทำงานของปรีสเกลเลอร์ในโปรแกรมที่ 3.4-1 และ 3.4-2

ไทเมอร์เคาน์เตอร์ภายใน PIC16F84 เป็นอีกส่วนหนึ่งที่มีความสำคัญมาก โดยเฉพาะอย่างยิ่งเมื่อนำมาใช้ในการควบคุมอุปกรณ์ภายนอกที่อาศัยเงื่อนไขของเวลาและความถี่นอกจากนั้นยังมีประโยชน์เมื่อทำงานร่วมกับวอตช์ด็อกไทเมอร์ในการตรวจสอบการทำงานของระบบไมโครคอนโทลเลอร์

3.5 หน่วยความจำข้อมูลอีอีพรอมภายใน PIC16F84

หน่วยความจำข้อมูลอีอีพรอมเป็นหน่วยความจำที่สามารถอ่านและเขียนได้ตลอดเวลาด้วยสัญญาณไฟฟ้า ที่ระดับไฟเลี้ยงสูงสุดคือ +5 V หน่วยความจำในส่วนนี้ไม่ได้ถูกจัดอยู่ในพื้นที่ของรีจิสเตอร์ไฟล์ ดังนั้นการติดต่อกับหน่วยความจำส่วนนี้ จึงต้องใช้วิธีการติดต่อโดยอ้อมผ่านรีจิสเตอร์ฟังก์ชันพิเศษ (SFR) 4 ตัว คือ EEDATA, EEADR, EECON1และ EECON2 ในตารางที่ 3.5-1 เป็นการสรุปรายละเอียดของรีจิสเตอร์ที่เกี่ยวข้องกับหน่วยความจำข้อมูลอีอีพรอมทั้งหมด

3.5.1 EEDATA

ใช้เป็นที่พักของหน่วยความจำข้อมูลขนาด 8 บิตที่ต้องการอ่านหรือเขียนมีจำนวน 64 ไบต์ มีแอดเดรสอยู่ที่ 00H-3F การเขียนและอ่านข้อมูลจะกระทำได้ในระดับไบต์หรือครั้งละ 8 บิตเท่านั้น การเขียนข้อมูลลงในหน่วยความจำอีอีพรอมทุกครั้งต้องทำการลบข้อมูลออกก่อนเสมอ ซึ่งอัตราเร็วในการลบข้อมูลจะสูง ในขณะที่อัตราในการเขียนข้อมูลจะขึ้นอยู่กับไทเมอร์ในตัว PIC16F84 ซึ่งจะเปลี่ยนแปลงตามแรงดันและอุณหภูมิในขณะที่ทำการเขียนข้อมูลนั้น

เมื่อทำการป้องกันข้อมูลในหน่วยความจำอีอีพรอมแล้วซีพียูยังสามารถอ่านและเขียนข้อมูลในหน่วยความจำได้เป็นปกติแต่เครื่องโปรแกรมภายจอกจะไม่สามารถเข้าถึงหน่วยความจำส่วนนี้ได้

3.5.2 EEADR

เป็นรีจิสเตอร์ ที่ใช้เก็บค่าแอดเดรสเพื่อเข้าถึงหน่วยความจำข้อมูลอีอีพรอม มีขนาด 8 บิต เนื่องจากหน่วยความจำข้อมูลมีจำนวน 64 ไบต์ ทำให้ 2 บิตบนของ EEADRต้องกำหนดให้เป็น "0" เสมอ เพื่อให้แน่ใจว่า EEADR จะติดต่อกับหน่วยความจำข้อมูลอีอีพรอมจำนวน 64 ไบต์ที่ต้องการเท่านั้น

3.5.3 EECON1และ EECON2

EECON1 เป็นรีจิสเตอร์ควบคุมขนาด 8 บิต แต่มีการใช้งานเพียง 5 บิตล่างเท่านั้น ดังนั้นใน 3 บิตบนจะต้องกำหนดให้เป็น "0" ทั้งหมด รายละเอียดของรีจิสเตอร์ EECONแสดงในภาพที่ 3.5-1

บิตควบคุมRDและ WR ใช้ในการอินิเชียลการอ่านและเขียนทั้ง 2 บิตนี้ไม่สามารถเคลียร์ได้ทางซอฟต์แวร์ แต่สามารถเคลียร์ได้ด้วยกระบวนการอ่านและเขียนทางฮาร์ดแวร์

บิต WRERR จะถูกเซตถ้าหากกระบวนการเขียนข้อมูลในหน่วยความจำอีอีพรอมถูกขัดจังหวะโดยการรีเซตของ MCLR หรือการรีเซตอันเนื่องจาก WDT เกินเวลาหรือเกิดไทม์เอาต์ (time-out) ขณะอยู่โหมดการทำงานปกติ ถ้าหากเกิดเหตุการณ์นี้ขึ้น หลังจากการรีเซตระบบ ผู้ใช้งานสามารถตรวจสอบบิต แล้วทำการเคลียร์ จากนั้นจึงดำเนินการเขียนข้อมูลที่ตำแหน่งเดิมอีกครั้งเพราะทั้งข้อมูลและแอดเดรสที่อยู่ใน EEDATA และ EEADR จะยังไม่เปลี่ยนแปลง

แอดเดรส	รีจิสเตอร์	บิต 7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0	ค่าที่เกิดขึ้นหากเกิดเพาเวอร์ออนรีเซต	ค่าที่เกิดขึ้นหากเกิดการรีเซตแบบอื่น
0×008	EEDATA	รีจิสเตอร์สำหรับเก็บข้อมูลของหน่วยความจำอีอีพรอม								ไม่ทราบค่า	ไม่เปลี่ยนแปลง
0×009	EEADR	รีจิสเตอร์สำหรับเก็บค่าแอดเรสของหน่วยความจำอีอีพรอม								ไม่ทราบค่า	ไม่เปลี่ยนแปลง
0×088	EECON1	-	-	-	EEIF	WRERR	WREN	WR	RD	---0 X000	---0 q000
0×089	EECON2	รีจิสเตอร์ควบคุมหน่วยความจำอีอีพรอม 2 (ไม่สามารถอ่านข้อมูลได้โดยตรง)								---- ----	---- ----

หมายเหตุ: x หมายถึงไม่ทราบค่า, u หมายถึงไม่เกิดการเปลี่ยนแปลงข้อมูลที่บิตนั้น, q หมายถึง ค่าจะขึ้นอยู่กับเงื่อนไขของรีจิสเตอร์แต่ละตัว

(-) หมายถึง บิตนี้ไม่สามารถเขียนได้ อ่านค่าได้เท่ากับ 0 เสมอ ส่วนที่แรเงา เป็นบิตที่ใช้ในการติดต่อกับหน่วยความจำข้อมูลอีอีพรอม

ตารางที่ 3.5-1 ตารางสรุปรีจิสเตอร์ที่เกี่ยวข้องกับหน่วยความจำข้อมูลอีอีพรอมทั้งหมด

บิต 7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
U	U	U	R/W-0	R/W-X	R/W-0	R/S-0	R/S-X
-	-	-	EEIE	WRERR	WREN	WR	RD

R: เป็นบิตที่สามารถอ่านค่าได้
W: เป็นบิตที่สามารถเขียนค่าได้
U:เป็นบิตที่ไม่ใช้งาน อ่านค่าได้เท่ากับ "0"
-n:ค่าที่เกิดขึ้นหลังจากเกิดเพาเวอร์ออนรีเซต

บิต 7: IRP บิตเลือกแบงค์ของรีจิสเตอร์ แต่ใน

บิต 7-5: ใน PIC16F84 ไม่ใช่บิตนี้ ต้องกำหนดให้เป็น "0" เท่านั้น

บิต 4: EEIF (Write operation interrupt flag bit) บิตแสดงการเกิดอินเตอร์รัปต์เมื่อการเขียนข้อมูลลงในหน่วยความจำอีอีพรอมสมบูรณ์
เป็น "1" เมื่อมีการเขียนข้อมูลลงในหน่วยความจำอีอีพรอมสมบูรณ์ (ต้องเคลียร์ด้วยกระบวนการทางซอฟต์แวร์)
เป็น "0" เมื่อการเขียนข้อมูลลงในหน่วยความจำอีอีพรอมไม่สมบูรณ์หรือยังไม่เริ่มต้นการเขียนข้อมูล

บิต 3: WRERR(EEPROM Error flag bit) บิตแสดงความผิดพลาดในการเขียนข้อมูลลงในหน่วยความจำอีอีพรอม
เป็น "1" เมื่อการเขียนข้อมูลถูกทำให้ยกเลิกหรือถูกขัดจังหวะ ทำให้การเขียนข้อมูลเกิดความผิดพลาด
เป็น "0" เมื่อการเขียนข้อมูลสมบูรณ์

บิต 2: WREN (EEPROM Write enable bit) บิตอนุญาตการเขียนข้อมูลลงในหน่วยความจำอีอีพรอม
เป็น "1" ยอมให้เกิดการเขียนข้อมูล
เป็น "0"ไม่ยอมให้เกิดการเขียนข้อมูลลงในหน่วยความจำอีอีพรอม

บิต 1: WR (Write control bit) บิตควบคุมการเขียนข้อมูลลงในหน่วยความจำอีอีพรอม
เป็น "1" แสดงถึงไซเกิลของการอินิเชียลการเขียนข้อมูล หรือใช้เป็นบิตเริ่มต้นการเขียนข้อมูลสามารถเซตได้ทางซอฟต์แวร์ และบิตนี้จะเคลียร์อัตโนมัติ เมื่อการเขียนข้อมูลเสร็จสมบูรณ์
เป็น "0"เมื่อไซเกิลของการเขียนข้อมูลเกิดขึ้นอย่างสมบูรณ์

บิต 0: RD(Read control bit) บิตควบคุมการอ่านข้อมูลจากหน่วยความจำอีอีพรอม
เป็น "1" แสดงถึงไซเกิลของการอินิเชียลการอ่านข้อมูล หรือใช้เป็นบิตเริ่มต้นการอ่านข้อมูลสามารถเซตได้ทางซอฟต์แวร์ และบิตนี้จะเคลียร์อัตโนมัติ เมื่อการอ่านข้อมูลเสร็จสมบูรณ์
เป็น "0"เมื่อไซเกิลของการเขียนข้อมูลเกิดขึ้นอย่างสมบูรณ

ภาพที่ 3.5-1 รายละเอียดของรีจิสเตอร์ EECON1

บิตอินเตอร์รัปต์ EEIF จะถูกเซตเมื่อการเขียนข้อมูลสมบูรณ์ บิตนี้จะถูกเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์เท่านั้น

รีจิสเตอร์ EECON2ไม่ใช่รีจิสเตอร์ที่สามารถติดต่อได้โดยตรง(non-physical register) ดังนั้นเมื่อทำการอ่านข้อมูลจากEECON2 จะอ่านได้ "0" ทั้งหมด EECON2จะถูกใช้เฉพาะในขั้นตอนการเขียนข้อมูลลงในหน่วยความจำข้อมูลอีอีพรอมเท่านั้นซึ่งจะกล่าวถึงต่อไป

3.5.4 การอ่านข้อมูลจากหน่วยความจำข้อมูลอีอีพรอม ,

เริ่มต้นด้วยการเขียนแอดเดรสไปยัง EEADR ก่อนจากนั้นเซตบิต RD ในรีจิสเตอร์ EECON1 ข้อมูลจากหน่วยความจำจะพร้อมให้อ่านในไซเกิลการทำงานถัดไป นั่นหมายความว่าสามารถทำการอ่านข้อมูลจากหน่วยความจำข้อมูลได้ด้วยคำสั่งถัดไป รีจิสเตอร์ EEDATAจะเก็บข้อมูลที่อ่านจากหน่วยความจำข้อมูลอีอีพรอมไว้ จนกว่าผู้ใช้งานจะเรียกข้อมูลนี้ไปใช้งาน หรือมีการเขียนข้อมูลหรืออ่านข้อมูลใหม่เข้ามา

ในโปรแกรมที่ 3.5-1 เป็นโปรแกรมตัวอย่างของการอ่านข้อมูลจากหน่วยความจำข้อมูลอีอีพรอม

3.5.5 การเขียนข้อมูลลงบนหน่วยความจำข้อมูลอีอีพรอม

เริ่มต้นด้วยการกำหนดแอดเดรสที่ต้องการติดต่อไปยัง EEADR และกำหนดข้อมูลที่ต้องการเขียนลงบน EEDATA เซตบิต WRENใน EECON1เพื่ออีนาเบิลการเขียนข้อมูล จากนั้นเข้าสู่กระบวนการเขียนข้อมูล ซึ่งผู้ใช้งานต้องปฏิบัติตามลำดับอย่างเคร่งครัดดังนี้

1. เขียนข้อมูล 55Hไปยัง EECON2
2. เขียนข้อมูล AAHไปยัง EECON2
3. เซตบิตWRใน EECON1

โปรแกรมที่ 3.5-1 โปรแกรมย่อยของการอ่านข้อมูลจากหน่วยความจำข้อมูลอีอีพรอม

```
bcf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 0
movwf   CONFIG_ADDR     :
movwf   EEADR           : เลือกแอดเดรสที่ต้องการอ่าน
bsf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 1
bsf     EECON1,RD       : เซตบิต RD เพื่ออีนาเบิลการอ่านข้อมูล
bcf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 0
movf    EEDATA,W                : W= EEDATA
```

กระบวนการทั้ง 3 ขั้นตอนจะต้องกระทำเหมือนกันในทุกไบต์ของข้อมูลที่ทำการเขียน หลังจากเสร็จสิ้นการเขียนข้อมูลทุกค่า ต้องการเคลียร์บิต WREN ใน EECON1 ขณะเดียวกันบิต

WR จะถูกเคลียร์โดยอัตโนมัติด้วยกระบวนการฮาร์ดแวร์ และบิต EEIFจะเซตเพื่อแจ้งความสมบูรณ์ของการเขียนข้อมูล ซึ่งผู้ใช้งานสามารถนำผลของการเซตบิต EEIF ไปใช้ประโยชน์อื่น ๆ ได้ อย่างไรก็ตามต้องทำการเคลียร์บิต EEIF เสมอ ก่อนที่จะทำการเขียนข้อมูลต่อไป ตัวอย่างโปรแกรมการเขียนข้อมูลในหน่วยความจำข้อมูลอีอีพรอมแสดงในโปรแกรมที่ 3.5-2

โปรแกรมที่ 3.5-2 โปรแกรมย่อยของการเขียนข้อมูลลงในหน่วยความจำข้อมูลอีอีพรอม

```
bsf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 1
bcf     INCON, GIE      : ดิสเอเบิลการอินเตอร์รัปต์
bsf     EECON1, WREN    : อีนาเบิลการเขียนข้อมูล
movlw   0×55            :
movwf   EECON2  : เขียนข้อมูล 55 ลงใน EECON2
movlw   0×aa            :
movwf   EECON2  : เขียนข้อมูล AA ลงใน EECON2
bcf     EECON1,WR       : เซตบิต WR เพื่อเริ่มการเขียน
```

โปรแกรมที่ 3.5-3 โปรแกรมย่อยของการตรวจสอบการเขียนข้อมูลลงในหน่วยความจำข้อมูลอีอีพรอม

```
bcf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 0
movf    EEDATA,W        :
bsf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 1
bsf     EECON1,RD       : อ่านค่าที่ทำการเขียนก่อนหน้านี้
bcf     STATUS, RPO     : เลือกรีจิสเตอร์แบงก์ 1
subwf   EEDAT,W         : เปรียบเทียบกับค่าต้นฉบับ
bftss   STATUS, 7       :
goto    WRITE_ERR       : ถ้าหากไม่เท่ากัน แจ้งการเขียนผิดพลาด แต่ถ้าหากเท่ากันแสดงว่า
                          การเขียนข้อมูลถูกต้องให้กระทำคำสั่งถัดไป
```

3.5.6 การตรวจสอบการเขียนข้อมูล

การเขียนข้อมูลที่ดีควรต้องมีการตรวจสอบข้อมูลที่เขียนเรียบร้อยแล้วกับข้อมูลต้นฉบับว่าถูกต้องหรือไม่ โปรแกรมที่ 3.5-3 เป็นโปรแกรมตัวอย่างที่เขียนเพื่อตรวจสอบการเขียนข้อมูลความผิดพลาดที่มักจะเกิดขึ้นมีกรณีเดียวคือ เขียนข้อมูล “1” ลงในหน่วยความจำ แต่เมื่ออ่านค่ากลับได้เป็น “0”

ถึงแม้ว่าขนาดของหน่วยความจำข้อมูลอีอีพรอมใน PIC16F84 จะมีเพียง 64 ไบต์ก็ตาม แต่ก็พอเพียงสำหรับเก็บค่าของตัวแปรบางตัวที่มีความสำคัญ หากเกิดไฟดับในขณะที่ระบบยังคง

ทำงานอยู่ ค่าของตัวแปรที่ต้องการเก็บนั้นจะอยู่ในหน่วยความจำส่วนนี้ ไม่สูญหาย และเมื่อมีไฟฟ้าจ่ายกลับเข้ามาในระบบอีกครั้งก็สามารถเรียกข้อมูลส่วนนี้ไปใช้งานได้ต่อไป

3.6 คุณสมบัติพิเศษของซีพียูภายใน PIC16F84A

ซีพียูภายใน PIC16F84A มีคุณสมบัติพิเศษอยู่มากมายสามารถสรุปได้ดังนี้

- สามารถเลือกการกำเนิดสัญญาณนาฬิกาได้ 6 รูปแบบ
- รองรับการรีเซตได้ 3 รูปแบบ
- อินเตอร์รัปต์
- มีวอตช์ด็อกไทเมอร์
- มีโหมดสลีปเพื่อประหยัดพลังงาน
- มีระบบป้องกันก้ารคัดลอกข้อมูล
- สามารถโปรแกรมข้อมูลแบบอนุกรมภายในวงจรได้

3.6.1 การเลือกใช้คุณสมบัติพิเศษ

จะต้องมีการกำหนดค่าลงในหน่วยความจำโปรแกรมตำแหน่ง 2007H ซึ่งเรียกว่า Configuration memory โดยมีรูปแบบดังในภาพที่ 3.6-1 ข้อมูลที่ใช้เลือกคุณสมบัตินี้เรียกว่า configuration word จะมีขนาด 14 บิต

3.6.2 การทำงานในโหมดสลีปหรือเพาเวอร์ดาวน์โหมดสลีป

การเข้าสู่โหมดการทำงานแบบนี้จะเกิดขึ้นหลังจากกระทำคำสั่ง sleep เมื่อเริ่มทำงานวอตช์ด็อกไทเมอร์จะถูกเคลียร์ (แต่ยังทำงานอยู่) บิต PD ในรีจีสเตอร์ STATUS จะถูกเคลียร์ด้วย ในขณะที่บิต TO ใน รีจีสเตอร์ STATUS จะเซต วงจรออสซิลเลเตอร์ภายในหยุดทำงาน ส่วนขาสัญญาณทุกพอร์ตจะคงสถานะเดิมก่อนหน้าที่จะกระทำคำสั่ง sleep

เมื่อกำหนดให้ทำงานในโหมดสลีป ขา MCLR จะต้องได้รับสัญญาณ “1” อยู่ตลอดเวลา เพราะโหมดสลีปคือการกำหนดให้ไมโครคอนโทรลเลอร์ทำงานน้อยที่สุด แต่มิใช่ให้หยุดทำงานแล้วเริ่มใหม่

บิต13	บิต12	บิต11	บิต10	บิต9	บิต8	บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	PWRTE	WDTE	FOSC1	F0SC0

บิต13-4 : CP(Code protection bit) บิตป้องกันข้อมูล

เป็น "1" ไม่มีการป้องกันข้อมูล

เป็น "0" ป้องกันการอ่านและเขียนข้อมูลในหน่วยความจำ

บิต 3 : PWRTE (Power-up timer enable bit) บิตอีนาเบิลเพาเวอร์อัปไทเมอร์

เป็น "1" ดิสเอเบิลเพาเวอร์อัปไทเมอร์

เป็น "0" อีนาเบิลเพาเวอร์อัปไทเมอร์

บิต 2 : WDTE (Watchdog timer enable bit) บิตอีนาเบิลวอตช์ด็อกไทเมอร์

เป็น "1" อีนาเบิลวอตช์ด็อกไทเมอร์

เป็น "0" ดิสเอเบิลวอตช์ด็อกไทเมอร์

บิต 1-0 : FOSC 1-FOSC0 (Oscillator selection bit) บิตเลือกบนิดของวงจรกำเนิดสัญญาณนาฬิกา

11 = เลือกวงจรกำเนิดสัญญาณนาฬิกาแบบ RC

10= เลือกวงจรกำเนิดสัญญาณนาฬิกาแบบ HS (High speed X-tal 4-20 MHz)

01= เลือกวงจรกำเนิดสัญญาณนาฬิกาแบบ XT (X-tal)

00= เลือกวงจรกำเนิดสัญญาณนาฬิกาแบบ LP (Low power X-tal)

ภาพที่ 3.6-1 รายละเอียดของ Configuration word ขนาด 14 บิต

3.6.3 การออกจากโหมดสลีปหรือการเวกอัป (Wake UP)

ไมโครคอนโทรลเลอร์ PIC16F84A จะออกจากโหมดสลีปได้ หากมีเหตุการณ์ 1 ใน 3 นี้เกิดขึ้นคือ

1. การเกิดสัญญาณรีเซตจากภายนอกป้อนเข้าที่ขา MCLR
2. วอตช์ด็อกไทเมอร์เวกอัป (ถ้าหากมีการอีนาเบิลวอตช์ด็อกไทเมอร์ไว้)
3. เกิดอินเตอร์รัปต์ขึ้นจากการป้อนสัญญาณเข้าที่ขา RB0/INT, จากการเปลี่ยนแปลงข้อมูลที่พอร์ต B บิต 7-บิต4 หรือจากการที่เขียนข้อมูลในอีอีพรอมสมบูรณ์

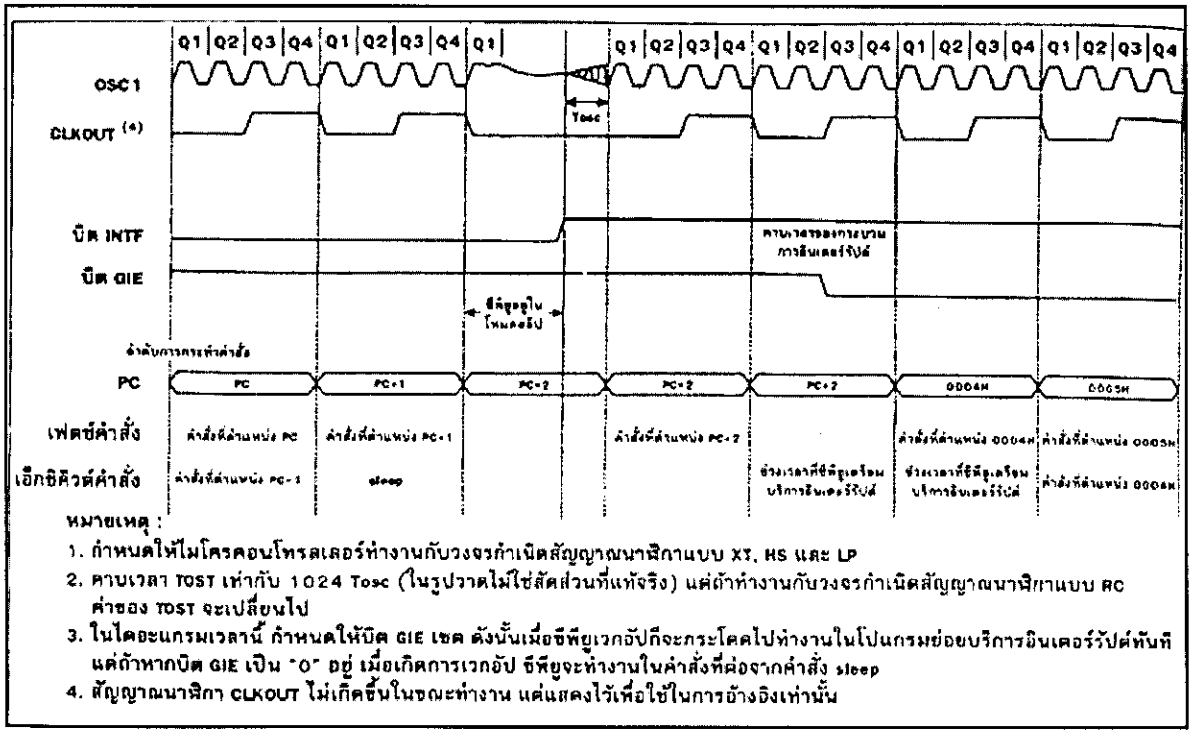
เมื่อไมโครคอนโทรลเลอร์ กระทำคำสั่ง sleep แล้วคำสั่งต่อไปจะถูกเฟตขึ้นมา ถ้าหากคำสั่งต่อไปเป็นคำสั่งที่ทำให้เกิดการอินเตอร์รัปต์ ไมโครคอนโทรลเลอร์จะเวกอัปเพื่อกระทำคำสั่งนั้น ย่างไรก็ตามต้องตรวจสอบสถานะที่บิต GIE ด้วย หากบิต GIE เป็น "0" อยู่ แล้วไมโครคอนโทรลเลอร์เกิดการเวกอัปซีพียูจะทำงานในคำสั่งที่ถัดจาก sleep ต่อไปตามลำดับ แต่ถ้ากำหนดบิต GIE เป็น "1" เมื่อเกิดการเวกอัปไมโครคอนโทรลเลอร์จะกระทำสั่งคำสั่งที่ถัดจาก sleep

แล้วกระโดดไปยังอินเตอร์รัปต์แอดเดรสที่ตำแหน่ง 0004H ดังนั้นคำสั่งที่ควรใช้หลังคำสั่ง sleep คือ nop (no operation)

ในกรณีที่บิต GIE เป็น "0" อยู่ อันเป็นการดิสเอเบิลการอินเตอร์รัปต์ทั้งหมดของ PIC16F84A แต่บิต INTE และ INTF เซตเป็น "1" อยู่ หากเกิดการอินเตอร์รัปต์ก่อนการเอ็กซิคิวต์คำสั่ง sleep คำสั่ง sleep จะมีค่าเท่ากับคำสั่ง nop ในขณะที่วอตช์ด็อกไทเมอร์จะไม่ถูกเคลียร์ บิต TO ไม่เซต และบิต PD ไม่เคลียร์

ถ้าหากการอินเตอร์รัปต์เกิดขึ้นระหว่างหรือหลังจากเอ็กซิคิวต์คำสั่ง sleep ซีพียูจะเวกอัปจากโหมดสลีปทันที ทั้งนี้ซีพียูจะกระทำคำสั่ง sleep จนเสร็จสิ้นก่อน ในขณะที่วอตช์ด็อกไทเมอร์จะเคลียร์ค่าทั้งหมด บิต TO จะเซต และ PD จะได้รับการเคลียร์

ดังนั้นก่อนเข้าสู่โหมดสลีปจะมีการตรวจสอบสถานะของบิต INTE และ INTF ก่อนและเพื่อให้แน่ใจว่าวอรช์ด็อกไทเมอรีรับการเคลียร์จริง ๆ ควรเอ็กซิคิวต์คำสั่ง clrwdt ก่อนที่จะทำการเอ็กซิคิวต์คำสั่ง sleep



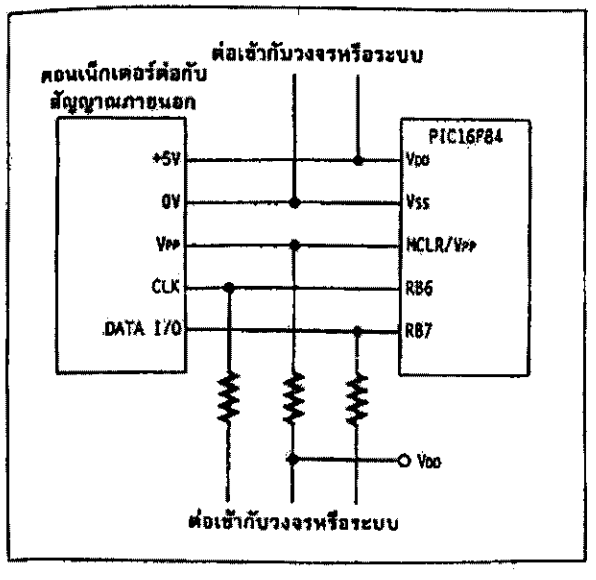
ภาพที่ 3.6-2 เป็นไดอะแกรมเวลาแสดงการเวกอัปของ PIC16F84A อันเนื่องมาจากการอนเตอร์รัปต์

3.6.4 การโปรแกรมแบบอนุกรมภายในวงจร

PIC16F84A สามารถที่จะโปรแกรมข้อมูลลงบนตัวมันได้แม้ว่าจะอยู่ในวงจรก็ตาม โดยใช้ขาสัญญาณ 2 ขา สำหรับสัญญาณนาฬิกาและข้อมูล อีก 3 ขาสัญญาณสำหรับไฟเลี้ยง, กราวด์ และแรงดันไฟฟ้าสำหรับโปรแกรม ดังแสดงในภาพที่ 3.6-3

PIC16F84A ถูกกำหนดให้ทำงานในโหมดโปรแกรม (program/verify mode) โดยการทำให้ขา RB6 และ RB7 ได้รับลอจิก "0" ขา RB6 จะถูกใช้เป็นขาสำหรับรับสัญญาณนาฬิกาส่วนขา RB7 รับข้อมูลที่ต้องการโปรแกรม ในขณะที่ขา MCLR ใช้รับแรงดันสำหรับการโปรแกรม

หลังจากการรีเซต โปรแกรมเคาน์เตอร์ของ PIC16F84A จะถูกกำหนดให้ไปที่ตำแหน่ง 00H คำสั่งขนาด 6 บิต และข้อมูลขนาด 14 บิต จะถูกส่งเข้าไปในหน่วยความจำโปรแกรมภายใน PIC16F84A โดยการใช้คำสั่งเกี่ยวดับการถ่ายทอดข้อมูล



ภาพที่ 3.6-3 วงจรของการโปรแกรม PIC16F84A แบบอนุกรมภายในวงจร

3.6.5 วอตช์ด็อกไทเมอร์

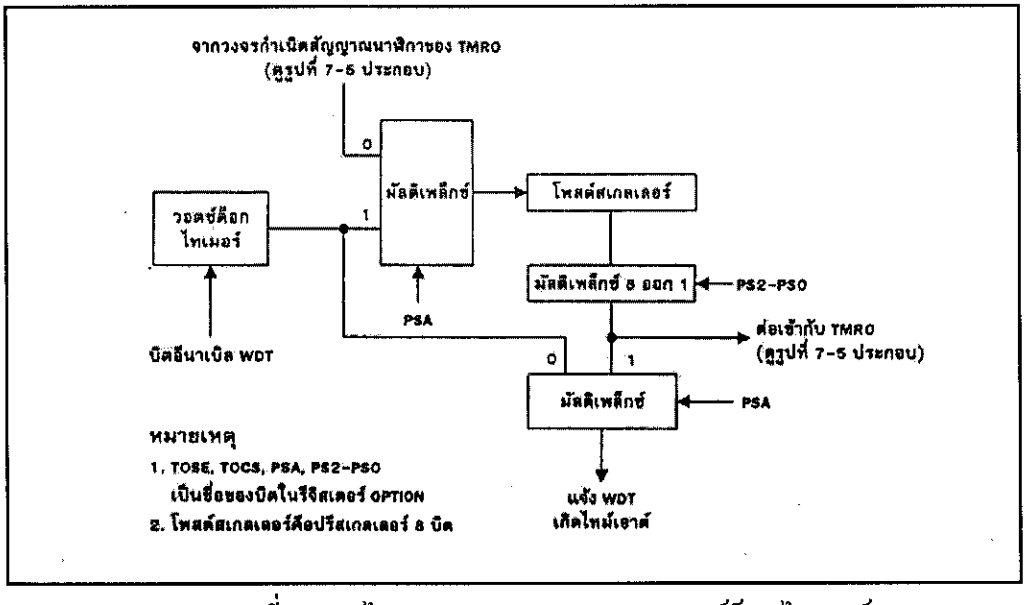
เป็นวงจรกำเนิดสัญญาณอย่างต่อเนื่องหรือที่เรียกว่า ฟรีรันนิ่งออสซิลเลเตอร์ (free running oscillator) แบบ RC บรรจุอยู่ภายในไมโครคอนโทรลเลอร์ PIC16F84A ในการทำงานจึงไม่ต้องอาศัยอุปกรณ์ภายนอกแต่อย่างใด มีบล็อกไดอะแกรมแสดงกรทำงานตามรูปที่ 3.6-4 วงจรออสซิลเลเตอร์ในส่วนของวอตช์ด็อกไทเมอร์นี้แยกการทำงานออกจากวงจรออสซิลเลเตอร์ที่ขา OSC1/CLKIN อย่างสิ้นเชิง ดังนั้นวอตช์ดือกไทเมอร์ หรือ WDT จะยังคงทำงานต่อเนื่องไป แม้ว่า

ไม่มีการป้อนสัญญาณนาฬิกาเข้ามาที่ขา OSC1/CLKIN และ OSC2/CLKOUT ก็ตามโดย WDT ยังคงทำงานตราบเท่าที่ยังมีการจ่ายไฟเลี้ยงให้ไมโครคอนโทรลเลอร์

การอีนาเบิลหรือดิสเอเบิลวอตช์ด็อกไทเมอร์สามารถกระทำได้ โดยการกำหนดค่าที่บิต WDTE ใน Configuration word ถ้ากำหนดค่าเป็น "0" จะเป็นการดิสเอเบิล WDT ถ้ากำหนดค่า "1" จะเป็นการอีนาเบิล WDT

3.6.6 คาบเวลาของวอตช์ด็อกไทเมอร์

WDT จะมีคาบเวลาไทเอาท์เท่ากับ 18 มิลลิวินาทีในกรณีที่ไม่มีการใช้ปรีสเกลเลอร์ คาบเวลานร้อาจคลาดเคลื่อนได้ตามอุณหภูมิใช้งานและค่าของแรงดันไฟเลี้ยง ถ้าต้องการให้คาบเวลาของ WDT ยาวนานขึ้น จะต้องนำปรีสเกลเลอร์มาช่วยซึ่งสามารถกำหนดได้ด้วยกระบวนการซอฟต์แวร์โดยการกำหนดค่าลงในรีจิสเตอร์ OPTION เมื่อนำปรีสเกลเลอร์มาช่วยเพิ่มคาบเวลา จะสามารถเพิ่มได้สูงถึง 128 เท่า หรือประมาณ 2.3 วินาที



ภาพที่ 3.6-4 ไดอะแกรมการทำงานของวอตช์ด็อกไทเมอร์

คำสั่ง clrwdt และ sleep เป็นคำสั่งที่ใช้ในการเคลียร์ค่าของ WDT เมื่อกระทำคำสั่งนี้ ค่าของ WDT จะกลายเป็น ศูนย์โดยที่ไม่เกิดการไทม์เอาต์และกำเนิดสัญญาณเพื่อทำการรีเซตระบบ

บิต TO ในรีจิสเตอร์ STATUS จะถูกเคลียร์เมื่อ WDT เกิดการไทม์เอาต์

ในตารางที่ 3.6-1 เป็นการสรุปรีจิสเตอร์ที่เกี่ยวข้องกับวอตช์ด็อกไทเมอร์ทั้งหมด

แอดเดรส	ชื่อรีจิสเตอร์	บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0	ค่าที่เกิดขึ้นหากเกิดเพาเวอร์ออนรีเซต	ค่าที่เกิดขึ้นหากเกิดการรีเซตในแบบอื่น
2007H	configuration bit	CP	CP	CP	CP	PWRTE	WDTE	POSC1	POSC0	ดูรายละเอียดในหัวข้อ configuration bit	
81H	OPTION	RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

ตารางที่ 3.6-1 ตารางสรุปรีจิสเตอร์ที่เกี่ยวข้องกับวอตช์ด็อกไทเมอร์

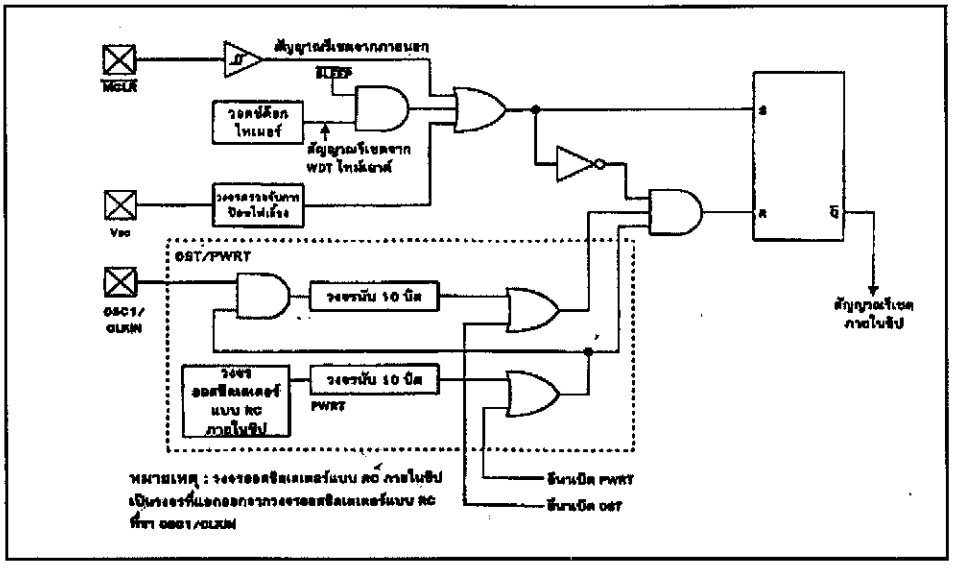
3.6.7 การรีเซตของ PIC16F84A

กระบวนการรีเซตที่เกิดขึ้นกับ ไมโครคอนโทรลเลอร์ PIC16F84A มีด้วยกัน 5 แบบ ดังนี้

1. เพาเวอร์ออนรีเซท (power-On Reset : POR)
2. เกิดการรีเซตที่ขา MCLR ในขณะทำงานปกติ
3. เกิดการรีเซตที่ขา MCLR ในขณะอยู่ในโหมดสลีป
4. การรีเซตอันเนื่องมาจากวอตช์ด็อกไทเมอร์ในขณะทำงานปกติ
5. การรีเซตอันเนื่องมาจากวอตช์ด็อกไทเมอร์เวกอัปในขณะอยู่ในโหมดสลีป

ในภาพที่ 3.6-5 แสดงบล็อกไดอะแกรมการทำงานของวงจรรีเซตภายในตัวไมโครคอนโทรลเลอร์ PIC16F84A ที่ขา MCLR จะมีวงจรอินเวอต์เตอร์ชมิตต์ทริกเกอร์ต่ออยู่ เพื่อช่วยกำจัดสัญญาณรบกวนหรือสัญญาณพัลส์ขนาดเล็ก ๆ ที่อาจเข้ามาที่ขา MCLR นี้อันจะทำให้ PIC16F84A เกิดการรีเซตทั้ง ๆ ที่ไม่ได้มีการรับสัญญาณเพื่อทำการรีเซตแต่อย่างใด

เมื่อเกิดการรีเซตจะส่งผลต่อค่าของรีจิสเตอร์ภายในตัว PIC16F84A โดยจะเกิดการเคลียร์ค่าเป็น 0 แต่ทั้งนี้ก็ต้องขึ้นอยู่กับว่าสาเหตุของการีเซตคืออะไร เพราะจะมีรีจีสเตอร์บางตัวที่ยังคงค่าเดิมอยู่แม้ว่าจะเกิดการรีเซตก็ตาม ในตารางที่ 3.6-2 แสดงถึงผลกระบบต่อโปรแกรมเคาน์เตอร์ (PC) และรีจิสเตอร์ STATUS เมื่อ PIC16F84A เกิดการรีเซตในสาเหตุต่าง ๆ ในขณะที่ตารางที่ 3.6-3 เป็นการสรุปผลกระทบต่อรีจิสเตอร์ทั้งหมดของ PIC16F84A เมื่อเกิดการรีเซตในลักษณะต่าง ๆ จะเห็นได้ว่าการรีเซตเนื่องจากการจ่ายไฟหรือเพาเวอร์ออนรีเซตจะเป็นการรีเซตที่มีผลสูงสุด ถัดมาเป็นการรีเซตอันเนื่องมาจากการป้อนสัญญาณเข้าที่ขา MCLR และสุดท้ายการรีเซตที่มีสาเหตุมาการวอตช์ด็อกไทเมอร์ จะส่งผลต่อรีจิสเตอร์เพียงไม่กี่ตัว



ภาพที่ 3.6-5 ไดอะแกรมการทำงานของวงจรรีเซตใน PIC16F84A

เงื่อนไข	ค่าของ PC	ค่าของรีจิสเตอร์ STATUS
เพาเวอร์ออนรีเซต	0x00	0001 1xxx
เกิดการรีเซตที่ MCLR ในโหมดปกติ	0x00	000u uuuu
เกิดการรีเซตที่ MCLR ในโหมดสลีป	0x00	0001 0uuuu
WDT เกิดการรีเซตในขณะทำงานปกติ	0x00	0000 1uuu
WDT เกิดการเวกอัป	PC+1	uuu0 0uuu
เกิดอินเตอร์รัปต์ออกจากโหมดสลีป	PC+1(1)	uuu1 0uuu

หมายเหตุ x หมายถึง ไม่ทราบค่า ,u หมายถึง ไม่มีการเปลี่ยนแปลงข้อมูลที่บิตนั้น

(1) เมื่อเกิดการอินเตอร์รัปต์ บิต GIE จะเซต ค่าของ PC=0x004

ตารางที่ 3.6-2 แสดงผลกระทบที่มีต่อรีจิสเตอร์ PC และ STATUS เมื่อเกิดการรีเซต

3.6.8 พาเวอร์ออนรีเซต (POR)

การรีเซตแบบนี้จะเกิดขึ้นเมื่อเกิดการตรวจจับแรงดันที่ ขา VDD ได้ในระหว่าง 1.2-1.7 V ในขณะที่ขา MCLR จะต่ออยู่กับไฟเลี้ยงโดยตรงหรือผ่านตัวต้านทาน ทำให้ขา R ของ RS ฟลิบฟลอปภายใน PIC16F84A ได้รับลอจิก "1" ส่งผลให้ฟลิปฟลอปเกิดการรีเซต การรีเซตแบบนี้จะไม่เกิดขึ้นหากไม่มีการจ่ายไฟเลี้ยงให้แก่ PIC16F84A ในภาพที่7-6 เป็นการต่อวงจรเพื่อทำการรีเซตตัว PIC16F84A เมื่อเริ่มจ่ายไฟเลี้ยงหรือเพาเวอร์ออนรีเซต

3.6.9 เพาเวอร์อัปไทเมอร์ (PWRT)

เมื่อเกิดการเพาเวอร์ออนรีเซต จะทำให้ไทเมอร์ตัวหนึ่งภายใน PIC16F84A ทำงาน ไทเมอร์ตัวนั้นคือ *เพาเวอร์อัปไทเมอร์ (power-up Timer : PWRT)* เพาเวอร์อัปไทเมอร์จะทำการกำหนดคาบเวลาไทม์เอาต์ (Time-Out : TRWRT) ไว้ที่ 72 มิลลิวินาที หลังจากที่เกิดเพาเวอร์ออนรีเซต

คาบเวลาของเพาเวอร์อัปไทเมอร์ได้รับการกำหนดโดยวงจรออสซิลเลเตอร์ RC ภายในตัวไมโครคอนโทรลเลอร์ โดยไมโครคอนโทรรลเลอร์จะคำรงสถานการณ์รีเซตไว้จนกว่าเพาเวอร์อัปไทเมอร์จะแอกตีฟ

การอีนาเบิลและดีสเอเบิลเพาเวอร์อัปไทเมอร์สามารถทำได้โดยการกำหนดที่บิต PWRTE ใน Configuration Bit

3.6.10 ออสซิลเลเตอร์ สตาร์อัปไทเมอร์ (OST)

OST ทำหน้าที่กำหนดสัญญาณเพื่อหน่วงเวลาขนาด 1,024 ไซเคิล หลังจากสิ้นสุดการทำงานของ PWRT ทั้งนี้เพื่อให้แน่ใจว่าวงจรกำไดสัญญาณนาฬิกาทั้งแบบคริสคอลและเซรามิกเรโซเนเตอร์พร้อมจะเริ่มทำงานแล้ว

OST จะทำงานเมื่อกำหนดให้ PIC16F84A ทำงานกับวงจรออสซิลเลเตอร์แบบ XT, LP และ HS เริ่มทำงานเมื่อเกิดเพาเวอร์ออนรีเซตหรือเกิดการเวกอัปจากโหมดสลีป

ชื่อรีจิสเตอร์	แอดเดรส	ค่าที่เกิดขึ้นหากเกิดเพาเวอร์ออนรีเซต	ค่าที่เกิดขึ้นหากเกิดการรีเซตที่ MCLR ในทุกโหมดการทำงานและ WDT ในสภาวะปกติ	ค่าที่เกิดขึ้นกากเกิดการรีเซตเนื่องจากการเวกอัปทั้งจากอินเตอร์รัปและ WDT เกิดไทเอาท์
W	-	ไม่ทราบค่า	ไม่เปลี่ยนแปลง	ไม่เปลี่ยนแปลง
INDF	0X000	----------	---- ----	---- ----
TMR0	0X001	ไม่ทราบค่า	ไม่เปลี่ยนแปลง	ไม่เปลี่ยนแปลง
PCL	0X002	0000 0000	0000 0000	PC+1[2]
STATUS	0X003	0001 1xxxx	000q quuu	uuuq quuu
FSR	X0004	---x xxxx	---u uuuu	---u uuuu
PORTA	0X005	ไม่ทราบค่า	ไม่เปลี่ยนแปลง	ไม่เปลี่ยนแปลง
PORTB	0X006	ไม่ทราบค่า	ไม่เปลี่ยนแปลง	ไม่เปลี่ยนแปลง
EEDATA	0X008	ไม่ทราบค่า	ไม่เปลี่ยนแปลง	ไม่เปลี่ยนแปลง
EEADR	0X009	ไม่ทราบค่า	ไม่เปลี่ยนแปลง	ไม่เปลี่ยนแปลง
PCLATH	0X00a	---0 0000	---0 0000	---u uuuu
INTCON	0X00b	0000 000ป	0000 000u	uuuu uuuu[1]
INDF	0x080	---- ----	---- ----	---- ----
OPTION	0x081	1111 1111	1111 1111	uuuu uuuu

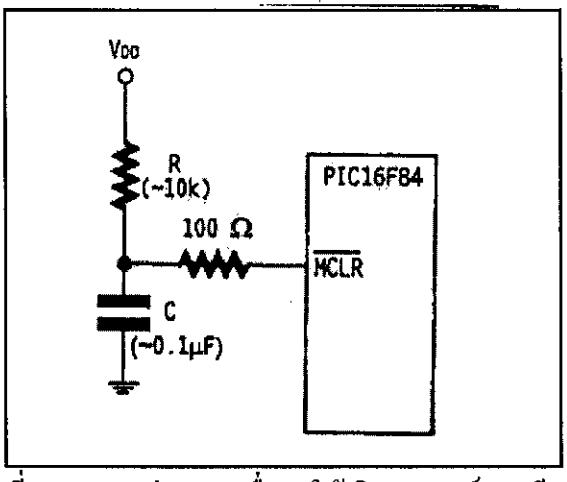
PCL[2]	0x082	0000 0000	0000 0000	PC+1
STATUS	0x083	0001 1xxxx	000q quuu	uuuq quuu
FSR	0x084	ไม่ทราบค่า	ไม่เปลี่ยนแปลง	ไม่เปลี่ยนแปลง
TRISA	0x085	---1 1111	---1 1111	---u uuuu
TRISB	0x086	1111 1111	1111 1111	uuuu uuuu
EECON1E	0x088	---0 x000	---0 q000	---0 q000
ECON2	0x089	---- ----	---- ----	---- ----
PCLATH	0x08a	---0 0000	---0 0000	---u uuuu
INTCON	0x08b	0000 000x	0000 000u	uuuu uuuu[1]

หมายเหตุ x หมายถึง ไม่ทราบค่า , u หมายถึง ข้อมูลที่บิตนั้นไม่เปลี่ยนแปลง ,q หมายถึง ค่าจะขึ้นอยู่กับเงื่อนไขของรีจิสเตอร์แต่ละตัว

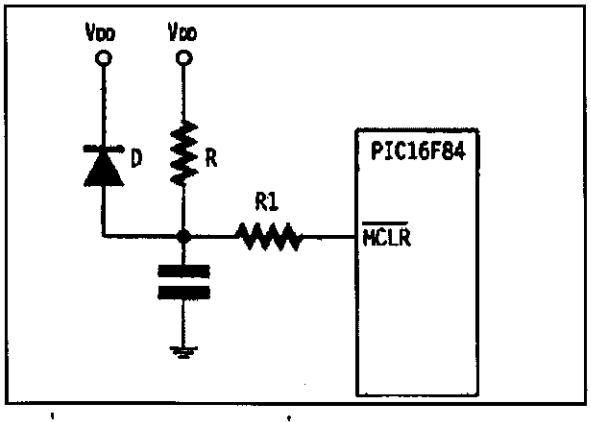
(1) จะมีบิตใดบิตหนึ่งหรือมากกว่าเกิดการเปลี่ยนแปลงเพื่อทำให้เกิดการเวกอัป

(2) เมื่อเกิดการเวกอัปเนื่องจากการอินเตอร์รัปต์ บิต GIE จะเซต PC=0x004

ตารางที่ 3.6-3 แสดงผลกระทบที่มีต่อรีจิสเตอร์ไฟล์ทั้งหมดเมื่อเกิดการรีเซต



ภาพที่ 3.6-6 การต่อวงจรเพื่อทำให้เกิดเพาเวอร์ออนรีเซต



ภาพที่ 3.6-7 การต่อวงจรเพื่อทำการรีเซตจากภายนอก

ในกรณีที่ ขา VDD เกิดการเพิ่มขึ้นของแรงดันช้า ๆ ทำให้ TPWRT และ TOST จบลงไปก่อนที่ขา VDD จะได้รับแรงดันไฟเลี้ยงสูงสุด การต่อวงจรเพื่อทำการรีเซตจากภายนอกดังในภาพที่ 3.6-7 จึงเป็นสิ่งที่จำเป็นต้องมี

3.6.11 บิต TO และบิต PD

เมื่อเกิดการจ่ายไฟเลี้ยงเพาเวอร์อัปจะเกิดกระบวนการเพาเวอร์ออนรีเซต (POR) ขึ้นก่อนเป็นบำดับแรก หลังจากนั้น PWRT จะเริ่มทำงาน ทำการหน่วงเวลาไป 72 มิลลิวินาที จากนั้น OST จะเริ่มทำงาน รวมเวลาที่หน่วงไปทั้งสิ้น 72 มิลลิวินาที + 1,024 Tosc (Tosc คือคาบเวลาของสัญญาณนาฬิกา) แต่ถ้าหากวงจรกำเนิดสัญญาณนาฬิกาเป็นแบบ RC OST จะถูกดิสเอเบิลทำให้เกิดการหน่วงเวลาเพียง 72 มิลลิวินาที และถ้าหากดิสเอเบิล PWRT ด้วยจะไม่มีการหน่วงเวลาใด ๆ ทั้งสิ้น ดังแสดงรายละเอียดโดยสรุปในตารางที่ 3.6-4

TO	PD	**สิ่งที่เกิดขึ้นในระบบ**
1	1	เพาเวอร์ออน-รีเซต
0	x	เกิดความผิดพลาด
x	0	เกิดความผิดพลาด
0	1	เกิดการรีเซตที่ WDT ในโหมดปกติ
0	0	WDT เกิดการเวกอัป
1	1	เกิดการรีเซตที่ MCLR ในโหมดปกติ
1	0	เกิดการรีเซตที่ MCLR ในโหมดสลีป หรือเกิดการเวกอัปอันเนื่องมาจากการอินเตอร์รัปต์ขณะทำงานในโหมดสลีป

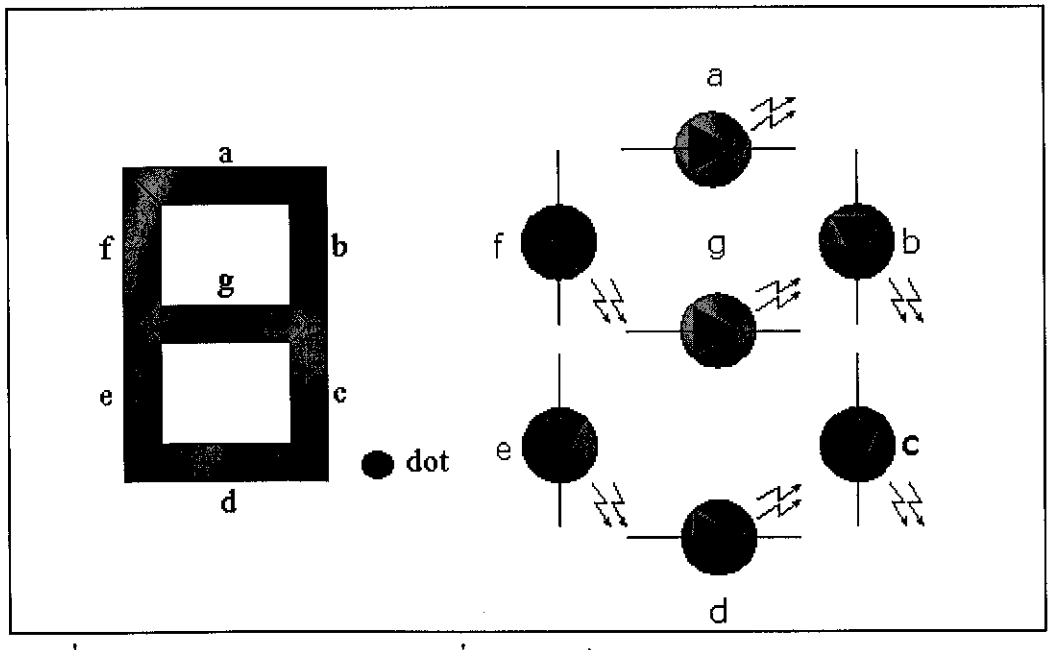
ตารางที่ 3.6-4 ผลที่เกิดขึ้นจากการกำหนดค่าให้แก่บิต TO และ PD

บทที่ 4
การใช้งาน PIC16F84A ขับ LED ตัวเลข 7 ส่วน

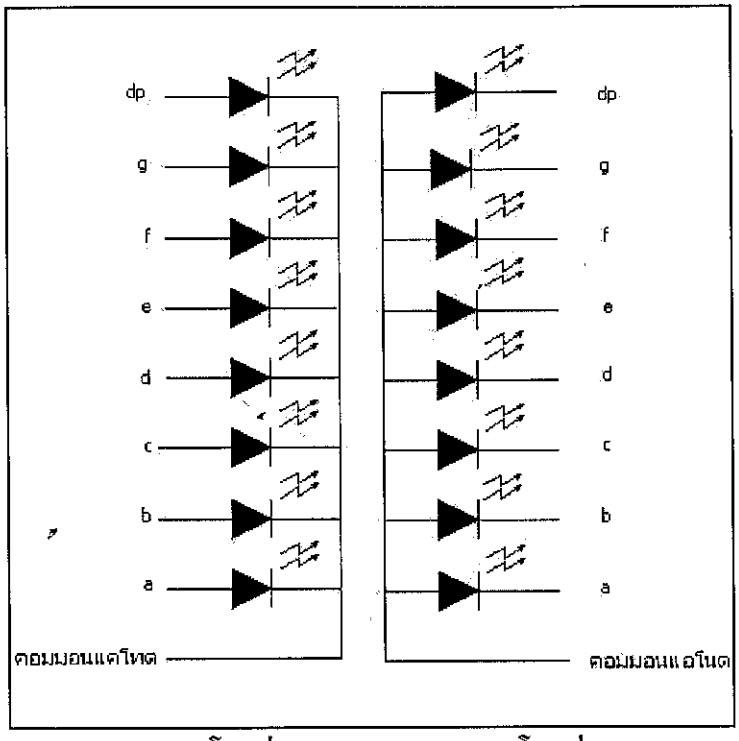
4.1 ความรู้เบื้องต้นเกี่ยวกับ LED ตัวเลข 7 ส่วน

LED ตัวเลข 7 ส่วนประกอบขึ้นจาก LED จำนวน 7 ตัวที่บรรจุอยู่ในตัวถังเดียวกันและได้รับการจัดเรียงเป็นรูตัวเลข LED แต่ละตัวจะถูกเรียกว่า ส่วน หรือ เซกเมนต์ (segment) แต่ละส่วนหรือเซกเมนต์มีชื่อเรียกแตกต่างกันตามตำแหน่งที่ได้รับการจัดวางคือ a, b, c, d, e, f และ g ดังแสดงในภาพที่ 4-1 ส่วน dp เป็น LED อีก 1 ตัวที่บรรจุอยู่ใน LED ตัวเลข 7 ส่วนนี้ใช้เป็นตัวแสดงจุดทศนิยมในกรณีที่มีการแสดงผลในลักษณะเลขที่มีทศนิยม

LED ทุกตัวที่บรรจุอยู่ใน LED ตัวเลข 7 ส่วนนี้มีขาต่อร่วมกันซึ่งก็มีทั้งแบบต่อขาแคโทดร่วมกันเรียกว่า *แบบแคโทดร่วม (common cathode)* และแบบต่อขาแอโหดร่วมกันเรียกว่า *แบบแอโนดร่วม (common anode)* การขับให้ LED ตัวเลข 7 ส่วนแบบแคโทดร่วมสว่างจะต้องจ่ายไฟลบเข้าที่ขาร่วม แล้วจ่ายไฟบวกเข้าที่ขาแอโนดซึ่งก็คือขาของแต่ละเซกเมนต์นั่นเอง ดังแสดงในภาพที่ 4-2 (ก) ในขณะที่ LED ตัวเลข 7 ส่วนแบบแอโนดร่วมจะต้องจ่ายไฟบวกเข้าที่ขาร่วม แล้วจ่ายไฟลบเข้าที่ขาแคโทด ซึ่งเป็นขาของแต่ละเซกเมนต์ดังแสดงในภาพที่ 4-2 (ข)



ภาพที่ 4-1 แสดงรูปร่างและการกำหนดชื่อเซกเมนต์ต่างๆ ของ LED ตัวเลข 7 ส่วน



(ก) แคโทดร่วม (ข) แอโนดร่วม

ภาพที่ 4-2 วงจรภายในของ LED ตัวเลข 7 ส่วนทั้งแบบแคโทดร่วมและแอโนดร่วม

4.2 การขับ LED ตัวเลข 7 ส่วนแบบหลักเดี่ยว

PIC16F84A สามารถขับ LED ได้โดยต้องมีตัวต้านทานจำกัดกระแส ในกรณีที่ใช้ไฟเลี้ยง +5V เมื่อนำมาขับ LED ตัวเลข 7 ส่วนก็เช่นกันต้องมีการต่อตัวต้านทานจำกัดกระแสให้แก่ LED ในทุกเซกเมนต์ การกำหนดให้ LED ตัวเลข 7 ส่วนแสดงข้อมูลเป็นตัวเลขหรือเป็นสัญลักษณ์ใด ๆ ก็ตาม ต้องมีการกำหนดรูปแบบการแสดงผลของเซกเมนต์ต่างๆ ด้วยข้อมูลแต่ละบิตของไมโครคอนโทรลเลอร์ แล้วใช้วิธีการเปิดตารางหรือ look up table ดังแสดงตัวอย่างตารางข้อมูลของการแสดงผลตัวเลขฐานสิบหกของ LED ตัวเลข 7 ส่วนในตารางที่ 4-1

การเขียนโปรแกรมเพื่อขับ LED ตัวเลข 7 ส่วนควรใช้การอ้างถึงแบบสัมพัทธ์ (relative addressing) แล้วใช้โปรแกรมหน่วงเวลาเพื่อให้ LED ในเซกเมนต์ที่ถูกสั่งให้ทำงานนั้นติดสว่างนานพอให้ผู้ใช้งานเห็นข้อมูลที่นำมาแสดงผลที่ LED ตัวเลข 7 ส่วนนั้น

ข้อมูลดิจิตอลเอาต์พุตสำหรับขับ LED ตัวเลข 7 ส่วน								ค่าเลขฐานสิบหกที่ใช้กับ PIC16F84	ค่าตัวเลขที่แสดงที่ 7 เซกเมนต์
D7	D6	D5	D4	D3	D2	D1	D0		
0	0	1	1	1	1	1	1	0x3F	0
0	0	0	0	0	1	1	0	0x06	1
0	1	0	1	1	0	1	1	0x5B	2
0	1	0	0	1	1	1	1	0x4F	3
0	1	1	0	0	1	1	0	0x66	4
0	1	1	0	1	1	0	1	0x6D	5
0	1	1	1	1	1	0	1	0x7D	6
0	0	0	0	0	1	1	1	0x07	7
0	1	1	1	1	1	1	1	0x7F	8
0	1	1	0	1	1	1	1	0x6F	9
0	1	1	1	0	1	1	1	0x77	A
0	1	1	1	1	1	0	0	0x7C	b
0	0	1	1	1	0	0	1	0x39	C
0	1	0	1	1	1	1	0	0x5E	d
0	1	1	1	1	0	0	1	0x79	E
0	1	1	1	0	0	0	1	0x71	F
1	1	1	1	1	1	1	1	0xFF	.

ตารางที่ 4-1 ตารางข้อมูลของการแสดงผลตัวเลข 0-F ของ LED ตัวเลข 7 ส่วน

4.3 การขับ LED ตัวเลข 7 ส่วนแบบมัลติเพล็กซ์

ในกรณีที่ต้องการให้ PIC16F84A ขับ LED ตัวเลข 7 ส่วนมากกว่า 1 หลัก จะต้องใช้เทคนิคที่เรียกว่า การแสดงผลแบบมัลติเพล็กซ์ (multiplex) อันเป็นวิธีการขับให้ LED สว่างที่ละหลักด้วยอัตราเร็วที่ตาของมนุษย์ไม่สามารถตรวจจับได้ทัน จึงดูเหมือนว่า LED ตัวเลข 7 ส่วนทุกหลักติดสว่างในเวลาเดียวกัน

การแสดงผลแบบมัลติเพล็กซ์นี้มีประโยชน์หลายประการดังนี้

1. ช่วยลดพลังงานไฟฟ้าที่ใช้ ทำให้ขนาดของแหล่งจ่ายกำลังงานไฟฟ้าเล็กลง ส่งผลให้ขนาดโดยรวมของระบบเล็กลงด้วย

2. ช่วยให้ PIC16F84A ที่มีจำนวนพอร์ตจำกัดสามารถขับ LED ตัวเลข 7 ส่วนได้มากกว่า 1 หลัก โดย PIC16F84A สามารถขับ LED ตัวเลข 7 ส่วนได้สูงสุดถึง 5 หลักในกรณีที่ใช้พอร์ตทั้งหมดของ PIC16F84A ต่อเข้ากับ LED ตัวเลข 7 ส่วน

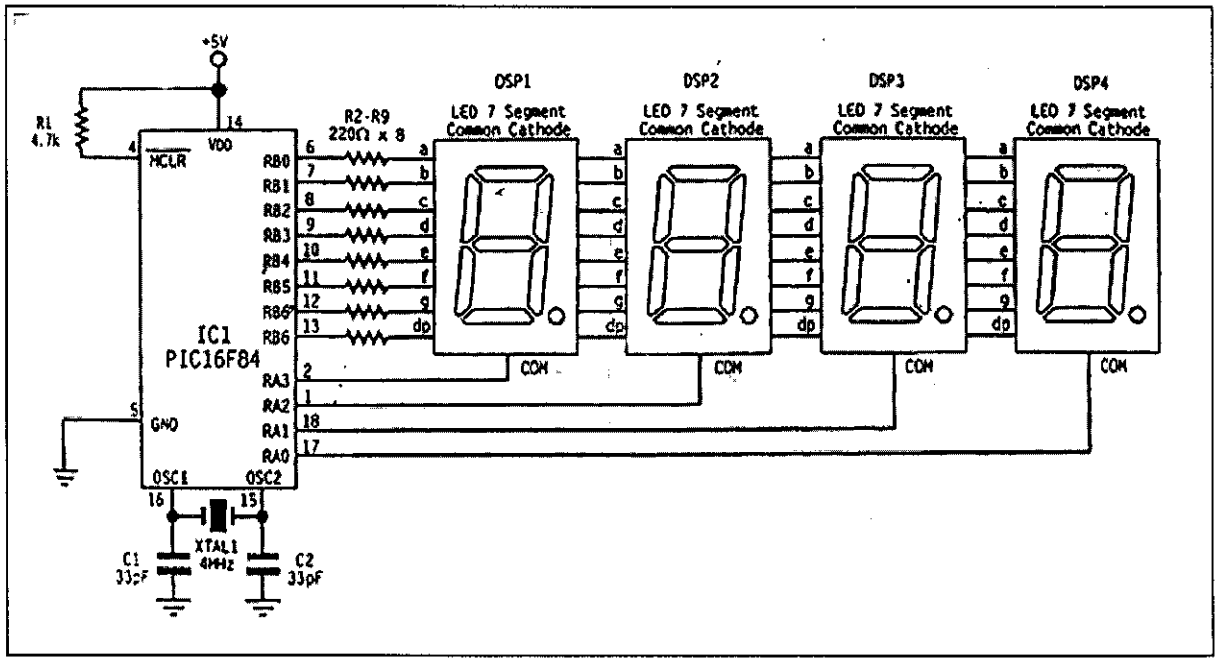
3. ลดจำนวนตัวต้านทานที่ใช้ในการจำกัดกระแสของ LED ในแต่ละเซกเมนต์ ยกตัวอย่าง LED ตัวเลข 7 ส่วนหนึ่งหลักต้องใช้ตัวต้านทานจำกัดกระแส 8 ตัว ถ้าหากขับ LED ตัวเลข 7ส่วน 4 หลักโดยตรง ต้องใช้ตัวต้านทานมากถึง 32 ตัว ในขณะที่หากใช้วิธีการแสดงผลแบบมัลติเพล็กซ์ ยังคงใช้ตัวต้านทานเพื่อจำกัดกระแสให้ LED ในแต่ละเซกเมนต์เพียง 8 ตัวไม่ว่าจะขับ LED ตัวเลข 7 ส่วนก็หลักก็ตาม

การขับ LED ตัวเลข 7 ส่วนแบบมัลติเพล็กซ์จะทำการต่อขาของแต่ละเซกเมนต์ร่วมกันคือ เซกเมนต์ a ของทุกหลักจะต่อถึงกันไล่เรียงไปจนถึงเซกเมนต์ g ในบางงานที่ต้องใช้จุด dp ก็ต้องต่อขาของจุด dp รวมกันด้วย การควบคุมให้ LED ตัวเลข 7 ส่วนหักใดติดสว่าง ทำได้โดยการจ่ายไฟเข้าที่ขาร่วมของ LED ตัวเลข 7 ส่วนที่เป็นแบบแคโทดร่วม หากต้องการให้ LED ตัวเลข 7 ส่วนหลักที่ 3 ติดสว่างก็ต่อขาร่วมของหลักที่ 3 ลงกราวด์ หรือจ่ายไฟลบ LED ตัวเลข 7 ส่วนหลักที่ 3 ก็จะติดสว่างตามข้อมูลที่ส่งเข้ามายังขาของแต่ละเซกเมนต์

การใช้ PIC16F84A เข้ามาควบคุมการแสดงผลในลักษณะนี้จึงเป็นการเข้ามาควบคุมการจ่ายไฟเข้าที่ขาร่วมของ LED ตัวเลข 7 ส่วนแต่ละหลักนั่นเอง โดย PIC16F84A จะจ่ายไฟให้แก่ขาร่วมของ LED ตัวเลข 7 ส่วนทีละหลักไล่ไปตามลำดับด้วยความเร็วสูง ส่วนขาของแต่ละเซกเมนต์จะถูกต่อเข้ากับพอร์ตเอาท์พุตของ PIC16F84A ดังแสดงวงจรในภาพที่ 4-3โดย LED ตัวเลข 7 ส่วนที่ใช้เป็นแบบแคโทดร่วม

กระบวนการเริ่มต้นโดย PIC16F84A ส่งข้อมูลออกไปยังขาพอร์ตที่ต่ออยู่กับเซกเมนต์ a – g และ dp (ในกรณีที่ต้องการใช้ dp) ของ LED ตัวเลข 7 ส่วนก่อนจากนั้นจึงส่งข้อมูล “0” ไปยังขาร่วมของ LED ตัวเลข 7 ส่วนในหลักที่ต้องการให้แสดงผล ยกตัวอย่างต้องการแสดงตัวเลข 1234 PIC16F84A ต้องส่งข้อมูลของเลข 1 ไปก่อนแล้วจึงส่งข้อมูล “0” ไปยังขาร่วมของ DSP4 จากนั้นจึงส่งข้อมูลเลข 2 แล้วส่งข้อมูล “0” ไปยังขาร่วมของ DSP3 ทยอยไล่ไปตามลำดับด้วยความเร็วสูงในอัตราที่ตาของมนุษย์ไม่สามารถสังเกตเห็นความเปลี่ยนแปลงดังกล่าว ภาพที่เห็นจึงกลายเป็นว่า LED ตัวเลข 7 ส่วนทั้ง 4 หลักแสดงตัวเลข 1234 พร้อมกัน จะเห็นได้ว่าด้วยกระบวนการนี้ LED ตัวเลข 7 ส่วนจะทำงานไม่พร้อมกัน การกินกระแสไฟฟ้าจึงมีค่าสูงสุดเท่ากับ LED ในแต่ละเซกเมนต์

ต้องการติดสว่างพร้อมกันเท่านั้น หาก LED ทุกเซกเมนต์ในหนึ่งหลักติดสว่างพร้อมกันเท่านั้น หาก LED ในแต่ละเซกเมนต์ต้องการกระแสไฟฟ้า 10 mA การขับ LED ตัวเลข 7 ส่วนแบบมัลติเพล็กซ์จะมีความต้องการกระแสไฟฟ้าสูงสุดเพียง 80 mA เท่านั้นไม่ว่าจะขับ LED ตัวเลข 7 ส่วนเป็นจำนวนเท่าใดก็ตาม



ภาพที่ 4-3 วงจรขับ LED ตัวเลข 7 ส่วน 4 หลักแบบมัลติเพล็กซ์โดยใช้ PIC16F84A

บทที่5
การเชื่อมต่อ PIC16F84A กับอุปกรณ์ขับกระแส และแรงดันทางเอาต์พุต

โดยความสามารถพื้นฐานของพอร์ตเอาต์พุตแต่ละขาของไมโครคอนโทรลเลอร์ PIC16F84A สามารถที่จะขับอุปกรณ์เอาต์พุตได้โดยตรง แต่มีข้อจำกัดในเรื่องของกระแสเอาต์พุต ซึ่งสามารถจ่ายกระแสได้เพียง 20 mA ทำให้เมื่อนำไปใช้กับโหลดที่มีความต้องการกระแสและแรงดันสูงกว่าที่ไมโครค้อนโทรลเลอร์ PIC16F84A จะสามารถขับได้โดยตรงจึงต้องขับโหลดเหล่านั้นผ่านอุปกรณ์ขับกระแสและแรงดันเอาต์พุตหรือเรียกสั้น ๆ ว่า อุปกรณ์ไดรเวอร์ (driver) ซึ่งในที่นี้นำมาอธิบายทั้งสิ้น 3 รูปแบบดังนี้

1. ใช้ทรานซิสเตอร์ขับ
2. ใช้ไอซีขับ
3. ใช้อุปกรณ์เชื่อมโยงทางแสงหรือออปโต้คัปเปอร์ (optocoupler) สำหรับโหลดที่ต้องการกระแสสูงมาก ๆ ทำให้ต้องมีการแยกระบบกราวด์ของไมโครคอนโทรลเลอร์ออกจากวงจรเอาต์พุต เพื่อลดสัญญาณรบกวน

5.1 การใช้ทรานซิสเตอร์ขับ

มีด้วยกัน 3 รูปแบบ โดยใช้ทรานซิสเตอร์ที่มีพิกัดแรงดันและกระแสแตกต่างกัน เพื่อรองรับโหลดที่มีความต้องการกระแสและแรงดันแตกต่างกันในที่นี้นำทรานซิสเตอร์ชนิด NPN 3 เบอร์ที่มีคุณสมบัติแตกต่างกันมาอธิบายให้เห็นอย่างชัดเจน ดังต่อไปนี้

	2N3904	TIP31	TIP122 หรือ 2N6387
Ic(max)	100 mA	2 A	2 A
V_{BE}(max)	0.8 V	1 V	1.4 V
V_{CEsat}(max)	0.85 V	1 V	2.8 V
H_{FE}(min)	40	15	750

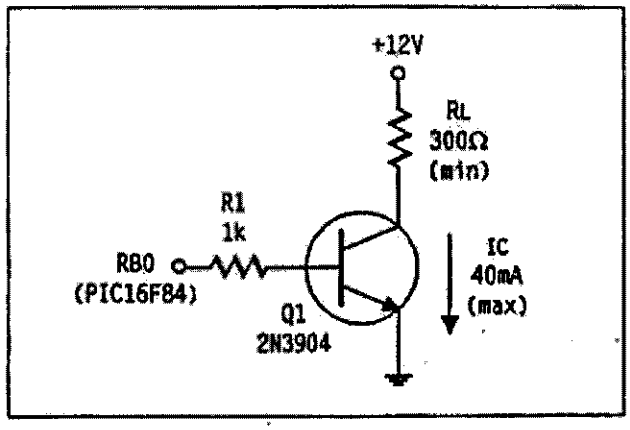
ทรานซิสเตอร์ทั้ง 3 เบอร์ที่ยกมาเป็นตัวอย่างนี้ มีความแตกต่างกันในเรื่องของกระแสและแรงดันอย่างชัดเจน สำหรับเบอร์ TIP 122 หรือ 2N6387 เป็นทรานซิสเตอร์แบบดาร์ลิงตัน ดังนั้นการนำทรานซิสเตอร์มาใช้เป็นอุปกรณ์ขับทางเอาต์พุตจึงสามารถอธิบายได้ 3 รูปแบบดังนี้

5.1.1 การใช้งานทรานซิสเตอร์ขับแบบเดี่ยว

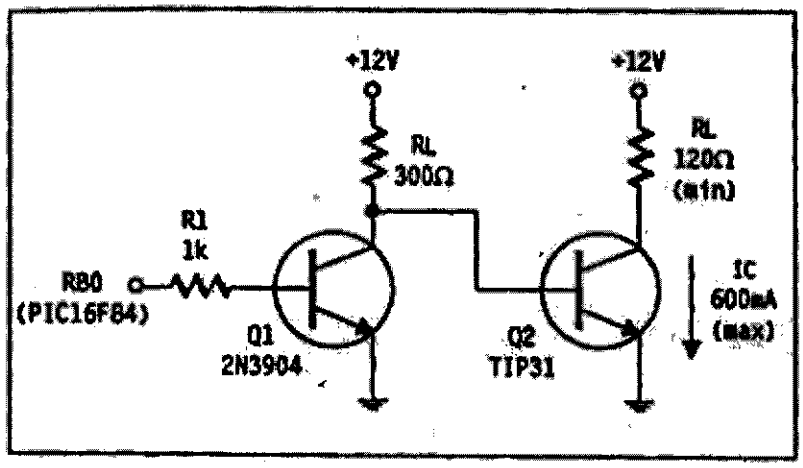
การขับโดยวิธีนี้เหมาะสมสำหรับโหลดที่มีความต้องการกระแสปานกลางตั้งแต่ 30 – 200 mA อาทิ เช่น รีเลย์กำลังต่ำไปจนคุงปานกลางที่มีค่าความต้านทานของขดลวดภายในรีเลย์ไม่ต่ำกล่า 100 W, หลอดไฟแฟลยกำลังต่ำ และมอเตอร์ไฟตรงขนาดเล็ก มีวงจรตัวอย่างตามภาพที่ 15-1 และ 15-2

ในภาพที่ 5-1 เป็นการต่อทรานซิสเตอร์เข้ากับขาพอร์ตเอาต์พุต RB0 ของไมโครคอนโทรลเลอร์โดยมี ตัวต้านทาน R1 ทำหน้าที่จำกัดกระแสเข้าขาเบสของทรานซิสเตอร์ Q1 ทรานซิสเตอร์ Q1 จะทำงานก็ต่อเมื่อขา RB0 มีสภานะลอจิกเป็น "1" เมื่อ Q1 ทำงานก็จะเกิดกระแสไหลผ่าน RL ซึ่งเป็นโหลดต่อยู่ทานต์พุตที่ขาคอลเล็กเตอร์ของ Q1 กระแสโหลดสูงสุด (I_{Lmax}) มีค่าเท่ากับ $12V / 300\,\Omega = 40$ mA

ถึงแม้ว่า Q1 เบอร์ 2N3904 มีค่ากระแสคอลเล็กเตอร์สูงสุดถึง 100 mA แต่ในทางปฏิบัติจริงไม่ควรที่จะออกแบบให้ทรานซิสเตอร์ทำงานถึงกระแสสูงสุด ย่านปลอดภัยของทานซิสเตอร์ควรอยู่ไม่เกินครึ่งหนึ่งของอัตราการทนได้สูงสุด ด้วยการจัดวงจรตามภาพที่ 5-1 สามารถใช้ไมโครคอนโทรลเลอร์กระตุ้นให้ทรานซิสเตอร์ทำงานเพื่อขับรีเลย์ขนาดเล็กได้อย่างปลอดภัย



ภาพที่ 5-1 วงจรขับโหลดกำลังต่ำโดยใช้ทรานซิสเตอร์



ภาพที่5-2วงจรขับโหลดโดยใช้ทรานซิสเตอร์ต่อคาสเคดกันเพื่อเพิ่มความสามาศในการจ่ายกระแส

5.1.2 การใช้ทรานซิสเตอร์ขับแบบคาสเคด

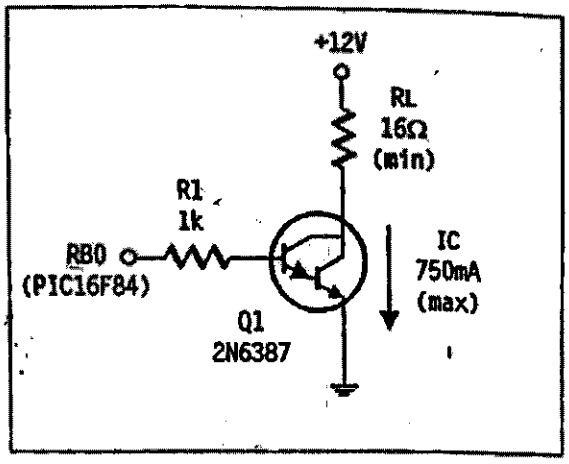
เนื่องจากข้อจำกัดของทรานซิสเตอร์ขนาดเล็ก ทำให้ไม่สามารถขับโหลดที่ต้องการกระแสสูงได้ จึงต้องใช้ทรานซิสเตอร์ที่มีอัตราการทนกระแสสูงได้ จึงต้องใช้ทรานซิสเตอร์ที่มีอัตราการทนกระแสคอลเล็กเตอร์ที่สูงขึ้นมาใช้ขับแต่ ทรานซิสเตอร์ที่สามารถขับโหลดกำลังปานกลางไปจนถึงกำลังสูง จะมีความเร็วในการทำงานช้ากว่าทรานซิสเตอร์กำลังต่ำ ในระบบไมโครคอนโทรลเลอร์ที่มีความเร็วในการทำงานสูงเมื่อต้องการนำมาขับโหลดกำลังปานกลางจึงต้องการนำมาขับโหลดกำลังปานกลางจึงต้องใช้การขับแบบคาสเคดโดยให้ไมโครคอนโทรลเลอร์กระตุ้นทานซิสเตอร์กำลังงานต่ำทำงานเพื่อไปขับทานซิสเตอร์กำลังงานปานกลางต่อไป ดังแสดงวงจรในภาพที่ 5-2

จากวงจรในภาพที่ 5-2 เมื่อทรานซิสเตอร์ Q1 ทำงานด้วยการป้อนลอจิก "1" ออกทางขา RB0 ของไมโครคอนโทรลเลอร์ จะเกิดกระแสไหลผ่าน R2 ไปเข้ายังเบสของทรานซิสเตอร์ Q2 เพื่อกระตุ้นให้ Q2 ทำงานเกิดกระแสไหลผ่านโหลด R_L มีค่าเท่ากับ 600 mA ทำให้สามารถนำไปขับมอเตอร์ไฟตรงหรือสเต็ปเปอร์มอเตอร์ขนาดเล็กได้

5.1.3 การใช้ทานซิสเตอร์แบบดาร์ลิงตันขับโหบดกระแสสูง

จากการใช้ทรานซิสเตอร์ต่อกันแบบคาสเคดเพื่อเพิ่มความสามารถในการขับกระแสให้สูงขึ้น นำมาสู่การใช้ทรานซิสเตอร์อีกแบบหนึ่งที่มีบรรจุทราสซิสเตอร์ 2 ตัวต่อกันแบบดาร์ลิงตันภายในตัวถึงเดียวกัน ทำให้สามารถขับกระแสเอาต์พุตได้สูงสุดและมีความเร็วในการทำงานสูงด้วยโดยใช้อุปกรณ์เพียงตัวเดียว ส่งผลให้ขนาดของวงจรเล็กลง ดังแสดงวงจรตามภาพที่ 5-3

จากวงจรในภาพที่ 5-3 ทรานซิสเตอร์ Q1 ซึ่งเป็นทรานซิสเตอร์แบบดาร์ลิงตันสามารถขับกระแสเอาต์พุตได้สูงถึง 700 mA ด้วยการต่อเข้ากับขาพอร์ตของไมโครคอนโทรลเลอร์โดยผ่านตัวต้านทานจำกัดกระแสเพียงตัวเดียวและไม่ต้องต่อทรานซิสเตอร์แบบคาสเคดทำให้มีความเร็วในการทำงานสูงตลอดจนยังสามารถขับกระแสเอาต์พุตได้สูงพอสมควร



ภาพที่ 5-3 วงจรขับโหลดโดยใช้ทรานซิสเตอร์แบบดาร์ลิงตัน

5.2 การใช้ไอซีขับ (Transistor driver IC)

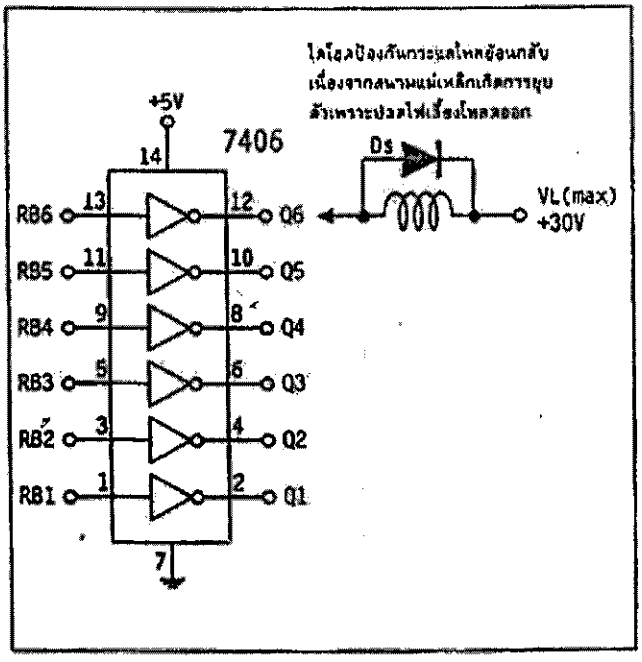
ไอซีที่ใช้ขับโหลดกระแสสูงมักจะมีวงจรทางเอาต์พุตเป็นแบบคอลเล็กเตอร์เปิด ทำให้สามารถใช้กับแรงดันไม่ต่ำกว่า 30 V ขึ้นอยู่กับไอซีในแต่ละเบอร์ สำหรับไอซีขับหรือไอซีไดรเวอร์ที่ยกมาอธิบายมีด้วยกัน 3 เบอร์คือ 74LS06 ซึ่งเป็นไอซีอินเวอร์เตอร์ไดรเวอร์ ภายในบรรจุอินเวอร์เตอร์เกตแบบคอลเล็กเตอร์เปิด 6 ตัว, 7407 เป็นไอซีบัฟเฟอร์ไดรเวอร์ ภายในบรรจุบัฟเฟอร์แบบคอลเล็กเตอร์ 6 ตัว และ ULN2003 เป็นไอซีอินเวอร์เตอร์ไดรเวอร์เช่นเดียวกับ 74LS06 แต่ภายในบรรจุอินเวอร์เตอร์เกต 7 ตัว สำหรับรายละเอียดของไอซีไดรเวอร์แต่ละตัวมีดังนี้

5.2.1 74LS06 และ 74LS07

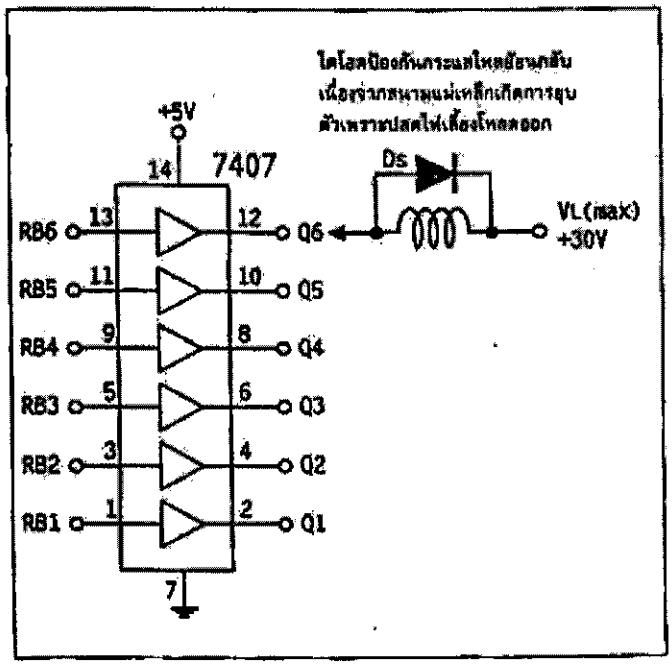
ไอซีไดรเวอร์เบอร์ 74LS06 และ 74LS07 แตกต่างกันตรงที่ลักษณะการทำงาน 74LS06 จะทำงานกลับลอจิกอินพุต ในขณะที่ 7407 จะไม่มีการกลับลอจิกแต่อย่างใด แรงดันเอาต์พุตสามารถไหลได้สูงสุด +30V กระแสซิงก์ในแต่ละขาเอาต์พุตของไอซีสูงสุด 40 mA การจัดขาของไอซีทั้งสองเบอร์แสดงในภาพที่ 5-4 และ 5-5

ที่ขาเอาต์พุตของ 7406 และ 7407 จะไม่มีไดโอดต่ออยู่ดังนั้นหากนำ 7406 และ 7407 ใช้ขับโหลดที่เป็นขดลวด อาทิเช่น มอเตอร์, สเต็ปเปอร์มอเตอร์ และรีเลย์ จะต้องมีการต่อไดโอดเพื่อ

ป้องกันกระแสไหลย้อนกลับเนื่องจากการยุบตัวของสนามแม่เหล็กของโหลดที่เป็นขดลวด ดังแสดงให้เห็นในภาพที่ 5-4 และ 5-5



ภาพที่ 5-4 การจัดขาของไอซี 7406 และการต่อใช้งาน

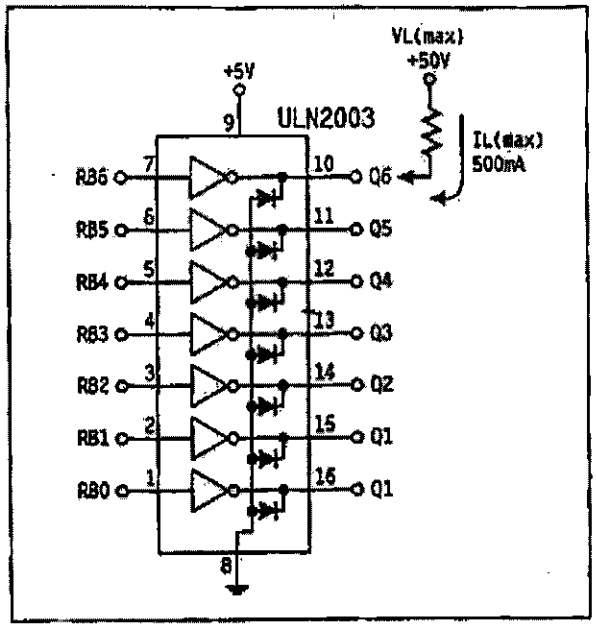


ภาพที่ 5-5 การจัดขาของไอซี 7407 และการต่อใช้งาน

ขาอินพุตของ7406และ 7407 สามารถต่อเข้ากับขาพอร์ตเอาต์พุตของไมโครคอนโทรลเลอร์ได้โดยตรง ส่วนเอาต์พุตต่อเข้ากับโหลด สำหรับการควบคุมให้ 7406 และ 7407 ขับโหลดก็จะมีความแตกต่างหันด้วย โดยหากต้องการให้ 7406 ทำงานเพื่อขับรีเลย์ที่ต่อยู่ทางเอาต์พุต ต้องป้อนลอจิก "1" เมื่อไอซี 7406 ทำงานก็จะกลับลอจิกเป็น "0" ทำให้เกิดกระแสไหลผ่านขดลวดของรีเลย์ที่ต่ออยู่กับขาพอร์ตเอาต์พุต รีเลย์ที่ต่อเป็นโหลดอยู่ก็จะทำงาน ในขณะที่หากต้องการควบคุมให้ 7407 ทำการขับอุปกรณ์ทางเอาต์พุต ต้องป้อนลอจิก "0" เข้าที่อินพุต ซึ่งตรงข้ามกับไอซีเบอร์ 7406

5.2.2 ULN2003

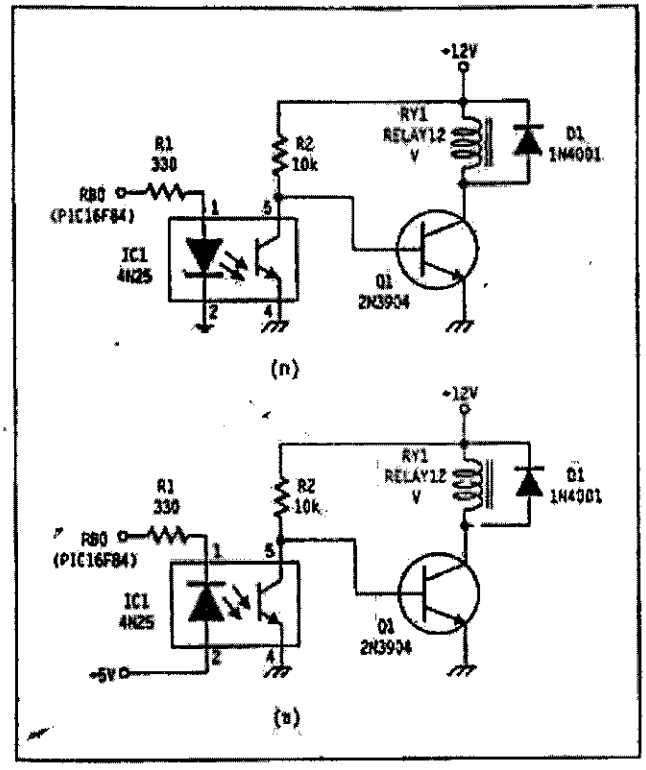
ไอซีไดรเวอร์เบอร์นี้มีความสามารถสูงกว่า 7406 และ 7407 โดยภายในบรรจุอินเวอร์เตอร์เกตแบบคอลเล็กเตอร์เปิด 7 ตัว สามารถใช้กับแรงดันได้สูงสุด +50 V กระแสเอาต์พุตสูงสุดในแต่ละขาเท่ากัน 500 mA นอกจากนั้นยังต่อไดโอดป้องกันไว้ที่ทุกขอเอาต์พุต ทำให้สามารถต่อโหลดที่เป็นขอลวดได้ทันที การจัดวงจรภายในและขาต่อใช้งานของ ULN2003 แสดงได้ดังภาพที่ 5-6



ภาพที่ 5-6 การจัดขาของไอซี ULN2003 และการต่อใช้งาน

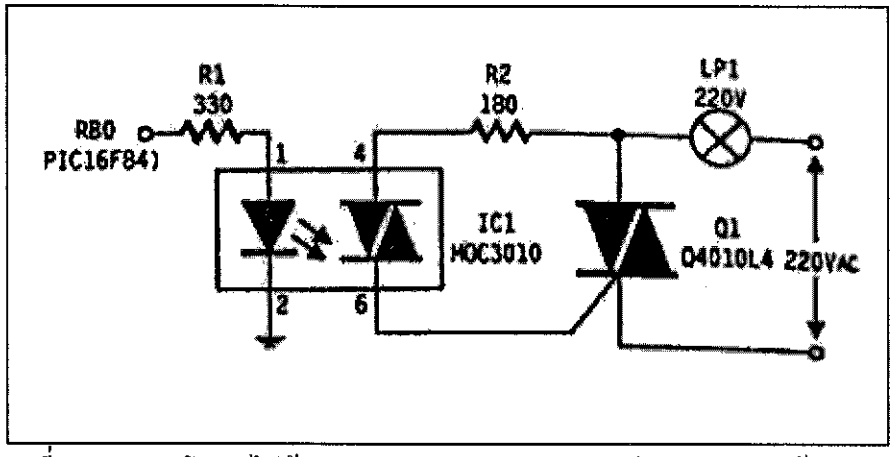
5.3 การขับโหลดโดยใช้ออปโต้คัปเปลอร์

ในงานที่ต้องใช้ไมโครคอนโทรลเลอร์ควบคุมโหลดที่ต้องการแรงดันและกระแสไฟฟ้าสูงมาก ๆ หรือเป็นโหลดทางไฟสลับหากใช้ไมโครคอนโทรลเลอร์เข้าไปควบคุมโดยตรงอาจส่งผลเสียต่อระบบไมโครคอนโทรลเลอร์เอง เนื่องจากที่โหลดกำลังไฟฟ้าสูงนั้นเมื่อเริ่มต้นหรือสิ้นสุด



ภาพที่ 5-8 การขับออปโต้คัปเปลอร์ด้วยข้อมูลดิจิตอล

(ก) ขับด้วยข้อมูล "1" (ข) ขับด้วยข้อมูล "0"



ภาพที่ 5-9 การขับโหลดไฟฟ้ากระแสสลับของ PIC16F84A โดยผ่านออปโต้คัปเปลอร์

ภาคอินพุตของออปโต้คัปเปลอร์ โดยส่วนใหญ่จะเป็น LED อินฟราเรด ในขณะที่ภาคเอาต์พุตจะเป็นอุปกรณ์สารกึ่งตัวนำที่ทำงานเมื่อได้มีแสงมาตกกระทบ เช่น โฟโต้ทรานซิสเตอร์, โฟโต้ดาร์ลิงตัน, โฟโต้ลอจิก และโฟโต้ไดแอกหรือโฟโต้ไทริสเตอร์ ดังแสดงสัญลักษณ์ของออปโต้คัปเปลอร์แบบต่าง ๆ ในภาพที่ 5-7

การทำงานจะเริ่มต้นด้วนการจ่ายแรงดันไบแอสตรงให้แก่ LED อินฟราเรดภายในออปโต้คัปเปลอร์ เมื่อ LED นำหระแส ก็จะกำเนิดแสงอินฟราเรดส่องไปตกกระทบที่ขาเบสของโฟโต้ทรานซิสเตอร์ (ในกรณีที่ภาคเอาต์พุตเป็นโฟโต้ทรานซิสเตอร์หรือโฟโต้ดาร์ลิงตัน) ทำให้โฟโต้ทรานซิสเตอร์นำกระแส จะเกิดกระแสไหลผ่านจากขาคอลเล็กเตอร์มายังอิมิตเตอร์ โดยแหล่งจ่ายแรงดันทางเอาต์พุตสามารถที่จะใช้แยกกับทางอินพุตได้อย่างอิสระ ไม่ต้องต่อกราวด์ถึงกันจึงสามารถต่อกับแหล่งกำเนิดแรงดันสูงหรือแหล่งกำเนิดไฟสลับได้โดยไม่มีการรบกวนจากอินพุตโดยผ่านทางสายกราวด์ได้ด้วย

ในภาพที่ 5-8 เป็นวงจรตัวอย่างขอูงการนำไมโครคอนโทรลเลอร์ขับออปโต้คัปเปลอร์เพื่อทำการขับโหลดรีเลย์กำลังไฟฟ้าสูงจะเห็นได้ว่าการใช้ไมโครคอนโทรลเลอร์กระตุ้นให้ออปโต้คัปเปลอร์ทำงานทำได้ 2 รูปแบบคือ ใช้ลอจิก “1” ดังแสดงในภาพที่ 5-8(ก) และใช้ลอจิก “0” ดังแสดงในภาพที่ 5-8 (ข)

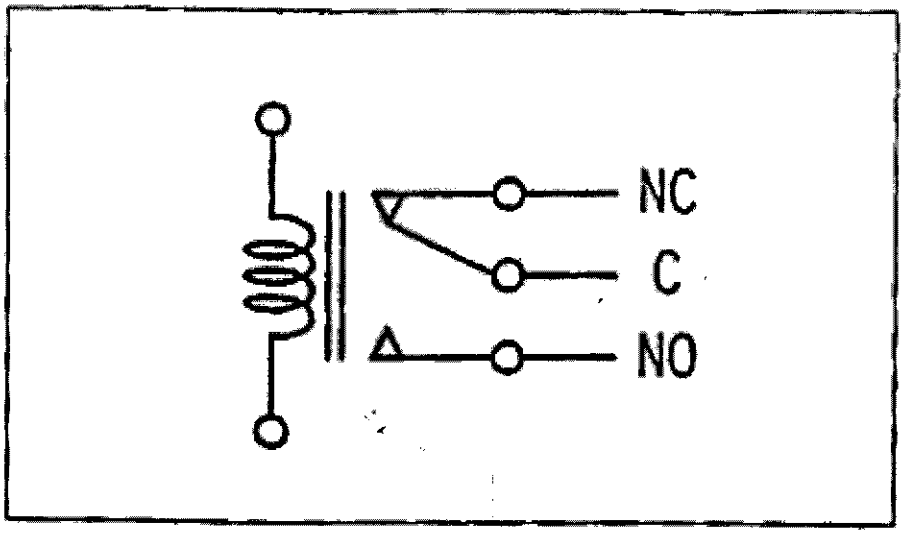
ส่วนในวงจรภาพที่ 5-9 เป็นการใช้ไมโครคอนโทรลเลอร์ควบคุมโหลดไฟฟ้ากระแสสลับผ่านออปโต้คัปเปลอร์แบบที่มีเอาต์พุตเป็นโฟโต้ไดแอก แล้วนำไปกระตุ้นให้ไดแอก Q1 ทำงาน ทำให้เกิดกระแสไหลผ่านโหลดในที่สุด

ข้อควรจำในการใช้ไมโครคอนโทรลเลอร์ร่วมกับออปโต้คัปเปลอร์ จะต้องต่อตัวต้านทานเพื่อจำกัดกระแสไฟฟ้าที่ไหลผ่าน LED ทางส่วนอินพุตของออปโต้คัปเปลอร์ด้วยเสมอ กระแสทางอินพุตสูงสุดไม่ควรมากกว่า 15 mA

5.4 ตัวอย่างการใช้งานอุปกรณ์ไดรเวอร์

ควบคุมรีเลย์โดยใช้ทรานซิสเตอร์

รีเลย์เป็นอุปกรณ์แม่เหล็กไฟฟ้าแบบหนึ่งที่ใช้ในการตัดต่อวงจร มีสัญลักษณ์แสดงตามภาพที่ 5-10 เมื่อขดลวดรีเลย์มีกระแสไฟฟ้าไหลผ่าน จะเกิดสนามแม่เหล็กขึ้น ทำให้หน้าสัมผัสโลหะที่ปกติเปิดวงจรแยกจากกันอยู่ ถูกดูดติดกัน เกิดการต่อวงจรขึ้น ส่วนหน้าสัมผัสที่ปกติต่อวงจรอยู่ก็จะแยกออกจากกัน เกิดการเปิดวงจรขึ้น

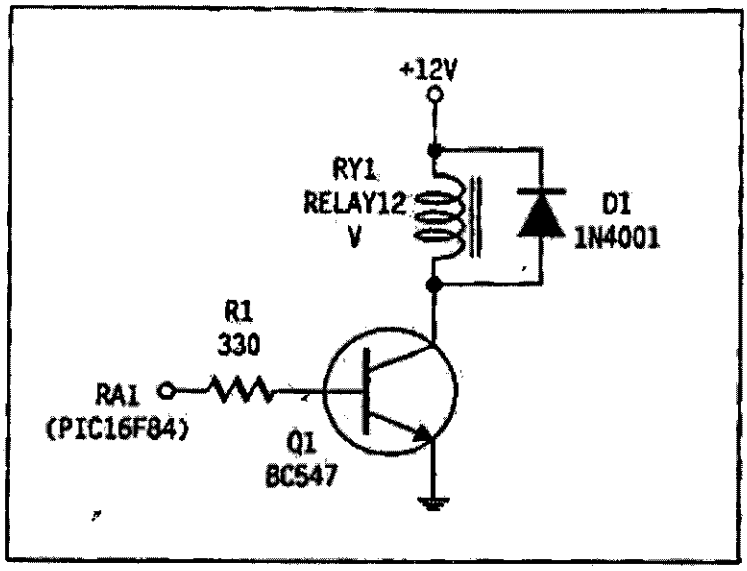


ภาพที่ 5-10 สัญลักษณ์ของรีเลย์แบบกลไก

รีเลย์โดยทั่วไปจะมีหน้าสัมผัสทางเอาต์พุต 2 แบบที่มีความเกี่ยวข้องกันคือ หน้าสัมผัสปกติต่อวงจรหรือ NC (Normaly Closed) และหน้าสัมผัสปกติเปิดวงจรหรือ NO (Normally Opened) ในกรณีของหน้าสัมผัสปกติต่อวงจร เมื่อรีเลย์ทำงานหน้าสัมผัสชุดนี้จะแยกจากกันกลายเป็นเปิดวงจร หน้าสัมผัสชุดนี้จึงใช้ก็ต่อเมื่อมีจุดประสงค์ให้รีเลย์ทำการตัดวงจรเมื่อทำงาน

ส่วนหน้าสัมผัสปกติ เปิดวงจรเมื่อรีเลย์ทำงาน หน้าสัมผัสชุดนี้จะก่อกัน กลายเป็นปิดวงจร ดังนั้นจึงใช้หน้าสัมผัสชุดนี้ก็ต่อเมื่อต้องการให้รีเลย์ต่อวงจรเมื่อทำงาน ดังนั้นหน้าสัมผัสรีเลย์ 1 ชุด จึงเสมือนกับสวิตซ์ 2 ทางหรือ DPST (Double Pole Single Throw)

รีเลย์ที่มีจำน่ายอยู่ทั่งไปมีชดลวด 1 ชุด ในขณะที่หน้าสัมผัสมีตั้งแต่ 1 ชุด (หน้าสัมผัส 1 ชุประกอบด้วย NO และ NC) จำนวน 5 ขา ไปจนถึง 4 ชุด 14 ขา ส่วนอัตราการทนกำลังไฟฟ้าของหน้าสัมผัสมีให้เลือกมากมาย ถ้าเป็นขนาดของแรงดันจะมีตั้งแต่ 30 V ขึ้นไปจนถึง 1000 V ส่วนขนาดของกระแสมีตั้งแต่ไม่กี่แอมป์ไปจนถึงหลาย ๆ สิบแอมป์



ภาพที่ 5-11 วงจรขับรีเลย์โดยใช้ทรานซิสเตอร์

วงจรขับรีเลย์โดยใช้ทรานซิสเตอร์แสดงตามภาพที่ 5-11 ขาพอร์ตเอาต์พุตต่อเข้ากับขาเบสของทรานซิสเตอร์ที่ใช้ขับรีเลย์ผ่านตัวต้านทาน R1 ซึ่งทำหน้าที่จำกัดกระแสเบสให้แก่ทรานซิสเตอร์ Q1 ขดลวดรีเลย์จะต่อเข้าที่คอลเล็กเตอร์โดยมีไดโอด D1 ต่อคร่อมขอลวดรีเลย์ RY1 อยู่ เพื่อป้องกันกระแสไหลย้อนกลับมาเข้าทรานซิสเตอร์ เมื่อรีเลย์หยุดทำงาน

เมื่อต้องการสั่งให้รีเลย์ทำงาน ต้องเขียนโปรแกรมเพื่อกำหนดให้ขา RA1 เป็นเอาต์พุตเสียก่อน จากนั้นจึงเขียนข้อมูล "1" ออกไปทางขา RA1 ทรานซิสเตอร์ Q1 จะนำกระแสทำให้เกิดกระแสไหลผ่านขดลวดรีเลย์ RY1 รีเลย์จึงทำงานโหลดในวงจรนี้คือ LED1 จะติดสว่าง เมื่อต้องการควบคุมให้รีเลย์หยุดทำงาน ทำได้โดยการเขียนข้อมูล "0" ออกไปทางขา RA1 อีกครั้ง ทรานซิสเตอร์ Q1 ก็จะหยุดนำกระแส ทำให้รีเลย์หยุดทำงาน LED1 จึงดับลง

ข้อแนะนำประการหนึ่งในการใช้รีเลย์เพื่อขับโหลดที่มีกำลังไฟฟ้าสูงคือ ไม่ควรใช้รีเลย์กับโหลดที่มีขนาดมากกว่า 20 แอมป์เนื่องจากเมื่อรีเลย์ทำงาน ทำการต่อไฟเลี้ยงให้แก่โหลดอาจเกิดประกายไฟขึ้นที่หน้าสัมผัสได้ ทำให้หน้าสัมผัสเสื่อมสมรรถภาพส่งผลให้เกิดความเสียหายได้ในที่สุด

บทที่6
การทดลองและทดสอบอุปกรณ์

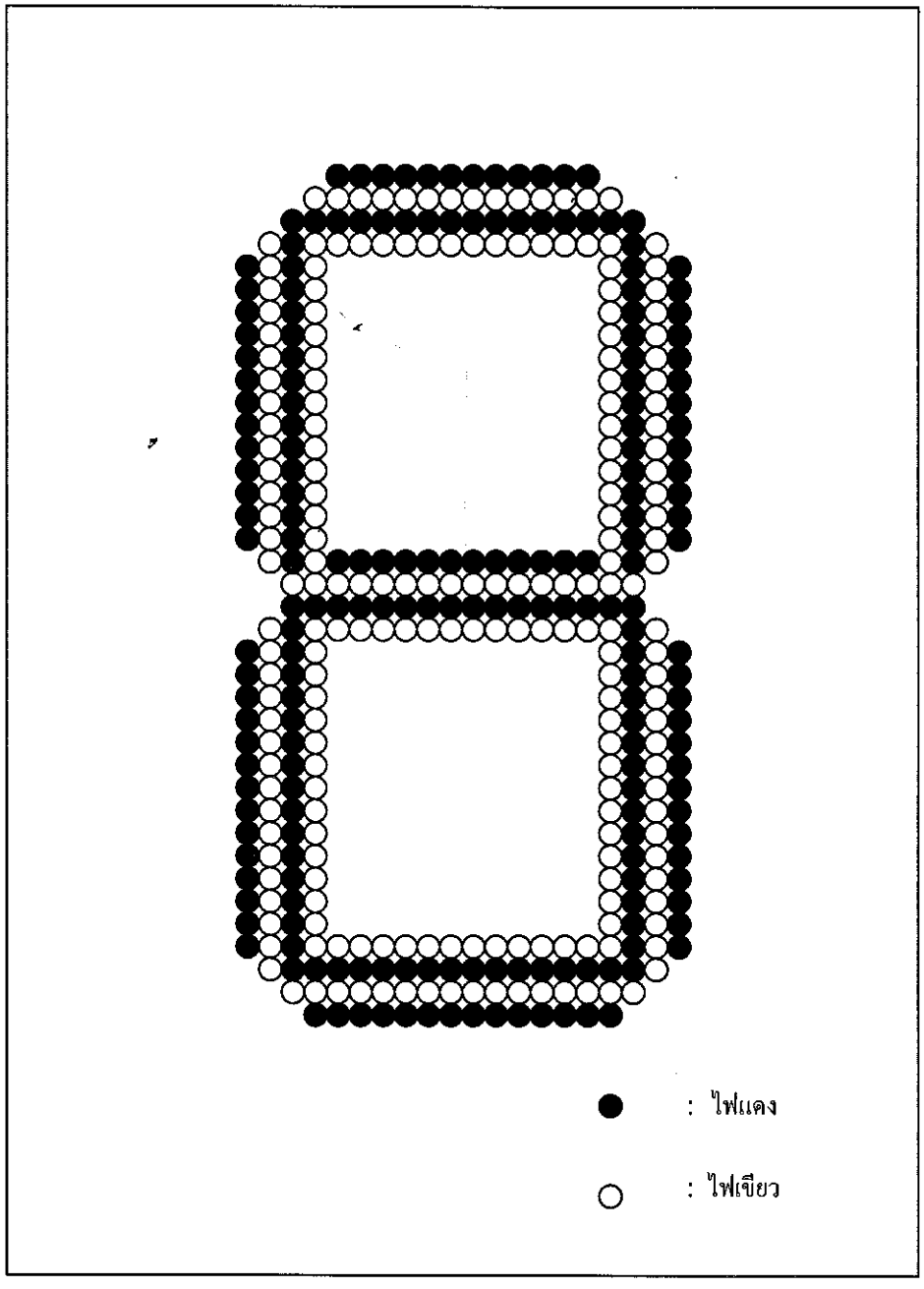
6.1 กล่าวนำ

เพื่อเป็นการยืนยันการออกแบบระบบสำหรับการทดลองและทดสอบอุปกรณ์แบ่งได้ดังนี้คือ การทดสอบแผงแสดงเวลา 7 – segment การทดสอบการทำงานเฉพาะส่วนไมโครคอนโทรลเลอร์ PIC16F84A และการทดสอบระบบรวมทั้งหมด

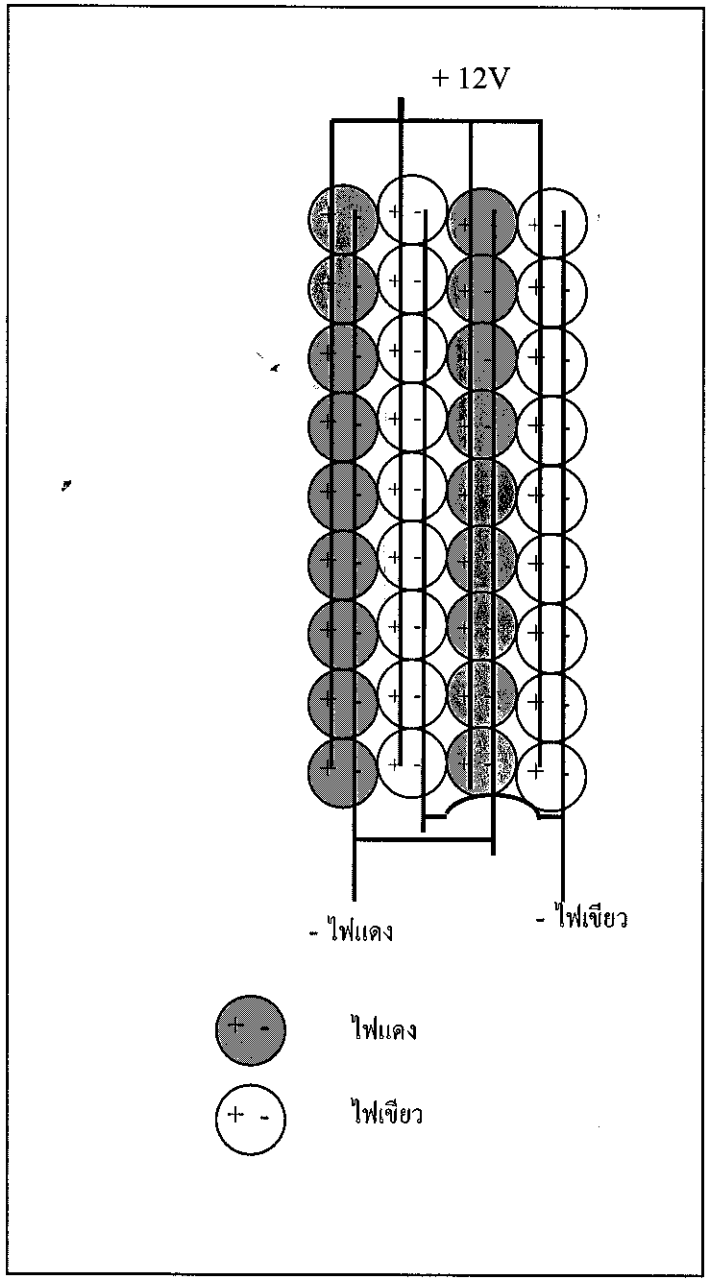
6.2 การทดสอบแผงแสดงเวลา 7 – segment

วัตถุประสงค์ของการทดสอบนี้เพื่อยืนว่าแผง 7-segment ที่ออกแบบมาสามารถที่จะแสดงผลได้ตามที่ต้องการ และความสว่างของ 7-segment ซึ่งมีขั้นตอนการทดสอบดังนี้

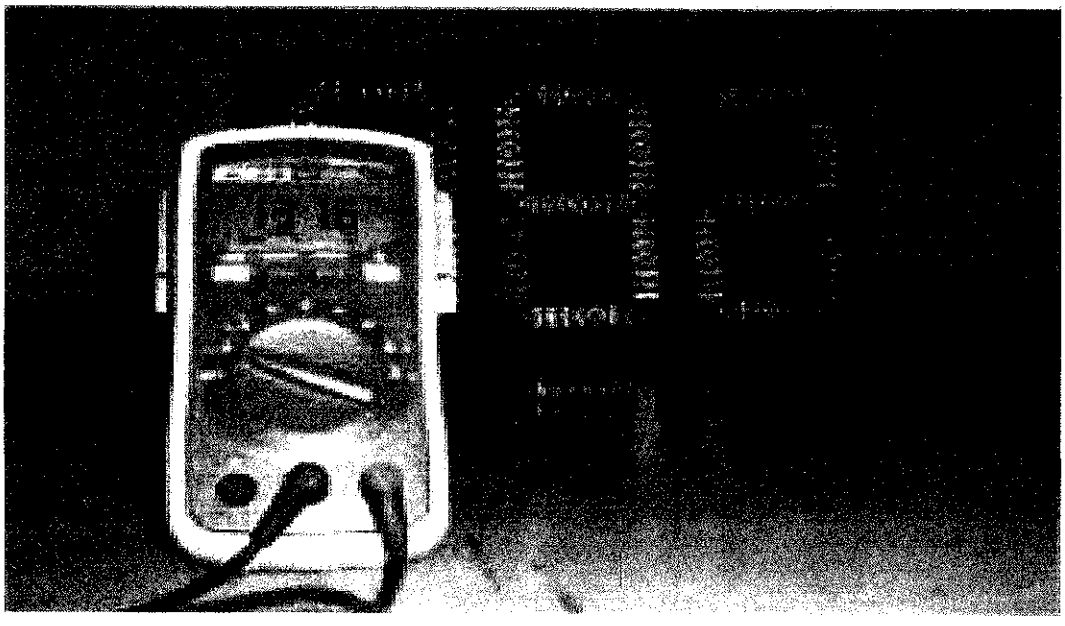
เนื่องจากแผงวงจร มีการทั้งหลอดไฟสีเขียวและหลอดไฟสีแดงสลับกัน และการแสดงแต่ละครั้งเราต้องการเพียงสีเดียว เพื่อเป็นการง่ายในง่ายในการเขียนโปรแกรม ที่จะทำให้แผง 7-segment รับคำสั่งจากไมโครคอนโทรลเลอร์ PIC16F84A เราจึงได้ต่อแผง 7-segment เป็นแบบคอมมอนแคโทด ซึ่งจะรวมเอาขั้วบวกของไฟเขียวและขั้วบวกของไฟแดงในแต่ละ segment เข้าไว้ด้วยกัน และแยกขั้วลบของไฟเขียวและไฟแดงออกจากกัน ซึ่งจะแสดงดังต่อไปนี้



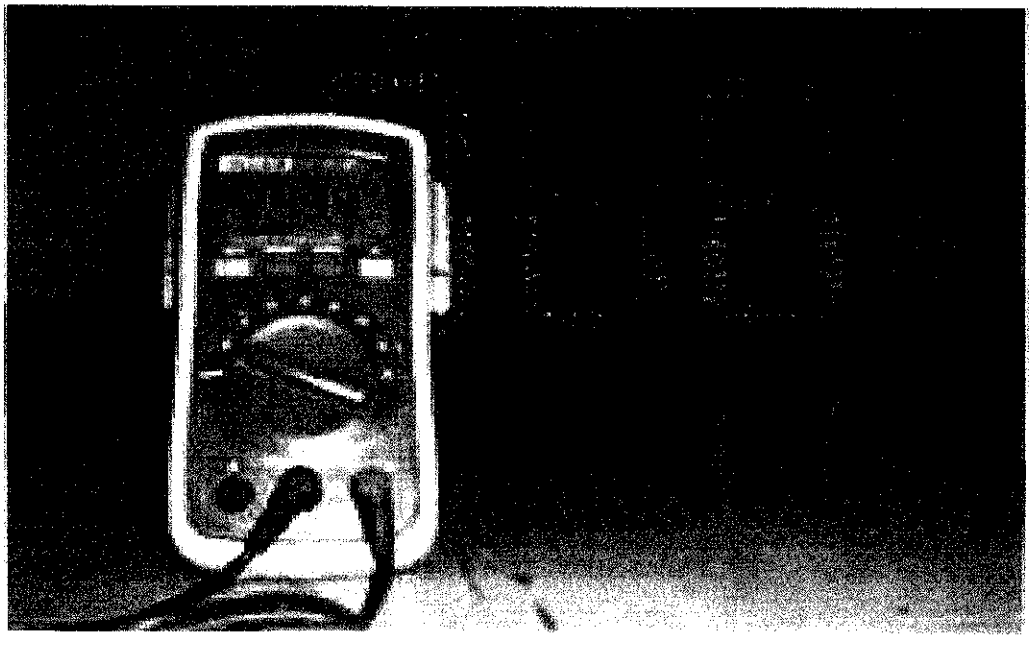
ภาพที่ 6.2-1 การว่างไฟเขียวและไฟแดง



ภาพที่ 6.2-2 ลักษณะการต่อแบบคอมมอนแคโทด



ภาพที่ 6.2-3 ผลของการทดสอบ

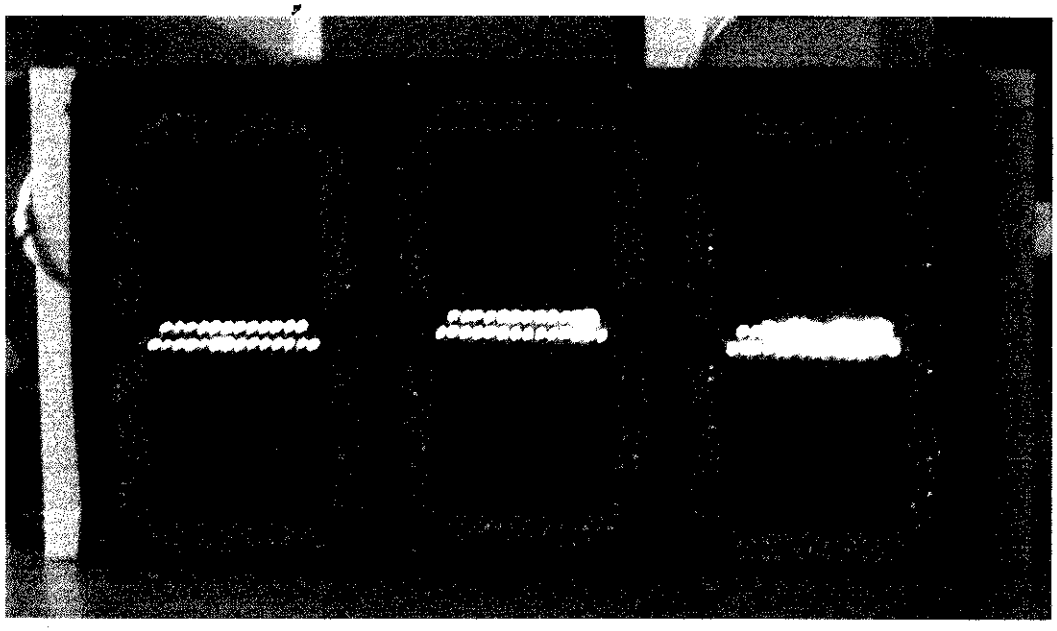


ภาพที่ 6.2-4 ผลของการทดสอบ

6.3 การทดสอบในส่วนของภาคควบคุมและประมวลผล

ในส่วนนี้จะเป็นการทดสอบในส่วนของไมโครคอนโทรลเลอร์และการเขียนโปรแกรมควบคุม การประมวลผลซึ่งขั้นตอนการทดลองเป็นดังนี้

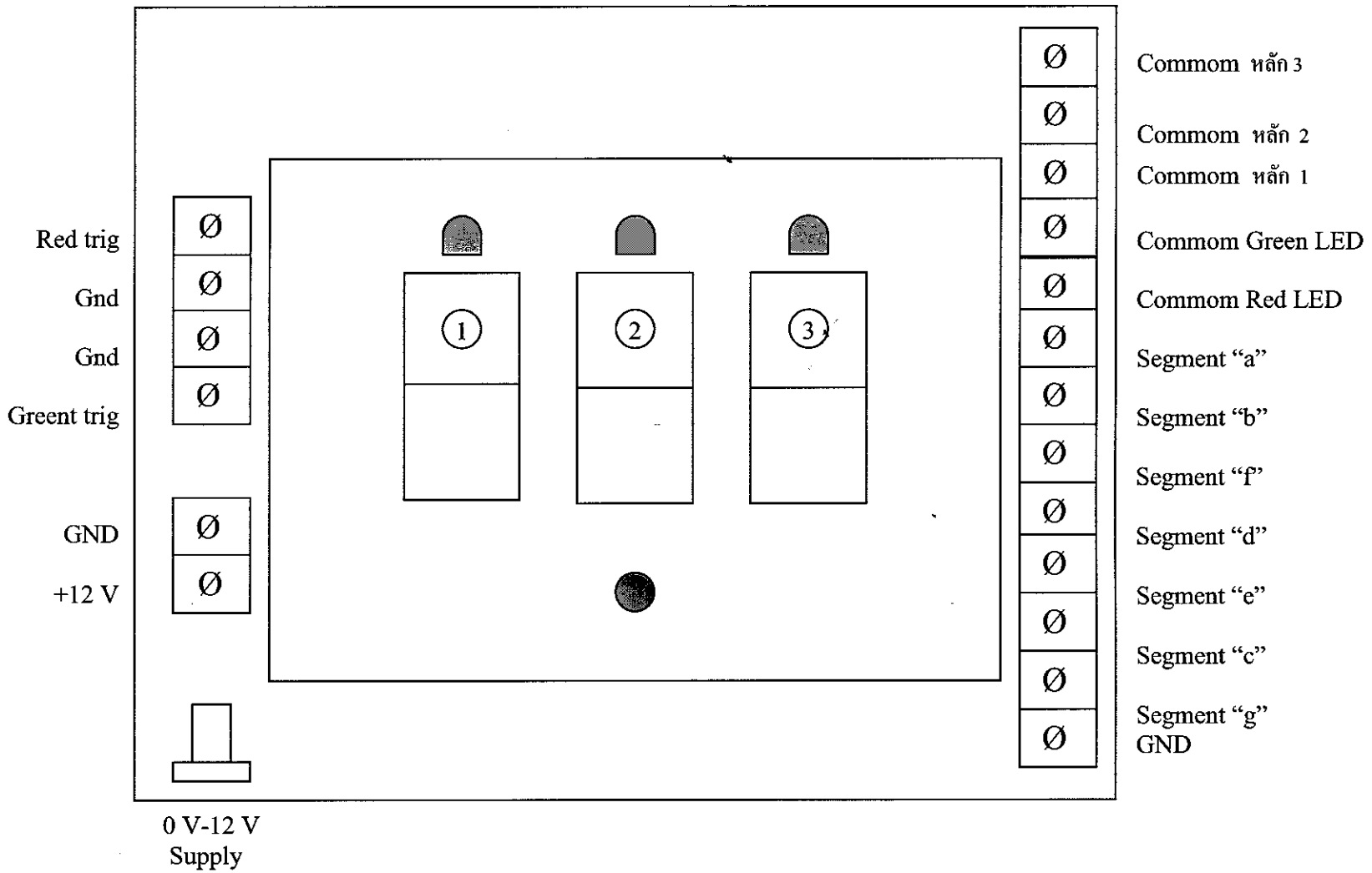
เขียนโปรแกรมโดยใช้ภาษาแอสเซมบลี้สำหรับ PIC และทำการทดสอบรันโปรแกรมที่ได้ซึ่งจะทำได้โดยการนำเอาขาที่ 3 และ ขา 2 ไปจับกันสัญญาณไฟที่จำลองขึ้นมาซึ่งจะแสดงให้เห็นดังรูปต่อไปนี้

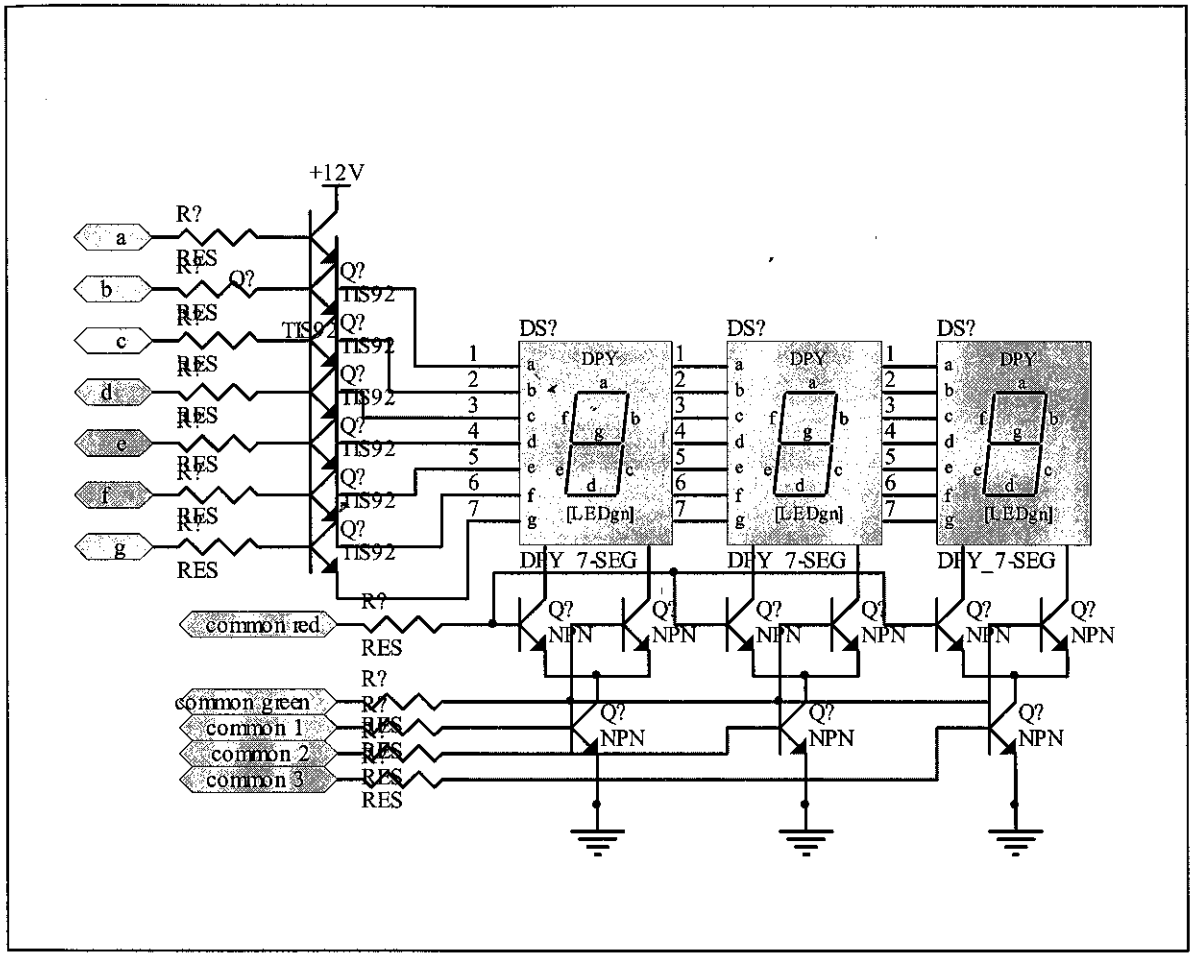


ภาพที่ 6.3-1 การแสดงผลเมื่อเริ่มต่อสัญญาณไฟจราจร

6.4 การทดสอบระบบรวม

ระบบรวมทั้งหมดจะเป็นการนำเอาส่วนของการแสดงผล 7 – segment ทั้ง 3 หลัก มาเชื่อมต่อเข้ากับ ส่วนของการประมวลผลที่ได้รันโปรแกรมเรียบร้อยแล้ว


Red trig
Gnd
Gnd
Greent trig
GND
+12 V
0 V-12 V
Supply
1
2
3
Commom หลัก 3
Commom หลัก 2
Commom หลัก 1
Commom Green LED
Commom Red LED
Segment "a"
Segment "b"
Segment "f"
Segment "d"
Segment "e"
Segment "c"
Segment "g"
GND



ภาพที่6.3-2 แสดงจุดเชื่อมต่อระหว่างภาคควบคุมและภาคแสดงผล

บทที่7
บทสรุป

ในบทนี้จะกล่าวถึงส่วนต่าง ๆ ของระบบทั้งหมดโดยสรุปแบ่งออกเป็น ส่วนของอุปกรณ์ที่สร้างขึ้น และโปรแกรมควบคุม โดยจะอธิบายถึงปัญหาที่พบในการทำงาน วิธีการแก้ไขปัญหาของส่วนต่าง ๆ

7.1 ข้อสรุปด้านงานที่พัฒนาขึ้นจากโครงงาน

เนื้อหาที่ได้พัฒนาในโครงงานระบบแสดงสัญญาณไฟจราจรอัตโนมัติโดยใช้ไมโครคอนโทรลเลอร์ PIC16F84A สามารถสรุปได้เป็นข้อ ๆ ดังนี้

1. ทำการออกแบบ สร้าง และการทดสอบ ส่วนที่ทำหน้าที่แสดงผล คือ แผงหลอด LED 7- Segment จำนวน 3 หลัก รวมถึงการออกแบบชุดวงจรขยายกระแส เพื่อนำมาใช้ในการขยายกระแสให้กับ 7 – segment ทั้ง 3 หลัก

2. ทำการออกแบบ สร้าง และทดสอบ ส่วนที่ทำหน้าที่รับสัญญาณไฟจราจร เพื่อนำค่าที่ได้ไปใช้ในการประมวลผลในขั้นตอนต่อไป

3. ทำการออกแบบ สร้าง และทดสอบ ส่วนที่ทำหน้าที่ควบคุม และประมวลผล โดยการใช้ไมโครคอนโทรลเลอร์ PIC16F84A มาใช้เป็นตัวหลักในการทำงานในส่วนนี้

4. ได้ทำการออกแบบ และเขียนโปรแกรมด้วยภาษาแอสเซมบลี้ เพื่อใช้ในการควบคุมไมโครคอนโทรลเลอร์ตระกูล PIC16F84 ให้สามารถควบคุมการทำงานและประมวลผลในส่วนต่าง ๆ เช่น ส่วนอุปกรณ์ตรวจรับสัญญาณ ส่วนแสดงผล และส่วนควบคุม

5. ได้ทำการทดลอง ทดสอบการใช้งานจริงของระบบทั้งหมด ซึ่งผลการทดสอบปรากฏว่าระบบทั้งหมดสามารถใช้งานได้จริง

7.2 ปัญหาที่พบ และแนวทางการแก้ไข

ปัญหาที่เกิดขึ้นกับโครงงานนี้ประกอบด้วยปัญหาที่ส่วนของการแสดงผล 7- Segment ที่ใช้หลอด LED หลาย ๆ หลอดมาต่อกันเป็นตัวเลขแสดงผล และปัญหาเกี่ยวกับกระแสและแรงดันที่จ่ายให้กับหลอดไฟ นอกจากนี้ยังพบปัญหาในส่วนของไมโครคอนโทรลเลอร์ที่จะนำมาใช้ในการประมวลผลและควบคุม โดยรายละเอียดและการแก้ไขดังตารางที่ 7.1

ปัญหาที่พบ	สาเหตุและแนวทางแก้ไข
1. ส่วนแสดงผล 7- Segment ทั้ง 3 หลัก สว่างไม่ เท่ากัน	สาเหตุ เนื่องจาก 7-segment ที่รับคำสั่งจากไมโครคอนโทรเลอร์ มีการเชื่อมต่อแบบอนุกรม เพราะฉะนั้นกระแสแรงดันจึงไปตกอยู่ที่ตัวแรก ทำให้ตัวแรกสว่างมากและตัวที่ 2 และ 3 ไม่ค่อยสว่าง และอีกสาเหตุหนึ่งก็คือคุณภาพของหลอด LED ที่นำมาใช้งาน โดยเฉพาะหลอดสีเขียว การแก้ไข นำเอาตัวต้านทาน ขนาด 10 โอมห์ มาต่อเพื่อจำกัดกระแสที่ไหลผ่าน 7 - Segment
2. ทรานซิเตอร์ที่นำมาใช้ในการขยายกระแสให้กับ 7-segment มีความร้อนมาก	สาเหตุ เกิดจากกระแสที่นำมาใช้มีค่าสูงทำให้ทรานซิเตอร์ระบายความร้อนไม่ทัน การแก้ไข เปลี่ยนทรานซิเตอร์จาก MJE305 มาเป็น 2N3055 ที่สามารถรับกระแสและแรงดันได้มากขึ้น
3. ปัญหาใช้ไมโครคอนโทรเลอร์ที่ใช้ในการควบคุมและประมวลผล	สาเหตุ การที่เขียนโปรแกรมควบคุมและประมวลผลต้องใช้คำสั่งที่ยาวและยุ่งยาก การแก้ไข เปลี่ยนจาก ไมโครคอนโทรเลอร์ AT89S252 มาเป็นไมโครคอนโทรเลอร์ PIC16F84A

ตารางที่ 7.1ปัญหาที่พบในโครงงาน และแนวทางแก้ไข

7.3 ผลที่ได้จากโครงงาน

สำหรับสิ่งที่ได้รับจากพัฒนาโครงงาน ระแสดงสัญญาณไฟจราจรอัตโนมัติ สามารถแบ่งเป็นหัวข้อย่อยได้ดังต่อไปนี้

1. ได้เรียนรู้และศึกษาวิธีการออกแบบรายวงจร โดยใช้โปรแกรม Protel 99SE
2. เข้าใจถึงคุณสมบัติและการใช้งานของหลอด LED และสามารถนำไปประยุกต์ใช้กับงานไดอย่างเหมาะสม

3. เข้าใจถึงคุณสมบัติและการใช้งานของทรานซิสเตอร์ และสามารถนำไปประยุกต์ใช้กับงานได้อย่างเหมาะสม
4. สามารถเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ PIC16F84A และเข้าใจถึงหลักการใช้ งาน
5. ได้ศึกษาการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ PIC16F84A โดยใช้ภาษาแอสเซมบลี้
6. ได้เครื่องต้นแบบและโปรแกรมควบคุมของระบบแสดงสัญญาณไฟจราจรอัตโนมัติซึ่งสามารถนำไปพัฒนาในเชิงพานิชย์ต่อไป

บรรณานุกรม

[1] กฤษดา ใจเย็น และชัยวัฒน์ ลิ้มพรจริตรวิไล, PIC16F84 Theory & Practical Approach, สำนักพิมพ์ อินโนเวตีฟ เอ็กเพอริเมินต์ จำกัด.

[2] ชัยวัฒน์ ลิ้มพรจริตรวิไล และณัฏฐพล วงค์สุนทรชัย, แนะนำการใช้โปรแกรม MPLAB PIC-micro development tool และการทดลองทางซอฟต์แวร์ของไมโครคอนโทรเลอร์ PIC, สำนักพิมพ์ อินโนเวตีฟ เอ็กเพอริเมินต์ จำกัด.

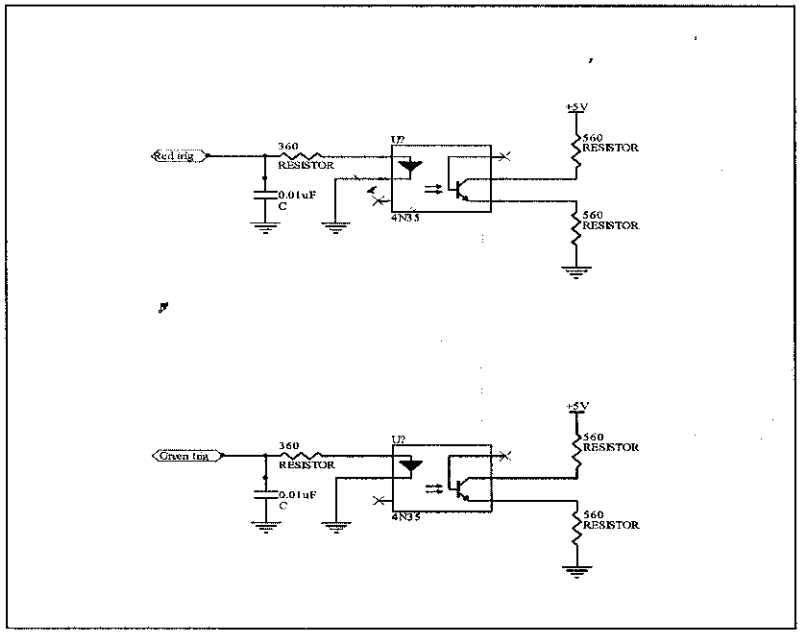
[3] ชัยวัฒน์ ลิ้มพรจริตรวิไล และวรพจน์ กรแก้ววัฒนกุล, เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช, สำนักพิมพ์ อินโนเวตีฟ เอ็กเพอริเมินต์ จำกัด.

[4] ประเมษฐ์ ประนยานันทน์ และปิยพงศ์, คู่มือการประยุกต์ใช้งานไมโครคอนโทรลเลอร์, สำนักพิมพ์ซีเอ็ดยูเคชั้น จำกัด(มหาชน).

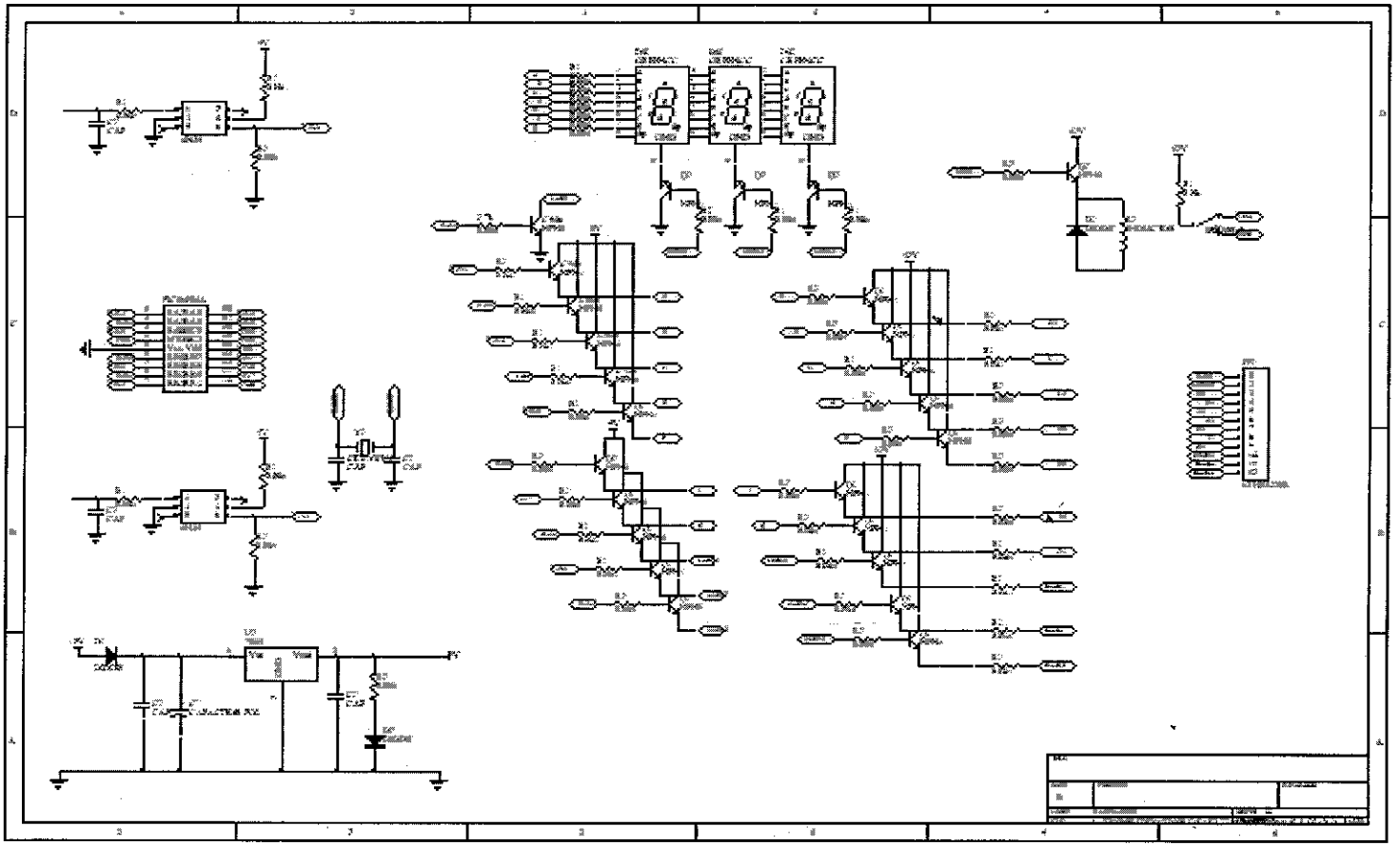
[5] อิเล็กทรอนิกส์,สำนักพิมพ์ ซีเอ็ด, ฉบับที่ 253 , กันยายน 2546.

ภาคผนวก ก

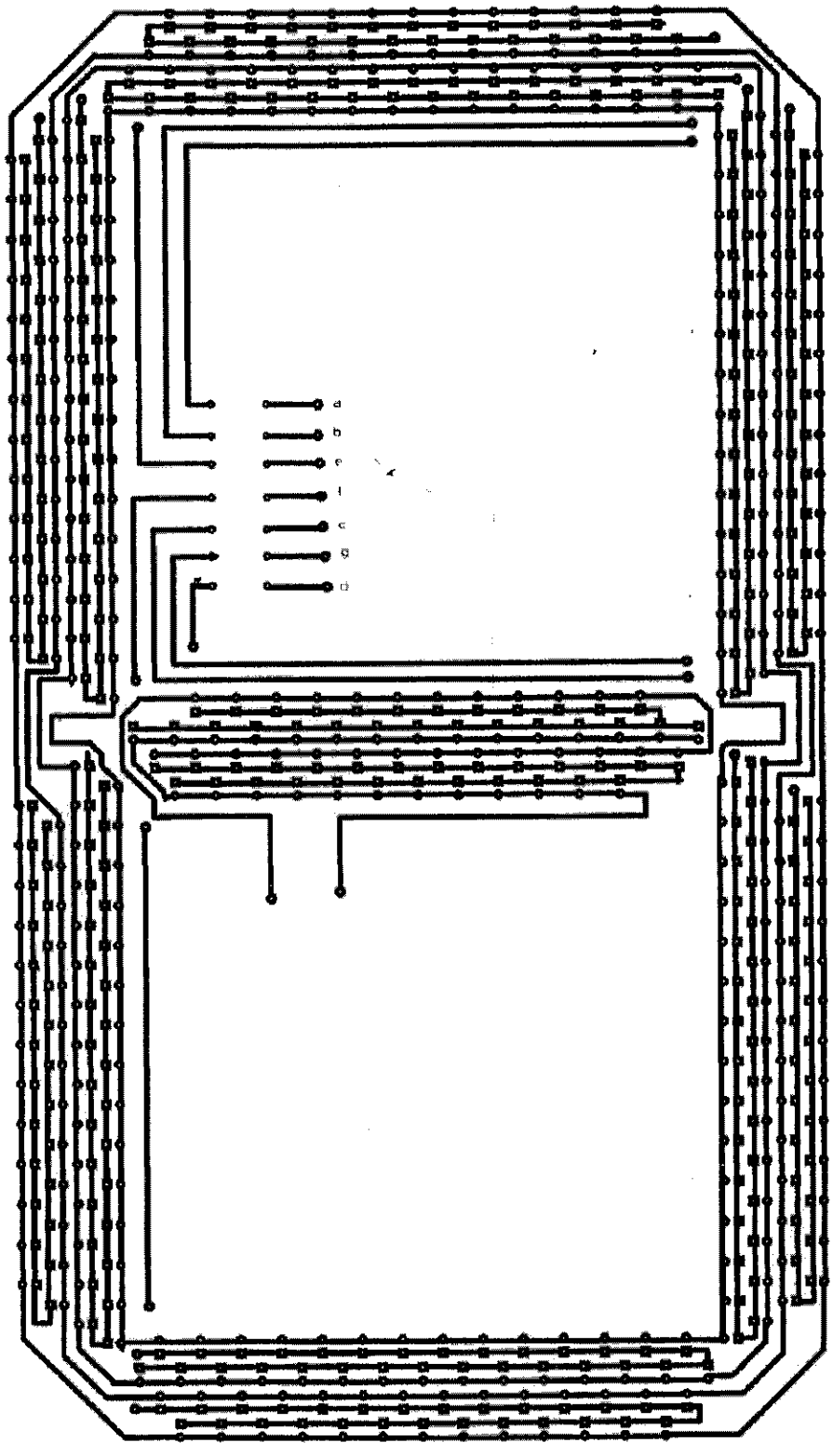
วงจรใช้งานและลายทองแดงแผ่นวงจรพิมพ์



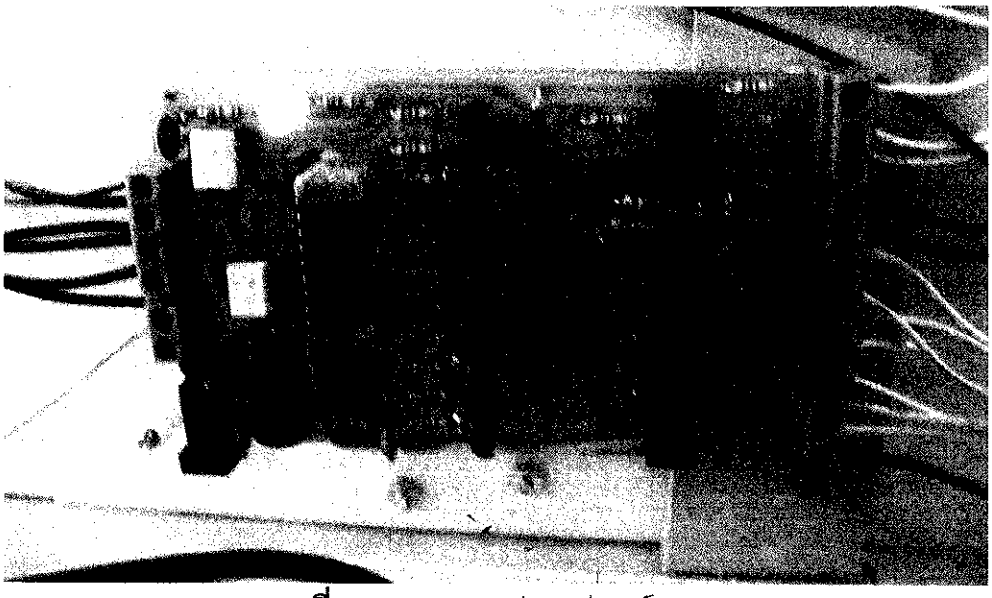
ภาพที่ ก.1วงจรรับสัญญาณไฟจราจรโดยใช้ออปโต้คัปเปลอร์แบบเอาต์พุตเป็นโฟโต้ทรานซิสเตอร์



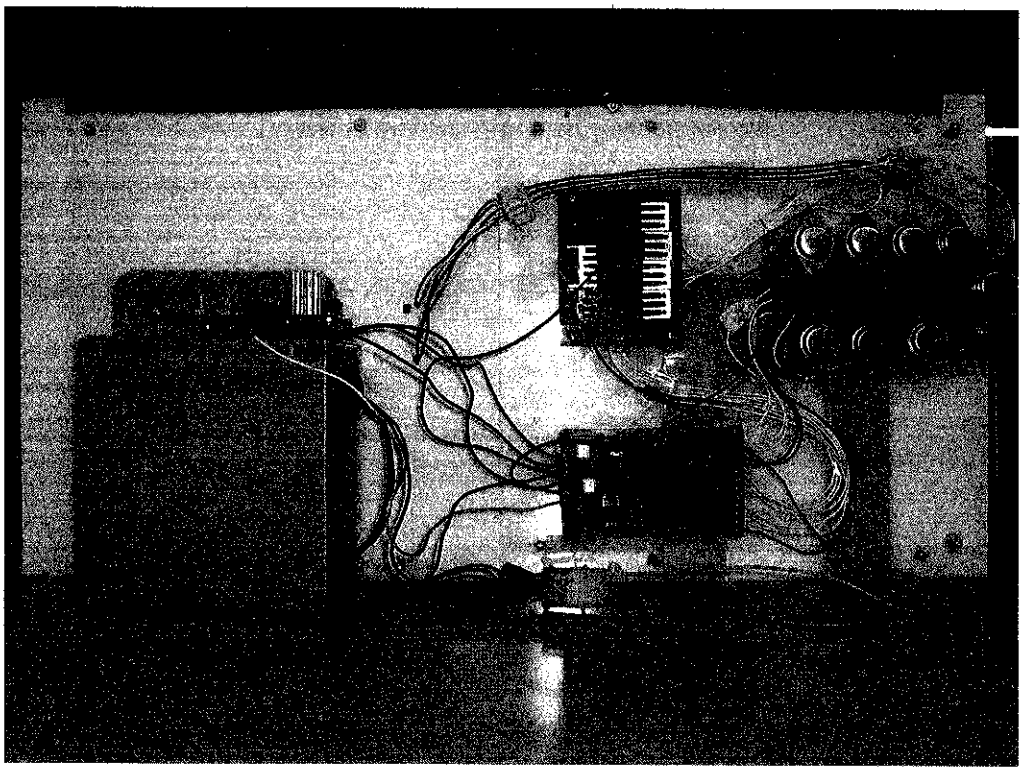
ภาพที่ ก.2 วงจรควบคุมการประมวลผล



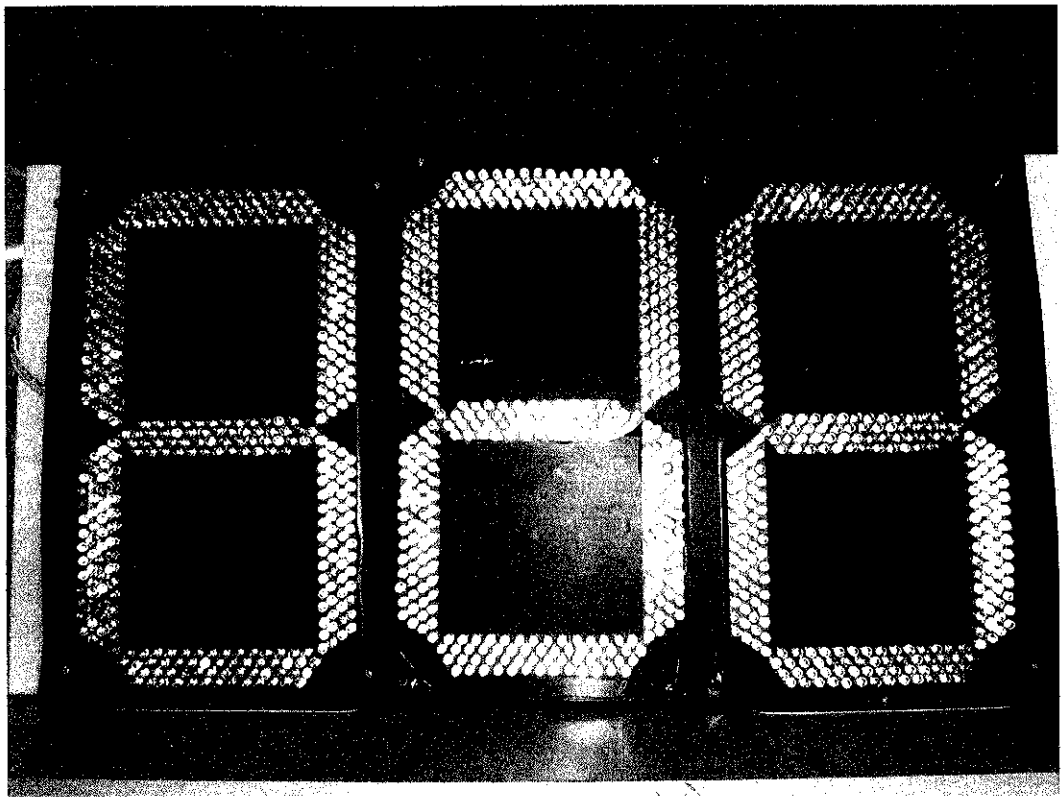
ภาพที่ ก.3 ลายวงจรแผนภาพแสงตัวเลข LED 7-segment



ภาพที่ ก.4 แสดงภาพถ่ายอุปกรณ์ควบคุม



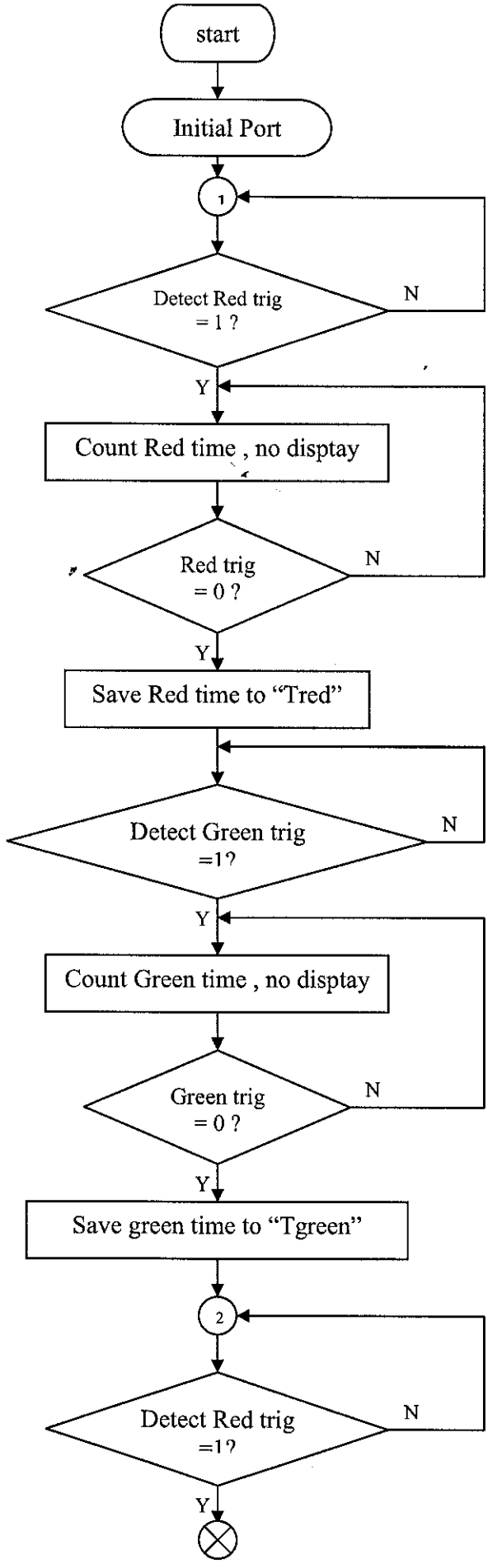
ภาพที่ ก.5 แสดงภาพถ่ายอุปกรณ์โดยรวมของระบบ

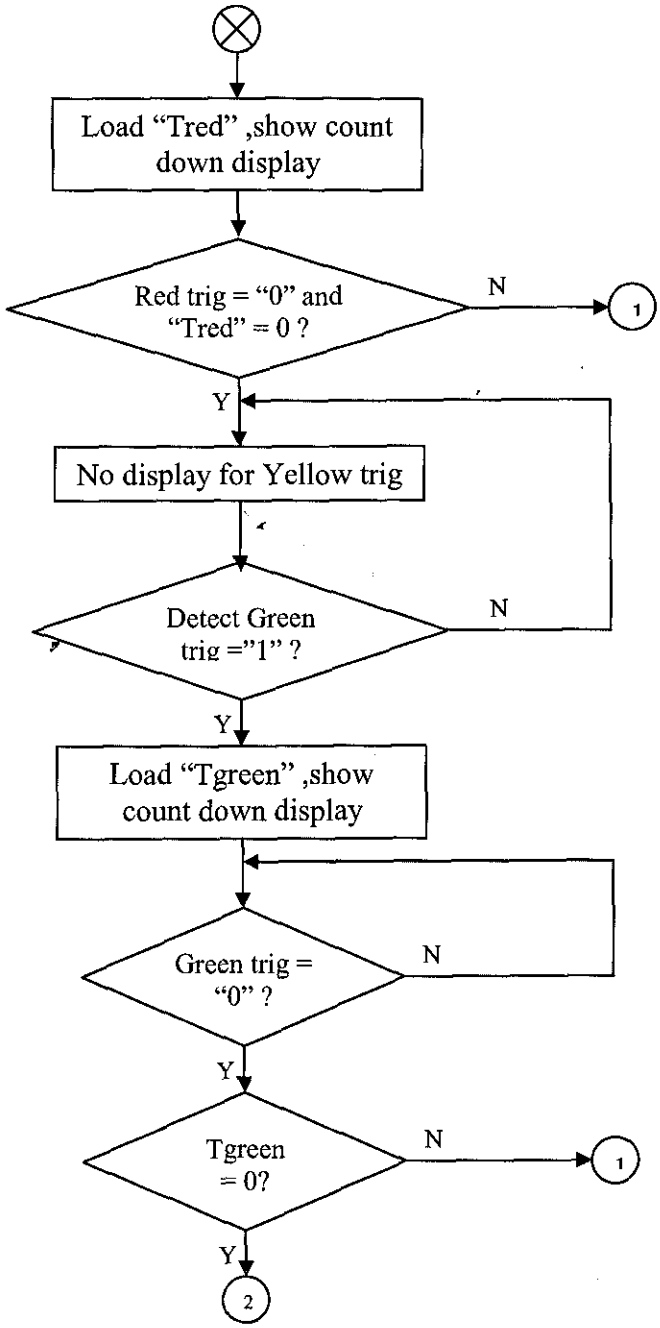


ภาพที่ ก.6 ภาพถ่ายอุปกรณ์ในส่วนแสดงผล

ภาคผนวก ข

โปรแกรมภาคควบคุมและประมวลผล


start
Initial Port
1
Detect Red trig
= 1 ?
N
Y
Count Red time , no disptay
Red trig
= 0 ?
N
Y
Save Red time to "Tred"
Detect Green trig
=1?
N
Y
Count Green time , no disptay
Green trig
= 0 ?
N
Y
Save green time to "Tgreen"
2
Detect Red trig
=1?
N
Y



ภาพที่ ข.1 Flowchart ของการเขียนโปรแกรมในภาคควบคุม

อธิบาย Flowchart

Initial Port: เริ่มจากการอินนิเชียลพอร์ตเพื่อให้พอร์ตสามารถเชื่อมต่อกับวงจรอื่นๆที่เกี่ยวข้องได้

Pre_main: เป็นส่วนที่ใช้ Detect สัญญาณไฟแดงและไฟเขียวเพื่อเก็บค่าการนับเวลาในการแสดงไฟแดงและไฟเขียว โดยในส่วนนี้จะยังไม่มีการโชว์ค่าการนับแต่อย่างใด

- เริ่มจากการ Detect สัญญาณไฟแดงว่าเข้ามาหรือยัง ถ้ามีสัญญาณไฟแดงเข้ามาก็จะ detect เจอลอจิก "1" แต่ถ้า detect เจอลอจิก "0" แสดงว่าขณะนี้ยังไม่มีสัญญาณไฟแดงเข้ามาก็ให้วนเช็คการ detect สัญญาณไฟแดงต่อไปจนกว่าจะเจอลอจิก "1"
- เมื่อ detect เจอลอจิก "1" แล้วให้เริ่มการนับเวลา นั่นคือเริ่มนับจาก 1, 2, 3..... เรื่อยไปจนกว่าจะ detect เจอลอจิก "0" เข้ามาก็จะสิ้นสุดการนับ ซึ่งในขณะนับเวลา 1 ครั้งก็จะเช็คการเกิดลอจิกลอจิก "0" ไปด้วย
- เมื่อ detect เจอลอจิก "0" ก็จะเก็บค่าการนับเวลาไว้ในตัวแปรชื่อ "Tred" ตัวอย่างเช่นเมื่อนับเวลาไปได้ 90 ซึ่งเท่ากับ 90 วินาทีแล้ว detect เจอลอจิก "0" ก็จะเก็บค่า 90 ไว้ในตัวแปร "Tred"นั่นเอง

    - จากนั้นทำเช่นเดียวกันกับสัญญาณไฟเขียวนั่นคือ

- Detect สัญญาณไฟเขียวว่าเข้ามาหรือยัง ถ้ามีสัญญาณไฟเขียวเข้ามาก็จะ detect เจอลอจิก "1" แต่ถ้า detect เจอลอจิก "0" แสดงว่าขณะนี้ยังไม่มีสัญญาณไฟเขียวเข้ามาก็ให้วนเช็คการ detect สัญญาณไฟเขียวต่อไปจนกว่าจะเจอลอจิก "1"
- เมื่อ detect เจอลอจิก "1" แล้วให้เริ่มการนับเวลา นั่นคือเริ่มนับจาก 1, 2, 3..... เรื่อยไปจนกว่าจะ detect เจอลอจิก "0" เข้ามาก็จะสิ้นสุดการนับ ซึ่งในขณะนับเวลา 1 ครั้งก็จะเช็คการเกิดลอจิกลอจิก "0" ไปด้วย
- เมื่อ detect เจอลอจิก "0" ก็จะเก็บค่าการนับเวลาไว้ในตัวแปรชื่อ "Tgreen" ตัวอย่างเช่นเมื่อนับเวลาไปได้ 100 ซึ่งเท่ากับ 100 วินาทีแล้ว detect เจอลอจิก "0" ก็จะเก็บค่า 100 ไว้ในตัวแปร "Tgreen"นั่นเอง
    - ขณะนี้เราเก็บค่าการนับของสัญญาณไฟเขียวและไฟแดงได้แล้ว

Main: เป็นส่วนสำคัญในการโชว์ค่าการนับเวลาแบบถอยหลังของสัญญาณไฟแดงและไฟเขียวที่ได้เก็บไว้ในตัวแปร “Tred” และ “Tgreen” ตามลำดับ

- เริ่มจากการ Detect สัญญาณไฟแดง(Red trig) ว่าเข้ามาหรือยัง ถ้ามีสัญญาณไฟแดง(Red trig) เข้ามาก็จะ detect เจอลอจิก “1” แต่ถ้า detect เจอลอจิก “0” แสดงว่าขณะนี้ยังไม่มีสัญญาณไฟแดง(Red trig) เข้ามาก็ให้วนเช็คการ detect สัญญาณไฟแดง(Red trig) ต่อไปจนกว่าจะเจอลอจิก “1”
- เมื่อมีลอจิก “1” เข้ามาแล้วให้โหลดค่าการนับเวลาจากตัวแปร “Tred” และทำการโชว์ค่าการนับถอยหลังออกทาง 7_segment(กลุ่มหลอด LED สีแดง) นั่นคือไฟแดงจะเริ่มโชว์ค่าการนับถอยหลังจาก 90, 89, 88, 87, 86, ….., 0
- ทำการเช็คค่าการนับว่าตรงกับเวลาในการให้สัญญาณไฟแดง(Red trig) หรือไม่โดยขณะที่ตัวแปร“Tred” มีลอจิกเท่ากับ “0” ซึ่งเท่ากับกำลังโชว์ค่า “0” บน 7_segment ให้นำค่าลอจิก “0” ของ “Tred” นำมาทำลอจิก AND กับค่าจากการ trig สัญญาณไฟแดงในขณะนั้น ถ้าค่าที่ได้จากการทำลอจิก AND เท่ากับศูนย์ แสดงว่าค่าใน “Tred” กับค่าการ trig สัญญาณไฟแดงในขณะนั้นไม่ตรงกันก็ให้วนกลับไปเก็บค่าการนับเวลาใหม่ในส่วน Pre_main เพื่อเก็บค่าการนับเวลาครั้งล่าสุดต่อไป
- ถ้าค่าที่ได้จากการทำลอจิก AND เท่ากับหนึ่งแสดงว่าค่าใน “Tred” กับค่าการ trig สัญญาณไฟแดงในขณะนั้นตรงกันก็ทำงานในส่วนถัดไปของโปรแกรมนั่นคือเช็คการ detect สัญญาณไฟเขียว (Green trig)

- จากนั้นทำเช่นเดียวกันกับสัญญาณไฟเขียวนั่นคือ

- Detect สัญญาณไฟเขียวว่าเข้ามาหรือยัง ถ้ามีสัญญาณไฟเขียว(Green trig)เข้ามาก็จะ detect เจอลอจิก “1” แต่ถ้า detect เจอลอจิก “0” แสดงว่าขณะนี้ยังไม่มีสัญญาณไฟเขียว (Green trig)เข้ามาก็ให้วนเช็คการ detect สัญญาณไฟเขียว(Green trig)ต่อไปจนกว่าจะเจอลอจิก “1”
- เมื่อมีลอจิก “1” เข้ามาแล้วให้โหลดค่าการนับเวลาจากตัวแปร “Tgreen” และทำการโชว์ค่าการนับถอยหลังออกทาง 7_segment (กลุ่มหลอด LED สีเขียว) นั่นคือไฟเขียวจะเริ่มโชว์ค่าการนับถอยหลังจาก 100, 99, 98, 97, 96, ….., 0
- ทำการเช็คค่าการนับว่าตรงกับเวลาในการให้สัญญาณไฟเขียว(Green trig) หรือไม่โดยขณะที่ตัวแปร “Tgreen” มีลอจิกเท่ากับ “0” ซึ่งเท่ากับกำลังโชว์ค่า “0” บน 7_segment ให้นำค่าลอจิก “0” ของ “Tred” นำมาทำลอจิก AND กับค่าจากการ trig สัญญาณไฟเขียวในขณะนั้น ถ้าค่าที่ได้จากการทำลอจิก AND เท่ากับศูนย์ แสดงว่าค่าใน “Tgreen” กับค่าการ

trig สัญญาณไฟเขียวในขณะนั้นไม่ตรงกันก็ให้วนกลับไปเก็บค่าการนับเวลาใหม่ในส่วน Pre_main เพื่อเก็บค่าการนับเวลาครั้งล่าสุดต่อไป

- ถ้าค่าที่ได้จากการทำลอจิก AND เท่ากับหนึ่งแสดงว่าค่าใน "Tgreen" กับค่าการ trig สัญญาณไฟเขียวในขณะนั้นตรงกันก็ทำงานในส่วนถัดไปของโปรแกรมนั่นคือวนกลับไปทำในส่วนของ Main ต่อไป

```
List  p=16F84A
; initial  port
_CP_ON          EQU   H'000F'
_CP_OFF         EQU   H'3FFF'
_PWRTE_ON       EQU   H'3FF7'
_PWRTE_OFF      EQU   H'3FFF'
_WDT_ON         EQU   H'3FFF'
_WDT_OFF        EQU   H'3FFB'
_LP_OSC         EQU   H'3FFC'
_XT_OSC         EQU   H'3FFD'
_HS_OSC         EQU   H'3FFE'
_RC_OSC         EQU   H'3FFF'

_ config _ XT_OSC &_CP_ON &_WDT_OFF

pc              equ   0x02
status          equ   0x03
porta           equ   0x05
portb           equ   0x06
cntdsp          equ   0x11
Tred            equ   0x12
Tgreen          equ   0x13
temtime         equ   0x14
temunit         equ   0x15
temten          equ   0x16
tembund         equ   0x17
temshow         equ   0x18
dhtm            equ   0x19
dly1            equ   0x1e
dly2            equ   0x1f
```

```
dly3            equ     0x20
cnt2            equ     0x21
datshow         equ     0x22

w               equ     0
f               equ     1
c               equ     0
z               equ     2
rp0             equ     5

;=============port assign  =============;
#define         greentrig       porta,4         ;input
#define         redtrig         prota,3         ;input
#define         clmn3rd         porta,2         ;output
#define         clmn2nd         porta,1         ;output
#define         clmn1st         porta,0         ;output
#define         RorG            portb,7         ;output

loopdsp         equ             h'22'
dogtime         equ             h'30'

;================================;

  org     0x000
  goto    start

start   bsf             starts,rp0
        movlw           b'00011000'
        movwf           porta
        clrf            portb
        bcf             starts,pr0
```

```
            clrf        portb
init        call        nodsp
            btfss       greentrig
            goto        init
premain     call        nodsp
            btfss       redtrig
            goto        premain     ;wait red trig
            clrf        Tred
            call        ndydsp
            btfsc       redtrig
            goto        incTred

waitgrn     call        nodsp
            btfsc       greentrig
            goto        waitgrn     ;wait green trig
            clrf        Tgreen
            call        ndydsp
incTgrn     incf        Tgreen,f
            movfw       Tgreen
            movwf       temtime
            call        cbtBCD
            call        ndydsp
            btfsc       greentrin
            goto        incTgrm

main        call        showdsp
            btfss       redtrig     ;test red light up?
            goto        main
            movfw       Tred
            movwf       temtime
showred     btfss       redtrig     ;test red already light up?
```

```
        goto    tstred      ;if not(red light down)
        call    cvtBCD
        call    dlydsp
        decfsz  temtime,f
        goto    showred
        cirf    temunit
        movlw   dogtime
        movfw   dgtm
wtyl    call    showdsp     ;wait for red light down
        btfss   redtrig
        goto    grmwt
        decfsz  dgtm,f
        goto    wtyl
        goto    wnit
tstred  bcf     status,z
        movf    temtime,f
        btfss   status,z
        goto    init

grnwt   call    nodsp  ;wait for yollow light down
        btfss   greentrig
        goto    grnwt
        movfw   Tgreen
        movwf   temtime
showgrn btfss   greentrig
        goto    tstgrn
        call    cvtBCD
        call    dlydsp
        decfsz  temtime,f
        toto    showgrn
        clrf    temunit
```

```
            movlw       dogtime
            movwf       dgtm
wtrd        call        showdsp
            btfss       greentrig
            toto        main
            decfsz      dgtm,f
            goto        wtrd
            toto        premain
tstgrn      bcf         status,z
            movf        temtime,f
            btfss       status,z
            goto        premain
            goto        main

            ;\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
            ;
cbtBCD      movfw       temtime
            movwf       temshow
            cirf        temhund
            clrf        temten
            clrf        temunit
stp1        movlw       d'100;
            subwf       temshow,w
            btfss       status,c            ;temtime<100?
            goto        stp2        ;if less than go to stp2
            incf        temhund,f
            movlw       d'100'
            subwf       temshow,f
            goto        stp1
stp2        movlw       d'10'
            subwf       temshow,w
```

```
        btfss   status,c            ;temtime<10?
        goto    srp3        ;if less than goto stp3
        incf    temten,f
        movlw   d'10'
        subwf   temshow,f
        goto    stp2
stp3    movfw   temshow
        movwf   temunit
        return

;-----------------------------------------------------------------

        dlydsp  ;return
        movlw   loopdsp
        movwf   cnt2
lp3     call    showdsp
        decfsz  cnt2,f
        goto    lp3
        return

        ndydsp  ;return
        movlw   loopdsp
        movwf   cnt2
lp6     call    nodsp
        decfsz  cnt2,f
        goto    lp6
        return

        nodsp   ;return
        movlw   d'10'
        movwf   temunit
```

```
            movwf   temten
            movwf   temhund
showdsp     bsf     clmn3rd
            movfw   temunit
            call    dattab
            movwf   datshow
            call    showseg
            call    dly10ms
            bcf     clmn3rd
            bsf     clmn2nd
            movfw   temten
            call    dattab
            movwf   datshow
            bcf     status,z        ;it temhund=0 AND
            temten =0 then no display
            movf    temhund,f   ;
            btfss   status,z
            goto    showten     ;
            bcf     status,z        ;
            movf    temten,f    ;
            btfsc   status,z        ;
            clrf    datshow     ;
showten     call    showseg
            call    dly10ms
            bcf     clmn2nd
            bsf     clmn1st
            movfw   temhund
            call    dattab
            movwf   datshow
            bcf     status,z        ;if temhund=0 then no
            display
```

```
            movf        temhund,f       ;
            btfsc       status,z        ;
            clrf        datshow         ;
            call        showseg
            call        dly10ms
            bcf         clmn1st
            return

showseg     btfsc       redtrig
            bsf         datshow,7
            movfw       datshow
            movwf       portb
            return

            ;-----------------------------------------------;

            dattab  addwf       pc,f

            retlw       b'01110111'   ;0 for common cathode 7-segment
            retlw       b'00110000'   ;1 for common cathode 7-segment
            retlw       b'01101101'   ;2 for common cathode 7-segment
            retlw       b'01110011'   ;3 for common cathode 7-segment
            retlw       b'00110011'   ;4 for common cathode 7-segment
            retlw       b'01011011'   ;5 for common cathode 7-segment
            retlw       b'01011111'   ;6 for common cathode 7-segment
            retlw       b'01100100'   ;7 for common cathode 7-segment
            retlw       b'01111111'   ;8 for common cathode 7-segment
            retlw       b'01111101'   ;9 for common cathode 7-segment
            retlw       b'00001000'   ;- for common cathode 7-segment

                ;-----------------------------------------------;
```

```
dly10ms  ;return
movlw           h'0f    ;h'08'
movwf           dly1
l1 movlw        h'd3'   ;h'48'
movwf           dly2
l2 decfsz       dly2,f
goto            l2
decfsz          dly1,f
goto            l1
return

dly             ;return
movlw           h'f0'
movwf           dly3
l3 call         dly10ms
decfsz          dly3,f
goto            l3
return

end
```

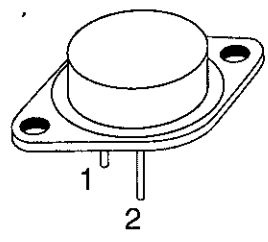
ภาคผนวก ค

Data Sheet

2N3055
MJ2955

COMPLEMENTARY SILICON POWER TRANSISTORS

- STMicroelectronics PREFERRED SALESTYPES
- COMPLEMENTARY NPN-PNP DEVICES



TO-3

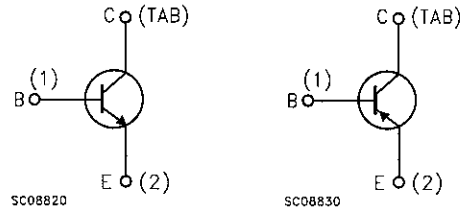
DESCRIPTION

The 2N3055 is a silicon Epitaxial-Base Planar NPN transistor mounted in Jedec TO-3 metal case.

It is intended for power switching circuits, series and shunt regulators, output stages and high fidelity amplifiers.

The complementary PNP type is MJ2955.

INTERNAL SCHEMATIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
	NPN	2N3055	
	PNP	MJ2955	
V_{CBO}	Collector-Base Voltage (I_E = 0)	100	V
V_{CER}	Collector-Emitter Voltage ($R_{BE} \leq 100\Omega$)	70	V
V_{CEO}	Collector-Emitter Voltage (I_B = 0)	60	V
V_{EBO}	Emitter-Base Voltage (I_C = 0)	7	V
I_C	Collector Current	15	A
I_B	Base Current	7	A
P_{tot}	Total Dissipation at $T_c \leq 25$ °C	115	W
T_{stg}	Storage Temperature	-65 to 200	°C
T_j	Max. Operating Junction Temperature	200	°C

For PNP types voltage and current values are negative.

THERMAL DATA

$R_{thj\text{-}case}$	Thermal Resistance Junction-case	Max	1.5	°C/W

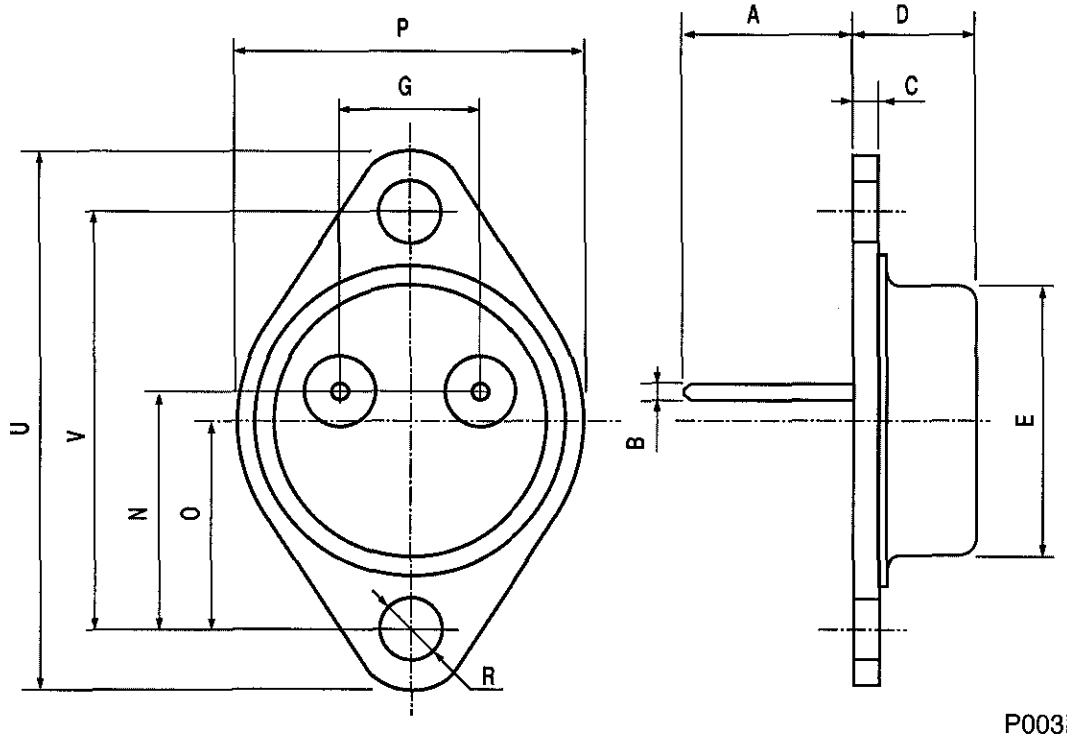
ELECTRICAL CHARACTERISTICS (T_{case} = 25 °C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
I_{CEX}	Collector Cut-off Current (V_{BE} = -1.5V)	V_{CE} = 100 V V_{CE} = 100 V T_j = 150 °C			1 5	mA mA
I_{CEO}	Collector Cut-off Current (I_B = 0)	V_{CE} = 30 V			0.7	mA
I_{EBO}	Emitter Cut-off Current (I_C = 0)	V_{EB} = 7 V			5	mA
$V_{CEO(sus)}$*	Collector-Emitter Sustaining Voltage (I_B = 0)	I_C = 200 mA	60			V
$V_{CER(sus)}$*	Collector-Emitter Sustaining Voltage (R_{BE} = 100 Ω)	I_C = 200 mA	70			V
$V_{CE(sat)}$*	Collector-Emitter Saturation Voltage	I_C = 4 A I_B = 400 mA I_C = 10 A I_B = 3.3 A			1 3	V V
V_{BE}*	Base-Emitter Voltage	I_C = 4 A V_{CE} = 4 A			1.8	V
h_{FE}*	DC Current Gain	I_C = 4 A V_{CE} = 4 A I_C = 10 A V_{CE} = 4 A	20 5		70	
f_T	Transition frequency	I_C = 0.5 A V_{CE} = 10 V	3			MHz
$I_{s/b}$*	Second Breakdown Collector Current	V_{CE} = 40 V	2.87			A

* Pulsed: Pulse duration = 300 µs, duty cycle 1.5 %
For PNP types voltage and current values are negative.

TO-3 MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A	11.00		13.10	0.433		0.516
B	0.97		1.15	0.038		0.045
C	1.50		1.65	0.059		0.065
D	8.32		8.92	0.327		0.351
E	19.00		20.00	0.748		0.787
G	10.70		11.10	0.421		0.437
N	16.50		17.20	0.649		0.677
P	25.00		26.00	0.984		1.023
R	4.00		4.09	0.157		0.161
U	38.50		39.30	1.515		1.547
V	30.00		30.30	1.187		1.193



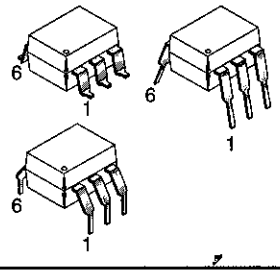
P003F

AIRCHILD
MICONDUCTOR®

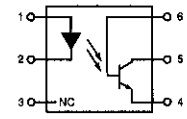
GENERAL PURPOSE 6-PIN PHOTOTRANSISTOR OPTOCOUPLERS

25 37	4N26 H11A1	4N27 H11A2	4N28 H11A3	4N35 H11A4	4N36 H11A5

WHITE PACKAGE (-M SUFFIX)

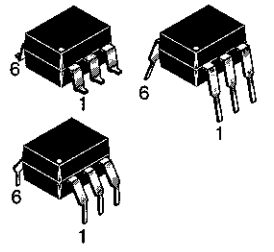


SCHEMATIC



PIN 1. ANODE
2. CATHODE
3. NO CONNECTION
4. EMITTER
5. COLLECTOR
6. BASE

BLACK PACKAGE (NO -M SUFFIX)



CRIPTION

neral purpose optocouplers consist of a gallium arsenide infrared emitting diode driving a silicon phototransistor in a 6-pin -line package.

TURES

available in white package by specifying -M suffix, eg. 4N25-M

ecognized (File # E90700)

recognized (File # 94766)

d option V for white package (e.g., 4N25V-M)

d option 300 for black package (e.g., 4N25.300)

LICATIONS

er supply regulators

tal logic inputs

oprocessor inputs

GENERAL PURPOSE 6-PIN PHOTOTRANSISTOR OPTOCOUPLERS

OLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise specified)

neter	Symbol	Value	Units
L DEVICE			
ge Temperature	T_{STG}	-55 to +150	°C
ting Temperature	T_{OPR}	-55 to +100	°C
solder temperature (see page 14 for reflow solder profiles)	T_{SOL}	260 for 10 sec	°C
Device Power Dissipation @ T_A = 25°C	P_D	250	mW
e above 25°C		3.3 (non-M), 2.94 (-M)	
TER			
verage Forward Input Current	I_F	100 (non-M), 60 (-M)	mA
se Input Voltage	V_R	6	V
rd Current - Peak (300µs, 2% Duty Cycle)	I_F(pk)	3	A
Power Dissipation @ T_A = 25°C	P_D	150 (non-M), 120 (-M)	mW
e above 25°C		2.0 (non-M), 1.41 (-M)	mW/°C
CTOR			
ctor-Emitter Voltage	V_{CEO}	30	V
ctor-Base Voltage	V_{CBO}	70	V
er-Collector Voltage	V_{ECO}	7	V
tor Power Dissipation @ T_A = 25°C	P_D	150	mW
e above 25°C		2.0 (non-M), 1.76 (-M)	mW/°C

CTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise specified)

VIDUAL COMPONENT CHARACTERISTICS

neter	Test Conditions	Symbol	Min	Typ*	Max	Unit
TER						
Forward Voltage	(I_F = 10 mA)	V_F		1.18	1.50	V
se Leakage Current	(V_R = 6.0 V)	I_R		0.001	10	µA
CTOR						
ctor-Emitter Breakdown Voltage	(I_C = 1.0 mA, I_F = 0)	BV_{CEO}	30	100		V
ctor-Base Breakdown Voltage	(I_C = 100 µA, I_F = 0)	BV_{CBO}	70	120		V
er-Collector Breakdown Voltage	(I_E = 100 µA, I_F = 0)	BV_{ECO}	7	10		V
ctor-Emitter Dark Current	(V_{CE} = 10 V, I_F = 0)	I_{CEO}		1	50	nA
ctor-Base Dark Current	(V_{CB} = 10 V)	I_{CBO}			20	nA
citance	(V_{CE} = 0 V, f = 1 MHz)	C_{CE}		8		pF

LATION CHARACTERISTICS

acteristic	Test Conditions	Symbol	Min	Typ*	Max	Units
Output Isolation Voltage	(Non '-M', Black Package) (f = 60 Hz, t = 1 min)	V_{ISO}	5300			Vac(rms)
	('-M', White Package) (f = 60 Hz, t = 1 sec)		7500			Vac(pk)
ion Resistance	(V_{I-O} = 500 VDC)	R_{ISO}	10^{11}			Ω
ion Capacitance	(V_{I-O} = &, f = 1 MHz)	C_{ISO}		0.5		pF
	('-M' White Package)			0.2	2	pF

al values at T_A = 25°C

ANSFER CHARACTERISTICS (T_A = 25°C Unless otherwise specified.)

Characteristic	Test Conditions	Symbol	Device	Min	Typ*	Max	Unit
ent Transfer Ratio, ctor to Emitter	(I_F = 10 mA, V_{CE} = 10 V)	CTR	4N35 4N36 4N37	100			%
			H11A1	50			
			H11A5	30			
			4N25 4N26 H11A2 H11A3	20			
			4N27 4N28 H11A4	10			
	(I_F = 10 mA, V_{CE} = 10 V, T_A = -55°C)		4N35 4N36 4N37	40			
	(I_F = 10 mA, V_{CE} = 10 V, T_A = +100°C)		4N35 4N36 4N37	40			
ctor-Emitter ration Voltage	(I_C = 2 mA, I_F = 50 mA)	$V_{CE\ (SAT)}$	4N25 4N26 4N27 4N28			0.5	V
	(I_C = 0.5 mA, I_F = 10 mA)		4N35 4N36 4N37			0.3	
			H11A1 H11A2 H11A3 H11A4 H11A5			0.4	
Characteristic							
Saturated -on Time	(I_F = 10 mA, V_{CC} = 10 V, R_L = 100Ω) (Fig.20)	T_{ON}	4N25 4N26 4N27 4N28 H11A1 H11A2 H11A3 H11A4 H11A5		2		µs
Saturated -on Time	(I_C = 2 mA, V_{CC} = 10 V, R_L = 100Ω) (Fig.20)	T_{ON}	4N35 4N36 4N37		2	10	µs

ANSFER CHARACTERISTICS (T_A = 25°C Unless otherwise specified.) (Continued)

haracteristic	Test Conditions	Symbol	Device	Min	Typ*	Max	Unit
off Time	(I_F = 10 mA, V_{CC} = 10 V, R_L = 100Ω) (Fig.20)	T_{OFF}	4N25 4N26 4N27 4N28 H11A1 H11A2 H11A3 H11A4 H11A5		2		µs
	(I_C = 2 mA, V_{CC} = 10 V, R_L = 100Ω) (Fig.20)		4N35 4N36 4N37		2	10	

al values at T_A = 25°C

PICAL PERFORMANCE CURVES

Fig. 1 LED Forward Voltage vs. Forward Current (Black Package)

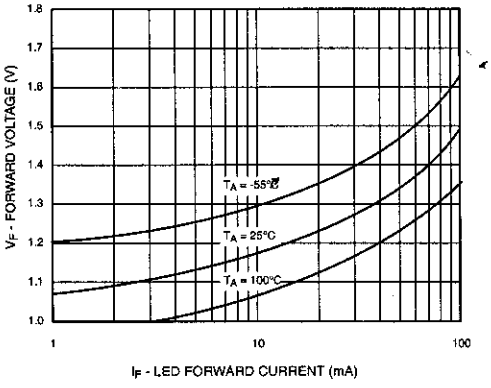


Fig. 2 LED Forward Voltage vs. Forward Current (White Package)

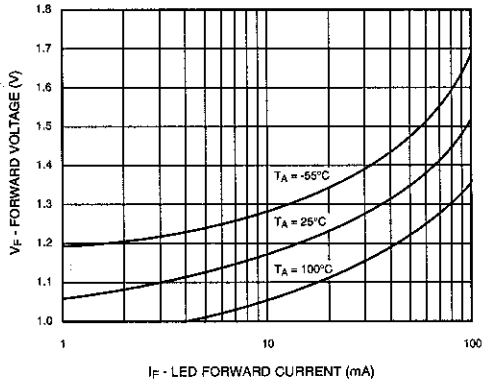


Fig.3 Normalized CTR vs. Forward Current (Black Package)

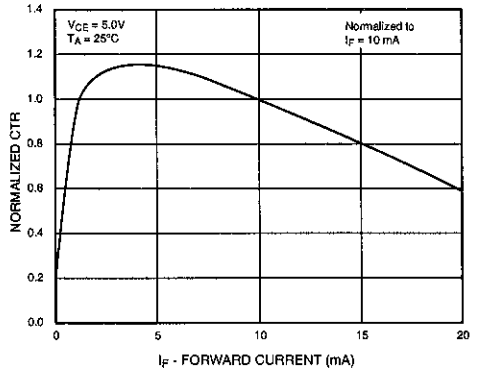


Fig.4 Normalized CTR vs. Forward Current (White Package)

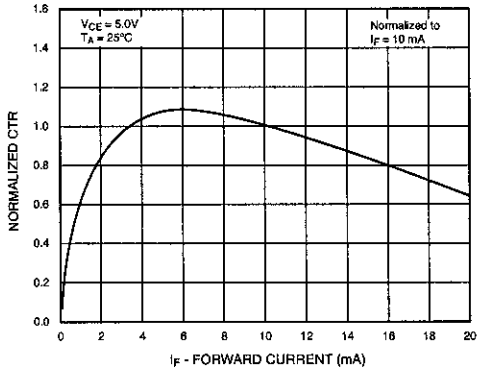


Fig. 5 Normalized CTR vs. Ambient Temperature (Black Package)

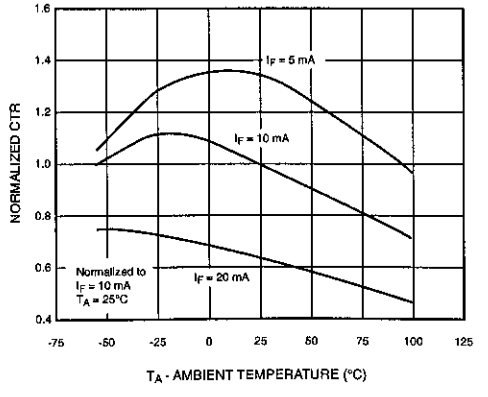


Fig. 6 Normalized CTR vs. Ambient Temperature (White Package)

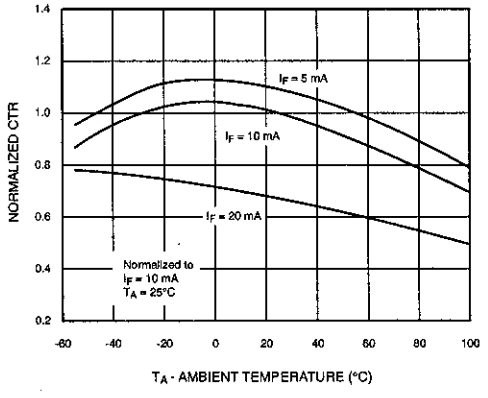


Fig. 7 CTR vs. RBE (Unsaturated) (Black Package)

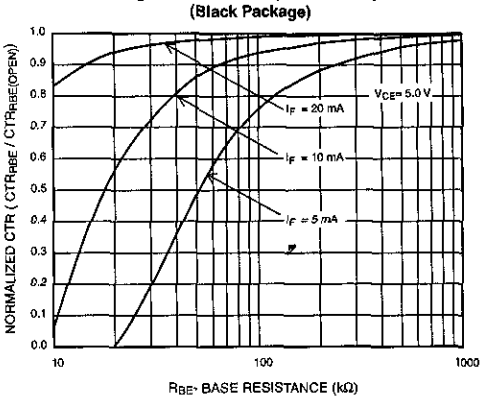


Fig. 8 CTR vs. RBE (Unsaturated) (White Package)

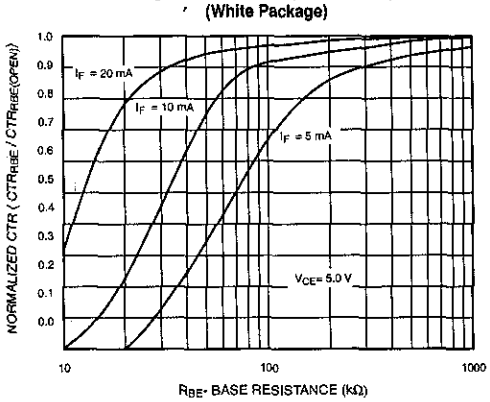


Fig. 9 CTR vs. RBE (Saturated) (Black Package)

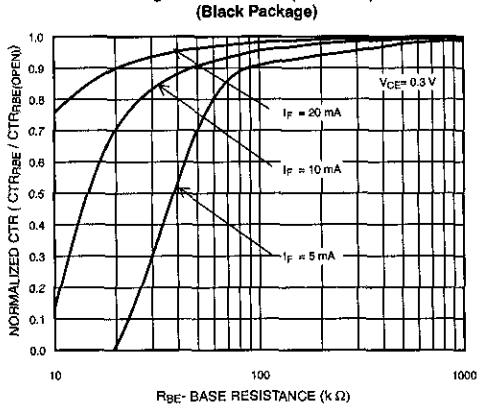


Fig. 10 CTR vs. RBE (Saturated) (White Package)

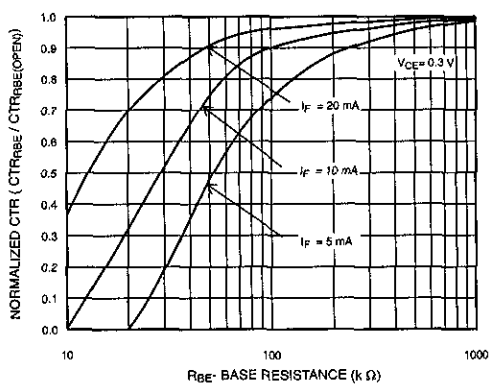


Fig. 11 Collector-Emitter Saturation Voltage vs Collector Current (Black Package)

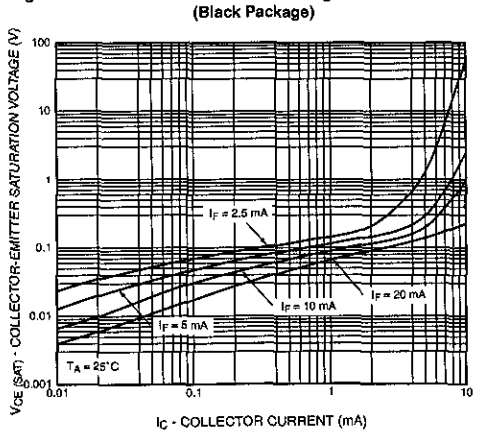


Fig. 12 Collector-Emitter Saturation Voltage vs Collector Current (White Package)

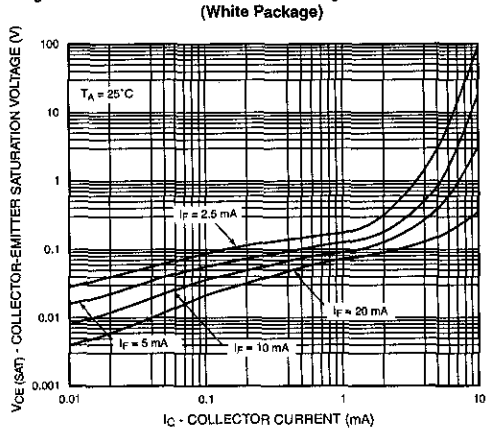


Fig. 13 Switching Speed vs. Load Resistor (Black Package)

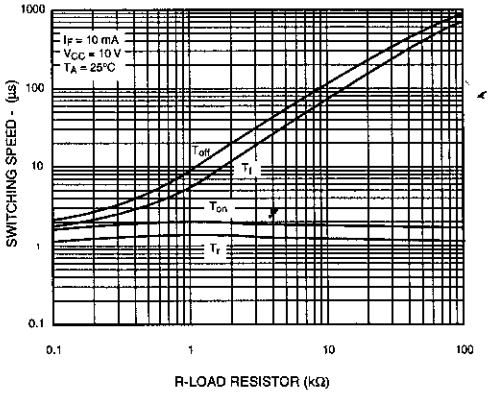


Fig. 14 Switching Speed vs. Load Resistor (White Package)

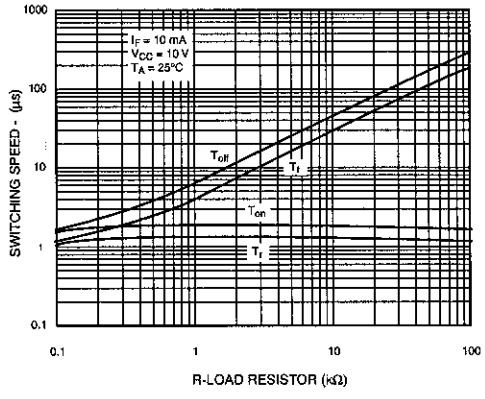


Fig. 15 Normalized t_{on} vs. R_{BE} (Black Package)

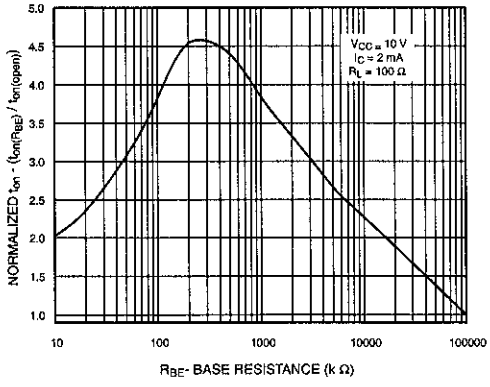


Fig. 16 Normalized t_{on} vs. R_{BE} (White Package)

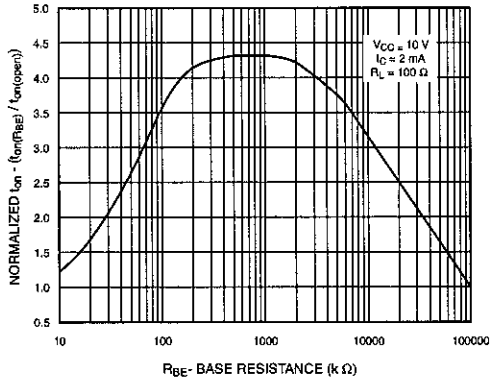


Fig. 17 Normalized t_{off} vs. R_{BE} (Black Package)

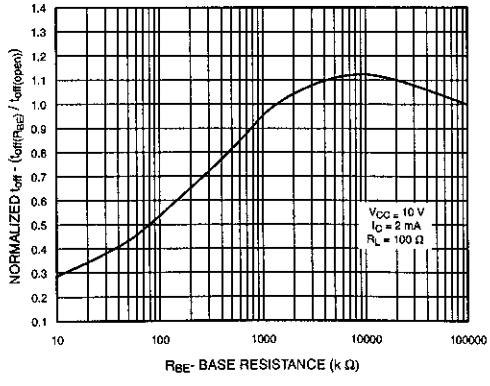


Fig. 18 Normalized t_{off} vs. R_{BE} (White Package)

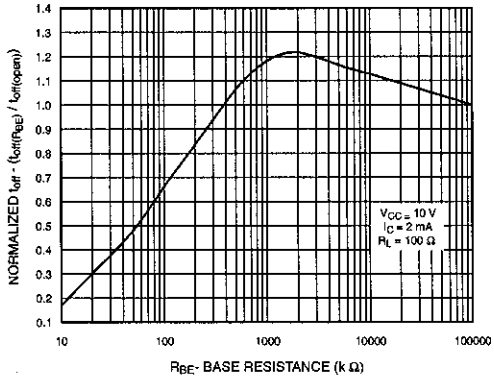
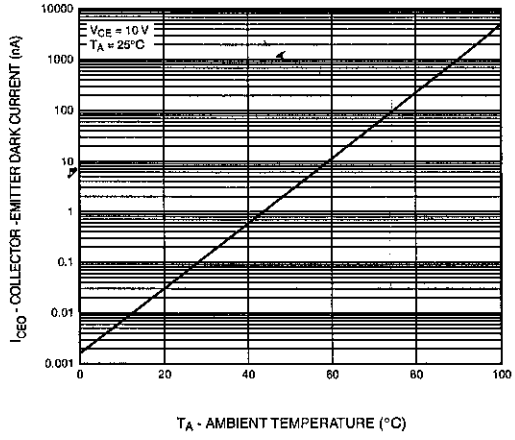


Fig. 19 Dark Current vs. Ambient Temperature



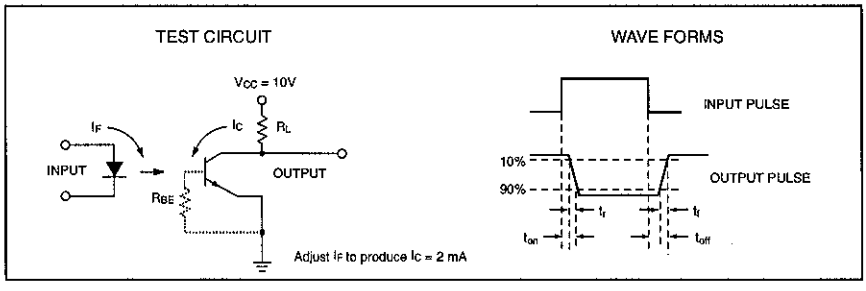
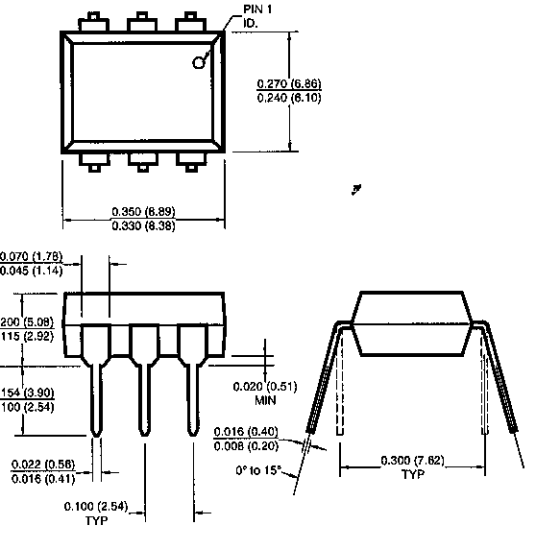


Figure 20. Switching Time Test Circuit and Waveforms

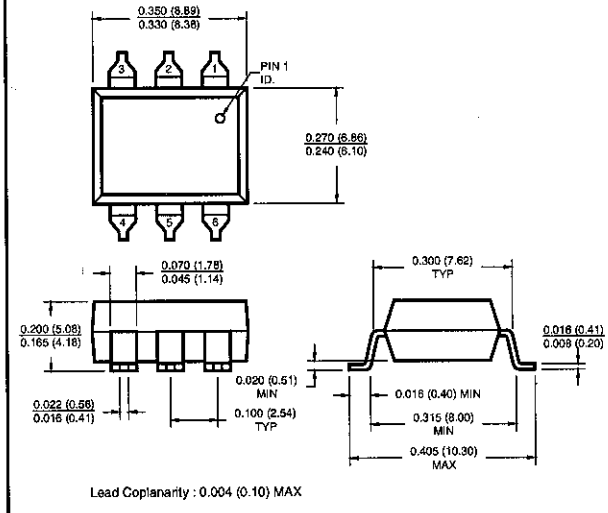
25 37	4N26 H11A1	4N27 H11A2	4N28 H11A3	4N35 H11A4	4N36 H11A5

k Package (No -M Suffix)

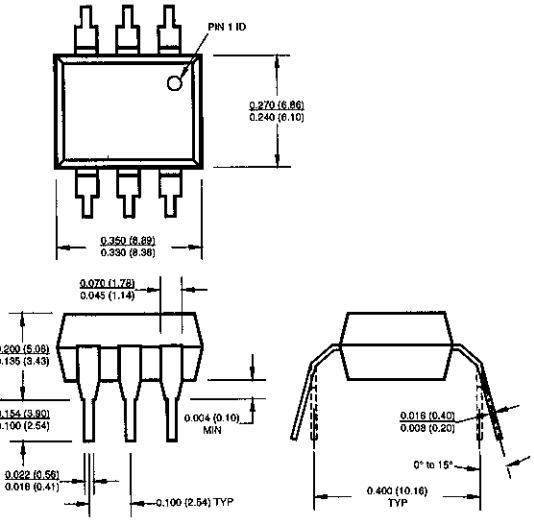
ackage Dimensions (Through Hole)



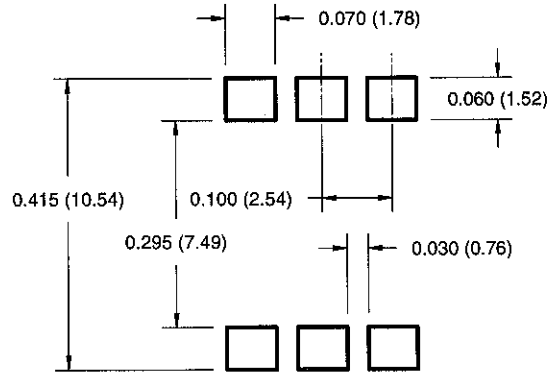
Package Dimensions (Surface Mount)



ckage Dimensions (0.4" Lead Spacing)



Recommended Pad Layout for Surface Mount Leadform



:

nensions are in inches (millimeters)

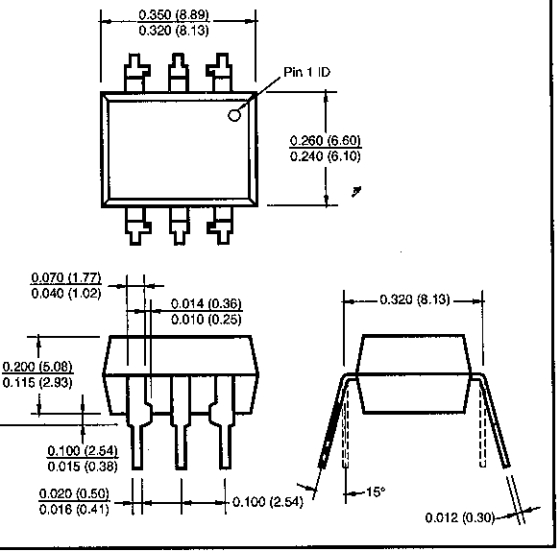
AIRCHILD
MICONDUCTOR®

GENERAL PURPOSE 6-PIN PHOTOTRANSISTOR OPTOCOUPLERS

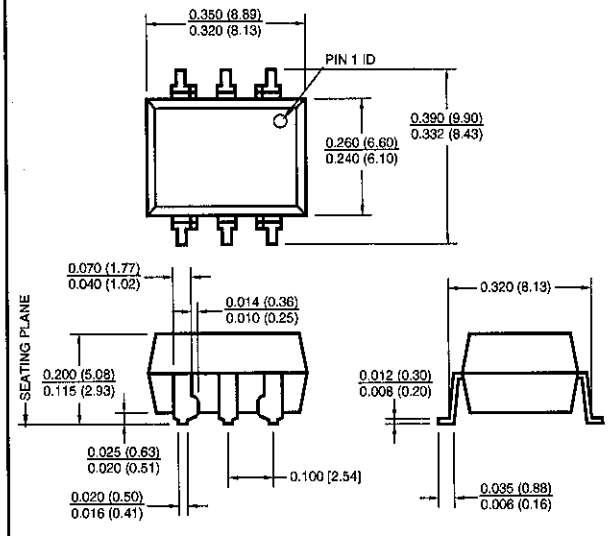
25 37	4N26 H11A1	4N27 H11A2	4N28 H11A3	4N35 H11A4	4N36 H11A5

e Package (-M Suffix)

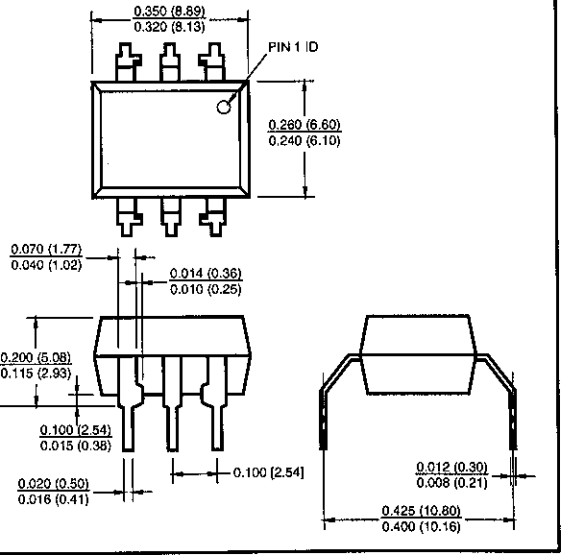
ackage Dimensions (Through Hole)



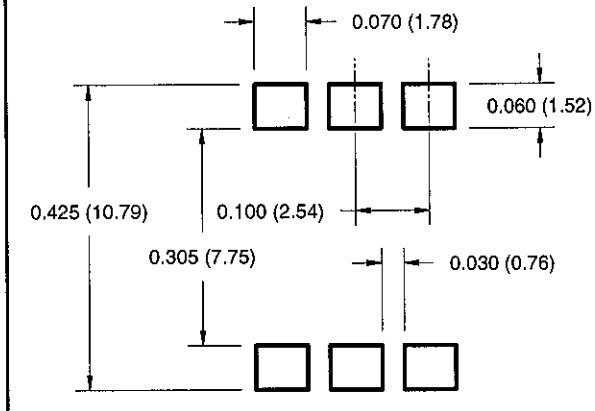
Package Dimensions (Surface Mount)



ckage Dimensions (0.4" Lead Spacing)



Recommended Pad Layout for Surface Mount Leadform



E

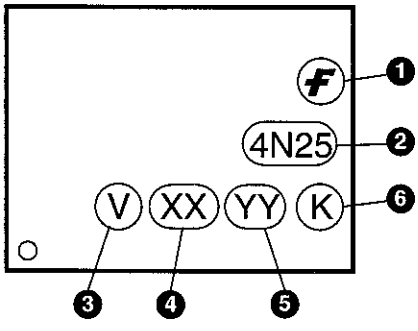
mensions are in inches (millimeters)

25	4N26	4N27	4N28	4N35	4N36
37	H11A1	H11A2	H11A3	H11A4	H11A5

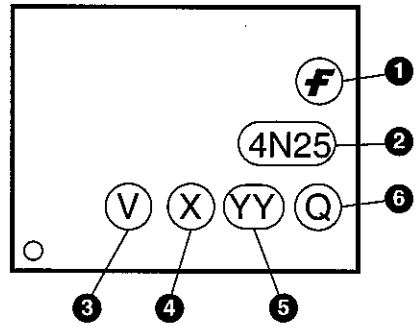
ERING INFORMATION

er Entry Identifier		
lack Package (No Suffix)	**White Package (-M Suffix)**	**Option**
.S	S	Surface Mount Lead Bend
.SD	SR2	Surface Mount; Tape and reel
.W	T	0.4" Lead Spacing
.300	V	VDE 0884
.300W	TV	VDE 0884, 0.4" Lead Spacing
.3S	SV	VDE 0884, Surface Mount
.3SD	SR2V	VDE 0884, Surface Mount, Tape & Reel

KING INFORMATION



Black Package, No Suffix

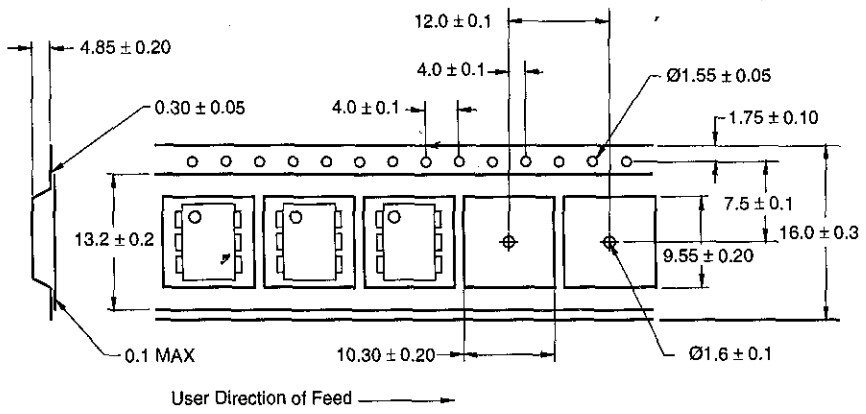


White Package, -M Suffix

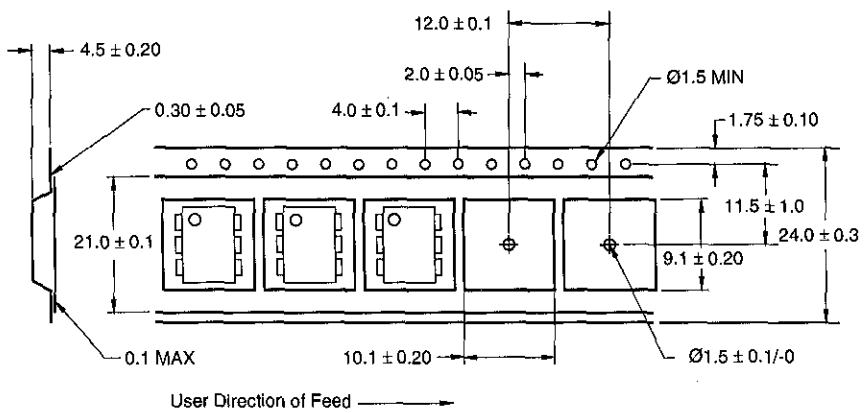
Definitions	
1	Fairchild logo
2	Device number
3	VDE mark (Note: Only appears on parts ordered with VDE option – See order entry table)
4	One or two digit year code • Two digits for black package parts, e.g., '03' • One digit for white package parts, e.g., '3'
5	Two digit work week ranging from '01' to '53'
6	Assembly package code

*Note – Parts built in the white package (M suffix) that do not have the 'V' option (see definition 3 above) that are marked with date code '325' or earlier are marked in the portrait format.

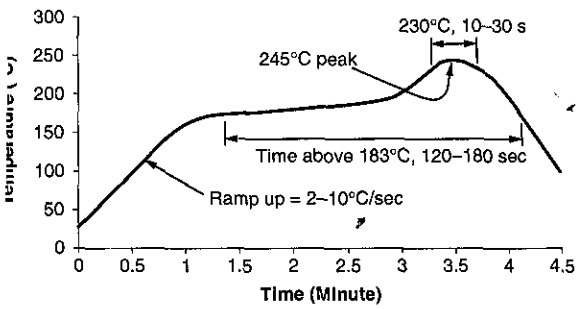
arrier Tape Specifications (Black Package, No Suffix)



Carrier Tape Specifications (White Package, -M Suffix)

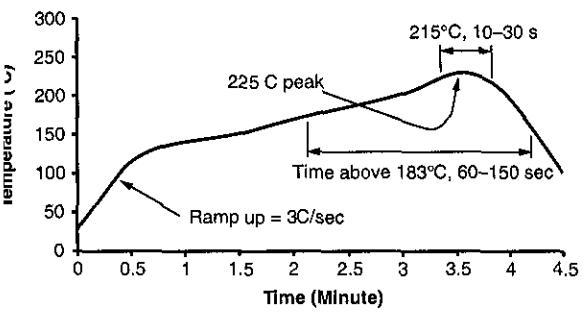


w Profile (White Package, -M Suffix)



- Peak reflow temperature: 245°C (package surface temperature)
- Time of temperature higher than 183°C for 120–180 seconds
- One time soldering reflow is recommended

w Profile (Black Package, No Suffix)



- Peak reflow temperature: 225°C (package surface temperature)
- Time of temperature higher than 183°C for 60–150 seconds
- One time soldering reflow is recommended

ประวัติผู้เขียน

นางสาวจันทร์จริยา ลุนละวงค์ สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย จากโรงเรียนนาดูนประชาสรรพ์ ต. นาดูน อ. นาดูน จ. มหาสารคาม เมื่อ ปีการศึกษา 2543 ปัจจุบันกำลังศึกษาอยู่ในระดับอุดมศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี อ. เมือง จ. นครราชสีมา

นางสาวนันทิดา ใจคำนึงคิด สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย จากโรงเรียนเชียงรายวิทยาคม อ.เมือง จ.เชียงราย เมื่อ พ.ศ. 2543 ปัจจุบันกำลังศึกษาอยู่ในระดับอุดมศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี อ. เมือง จ. นครราชสีมา

นางสาวพัฒนา แสงซอน สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย จากโรงเรียนดอยสะเก็ด ต.เชิงดอย อ.ดอยสะเก็ด จ.เชียงใหม่ เมื่อ พ.ศ. 2543 กำลังศึกษาอยู่ในระดับอุดมศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี อ. เมือง จ. นครราชสีมา ปัจจุบันได้เสียชีวิตด้วยอุบัติเหตุอย่างกะทันหัน ขณะที่ได้ร่วมทำโครงงานสำเร็จแล้ว