ระบบควบคุมสัญญาณไฟจราจรด้วยไมโครคอนโทรลเลอร์

TRAFFIC LIGHT CONTROLLER SYSTEM BY USING MICROCONTROLLER

โดย

นาย ปวริศ วรรณดีจรัสศรี

นาย ไพศาล ทองเส้ง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ระบบควบคุมสัญญาณไฟจราจรด้วยไมโครคอนโทรลเลอร์

TRAFFIC LIGHT CONTROLLER SYSTEM BY USING MICROCONTROLLER

โดย

นาย ปวริศ วรรณดีจรัสศรี 43010262

นาย ไพศาล ทองเส้ง 43010317

อาจารย์ที่ปรึกษา

รศ.สมยศ จุณณะปิยะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมสัญญาณไฟจราจรด้วยไมโครคอนโทรลเลอร์

TRAFFIC LIGHT CONTROLLER SYSTEM BY USING MICROCONTROLLER

ผู้จัดทำ

1. นายปวริศ วรรณดีจรัสศรี 43010269
2. นายไพศาล ทองเส้ง 43010317

..... อาจารย์ที่ปรึกษา

(รศ.สมยศ จุณณะปิยะ)

ระบบควบคุมสัญญาณไฟจราจรด้วยไมโครคอนโทรลเลอร์

TRAFFIC LIGHT CONTROLLER SYSTEM BY USING MICROCONTROLLER

โดย 1. นายปวริศ วรรณดีจรัสศรี 43010262

2. นายไพศาล ทองเส้ง 43010302

อาจารย์ที่ปรึกษา รศ.สมยศ จุณณะปิยะ

บทคัดย่อ

ระบบควบคุมสัญญาณไฟจราจรเป็นโครงงานที่มีหลักการทำงานด้วยไมโครคอนโทรลเลอร์เป็นตัวควบคุมสัญญาณไฟจราจร บริเวณ 4 แยกจำนวน 4 จุดซึ่งเชื่อมต่อกันเพื่อลดปัญหาการจราจรเนื่องจากเมื่อมีการเชื่อมต่อกันระหว่างแต่ละจุดจะทำให้สามารถทราบสภาวะการจราจรล่วงหน้า และสามารถส่งสัญญาณไปยังสี่แยกถัดไปได้ ซึ่งจะทำให้การจราจรมีความคล่องตัวมากขึ้น

Abstract

The traffic light controller system is operated by using microcontroller for four intersection. To reduce the traffic because of each connecting can make a trouble. Traffic light system will know where the heavy traffic is and make a decision for flexible.

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

เนื่องจากในปัจจุบัน เป็นที่รู้กันดีว่าปัญหาการจราจรนั้นเป็นปัญหาสำคัญอย่างหนึ่งที่ต้องมีการแก้ไขอย่างเร่งด่วนที่สุดเพราะนับวันปัญหายิ่งจะทวีความรุนแรงเพิ่มมากขึ้นทุกวัน ระบบการควบคุมการจราจรที่มีประสิทธิภาพจึงเป็นสิ่งสำคัญและจำเป็นอย่างยิ่ง เพื่อช่วยบรรเทาปัญหาการจราจรที่เป็นอยู่ทุกวันนี้ให้บรรเทาลง แต่สัญญาณไฟจราจรที่ใช้กันอยู่ในปัจจุบันนั้นเป็นระบบขนาดเล็กซึ่งสามารถควบคุมได้เพียงสี่แยกเดียวเท่านั้น โดยใช้ตำรวจจราจรเป็นผู้ควบคุมหรือทำงานตามโปรแกรมอัตโนมัติที่ตั้งไว้ ซึ่งปัญหาที่เกิดขึ้นนั้นคือระบบจะทำการควบคุมได้เฉพาะสี่แยก สี่แยกหนึ่งเท่านั้น โดยไม่มีการประสานงานกับสี่แยกถัดไปทำให้เกิดการจราจรติดขัดบนท้องถนนจนกลายเป็นปัญหาการจราจรที่เห็นกันอยู่ในปัจจุบัน ระบบควบคุมสัญญาณไฟที่จัดทำขึ้นมานั้นมีวัตถุประสงค์เพื่อให้การทำงานเป็นไปอย่างมีประสิทธิภาพและสะดวกรวดเร็วกว่าระบบขนาดเล็กที่ใช้อยู่ทุกวันนี้ แม้ว่าจะดีไม่เท่ากับระบบขนาดใหญ่ที่มีอยู่แล้วก็ตาม แต่เป็นที่น่าเสียดายที่ในประเทศปัจจุบันไม่สามารถทำระบบควบคุมสัญญาณไฟให้เป็นระบบขนาดใหญ่ได้ครอบคลุมพื้นที่ทั่วประเทศ เนื่องจากประสบปัญหาทางด้านเครื่องมือ อุปกรณ์ หรือแม้แต่บุคลากรที่มีความรู้ความสามารถ ความเชี่ยวชาญทางด้านวิศวกรรมจราจร ทั้งยังค่าใช้จ่ายในการบำรุงรักษาซ่อมแซมที่ค่อนข้างสูงมากทีเดียว

ระบบสัญญาณไฟจราจรที่ประดิษฐ์ขึ้นมานั้นมีระบบการทำงานที่เพิ่มเติมขึ้นจากระบบขนาดเล็กที่ใช้ในปัจจุบันโดยสามารถควบคุมการจราจรได้ถึง 4 สี่แยกเพื่อช่วยลดปัญหาจราจรแออัดที่จะเกิดขึ้นในการที่สามารถควบคุมได้เพียงสี่แยกเดียว โดยสี่แยกที่1 เมื่อทำการปล่อยรถไปยังสี่แยกที่ 2 แล้ว ระบบควบคุมจะทำการส่งสัญญาณไปแจ้งแก่สี่แยกที่ 2 ว่าได้ทำการปล่อยรถออกไปแล้ว ให้สี่แยกที่ 2 ทำการนับเวลาที่รถจะใช้วิ่งระหว่างสี่แยกที่ 1 ไปยังสี่แยกที่ 2 แล้วทำการเปิดสัญญาณไฟเขียว เพื่อให้รถสามารถวิ่งผ่านทะลุไป โดยไม่ติดสัญญาณไฟที่สี่แยกนี้อีก ส่วนสี่แยกที่ 2 ก็จะทำการแจ้งไปให้สี่แยกที่ 3 ทราบเช่นกันว่ามีการปล่อยรถออกไปแล้วและสี่แยกที่ 3 จะต้องนับเวลา เพื่อเตรียมเปิดสัญญาณไฟเขียว เพื่อให้รถวิ่งผ่านไปเช่นกัน ส่วนสี่แยกที่ 4 ก็จะทำการตามหลักการเดียวกันนี้ ทั้งระบบควบคุมสัญญาณไฟนี้ยังทำงานสอดคล้องกันทุกแยกในการเปิดสัญญาณไฟให้รถวิ่งผ่านตลอด อุปกรณ์ที่ใช้ในการควบคุมสัญญาณไฟจราจรที่ประดิษฐ์ขึ้นนั้นมีคุณสมบัติเหมือนกับระบบควบคุมไฟจราจรที่ใช้กันอยู่ทั่วไป รวมทั้งการใช้สัญญาณนาฬิกาเข้ามาช่วยควบคุมทางด้านระบบเวลา เพื่อให้อุปกรณ์สามารถทำงานได้ตามเวลาที่เรากำหนดไว้ ในกรณีที่ไม่มีคนคอยควบคุมอุปกรณ์

บทที่ 2
ทฤษฎีและหลักการ

2.1 ทฤษฎีและหลักการทำงานของสัญญาณไฟจราจร

สัญญาณไฟจราจรที่ใช้กันอยู่ทั่วไปนั้นมีทั้งหมด 3 ชนิด ได้แก่ สัญญาณไฟแดง สัญญาณไฟเขียวและสัญญาณไฟเหลือง ซึ่งทั้ง 3 สีนั้นจะใช้ในความหมายที่แตกต่างกันดังนี้

สัญญาณไฟแดง จะหมายถึง ให้รถหยุด

สัญญาณไฟเขียว จะหมายถึง ให้รถวิ่งได้

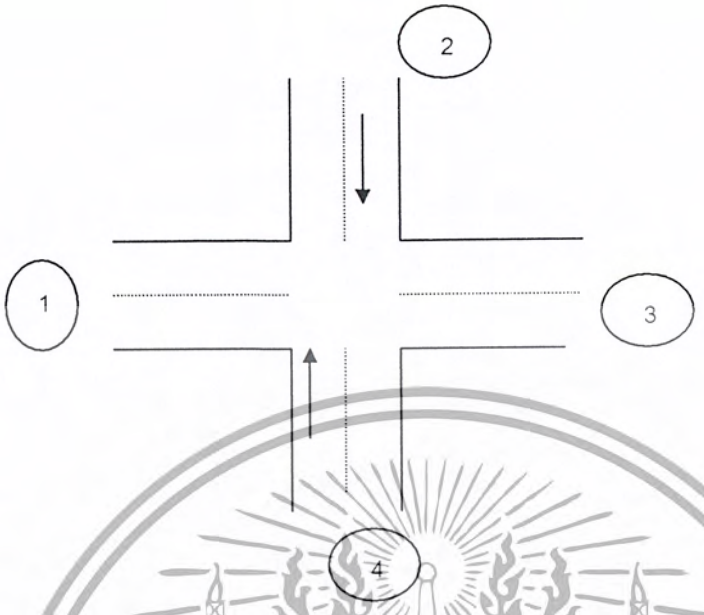
สัญญาณไฟเหลือง จะหมายถึง ให้รถเตรียมชะลอความเร็ว เพื่อหยุดรถ

สำหรับการทำงานทั่วไปของระบบการควบคุมสัญญาณไฟจราจรปกตินั้น ส่วนใหญ่จะเป็นระบบการควบคุมขนาดเล็กซึ่งสามารถปล่อยรถได้เพียง 4 แยกเดียว และสำหรับส่วนของอุปกรณ์การรับข้อมูลนั้น จะประกอบไปด้วยสัญญาณไฟจราจร 8 แบบ แต่ละแบบจะมีลักษณะดังต่อไปนี้

1. สวิทซ์ให้สัญญาณจราจรแบบที่ 1 สัญญาณไฟจราจรที่ได้จะมีลักษณะดังภาพ

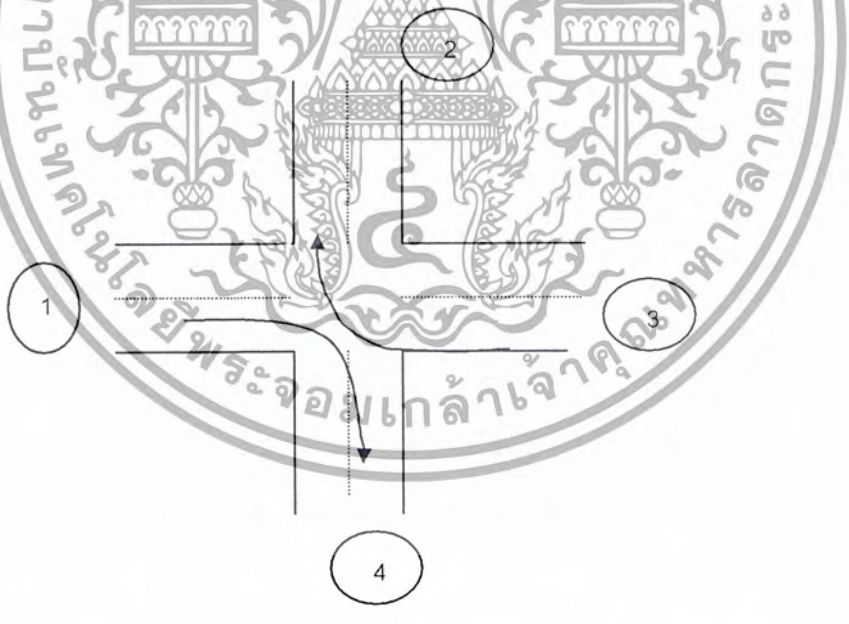
รูปที่ 2.1 แสดงรูปสัญญาณไฟจราจรแบบที่ 1

2. สวิทช์ให้สัญญาณจราจรแบบที่ 2 สัญญาณไฟจราจรที่ได้จะมีลักษณะดังภาพ



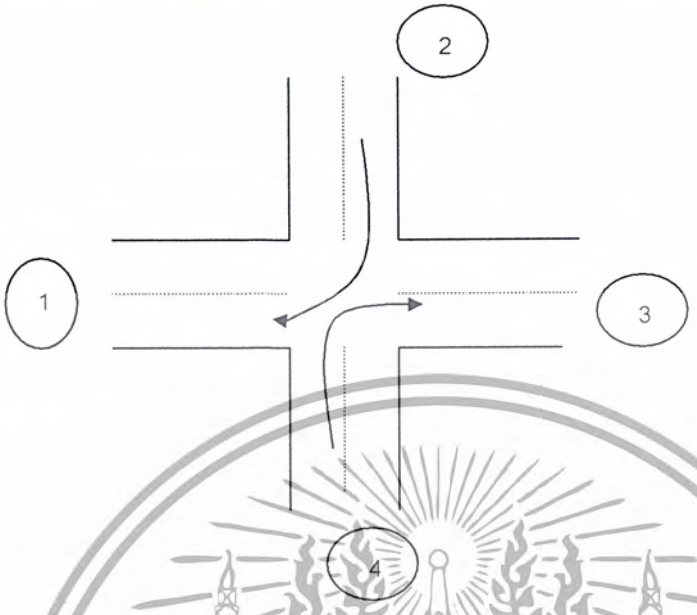
รูปที่ 2.2 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 2

3. สวิทช์ให้สัญญาณจราจรแบบที่ 3 สัญญาณไฟจราจรที่ได้จะมีลักษณะดังภาพ



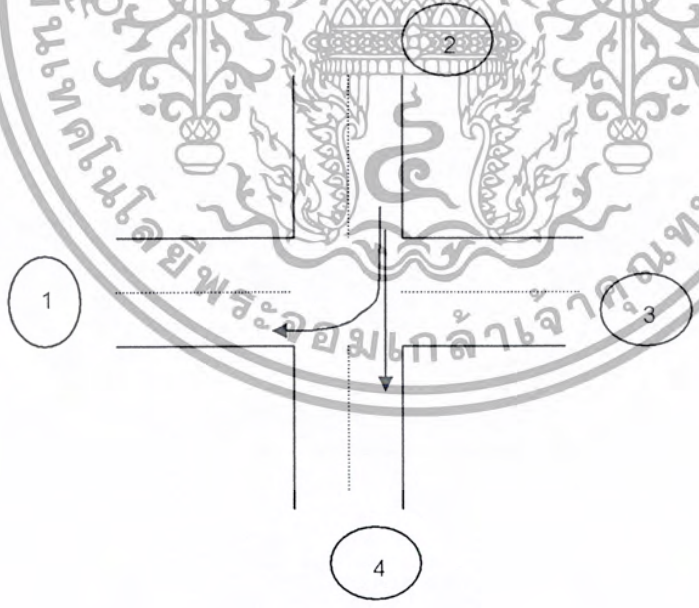
รูปที่ 2.3 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 3

4. สวิทช์ให้สัญญาณจราจรแบบที่ 4 สัญญาณไฟจราจรที่ได้จะมีลักษณะดังภาพ



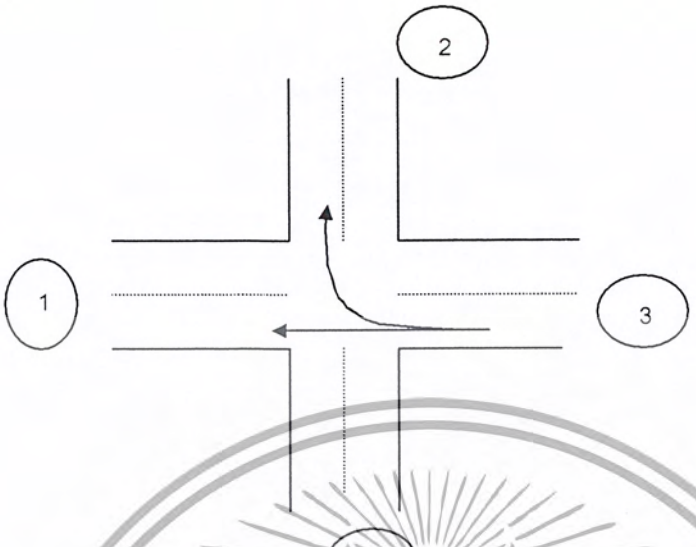
รูปที่ 2.4 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 4

5. สวิทช์ให้สัญญาณจราจรแบบที่ 5 สัญญาณไฟจราจรที่ได้จะมีลักษณะดังภาพ



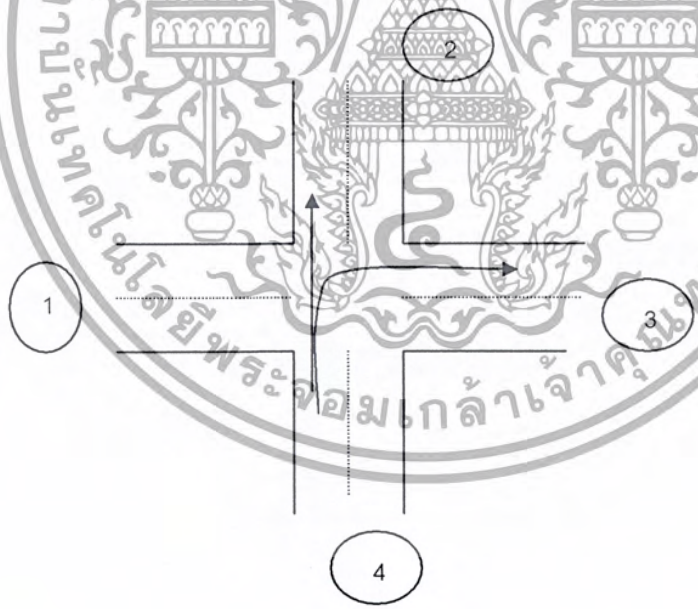
รูปที่ 2.5 แสดงรูปแบบไฟสัญญาณไฟจราจรแบบที่ 5

6. สวิทซ์ให้สัญญาณจราจรแบบที่ 6 สัญญาณไฟจราจรที่ได้จะมีลักษณะดังภาพ



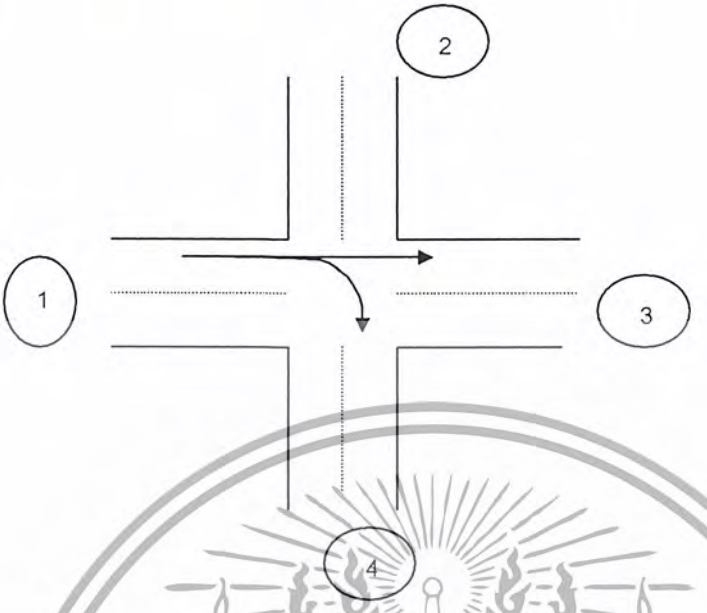
รูปที่ 2.6 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 6

7. สวิทซ์ให้สัญญาณจราจรแบบที่ 7 สัญญาณไฟจราจรที่ได้จะมีลักษณะดังภาพ



รูปที่ 2.7 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 7

8. สวิทซ์ให้สัญญาณจราจรแบบที่ 8 สัญญาณไฟจราจรที่ได้จะมีลักษณะดังภาพ



รูปที่ 2.8 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 8

2.2 ทฤษฎีและหลักการทำงานของระบบการควบคุมสัญญาณไฟจราจรที่ประดิษฐ์ขึ้น

ระบบการควบคุมการจราจรที่ทำขึ้นนั้นเป็นระบบที่การทำงานทุก 4 แยกมีความสอดคล้องกัน มีการส่งผ่านด้วยพอร์ท RS485 มีการแบ่งการทำงานออกเป็น 2 โหมด ดังนี้

โหมดแรก การทำงานของทุกแยกจะเป็นอิสระต่อกัน แต่ละแยกจะไม่ขึ้นต่อกันและมีการตั้งโปรแกรมอัตโนมัติโดยการใช้สัญญาณนาฬิกาเข้ามาช่วยในการควบคุมระบบเวลาเพื่อให้อุปกรณ์สามารถทำงานได้ตามเวลาที่เราตั้งโปรแกรมเอาไว้ ซึ่งจะช่วยเพิ่มความสะดวกในการใช้งานและทำให้เราสามารถไปใช้โปรแกรมอื่นๆ ได้ในเวลาที่เปลี่ยนแปลงไป

โหมดที่สอง การทำงานของสี่แยกแต่ละแยกนั้น จะมีการเชื่อมต่อกัน โดยมีตัวคอยส่งสัญญาณแจ้งให้แยกถัดไปทราบว่า ได้ทำการปล่อยรถออกไปแล้ว เพื่อให้แยกถัดไปเตรียมนับเวลาที่รถจะแล่นจากแยกแรกมาจนถึงแยกนี้ แล้วทำการเปิดสัญญาณไฟเขียวให้รถวิ่งผ่านไป ตัวควบคุมหลักนี้จะทำการส่งสัญญาณแจ้งไปยังแยกต่างๆ ข้างหน้าได้ถึง 4 แยกให้รับรู้และสามารถทำงานให้สอดคล้องกันกับแยกแรกในการเปิดสัญญาณไฟให้รถวิ่งผ่านโดยไม่ติดขัด

สำหรับอุปกรณ์ควบคุมสัญญาณไฟจราจร ชุดที่ได้ทำการประดิษฐ์ขึ้นนี้นั้นสามารถทำการควบคุมเชื่อมต่อกันได้ 4 แยกภายในอุปกรณ์ควบคุมสัญญาณไฟจราจรนั้นจะมีส่วนประกอบต่างๆ ที่ใช้ควบคุมการทำงานแบ่งเป็นส่วนควบคุมและประมวลผลกับส่วนที่เป็นส่วนแสดงผลโดยทำการพัฒนาจากอุปกรณ์ควบคุมระบบแบบเดิมให้มีประสิทธิภาพสูงขึ้น คือ ทำการเพิ่มส่วนที่สามารถทำการควบคุมเชื่อมต่อไปยังแยกถัดไปข้างหน้าได้ โดยมีอุปกรณ์ควบคุมหลัก 1 ตัว (Master Controller)เป็นตัวคอยควบคุม โดยส่งสัญญาณควบคุมจากชุดควบคุมผ่านอุปกรณ์อินเทอร์เฟส ไปยังอุปกรณ์ควบคุมซึ่งอยู่แยกถัดไปข้างหน้า ในขณะที่ตัว

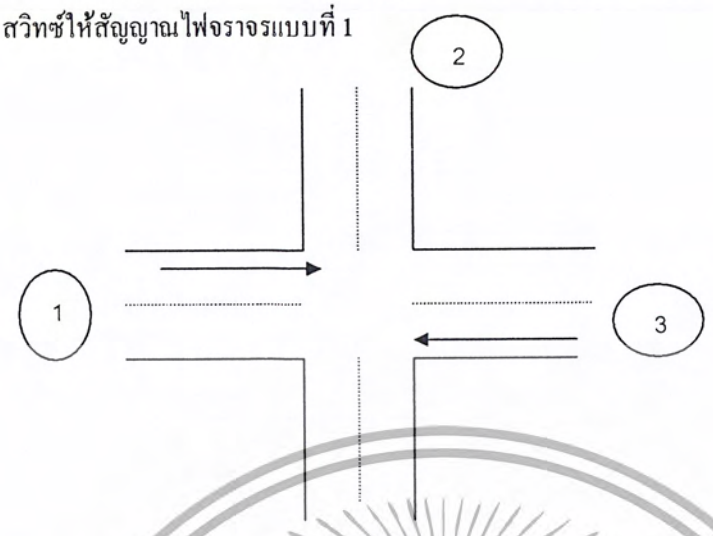
อุปกรณ์ยังสามารถทำงานได้เหมือนที่ในระบบเดิมซึ่งอุปกรณ์ควบคุมสัญญาณจราจรที่ใช้อยู่ในปัจจุบันทำได้อยู่ นอกจากนี้อุปกรณ์ยังสามารถแสดงผลให้ผู้ควบคุมทราบว่า ขณะนี้สัญญาณกำลังทำงานในโหมดใดอยู่ แสดงถึงวันเวลาได้ เพื่อให้การทำงานสอดคล้องกับสภาพการจราจรในขณะที่ทำงานอยู่ในโหมดอัตโนมัติ ซึ่งค่าเวลาเหล่านี้สามารถแสงดผลได้ทางจอภาพ

การออกแบบระบบควบคุมสัญญาณไฟจราจรนี้สามารถทำการควบคุมสัญญาณไฟจราจรได้ในแต่ละแยก และยังสามารถทำการเชื่อมต่อกันได้ทั้งหมด 4 สี่แยกเพื่อให้การควบคุมสัญญาณไฟจราจรเป็นไปอย่างต่อเนื่อง โดยการควบคุมจากตัวควบคุมที่มีอยู่ในแต่ละสี่แยกและรับสัญญาณควบคุมจากตัวควบคุมหลักซึ่งจะทำการส่งสัญญาณควบคุมไปยังสี่แยกอีก 3 สี่แยก ตามรูปดังนี้

รูปที่ 2.9 แสดงระบบการควบคุมสัญญาณไฟจราจร

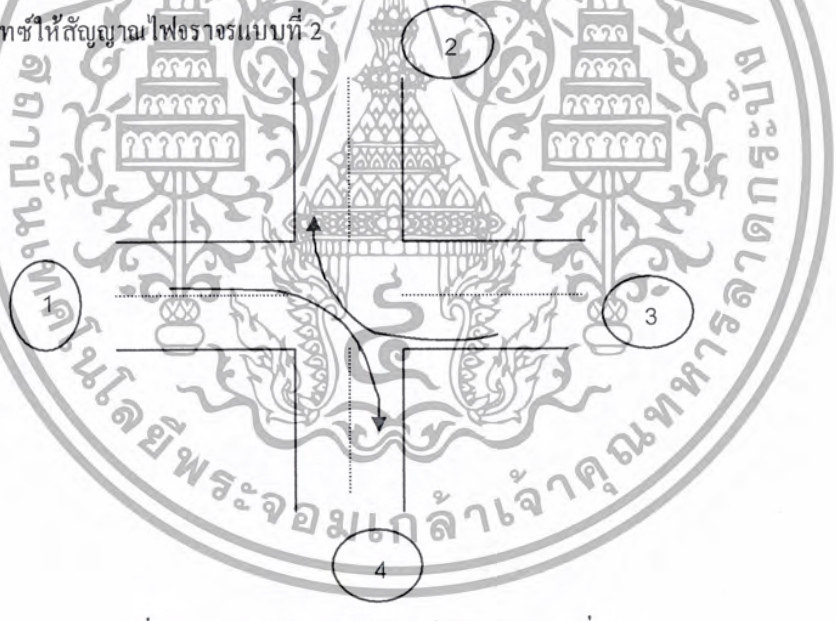
สำหรับการทำงานทั่วไปของระบบการควบคุมสัญญาณไฟจราจรที่ประดิษฐ์ขึ้นนั้นมีด้วยกัน 4 รูปแบบ คือ

1. สวิทช์ให้สัญญาณไฟจราจรแบบที่ 1



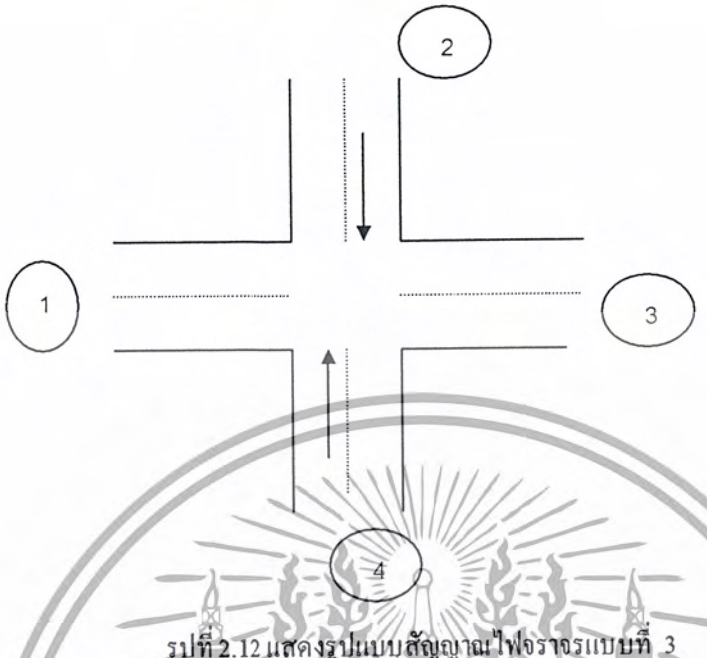
รูปที่ 2.10 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 1

2. สวิทช์ให้สัญญาณไฟจราจรแบบที่ 2



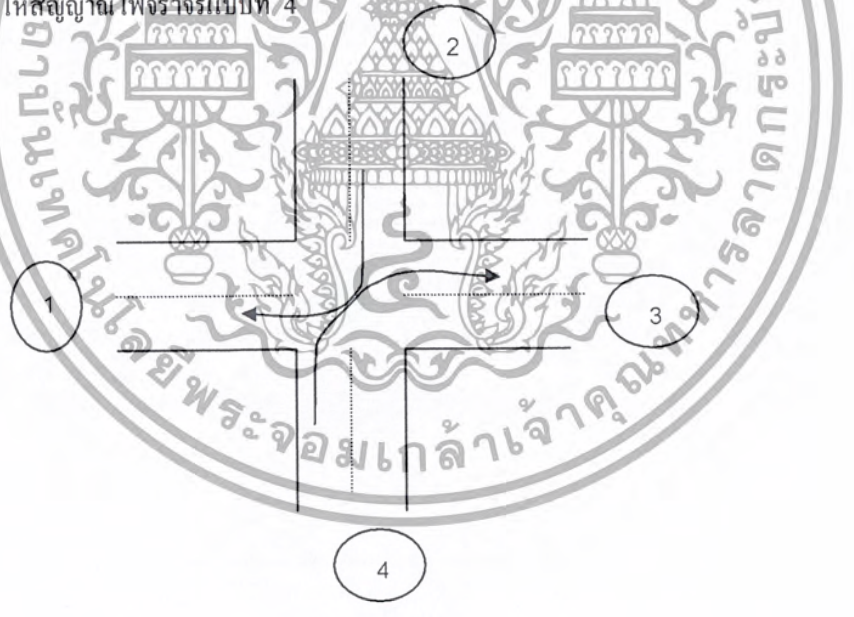
รูปที่ 2.11 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 2

3. สวิทซ์ให้สัญญาณไฟจราจรแบบที่ 3



รูปที่ 2.12 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 3

4. สวิทซ์ให้สัญญาณไฟจราจรแบบที่ 4



รูปที่ 2.13 แสดงรูปแบบสัญญาณไฟจราจรแบบที่ 4

2.3 โครงสร้างของ MCS - 51

ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล MCS – 51 ซึ่งมีหน่วยความจำภายในเป็นแบบแฟลช (Flash Memory)เหตุผลที่ใช้ ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีด้วยกันหลายประการดังนี้

1. หน่วยความจำโปรแกมภายในตัวไมโครคอนโทรลเลอร์ เป็นแบบแฟลช ทำให้สามารถลบและเขียนใหม่ได้นับพันครั้ง จึงสามารถใช้งานในรูปแบบของ ไมโครคอนโทรลเลอร์ ชิปเดี่ยวไม่ต้องใช้หน่วยความจำภายนอก ส่งผลให้สามารถใช้งานพอร์ทอินพุท เอาท์พุทของไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

2. ต้นทุนและเวลาในการพัฒนา ระบบไมโครคอนโทรลเลอร์ลดลงอย่างมาก เนื่องจากไม่ต้องใช้เครื่องมือพัฒนาจำพวกอีมูเลเตอร์และเครื่องโปรแกรมอีพรอมท์

3. ผู้ผลิตได้ทำการผลิตไมโครคอนโทรลเลอร์ ตระกูลนี้ออกมาหลายเบอร์ และ มีความสามารถแตกต่างกันไป ทำให้มีทางเลือกในการใช้งานสูง

4. ด้วยการใช้หน่วยความจำ ในตัวไมโครคอนโทรลเลอร์ทำ ให้สามารถป้องกันการคัดลอกข้อมูลของหน่วยความจำโปรแกรมได้เป็นอย่างดี

5. ในบางเบอร์ของไมโครคอนโทรลเลอร์ สามารถโปรแกรม ข้อมูลในหน่วยความจำ โปรแกรมได้โดยที่ไม่ต้องถอนตัวไมโครคอนโทรลเลอร์ออกมาทำการ โปรแกรมใหม่ หรือ เรียกว่าการโปรแกรมในวงจรหรือในระบบ (In-System Programing) โดยใช้ลักษณะการติดต่อแบบ SPI (Serial Peripheral Interface) ทำให้การพัฒนา หรือ การซ่อมบำรุงตลอดจนการปรับปรุง หรือ การอัพเกรดข้อมูลในหน่วยความจำโปรแกรมทำได้อย่างสะดวกภายใต้งบประมาณที่ไม่สูงมากนัก

6. ชุดคำสั่งสถาปัตยกรรมพื้นฐานเหมือนกับไมโครคอนโทรลเลอร์ MCS-51 ของผู้ผลิตอื่นไม่ว่าจะเป็นอินเทล ซีเมนส์หรือดัลรัส

คุณสมบัติของ MCS- 51 มีดังนี้

- ต้องการแหล่งจ่ายไฟ +5 โวลต์ ชุดเดียว
- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 5 กิโลไบต์ สำหรับเบอร์ 8051 และสำหรับเบอร์ 8031 และ 8032 นั้นไม่มีหน่วยความจำชุดนี้ ส่วน 8052 มีหน่วยความจำถึง 8 กิโลไบต์
- มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับ 8052 มีถึง 256 ไบต์
- หน่วยความจำสำหรับโปรแกรมและดาต้า (Program Memory And Data Memory) แยกจากกันอย่างละ 64 ไบต์
- คำสั่งที่ใช้เวลาน้อยที่สุดประมาณ 1 ไมโครวินาที เมื่อทำงานที่ความถี่ 12 เมกกะเฮิร์ต
- มีไทม์เมอร์/ เคาน์เตอร์ ขนาด 16 บิต 2 ชุด (สำหรับ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด
- รับอินเทอร์รัปท์ได้ 6 แหล่ง 5 เวคเตอร์
- มีพอร์ทรับส่งข้อมูลอนุกรม (UART) 2 พอร์ท ทั้งรับและส่งในเวลาเดียวกัน (Full Duplex) เลือกรูปแบบการส่งข้อมูลได้ 4 โหมด
- มีคำสั่งในการทำ AND, OR หรือ COMPLEMENT ได้ทั้งแบบ 8 บิตและ 1 บิต

การใช้งานเป็นพอร์ทอินพุท

เนื่องจากพอร์ททั้งหมดของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุทและเอาพุทท์ ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ทของไมโครคอนโทรลเลอร์MCS-51แบบแฟลช

ในการกำหนดให้เป็นพอร์ทอินพุทต้องเริ่มต้นด้วยการเขียนข้อมูล "1" มาที่แต่ละบิตของพอร์ททที่ต้องการใช้งานเป็นอินพุท เพื่อหยุดการทำงานของเฟตที่ใช้ในการขับสัญญาณเอาท์พุทของบิตนั้นๆทำให้ขาสัญญาณของพอร์ทเชื่อมต่อเข้ากับวงจรพูลอัพภายในโดยตรง ส่งผลให้ขาพอร์ทนั้นมีลอจิก "1" สามารถรับสัญญาณลอจิก "0" จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ท แล้วรอให้ซีพียูมาอ่านค่าเข้าไปเมื่อเป็นเช่นนี้อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ทอินพุทของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชควรกำหนดให้ทำงานในสภาวะลอจิก "0" จะดีและสะดวกที่สุด

การใช้งานเป็นพอร์ทเอาท์พุท

โดยปกติแล้วขาพอร์ทจะกำหนให้มีลักษณะเป็นเอาท์พุทอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปอย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการข้อมูล "0" ออกไปทางเอาท์พุทก็ให้เขียนข้อมูล "0" ไปยังวงจรแลตช์ ซึ่งก็ส่งต่อไปขับเฟต ทำให้เฟตทำงานที่ขาพอร์ทที่กำหนดให้ทำงาน ก็จะเกิดลอจิก "0" ขึ้น ในทางตรงกันข้ามหากต้องการส่งข้อมูล "1" ออกไปก็ให้เขียน "1" ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ขาของพอร์ตเชื่อมต่อกับวงจรพูลอัพภายในเกิดเป็นลอจิก "1" ที่ขาพอร์ทนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุทมากเพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูลโดยถ้าเป็นอินพุทจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาท์พุทจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างไร เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาท์พุท

เมื่อใช้งาน พอร์ทของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเป็นพอร์ทเอาท์พุทแต่ละขา (หรือแต่ละบิต) ของแต่ละพอร์ทมีความสามารถในการจ่ายกระแสหรือเรียกว่า กระแสซอร์ส (Source Current) ได้สูงสุด 10 mA และทุกขารวมกันในแต่ละพอร์ท (ทั้ง 8 บิต) สูงสุด 26 mA สำหรับพอร์ท 0 ได้ 15 mA สำหรับพอร์ท 1-3 ในกรณีที่ใช้งานทุกพอร์ท เอาท์พุทจะสามารถจ่ายกระแสได้รวมกันสูงสุด 71 mA ดังนั้นในการใช้งานเป็นพอร์ทเอาท์พุทเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาท์พุทเพื่อช่วยในการขับกระแสอีกทางหนึ่ง

2.4 ทฤษฎีและหลักการของ LCD

ทฤษฎี

รายละเอียดเกี่ยวกับโมดูล LCD

ในโมดูล LCD จะมีส่วนประกอบหลัก ๆ 3 ส่วนดังนี้

ตัวแสดงผล (Display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะ ชิปที่นิยม

ใช้คือ เบอร์ HD 44780 และ HD 61830 โดย HD 44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD 61830 ใช้ควบคุม LCD แบบกราฟฟิก

ตัวขับ (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD 44100 H และ MSM5259 เป็นต้น

โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ในที่นี้จะขอยกตัวอย่างโมดูล LCD แบบอักขระ เพราะสามารถทำความเข้าใจได้ง่าย ซึ่งใช้ในโมดูล LCD แบบอักขระ ประกอบด้วย

บัฟเฟอร์อินพุทเอาท์พุท เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register : IR) เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register : DR) เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอก เพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เป็นข้อมูลแสดงผล หรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

แรมเก็บข้อมูลแสดงผล (Display Data RAM : DDRAM) เป็นหน่วยความจำแรมที่ทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look Up Table) ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษร เพื่อนำไปแสดงที่ตัวแสดงผล

รอมเก็บตัวอักษร (Character Generator ROM : CGROM) เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงที่ตัวแสดงผลได้ มีขนาด 7200 บิต โดยจะถูกอ่านด้วยค่าของข้อมูลใน DDRAM

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGRAM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานการณ์ทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่า ตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

โมดูล LCD ขนาด 16 ตัวอักษร 2 บรรทัด (LCD 16x2)

สำหรับโมดูล LCD ที่ยกมาใช้ในการเรียนรู้ในการทดลอง เป็นขนาด 16 ตัวอักษร 2 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตราฐาน มีผู้ผลิตหลายราย และมีการระบุเบอร์ที่แตกต่างกันออกไปตามผู้ผลิต อาทิ LM020L ของฮิตาชิ , DMC-16117A ของคอปเทร็กซ์ (Optrex) เป็นต้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือ เบอร์เดียวกันนั่นคือ เบอร์ HD44750 ของฮิตาชิ

โมดูล LCD ขนาด 16x12มีขาต่อใช้งานทั้งสิ้น 14 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

V_{ss} (ขา 1) : ต่อกราวด์

V_{DD} (ขา 2) : ต่อไฟเลี้ยง + 5 โวลต์

V_O (ขา 3) : เป็นขาอินพุทรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) : เป็นขาอินพุทใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

R/W (ขา 5) : เป็นขาที่เลือกใช้การอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาอีนาเบิล LCD ให้ทำงาน

D0-D7 (ขา 7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต

RS ,R/W และ E จะใช้งานร่วมกัน

คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าต้องกำหนดให้ขา RS และ R/W เป็น “0” แล้วเขียนตามคำสั่งตามไป คำสั่งควบคุมโมดูล LCD ของชิปควบคุม HD44780 ที่สำคัญมี 10 คำสั่ง ดังนี้

1. คำสั่งเคลียร์ตัวแสดงผล (Clear Display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ Space เข้าไปใน DDRAM เป็น 0 เคอร์เซอร์ จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซต I/D (ซึ่งจะกล่าวถึงที่หลัง) ให้เป็น“1”

2. คำสั่ง Return Home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น “1” เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผลแต่ข้อมูลบนหน้าจอแสดงผลไม่เปลี่ยนแปลง นั่นคือ ข้อมูลคำสั่งของคำสั่งนี้จะเป็น O2H หรือ O3H ก็ได้

3. คำสั่งเลือกโหมดการป้อนข้อมูล (Entey Mode Set)

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	0	1	I/D	S

บิต S เป็นบิตที่ใช้ในการกำหนดลักษณะของการแสดงผล เมื่อมีการป้อนข้อมูล ถ้าหากบิต S เป็น “1” เมื่อเกิดข้อมูลใหม่บนหน้าจอแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางซ้าย แต่ถ้าหากบิตนี้เป็น “0” เมื่อเกิดข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางขวามือ

บิต I/D เป็นบิตที่ใช้ในการกำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว ทำให้แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น “1” แอดเดรสของ DDRAM จะเพิ่มขึ้นแต่ถ้าเป็น “0” แอดเดรสจะลดลง

ดังนั้น ข้อมูลคำสั่งที่เกิดขึ้นสำหับคำสั่งนี้ได้แก่ O4H – O7H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ O6H หมายถึง กำหนดให้เกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ และแอดเดรสของDDRAM จะเพิ่มขึ้น

4. คำสั่งควบคุมการแสดงผล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	1	D	C	B

บิต D ใช้ควบคุมการเปิดปิดจอแสดงผล ถ้าบิตนี้เป็น “1” จะเป็นการเปิดจอแสดงผล ถ้าเป็น “0” จะเป็นการปิดจอ

บิต C ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนหน้าจอแสดงผล ถ้าต้องการให้เคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดให้บิตนี้เป็น “1” ถ้ากำหนดให้เป็น “0” จะเป็นการปิดเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ถ้าบิตนี้เป็น “1” เคอร์เซอร์จะกระพริบดั้งนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ O8H – OFH (8 รูปแบบคำสั่ง) ที่ใช้บ่อยคือ OCH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์และ OFH เป็นการสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์และ สั่งให้เคอร์เซอร์กระพริบ

5. คำสั่งควบคุมเคอร์เซอร์และข้อมูลตัวอักษร

มีรายระเอียดของข้อมูลรูปแบบคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	1	S/C	R/L	*	*

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผลขึ้นกับการกำหนดบิต S/C และR/L ซึ่งสามารถสรุปได้ดังนี้

S/C	R/L	ลักษณะการเลื่อน	ข้อมูลคำสั่ง
0	0	เลื่อนเคอร์เซอร์ไปทางซ้าย	10H-13H
0	1	เลื่อนเคอร์เซอร์ไปทางขวา	14H-17H
1	0	เลื่อนตัวอักษรใหม่ไปทางซ้าย	18H-IBH
1	1	เลื่อนตัวอักษรใหม่ไปทางขวา	1C-1FH

6. คำสั่งกำหนดฟังก์ชันการทำงาน

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	1	DL	N	F	*	*

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ติดต่อที่ใช้ติดต่อส่งผ่านข้อมูล ถ้าบิตนี้เป็น “0” จะเป็นการติดต่อแบบ 4 บิต แต่ถ้าเป็น “1” จะเป็นแบบ 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น “0” จะแสดงผล 1 บรรทัด ถ้าเป็น “1” จะแสดง 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้มากกว่า 2 บรรทัด และต้องการให้แสดงผลมากกว่า 2 บรรทัด ก็กำหนดบิต N นี้ให้เป็น “1”

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น “0” จะเป็นการแสดงผลแบบ 5*7 จุด และถ้าเป็น “1” จะแสดงผลเป็นแบบ 5*10 06f

ข้อมูลคำสั่งที่ใช้บ่อย คือ 38H เป็นการกำหนดให้โมดูล LCD ทำงานในแบบ 8 บิต แสดงผล 2 บรรทัด และเลือกความละเอียดเป็น 5*7 จุด

จุดที่น่าสังเกต คือ โมดูล LCD แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีการแสดงผลเพียง 1 บรรทัด แต่จะต้องกำหนด N ให้เป็น “1” เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ00H และ 40H

7. คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น “0” บิต 6 เป็น “1” ส่วนอีก 6 บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ก่อนที่จะอ่านหรือเขียนข้อมูลให้ CGRAM โดยแอดเดรสของ CGRAM อยู่ระหว่าง 00H-3FH

8. คำสั่งเลือกแอดเดรสของ DDRAM

ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูลโดยบิต 7 ต้องเป็น “1” และข้อมูลอีก 7 บิตที่เหลือจะเป็น ค่าแอดเดรสของ DDRAM ซึ่งแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH-0FFH ทั้งนี้จำนวนแอดเดรสยังขึ้นกับการกำหนดสถานะที่บิต N ด้วย หากบิต N เป็น “0” แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H-0CFH และถ้าบิต N เป็น “1” แอดเดรสของ DDRAM จะมีช่วงคือ 8CH-87H และ 0C0H-0C7H

9. คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่ง ดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
BF	A	A	A	A	A	A	A

เป็นคำสั่งที่ใช้อ่านแฟลก BUSY (BF) โดยแฟลกนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลอยู่หรือไม่ ถ้าหากบิต BF เป็น “0” แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น “1” แสดงว่าขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง

เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา R/W เป็น “1” ด้วย แต่สัญญาณที่ RS ยังต้องเป็น “0” อยู่ เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง

นอกจากนี้ ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต 0- บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

2.5 การใช้งานของพอร์ทสื่อสารอนุกรม RS485

ผู้ใช้งานสามารถทำการเขียนโปรแกรมควบคุมการสื่อสารข้อมูลของ MCU กับอุปกรณ์อื่นๆได้ตามต้องการ โดยในส่วนของโปรแกรมนั้น ผู้ใช้สามารถกำหนดรูปแบบของการสื่อสารข้อมูลได้เองจากโปรแกรมที่เขียนขึ้น ไม่ว่าจะเป็นความเร็วในการสื่อสาร (Baud Rate)จำนวนบิตขอมูลในการรับส่ง (Data Bit) การกำหนดบิตตรวจสอบความถูกต้องข้อมูล (Parity) และคุณสมบัติอื่นๆซึ่งในรายละเอียดส่วนนี้จะไม่ขอกล่าวถึงขอให้ผู้ใช้ศึกษาจากคู่มือสถาปัตยกรรมทางฮาร์ดแวร์หรือ Data Sheet ของMCUเองซึ่งปกติแล้วขาสัญญาณสำหรับ รับ-ส่ง ข้อมูลของ MCU นั้นสามารถนำไปเชื่อมต่อกับขาสัญญาณรับ-ส่งของอุปกรณ์อื่นๆได้ โดยขาส่ง (TX) ของ MCU ต้องนำไปต่อกับขารับ (RX) ของอุปกรณ์ที่จะนำมาสื่อสารกัน ส่วนขารับข้อมูล (RX) ของ MCU ก็ต้องต่อกับขาส่งข้อมูล (TX) จากอุปกรณ์ที่จะนำมาสื่อสารกัน แต่เนื่องจากขาสัญญาณ RX และ TX ของ MCU นั้น จะสามารถเชื่อมต่อกับสัญญาณที่มีคุณสมบัติเป็นแบบระดับลอจิก TTL เท่านั้น ซึ่งถ้าใช้วิธีการเชื่อมต่อสัญญาณรับส่งของ MCU กับอุปกรณ์โดยตรงนั้นจะสามารถสื่อสารกันได้เพียงระยะทางใกล้ๆหรือภายในแผงวงจรเดียวกันเท่านั้นไม่สามารถสื่อสารกันด้วยระยะทางไกลๆ ได้ ดังนั้น จึงได้ออกแบบวงจร Line Driver สำหรับทำหน้าที่เป็น Buffer เพื่อเปลี่ยนแปลงระดับสัญญาณทางไฟฟ้าของขาสัญญาณ รับ-ส่งข้อมูลของ MCU ที่เป็นแบบ TTL ให้สามารถรับส่งข้อมูลกันได้ ในระยะทางที่ไกลมากขึ้นโดย จะสามารถเลือกกำหนดรูปแบบของวงจรLine Driver สำหรับการสื่อสารอนุกรม คือ

การสื่อสารอนุกรมแบบ RS485

ในการสื่อสารแบบ RS485 นี้จะมีคุณสมบัติของสัญญาณทางไฟฟ้าเหมือนกัน RS422 ทุกประการ เพียงแต่ว่าในการสื่อสารแบบ RS485 นี้จะใช้สายสัญญาณในการรับส่งข้อมูลกันเพียง 2 เส้นเท่านั้น แต่จะมีความพิเศษกว่าแบบ RS422 ตรงที่ทิศทางของสัญญาณจะสามารถปรับเปลี่ยนได้จากโปรแกรม กล่าวคือ สัญญาณทั้ง 2 เส้นนี้สามารถจะสลับหน้าที่เป็นด้านส่งและเป็นด้านรับได้ตามต้องการ โดยการควบคุมจาก MCU นั้น จะกำหนดให้สัญญาณ PTC3 ทำหน้าที่สำหรับควบคุมทิศทางของข้อมูลว่าจะให้เป็นรับหรือส่ง โดยถ้าควบคุมให้ PTC3 มีสภาวะเป็น "1" จะเป็นการกำหนดทิศทางให้เป็นฝ่ายส่งข้อมูล แต่ถ้าสภาวะของ PTC3 เป็น "0" จะเป็นการกำหนดทิศทางให้เป็นฝ่ายรับข้อมูล ซึ่งจากคุณสมบัติข้อนี้จะทำให้การสื่อสารแบบ RS485 สามารถทำการต่อขนานอุปกรณ์ร่วมกันในสายส่งเดียวกันได้จำนวนหลายๆจุด โดยถ้าใช้ไอซี Line Driver เบอร์ 75176 จะสามารถต่อขนานอุปกรณ์กันได้จำนวน 32 จุด แต่ถ้าเลือกใช้ไอซี Line Driver เบอร์ MAX3088 แล้วจะสามารถต่อขนานอุปกรณ์ ในสายคู่เดียวกัน ได้มากถึง 256 แต่มีข้อแม้ว่าเมื่อมีการต่ออุปกรณ์ขนานกัน ในสายสัญญาณคู่เดียวกันมากกว่า 2 จุดแล้ว จะต้องเขียนโปรแกรมควบคุมให้มีการส่งข้อมูลออกมา ในสายครั้งละ 1 จุดเท่านั้น เพราะถ้ามีการกำหนดทิศทางของข้อมูลให้เป็นส่งในเวลาเดียวกันมากกว่า 1 จุดแล้วจะทำให้เกิดการชนกันของข้อมูลและไม่สามารถสื่อสารกันได้อย่างถูกต้อง โดยเมื่อต้องการใช้วิธีการสื่อสารแบบ RS485 นี้ จะต้องทำการติดตั้งไอซีLine Driverเบอร์75176หรือ MAX3088 ในตำแหน่งของ "TXD/485" เพียงตัวเดียว พร้อมกับเลือกกำหนดเป็นแบบ RS485 ดังนี้

- ทำการเลือก Jumper สำหรับเลือก "422/485" ไว้ทางด้าน 485 (RS485)
- ทำการเลือก Jumper "F/H" ไว้ทางด้าน H (Half Duplex)
- ทำการ Short Jumper สำหรับต่อตัวต้านทาน Fail Safe Resister คือ "TL" ไว้
- ทำการ Short Jumper สำหรับต่อตัวต้านทาน Fail Safe Resister คือ "TH" ไว้
- สายสัญญาณที่ใช้จะต่อจาก TXB(TX-) และ TXA(TX+) เพียง 2 เส้นออกไปใช้งาน

ซึ่งในการสื่อสารข้อมูลแบบ RS485 นี้ จะต้องเขียนโปรแกรมขึ้นมารองรับการสื่อสารโดยเฉพาะ เนื่องจากทิศทางของข้อมูลสามารถจะกำหนดจากโปรแกรมได้โดยตรง ซึ่งการสื่อสารวิธีนี้จะมีข้อดีคือ ใช้สายสัญญาณในการรับส่งน้อยเส้น แต่จะเสียเวลาในการสื่อสารมากกว่าวิธีอื่นๆ เนื่องจากว่าการสื่อสารแบบนี้จะไม่สามารถทำการรับและส่งข้อมูลในเวลาเดียวกันได้ แต่จะต้องใช้วิธีการผลัดกันรับผลัดกันส่งแทน ซึ่งในความเป็นจริงแล้วในปัจจุบันนี้ราคาของสายสัญญาณแบบ 2 เส้นและ 4 เส้นแทบจะไม่มีความแตกต่างกันเลย ดังนั้นเพื่อลดความยุ่งยากในการเขียนโปรแกรมสำหรับควบคุมการรับส่งข้อมูลของ MCU ให้เลือกใช้วิธีการสื่อสารแบบ RS422 จะง่ายและสะดวกรวดเร็วกว่ากันมาก

2.6 ความรู้เบื้องต้นของ I^2C

I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลัก คือ ต้องการให้ไอซีหรือโมดูลสามารถติดต่อสั่งงานและควบคุมภายใต้สัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือ สายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป ส่วนการกำหนดแอดเดรส หรือ ตำแหน่งสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2C มีชื่อเรียกอีกอย่างเป็นทางการว่า สายข้อมูลอนุกรมหรือ SDA (Serial Data Line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรมหรือ SCL (Serial Clock Line)

คุณสมบัติโดยทั่วไปของบัส I^2C

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (Bi-Directional Line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (Open-Drain) หรือคอลเล็กเตอร์เปิด (Open-Collector)

อัตราการถ่ายทอดข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติและสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400 pF การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่า คือ 7 บิต หรือ 10 บิต

ข้อเด่นอีกประการหนึ่งของบัส I^2C คือ สามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถติดต่อสื่อสารกันได้ โดยอุปกรณ์บนบัส I^2C ตัวหนึ่งอาจใช้ไฟเลี้ยง +5V ในขณะที่อีกตัวหนึ่งใช้ไฟเลี้ยง +12 V การต่อร่วมกันบนบัส I^2C สามารถกระทำได้ในลักษณะเดียวกับกรณีที่ อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเท่ากัน กล่าวคือ ให้ต่อสาย SDA และ SCL ของอุปกรณ์แต่ละตัวเข้าด้วยกัน และต้องต่อตัวต้านทานพูลอัพ (R_p) เข้ากับแรงดัน + 5 V ไว้ด้วยเสมอ

หลักการของบัส I^2C

บัส I^2C ประกอบด้วยสายสัญญาณ 2 เส้น ดังที่ได้กล่าวมาแล้ว คือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส เรียกว่า โปรโตคอล (Protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ตัวใดเป็นตัวรับ หรือ ตัวส่ง ต่อไปนี้จะขออธิบายลักษณะ หน้าที่และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I^2C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่จะอธิบายการทำงานของบัส I^2C ต่อไป

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่า ตัวส่ง (Transmitter)

อุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่า ตัวรับ (Receiver) ในอุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับเพียงอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I^2C ที่ทำหน้าที่เป็นตัวส่งอย่างเดียว

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ (Master)

อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่า สเลฟ (Slave)

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

(1) การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่าง

(2) ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้นสัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

สภาวะที่เกิดขึ้นบนบัส I^2C

มีด้วยกัน 5 สถานะ คือ

(1) **บัสว่าง** (Bus Not Busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

(2) **เริ่มต้นการถ่ายทอดข้อมูล** (Start Data Transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสถานะที่เกิดขึ้นนี้ว่า สภาวะเริ่มต้น (Start)

(3) **หยุดการถ่ายทอดข้อมูล** (Stop Data Transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูงเรียกว่า สภาวะที่เกิดขึ้นว่า สภาวะหยุด(Stop)

(4) **ข้อมูลคงอยู่บนบัส** (Data Valid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือ ข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่าเป็น "0" หรือ "1" ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA

ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น

(5) **รับรู้ข้อมูล** (Acknowledge) เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิต เรียกว่า บิตรับรู้ (Acknowlegde Bit)มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจากตัวส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว

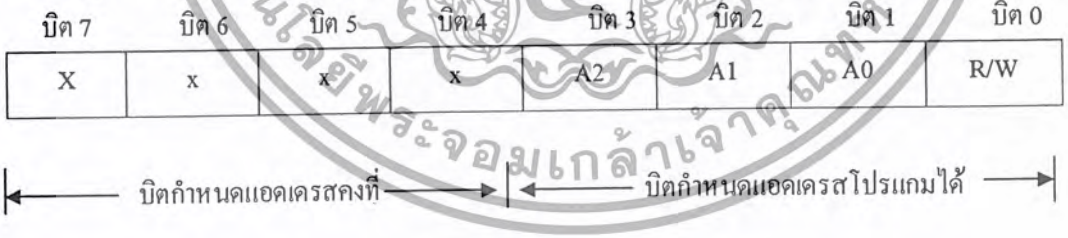
การทำงานบนบัส I^2C

ก่อนที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆ ที่ต่ออยู่บนบัส ต้องมีการอ้างถึงเสียก่อน โดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะใช้การอ้างถึงแบบ 7 บิตหรือ 10 บิต ในกรณีที่มีอุปกรณ์ต่อยู่บนบัสไม่มากนัก ใช้การอ้างถึงแบบ 7 บิตก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้วก็จะเริ่มต้นการถ่ายทอดข้อมูลกันต่อไป

ดังนั้นหัวใจสำคัญในอันดับแรกของการทำงานบนบัส I^2C คือ การอ้างถึงอุปกรณ์แต่ละตัว ซึ่งในที่นี้จะอธิบายรายละเอียดของการอ้างถึงทั้ง 2 รูปแบบ ดังนี้

1. การอ้างถึงแบบ 7 บิต (7- Bit Addessing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อหรือ ข้อมูลกำหนดแอดเดรส โดยมีรูปแบบแสดงในรูป



รูปแบบของข้อมูลกำหนดแอดเดรสที่ใช้ในการอ้างถึงแบบ 7 บิต

ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็น บิตกำหนดแอดเดรสคงที่ (Fixed Address Bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (Programmable Address Bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับ

อุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB เป็น "0" หมายถึง ต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น "1" จะเป็นการอ่านข้อมูลจากสเลฟ

ข้อมูลในไบต์ต่อมา คือ ข้อมูลควบคุม (Control Byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่าง ไอซีขยายพอร์ทมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุท บิตใดเป็นเอาท์พุท ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมา คือ ข้อมูลที่ทำการถ่ายทอดจริง (Data) หลังจากมีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

2. การอ้างถึงแบบ 10 บิต

ในการอ้างถึงแบบนี้ยังคงใช้รูปแบบข้อมูลอนุกรมที่เหมือนกับแบบ 7 บิต หากแต่จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในข้อมูลไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่า ต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อเช่นเดียวกับการอ้างถึงแบบ 7 บิต หลังจากถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

อุปกรณ์ที่ใช้เชื่อมต่อแบบบัส I^2C

ในปัจจุบัน บัส I^2C ได้รับความนิยมเพิ่มมากขึ้นเรื่อยๆ ด้วยข้อดีที่ชัดเจน คือ ใช้สายสัญญาณเพียง 2 เส้นเท่านั้น และการขยายระบบไมโครคอนโทรลเลอร์ ที่มีจำนวนอินพุท เอาท์พุทและหน่วยความจำจำกัดสามารถทำได้ง่ายขึ้นด้วยระบบบัส I^2C เมื่อเป็นเช่นนี้จึงมีอุปกรณ์เพอริเฟอรัลที่ใช้การเชื่อมต่อแบบบัส I^2C มากมายจากหลายผู้ผลิตออกมาให้ใช้งานกัน ดังตัวอย่างต่อไปนี้

ไอซีขยายพอร์ทอินพุทเอาท์พุท (I/O Expander) PCF8574 , PCF8582 , PCF8584

ไอซีหน่วยความจำอีพรอมท์อนุกรม (Serial EEPROM) : 24Cxx , PCF8570 , PCF72/73 , PCF8582

ไอซี ADC/DAC : PCF8591

ไอซีรีลไทม์คล็อก (Real-Time Clock : RTC) : PCF8583 , PCF8593 , PCF8598 , 41T56C

ไอซีขับ LCD โมดูล (LCDDriver) : PCF8466 , PCF8576 , PCF8577/78 , PCF8579 , SAA1064

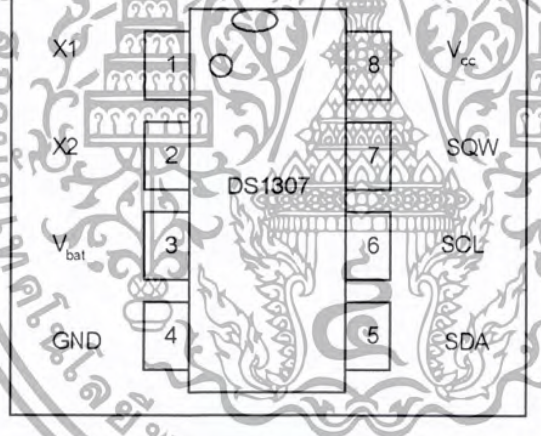
ไอซีกำเนิดสัญญาณ DTMF (DTMF Generator) : PCD3311/12

2.7 DS1307 ไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (RTC)

ผู้ผลิต คือ ดัลลัสเซมิคอนดักเตอร์ (Dallas Semiconductor) มีหน้าที่สร้างฐานเวลาจริงให้แก่ ระบบไมโครคอนโทรลเลอร์ โดย DS1307 จะให้ข้อมูลเกี่ยวกับเวลาทั้งหมด ไม่ว่าจะเป็นค่าของเวลาที่ละเอียดถึงหลักวินาที , นาที , ชั่วโมง , วันที่ (Date) , เดือน และปี โดยสามารถปรับวันเดือนปีให้ตรงตามปฏิทินได้อย่างถูกต้อง คุณสมบัติทางเทคนิคที่สำคัญมีดังนี้

- เป็นไอซีรีลไทม์คล็อกให้ข้อมูลตั้งแต่วินาทีจนถึงปี รวมถึงการปรับวันในปีอธิกสุรทินด้วย สามารถให้ข้อมูลเวลาได้อย่างเที่ยงตรงถึงปีคริสตศักราช 2100
- มีหน่วยความจำนอนโวลาไทล์แรม 56 ไบต์อยู่ภายใน สามารถใช้เป็นข้อมูลทั่วไปได้
- ใช้การเชื่อมต่อแบบระบบบัส I^2C
- มีวงจรตรวจจับไฟเลี้ยงต่ำหรือหายไปอย่างอัตโนมัติและสามารถรักษาข้อมูลเวลาไว้ได้แม้ไม่มีไฟเลี้ยงไอซี

รายละเอียดขาต่อใช้งานของ DS1307



รูปที่ 2.14รูปการจัดขาของไอซี DS1307

V_{cc} , GND (ขา 8, 4) ต่อกับไฟเลี้ยง +5V

V_{bat} (ขา 3) ใช้ต่อกับแบตเตอรี่ 3V เพื่อรักษาการทำงานของวงจรสร้างฐานเวลาของ DS1307 ให้คงอยู่ต่อไป แม้ว่าไม่มีไฟเลี้ยงจ่ายให้แก่ DS1307 ชนิดของแบตเตอรี่ที่เหมาะสม คือ แบตเตอรี่แบบลิเธียม ซึ่งมีความจุ 40 mAhr หรือมากกว่า จะสามารถรักษาข้อมูลได้นาน 10 ปีที่อุณภูมิ 25 องศาเซลเซียส

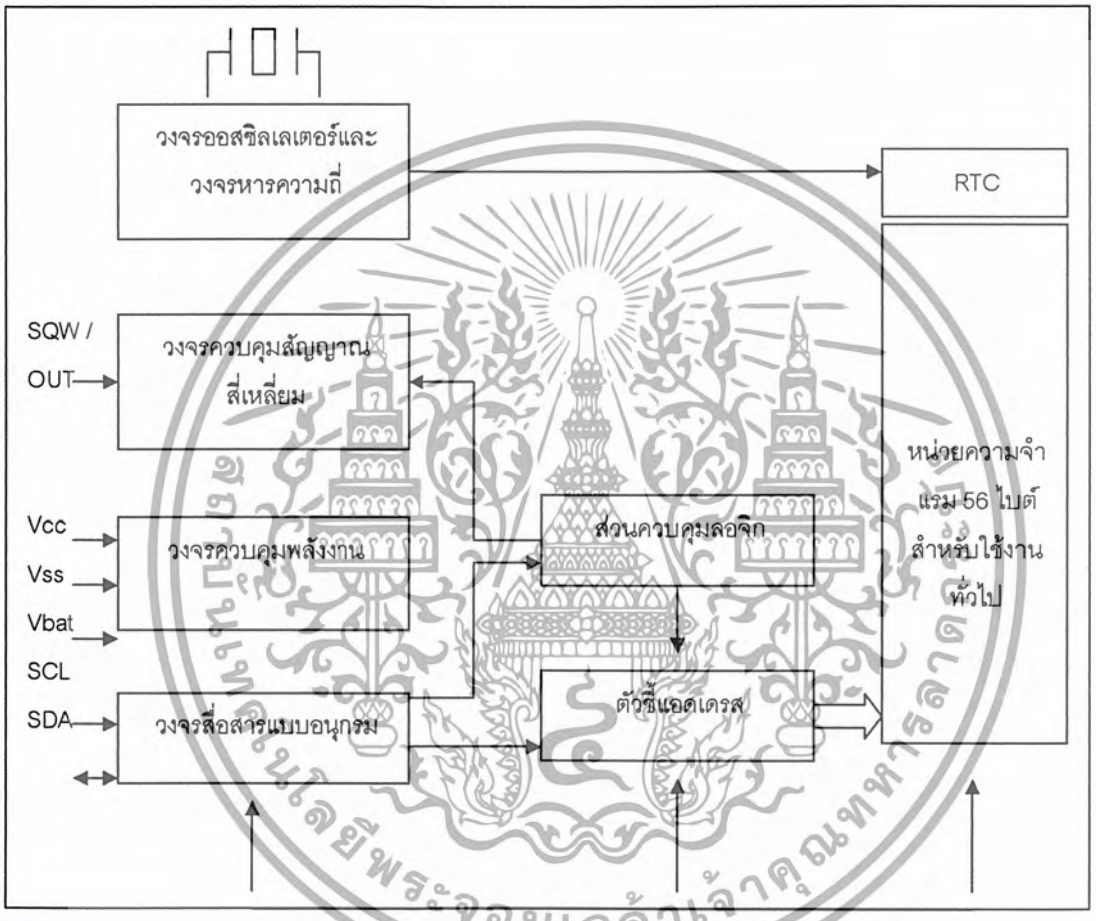
SDA , SCL (ขา 5และ 6) เป็นขาสำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์บนระบบบัส I^2C

SQW/OUT (ขา 7) ที่ขานี้จะมีสัญญาณรูปสี่เหลี่ยมส่งออกมา โดยสามารถเลือกความถี่ได้ 1Hz , 4.096 kHz , 8.192 kHz และ 32 kHz ในการใช้งานต้องต่อตัวต้านทาน 1k พูลอัพที่ขานี้ด้วย

X1 , X2 (ขา 1 และ 2) ใช้ต่อกับคริสตอลความถี่มาตราฐาน 32.768 kHz เพื่อใช้เป็นฐานเวลาในการสร้างค่าเวลาจริง ในการใช้งานต้องต่อคริสตอลเข้ากับขาทั้งสองนี้และที่แต่ละขาต้องต่อตัวเก็บประจุค่าต่ำๆ ประมาณ 15 pF คร่อมกับขากราวด์ด้วย

การทำงานของ DS1307

ไอซี DS1307 จัดการเชื่อมต่อในแบบบัส I^2C โดยจะทำงานเป็นอุปกรณ์สเลฟเสมอ ดังนั้นการติดต่อเพื่อใช้งานจึงต้องกำหนดรูปแบบตามที่กำหนดไว้ในการติดต่อแบบ I^2C



รูปที่ 2.15 แสดงส่วนประกอบหลักที่สำคัญและไดอะแกรมการทำงานของ DS1307

วงจรออสซิลเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริง ในขณะที่ DS1307 ทำงานที่ขา SQW / OUT จะมีสัญญาณพัลส์สี่เหลี่ยมส่งออกมาตลอดเวลาในกรณีที่มีการอีนาเบิลวงจรกำเนิดสัญญาณพัลส์ที่รีจิสเตอร์ควบคุม ค่าความถี่ของสัญญาณนี้สามารถเลือกได้ 4 ค่า คือ 1Hz , 4.096 kHz , 8.192 kHzและ 32 kHz พร้อมกันนั้นก็จะมีการเก็บค่าของเวลาไว้ในหน่วยความจำนอนโวลาไทล์แรม ซึ่งมีขนาดรวม 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์และเป็นหน่วยความจำสำหรับเป็นข้อมูลทั่วไปสำหรับผู้ใช้งานอีก 56 ไบต์

วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า 1.25 xV_{bat} ก็จะควบคุมให้ DS1307 หยุดการทำงานรีเซตค่าตัวนับแอดเดรสภายในทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นในการใช้งาน DS1307 ต้องระมัดระวังอย่าให้ไฟเลี้ยงมีค่าต่ำกว่า 1.25xV_{bat} หรือประมาณ 3.75 V ในกรณีที่ใช้ V_{bat} เท่ากับ 3V ถ้าหากไฟเลี้ยงมีค่าต่ำกว่า V_{bat} ไอซี DS1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที จะไม่มีการส่งสัญญาณพัลส์ออกมาที่ขา SQW / OUT แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็จะสามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานได้ต่อไป

วงจรสื่อสารอนุกรมภายใน DS1307 ได้รับการกำหนดให้ทำงานตามรูปแบบของบัส I^2C เป็นช่องทางการสื่อสารระหว่าง DS1307 กับอุปกรณ์มาสเตอร์ ผู้ใช้งานสามารถเข้าถึงหน่วยความจำที่ใช้เก็บค่าเวลาและหน่วยความจำใช้งานทั่วไปได้โดยการเขียนข้อมูลตามรูปแบบที่กำหนดในระบบบัส I^2C

โหมดการทำงานของ DS1307

มีด้วยกัน 2 โหมด คือ โหมดเขียนข้อมูลและโหมดอ่านข้อมูล ในการใช้งาน DS1307 ตามปกติจะใช้งานเฉพาะโหมดอ่านข้อมูลเท่านั้น เนื่องจากไมโครคอนโทรลเลอร์จะติดต่อกับ DS1307 เพื่ออ่านข้อมูลของเวลาไปใช้งาน โหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อต้องการตั้งค่าเวลาใหม่และต้องการเขียนข้อมูลลงในหน่วยความจำใช้งานทั่วไป อย่างไรก็ตามเมื่อเริ่มต้นติดต่อกับ DS1307 จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเขียนข้อมูลก่อน เพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูล จากนั้นจึงเปลี่ยนโหมดการทำงานมาเป็นโหมดการอ่านข้อมูลต่อไป

-โหมดการเขียนข้อมูล เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ ทำการกำหนดสภาวะเริ่มต้น (Start:S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียน นั่นคือ ค่า 0 จากนั้นจะรอการตอบรับจาก DS1307 ขั้นตอนต่อมา คือ ส่งข้อมูลเพื่อเลือกแอดเดรสที่ต้องการเขียน จากนั้นรอการตอบรับจาก DS1307 เมื่อมีการตอบรับมาเรียบร้อย ก็เริ่มทยอยเขียนข้อมูลลงไปครั้งละแอดเดรส หลังจากเขียนข้อมูลในแต่ละแอดเดรสจะต้องหยุดรอการตอบรับจาก DS1307 ทุกครั้ง จึงจะสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนเรียบร้อยแล้วให้ส่งสภาวะหยุด (Stop : P) เป็นอันสิ้นสุดกระบวนการเขียนข้อมูล

- โหมดการอ่านข้อมูล เริ่มต้นการทำงานเหมือนกับโหมดการเขียนข้อมูล คือไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้นแล้วส่งข้อมูลกำหนดแอดเดรสตามด้วยข้อมูลเลือกการอ่านซึ่งเท่ากับ 1 จากนั้นรอการตอบรับจาก DS1307 เมื่อตอบรับเรียบร้อย DS1307 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์ครางละ 1 แอดเดรสหรือ 1 ไบต์ โดยแอดเดรสที่เลือกอ่านข้อมูลจะต้องมีการกำหนดมาก่อนล่วงหน้าด้วยโหมดการเขียนข้อมูล วิธีการง่ายๆ คือ เข้าสู่โหมดการเขียนข้อมูลก่อน เมื่อถึงจังหวะที่ต้องเขียนข้อมูล ให้ทำการสร้างสภาวะเริ่มต้นและส่งข้อมูลกำหนดแอดเดรสใหม่อีกครั้ง ตามด้วยเลือกโหมดการอ่าน ข้อมูล ข้อมูลที่ออกมาจาก DS1307 ก็จะเป็นข้อมูลจากแอดเดรสที่กำหนดไว้ก่อนหน้านี้

บทที่ 3

ขั้นตอนการทำงานของเครื่องควบคุม และ วงจรการทำงานของเครื่องควบคุมสัญญาณไฟจราจร

3.1 ลักษณะการทำงานของชุดควบคุมสัญญาณไฟจราจร

ในการออกแบบเครื่องควบคุมสัญญาณไฟจราจร เราทำการออกแบบเครื่องควบคุมสำหรับใช้ควบคุมไฟจราจร ณ. สี่แยก ซึ่งมีถนนสายหลักเป็น ถนนแปดเลน ตัดกับสายรอง ซึ่งเป็นถนนขนาดสี่เลน ซึ่งติดต่อกันเป็นระยะต่างๆกันและติดต่อกันเป็นจำนวนสี่จุดด้วยกัน ซึ่งในแต่ละจุดจะประกอบด้วยชุดของไฟตรง และไฟเลี้ยวจำนวนสี่ชุดด้วยกัน

สำหรับการทำงานของชุดควบคุมนั้น จะแบ่งการทำงานออกเป็น 3 โหมด คือ

1. การทำงานควบคุมสัญญาณไฟซึ่งอิสระต่อกันจุดต่อจุด
2. การทำงานควบคุมสัญญาณไฟจราจรแบบเชื่อมต่อกัน
3. สัญญาณไฟกระพริบ

การทำงานของชุดควบคุม มีดังนี้

1.การทำงานควบคุมสัญญาณไฟโหมดอิสระต่อกันจุดต่อจุด กล่าวคือในแต่ละจุดจะมีลักษณะการทำงานแบบ Auto แต่ไม่มีการเชื่อมต่อกันกับสี่แยกอื่นๆ ซึ่งจะมีหลักการทำงานในแต่ละจุดคือ ในการทำงานของไฟจราจรแต่ละขั้นจะมีการทำงานอยู่สาม Stage ด้วยกัน

Stage ที่1 โดยไฟจราจรจะทำงานโดยปรากฏเป็นไฟเขียว ซึ่งจะทำให้ไฟจราจรในจุดอื่นๆ เป็นไฟแดง เพื่อให้รถวิ่งผ่านได้

Stage ที่ 2 ไฟจราจรจะทำงานโดยปรากฏเป็นไฟเหลือง ซึ่งไฟจราจรจุดอื่นๆจะเป็นสีแดง เพื่อเป็นสัญญาณระวังให้เตรียมตัวที่จะหยุด

Stage ที่ 3 ไฟจราจรทุกจุดทำงานเป็นสีแดง เพื่อเป็นสัญญาณให้รถหยุด และเป็นการรอจังหวะให้รถได้ระบายออกหมดเพื่อไม่ให้เกิดปัญหารถติดค้างเนื่องจากไฟจราจรต่อเนื่องกันเกินไป

เมื่อชุดสัญญาณไฟจราจรทำงานครบทั้ง 3 Stage ก็จะเปลี่ยนการทำงานไปเรื่อย คือ เมื่อหลังจากปล่อยไฟตรงแล้วก็จะเป็นไฟเลี้ยวขวา และเปลี่ยนเป็นไฟตรงของเส้นทางสายที่สองและเปลี่ยนเป็นเลี้ยวขวาของเส้นทางสายที่ สอง ตามลำดับ ซึ่งการทำงานของถนนสายที่หนึ่ง จะเหมือนกับ ถนนสายที่ สามและการทำงานของไฟจราจรสายที่สอง ก็จะมีการทำงานเหมือนกับถนนสายที่สี่ และจะทำตามลำดับขั้นตอนอย่างนี้ไปเรื่อยๆ โดยเวลาของสัญญาณไฟแต่ละชุดจะพิจารณาตามจำนวนความหนาแน่นของปริมาณรถยนต์ในแต่ละช่วงเวลาโดยแบ่งออกเป็น 6 ช่วงเวลาซึ่งแปรผันตามจำนวนรถที่วิ่งในช่วงเวลานั้นๆ คือ

	ไฟเขียว 1	ไฟเขียว	ไฟเหลือง
05.01 – 07.00	30	30	4
07.01 – 09.00	45	30	4
09.01 – 16.00	30	30	4
16.01 - 20.00	45	30	4
20.01 – 24.00	30	30	4
24.01 – 05.00	ไฟเหลืองกระพริบ 2 วินาที		

2. การทำงานควบคุมสัญญาณไฟจราจรแบบต่อเนื่อง หมายถึงการควบคุมไฟจราจรโดยสามารถส่งสัญญาณไปยังจุดควบคุมสัญญาณไฟจราจรถัดไป เพื่อไปเป็นตัวแปรที่จะกำหนดการทำงานของไฟจราจรจุดนั้นๆ ต่อไปได้ โดยการทำงานของไฟจราจรแต่ละชุดของแต่ละแยกทำงานในทำนองเดียวกันกับการทำงานแบบ Auto ต่างกันก็แต่ จะมีการกำหนดสี่แยกใดสี่แยกหนึ่งที่เป็นต้นทางให้เป็น Master ส่วนอีกสามสี่แยกก็จะเป็น Slave1, Slave2 และ Slave3 ตามลำดับโดยที่ Slave1 จะได้รับสัญญาณจาก Master เพื่อนำเวลาที่ได้มาคำนวณรวมกับเวลาหน่วงของตัวเองและคำนวณเวลาหน่วงตามต้องการ โดยมีหลักการดังนี้

จาก Hardware ซึ่งจะกำหนดให้มี Board Master เป็นตัวควบคุมหลักและ Board Slave 1 Slave 2 Slave 3 และ Slave 4 ซึ่งควบคุมไฟจราจรแยกที่ 1 2 3 และ 4 ตามลำดับ โดยที่ Board Master จะประกอบด้วยชุดแสดงผล LCD ชุดรับข้อมูล (Keypad) ชุดสร้างสัญญาณนาฬิกา (Real Time Clock) และชุดขยายสัญญาณเชื่อมโยง RS-485 ส่วนชุด Board Slave ซึ่งประกอบด้วยชุดแสดงผลไฟจารจร ชุดสร้างสัญญาณนาฬิกา Real Time Clock และชุดขยายสัญญาณเชื่อมต่อ RS-485 โดยเริ่มจาก Board Master รับค่าจาก Keypad และแสดงผลทางLCD ซึ่งเป็นค่าที่เลือกเพื่อเปลี่ยนโหมดทำงานจาก 01 ไป 02 หรือ 03 และส่งสัญญาณไปยัง Board Slave 1 เมื่อ Board Slave 1 รับค่าก็จะเปลี่ยนไปยังโหมดเชื่อมโยง ซึ่งจะเพิ่ม Delay ของไฟเขียว ไปอีก 10 วินาที และเมื่อถึงไฟเขียวรอบต่อไปก็จะส่งสัญญาณไปยัง Board Slave 2 ซึ่งเป็นตัวควบคุมสัญญาณไฟจารจรชุดที่ 2 ให้เข้าสู่โหมดเชื่อมโยงเช่นกัน

3.สัญญาณไฟกระพริบ ณ สี่แยกใดๆ เมื่อเกิดกรณีอื่นๆทั้งหมดที่ไม่สามารถใช้งานแบบ Auto ได้ อาทิเช่น เกิดอุบัติเหตุ เป็นต้น

การออกแบบในส่วนของหน่วยควบคุมและประมวลผลของชุดควบคุมสัญญาณไฟจราจรสำหรับในส่วนของอุปกรณ์ควบคุมสัญญาณไฟจราจรนั้น สามารถทำงานได้ภายในตัวของมันเองอย่างอิสระ ดังนั้นในส่วนของหน่วยควบคุมและประมวลผลที่ทำการออกแบบนั้น ตามที่ได้กล่าวมาแล้วจะประกอบด้วยส่วนต่างๆ ซึ่งจะทำงานในลักษณะเกี่ยวข้องกันหมด โดยแบ่งเป็นส่วนต่างๆดังต่อไปนี้ได้แก่

1.ส่วนที่ทำการควบคุมสัญญาณไฟจราจร ซึ่งในแต่ละแยกประกอบด้วยส่วนของ CPU 8051 โดยมีรายละเอียดดังนี้

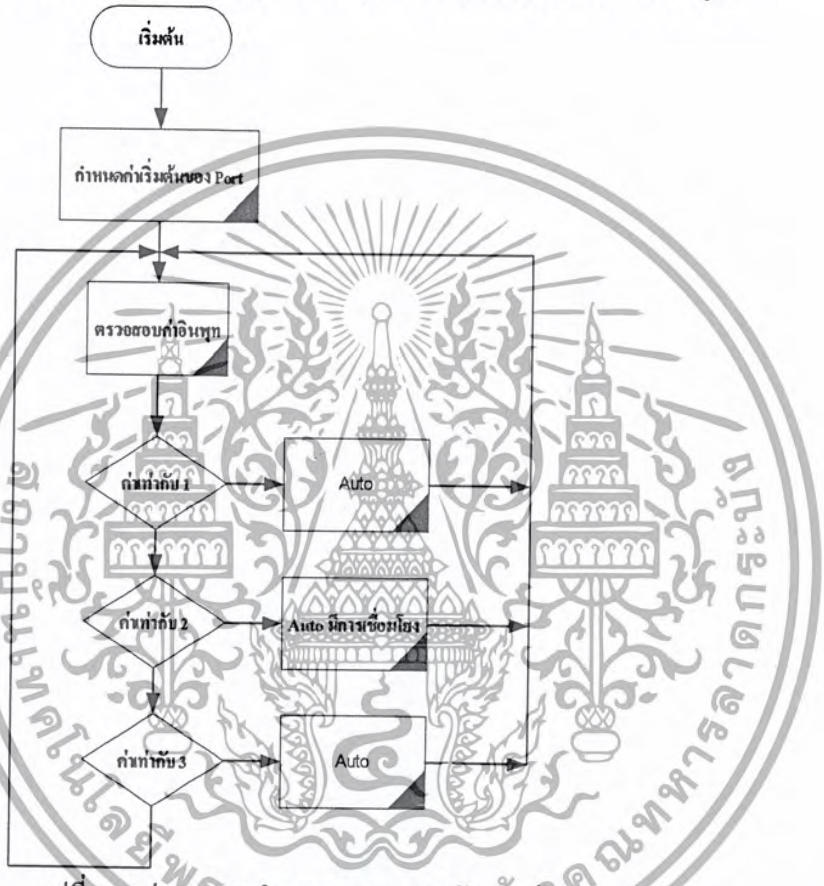
Board Master กำหนดให้พอร์ท 0 และ พอร์ท 3.6 กับ 3.7 เป็น Output แสดงผล LCD พอร์ท 2 เป็นอินพุท รับค่าจากอุปกรณ์รับค่า Keypad ติดต่อกับ Realtime Clock โดยใช้ Port 1.0 และ 1.1ส่วน ชุด Boarb Slave ใช้ พอร์ท 0 และ พอร์ท 1 เป็น Output ขับ LED ซึ่งเป็นสัญญาณไฟจราจร ซึ่งเป็นส่วนควบคุมสัญญาณไฟจราจร โดยจะเป็นชุดที่ทำหน้าที่ควบคุมการปิด และ เปิดของสัญญาณไฟจราจรแต่ละด้านของสี่แยกว่าควรจะมีไฟใดขึ้นที่แยกด้านใด ตามที่ผู้โปรแกรมต้องการ

2.ส่วนของนาฬิกา และ LCD โดยที่สัญญาณที่ได้จากส่วนของนาฬิกานั้นเราจะนำมาเป็นตัวอ้างอิงในการเปิดปิดสัญญาณจราจร เช่นต้องการให้เปิดไฟเขียวในแต่ละแยกนานเท่าไร ไฟเหลืองนานเท่าใดในเวลาใด หรือไม่กีใช้ควบคุมลักษณะการทำงานของไฟจราจรตามโปรแกรมอัตโนมัติที่ตั้งไว้ตามวันและเวลาต่างๆ ซึ่งอาจจะต่างกันไปตามความเหมาะสมหรือสถิติที่เก็บได้ เช่น ในวันหนึ่งๆน่าจะมีความหนาแน่นของการจราจรที่ต่างกันอยู่ ระหว่างชั่วโมงเร่งด่วนและเวลาปกติ ซึ่ง สามารถโปรแกรมให้มีเวลาหน่วงของไฟจราจรของไฟเขียว ไฟเหลืองและไฟแดงต่างกันตามเวลาดังกล่าวได้เพื่อจุดประสงค์ช่วยระบายรถได้มากที่สุด

ในชั่วโมงการจราจรหนาแน่นที่สุด

3.ส่วนที่เป็นตัวเชื่อมสัญญาณไฟจราจรจากชุดหนึ่งไปยังอีกชุดหนึ่ง ประกอบด้วย ไอซี เบอร์ SN75176GBP ซึ่งเป็นตัวขยายสัญญาณหรือเรียกได้อีกอย่างคือ RS485 โดยอาจทำหน้าที่เป็นตัวส่งสัญญาณไปยังตัวควบคุมชุดต่างๆ หรือทำหน้าที่เป็นตัวรับสัญญาณที่ส่งมาจากตัวควบคุมหลักเพื่อทำงานตามที่ตัวควบคุมหลักส่งสัญญาณมาเพื่อให้ปล่อยรถต่อเนื่องกันไปเรื่อยๆ

จากส่วนประกอบต่างๆของชุดควบคุมที่กล่าวมาแล้วนั้น เราสามารถนำมาสร้าง Block Diagram ซึ่งจะสัมพันธ์ระหว่างส่วนประกอบของตัวควบคุมในหนึ่งชุดได้ ซึ่งจะได้ Block Diagram ดังรูป



รูปที่ 3.1 รูปแสดงการทำงานภาพรวมของสัญญาณไฟจราจร

การออกแบบในส่วนคีย์รับข้อมูลนั้น เมื่อทำการกดคีย์แล้วคีย์จะส่งสัญญาณอินเทอร์รัปท์เข้าสู่ส่วนควบคุมเพื่อบอกให้ตัวควบคุมทราบว่ามีการป้อนข้อมูลข้ามาแล้ว ทางส่วนควบคุมก็จะหยุดการทำงานในส่วนที่ทำอยู่มาตรวจสอบคีย์รับข้อมูล ว่าทางผู้ควบคุมกดคีย์อะไร แล้วก็จะไปประมวลผลการทำงานตามโปรแกรมของคีย์ที่ผู้ควบคุมทำตามกด

การออกแบบโปรแกรมที่ใช้ควบคุมอุปกรณ์ควบคุมสัญญาณไฟจราจรสำหรับในส่วนของโปรแกรมที่ใช้ควบคุมอุปกรณ์ควบคุมสัญญาณไฟจราจรนั้น สามารถแบ่งออกได้เป็นหลายส่วนด้วยกันโดยทางผู้ออกแบบเลือกใช้ภาษา C ซึ่งขั้นตอนการออกแบบวงจรนั้น เริ่มต้นจากการเขียนแผนภาพแสดงลำดับการทำงานของโปรแกรม (Flow Chart) แล้วค่อยนำแผนภาพนั้นมาเขียนเป็นโปรแกรมต่อไป

บทที่4

การทดลองและผลการทดลอง

สำหรับในอุปกรณ์เครื่องควบคุมสัญญาณไฟจราจรนั้น ได้แบ่งการทดลองออกเป็นส่วนต่างๆ ทีละส่วน โดยในการทดลองทีละส่วนนี้มีวัตถุประสงค์เพื่อที่จะได้ง่ายในการตรวจสอบและสังเกตผลการทดลองต่างๆ ว่าในการทำงานของส่วนต่างๆ จะสามารถทำงานได้ตามที่เราต้องการหรือไม่ ซึ่งได้กำหนดหัวข้อในการทำการทดลองชุดควบคุมสัญญาณไฟจราจรเป็นหัวข้อต่างๆ ได้ดังนี้

1. การทดลอง ทำการทดสอบการควบคุมการทำงานของส่วนประมวลผล และ ทดสอบการ ทำงานของพอร์ท
2. การทดสอบการทำงานของบอร์ดแยกสัญญาณโดยใช้โปรแกรมไฟวิ่ง กับ บอร์ดแยกสัญญาณ
3. การทดสอบการทำงานของโปรแกรมสัญญาณไฟจราจรอัตโนมัติแบบอิสระต่อกัน จำนวน 1 แยก
4. การทดสอบการทำงานของอุปกรณ์แสดงผล LCD และ อุปกรณ์รับข้อมูล(Keypad)
5. การทดลองการทำงานของอุปกรณ์ให้สัญญาณนาฬิกา
6. การทดสอบการทำงาน ของโปรแกรมสัญญาณไฟจราจรอัตโนมัติแบบเชื่อมต่อโดยใช้บอร์ดแยกสัญญาณ จำนวน 2 แยก
7. การทดลองของชุดควบคุมสัญญาณไฟจราจรจำนวน 4 สี่แยก

4.1 การทดลอง ทำการทดสอบการควบคุมการทำงานของส่วนประมวลผล และ ทดสอบการทำงานของพอร์ท

ในการทดลองนี้ มีวัตถุประสงค์เพื่อทดสอบการทำงานของส่วนประมวลผลให้สามารถควบคุมการทำงานเพื่อให้ได้ผลตามโปรแกรมที่สั่งงานเข้าไปได้อย่างถูกต้อง เพื่อที่จะได้แน่ใจว่าส่วนประมวลผลนี้จะสามารถนำไปใช้ควบคุมไฟจราจรได้ตามที่ต้องการโดยไม่ผิดพลาด และทดสอบการทำงานของพอร์ทของ 74HC540N ว่าสามารถแสดงผลออกมาตามที่เราต้องการได้ เพื่อที่จะนำไปใช้ในการแสดงสัญญาณไฟจราจรทางอุปกรณ์แสดงผลจำลองสัญญาณไฟจราจร หรือเชื่อมต่อกับอุปกรณ์แสดงผลแบบ LCD ให้แสดงผลได้ตามต้องการโดยส่วนของวงจรที่จะใช้ในการทดลองในส่วนนี้ จะใช้เฉพาะส่วนต่างๆดังต่อไปนี้ ได้แก่ ส่วนของ 89C51 ,พอร์ทที่เชื่อมต่อกับ MAX 232CPE ซึ่งเป็นส่วนของอินพุท กับพอร์ทต่าง ๆ ที่เชื่อมต่อกับ 89C51,ส่วนของ 74HC540N ,โปรแกรมทดสอบซึ่งจะส่งผ่านเข้าไปเพื่อให้ IC 89C51 ทำงานตามต้องการ

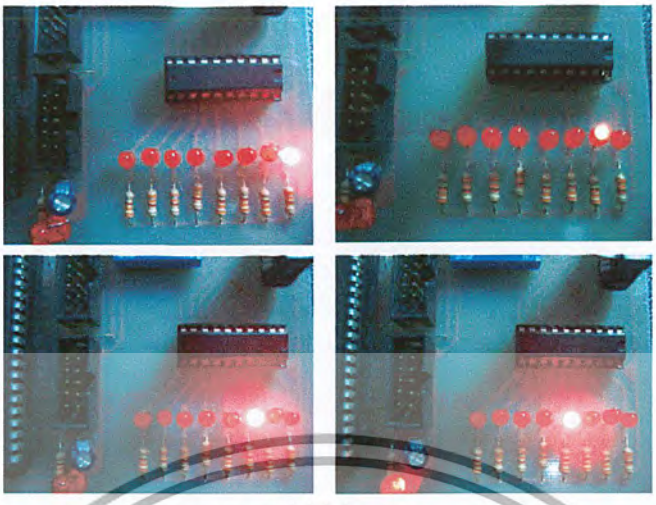
วิธีทำการทดลอง สามารถทำได้โดยการใช้โปรแกรมทดสอบเพื่อสั่งให้ IC 89C51 ทำงาน โดยในโปรแกรมจะสั่งงานให้ส่งข้อมูลออกพอร์ท 1 ของIC89C51 โดยส่งข้อมูลออกเป็นไฟติดวิ่งสลับกันไปเลื่อย ๆ

สำหรับโปรแกรมที่ใช้ทดลองนี้มีดังนี้

```
/* FLASH.C - LED Flasher for the Keil MCB251 Evaluation Board with
80C51 device*/

#include <REG51.H>

void wait (void)  {                    /* wait function */
  ;                                    /* only to delay for LED flashes
*/
}

void main (void)  {
  unsigned int i,x;                      /* Delay var */
  unsigned char j;                     /* LED var */
  //unsigned char x;
  x=600000;
while (1) {
      P0=0x01;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                          /* call wait function */
    }
      P0=0x02;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                          /* call wait function */
    }
      P0=0x04;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                          /* call wait function */
    }
      P0=0x08;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                          /* call wait function */
    }
      P0=0x10;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                          /* call wait function */
    }
      P0=0x20;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                          /* call wait function */
    }
      P0=0x40;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                          /* call wait function */
    }
      P0=0x80;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                          /* call wait function */
    }
}
}
```

ผลการทดลอง

หลังจากที่ได้ทดลองตามขั้นตอนข้างต้นแล้ว โดยทำการสังเกตและเก็บผลด้วยการถ่ายรูป ตำแหน่งพอร์ทที่ 1 แล้วปรากฏว่าข้อมูลที่ส่งออกมาทางพอร์ท 1 แสดงเป็นไฟวิ่งเริ่มจากดวงแรก ไปดวงที่สอง และดวงที่สามตามลำดับตามโปรแกรม แสดงว่า 89C51 สามารถทำงานได้ตามวัตถุประสงค์ของการทดลองจึงทำการทดลองในหัวข้อต่อไป โดยรูปผลการทดลองแสดงได้ดังนี้



รูปที่ 4.1 การทดสอบการควบคุมการทำงานของส่วนประมวลผลและ ทดสอบการทำงานของพอร์ท

4.2 ทำการทดสอบการทำงานของบอร์ดแยกสัญญาณโดยใช้โปรแกรมไฟวิ่ง กับ บอร์ดแยกสัญญาณ

ในการทดลองนี้ มีวัตถุประสงค์ เพื่อทดสอบการทำงานของ บอร์ดแยกสัญญาณว่าสามารถที่จะใช้งานได้หรือไม่ โดยการทดลองนี้ จะใช้ส่วนประกอบต่าง ๆ ในวงจร ได้แก่ ส่วนของ 74HC540N , ส่วนของอุปกรณ์แสดงผลจำลองสัญญาณไฟจราจร,ส่วนของบอร์ดแยกสัญญาณ และ ส่วนของโปรแกรมทดสอบ

วิธีทำการทดลอง ทำโดยการทดสอบการทำงานของพอร์ท 89C51 กับชุดแสดงผลสัญญาณไฟจราจรจำลอง โดยจะใช้โปรแกรมซึ่งจะให้อุปกรณ์แสดงผลสัญญาณไฟจำลอง ทำการแสดงสัญญาณไฟในแต่ละดวงติดสลับกันไปตามลำดับ เขียว เหลือง แดง เพื่อที่จะทดสอบว่าอุปกรณ์ที่ใช้ทดลองนี้สามารถใช้งานได้ครบถ้วน เพื่อที่จะนำไปใช้ในการทดสอบโปรแกรมจำลองรูปแบบไฟจราจรในส่วนต่อไป โดยจะมีโปรแกรมที่ใช้ทดลองในส่วนแรกดังต่อไปนี้

สำหรับโปรแกรมที่ใช้ในการทดลองมีดังนี้

```
/* FLASH.C - LED Flasher for the Keil MCB251 Evaluation Board with
80C51 device*/

#include <REG51.H>

void wait (void)  {                    /* wait function */
  ;                                    /* only to delay for LED flashes
*/
}

void main (void)  {
  unsigned int i,x;                      /* Delay var */
  unsigned char j;                     /* LED var */
```

```
  //unsigned char x;
  x=600000;
while (1) {
      P0=0x01;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                    /* call wait function */
    }
      P0=0x02;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                    /* call wait function */
    }
      P0=0x04;
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                    /* call wait function */
    }

}
}
```

ผลการทดลอง

หลังจากที่ได้ทดลองตามขั้นตอนข้างต้นแล้ว โดยทำการสังเกตและเก็บผลด้วยการถ่ายรูป ตำแหน่งพอร์ทที่ 1 แล้วปรากฏว่าข้อมูลที่ส่งออกมาทางพอร์ท 1 แสดงเป็นไฟวิ่งเริ่มจากไฟเขียว ไฟเหลือง และไฟแดงตามลำดับตามโปรแกรม แสดงว่า 89C51บอร์ดแยกสัญญาณ สามารถทำงานได้ตามวัตถุประสงค์ของการทดลองจึงทำการทดลองในหัวข้อต่อไป โดยรูปผลการทดลองแสดงได้ดังนี้

รูปที่ 4.2 การทดสอบการทำงานของบอร์ดแยกสัญญาณโดยใช้โปรแกรมไฟวิ่ง กับ บอร์ดแยกสัญญาณ

4.3 การทดสอบการทำงานของโปรแกรมสัญญาณไฟจราจรอัตโนมัติแบบอิสระต่อกันจำนวน1 แยก

ในการทดลองนี้ มีวัตถุประสงค์เพื่อทดสอบการทำงาน ของโปรแกรมเครื่องควบคุมสัญญาณไฟจราจร โดยการทำงานของเครื่องควบคุมไฟจราจรนี้นั้น สัญญาณไฟจราจรจะทำงานแบบอัตโนมัติตามที่เราได้ทำการโปรแกรมไว้ก่อนหน้านี้ โดยค่าช่วงของการปล่อยสัญญาณไฟจราจรนี้เราจะทำการเก็บข้อมูลจากสถิติ

สำหรับโปรแกรมที่ใช้ในการทดลองนี้ ก็คือโปรแกรมของเครื่องควบคุมไฟจราจรจริง ๆ เพียงแต่จะยังไม่สามารถทำงานเชื่อมต่อกับเครื่องควบคุมไฟจราจรเครื่องอื่น ๆ ได้เท่านั้น ซึ่งตัวโปรแกรมนั้นจะแสดงอยู่ในการทดลองสุดท้าย คือ การทดลองโปรแกรมทั้งหมดของเครื่องควบคุมสัญญาณไฟจราจร (เพียงแต่ตัด

ส่วนที่เป็นโปรแกรมเชื่อมต่อกับอุปกรณ์ควบคุมตัวอื่นออกเท่านั้น โดยมีการทดลองเป็นสอง สถานะ คือ ไฟจราจรแบบ Auto โดยสามารถเลือกเวลาหน่วงที่กำหนดไว้เป็นช่วงๆและเลือกสถานะ โดย การใช้ Dip Switch และการแสดงผลไฟกระพริบ บล็อกไดอะแกรมของการทำงานของสัญญาณไฟจราจร 1 แยก

รูปที่ 4.3 แสดงบล็อกไดอะแกรมของการทำงานของสัญญาณไฟจราจร 1 แยก

สำหรับโปรแกรมที่ใช้ในการทดลองมีดังนี้

```
/* FLASH.C - LED Flasher for the Keil MCB251 Evaluation Board with
80C51 device*/

#include <REG51.H>

void wait (void)  {                    /* wait function */
  ;                                    /* only to delay for LED flashes
*/
}

void main (void)  {
  unsigned long x,y;
  unsigned long i,a,b,c,d ;                                    /* Delay
var */
  unsigned char j,P2_0,P2_1,P2_2,P2_3,P2_4,P2_5,P2_6,P2_7;
/* LED var */
  unsigned char dsw;                   /* DIVSW var */
  dsw=P2;
  P2_0=(dsw&0x01);
  P2_1=(dsw&0x02);
  P2_2=(dsw&0x04);
  P2_3=(dsw&0x08);
  P2_4=(dsw&0x10);
  P2_5=(dsw&0x20);
  P2_6=(dsw&0x40);
  P2_7=(dsw&0x80);
  x=30000;
  y=20000;
  a=90000;
  b=60000;
  c=90000;
  d=40000;
  /* SELECT AUTO YELLOW BLINK */
  if (P2_0==0x00) {
      while (1) {                  /* Output to LED Port */
        P0 = 0x92;
        P1 = 0x04;
                        /* Output to LED Port */
      for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
       wait ();                    /* call wait function */
      }
        P0 = 0x00;                          /* Output to LED Port */
        P1 = 0x00;                          /* Output to LED Port */
      for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
       wait ();                    /* call wait function */
      }
      }
  }

  /* AUTO NORMAL TRAFFIC */
 // else {
      if ((P2_1==0x00)&&(P2_2==0x00)) { /* SUB NORMAL 1 */
      while (1) {                        /* Loop forever */
  // 1 2109
        P0 = 0x21;                        /* Output to LED Port */
        P1 = 0x09;                        /* Output to LED Port */
      for (i = 0; i < a; i++)  {  /* Delay for 10000 Counts */
       wait ();                    /* call wait function */
      }
```

```
// 2 2209
      P0 = 0x22;                          /* Output to LED Port */
      P1 = 0x09;                          /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 3 2409
      P0 = 0x24;                          /* Output to LED Port */
      P1 = 0x09;                          /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 4 0C09
      P0 = 0x0C;                          /* Output to LED Port */
      P1 = 0x09;                          /* Output to LED Port */
    for (i = 0; i < a; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 5 1409
      P0 = 0x14;                          /* Output to LED Port */
      P1 = 0x09;                          /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 6 2409
      P0 = 0x24;                          /* Output to LED Port */
      P1 = 0x09;                          /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 7 6408
      P0 = 0x64;                          /* Output to LED Port */
      P1 = 0x08;                          /* Output to LED Port */
    for (i = 0; i < a; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 8 A408
      P0 = 0xA4;                          /* Output to LED Port */
      P1 = 0x08;                          /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 9 2409
      P0 = 0x24;                          /* Output to LED Port */
      P1 = 0x09;                          /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 10 2403
      P0 = 0x24;                          /* Output to LED Port */
      P1 = 0x03;                          /* Output to LED Port */
    for (i = 0; i < a; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 11 2405
      P0 = 0x24;                          /* Output to LED Port */
      P1 = 0x05;                          /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                             /* call wait function */
    }
// 12 2409
```

```
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
}
    }
    if ((P2_1==0x00)&&(P2_2==0x04)) { /* SUB NORMAL 2 */
          while (1) {                              /* Loop forever */
// 1 2109
      P0 = 0x21;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < b; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 2 2209
      P0 = 0x22;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 3 2409
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 4 0C09
      P0 = 0x0C;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < b; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 5 1409
      P0 = 0x14;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 6 2409
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 7 6408
      P0 = 0x64;                              /* Output to LED Port */
      P1 = 0x08;                              /* Output to LED Port */
    for (i = 0; i < b; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 8 A408
      P0 = 0xA4;                              /* Output to LED Port */
      P1 = 0x08;                              /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 9 2409
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
```

```
    wait ();                              /* call wait function */
   }
// 10 2403
     P0 = 0x24;                           /* Output to LED Port */
     P1 = 0x03;                           /* Output to LED Port */
   for (i = 0; i < b; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
// 11 2405
     P0 = 0x24;                           /* Output to LED Port */
     P1 = 0x05;                           /* Output to LED Port */
   for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
// 12 2409
     P0 = 0x24;                           /* Output to LED Port */
     P1 = 0x09;                           /* Output to LED Port */
   for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
}
   }

   if ((P2_1==0x02)&&(P2_2==0x00)) { /* SUB NORMAL 3 */
       while (1) {                        /* Loop forever */
// 1 2109
     P0 = 0x21;                           /* Output to LED Port */
     P1 = 0x09;                           /* Output to LED Port */
   for (i = 0; i < c; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
// 2 2209
     P0 = 0x22;                           /* Output to LED Port */
     P1 = 0x09;                           /* Output to LED Port */
   for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
// 3 2409
     P0 = 0x24;                           /* Output to LED Port */
     P1 = 0x09;                           /* Output to LED Port */
   for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
// 4 0C09
     P0 = 0x0C;                           /* Output to LED Port */
     P1 = 0x09;                           /* Output to LED Port */
   for (i = 0; i < c; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
// 5 1409
     P0 = 0x14;                           /* Output to LED Port */
     P1 = 0x09;                           /* Output to LED Port */
   for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
// 6 2409
     P0 = 0x24;                           /* Output to LED Port */
     P1 = 0x09;                           /* Output to LED Port */
   for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
    wait ();                              /* call wait function */
   }
```

```
// 7 6408
      P0 = 0x64;                              /* Output to LED Port */
      P1 = 0x08;                              /* Output to LED Port */
    for (i = 0; i < c; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 8 A408
      P0 = 0xA4;                              /* Output to LED Port */
      P1 = 0x08;                              /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 9 2409
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 10 2403
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x03;                              /* Output to LED Port */
    for (i = 0; i < c; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 11 2405
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x05;                              /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 12 2409
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
}
    }
    if ((P2_1==0x02)&&(P2_2==0x04)) {  /* SUB NORMAL 4 */
          while (1) {                        /* Loop forever */
// 1 2109
      P0 = 0x21;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < d; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 2 2209
      P0 = 0x22;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 3 2409
      P0 = 0x24;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                                 /* call wait function */
    }
// 4 0C09
      P0 = 0x0C;                              /* Output to LED Port */
      P1 = 0x09;                              /* Output to LED Port */
```

```
    for (i = 0; i < d; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
// 5 1409
      P0 = 0x14;                 /* Output to LED Port */
      P1 = 0x09;                 /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
// 6 2409
      P0 = 0x24;                 /* Output to LED Port */
      P1 = 0x09;                 /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
// 7 6408
      P0 = 0x64;                 /* Output to LED Port */
      P1 = 0x08;                 /* Output to LED Port */
    for (i = 0; i < d; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
// 8 A408
      P0 = 0xA4;                 /* Output to LED Port */
      P1 = 0x08;                 /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
// 9 2409
      P0 = 0x24;                 /* Output to LED Port */
      P1 = 0x09;                 /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
// 10 2403
      P0 = 0x24;                 /* Output to LED Port */
      P1 = 0x03;                 /* Output to LED Port */
    for (i = 0; i < d; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
// 11 2405
      P0 = 0x24;                 /* Output to LED Port */
      P1 = 0x05;                 /* Output to LED Port */
    for (i = 0; i < x; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
// 12 2409
      P0 = 0x24;                 /* Output to LED Port */
      P1 = 0x09;                 /* Output to LED Port */
    for (i = 0; i < y; i++)  {  /* Delay for 10000 Counts */
     wait ();                    /* call wait function */
    }
 }
     }

 //}


}
```

ผลการทดลอง

หลังจากที่ได้ทดลองตามขั้นตอนข้างต้นแล้ว โดยทำการสังเกตและเก็บผลด้วยการถ่ายรูป ตำแหน่งพอร์ทที่ 0 และ 1 แล้วปรากฏว่าข้อมูลที่ส่งออกมาทางพอร์ท 0 และ 1 แสดงเป็นไฟวิ่งเริ่มจากไฟเขียว ไฟเหลือง และไฟแดงตามลำดับและเปลี่ยนไป สถานะอื่น ตามโปรแกรมที่เป็นไปได้ของสัญญาณไฟจราจร และสามารถเปลี่ยนเป็นโหมดไฟกระพริบ รวมทั้งสามารถเลือกช่วงโปรแกรมที่ได้กำหนดไว้ได้ แสดงว่า 89C51 บอร์ดแยกสัญญาณ สามารถทำงานได้ตามวัตถุประสงค์ของการทดลอง โดยรูปผลการทดลองแสดงได้ดังนี้

รูปที่ 4.4 แสดงการทำการทดสอบการทำงานของโปรแกรมสัญญาณไฟจราจรอัตโนมัติแบบอิสระต่อกัน จำนวน1 แยก

4.4 การทดสอบการทำงานของอุปกรณ์แสดงผล LCD และ อุปกรณ์รับข้อมูล (Keypad)

ในการทดลองนี้มีวัตถุประสงค์ เพื่อทดสอบการทำงานของอุปกรณ์ LCD และอุปกรณ์รับข้อมูล (Keypad) ว่าสามารถใช้งานได้จริง โดยเราจะใช้อุปกรณ์นี้ เพื่อเป็นตัวรับค่า และ แสดงผล ในการเลือกใช้โปรแกรมโหมดต่างๆ

วิธีทำการทดลอง ทำการต่อชุดอุปกรณ์ LCD และชุดรับข้อมูล กับวงจรควบคุมสัญญาณไฟจราจร โดยการทดสอบการทำงานของอุปกรณ์ LCD และ อุปกรณ์รับข้อมูล(Keypad)โดยจะใช้โปรแกรมซึ่งจะทำให้อุปกรณ์ LCD สามารถแสดงผลได้ และ สามารถรับค่าจากอุปกรณ์รับข้อมูลได้ถูกต้อง

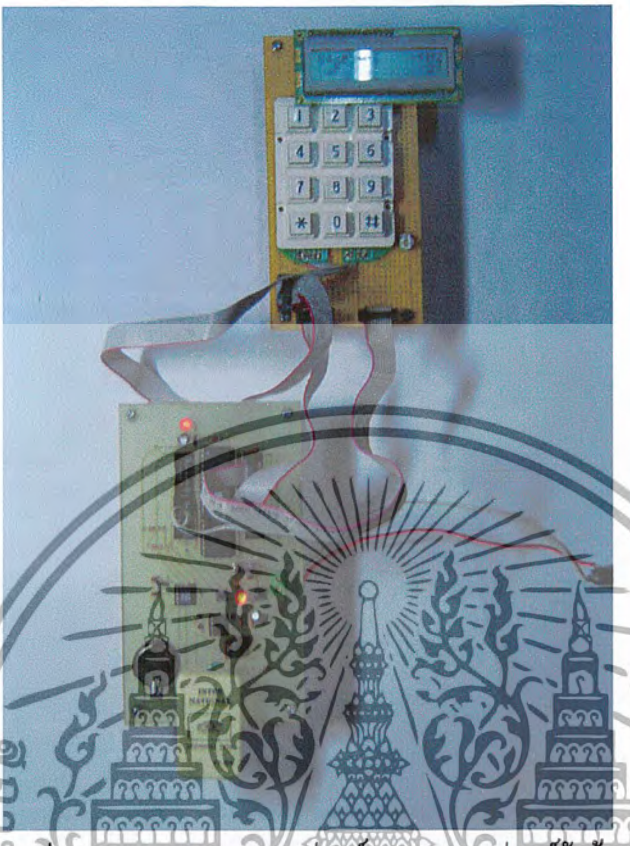
โดยจะมีโปรแกรมที่ใช้ทดลองดังต่อไปนี้

```
#pragma code
#include <stdio.h>
#include <reg52.h>
#include <string.h>
#include <intrins.h>
#include <absacc.h>
code unsigned char keycode[] = {0xb7,0xee,0xbe,0xde,

0xed,0xbd,0xdd,0xeb,

0xbb,0xdb,0xe7,0xd7};
code unsigned char userkey[] = {0x01,0x02,0x03,0x04,0x0b};
sbit lcd_en          =    P3^6;
sbit lcd_rs          =    P3^7;



unsigned char line_lcd[16];
unsigned char  key,keycount,digit;
unsigned char userpress[4];
bit keypress,output;
void dmsec (unsigned int count)
{
    unsigned char i;
    while (count) {
        for (i=1;i<=228;i++);
        count--;
    }
}

unsigned char scankey()
{     unsigned char i,j,k,a,x;
      k = 0xef;
      for (i=0;i<=2;i++)              //column
      {
            a = k|0xf;
            P2 = a;
            x  = P2;
            x  = x&0xf;
          dmsec(1);
            if (x != 0x0f)
```

```
                {
                        if(!keypress)
                        {
                                keypress = 1;
                                a = k&0xf0;
                                x = x|a;
                                //P3 = x;
                                for(j=0;j<=11;j++)
                                if(x==keycode[j]) return(j);
                                return(0xff);
                         }
                return (0xff);
                }
                k = k<<1;
        }
        digit += 1;
        keypress = 0;
        return(0xff);
}

//-----------------------For LCD---------------------------

void lcd_delay (void) {                         // pluse delay
    unsigned char i;
    i = 100;
    while (i>0) i--;

}
void pulse (void)
{       lcd_en          =       1;
        lcd_delay();
        lcd_en          =       0;
        lcd_delay();


}
void lcd_command(unsigned char command)
{
        lcd_rs                  =       0;
        P0              =       command;
        pulse();
}
void init_lcd (void)
{
        dmsec(100);
        lcd_rs          =       0;
        lcd_command(0x38);                                              //8
Bit Mode
        lcd_command(0x08);
//Display off
        lcd_command(0x01);
//Display Clear
        lcd_command(0x06);
//Entry Mode
        lcd_command(0x02);
//Return Home


}
void set_address(unsigned char address)
{
```

```
        lcd_rs      =     0;
        address |= 0x80;
        P0    =     address;
        pulse();
}
void write_lcd (void)
{       unsigned char i;
        set_address(0x00);
        for (i=0;i<=7;i++)
        {       lcd_rs      =     1;
                P0          =     line_lcd[i];
                pulse();
        }
        set_address(0x40);
                for (i=8;i<=15;i++)
        {       lcd_rs      =     1;
                P0          =     line_lcd[i];
                pulse();
        }
        lcd_command(0x0c);// Display On
}

//-----------------------End LCD-------------------------

void main()
{       //led_green = 1
        init_lcd();
        while(1)
        {
                digit = 0;
                key = scankey();//0123456789012345
                if ((key < 10)&&(digit<=2))
                {
                                                //0123456789012345
                        strncpy(line_lcd,"              ",16);
                        line_lcd[14] = key+0x30;
                        write_lcd();
                }
        }

}
```

สรุปผลการทดลอง จอ LCD สามารถแสดงผลได้ถูกต้องตามที่ได้มีการป้อนข้อมูลเข้าไป



รูปที่ 4.5 รูปแสดงผลการทดลองของอุปกรณ์ LCD และ อุปกรณ์รับข้อมูล (Keypad)

4.5 การทดลองการทำงานของอุปกรณ์ให้สัญญาณนาฬิกา

ในการทดลองนี้มีวัตถุประสงค์ เพื่อทำการทดสอบการทำงานของอุปกรณ์ให้สัญญาณนาฬิกา ซึ่งใช้เป็นตัวอ้างอิงในการกำหนดช่วงเวลาต่างๆ

วิธีทำการทดลอง ทำการติดต่อชุดอุปกรณ์ให้สัญญาณนาฬิกา กับวงจรควบคุมสัญญาณไฟจราจร โดยการทดสอบการทำงานของอุปกรณ์ให้สัญญาณนาฬิกา เราจะทำการโปรแกรม ชุดอุปกรณ์ให้สัญญาณนาฬิกาแสดงเวลา เป็น ชั่วโมง นาที และ วินาที ออกทางอุปกรณ์แสดงผล LCD

โดยจะมีโปรแกรมที่ใช้ทดลองดังต่อไปนี้

```
//Program : Real Time Clock DS1307
#include <stdio.h>
#include <reg52.h>
#include <string.h>
#include <intrins.h>
#include <absacc.h>
```

```
sbit IO                                 =    P1^0;
sbit CLK                =       P1^1;
sbit lcd_en             =       P3^6;
sbit lcd_rs             =       P3^7;


unsigned char line_lcd[16];
unsigned char time_dat[7];

void dmsec (unsigned int count)
{
    unsigned char i;
    while (count)
      {
        for (i=1;i<=228;i++);
        count--;
    }
}
void smcdelay (void)
{
    unsigned char i;
    i = 20;
    while (i>0) i--;
}
//----------------------For LCD-------------------------
void lcd_delay (void) {                 // pluse delay
    unsigned char i;
    i = 100;
    while (i>0) i--;
}
void pulse (void)
{       lcd_en          =       1;
        lcd_delay();
        lcd_en          =       0;
        lcd_delay();

}
void lcd_command(unsigned char command)
{
        lcd_rs                  =       0;
        P0              =       command;
        pulse();
}
void init_lcd (void)
{
        dmsec(100);
        lcd_rs          =       0;
        lcd_command(0x38);                                      //8
Bit Mode
        lcd_command(0x08);
//Display off
        lcd_command(0x01);
//Display Clear
        lcd_command(0x06);
//Entry Mode
        lcd_command(0x02);
//Return Home

}
void set_address(unsigned char address)
{
```

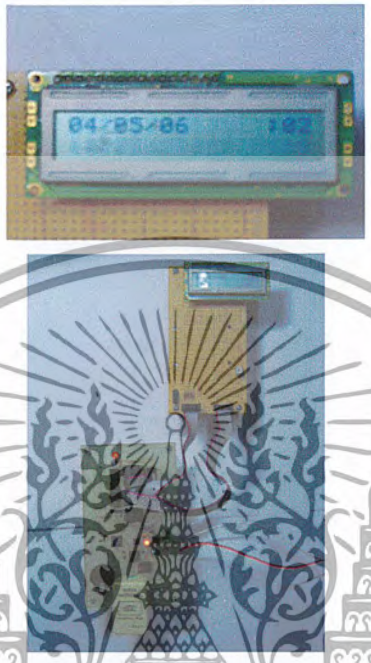
```
        lcd_rs      =     0;
        address |= 0x80;
        P0    =     address;
        pulse();
}
void write_lcd (void)
{     unsigned char i;
      for (i=0;i<=7;i++)
      {     lcd_rs      =     1;
            P0          =     line_lcd[i];
            pulse();
      }
      set_address(0x40);
            for (i=8;i<=15;i++)
      {     lcd_rs      =     1;
            P0          =     line_lcd[i];
            pulse();
      }
      lcd_command(0x0c);// Display On
}
//-----------------------End LCD-------------------------
void i2cstart (void) {                    // start condition
    CLK = 0;
    IO  = 1;
    CLK = 1;
    smcdelay ();
    IO  = 0;
    smcdelay ();
    CLK = 0;
    smcdelay ();
}

void i2cstop (void) {                     // stop condition
    IO  = 0;
    CLK = 1;
    smcdelay ();
    IO  = 1;
    smcdelay ();
    CLK = 0;
    smcdelay ();
}
void ack (void)
{
      IO = 0;
      _nop_();
    CLK = 1;
      smcdelay ();
      CLK = 0;
      smcdelay ();
}
bit wr_i2c (unsigned char x)              // Write 1 Byte
{     unsigned char i;
      bit   outport;
      for (i=1;i<=8;i++)
      {
            outport     =     x & 0x80;               //0x80 =
10000000B
            IO          =     outport;
            x = x<<1;                               //MSB first
            CLK = 1;
        smcdelay ();
```

```
            CLK = 0;
            smcdelay ();
          }
          IO          =     1;
          CLK = 1;
        smcdelay ();
          outport = IO;                                    // Check
ACK
        CLK = 0;
        smcdelay ();
          return(0);
      }
      unsigned char rd_i2c ()                          // Read 1 Byte
      {
            unsigned char i,dat ;
            bit inport;
            dat = 0;
            IO = 1;
            for (i=1;i<=8;i++)
            {
            CLK = 1;
            smcdelay ();
                  inport = IO;
                  dat = dat<<1;
                  dat = dat|inport;
            CLK = 0;
            smcdelay ();
            }
            return(dat);
      }
      void wr_ds1307(unsigned char address,dat)                //
Write 1 Byte
      {
            i2cstart();
            wr_i2c(0xd0);
            wr_i2c(address);
            wr_i2c(dat);
            i2cstop();
      }
      unsigned char rd_ds1307(unsigned char address)           //
Read 1 Byte
      {     unsigned char temp;
            i2cstart();
            wr_i2c(0xd0);
            wr_i2c(address);
            i2cstart();
            wr_i2c(0xd1);
            temp = rd_i2c();
            i2cstop();
            return(temp);
      }

      void write_time(unsigned char sec,min,hour,date,month,year)
      {
            wr_ds1307(0x00,sec);
            wr_ds1307(0x01,min);
            wr_ds1307(0x02,hour);
            wr_ds1307(0x04,date);
            wr_ds1307(0x05,month);
            wr_ds1307(0x06,year);
      }
```

```
void read_time()
{
      time_dat[0] = rd_ds1307(0x00);
      time_dat[1] = rd_ds1307(0x01);
      time_dat[2] = rd_ds1307(0x02);
      time_dat[3] = rd_ds1307(0x04);
      time_dat[4] = rd_ds1307(0x05);
      time_dat[5] = rd_ds1307(0x06);
}
void show_time()
{
      unsigned char ten,unit;

      set_address(0x00);
      //                dd/mm/yy   hh:mm
      //                          0123456789012345
      strncpy(line_lcd,"  /  /       :  ",16);

      ten = time_dat[3]&0xf0;                          //Date
      ten >>= 4;
      unit = time_dat[3]&0x0f;
      line_lcd[0] = ten + 0x30;
      line_lcd[1] = unit + 0x30;

      ten = time_dat[4]&0xf0;                          //month
      ten >>= 4;
      unit = time_dat[4]&0x0f;
      line_lcd[3] = ten + 0x30;
      line_lcd[4] = unit + 0x30;

      ten = time_dat[5]&0xf0;                          //Year
      ten >>= 4;
      unit = time_dat[5]&0x0f;
      line_lcd[6] = ten + 0x30;
      line_lcd[7] = unit + 0x30;

      ten = time_dat[1]&0xf0;                          //minute
      ten >>= 4;
      unit = time_dat[1]&0x0f;
      line_lcd[14] = ten + 0x30;
      line_lcd[15] = unit + 0x30;
      write_lcd();

      ten = time_dat[2]&0xf0;                          //hour
      ten >>= 4;
      unit = time_dat[2]&0x0f;
      line_lcd[11] = ten + 0x30;
      line_lcd[12] = unit + 0x30;

      write_lcd();
}
void main()
{
      dmsec(100);
      init_lcd();
      write_time(1,2,3,4,5,6);
      while(1)
      {
      dmsec(500);
      read_time();
```

```
            show_time();
            }
}
```

สรุปผลการทดลอง จอแสดงผล LCD สามารถแสดงเวลาตามที่โปรแกรมได้ดังนี้



รูปที่ 4.6 รูปแสดงผลการทดลองของอุปกรณ์ให้สัญญาณนาฬิกา

4.6 การทดสอบการทำงานของโปรแกรมสัญญาณไฟจราจรอัตโนมัติแบบเชื่อมต่อโดยใช้บอร์ดแยกสัญญาณจำนวน 2 แยก

ในการทดลองนี้มีวัตถุประสงค์ เพื่อทำการทดสอบการเชื่อมโยงของชุดควบคุมสัญญาณไฟจราจรจำนวน 2 ชุด โดยผ่านทางพอร์ท RS485

วิธีทำการทดลอง ทำการติดตั้งชุดอุปรณ์ SN75176BP กับวงจรควบคุมสัญญาณไฟจราจร และทำการเชื่อมโยงภาครับของวงจรควบคุมสัญญาณไฟจราจรชุดแรก กับ ภาคส่งของชุดควบคุมสัญญาณไฟจราจรชุดที่สอง และทำการเชื่อมโยงภาครับของวงจรควบคุมสัญญาณไฟจราจรชุดสอง กับ ภาคส่งของชุดควบคุมสัญญาณไฟจราจรชุดแรก หลังจากนั้นทำการโปรแกรมชุดควบคุมหลัก เพื่อไปควบคุมให้ชุดควบคุมรองทำงานตามคำสั่ง โดยจะโปรแกรมให้ชุดควบคุมรองมีสองโหมดการทำงาน โดยโหมดการทำงานแรกให้แสดงไฟกระพริบที่หลอดไฟ LED ที่หลอด 1-4 และโหมดที่สอง ให้แสดงไฟกระพริบที่หลอดไฟ LED หลอดที่ 5-8 หลังจากนั้น ให้กดป้อนข้อมูลเพื่อทำการเปลี่ยนโหมดการทำงานจากชุดควบคุมหลักส่งไปยังชุดควบคุมรอง

โดยจะมีโปรแกรมที่ใช้ทดลองดังต่อไปนี้

```
#pragma code
#include <stdio.h>
#include <reg52.h>
#include <string.h>
#include <intrins.h>
#include <absacc.h>

code unsigned char keycode[] = {0xd7,0xbe,0xde,0xee, //- 1 2 3
0xbd,0xdd,0xed,0xbb, //4 5 6 7
0xdb,0xeb,0xe7,0xe7};//8 9
/*
code unsigned char keycode[] = {0xb7,0xee,0xbe,0xde,
0xed,0xbd,0xdd,0xeb,
0xbb,0xdb,0xe7,0xd7};
*/
code unsigned char userkey[] = {0x01,0x02,0x03,0x04,0x0b};  //
sbit IO                              =   P1^0;
sbit CLK                =     P1^1;
sbit lcd_en             =     P3^6;
sbit lcd_rs             =     P3^7;
sbit en485              =           P3^4;

unsigned char line_lcd[16];
unsigned char time_dat[7];
unsigned char  key,keycount,digit,buff,keymode;
unsigned char userpress[4];
bit keypress,output;
void dmsec (unsigned int count)
{
    unsigned char i;
    while (count) {
       for (i=1;i<=228;i++);
       count--;
    }
}
void start (void)
{
    SCON = 0x52;                          // Rx Tx Enable
    TMOD = 0x21;                             //old 0x20
    TH1  = 0xfd;                          // speed 9600
    TR1  = 1;
}
void smcdelay (void)
{
    unsigned char i;
    i = 20;
    while (i>0) i--;
}
unsigned char scankey()
{     unsigned char i,j,k,a,x;
      k = 0xef;
      for (i=0;i<=2;i++)              //column
      {
            a = k|0xf;
            P2 = a;
            x  = P2;
            x  = x&0xf;
          dmsec(1);
            if (x != 0x0f)
            {
                  if(!keypress)
```

```
                        {
                              keypress = 1;
                              a = k&0xf0;
                              x = x|a;
                              //P3 = x;
                              for(j=0;j<=11;j++)
                              if(x==keycode[j]) return(j);
                              return(0xff);
                        }
                  return (0xff);
                  }
                  k = k<<1;
            }
            digit += 1;
            keypress = 0;
            return(0xff);
      }
      void send_byte (unsigned char data_s)
      {
            en485 = 1;
            TI = 0;
            SBUF = data_s;
            while(~TI);
      }
      //------------------------For LCD-----------------------------
      void lcd_delay (void) {                    // pluse delay
          unsigned char i;
          i = 100;
          while (i>0) i--;
      }
      void pulse (void)
      {     lcd_en      =     1;
            lcd_delay();
            lcd_en      =     0;
            lcd_delay();

      }
      void lcd_command(unsigned char command)
      {
            lcd_rs            =     0;
            P0          =     command;
            pulse();
      }
      void init_lcd (void)
      {
            dmsec(100);
            lcd_rs      =     0;
            lcd_command(0x38);                                    //8
Bit Mode
            lcd_command(0x08);
      //Display off
            lcd_command(0x01);
      //Display Clear
            lcd_command(0x06);
      //Entry Mode
            lcd_command(0x02);
      //Return Home

      }
      void set_address(unsigned char address)
      {
```

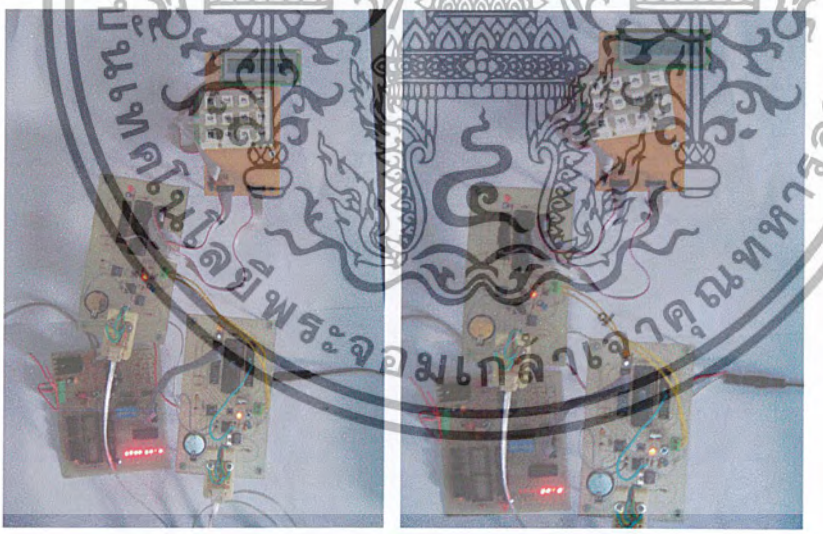
```
        lcd_rs      =     0;
        address |= 0x80;
        P0    =     address;
        pulse();
}
void write_lcd (void)
{     unsigned char i;
      for (i=0;i<=15;i++)
      {     lcd_rs      =     1;
            P0          =     line_lcd[i];
            pulse();
      }
      lcd_command(0x0c);// Display On
}
void i2cstart (void) {                    // start condition
    CLK = 0;
    IO  = 1;
    CLK = 1;
    smcdelay ();
    IO  = 0;
    smcdelay ();
    CLK = 0;
    smcdelay ();
}

void i2cstop (void) {                     // stop condition
    IO  = 0;
    CLK = 1;
    smcdelay ();
    IO  = 1;
    smcdelay ();
    CLK = 0;
    smcdelay ();
}
void ack (void)
{
      IO = 0;
      _nop_();
    CLK = 1;
      smcdelay ();
      CLK = 0;
      smcdelay ();
}
bit wr_i2c (unsigned char x)              // Write 1 Byte
{     unsigned char i;
      bit   outport;
      for (i=1;i<=8;i++)
      {
            outport     =     x & 0x80;             //0x80 =
10000000B
            IO          =     outport;
            x = x<<1;                               //MSB first
            CLK = 1;
        smcdelay ();
        CLK = 0;
        smcdelay ();
      }
      IO          =     1;
      CLK = 1;
     smcdelay ();
```

```
            outport = IO;                                    // Check
ACK
          CLK = 0;
          smcdelay ();
            return(0);
      }
      unsigned char rd_i2c ()                          // Read 1 Byte
      {
            unsigned char i,dat ;
            bit inport;
            dat = 0;
            IO = 1;
            for (i=1;i<=8;i++)
            {
            CLK = 1;
            smcdelay ();
                  inport = IO;
                  dat = dat<<1;
                  dat = dat|inport;
            CLK = 0;
            smcdelay ();
            }
            return(dat);
      }
      void wr_ds1307(unsigned char address,dat)                //
Write 1 Byte
      {
            i2cstart();
            wr_i2c(0xd0);
            wr_i2c(address);
            wr_i2c(dat);
            i2cstop();
      }
      unsigned char rd_ds1307(unsigned char address)           //
Read 1 Byte
      {     unsigned char temp;
            i2cstart();
            wr_i2c(0xd0);
            wr_i2c(address);
            i2cstart();
            wr_i2c(0xd1);
            temp = rd_i2c();
            i2cstop();
            return(temp);
      }

      void write_time(unsigned char sec,min,hour,date,month,year)
      {
            wr_ds1307(0x00,sec);
            wr_ds1307(0x01,min);
            wr_ds1307(0x02,hour);
            wr_ds1307(0x04,date);
            wr_ds1307(0x05,month);
            wr_ds1307(0x06,year);
      }

      void read_time()
      {
            time_dat[0] = rd_ds1307(0x00);
            time_dat[1] = rd_ds1307(0x01);
            time_dat[2] = rd_ds1307(0x02);
```

```
        time_dat[3] = rd_ds1307(0x04);
        time_dat[4] = rd_ds1307(0x05);
        time_dat[5] = rd_ds1307(0x06);
}
void show_time()
{
        unsigned char ten,unit;

        set_address(0x00);
        //                  dd/mm/yy   hh:mm
        //                          0123456789012345
        strncpy(line_lcd,"  /  /       :  ",16);

        ten = time_dat[3]&0xf0;                         //Date
        ten >>= 4;
        unit = time_dat[3]&0x0f;
        line_lcd[0] = ten + 0x30;
        line_lcd[1] = unit + 0x30;

        ten = time_dat[4]&0xf0;                         //month
        ten >>= 4;
        unit = time_dat[4]&0x0f;
        line_lcd[3] = ten + 0x30;
        line_lcd[4] = unit + 0x30;

        ten = time_dat[5]&0xf0;                         //Year
        ten >>= 4;
        unit = time_dat[5]&0x0f;
        line_lcd[6] = ten + 0x30;
        line_lcd[7] = unit + 0x30;

        ten = time_dat[1]&0xf0;                         //minute
        ten >>= 4;
        unit = time_dat[1]&0x0f;
        line_lcd[14] = ten + 0x30;
        line_lcd[15] = unit + 0x30;
        write_lcd();

        ten = time_dat[2]&0xf0;                         //hour
        ten >>= 4;
        unit = time_dat[2]&0x0f;
        line_lcd[11] = ten + 0x30;
        line_lcd[12] = unit + 0x30;

        write_lcd();
}
//-----------------------End LCD-------------------------
void main()
{       //led_green = 1
        unsigned char i,j;
        j = 1;
        init_lcd();
        //write_time(1,2,3,4,5,6);
        start();
        while(1)
        {
                digit = 0;
                //read_time();
                //show_time();
                key = scankey();//0123456789012345
                if (key < 10)
```

```
        {
                lcd_command(0x01);
        //Display Clear

                if ((key > 0)&&(key < 10))
                {
                        keymode = key;

//0123456789012345
                        strncpy(line_lcd,"        mode    ",16);
                        line_lcd[14] = key+0x30;
                        set_address(0x00);
                        write_lcd();
                }
                if ((key == 0)&&(keymode>0))
                {
                        send_byte(keymode);

//0123456789012345
                        strncpy(line_lcd,"  send command  ",16);
                        set_address(0x40);
                        write_lcd();
                }
            }
        }
}
```

ผลการทดลอง



รูปที่ 4.7 รูปแสดงผลการทดสอบการทำงานของโปรแกรมสัญญาณไฟจราจรอัตโนมัติแบบเชื่อมต่อโดยใช้บอร์ดแยกสัญญาณ จำนวน 2 แยก

สรุปผลการทดลอง จากโปรแกรมข้างต้น ชุดควบคุมหลักสามารถเปลี่ยนโหมดการทำงานของชุดควบคุมรอง ให้เป็นไปตามโปรแกรมตามที่ต้องการได้

4.7 การทดลองของชุดควบคุมสัญญาณไฟจราจรจำนวน 4 สี่แยก

ในการทดลองนี้มีวัตถุประสงค์ เพื่อทำการทดสอบการเชื่อมโยงของชุดควบคุมสัญญาณไฟจราจรจำนวน 4 ชุด โดยผ่านทางพอร์ท RS485

วิธีทำการทดลอง จะทำการโปรแกรมตัวส่ง โดยโปรแกรมจะมีการทำงานแบ่งเป็น 3 โหมดเมื่อกดเลข 1 นั้นจะเข้าโหมดที่ 1 ตัวรับจะได้รับคำสั่งให้ เป็นสัญญาณไฟอิสระต่อกัน ซึ่งสี่แยกแต่ละแยกนั้นจะให้สัญญาณไฟที่อิสระต่อกัน เมื่อกดเลข 2 ตัวรับจะได้รับคำสั่งให้เข้าสู่โหมดสอง ซึ่งสัญญาณไฟนั้นมีการเชื่อมต่อกันตามที่ได้โปรแกรม และเมื่อกดเลข 3 นั้นตัวรับจะเข้าสู่โหมดสาม ทำให้แต่ละสี่แยกนั้นแสดงผลเป็นสัญญาณไฟเหลืองกระพริบ

โดยจะมีโปรแกรมบอร์ดส่งสัญญาณดังต่อไปนี้

```
#pragma code
#include <stdio.h>
#include <reg52.h>
#include <string.h>
#include <intrins.h>
#include <absacc.h>

code unsigned char keycode[] = {0xd7,0xbe,0xde,0xee, //- 1 2 3
                                0xbd,0xdd,0xed,0xbb,
//4 5 6 7

      0xdb,0xeb,0xe7,0xe7};//8 9
/*
code unsigned char keycode[] = {0xb7,0xee,0xbe,0xde,
                                0xed,0xbd,0xdd,0xeb,
                                0xbb,0xdb,0xe7,0xd7};
                                */
code unsigned char userkey[] = {0x01,0x02,0x03,0x04,0x0b};  //
sbit IO                 =     P1^0;
sbit CLK                =     P1^1;
sbit lcd_en             =     P3^6;
sbit lcd_rs             =     P3^7;
sbit en485              =  P3^4;

unsigned char line_lcd[16];
unsigned char time_dat[7];
unsigned char  key,keycount,digit,buff,keymode;
unsigned char userpress[4];
bit keypress,output;
void dmsec (unsigned int count)
{
    unsigned char i;
    while (count) {
        for (i=1;i<=228;i++);
        count--;
    }
}
void start (void)
```

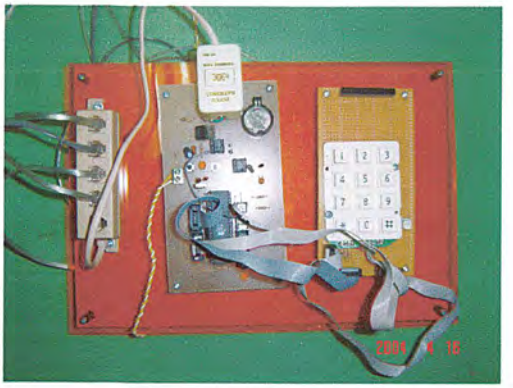
```
{
    SCON = 0x52;                          // Rx Tx Enable  52 old
    TMOD = 0x21;                                // old 0x20
    TH1  = 0xfd;                          // speed 9600
      TL1  = 0xfd;
      IE     = 0x90;
    TR1  = 1;
}
void smcdelay (void)
{
    unsigned char i;
    i = 20;
    while (i>0) i--;
}
unsigned char scankey()
{     unsigned char i,j,k,a,x;
      k = 0xef;
      for (i=0;i<=2;i++)                    //column
      {
            a = k|0xf;
            P2 = a;
            x  = P2;
            x  = x&0xf;
          dmsec(1);
            if (x != 0x0f)
            {
                if(!keypress)
                {
                  keypress = 1;
                  a = k&0xf0;
                  x = x|a;
                  //P3 = x;
                  for(j=0;j<=11;j++)
                  if(x==keycode[j]) return(j);
                  return(0xff);
                }
            return (0xff);
            }
            k = k<<1;
      }
      digit += 1;
      keypress = 0;
      return(0xff);
}
void send_byte (unsigned char data_s)
{
      en485 = 1;
      TI = 0;
      SBUF = data_s;
      while(~TI);
}
//-----------------------For LCD--------------------------
void lcd_delay (void) {                     // pluse delay
    unsigned char i;
    i = 100;
    while (i>0) i--;
}
void pulse (void)
{     lcd_en      =     1;
      lcd_delay();
      lcd_en      =     0;
```

```
        lcd_delay();

}
void lcd_command(unsigned char command)
{
        lcd_rs              =     0;
        P0          =     command;
        pulse();
}
void init_lcd (void)
{
        dmsec(100);
        lcd_rs      =     0;
        lcd_command(0x38);                                    //8 Bit
Mode
        lcd_command(0x08);                                    //Display
off
        lcd_command(0x01);                                    //Display
Clear
        lcd_command(0x06);                                    //Entry
Mode
        lcd_command(0x02);                                    //Return
Home

}
void set_address(unsigned char address)
{
        lcd_rs      =     0;
        address |= 0x80;
        P0      =     address;
        pulse();
}
void write_lcd (void)
{       unsigned char i;
        for (i=0;i<=15;i++)
        {       lcd_rs      =     1;
                P0          =     line_lcd[i];
                pulse();
        }
        lcd_command(0x0c);// Display On
}
void i2cstart (void) {                          // start condition
    CLK = 0;
    IO  = 1;
    CLK = 1;
    smcdelay ();
    IO  = 0;
    smcdelay ();
    CLK = 0;
    smcdelay ();
}

void i2cstop (void) {                           // stop condition
    IO  = 0;
    CLK = 1;
    smcdelay ();
    IO  = 1;
    smcdelay ();
    CLK = 0;
    smcdelay ();
}
```

```
void ack (void)
{
      IO = 0;
      _nop_();
    CLK = 1;
      smcdelay ();
      CLK = 0;
      smcdelay ();
}
bit wr_i2c (unsigned char x)              // Write 1 Byte
{     unsigned char i;
      bit   outport;
      for (i=1;i<=8;i++)
      {
            outport     =     x & 0x80;              //0x80 =
10000000B
            IO          =     outport;
            x = x<<1;                              //MSB first
            CLK = 1;
        smcdelay ();
        CLK = 0;
        smcdelay ();
      }
      IO          =     1;
      CLK = 1;
    smcdelay ();
      outport = IO;                                // Check ACK
    CLK = 0;
    smcdelay ();
      return(0);
}
unsigned char rd_i2c ()                          // Read 1 Byte
{
      unsigned char i,dat ;
      bit inport;
      dat = 0;
      IO = 1;
      for (i=1;i<=8;i++)
      {
      CLK = 1;
      smcdelay ();
            inport = IO;
            dat = dat<<1;
            dat = dat|inport;
      CLK = 0;
      smcdelay ();
      }
      return(dat);
}
void wr_ds1307(unsigned char address,dat)                  // Write 1
Byte
{
      i2cstart();
      wr_i2c(0xd0);
      wr_i2c(address);
      wr_i2c(dat);
      i2cstop();
}
unsigned char rd_ds1307(unsigned char address)             // Read 1
Byte
{     unsigned char temp;
```

```
        i2cstart();
        wr_i2c(0xd0);
        wr_i2c(address);
        i2cstart();
        wr_i2c(0xd1);
        temp = rd_i2c();
        i2cstop();
        return(temp);
}

void write_time(unsigned char sec,min,hour,date,month,year)
{
        wr_ds1307(0x00,sec);
        wr_ds1307(0x01,min);
        wr_ds1307(0x02,hour);
        wr_ds1307(0x04,date);
        wr_ds1307(0x05,month);
        wr_ds1307(0x06,year);
}

void read_time()
{
        time_dat[0] = rd_ds1307(0x00);
        time_dat[1] = rd_ds1307(0x01);
        time_dat[2] = rd_ds1307(0x02);
        time_dat[3] = rd_ds1307(0x04);
        time_dat[4] = rd_ds1307(0x05);
        time_dat[5] = rd_ds1307(0x06);
}
void show_time()
{
        unsigned char ten,unit;

        set_address(0x00);
        //                  dd/mm/yy   hh:mm
        //                             0123456789012345
        strncpy(line_lcd,"  /  /       :  ",16);

        ten = time_dat[3]&0xf0;                    //Date
        ten >>= 4;
        unit = time_dat[3]&0x0f;
        line_lcd[0] = ten + 0x30;
        line_lcd[1] = unit + 0x30;

        ten = time_dat[4]&0xf0;                    //month
        ten >>= 4;
        unit = time_dat[4]&0x0f;
        line_lcd[3] = ten + 0x30;
        line_lcd[4] = unit + 0x30;

        ten = time_dat[5]&0xf0;                    //Year
        ten >>= 4;
        unit = time_dat[5]&0x0f;
        line_lcd[6] = ten + 0x30;
        line_lcd[7] = unit + 0x30;

        ten = time_dat[1]&0xf0;                    //minute
        ten >>= 4;
        unit = time_dat[1]&0x0f;
        line_lcd[14] = ten + 0x30;
        line_lcd[15] = unit + 0x30;
```

```
        write_lcd();

        ten = time_dat[2]&0xf0;                         //hour
        ten >>= 4;
        unit = time_dat[2]&0x0f;
        line_lcd[11] = ten + 0x30;
        line_lcd[12] = unit + 0x30;

        write_lcd();
}
//-----------------------End LCD--------------------------
void main()
{       //led_green = 1
        unsigned char i,j;
        j = 1;
        init_lcd();
        //write_time(1,2,3,4,5,6);
        start();
        keymode = 0;

        strncpy(line_lcd,"INSERT MODE      ",16);
                        line_lcd[14] = key+0x30;
                        set_address(0x00);
                        write_lcd();
        while(1)
        {
        //      digit = 0;
                //read_time();
                //show_time();
                key = scankey();//0123456789012345
                        lcd_command(0x01);
        //Display Clear

                        if ((key == 1)||(key == 2)||(key == 3))
                        {
                          keymode = key;
                                                        //0123456789012345
                          strncpy(line_lcd,"          mode    ",16);
                          line_lcd[14] = key+0x30;
                          set_address(0x00);
                          write_lcd();
                        }
                        if (key == 0)
                        {
                        //      if (keymode == 1) send_byte(1);
                        //      else if (keymode == 2) send_byte(2);
                                send_byte(keymode);
                        //0123456789012345
                              strncpy(line_lcd,"  send command  ",16);
                              line_lcd[14] = keymode+0x30;
                              set_address(0x40);
                              write_lcd();
                        }
        }                  //WHILE

}                       // MAIN
```



รูปที่ 4.8รูปบอร์ดส่งสัญญาณ

โดยมีโปรแกรมบอร์ดรับสัญญาณดังต่อไปนี้

```
#pragma code
#include <stdio.h>
#include <reg52.h>
#include <string.h>
#include <intrins.h>
#include <absacc.h>
code unsigned char keycode[] = {0xd7,0xbe,0xde,0xee, //- 1 2 3
                                0xbd,0xdd,0xed,0xbb,
//4 5 6 7

      0xdb,0xeb,0xe7,0xe7};//8 9


code unsigned char userkey[] = {0x01,0x02,0x03,0x04,0x0b}; //
sbit IO                 =    P1^0;
sbit CLK                =    P1^1;
sbit lcd_en             =    P3^6;
sbit lcd_rs             =    P3^7;
sbit en485              =    P3^4;


unsigned char line_lcd[16];
unsigned char rtc_data[16];
unsigned char time_dat[7];
unsigned char  key,keycount,digit,buffer,keymode,keyexit;
unsigned int i,check,count;
unsigned int green1,green2,green3,yellow,red;
unsigned char userpress[4];
bit keypress,output,send_bit;
bit TK_OK,RX_OK,change_mode;
unsigned char serial_text[10] = {'a','b','c','d','e','f','g'};
unsigned char temp,buff;


void dmsec (unsigned int count)
{
    unsigned char i;
    while ((count)&&(change_mode)) {
        for (i=1;i<=152;i++);
        count--;
    }
```

```
}
void start () {
    SCON = 0x50;                        // Rx Tx Enable  old 0x52
    TMOD = 0x21;                                //old 0x20
    TH1  = 0xfd;                          // speed 9600
      TL1 = 0xfd;
      IE     = 0x90;
    TR1  = 1;
}

void smcdelay (void)
{
    unsigned char i;
    i = 20;
    while (i>0) i--;
}
unsigned char scankey()
{     unsigned char i,j,k,a,x;
      k = 0xef;
      for (i=0;i<=2;i++)                //column
      {
            a = k|0xf;
            P2 = a;
            x  = P2;
            x  = x&0xf;
          dmsec(1);
            if (x != 0x0f)
            {
                  if(!keypress)
                  {
                    keypress = 1;
                    a = k&0xf0;
                    x = x|a;
                    //P3 = x;
                    for(j=0;j<=11;j++)
                    if(x==keycode[j]) return(j);
                    return(0xff);
                  }
            return (0xff);
            }
            k = k<<1;
      }
      digit += 1;
      keypress = 0;
      return(0xff);
}

//-----------------------For LCD---------------------------

void lcd_delay (void) {                   // pluse delay
    unsigned char i;
    i = 100;
    while (i>0) i--;
}

void pulse (void)
{     lcd_en      =     1;
      lcd_delay();
      lcd_en      =     0;
      lcd_delay();
```

```
}
void lcd_command(unsigned char command)

{
      lcd_rs            =     0;
      P0          =     command;
      pulse();
}
void init_lcd (void)
{
      dmsec(100);
      lcd_rs      =     0;
      lcd_command(0x38);                                    //8 Bit
Mode
      lcd_command(0x08);                                    //Display
off
      lcd_command(0x01);                                    //Display
Clear
      lcd_command(0x06);                                    //Entry
Mode
      lcd_command(0x02);                                    //Return
Home

}

void set_address(unsigned char address)
{
      lcd_rs      =     0;
      address |= 0x80;
      P0    =     address;
      pulse();
}

void write_lcd (void)
{     unsigned char i;
      for (i=0;i<=15;i++)
      {     lcd_rs      =     1;
            P0          =     line_lcd[i];
            pulse();
      }
      lcd_command(0x0c);// Display On
}

void i2cstart (void) {                          // start condition
    CLK = 0;
    IO  = 1;
    CLK = 1;
    smcdelay ();
    IO  = 0;
    smcdelay ();
    CLK = 0;
    smcdelay ();
}

void i2cstop (void) {                   // stop condition
    IO  = 0;
    CLK = 1;
    smcdelay ();
    IO  = 1;
    smcdelay ();
    CLK = 0;
```

```
    smcdelay ();
}

void ack (void)
{
      IO = 0;
      _nop_();
    CLK = 1;
      smcdelay ();
      CLK = 0;
      smcdelay ();
}
bit wr_i2c (unsigned char x)                    // Write 1 Byte
{     unsigned char i;
      bit   outport;
      for (i=1;i<=8;i++)
      {
            outport     =     x & 0x80;                  //0x80 =
10000000B
            IO          =     outport;
            x = x<<1;                                    //MSB first
            CLK = 1;
        smcdelay ();
        CLK = 0;
        smcdelay ();
      }
      IO          =     1;
      CLK = 1;
    smcdelay ();
      outport = IO;                                      // Check ACK
    CLK = 0;
    smcdelay ();
      return(0);
}
unsigned char rd_i2c ()                                  // Read 1 Byte
{
      unsigned char i,dat ;
      bit inport;
      dat = 0;
      IO = 1;
      for (i=1;i<=8;i++)
      {
      CLK = 1;
      smcdelay ();
            inport = IO;
            dat = dat<<1;
            dat = dat|inport;
      CLK = 0;
      smcdelay ();
      }
      return(dat);
}

void wr_ds1307(unsigned char address,dat)                // Write 1
Byte
{
      i2cstart();
      wr_i2c(0xd0);
      wr_i2c(address);
      wr_i2c(dat);
      i2cstop();
```

```
}
unsigned char rd_ds1307(unsigned char address)                    // Read 1
Byte
{     unsigned char temp;
      i2cstart();
      wr_i2c(0xd0);
      wr_i2c(address);
      i2cstart();
      wr_i2c(0xd1);
      temp = rd_i2c();
      i2cstop();
      return(temp);
}

void write_time(unsigned char sec,min,hour,date,month,year)
{
      wr_ds1307(0x00,sec);
      wr_ds1307(0x01,min);
      wr_ds1307(0x02,hour);
      wr_ds1307(0x04,date);
      wr_ds1307(0x05,month);
      wr_ds1307(0x06,year);
}

void read_time()
{
      time_dat[0] = rd_ds1307(0x00);
      time_dat[1] = rd_ds1307(0x01);
      time_dat[2] = rd_ds1307(0x02);
      time_dat[3] = rd_ds1307(0x04);
      time_dat[4] = rd_ds1307(0x05);
      time_dat[5] = rd_ds1307(0x06);
}

void show_time()
{
      unsigned char ten,unit;

      //set_address(0x00);
      //              dd/mm/yyhh:mm:ss
      //                0123456789012345
      //strncpy(line_lcd," /  /20        ",16);

      ten = time_dat[3]&0xf0;                              //Date
      ten >>= 4;
      unit = time_dat[3]&0x0f;
      //line_lcd[0] = ten + 0x30;
      //line_lcd[1] = unit + 0x30;
      //RTC
      rtc_data[0] = ten;
      rtc_data[1] = unit;

      ten = time_dat[4]&0xf0;                              //month
      ten >>= 4;
      unit = time_dat[4]&0x0f;
      //line_lcd[3] = ten + 0x30;
      //line_lcd[4] = unit + 0x30;
      //RTC
      rtc_data[3] = ten;
      rtc_data[4] = unit;
```

```
        ten = time_dat[5]&0xf0;                          //Year
        ten >>= 4;
        unit = time_dat[5]&0x0f;
        //line_lcd[8] = ten + 0x30;
        //line_lcd[9] = unit + 0x30;
        //RTC
        rtc_data[8] = ten;
        rtc_data[9] = unit;
        //write_lcd();

        //set_address(0x40);
        //                        hh:mm:ss
        //                        0123456789012345
        //strncpy(line_lcd,"          :  :  ",16);

        ten = time_dat[2]&0xf0;                          //hour
        ten >>= 4;
        unit = time_dat[2]&0x0f;
        //line_lcd[8] = ten + 0x30;
        //line_lcd[9] = unit + 0x30;
        //RTC
        rtc_data[8] = ten;
        rtc_data[9] = unit;

        ten = time_dat[1]&0xf0;                          //minute
        ten >>= 4;
        unit = time_dat[1]&0x0f;
        //line_lcd[11] = ten + 0x30;
        //line_lcd[12] = unit + 0x30;
        //RTC
        rtc_data[11] = ten;
        rtc_data[12] = unit;

        ten = time_dat[0]&0xf0;                          //second
        ten >>= 4;
        unit = time_dat[0]&0x0f;
        //line_lcd[14] = ten + 0x30;
        //line_lcd[15] = unit + 0x30;
        //RTC
        rtc_data[14] = ten;
        rtc_data[15] = unit;
        //write_lcd();
}

//----------------------End LCD-------------------------

unsigned char  receive_byte ()//Receive data sub routine
{
      en485 = 0;
      RI = 0;
      while(~RI) {
      }
      return(SBUF);
}
void send_byte (unsigned char data_s)
{
      EA    = 0;
      en485 = 1;
      TI = 0;
      SBUF = data_s;
      while(~TI);
```

```
        TI = 0;
        en485 = 0;
        EA    = 1;
}
//Serial Interrupt
void serial() interrupt 4
{
        if(TI)
        {
                TI=0;
                SBUF = buff;
                while(~TI);
                TI = 0;
                TK_OK = 1;
        }
        if(RI)
        {
                RI = 0;
                RX_OK =0;
                if ((SBUF==1)||(SBUF==8)||(SBUF==3))
                {
                      change_mode = 0;
                      temp = SBUF;
                }
                if (SBUF==8)
                send_bit = 1;
                SBUF = 0;
        }
}

void main()
{
        green1 = 4000;
        green2 = 6000;
        green3 = 8000;
        yellow = 2000;
        red = 1000;
        //init_lcd();
        start();
        write_time(0,0x11,9,8,4,4); //sec,min,hour,date,month,year
        change_mode = 1;
        check = 1;
        temp = 0xff;
        en485 = 0;                          //receiver
        while(1)
        {
                read_time();
                show_time();

                if (temp == 1) { // MODE 1 AUTO BY TIME
                      change_mode = 1;
                //05.00 - 06.59
                if ((rtc_data[8] == 0) && (rtc_data[9] >= 5) &&
(rtc_data[9] < 7)) {
                      // 1 2109
                            P0 = 0x21;
                            P2 = 0x09;
                      dmsec(green1);
                            // 2 2209
                            P0 = 0x22;
                            P2 = 0x09;
```

```
                dmsec(yellow);
                        // 3 2409
                        P0 = 0x24;
                        P2 = 0x09;
                dmsec(red);
                        // 4 0C09
                        P0 = 0x0C;
                        P2 = 0x09;
                        dmsec(green1);
                        // 5 1409
                        P0 = 0x14;
                        P2 = 0x09;
                dmsec(yellow);
                // 6 2409
                        P0 = 0x24;
                        P2 = 0x09;
                dmsec(red);
                // 7 6408
                        P0 = 0x64;
                        P2 = 0x08;
                dmsec(green1);
                        // 8 A408
                        P0 = 0xA4;

                        P2 = 0x08;
                dmsec(yellow);
                        // 9 2409
                        P0 = 0x24;
                        P2 = 0x09;
                dmsec(red);
                        // 10 2403
                        P0 = 0x24;
                        P2 = 0x03;
                        dmsec(green1);
                        // 11 2405
                        P0 = 0x24;
                        P2 = 0x05;
                dmsec(yellow);
                // 12 2409
                        P0 = 0x24;
                        P2 = 0x09;
                dmsec(red);
        }
                //07.00 - 08.59
                else if ((rtc_data[8] == 0) && (rtc_data[9] >= 7) &&
(rtc_data[9] < 9)) {
                // 1 2109
                        P0 = 0x21;
                        P2 = 0x09;
                dmsec(green2);
                        // 2 2209
                        P0 = 0x22;
                        P2 = 0x09;
                dmsec(yellow);
                        // 3 2409
                        P0 = 0x24;
                        P2 = 0x09;
                dmsec(red);
                        // 4 0C09
                        P0 = 0x0C;
                        P2 = 0x09;
```

```
                dmsec(green1);
                // 5 1409
                P0 = 0x14;
                P2 = 0x09;
            dmsec(yellow);
            // 6 2409
                P0 = 0x24;
                P2 = 0x09;
            dmsec(red);
            // 7 6408
                P0 = 0x64;
                P2 = 0x08;
            dmsec(green1);
                // 8 A408
                P0 = 0xA4;
                P2 = 0x08;
            dmsec(yellow);
                // 9 2409
                P0 = 0x24;
                P2 = 0x09;
            dmsec(red);
                // 10 2403
                P0 = 0x24;
                P2 = 0x03;
                dmsec(green1);
                // 11 2405
                P0 = 0x24;
                P2 = 0x05;
            dmsec(yellow);
            // 12 2409
                P0 = 0x24;
                P2 = 0x09;
            dmsec(red);
        }
        //09.00 - 15.59
        else if ((((rtc_data[8] == 0) && (rtc_data[9] == 9)))
|| ((rtc_data[8] == 1) && (rtc_data[9] < 6))) {
        // 1 2109
                P0 = 0x21;
                P2 = 0x09;
            dmsec(green1);
                // 2 2209
                P0 = 0x22;
                P2 = 0x09;
            dmsec(yellow);
                // 3 2409
                P0 = 0x24;
                P2 = 0x09;
            dmsec(red);
                // 4 0C09
                P0 = 0x0C;
                P2 = 0x09;
                dmsec(green1);
                // 5 1409
                P0 = 0x14;
                P2 = 0x09;
            dmsec(yellow);
            // 6 2409
                P0 = 0x24;
                P2 = 0x09;
            dmsec(red);
```

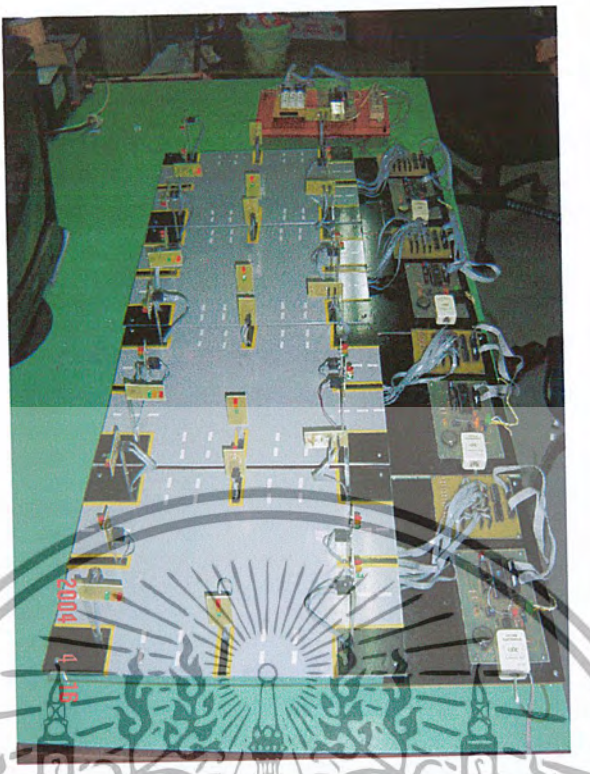
```
                // 7 6408
                    P0 = 0x64;
                    P2 = 0x08;
                dmsec(green1);
                    // 8 A408
                    P0 = 0xA4;
                    P2 = 0x08;
                dmsec(yellow);
                    // 9 2409
                    P0 = 0x24;
                    P2 = 0x09;
                dmsec(red);
                    // 10 2403
                    P0 = 0x24;
                    P2 = 0x03;
                    dmsec(green1);
                    // 11 2405
                    P0 = 0x24;
                    P2 = 0x05;
                dmsec(yellow);
                // 12 2409
                    P0 = 0x24;
                    P2 = 0x09;
                dmsec(red);
            }
            //16.00 - 19.59
            else if ((rtc_data[8] == 1) && (rtc_data[9] >= 6) &&
(rtc_data[9] < 9 )) {
            // 1 2109
                P0 = 0x21;
                P2 = 0x09;
            dmsec(green2);
                // 2 2209
                P0 = 0x22;
                P2 = 0x09;
            dmsec(yellow);
                // 3 2409
                P0 = 0x24;
                P2 = 0x09;
            dmsec(red);
                // 4 0C09
                P0 = 0x0C;
                P2 = 0x09;
                dmsec(green1);
                // 5 1409
                P0 = 0x14;
                P2 = 0x09;
            dmsec(yellow);
            // 6 2409
                P0 = 0x24;
                P2 = 0x09;
            dmsec(red);
            // 7 6408
                P0 = 0x64;
                P2 = 0x08;
            dmsec(green1);
                // 8 A408
                P0 = 0xA4;
                P2 = 0x08;
            dmsec(yellow);
                // 9 2409
```

```
                P0 = 0x24;
                P2 = 0x09;
        dmsec(red);
                // 10 2403
                P0 = 0x24;
                P2 = 0x03;
                dmsec(green1);
                // 11 2405
                P0 = 0x24;
                P2 = 0x05;
        dmsec(yellow);
        // 12 2409
                P0 = 0x24;
                P2 = 0x09;
        dmsec(red);
}
//20.00 - 23.59
        else if ((rtc_data[8] == 2) && (rtc_data[9] < 4 )) {
        // 1 2109
                P0 = 0x21;
                P2 = 0x09;
        dmsec(green1);
                // 2 2209
                P0 = 0x22;
                P2 = 0x09;
        dmsec(yellow);
                // 3 2409
                P0 = 0x24;
                P2 = 0x09;
        dmsec(red);
                // 4 0C09
                P0 = 0x0C;
                P2 = 0x09;
                dmsec(green1);
                // 5 1409
                P0 = 0x14;
                P2 = 0x09;
        dmsec(yellow);
        // 6 2409
                P0 = 0x24;
                P2 = 0x09;
        dmsec(red);
        // 7 6408
                P0 = 0x64;
                P2 = 0x08;
        dmsec(green1);
                // 8 A408
                P0 = 0xA4;
                P2 = 0x08;
        dmsec(yellow);
                // 9 2409
                P0 = 0x24;
                P2 = 0x09;
        dmsec(red);
                // 10 2403
                P0 = 0x24;
                P2 = 0x03;
                dmsec(green1);
                // 11 2405
                P0 = 0x24;
                P2 = 0x05;
```

```
                dmsec(yellow);
                // 12 2409
                        P0 = 0x24;
                        P2 = 0x09;
                dmsec(red);
        }
                //00.00 - 04.59
                //else if ((rtc_data[8] == 0) && (rtc_data[9] < 5 ))
{
                else {
                // 1 0492
                        P0 = 0x92;
                        P2 = 0x04;
                dmsec(yellow);
                        // 2 0000
                        P0 = 0x00;
                        P2 = 0x00;
                dmsec(yellow);
                }

        } // End MODE 1
        else if (temp == 8) { //MODE 2
                change_mode = 1;
                // 1 2109
                P0 = 0x21;
                P2 = 0x09;
        dmsec(green3);
                // 2 2209
                P0 = 0x22;
                P2 = 0x09;
        dmsec(yellow);
                // 3 2409
                P0 = 0x24;
                P2 = 0x09;
        dmsec(red);
                // 4 0C09
                P0 = 0x0C;
                P2 = 0x09;
                dmsec(green1);
                // 5 1409
                P0 = 0x14;
                P2 = 0x09;
        dmsec(yellow);
        // 6 2409
                P0 = 0x24;
                P2 = 0x09;
        dmsec(red);
        // 7 6408
                P0 = 0x64;
                P2 = 0x08;
        dmsec(green1);
                // 8 A408
                P0 = 0xA4;
                P2 = 0x08;
        dmsec(yellow);
                // 9 2409
                P0 = 0x24;
                P2 = 0x09;
        dmsec(red);
                // 10 2403
                P0 = 0x24;
```

```
            P2 = 0x03;
            dmsec(green1);
            // 11 2405
            P0 = 0x24;
            P2 = 0x05;
            dmsec(yellow);
// 12 2409
            P0 = 0x24;
            P2 = 0x09;
dmsec(red);

}
else if (temp == 3) { //MODE 3
            change_mode = 1;
            // 1 0492
            P0 = 0x92;
            P2 = 0x04;
dmsec(yellow);
            // 2 0000
            P0 = 0x00;
            P2 = 0x00;
dmsec(red);
}
else {
            change_mode = 1;
            // 1 0492
            P0 = 0x92;
            P2 = 0x04;
dmsec(yellow);
            // 2 0000
            P0 = 0x00;
            P2 = 0x00;
dmsec(red);
}
    }
}
```

รูปที่ 4.9 รูปบอร์ดรับสัญญาณ



รูปที่ 4.10 รูปบอร์ดส่งสัญญาณและรับสัญญาณที่ทำการเชื่อมต่อกัน

สรุปผลการทดลอง จากโปรแกรมข้างต้นเราสามารถโปรแกรมบอร์ดส่งสัญญาณให้มีการทำงานแบ่งเป็นสามโหมด โดยสามารถส่งสัญญาณแต่ละโหมดไปยังบอร์ครับสัญญาณได้ และทำงานตามที่ได้โปรแกรม

บทที่ 5
บทวิจารณ์และบทสรุป

จากผลการทดลอง เราสามารถทำการควบคุมสัญญาณไฟจราจร บริเวณสี่แยกจำนวนสี่ชุด โดยสามารถแบ่งโหมดการใช้งานเป็น สามโหมดกล่าวคือ

โหมด 1 ทำงานแบบอิสระต่อกัน โดยแต่ละแยกจะมีเวลาหน่วงแตกต่างกันซึ่งเป็นไปตามความหนาแน่นของการจราจรในแต่ละช่วงเวลาได้

โหมด 2 ทำงานแบบเชื่อมต่อกัน โดยมีชุดวงจรควบคุมหลักทำการควบคุมวงจรควบคุมรองแบบเชื่อมต่อกันตามโปรแกรมที่กำหนดได้

โหมด 3 ทำงานในโหมดสัญญาณไฟกระพริบ

เนื่องจากในแต่ละวัน การจราจรในแต่ละช่วงเวลามีความหนาแน่นไม่เท่ากัน ฉะนั้นการทำงานในโหมด 1 ซึ่งมีการทำงานแบบอิสระต่อกัน จะสามารถช่วยการจราจรนั้นเกิดความคล่องตัว โดยไม่ต้องมีผู้ควบคุมเนื่องจาก ได้มีการกำหนดในโปรแกรมให้เป็นแบบอัตโนมัติ แต่ถ้าความหนาแน่นของการจราจร นั้นมีปริมาณสูงกว่าปกติจนการทำงานในโหมด1 ไม่สามารถรองรับได้ก็จะมีการทำงานในโหมด2 ซึ่งมีการทำงานแบบเชื่อมต่อกัน จะสามารถช่วยระบายการจราจรในเส้นทางที่มีความหนาแน่นนั้น ได้ดีกว่าการทำงานในโหมดการทำงานแรก ส่วนจุดด้อยของการทำงานในโหมดนี้อาจมีอยู่บ้าง กล่าวคือ ต้องการเวลาในการเข้าสู่ระบบการทำงาน ประการที่สองจะทำให้ผู้ที่ใช้ถนนในเส้นทางอื่นๆ ต้องรอสัญญาณไฟเขียวนานขึ้น เนื่องจากมีการเพิ่มเวลาหน่วง ให้กับเส้นทางที่การจราจรหนาแน่นเพิ่มขึ้น ส่วนการทำงานในโหมดที่สาม นั้นมีไว้สำหรับกรณีที่เกิดภาวะฉุกเฉิน เช่น กรณีที่เกิดอุบัติเหตุในแยกใดแยกหนึ่งขึ้น

จะเห็นได้ว่าระบบการควบคุมสัญญาณไฟจราจรบริเวณสี่แยกจำนวนสี่ชุด ที่จัดทำขึ้นน่าจะสามารถช่วยให้การจราจรเกิดความคล่องตัวได้มากขึ้น

บรรณานุกรม

1. รศ.สมยศ จุณณะปิยะ, การใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51, กรุงเทพ : โครงการตำราสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2537
2. ชัยวัฒน์ ลิ้มพรจิตรวิไล วรพจน์ กรแก้ววัฒนกุล, เรียนรู้และปฏิบัติการไมโคอนโทรลเลอร์ MCS-51 , กรุงเทพ : สำนักพิมพ์ Inex , 2545
3. ผศ. ธีรวัฒน์ ประกอบผล การพัฒนาไมโครคอนโทรลเลอร์ด้วยภาษาซี , กรุงเทพ : สำนักพิมพ์ Inex , 2546
4. พ.อ.เจนวิทย์ เหลืองอร่าม , การใช้ Turbo c++ เขียนโปรแกรมภาษาซี , กรุงเทพ : สำนักพิมพ์สุขภาพใจ , 2538
5. ธันวา ศรีประมง ,การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม , กรุงเทพ : โครงการตำรา มหาวิทยาลัยเทคโนโลยีมหานคร , 2539

กิตติกรรมประกาศ

ปริญญานิพนธ์ชิ้นนี้สำเร็จลุล่วงได้ด้วยดี ด้วยความช่วยเหลือจากบุคคลหลาย ๆ ท่าน ทั้งในด้านความรู้ ความช่วยเหลือ เครื่องมือและอุปกรณ์ต่าง ๆ ตลอดจนถึงการสั่งสอนให้คำแนะนำ อีกทั้งยังคอยเป็นกำลังใจในปริญญานิพนธ์นี้ ซึ่งต้องกล่าวถึง

บิดา มารดา ที่คอยเป็นกำลังใจและคอยสนับสนุนในเรื่องต่าง ๆ ในการทำปริญญานิพนธ์ชิ้นนี้

รศ.สมยศ จุณณะปิยะ อาจารย์ที่ปรึกษาในการทำปริญญานิพนธ์ ที่คอยให้คำแนะนำ ความช่วยเหลือ ให้คำปรึกษาในเรื่องต่าง ๆ และยังให้ความสนับสนุนในทุก ๆ ด้าน

พี่ และเพื่อนๆ ในห้องโปรเจค และ เพื่อนที่หอชัยพฤกษ์ ห้อง 302 ที่คอยให้คำแนะนำและคำปรึกษาต่าง ๆ ในเรื่องข้อมูลที่เป็นประโยชน์อย่างสูง แก่ปริญญานิพนธ์ชิ้นนี้

คณะผู้จัดทำ