การใช้งานรีโมทแบบอินฟราเรดทั่วไปมาควบคุมอุปกรณ์

USING GENERAL IR REMOTE CONTROLLER CONTROLLED EQUIPMENT

โดย

นางสาวาริสรา แท่นทอง

นางสาวอรุณี ตั้งเจริญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

USING GENERAL IR REMOTE CONTROLLER CONTROLLED EQUIPMENT

BY

MS. VARISSARA TANTONG

MS. ARUNEE TANGCHAROEN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE DEGREE OF BACHELOR IN CONTROL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชา วิศวกรรมระบบควบคุม

คณะ วิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การใช้งานรีโมทแบบอินฟราเรดทั่วไปมาควบคุมอุปกรณ์
USING GENERAL IR REMOTE CONTROLLER CONTROLLED EQUIPMENT

ผู้จัดทำ 1. นางสาววริสรา แท่นทอง
2. นางสาวอรุณี ตั้งเจริญ

.....อาจารย์ที่ปรึกษา
(อาจารย์เกียรติศักดิ์ คมวัชระ)

การใช้รีโมทแบบอินฟราเรดทั่วไปมาควบคุมอุปกรณ์

นางสาววริสรา แท่นทอง
นางสาวอรุณี ตั้งเจริญ

อาจารย์ที่ปรึกษา
อาจารย์เกียรติศักดิ์ คมวัชระ

ปีการศึกษา 2546

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอการใช้รีโมทแบบอินฟราเรดทั่วไปมาควบคุมอุปกรณ์ประกอบด้วย 5 ส่วน คือ ส่วนภาครับสัญญาณอินฟราเรด ส่วนประมวลผล ส่วนแสดงผลLCD ส่วนควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า และส่วนภาคส่งสัญญาณอินฟราเรด ทำงานโดยการรับสัญญาณอินฟราเรดจากรีโมทแบบอินฟราเรดทั่วไปเข้ามาที่ไมโครคอนโทรลเลอร์ตระกูล PIC ทำการประมวลผลได้ค่าคำสั่ง นำค่าคำสั่งนั้นส่งออกไปควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า และควบคุมอุปกรณ์อื่นๆ โดยมีการแสดงผลโหมดการทำงานของฮาร์ดแวร์ออกมาทาง LCD

ABSTRACT

This thesis represents using general IR remote controller controlled equipment. This project composed of 5 parts as IR receiver part, processing part, on-off control part, and IR transmitter part for receive IR signals from general IR remote controller, processing them, then result in patterns of command to on-off electrical equipment, controls other equipment, and displays working mode of hardware by LCD.

สารบัญ

สารบัญ (ต่อ)

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

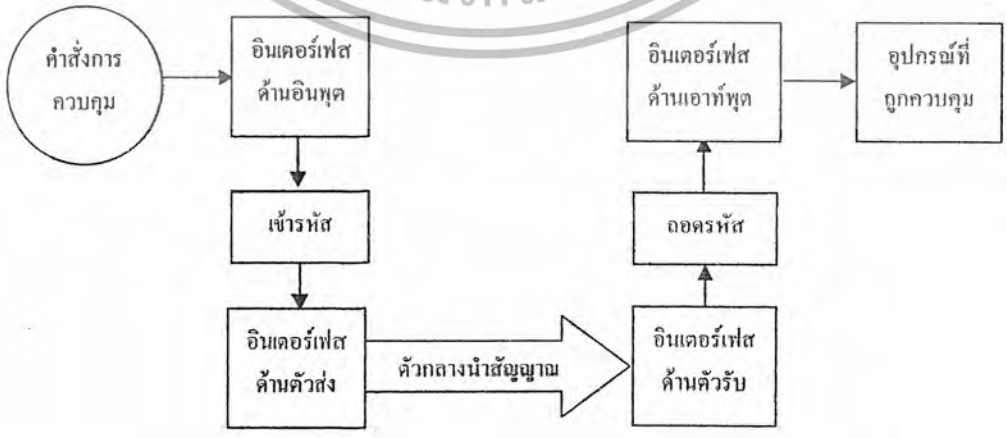
สารบัญตาราง

บทที่ 1

บทนำ

ปัจจุบันรีโมทแบบอินฟราเรดกลายเป็นอุปกรณ์ควบคุมพื้นฐานที่ใช้กับเครื่องใช้ไฟฟ้าภายในบ้านส่วนใหญ่ ไม่ว่าจะเป็นเครื่องรับโทรทัศน์ เครื่องเล่นวีดีโอเทป เครื่องเล่นวีดีโอซีดี อุปกรณ์เครื่องเสียงหรืออุปกรณ์ไฟฟ้าชนิดอื่นๆอีกมากมาย เพื่อความสะดวกสบายในการควบคุมอุปกรณ์ภายไฟฟ้าภายในบ้าน รีโมทภายในบ้านจึงมีหลายชนิดโดยส่งสัญญาณได้หลายรูปแบบ ทำให้เกิดโครงงานการใช้รีโมทแบบอินฟราเรดทั่วไปมาควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าภายในบ้าน โดยใช้รีโมทแบบอินฟราเรดส่งสัญญาณอินฟราเรดมาที่ตัวรับสัญญาณ แล้วนำสัญญาณที่ได้มาประมวลผลได้ชุดคำสั่งและรูปแบบสัญญาณที่เก็บไว้ในไมโครคอนโทรลเลอร์ แล้วส่งสัญญาณอินฟราเรดในแบบต่างๆเพื่อควบคุมอุปกรณ์ไฟฟ้าที่กำหนดไว้

โดยอาศัยหลักการทำงานของระบบควบคุมระยะไกลแบบทั่วไปในบล็อกไดอะแกรมรูปที่ 1.1 เริ่มจากตัวกำหนดคำสั่งที่ใช้สำหรับการควบคุมว่ามีคำสั่งอะไรบ้าง จำนวนชุดคำสั่งทั้งหมด เมื่อมีการรับรูปแบบคำสั่งแล้ว รูปแบบของคำสั่งที่ถูกเลือก จะถูกส่งไปยังภาคส่งสัญญาณทำหน้าที่แปลงสัญญาณ หรือรวมสัญญาณควบคุมให้มีรูปแบบที่เหมาะสมกับวงจร โดยอาจทำการเข้ารหัสสัญญาณให้แต่ละคำสั่งมีรหัสคำสั่งเฉพาะของตัวมันเองให้เป็นสัญญาณทางไฟฟ้า ก่อนที่จะถูกส่งออกไปยังภาครับโดยตัวอินเตอร์เฟสด้านตัวส่งเพื่อทำหน้าที่ส่งสัญญาณที่ภาครับต้องเข้าใจได้ สัญญาณที่ส่งออกมาในรูปของแสงอินฟราเรด สัญญาณที่เข้ามายังเครื่องรับหรือภาครับ จะถูกตัวอินเตอร์เฟสทำหน้าที่แปลงสัญญาณ ให้อยู่ในรูปของสัญญาณไฟฟ้าที่เข้ากับระบบของตัวรับ ก่อนถูกถอดรหัสเพื่อทราบวัตถุประสงค์ของคำสั่ง จากนั้นส่วนของอินเตอร์เฟสด้านเอาท์พุตจะทำหน้าที่ควบคุมการทำงานของอุปกรณ์ที่ต้องการ ตามลักษณะคำสั่งที่ได้รับ



รูปที่ 1.1 หลักการทำงานของระบบควบคุมระยะไกลแบบทั่วไป

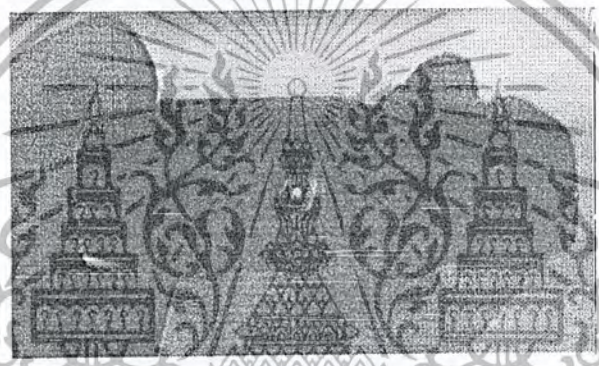
บทที่ 2

ทฤษฎีและหลักการ

2.1 ทฤษฎีรีโมทแบบอินฟราเรด

2.1.1 แสงอินฟราเรด

อินฟราเรด คือแสงปกติที่มีสีเฉพาะตัว ที่มนุษย์ไม่สามารถมองเห็นแสงนี้ได้เนื่องจากความยาวคลื่นของแสงอยู่ที่ 950 nm. ซึ่งน้อยกว่าแสงที่สามารถมองเห็นได้ เป็นเหตุผลที่เลือกใช้แสงอินฟราเรดในรีโมท เพราะเราต้องการใช้แต่ไม่ต้องการที่จะเห็นแสงอินฟราเรด และเนื่องจาก LED อินฟราเรดค่อนข้างผลิตได้ง่ายและมีราคาถูก

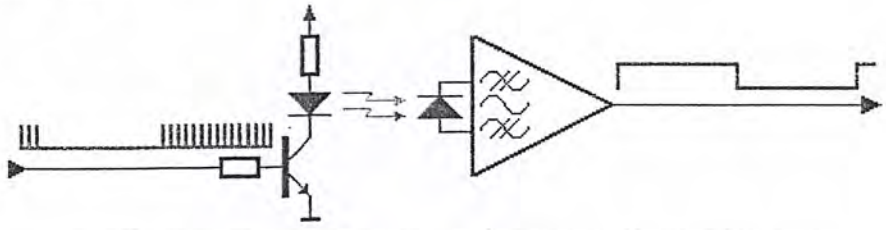


รูปที่ 2.1 แสงอินฟราเรด

แหล่งกำเนิดแสงอินฟราเรดมีมากมาย แสงอาทิตย์เป็นแหล่งกำเนิดที่สว่างที่สุด แต่ก็ยังมีแหล่งกำเนิดอีกหลายแหล่ง เช่นแสงจากหลอดไฟ แสงเทียน หรือแม้แต่ร่างกายเรา ตามหลักความจริงแล้วทุกสิ่งทุกอย่างที่แผ่ความร้อนได้ สามารถแผ่อินฟราเรดได้เช่นกัน เราต้องหาวิธีทำการส่งข้อความไปยังตัวรับอย่างไม่มีข้อผิดพลาด

2.1.2 การ Modulation

การ Modulation เป็นการทำให้สัญญาณแยกออกจากสัญญาณรบกวนได้ เนื่องด้วยการ modulation ทำให้แหล่งจ่ายแสงอินฟราเรดส่องออกมาในค่าความถี่เฉพาะ ตัวรับอินฟราเรดจึงถูกปรับไปที่ความถี่นั้น



รูปที่ 2.2 การ Modulation สัญญาณที่ส่งออกและสัญญาณที่รับได้

จากรูปด้านบนเห็นได้ว่าสัญญาณที่ถูก modulate กำลังขับ LED อินฟราเรดของตัวส่งสัญญาณทางด้านซ้ายมือ สัญญาณที่จับได้จะออกมาทางตัวรับทางด้านขวามือ

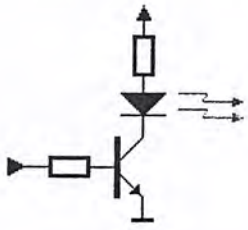
ในการสื่อสารแบบอนุกรมเรามักพูดถึงคำว่า 'mark' และ 'space' คำว่า 'space' คือสัญญาณที่ขาดหายไปซึ่งอยู่ในสภาวะ off ที่ตัวส่งสัญญาณ ไม่มีแสงฉายออกมาระหว่างสภาวะ 'space' ระหว่างสภาวะ 'mark' ของสัญญาณ แสงอินฟราเรดถูกส่งและหยุดส่งตามค่าความถี่เฉพาะ โดยทั่วไปอุปกรณ์ทางอิเล็กทรอนิคส์มักใช้ค่าความถี่ระหว่าง 30kHz และ 60kHz

ที่ด้านตัวรับ คำว่า 'space' แสดงถึงค่าสัญญาณ high ที่เอาท์พุตทางด้านตัวรับ คำว่า 'mark' จึงเป็นค่าสัญญาณ low ไปโดยอัตโนมัติ คำว่า 'mark' และ 'space' ไม่ได้มีแค่ค่า 0 และ 1 เท่านั้นที่เราต้องการจะส่ง ความสัมพันธ์ที่แท้จริงระหว่าง คำว่า 'mark' กับ 'space' และ 1 กับ 0 ขึ้นอยู่กับโปรโตคอลที่ใช้ ซึ่งจะกล่าวถึงในหัวข้อถัดไป

2.1.3 ตัวส่งสัญญาณ

ตัวส่งสัญญาณมักเป็นแบบพกพาที่ใช้พลังงานแบตเตอร์รี่ ตัวส่งควรใช้พลังงานให้น้อยที่สุดเท่าที่ทำได้ และ สัญญาณอินฟราเรดควรเข้มมากพอที่จะส่งสัญญาณควบคุมได้ในระยะทางที่ยอมรับได้ ชิพจำนวนมากถูกออกแบบมาเพื่อใช้กับตัวส่งสัญญาณอินฟราเรด ชิพตัวเก่าๆใช้ได้กับโปรโตคอลเดียวเท่านั้น แต่ในทุกวันนี้ไมโครคอนโทรลเลอร์กำลังต่ำถูกใช้ในตัวส่งอินฟราเรดเพื่อเหตุผลง่ายๆที่ว่าให้ใช้ได้ยืดหยุ่นมากขึ้น เมื่อไม่ได้กดปุ่มตัวส่งจะอยู่ในภาวะกำลังต่ำ(sleep mode) ซึ่งไม่กินกระแสมากนัก ตัวประมวลผลสั่งงานให้ส่งคำสั่งอินฟราเรดเมื่อกดปุ่มเท่านั้น

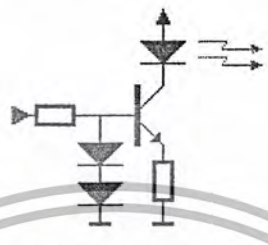
กระแสที่ไหลผ่านตัว LED มีค่าได้ตั้งแต่ 100mA ถึงมากกว่า 1A เพื่อที่จะหาระยะควบคุมที่ยอมรับได้ กระแสที่ LED จำเป็นต้องสูงเท่าที่จะทำได้ ตัวส่งสัญญาณควรสร้างตามค่าพารามิเตอร์ของ LED เช่น อายุการใช้งานของแบตเตอรี่และระยะการควบคุมมากสุด กระแส LED เป็นค่าสูงได้เพราะสัญญาณพัลซ์ที่ขับ LED สั้นมาก กำลังสูญเสียโดยเฉลี่ยของ LED ไม่ควรเกินค่าที่มากที่สุดที่ยอมให้ผ่านได้ กระแสสูงสุดจึงไม่ควรเกินค่าสูงสุดที่กำหนด ค่าพารามิเตอร์เหล่านี้ดูได้ตาม data sheet



รูปที่ 2.3 วงจรส่ง

วงจรทรานซิสเตอร์แบบง่ายๆที่ใช้ขับ LED เลือกทรานซิสเตอร์ที่เหมาะสม ค่าความต้านทาน สามารถคำนวณได้จากกฎของโอห์ม โดยค่าศักดาตกคร่อมLEDมีค่าประมาณ1.1V.

ตัวขับสัญญาณปกติที่ได้อธิบายไว้ข้างต้นมีข้อเสีย คือ เมื่อค่าศักดาของแบตเตอรี่ตกลง กระแสไหลผ่าน LED จะลดลงด้วย ส่งผลให้ระยะควบคุมสั้นลง



รูปที่ 2.4 วงจรส่งแบบ emitter follower

วงจร emitter follower สามารถหลีกเลี่ยงเหตุการณ์นี้ได้ นำไดโอด 2 ตัวมาต่ออนุกรมกันจะจำกัดสัญญาณพัลซ์ที่ขาเบสของทรานซิสเตอร์ไว้ที่ 1.2V ค่าศักดาตกคร่อมขาเบสและอิมิตเตอร์ของทรานซิสเตอร์ลบออก 0.6V จากค่านั้น ผลที่ได้จะมีค่าแอมปลิจูดคงที่ขนาด 0.6 V ที่ขาอิมิตเตอร์ ค่าแอมปลิจูดคงที่ที่ตกคร่อมตัวต้านทานคงที่ให้ผลเป็นสัญญาณกระแสที่เป็นพัลซ์มีขนาดคงที่ การคำนวณค่ากระแสที่ผ่านก็ใช้กฎของโอห์มอีกครั้ง

2.1.4 ตัวรับสัญญาณ

วงจรตัวรับสัญญาณมากมายที่มีอยู่ในท้องตลาด หลักการเลือกที่สำคัญที่สุดคือ ความถี่ที่ใช้ในการ modulation

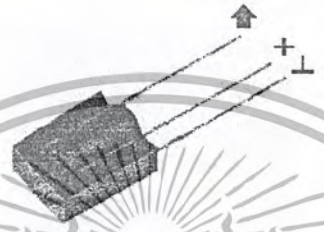
รูปที่ 2.5 ตัวอย่างบล็อกไดอาแกรมของตัวรับสัญญาณอินฟราเรด

ตัวรับสัญญาณอินฟราเรดจับสัญญาณโดยใช้ไดโอดอินฟราเรดจับสัญญาณที่อยู่ทางด้านซ้ายมือของบล็อกไดอะแกรม สัญญาณนี้ถูกขยายและจำกัดโดย 2 ตอนแรก ตัว limitter แสดงตัวเป็นวงจรAGC เพื่อหาระดับสัญญาณพัลซ์คงที่โดยไม่คำนึงถึงระยะระหว่างอุปกรณ์

ดังรูปสัญญาณไฟฟ้ากระแสสลับเท่านั้นที่ส่งไปที่ Band Pass Filter ตัว Band Pass Filter ถูกปรับไปที่ความถี่ modulation ของอุปกรณ์ส่ง ผู้ใช้อุปกรณ์ทางอิเล็คทรอนิคส์ใช้ช่วงความถี่ปกติมีค่าตั้งแต่ 30kHz ถึง 60kHz

ตอนต่อไปเป็นตัวจับสัญญาณ (integrator และ comparator) จุดมุ่งหมายหลักของ 3 บล็อกนี้คือ เพื่อจับความถี่การ modulation แล้วทำการ demodulation เอาส่วนที่เป็นความถี่พาหะออกและให้เอาท์พุตเฉพาะส่วนที่เป็นข้อมูลท่านั้น

ดังที่กล่าวไปแล้ว บล็อกเหล่านี้ถูกรวมอยู่ในอุปกรณ์ทางอิเล็คทรอนิคส์ตัวเดียว มีผู้ผลิตหลายบริษัทที่ผลิตอุปกรณ์เหล่านี้ออกสู่ท้องตลาด และอุปกรณ์ส่วนใหญ่ที่หาได้มีหลายแบบและได้ถูกปรับให้มีความถี่ของการ modulation เฉพาะตัว



รูปที่ 2.6 ตัวรับสัญญาณ

ที่ผ่านมาเป็นการอธิบายถึงทฤษฎีพื้นฐานในการใช้รีโมทคอนโทรลอินฟราเรด แต่ไม่ได้กล่าวถึง โปรโตคอลที่ใช้ในการสื่อสารระหว่างตัวส่งสัญญาณและตัวรับสัญญาณ โปรโตคอลโดยมากถูกออกแบบโดยหลายบริษัทผู้ผลิตสินค้า ในที่นี้ยกตัวอย่างโปรโตคอลบางตัว คือ

- Sharp Protocol
- Sony SIRC Protocol
- Phillips RC5 Protocol
- Nokia KRC17 Protocol

2.1.5 รูปแบบการส่งสัญญาณ(protocol)

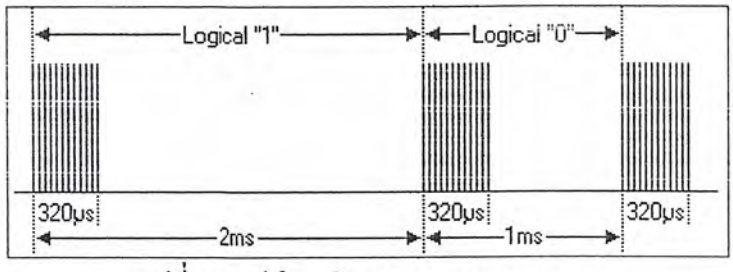
2.1.5.1 Sharp Protocol

ใช้ในเครื่องใช้จำพวก VCR ที่ผลิตโดยบริษัท Sharp

Features (ลักษณะเด่น)

- ความยาว 8 bit command, 5 bit address
- Pulse distance modulation
- ค่าความถี่พาหะ 38 kHz
- ข้อมูล1 บิตใช้เวลา 1 ms or 2 ms

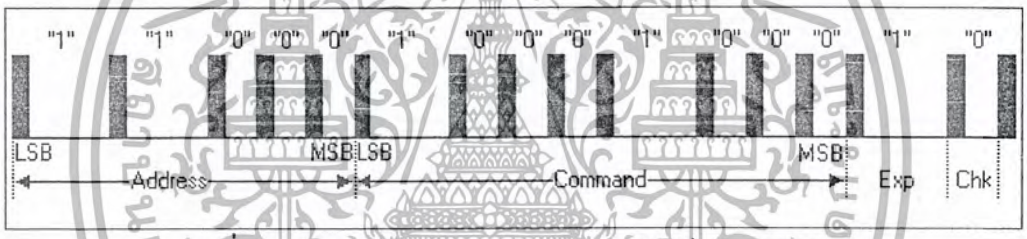
Modulation



รูปที่ 2.7 รูปค่าลอจิกของ Sharp Protocol

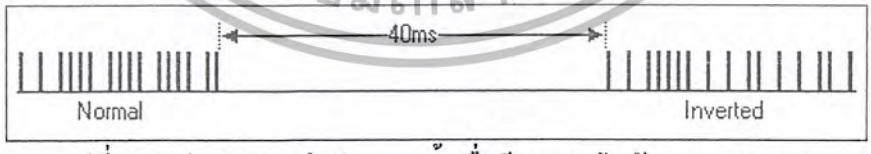
โปรโตคอล Sharpใช้ การเข้ารหัส pulse distance ของบิตข้อมูล ในแต่ละพัลซ์กว้าง 320 µs มีค่าความถี่ของพาหะนำสัญญาณอินฟราเรด 38 kHz (ประมาณ 12 cycle) ค่าลอจิก "1" ใช้เวลา 2 msในขณะที่ค่าลอจิก "0" ใช้เวลาเพียง 1 ms เท่านั้น ค่า duty-cycle* ของพาหะที่แนะนำอยู่ที่ประมาณ 1/4 หรือ 1/3

Protocol



รูปที่ 2.8 รูปแสดงตัวอย่างขบวนสัญญาณพัลซ์ของ Sharp Protocol

ภาพนี้แสดงตัวอย่างขบวนสัญญาณพัลซ์ที่ส่งคำสั่ง 00010001B(ขนาด 8 บิต) และ address 00011B(ขนาด 5 บิต) โดย address ถูกส่งออกไปเป็นอันดับแรกตามด้วยคำสั่งและทั้ง 2 อย่างจะส่งบิต LSB ออกไปก่อน (ส่งแบบโดยตรง)



รูปที่ 2.9 รูปแสดงการส่งสัญญาณซ้ำเมื่อมีการกดปุ่มค้างของ Sharp Protocol

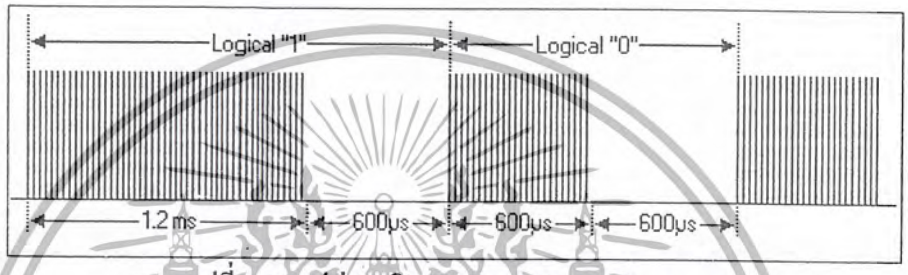
คำสั่งที่สมบูรณ์ตอนหนึ่งๆประกอบด้วย 2 ข้อความ การส่งแบบโดยตรงได้อธิบายไว้แล้วข้างต้น การส่งครั้งที่ 2 (ตามหลังครั้งแรกห่างกัน 40 ms) ข้อมูลเหมือนเดิม แต่แตกต่างกันที่ทุกบิตที่รับมาใน address ถูกกลับค่า (0⟶1,1⟶0) ดังนั้นตัวรับจึงสามารถตรวจสอบค่าได้ว่าข้อความที่รับมาเชื่อถือได้หรือไม่

2.1.5.2 Sony SIRC Protocol

Features (ลักษะเด่น)

- address ยาว 5 บิต และ คำสั่งยาว 7 บิต (โปรโตคอล 12 บิต)
- Pulse width modulation
- ความถี่พาหะ 40 kHz
- ข้อมูล 1 บิตใช้เวลา 1.2 ms or 0.6 ms

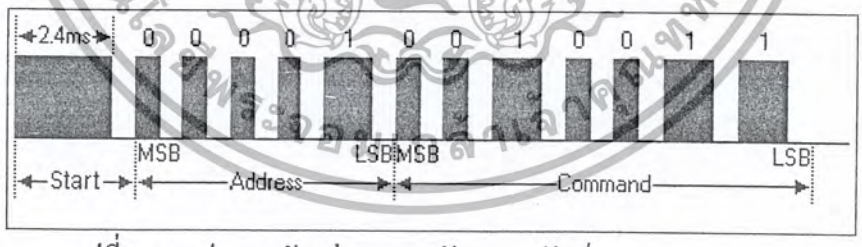
Modulation



รูปที่ 2.10 รูปค่าลอจิกของ Sony SIRC Protocol

โปรโตคอล SIRC ใช้การเข้ารหัส pulse width กับบิตใดๆในข้อมูล สัญญาณพัลซ์ที่แสดง ค่าลอจิก "1" พาหะนำสัญญาณอินฟราเรดกว้าง 1.2ms 40kHz ในขณะที่ค่าลอจิก "0" กว้าง 0.6ms แต่ละช่วงของการส่งสัญญาณเว้น 0.6ms ค่า duty-cycle ของพาหะนำสัญญาณอินฟราเรดที่แนะนำอยู่ที่ 1/4 หรือ 1/3.

Protocol



รูปที่ 2.11 รูปแสดงตัวอย่างขบวนสัญญาณพัลซ์ของ Sony SIRC Protocol

ภาพด้านบนแสดงตัวอย่างขบวนสัญญาณพัลซ์ของโปรโตคอล SIRC โปรโตคอลนี้จะส่งMSB ออกไปอันดับแรก บิตเริ่มต้นมักจะกว้าง 2.4 ms ตามด้วยช่องว่างขนาดมาตรฐานกว้าง 0.6 ms เป็นการแยกส่วนเริ่มออกจากสัญญาณข้อมูล หลังจากนั้นจึงส่ง address 5 บิต ตามด้วยคำสั่ง 7 บิต ในกรณีนี้ address 00001B และ คำสั่ง 00010011B โดยคำสั่งส่งทุกๆ 45 ms (วัดจากจุดเริ่มถึงจุดเริ่ม) เมื่อกดคีย์ของรีโมทค้างไว้

ตัวอย่างค่าคำสั่ง

ตารางข้างล่างแสดงข้อความบางอย่างที่รีโมทคอนโทรลของSony ส่งออกมาในโปรโตคอลแบบ 12 บิต

Address	Device
1	TV
2	VCR 1
3	VCR 2
6	Laser Disc Unit
12	Surround Sound
16	Cassette deck / Tuner
17	CD Player
18	Equaliser

Command	Command
0 - 9	Digit keys 0 - 9
16	Channel +
17	Channel -
18	Volume +
19	Volume -
20	Mute
21	Power
22	Reset
23	Audio Mode
24	Contrast +
25	Contrast -
26	Colour +
27	Colour -
30	Brightness +
31	Brightness -
38	Balance Left
39	Balance Right
47	Standby

ตารางที่ 2.1 ค่าสัญญาณ ADDRESS และ COMMAND ของ Sony SIRC Protocol

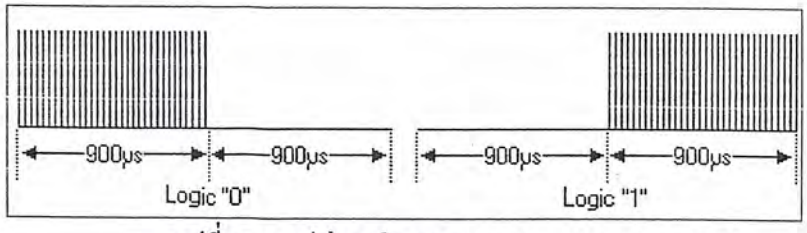
2.1.5.3 Philips RC-5 Protocol

รหัส RC5 จาก บริษัท Philips ค่อนข้างเป็นโปรโตคอลที่ใช้กันอย่างแพร่หลายในหมู่นักประดิษฐ์ อาจเป็นเพราะเป็นรีโมทคอนโทรลที่หาได้ง่ายและราคาถูก โปรโตคอลนี้กำหนดให้ทำงานสำหรับอุปกรณ์ต่างชนิดกัน ทำให้แน่ใจได้ว่าจะเข้าได้ระบบเพื่อความบันเทิงของคุณได้ทั้งหมด

Features

- address ขนาด 5 บิต และ คำสั่งขนาด 6 bit
- Bi-phase coding (Aka Manchester coding)
- ความถี่พาหะ 38 kHz
- ข้อมูล 1 บิตใช้เวลา 1.8 ms

Modulation

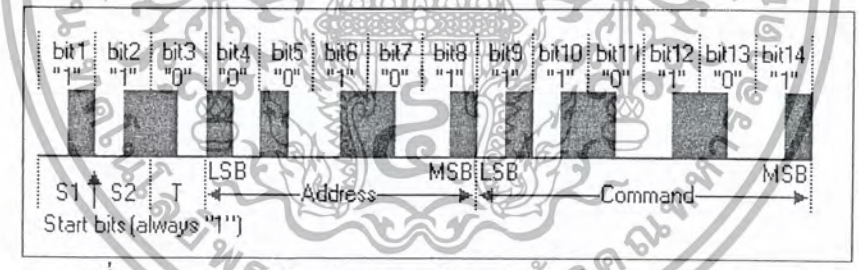


รูปที่ 2.12 รูปค่าลอจิกของ Phillips RC-5 Protocol

โปรโตคอลนี้ใช้ bi-phase modulation (หรือที่เรียกว่า Manchester coding) ความถี่พาหะอินฟราเรด 38 kHz ทุกบิตกว้างเท่ากันหมด คือ 1.8 ms ในโปรโตคอลนี้ ครึ่งหนึ่งของเวลาใน 1 บิตประกอบด้วยการส่งสัญญาณโดยความถี่พาหะ 38 kHz และ อีกครึ่งหนึ่งปล่อยไว้เฉยๆ ค่าลอจิก "0" แสดงการส่งข้อมูลในช่วงเวลาของครึ่งบิตแรก ค่าลอจิก "1" แสดงการส่งสัญญาณในช่วงเวลาครึ่งบิตที่สอง โดยอัตราของ pulse/pause ratio เท่ากับ 38kHz ความถี่พาหะ 1/3 หรือ 1/4 (เพื่อลดกำลัง)

Protocol

ดังรูปสัญญาณที่แสดงตัวอย่างขบวนสัญญาณพัลซ์ ของสัญญาณ RC5 โดยตัวอย่างนี้ส่งคำสั่ง 00001B ไปที่ address 10010B



รูปที่ 2.13 รูปแสดงตัวอย่างขบวนสัญญาณพัลซ์ของ Phillips RC-5 Protocol

พัลซ์ 2 ลูกแรกเป็นพัลซ์เริ่มต้น และ มีค่าลอจิก "1" ทั้งคู่ ครึ่งช่วงเวลาบิตที่สองปล่อยว่างไว้ ก่อนที่ตัวรับจะรับรู้ว่ามีเริ่มข้อความส่งมา การขยายการใช้งานของ RC5 ใช้เฉพาะ บิตเริ่ม 1 บิต ,บิต S2 จะถูกแปลงคำสั่งให้อยู่ในรูป คำสั่ง 6 บิต ทำให้บิตคำสั่งมีทั้งหมด 7 บิต บิตที่ 3 คือ บิตทอกเกิล บิตนี้จะกลับค่าทุกครั้งที่คีย์ถูกกด วิธีนี้จะทำให้ตัวรับรับรู้ว่าคีย์ยังกดอยู่หรือ กดซ้ำ บิตถัดไป 5 บิต แสดง address ของอุปกรณ์อินฟราเรดซึ่งส่ง LSB ก่อน จึงตามด้วยคำสั่ง 6 บิต ข้อความประกอบด้วย 14 บิต ซึ่งอาจกินเวลาถึง 25.2 ms บางครั้งข้อความอาจสั้นลงเพราะครึ่งช่วงแรกของเวลา 1 บิตในบิตเริ่ม S1 ยังคงปล่อยว่างอยู่ และถ้าบิตสุดท้ายของข้อความเป็นลอจิก "0" ครึ่งช่วงสุดท้ายของเวลา 1 บิตในบิต

สุดท้ายจะถูกปล่อยด้วย ตราบใดที่คีย์ถูกกดค้าง ข้อความจะซ้ำในเวลา 114 ms. บิตทอกเกิล จะยังคงเหมือนเดิมในช่วงนี้ ขึ้นอยู่กับ software ของตัวรับว่าจะรับรู้ว่ามีการส่งลักษณะซ้ำๆแบบนี้ได้หรือไม่

Pre-defined Commands

RC5 Address	Device
$00 - 0	TV1
$01 - 1	TV2
$02 - 2	Teletext
$03 - 3	Video
$04 - 4	LV1
$05 - 5	VCR1
$06 - 6	VCR2
$07 - 7	Experimental
$08 - 8	Sat1
$09 - 9	Camera
$0A - 10	Sat2
$0B - 11	
$0C - 12	CDV
$0D - 13	Camcorder
$0E - 14	
$0F - 15	
$10 - 16	Pre-amp
$11 - 17	Tuner
$12 - 18	Recorder1
$13 - 19	Pre-amp
$14 - 20	CD Player
$15 - 21	Phono
$16 - 22	Sat4
$17 - 23	Recorder2
$18 - 24	
$19 - 25	
$1A - 26	CDR
$1B - 27	
$1C - 28	
$1D - 29	Lighting
$1E - 30	Lighting
$1F - 31	Phone

RC5 Command	TV Command	VCR Command
$00 - 0	1	1
$01 - 1	2	2
$02 - 2	2	2
$03 - 3	3	3
$04 - 4	4	4
$05 - 5	5	5
$06 - 6	6	6
$07 - 7	7	7
$08 - 8	8	8
$09 - 9	9	9
$0C - 12	Standby	Standby
$10 - 16	Volume +	
$11 - 17	Volume -	
$12 - 18	Brightness +	
$13 - 19	Brightness -	
$32 - 50		Fast Rewind
$34 - 52		Fast Forward
$35 - 53		Play
$36 - 54		Stop
$37 - 55		Recording

ตารางที่ 2.2 ค่าสัญญาณ ADDRESS และ COMMAND ของ Phillips RC-5 Protocol

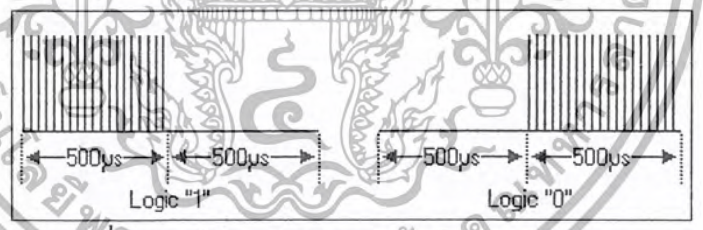
2.1.5.4 Nokia NRC17 Protocol

รีโมทคอนโทรลที่ใช้โปรโตคอล Nokia ใช้การส่งคำสั่งทางสัญญาณอินฟราเรด 17 บิต โปรโตคอลนี้ถูกออกแบบมาเพื่อผู้ใช้สินค้าอิเล็คทรอนิคส์ของ Nokia ถูกใช้เมื่อไม่กี่ปีที่ผ่านมาซึ่งทางบริษัท Nokia ได้ผลิตชุด TV และ VCR เช่นเดียวกันกับยี่ห้อ Finlux และ Salora ก็ใช้โปรโตคอลนี้เช่นกัน ในทุกวันนี้โปรโตคอลนี้ใช้มากในตัวรับดาวเทียมของ Nokia

Features

- คำสั่งขนาด 8 บิต, address ขนาด 4 บิตและ subcode ขนาด 4 บิต
- Bi-phase coding
- ความถี่พาหะ 38 kHz
- ข้อมูล 1 บิตใช้เวลา 1 ms
- มีตัวบอกว่าแบตเตอรี่จะหมด
- บริษัทผู้ผลิต : Nokia CE

Modulation

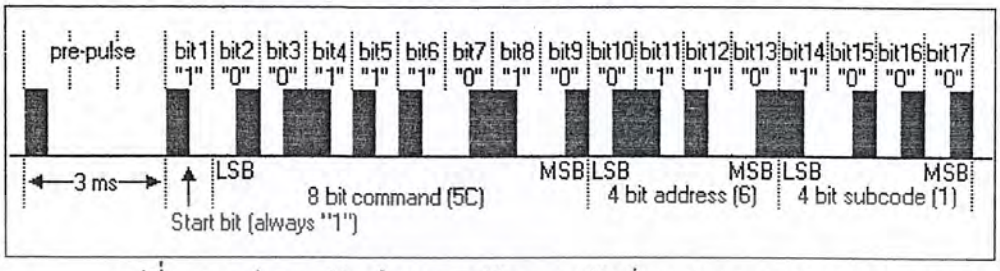


รูปที่ 2.14 รูปค่าลอจิกของ Nokia NRC-17 Protocol

โปรโตคอลนี้ใช้ bi-phase (หรือที่เรียกว่า NRZ - Non Return to Zero) modulation ของพาหะของสัญญาณอินฟราเรดมีความถี่ 38 kHz โปรโตคอลนี้ทุกบิตยาว 1 ms โดยช่วงครึ่งหนึ่งของเวลาที่ใช้ใน 1 บิตประกอบด้วยพาหะความถี่ 38 kHz และอีกครึ่งหนึ่งปล่อยวางไว้ ค่าลอจิก “1” แสดงถึงการส่งสัญญาณในช่วงครึ่งเวลาแรกของเวลาใน 1 บิต ค่าลอจิก “0” แสดงถึงการส่งสัญญาณในช่วงครึ่งที่ 2 ของเวลาใน 1 บิต ค่าอัตรา pulse/pause ของความถี่พาหะ 38 kHz คือ 1/4 (เพื่อเป็นการลดการใช้กำลังงาน)

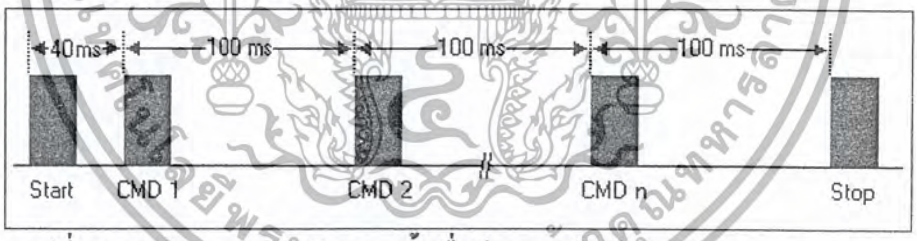
Protocol

ภาพข้างใต้แสดงตัวอย่างของขบวนสัญญาณพัลซ์ของข้อความในโปรโตคอล NRC17 ตัวอย่างนี้ส่งข้อมูล 01011100B ไป address 0110B subcode 0001B



รูปที่ 2.15 รูปแสดงตัวอย่างขบวนสัญญาณพัลซ์ของ Nokia NRC-17 Protocol

พัลซ์ลูกแรกเรียกว่า pre-pulse สร้างจากการส่งสัญญาณกว้าง 500 µs ตามด้วยการหยุดส่งอีก 2.5 ms ใช้เวลาทั้งหมดเท่ากับเวลาในข้อมูล 3 บิต หลังจากนั้นบิตเริ่มต้นจะถูกส่งออกไปซึ่งมักจะเป็นลอจิก "1" ซึ่งสัญญาณพัลซ์นี้สามารถใช้เป็นตัวเทียบวัดเวลาใน 1 บิตทางด้านตัวรับได้ เพราะเวลาส่งสัญญาณจริงๆใช้แค่ครึ่งหนึ่งของเวลาใน 1 บิต 8บิตถัดไปแสดงคำสั่งของสัญญาณอินฟราเรดซึ่งส่ง LSB ก่อน ตามด้วย address ของอุปกรณ์อีก 4 บิตสุดท้ายตามด้วย subcode 4 บิต ซึ่งจะเห็นได้จากส่วนที่ต่อจาก บิต address ข้อความประกอบด้วย pre-pulse ขนาด 3 ms และ 17 บิตในแต่ละ 1 ms (จึงใช้เวลาทั้งหมด 20 ms ต่อการส่ง 1 ข้อความ)



รูปที่ 2.16 รูปแสดงการส่งสัญญาณซ้ำเมื่อมีการกดปุ่มค้างของ Nokia NRC-17 Protocol

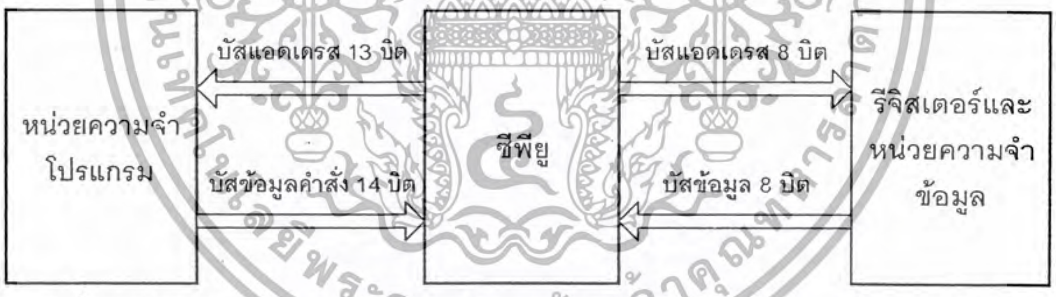
ทุกครั้งที่คีย์ของรีโมทคอนโทรลถูกกด บิตเริ่มที่ถูกส่งออกไปประกอบด้วย คำสั่ง 11111110B และ address/subcode 11111111B ข้อความจริงๆถูกส่งใน 40 ms ต่อมา และส่งซ้ำอีกในทุกๆ100 ms ถ้าคีย์ของรีโมทคอนโทรลยังคงถูกกดอยู่ เมื่อปล่อยคีย์ ข้อความหยุดถูกปล่อยออกมาเป็นการจบชุดข้อความ ข้อความหยุดยังคงใช้ คำสั่ง 11111110B และ address/subcode 11111111B ชุดคำสั่งทุกชุดสามารถทำให้เป็นชุดคำสั่งเดียวได้ที่ท้ายตัวรับ เนื่องจากข้อความเริ่มและข้อความหยุด ทำให้คีย์ที่กดโดยบังเอิญจะไม่มีผลอะไร ตัวรับอาจจะตัดสินใจที่จะเชื่อฟังข้อความที่ซ้ำมาหรือไม่ก็ได้ ยกตัวอย่างเช่น การเลื่อนเคอร์เซอร์อาจทำซ้ำๆเท่าที่คีย์ถูกกด ค่าอินพุตที่เป็นตัวเลขจะดีกว่าตรงที่ไม่รับการทำซ้ำเรื่อยๆ

2.2 ทฤษฎีไมโครคอนโทรลเลอร์

2.2.1 โครงสร้างทางฮาร์ดแวร์ของไมโครคอนโทรลเลอร์ตระกูล PIC 16F877

ไมโครคอนโทรลเลอร์ตระกูล PIC มีสถาปัตยกรรมแบบฮาร์วาร์ด (Harvard architecture) กล่าวคือ มีการแยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลออกจากกัน โดยมีบัสสำหรับติดต่อแยกกันด้วย ดังแสดงในรูป 2.17 จะเห็นว่าซีพียูภายในไมโครคอนโทรลเลอร์จะติดต่อกับหน่วยความจำโปรแกรมด้วยบัสแอดเดรส 13 บิต และบัสข้อมูลหน่วยความจำโปรแกรม 14 บิต ในขณะที่บัสสำหรับติดต่อกับหน่วยความจำข้อมูลและรีจิสเตอร์ภายในเป็นแบบ 8 บิตทั้งบัสแอดเดรสและบัสข้อมูล

นอกจากการจัดสถาปัตยกรรมแบบนี้แล้ว การกระทำคำสั่งของไมโครคอนโทรลเลอร์ PIC ยังใช้กระบวนการที่เรียกว่า ไปป์ไลน์ (Pipeline) ทำให้สามารถเฟตช์คำสั่งถัดไป ในขณะที่กำลังเอ็คซิคิวต์คำสั่งในปัจจุบัน ส่งผลให้ความเร็วในการทำงานของไมโครคอนโทรลเลอร์เพิ่มมากขึ้น นั่นจึงเป็นที่มาของความสามารถในการทำคำสั่ง 1 คำสั่งในสัญญาณนาฬิกา 1 ลูก (เฟตช์ : fetch เป็นกระบวนการเรียกคำสั่งออกจากหน่วยความจำโปรแกรมแล้วแปลเป็นเลขฐานสิบหกเพื่อให้ซีพียูเข้าใจ ส่วนกระบวนการเอ็คซิคิวต์ (execute) เป็นการกระทำคำสั่งให้เกิดผลลัพธ์ตามที่คำสั่งนั้นๆ กำหนด)

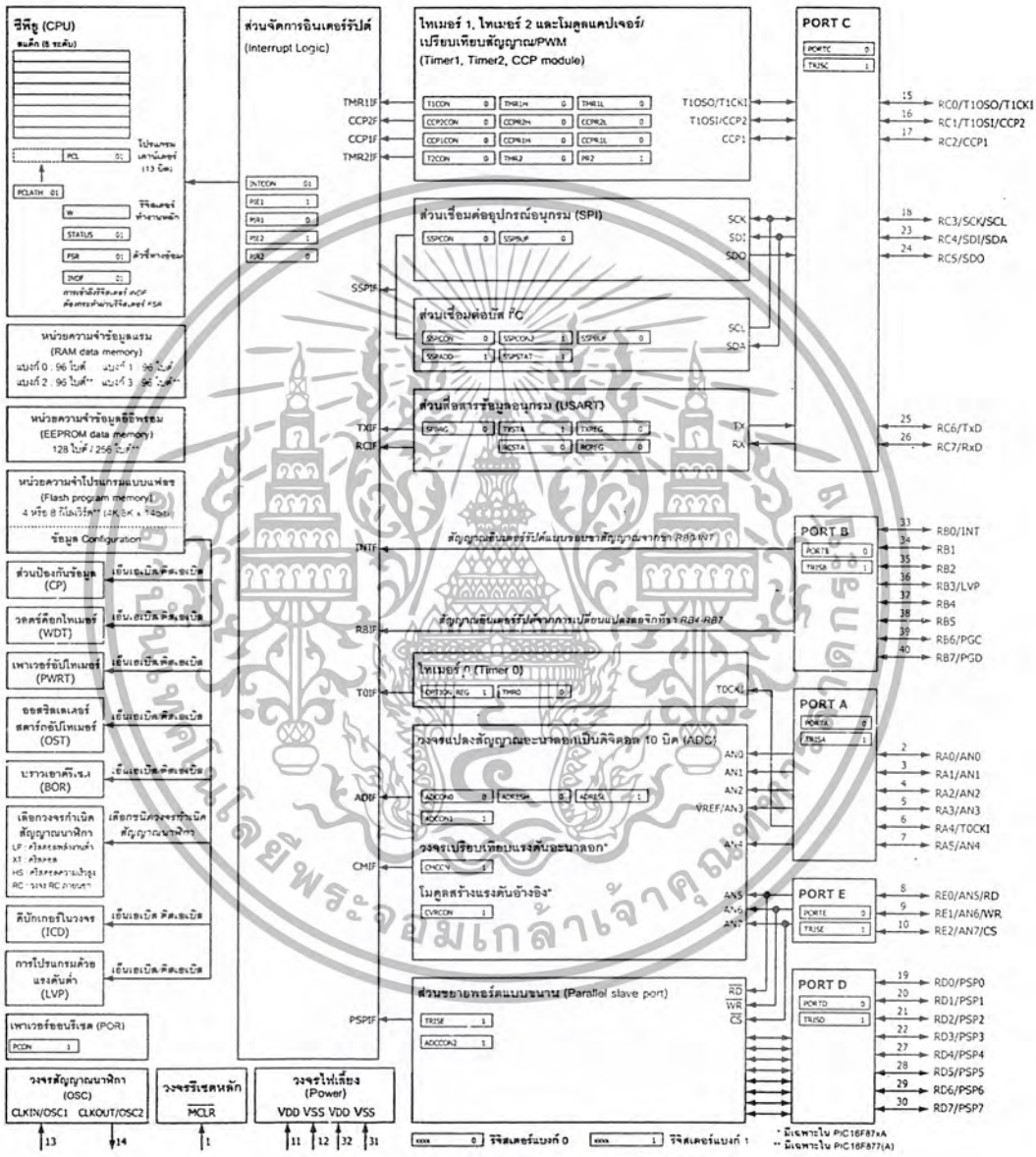


รูปที่ 2.17 ไดอะแกรมแสดงรูปแบบสถาปัตยกรรมของไมโครคอนโทรลเลอร์แบบฮาร์วาร์ด

Configuration word : ข้อมูลหลักสำหรับกำหนดการทำงานของไมโครคอนโทรลเลอร์ PIC

ไมโครคอนโทรลเลอร์ PIC16F87x มีรีจิสเตอร์พิเศษตัวหนึ่งที่บรรจุข้อมูลสำหรับกำหนดการทำงานทั้งหมดเอาไว้ คือ Configuration word โดยภายใน Configuration word จะบรรจุข้อมูลของการเลือกป้องกันการอ่านข้อมูล , เลือกความสามารถการรีเซตอัตโนมัติเมื่อไฟเลี้ยงลดต่ำถึงค่าที่กำหนด , ควบคุมการทำงานของวอตช์ด็อกไทเมอร์ หรือกระทั่งการเลือกชนิดของวงจรกำเนิดสัญญาณนาฬิกาของไมโครคอนโทรลเลอร์ การกำหนดข้อมูลสำหรับรีจิสเตอร์ตัวนี้สามารถกระทำได้ 2 ทางคือ ด้วยคำส่ง _CONIG ในส่วนต้นของโปรแกรมภาษาแอสเซมบลี แล้วแอสเซมเบลอร์ด้วย MPASM ซึ่งบรรจุอยู่ในชุดของโปรแกรม MPLAB อันเป็นsoftwareในการพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ PIC ทางที่ 2 คือ กำหนด software ที่ใช้ในการโปรแกรม

หน่วยความจำของไมโครคอนโทรลเลอร์ การกำหนดสามารถกระทำในทางใดทางหนึ่งหรือทั้งสองทางก็ได้ แต่ถ้าการกำหนดทั้ง 2 ทางแตกต่างกัน การกำหนดที่ software ของ เครื่องโปรแกรม จะมีนัยสำคัญที่สูงกว่า



รูปที่ 2.18 โครงสร้างการทำงานของไมโครคอนโทรลเลอร์ PIC 16F877 รุ่น 40 ขา

ชื่อขา	ตำแหน่งขา	ชนิดของขา	ชนิดของวงจรบัฟเฟอร์	รายละเอียดการทำงาน
OSC1/CLKIN	9 (13)	อินพุต	ชมิตต์ทริกเกอร์/ซีมอส (3)(4)	ขาต่อคริสตอล/รับสัญญาณนาฬิกาจากภายนอก
OSC2/CLKOUT	10 (14)	เอาต์พุต	-	ขาต่อคริสตอล/ในโหมด RC เป็นขาเอาต์พุต สัญญาณนาฬิกาความถี่ 1/4 ของสัญญาณที่ขา OSC1
$\overline{MCLR}$/Vpp	1	อินพุต	ชมิตต์ทริกเกอร์	- ขารับสัญญาณรีเซตหลัก (Master Clear Input) ทำงานที่ลอจิก "0" - ขารับแรงดันโปรแกรม (programming voltage)
ขาพอร์ต A เป็นขาพอร์ต 2 ทิศทาง				
RA0/AN0	2	อินพุต/เอาต์พุต	ทีทีแอล/อะนาลอก	- ขาพอร์ต RA0 - อินพุตวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ช่อง 0
RA1/AN1	3	อินพุต/เอาต์พุต	ทีทีแอล/อะนาลอก	- ขาพอร์ต RA1 - อินพุตวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ช่อง 1
RA2/AN2/V_{REF-}/ CV_{REF}*	4	อินพุต/เอาต์พุต	ทีทีแอล/อะนาลอก	- ขาพอร์ต RA2 - อินพุตวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ช่อง 2 - อินพุตแรงดันอ้างอิงลบของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล - เอาต์พุตแรงดันอ้างอิงของโมดูลแรงดันอ้างอิง (PIC16F87xA)
RA3/AN3/V_{REF+}	5	อินพุต/เอาต์พุต	ทีทีแอล/อะนาลอก	- ขาพอร์ต RA3 - อินพุตวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ช่อง 3 - อินพุตแรงดันอ้างอิงบวกของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล
RA4/T0CKI/ C1OUT*	6	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RA4 กรณีใช้พอร์ตเอาต์พุตมีโครงสร้างแบบเดรนเปิด - อินพุตสัญญาณนาฬิกาของไทเมอร์ 0 - เอาต์พุตวงจรเปรียบเทียบแรงดันอะนาลอกช่อง 1 (PIC16F87xA)
RA5/AN4/$\overline{SS}$/ C2OUT*	7	อินพุต/เอาต์พุต	ทีทีแอล/อะนาลอก	- ขาพอร์ต RA5 - อินพุตวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ช่อง 4 - ขาสัญญาณ Slave Select ใช้ในการสื่อสารข้อมูลอนุกรมแบบซิงโครนัส - เอาต์พุตวงจรเปรียบเทียบแรงดันอะนาลอกช่อง 2 (PIC16F87xA)
ขาพอร์ต B เป็นขาพอร์ต 2 ทิศทาง สามารถกำหนดให้ต่อตัวต้านทานพูลอัปภายในเมื่อทำงานเป็นอินพุตได้ทางซอฟต์แวร์				
RB0/INT	21 (33)	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์ (1)	- ขาพอร์ต RB0 - อินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอก
RB1	22 (34)	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB1
RB2	23 (35)	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB2
RB3/LVP	24 (36)	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB3 - อินพุตรับแรงดันโปรแกรมต่ำ (+5V) ถ้าเอ็นเอเบิลไว้
RB4	25 (37)	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB4 และสามารถเกิดอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงลอจิกขึ้นที่ขานี้ หากเอ็นเอเบิลไว้
RB5	26 (38)	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB5 และสามารถเกิดอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงลอจิกขึ้นที่ขานี้ หากเอ็นเอเบิลไว้
RB6/PGC	27 (39)	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์ (2)	- ขาพอร์ต RB6 - เป็นขาสัญญาณนาฬิกาของการดีบักในวงจร (ICD) - สามารถเกิดอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงลอจิกขึ้นที่ขานี้ หากเอ็นเอเบิลไว้
RB7/PGD	28 (40)	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์ (2)	- ขาพอร์ต RB7 - เป็นขาสัญญาณข้อมูลของการดีบักในวงจร (ICD) - สามารถเกิดอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงลอจิกขึ้นที่ขานี้ หากเอ็นเอเบิลไว้

ตารางที่ 2.3 ตารางสรุปการทำงานของขาพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ PIC 16F877 (มีต่อ)

ชื่อขา	ตำแหน่งขา	ชนิดของขา	ชนิดของวงจรบัฟเฟอร์	รายละเอียดการทำงาน
ขาพอร์ต C เป็นขาพอร์ต 2 ทิศทาง				
RC0/T1OSO/ T1CKI	11 (15)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RC0 - เอาต์พุตวงจรออสซิลเลเตอร์ของไทเมอร์ 1 - อินพุตสัญญาณนาฬิกาของไทเมอร์ 1
RC1/T1OSI/ CCP2	12 (16)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RC1 - อินพุตวงจรออสซิลเลเตอร์ของไทเมอร์ 1 - อินพุตวงจรแคปเจอร์/เอาต์พุตวงจรเปรียบเทียบ/ เอาต์พุต PWM สำหรับโมดูล CCP2
RC2/CCP1	13 (17)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RC2 - อินพุตวงจรแคปเจอร์/เอาต์พุตวงจรเปรียบเทียบ/ เอาต์พุต PWM สำหรับโมดูล CCP1
RC3/SCK/SCL	14 (18)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RC3 - ขาสัญญาณนาฬิกาของวงจร SPI และระบบบัส I^2C
RC4/SDI/SDA	15 (23)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RC4 - ขาข้อมูลอินพุตวงจร SPI - ขาข้อมูลอนุกรมของระบบบัส I^2C
RC5/SDO	16 (24)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RC5 - ขาข้อมูลเอาต์พุตของวงจร SPI
RC6/TxD	17 (25)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RC6 - ขาเอาต์พุตวงจร USART สำหรับเชื่อมต่อพอร์ตอนุกรม
RC7/RxD	18 (26)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RC7 - ขาอินพุตวงจร USART สำหรับเชื่อมต่อพอร์ตอนุกรม
ขาพอร์ต D เป็นขาพอร์ต 2 ทิศทาง สามารถใช้เป็นส่วนขยายพอร์ตแบบขนานเพื่อติดต่อกับระบบบัสอื่น ไม่มีใน PIC16F873(A)/876(A)				
RD0/PSP0	(19)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล[3]	- ขาพอร์ต RD0 - ขาขยายพอร์ตแบบขนานบิต 0
RD1/PSP1	(20)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล[3]	- ขาพอร์ต RD1 - ขาขยายพอร์ตแบบขนานบิต 1
RD2/PSP2	(21)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล[3]	- ขาพอร์ต RD2 - ขาขยายพอร์ตแบบขนานบิต 2
RD3/PSP3	(22)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล[3]	- ขาพอร์ต RD3 - ขาขยายพอร์ตแบบขนานบิต 3
RD4/PSP4	(27)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล[3]	- ขาพอร์ต RD4 - ขาขยายพอร์ตแบบขนานบิต 4
RD5/PSP5	(28)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล[3]	- ขาพอร์ต RD5 - ขาขยายพอร์ตแบบขนานบิต 5
RD6/PSP6	(29)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล[3]	- ขาพอร์ต RD6 - ขาขยายพอร์ตแบบขนานบิต 6
RD7/PSP7	(30)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล[3]	- ขาพอร์ต RD7 - ขาขยายพอร์ตแบบขนานบิต 7

ตารางที่ 2.3 ตารางสรุปการทำงานของขาพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ PIC 16F877

(มีต่อ)

ชื่อขา	ตำแหน่งขา	ชนิดของขา	ชนิดของวงจรบัฟเฟอร์	รายละเอียดการทำงาน
ขาพอร์ต E เป็นขาพอร์ต 2 ทิศทาง ไม่มีใน PIC16F873(A)/876(A)				
RE0/AN5/$\overline{RD}$	(8)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล(3)	- ขาพอร์ต RE0 - อินพุตวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ช่อง 5 - ขาสัญญาณ $\overline{RD}$ สำหรับส่วนขยายพอร์ตแบบขนาน
RE1/AN6/$\overline{WR}$	(9)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล(3)	- ขาพอร์ต RE1 - อินพุตวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ช่อง 6 - ขาสัญญาณ $\overline{WR}$ สำหรับส่วนขยายพอร์ตแบบขนาน
RE2/AN7/$\overline{CS}$	(10)	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์ / ทีทีแอล(3)	- ขาพอร์ต RE2 - อินพุตวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล ช่อง 7 - ขาสัญญาณ $\overline{CS}$ สำหรับส่วนขยายพอร์ตแบบขนาน
ขาต่อไฟเลี้ยง				
V_{DD}	20 (11, 32)	อินพุต	-	- ขาต่อไฟเลี้ยง ใช้ได้ตั้งแต่ +2 ถึง +5.5V
V_{SS}	8, 19 (12, 31)	อินพุต	-	- ขาต่อกราวด์

หมายเหตุ

(1) อินพุตของวงจรบัฟเฟอร์จะเป็นแบบชมิตต์ทริกเกอร์ เมื่อใช้งานเป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอก

(2) อินพุตของวงจรบัฟเฟอร์จะเป็นแบบชมิตต์ทริกเกอร์ เมื่อทำงานในโหมดโปรแกรมข้อมูลอนุกรม (Serial programming mode)

(3) อินพุตของวงจรบัฟเฟอร์จะเป็นแบบชมิตต์ทริกเกอร์ เมื่อกำหนดให้ทำงานเป็นขาพอร์ตปกติ และเป็นแบบทีทีแอลเมื่อกำหนดให้ทำงานเป็นส่วนขยายพอร์ตแบบขนาน (PSP) สำหรับเชื่อมต่อกับระบบบัสไมโครโปรเซสเซอร์อื่น

(4) สำหรับ PIC16F874/877 อินพุตของวงจรบัฟเฟอร์จะเป็นแบบชมิตต์ทริกเกอร์ เมื่อกำหนดให้ทำงานในโหมด RC และเป็นซีมอสเมื่อทำงานในโหมดอื่น

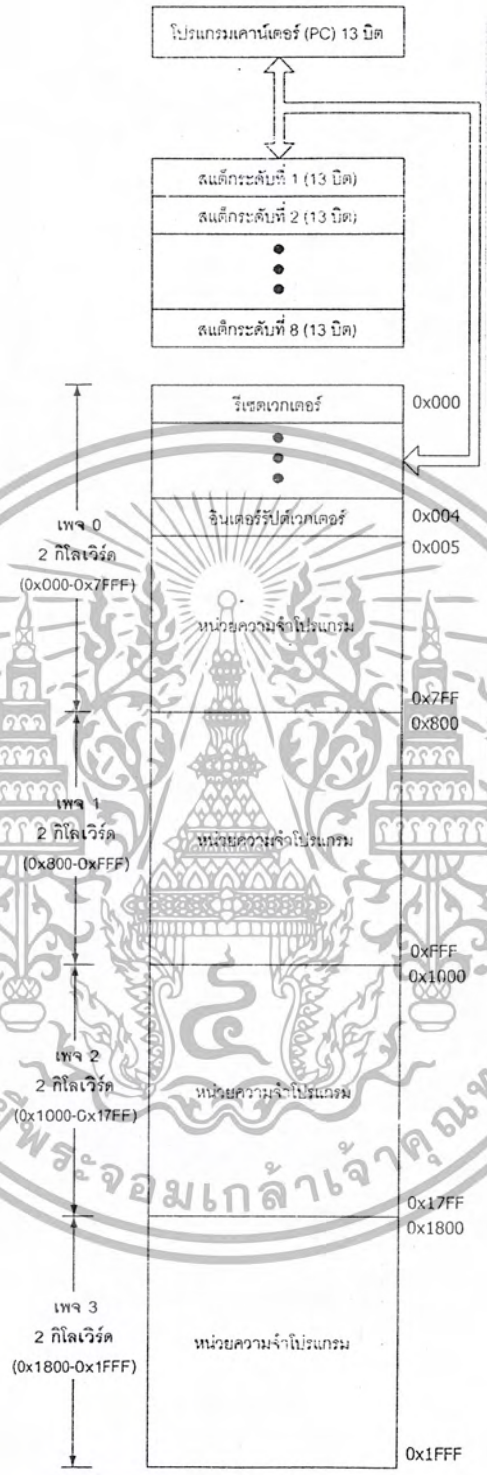
ตารางที่ 2.3 ตารางสรุปการทำงานของขาพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ PIC 16F877

2.2.2 การจัดสรรหน่วยความจำโปรแกรม

หน่วยความจำโปรแกรม(program memory) เป็นส่วนที่สำคัญมากเพราะเป็นที่เก็บข้อมูลคำสั่งทั้งหมดซึ่งใช้ในการกำหนดให้ไมโครคอนโทรลเลอร์ทำงาน หน่วยความจำโปรแกรมของ PIC16F877 เป็นแบบแฟลช (flash memory) ทำให้สามารถลบ และเขียนใหม่ได้แสนครั้ง แต่อย่างไรก็ตามโดยปกติ หน่วยความจำโปรแกรม หลังจากที่ทำการเขียนในขั้นตอนของการโปรแกรมแล้วก็จะมีไว้สำหรับอ่านออกมาได้เพียงทางเดียว

PIC16F877 มีโปรแกรมเคาน์เตอร์ (PC) ขนาด 13 บิต เพื่อกำหนดการเข้าถึงหน่วยความจำโปรแกรม มีขนาด 8K 14 บิต (หรือ 8 กิโลเวิร์ด) เนื่องจากไมโครคอนโทรลเลอร์ในอนุกรมนี้มีขนาดหน่วยความจำที่ค่อนข้างใหญ่มากจึงต้องจัดสรรเป็นเพจ (page) หรือ เป็นหน้า โดยในแต่ละเพจจะมีขนาด 2 กิโลเวิร์ด ทั้งนี้เนื่องจากชุดคำสั่งเกี่ยวกับการกระโดดของไมโครคอนโทรลเลอร์ ตระกูล PIC สามารถอ้างถึงตำแหน่งของหน่วยความจำได้สูงสุด 2,048 ตำแหน่ง

ในรูปที่ 2.19 แสดงการจัดสรรพื้นที่ ของหน่วยความจำโปรแกรมของ PIC16F877 ทั้งสองขนาด การจัดสรรดังกล่าวเป็นการจัดสรรปกติ



รูปที่ 2.19 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ PIC 16F877

สำหรับ PIC16F877 ซึ่งมีขนาดหน่วยความจำโปรแกรม 8 กิโลเวิร์ด มีการจัดสรรพื้นที่ดังในรูป 2.19 มีการสงวนแอดเดรส 0x0000 และ 0x0004 ไว้เช่นกัน หรืออาจกล่าวได้ว่า เป็นรูปแบบมาตรฐานของการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ สำหรับ PIC16F877 มีการแบ่งหน่วยความจำออกเป็น 4 เพจ ดังนี้

เพจ 0 มีแอดเดรสในช่วง 0x0000-0x07FF (โดยควรสงวนแอดเดรส 0x0000 และ 0x0004 ไว้)

เพจ 1 มีแอดเดรสในช่วง 0x0800-0x0FFF

เพจ 2 มีแอดเดรสในช่วง 0x1000-0x17FF

เพจ 3 มีแอดเดรสในช่วง 0x1800-0x1FFF

นอกจากนั้นใน PIC16F877 ยังมีพื้นที่หน่วยความจำพิเศษสำหรับเก็บค่าของโปรแกรมชั่วคราวขนาด 13 บิต เรียกว่า สแต็ก (stack) ซึ่งมีบทบาทมากสำหรับการกระโดดไปทำงานยังโปรแกรมย่อยของ PIC16F877 โดยเมื่อกระทำคำสั่งให้กระโดดไปทำงานยังโปรแกรมย่อย ซีพียูจะทำการเก็บค่าโปรแกรมเคาร์เตอร์หรือ PC ในขณะนั้นไว้ในสแต็ก จากนั้นกระโดดไปทำงานยังโปรแกรมย่อย เมื่อทำงานเรียบร้อยแล้ว ซีพียูจะไปอ่านค่า PC จากสแต็กกลับมา แล้วทำงานตามกระบวนการในโปรแกรมหลักต่อไป สำหรับสแต็กใน PIC16F877 มีขนาด 13 บิต สามารถเก็บค่าของ PC ได้ 8 ระดับ

2.2.3 การจัดสรรหน่วยความจำข้อมูลแรมและรีจิสเตอร์ไฟล์

ใน PIC 16F877 มีหน่วยความจำข้อมูลแรมสำหรับใช้งานทั่วไป 368 ไบต์ และมีรีจิสเตอร์ไฟล์ 8 บิต 57 ตัว ดังในรูปที่ 3.4 แต่ละแบงค์มีขนาดสูงสุด 128 ไบต์ แต่มีการใช้งานได้จริงในแต่ละแบงค์ต่างกัน โดยในแต่ละแบงค์มีการจัดสรรพื้นที่ดังนี้

แบงค์ 0 มีช่วงแอดเดรส 0x00-0x7F

แอดเดรส 0x00-0x1F เป็นพื้นที่ของรีจิสเตอร์ไฟล์

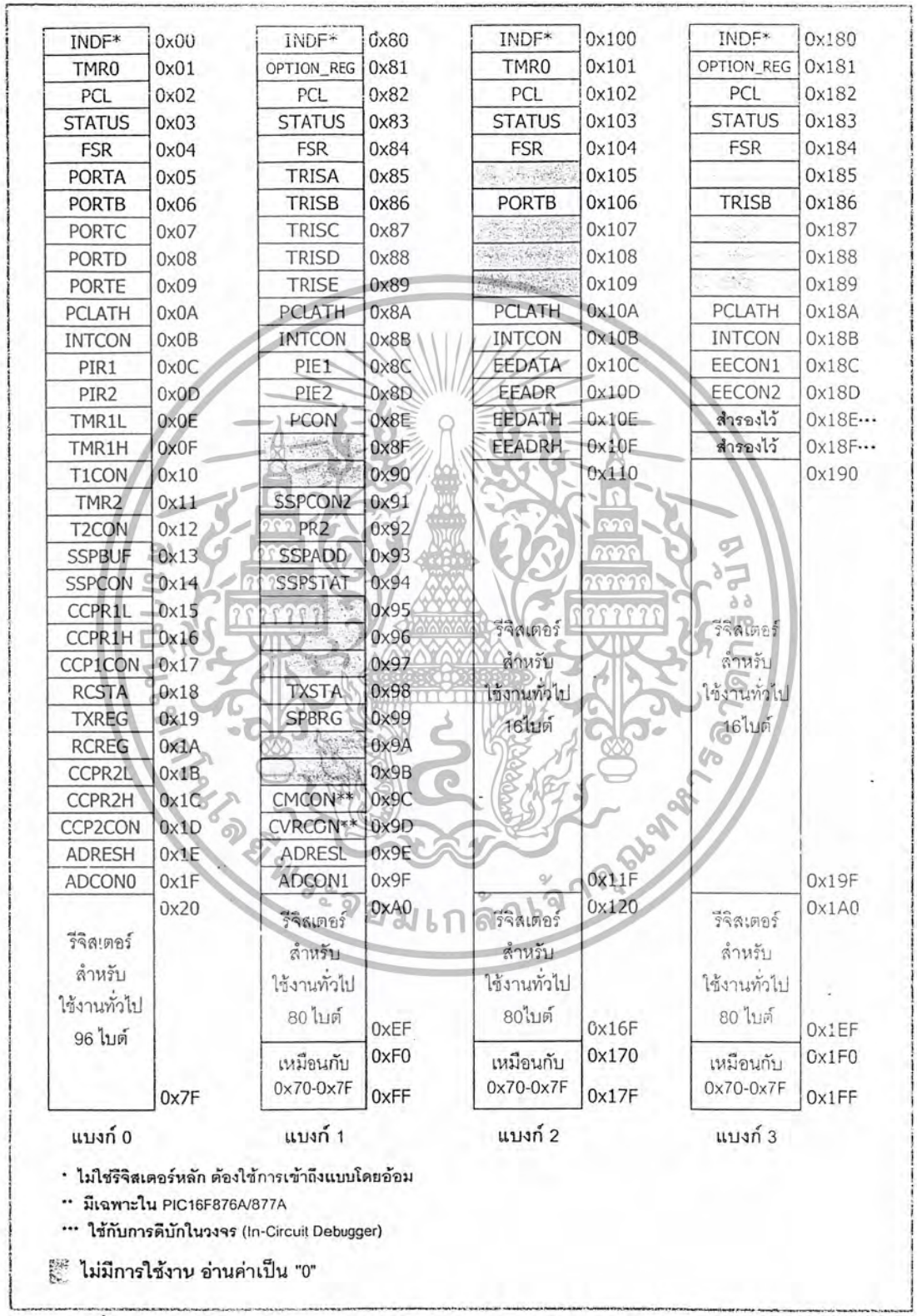
แอดเดรส 0x1F-0x7F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 96 ไบต์

แบงค์ 1 มีช่วงแอดเดรส 0x80-0xFF

แอดเดรส 0x80-0x9F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบางแอดเดรสไม่ใช้งาน

แอดเดรส 0xA0-0xEF เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80 ไบต์

แอดเดรส 0xF0-0xFF บรรจุข้อมูลเหมือนกับในแอดเดรส 0x70-0x7F ในแบงค์ 0
เพื่อช่วยให้สามารถใช้ข้อมูลจากแอดเดรส 0x70-0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยนแบงค์



รูปที่ 2.20 การจัดสรรหน่วยความจำข้อมูลแรมของไมโครคอนโทรลเลอร์ PIC 16F877

แบงค์ 2 มีช่วงแอดเดรส 0x100-0x17F

แอดเดรส 0x100-0x10F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบางแอดเดรสไม่ใช้งาน

แอดเดรส 0x110-0x11F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 16 ไบต์

แอดเดรส 0x120-0x16F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80 ไบต์

แอดเดรส 0x170-0x17F บรรจุข้อมูลเหมือนกับในแอดเดรส 0x70-0x7F ในแบงค์ 0 เพื่อช่วยให้สามารถใช้ข้อมูลจากแอดเดรส 0x70-0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยนแบงค์

แบงค์ 3 มีช่วงแอดเดรส 0x180-0x1FF

แอดเดรส 0x180-0x18F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบางแอดเดรสไม่ใช้งาน

แอดเดรส 0x190-0x19F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 16 ไบต์

แอดเดรส 0x1A0-0x1AF เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80 ไบต์

แอดเดรส 0x1F0-0x1FF บรรจุข้อมูลเหมือนกับในแอดเดรส 0x70-0x7F ในแบงค์ 0 เพื่อช่วยให้สามารถใช้ข้อมูลจากแอดเดรส 0x70-0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยนแบงค์

2.2.4 พอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ PIC16F87x

PIC16F877 มีพอร์ตให้ใช้งานตั้งแต่ 3-5 พอร์ต จำนวน 20-33 บิต ขึ้นอยู่กับเบอร์ของไมโครคอนโทรลเลอร์ และด้วยความสามารถของพอร์ตใน PIC16F877 ที่สามารถทำงานได้หลายอย่าง

ความสามารถในการจ่ายกระแสเอาต์พุตของพอร์ตที่ไฟเลี้ยง +5V คือ 25 mA ต่อขาทั้งกระแสซิงก์และกระแสซอร์ซ ในขณะที่กระแสเอาต์พุต รวมของพอร์ต A,B และ E มีค่าสูงสุด 200 mA ส่วน กระแสเอาต์พุตรวมของพอร์ต C และ D มีค่าสูงสุด 200 mA ดังนั้นในการออกแบบเพื่อขับโหลดทางเอาต์พุตรวมของพอร์ตต้องระวังเรื่องเอาต์พุตรวมที่ไมโครคอนโทรลเลอร์สามารถขับได้ด้วย

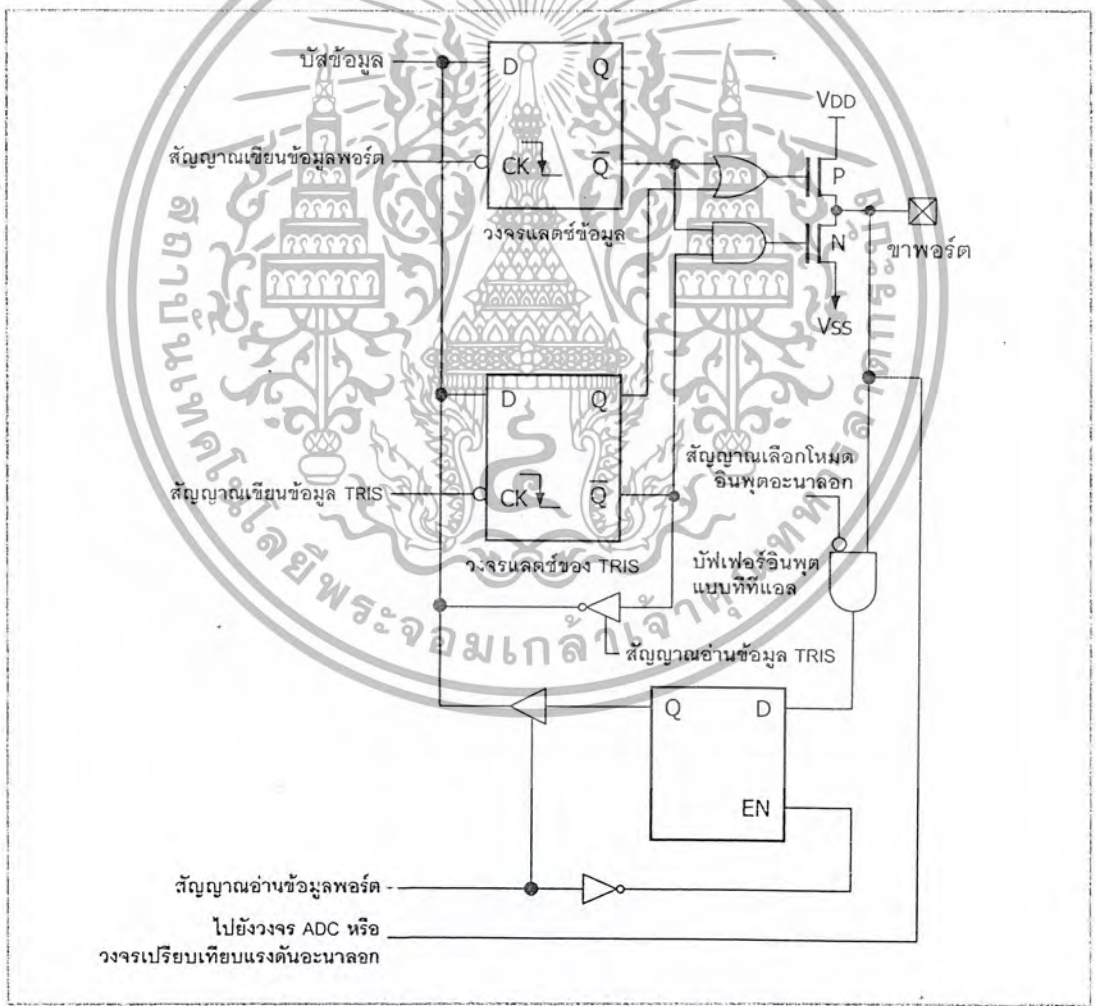
พอร์ต A

มีทั้งสิ้น 6 ช่องหรือ 6 บิต กำหนดชื่อขาเป็น RA0-RA5 รีจิสเตอร์ที่ใช้ในการเก็บข้อมูลคือ PORT A มีแอดเดรสอยู่ที่ 0x05 (แบงก์ 0) เป็นรีจิสเตอร์ขนาด 8 บิต แต่ใช้งานจริงเพียง 6 บิต ที่เหลือ 2 บิตต้องกำหนดให้เป็น "0" ส่วนการกำหนดทิศทางของพอร์ทนี้กระทำผ่านรีจิสเตอร์ TRISA ซึ่งมีแอดเดสอยู่ที่ 0x85 (แบงก์ 1) มีขนาด 8 บิต และใช้เพียง 6 บิตเช่นกัน 2 บิตบนคือบิต 6

และบิต 7 ต้องกำหนดให้เป็น "0" บิตของ TRISA ใช้กำหนดทิศทางของพอร์ท RA0 ไล่เรียงลำดับจนถึงบิต 5 ของ TRISA ใช้กำหนดทิศทางของขาพอร์ต RA5 หากต้องการกำหนดให้ขาพอร์ตในบิตเป็นอินพุตต้องเขียนข้อมูล "1" ไปยังบิตนั้น และในทางตรงกันข้ามหากต้องการกำหนดให้เป็นขาเอาต์พุตให้เขียนข้อมูล "0" ไปยังบิตนั้น

โครงสร้างทางฮาร์ดแวร์ ของพอร์ต A

พอร์ต A สามารถทำงานเป็นขาพอร์ตอินพุตปกติและเป็นขาอินพุตสัญญาณอะนาลอกสำหรับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 10 บิตภายในไมโครคอนโทรลเลอร์ โดยขา RA0-RA3 และ RA5 จะมีการทำงานที่เหมือนกัน ส่วน RA4 จะแตกต่างตรงที่ขานี้นอกจากเป็นขาพอร์ตอินพุตเอาต์พุตปกติแล้วยังใช้เป็นขาอินพุตสำหรับไทเมอร์ 0 ภายในไมโครคอนโทรลเลอร์ด้วย และขา RA4 นี้ไม่สามารถใช้งานเป็นขาอินพุตรับสัญญาณอะนาลอกได้



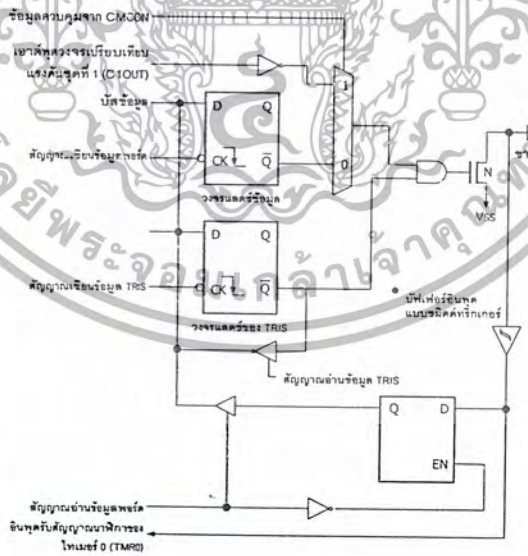
รูปที่ 2.21 โครงสร้าง RA0-RA3 ของพอร์ต A ในไมโครคอนโทรลเลอร์ PIC 16F877

ในรูปที่ 2.21 แสดงไดอะแกรมของพอร์ต A ในบิต RA0-RA3 จะเห็นได้ว่าที่ขาพอร์ตจะมีแอนด์เกตทำหน้าที่เลือกลักษณะการทำงานของขาพอร์ตเมื่อเป็นขาอินพุต หากเลือกให้ขาพอร์ตนี้

เป็นขาพอร์ตอินพุตดิจิตอล สัญญาณเลือกโหมดอินพุตอะนาลอกจะต้องเป็นลอจิก "0" แต่ถ้าหากต้องการกำหนดให้เป็นขาอินพุตสัญญาณอะนาลอก สัญญาณเลือกโหมดต้องเป็นลอจิก "1" สัญญาณอะนาลอกที่ป้อนเข้ามายังขานี้ จะผ่านเข้าสู่วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลหรือวงจรเปรียบเทียบแรงดันอะนาลอก (เฉพาะในอนุกรม PIC16F87xA) โดยตรง

เมื่อขาพอร์ต RA0-RA3 ทำงานเป็นขาพอร์ตอินพุตดิจิตอล จะสามารถรับสัญญาณดิจิตอลระดับทีทีแอล (0-5V) ได้โดยตรง หากการทำงานเป็นเอาต์พุตจะสามารถขับโหลดที่ต้องการกระแส 20 mA ได้ หากนำมาขับ LED ต้องต่อตัวต้านทานจำกัดกระแส หรือถ้าใช้ไฟเลี้ยง 3 V ก็จะขับ LED ได้โดยตรง

ในรูปที่ 2.22 เป็นไดอะแกรมของขาพอร์ต RA4/T0CKI โดยขานี้จะเป็นขาพอร์ตอินพุตเอาต์พุตปกติและขาอินพุตรับสัญญาณนาฬิกาจากภายนอกโมดูลไทเมอร์ 0 วงจรอินพุตบัฟเฟอร์ที่ขาพอร์ตนี้เป็นแบบชมิตต์ทริกเกอร์ ทั้งนี้เพื่อจัดการให้สัญญาณอินพุตที่เข้ามามีความสมบูรณ์มากที่สุด และจะต้องต่อตัวต้านทานพูลอัปค่าประมาณ 4.7k-10k ที่ขานี้เสมอเมื่อใช้งานเป็นอินพุต



รูปที่ 2.22 โครงสร้างขา RA4 ของพอร์ต A ในไมโครคอนโทรลเลอร์ PIC 16F877

พอร์ต B

มี 8 บิต กำหนดชื่อขาเป็น RB0-RB7 รีจิสเตอร์ที่ใช้ในการเก็บข้อมูลคือ PORTA มีแอดเดรสอยู่ที่ 0x06 (แบงก์ 0) และ 0x106 (แบงก์ 2) เป็นรีจิสเตอร์ขนาด 8 บิต ส่วนการกำหนดทิศทางของพอร์ตนี้กระทำผ่านรีจิสเตอร์ TRISB ซึ่งมีแอดเดรสอยู่ที่ 0x86 (แบงก์ 1) และ 0x186 (แบงก์ 3) มีขนาด 8 บิต เช่นเดียวกับพอร์ต A บิต 0 ของ TRISBใช้กำหนดทิศทางของขาพอร์ต RB0 ไล่เรียงลำดับจนถึงบิต 7 ของ TRISB ใช้กำหนดทิศทางของขาพอร์ต RB7 หากต้องการกำหนดให้ขาพอร์ตในบิตใดเป็นอินพุตต้องเขียนข้อมูล "1" ไปยังบิตนั้น ในทางตรงกันข้ามหากต้องการกำหนดให้เป็นขาเอาต์พุตให้เขียนข้อมูล "0" ไปยังบิตนั้น

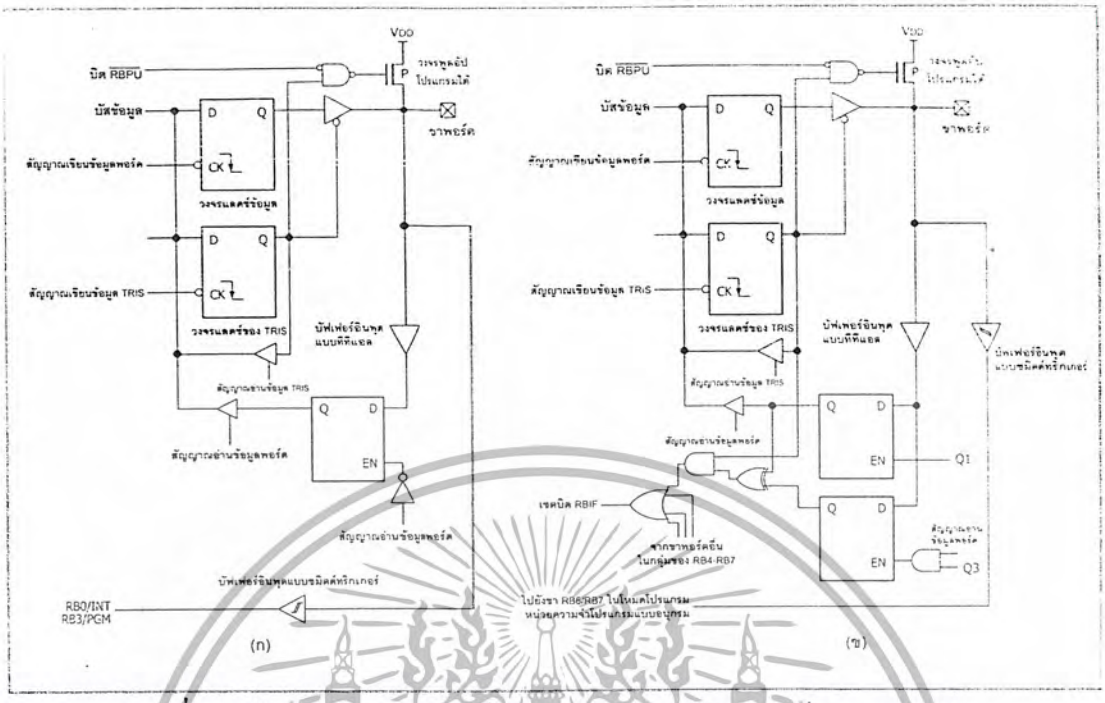
โครงสร้างทางฮาร์ดแวร์ของพอร์ต B

พอร์ต B สามารถใช้งานในลักษณะต่าง ๆ ได้ 5 แบบคือ

1. เป็นขาพอร์ตอินพุตเอาต์พุตปกติ
2. เป็นขาอินพุตสัญญาณอินเตอร์รัปต์จากภายนอก โดยใช้ขา RB0/INT
3. เป็นขาพอร์ตอินพุตสำหรับรับแรงดันโปรแกรมระดับต่ำ (low voltage programming) โดยใช้ขา RB3/PGM
4. เป็นขาข้อมูลอนุกรมและสัญญาณนาฬิกาอนุกรมสำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ ซึ่งใช้ 2 ขา คือ RB7/PGD และ RB6/PGC
5. ใช้เป็นแหล่งกำเนิดสัญญาณอินเตอร์รัปต์แบบตรวจสอบการเปลี่ยนแปลงข้อมูลหรือระดับสัญญาณที่ขา RB4-RB7

วงจรอินพุตบัฟเฟอร์ที่ขาพอร์ตนี้มีทั้งแบบทีทีแอลและชมิตต์ทริกเกอร์ ทั้งนี้เพื่อจัดการให้สัญญาณอินพุตที่เข้ามามีความเหมาะสมและสมบูรณ์มากที่สุด และยังคงสามารถรองรับการพูลอัปภายในแบบอัตโนมัติได้

ในกรณีที่มีการเอ็นเอเบิลตอบสนองอินเตอร์แบบตรวจสอบ การเปลี่ยนแปลงลอจิกที่พอร์ต RB4-RB7 หากเกิดการอินเตอร์รัปต์ขึ้น บิต RBIF (บิต 0 ในรีจิสเตอร์ INTCON) จะเซตและหลังจากการตอบสนองการอินเตอร์รัปต์แล้ว ต้องเคลียร์บิต RBIF ด้วยกระบวนการทางซอฟต์แวร์เสมอ



รูปที่ 2.23 โครงสร้างของขาพอร์ต B ในไมโครคอนโทรลเลอร์ PIC 16F877

(ก) ขา RB0-RB3 (ข) ขา RB4-RB7

พอร์ต C

มีทั้งสิ้น 8 บิต กำหนดชื่อขาเป็น RC0-RC7 รีจิสเตอร์ที่ใช้เก็บข้อมูลคือ PORTC มีแอดเดรสอยู่ที่ 0x07(แบงก์ 0) เป็นรีจิสเตอร์ขนาด 8 บิต ส่วนการกำหนดทิศทางของพอร์ตนี้กระทำผ่านรีจิสเตอร์ TRISC มีแอดเดรสอยู่ที่ 0x87(แบงก์ 1 มีขนาด 8 บิต เช่นเดียวกับพอร์ต A และ B ของ TRISC ใช้กำหนดทิศทางของขาพอร์ต RC0 ไล่เรียงลำดับจนถึงบิต 7 ของ TRISC ใช้กำหนดทิศทางของขาพอร์ต RC 7 หากต้องการกำหนดให้ขาพอร์ตในบิตใดเป็นอินพุตต้องเขียนข้อมูล "1 " ไปยังบิตนั้น และในทางตรงข้ามหากต้องการกำหนดให้เป็นขาเอาต์พุต ให้เขียนข้อมูล "0" ไปยังบิตนั้น

โครงสร้างทางฮาร์ดแวร์ของพอร์ต C

พอร์ต C สามารถใช้งานในลักษณะต่างๆ ได้หลายรูปแบบ และเป็นขาพอร์ตที่มีความสามารถสูงมาก ไม่ว่าจะเป็นขาพอร์ตอินพุตเอาต์พุตปกติ, ขาอินพุตเอาต์พุตออสซิลเลเตอร์ของโมดูลไทเมอร์ 1, ขาอินพุตสำหรับรับสัญญาณนาฬิกาของโมดูลไทเมอร์ 1,ขาเชื่อมต่อระบบบัส I2C, ขาเชื่อมต่อแบบ SPI(Serial Peripheral Interface), ขาเชื่อมต่ออนุกรมแบบ USART, ขาอินพุตวงจรแคปเจอร์ (capture) หรือวงจรตรวจจับสัญญาณ ขาเอาต์พุตของวงจรเปรียบเทียบสัญญาณ (compare) และขาเอาต์พุตวงจร PWM (Pulse Width Modulation) หรืออาจกล่าวได้ว่าพอร์ต C เป็น

พอร์ตสำหรับเชื่อมต่ออุปกรณ์ภายนอก (peripheral function port) ที่มีความสมบูรณ์แบบมากที่สุด สามารถสรุปหน้าที่และการทำงานที่หลากหลายของขาพอร์ต C ดังตารางที่ 2.4

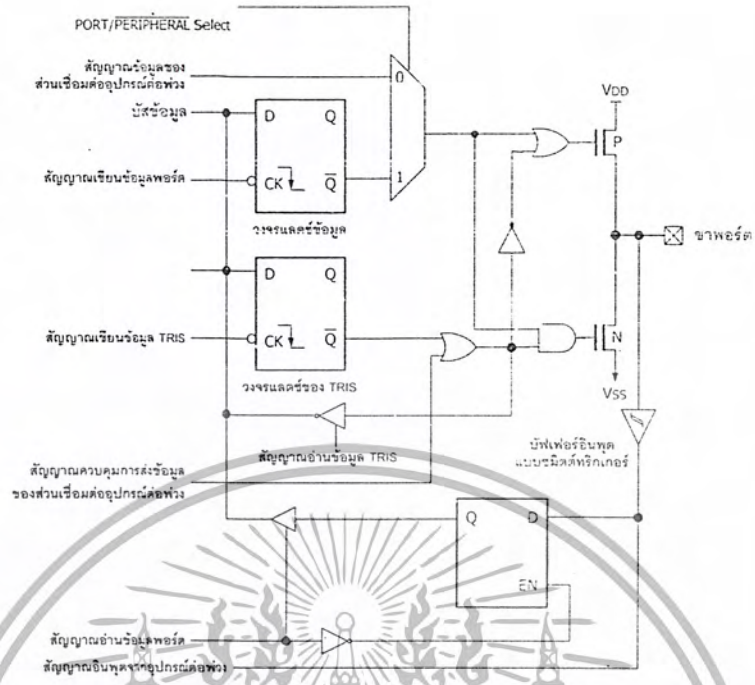
ในรูปที่ 2.24 แสดงไดอะแกรมของพอร์ต C ในบิต RC0-RC2 และ RC5-RC7 จะเห็นได้ว่ามีสัญญาณควบคุมการทำงานของขาพอร์ตมากมาย ทั้งนี้เนื่องจากขาพอร์ต C สามารถทำงานได้หลากหลายนั่นเอง สัญญาณควบคุมที่สำคัญคือ สัญญาณเลือกการทำงานระหว่างเป็นพอร์ตปกติหรือเป็นขาเชื่อมต่ออุปกรณ์พิเศษ (PORT/Peripheral Select) และสัญญาณควบคุมการส่งข้อมูลของวงจรเชื่อมต่ออุปกรณ์ (Peripheral Output Enable) สำหรับข้อมูลของวงจรเชื่อมต่ออุปกรณ์ที่ส่งออกและรับเข้ามาจะผ่านทางขาพอร์ตปกติ แต่เมื่อผ่านวงจรสำหรับเลือกสัญญาณข้อมูลแล้ว สายสัญญาณข้อมูลของพอร์ต (data bus) กับข้อมูลของวงจรเชื่อมต่ออุปกรณ์ (peripheral output/peripheral input) จะแยกกัน

ในรูปที่ 2.25 เป็นไดอะแกรมของขาพอร์ต RC3-RC4 ทั้ง 2 ขานี้มีความพิเศษที่สามารถใช้งานเป็นขาเชื่อมต่ออุปกรณ์อนุกรมแบบซิงโครนัส ซึ่งแบ่งเป็นระบบ SPI และระบบบัส I2C ซึ่งทำให้ต้องเพิ่มสายสัญญาณควบคุมอินพุตเพิ่มเข้ามาอีก 1 เส้น เพื่อเลือกสัญญาณอินพุตระหว่าง SPI และบัส

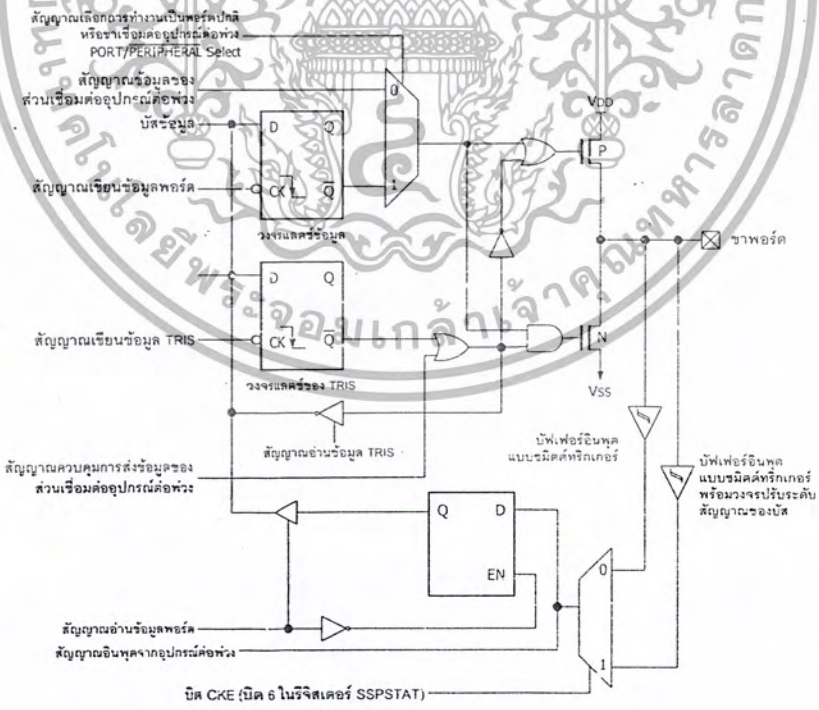
วงจรอินพุตบัฟเฟอร์ของขาพอร์ต C นี้แบบชมิตต์ทริกเกอร์ทั้งหมด ทั้งนี้เพื่อจัดการให้สัญญาณอินพุตที่เข้ามามีความเหมาะสมและสมบูรณ์มากที่สุด และยังคงสามารถรองรับการพูลอัปภายในแบบอัตโนมัติได้

ชื่อขา	การทำงาน
$\overline{RC0}$/T1OSO/T1CKI	- ขาพอร์ตอินพุตเอาต์พุตบิต 0 ของพอร์ต C - เอาต์พุตวงจรออสซิลเลเตอร์ของไทเมอร์ 1 (T1OSO) - อินพุตรับสัญญาณนาฬิกาของไทเมอร์ 1 (T1CKI)
RC1/T1OSI/CCP2	- ขาพอร์ตอินพุตเอาต์พุตบิต 1 ของพอร์ต C - อินพุตวงจรออสซิลเลเตอร์ของไทเมอร์ 1 (T1OSI) - อินพุตแคปเจอร์หรือวงจรตรวจจับสัญญาณของโมดูล CCP2 - เอาต์พุตวงจรเปรียบเทียบของโมดูล CCP2 - เอาต์พุต PWM ของโมดูล CCP2
RC2/CCP1	- ขาพอร์ตอินพุตเอาต์พุตบิต 2 ของพอร์ต C - อินพุตแคปเจอร์หรือวงจรตรวจจับสัญญาณของโมดูล CCP1 - เอาต์พุตวงจรเปรียบเทียบของโมดูล CCP1 - เอาต์พุต PWM ของโมดูล CCP1
RC3/SCK/SCL	- ขาพอร์ตอินพุตเอาต์พุตบิต 3 ของพอร์ต C - ขาสัญญาณนาฬิกาอนุกรมของระบบ SPI (SCK) - ขาสัญญาณนาฬิกาอนุกรมของระบบบัส I^2C (SCL)
RC4/SDI/SDA	- ขาพอร์ตอินพุตเอาต์พุตบิต 4 ของพอร์ต C - ขาข้อมูลอินพุตอนุกรมของระบบ SPI (SDI) - ขาข้อมูลอนุกรมของระบบบัส I^2C (SDA)
RC5/SDO	- ขาพอร์ตอินพุตเอาต์พุตบิต 5 ของพอร์ต C - ขาข้อมูลเอาต์พุตอนุกรมของระบบ SPI (SDO)
RC6/TxD/CK	- ขาพอร์ตอินพุตเอาต์พุตบิต 6 ของพอร์ต C - ขาส่งข้อมูลพอร์ตอนุกรม USART (TxD) - ขาสัญญาณนาฬิกาซิงโครนัส (CK)
RC7/RxD/DT	- ขาพอร์ตอินพุตเอาต์พุตบิต 7 ของพอร์ต C - ขารับข้อมูลพอร์ตอนุกรม USART (RxD) - ขาสัญญาณข้อมูลซิงโครนัส (DT)

ตารางที่ 2.4 สรุปหน้าที่การทำงานของขาพอร์ต C ในไมโครคอนโทรลเลอร์ PIC 16F877



รูปที่ 2.24 โครงสร้างขา RC0-RC2, RC5-RC7 ของพอร์ต C ในไมโครคอนโทรลเลอร์ PIC 16F877



รูปที่ 2.25 โครงสร้างขา RC3 และ RC4 ของพอร์ต C ในไมโครคอนโทรลเลอร์ PIC 16F877

พอร์ต D

มี 8 บิต กำหนดชื่อขาเป็น RD0-RD7 รีจิสเตอร์ที่ใช้ในการเก็บข้อมูลคือ PORTD มีแอดเดรสอยู่ที่ 0x08 (แบงก์ 0) เป็นรีจิสเตอร์ขนาด 8 บิต ส่วนการกำหนดทิศทางของพอร์ตนี้กระทำผ่านรีจิสเตอร์ TRISD มีแอดเดรสอยู่ที่ 0x088 (แบงก์ 1) มีขนาด 8 บิต หากต้องการกำหนดให้ขาพอร์ตในบิตใดเป็นอินพุตต้องเขียนข้อมูล "1" ไปยังบิตนั้น และในทางตรงข้ามหากต้องการกำหนดให้ขาเอาต์พุต ให้เขียนข้อมูล "0" ไปยังบิตนั้น สำหรับพอร์ต D นี้จะมีเฉพาะในไมโครคอนโทรลเลอร์ PIC รุ่น 40 ขาเท่านั้น สำหรับในอนุกรม PIC19F877

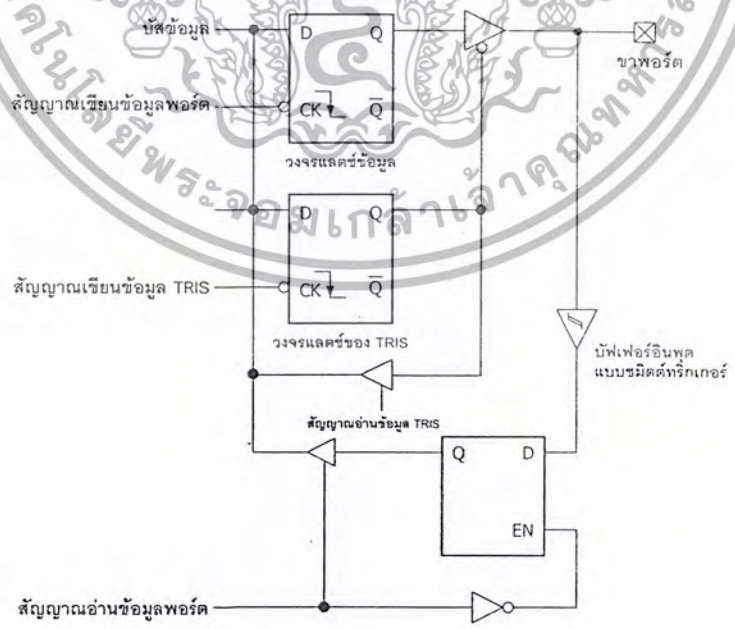
โครงสร้างทางฮาร์ดแวร์ของพอร์ต D

พอร์ต D สามารถใช้งานได้ 2 โหมดคือ เป็นขาพอร์ตอินพุตเอาต์พุตปกติและเป็นส่วนขยายพอร์ตแบบขนาน (Parallel Slave Port : PSP) สำหรับเชื่อมต่ออุปกรณ์ภายนอกที่มีการจัดระบบบัสแบบไมโครโปรเซสเซอร์คือ สายสัญญาณควบคุมการอ่าน (RD : Read)

เขียน (WR : Write) และเลือกอุปกรณ์ (CS : Chip Select)

ในรูปที่ 2.26 แสดงไดอะแกรมของพอร์ต D ซึ่งมีโครงสร้างเหมือนกันทุกบิต เมื่อทำงานในโหมดพอร์ตอินพุตเอาต์พุตปกติ วงจรอินพุตจะเป็นแบบชมิตต์ทริกเกอร์ แต่เมื่อทำงานในโหมดขยายพอร์ตแบบขนานหรือ PSP วงจรอินพุตจะเปลี่ยนเป็นแบบทีทีแอล

การเลือกโหมดการทำงานของพอร์ต D นี้ขึ้นกับบิต PSPMODE (บิต 4 ในริจิสเตอร์ TRISE) ถ้าเป็น "0" เป็นการกำหนดให้พอร์ต D เป็นพอร์ตปกติ และถ้าเป็น "1" พอร์ต D จะทำงานในโหมด PSP



รูปที่ 2.26 โครงสร้างของพอร์ต D ในไมโครคอนโทรลเลอร์ PIC 16F877

พอร์ต E

มี 3 บิต กำหนดชื่อขาเป็น RE0-RE2 รีจิสเตอร์ที่ใช้ในการเก็บข้อมูลคือ PORTE มีแอดเดรสอยู่ที่ 0x09 (แบงก์ 0) เป็นรีจิสเตอร์ขนาด 8 บิต แต่ใช้งานเพียง 3 บิตล่างคือ บิต 0- บิต 2 เท่านั้น ที่เหลือกำหนดเป็น "0" ส่วนการกำหนดทิศทางของพอร์ตนี้กระทำผ่านรีจิสเตอร์ TRISE ซึ่งมีแอดเดรสอยู่ที่ 0x89 (แบงก์ 1) มีขนาด 8 บิต โดยใช้ 3 บิตล่างในการกำหนดทิศทางของพอร์ต E ส่วนที่เหลือใช้ควบคุมการทำงานในโหมด PSP ของพอร์ต D

พอร์ต E สามารถใช้งานเป็นพอร์ตอินพุตเอาต์พุตปกติ, ขาอินพุตอะนาลอกของโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล และขาควบคุมการติดต่อกับอุปกรณ์ภายนอกแบบ PSP ทั้งนี้ขึ้นอยู่กับการกำหนดข้อมูลของรีจิสเตอร์ที่ใช้ควบคุมการทำงานของพอร์ตนี้ เช่นเดียวกับพอร์ต D พอร์ต E จะมีเฉพาะในไมโครคอนโทรลเลอร์ PIC รุ่น 40 ขาเท่านั้น

รูปที่ 2.27 โครงสร้างของพอร์ต E ในไมโครคอนโทรลเลอร์ PIC 16F877

รีจิสเตอร์ TRISE

เมื่อมีการกำหนดให้พอร์ต D ทำงานในโหมด PSP 4 บิตบนของรีจิสเตอร์ TRISE จะใช้เป็นบิตแสดงสถานะของวงจรบัฟเฟอร์ที่ใช้ในการถ่ายทอดข้อมูลของส่วนขยายพอร์ตแบบขนานหรือ PSP ส่วน 3 บิตล่างใช้ในการเลือกโหมดการทำงานของพอร์ต D

ถ้าหากพอร์ต E ทำงานเป็นพอร์ตอินพุตเอาต์พุตปกติ 3 บิตล่างของรีจิสเตอร์ TRISE จะใช้ในการกำหนดทิศทางการถ่ายทอดสัญญาณข้อมูลของพอร์ต ดังมีรายละเอียดการทำงานของแต่ละบิตของรีจิสเตอร์ TRISE ดังนี้

	บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
TRISE	IBF	OBF	IBOV	PSPMODE	-	DIR-RE2	DIR-RE1	DIR-RE0
	R -0	R -0	R/W -0	R/W -0		R/W -1	R/W -1	R/W -1

IBF (Input Buffer Full status bit) : บิตแสดงสถานะบัฟเฟอร์ทางอินพุตของ PSP

"0" แสดงว่า ไม่มีข้อมูลรับเข้ามาในบัฟเฟอร์

"1" แสดงว่า มีข้อมูลเข้ามาแล้วและรอการอ่านจากซีพียู

OBF (Output Buffer Full status bit) : บิตแสดงสถานะบัฟเฟอร์ทางเอาต์พุตของ PSP

"0" แสดงว่า ข้อมูลในบัฟเฟอร์ถูกส่งออกไปแล้ว

"1" แสดงว่า ยังคงมีข้อมูลเดิมอยู่ในบัฟเฟอร์ทางเอาต์พุต

IBOV (Input Buffer Overflow Detect bit) : แสดงว่าการรับข้อมูลเกินของบัฟเฟอร์อินพุต

"0" แสดงว่า ไม่มีการรับข้อมูลเกินหรือ โอเวอร์โฟลวเกิดขึ้น

"1" แสดงว่า เกิดการเขียนข้อมูลมายังบัฟเฟอร์อินพุต ทั้งที่ยังคงมีข้อมูลเดิมอยู่

PSPMODE (Parallel Slave Port Mode select bit) : บิตเลือกการทำงาน PSP ของพอร์ต D และ E

"0" แสดงว่า กำหนดให้ทำงานในโหมดพอร์ตอินพุตเอาต์พุตปกติ

"1" แสดงว่า กำหนดให้ทำงานในโหมด PSP

DIR-RE2 หรือ DIR-RE0 (Direction control bit for RE2-RE0) : บิตควบคุมทิศทางของพอร์ต E เมื่อทำงานในโหมดพอร์ตอินพุตเอาต์พุตปกติ

"0" เป็นเอาต์พุต

"1" เป็นอินพุต

<u>โครงสร้างทางฮาร์ดแวร์ของพอร์ต E</u>

ในรูปที่ 2.28 แสดงไดอะแกรมของพอร์ต E ในบิต RE0-RE2 เมื่อทำงานในโหมดพอร์ตอินพุตเอาต์พุตปกติ จะเห็นได้ว่าความคล้ายคลึงกับขาพอร์ตนี้จะเป็นแบบชมิตต์ทริกเกอร์ ในขณะที่หากทำงานในโหมดการแปลงสัญญาณอะนาลอกเป็นดิจิตอล วงจรอินพุตจะเปลี่ยนเป็นแบบทีทีแอล

ถึงแม้ว่าพอร์ต E ใน PIC 16F877 มีจำนวนน้อยเพียง 3 บิต แต่สามารถเลือกรูปแบบการทำงานได้มาถึง 3 แบบ คือเป็นพอร์ตอินพุตเอาต์พุตปกติ, อินพุตสำหรับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 10 บิตในไมโครคอนโทรลเลอร์ และพอร์ตสัญญาณควบคุมสำหรับการ

ติดต่อกับอุปกรณ์ในโหมด PSP ดังนั้นในการเลือกรูปแบบการทำงานต้องระมัดระวังเช่นเดียวกับพอร์ต C ที่ได้กล่าวมาแล้วก่อนหน้านี้

ในตารางที่ 2.5 แสดงการทำงานอย่างละเอียดของขาพอร์ต E ในโหมดต่างๆ

ชื่อขา	การทำงาน
RE0/$\overline{RD}$/AN5	- ขาพอร์ตอินพุต/เอาต์พุตบิต 0 ของพอร์ต E - ขาควบคุมการอ่านข้อมูลสำหรับส่วนขยายพอร์ตแบบขนานหรือโหมด PSP ($\overline{RD}$) "0" = มีการอ่านข้อมูลเกิดขึ้น โดยทำการอ่านข้อมูลจากรีจิสเตอร์ PORTD ในกรณีที่มีการเลือกอุปกรณ์ที่ต้องการติดต่อด้วย (ขา $\overline{CS}$ = "0") "1" = ไม่มีการอ่านข้อมูลเกิดขึ้น - อินพุตรับสัญญาณอะนาลอกช่อง 5
RE1/$\overline{WR}$/AN6	- ขาพอร์ตอินพุต/เอาต์พุตบิต 1 ของพอร์ต E - ขาควบคุมการเขียนข้อมูลสำหรับส่วนขยายพอร์ตแบบขนานหรือโหมด PSP ($\overline{RD}$) "0" = มีการเขียนข้อมูลเกิดขึ้น โดยเขียนข้อมูลไปยังจากรีจิสเตอร์ PORTD ในกรณีที่มีการเลือกอุปกรณ์ที่ต้องการติดต่อด้วย (ขา $\overline{CS}$ = "0") "1" = ไม่มีการเขียนข้อมูลเกิดขึ้น - อินพุตรับสัญญาณอะนาลอกช่อง 6
RE2/$\overline{CS}$/AN7	- ขาพอร์ตอินพุต/เอาต์พุตบิต 2 ของพอร์ต E - ขาเลือกอุปกรณ์ที่ต้องการติดต่อด้วยสำหรับส่วนขยายพอร์ตแบบขนานหรือ PSP ($\overline{CS}$) "0" = มีการเลือกอุปกรณ์ "1" = ไม่มีการเลือกอุปกรณ์ - อินพุตรับสัญญาณอะนาลอกช่อง 7

ตารางที่ 2.5 แสดงหน้าที่การทำงานของขาพอร์ต E ในไมโครคอนโทรลเลอร์ PIC 16F877

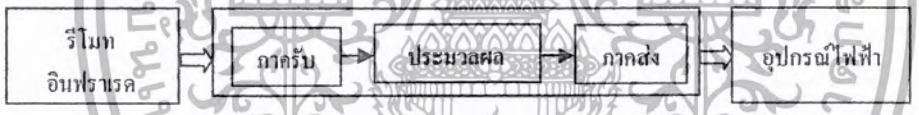
บทที่ 3

หลักการออกแบบและการสร้าง

3.1 รายละเอียดของโครงงาน

โครงงานนี้พัฒนาการใช้รีโมทแบบอินฟราเรดทั่วไปมาควบคุมอุปกรณ์ไฟฟ้าต่างๆ โดยจะทำการสร้างอุปกรณ์รับส่งสัญญาณอินฟราเรดภายในตัวเอง และมีไมโครคอนโทรลเลอร์ PIC 16F877 เป็นตัวเก็บข้อมูลของสัญญาณอินฟราเรดและประมวลผลสัญญาณเอาท์พุต โดยมีหลักการทำงานดังนี้

เมื่อมีการกดรีโมทคอนโทรลทั่วไปจะเป็นสัญญาณอินพุตของภาครับสัญญาณอินฟราเรด ในภาครับจะทำการขยายสัญญาณ กรองความถี่แบนด์พาสและดีมอดูเลตสัญญาณที่รับมา ทำให้ได้เฉพาะสัญญาณข้อมูลเข้าไมโครคอนโทรลเลอร์ แล้วไมโครคอนโทรลเลอร์จะนำสัญญาณมาประมวลผลในการกดรีโมทครั้งแรกเป็นการเลือกอุปกรณ์ไฟฟ้า ทำให้โครงงานเสมือนเป็นรีโมทของเครื่องใช้ไฟฟ้าที่ต้องการ เมื่อมีการส่งสัญญาณข้อมูลจากรีโมทอินฟราเรดครั้งต่อๆไป ไมโครคอนโทรลเลอร์จะนำมาประมวลผลและส่งสัญญาณเป็นชุดคำสั่งส่งออกไปยังภาคส่งสัญญาณอินฟราเรด ให้ส่งสัญญาณข้อมูลไปควบคุมอุปกรณ์ไฟฟ้าที่กำหนดไว้



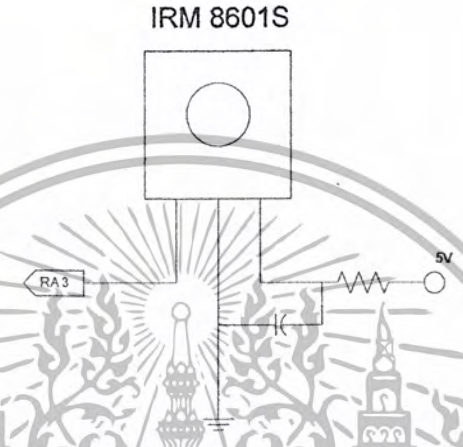
รูปที่ 3.1 บล็อกไดอะแกรมการทำงานของโครงงานนี้

ในโครงงานนี้มีส่วนประกอบหลัก 5 ส่วนคือ

1. ส่วนภาครับสัญญาณ
2. ส่วนแสดงผลLCD
3. ส่วนควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า
4. ส่วนภาคส่งสัญญาณ
5. ส่วนประมวลผล

3.2 ภาครับสัญญาณอินฟราเรด

ในภาครับสัญญาณได้ทดลองทำการเก็บค่าสัญญาณรีโมทที่จะนำมาเป็นสัญญาณอินพุตของไมโครคอนโทรลเลอร์ โดยเลือกการส่งสัญญาณแบบ RC5 ในการทดลองและได้ทำการเก็บค่าสัญญาณจากการกดปุ่มต่างๆของรีโมท โดยเลือกใช้ICเบอร์ IRM-8601S เป็นตัวรับสัญญาณอินฟราเรด



รูปที่3.2 วงจรรับสัญญาณอินฟราเรด

IRM-8601S เป็น IC สำเร็จรูปที่สามารถให้เอาท์พุตที่สามารถนำไปต่อเข้ากับไมโครคอนโทรลเลอร์ได้เลย โดยIC IRM –8601Sมีคุณสมบัติดังนี้

- ใช้ low voltage และ กำลังไฟฟ้าต่ำ
- มีวงจรขยายสัญญาณ กรองความถี่แบนด์พาสและดีมอดูเลตอยู่ภายใน
- มีความไวสูง
- รวมการใช้ TTLและ CMOSภายในIC
- ป้องกันการรบกวนของแสงได้ดี
- สามารถป้องกันการรบกวนจากสนามแม่เหล็กได้
- รับสัญญาณได้ในระยะไกล

สามารถรับสัญญาณของรีโมทคอนโทรลของอุปกรณ์ต่างๆดังนี้

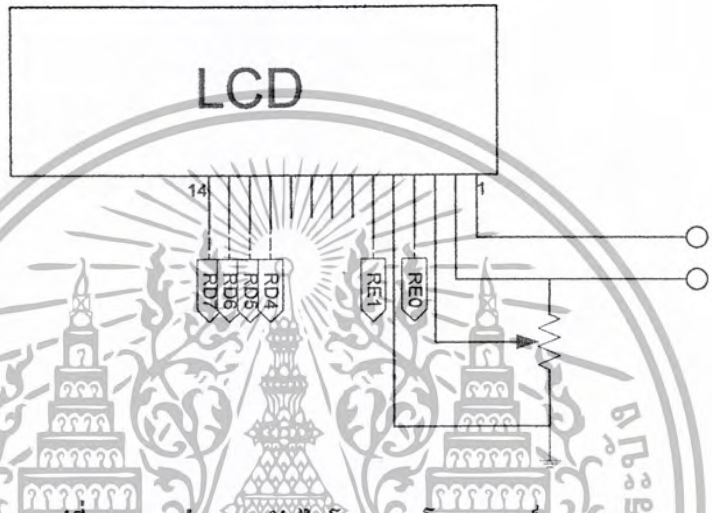
- TVs
- VCRs
- เครื่องเสียง
- เครื่องปรับอากาศ
- พัดลมไฟฟ้า

- สวิตช์ที่ใช้แสงควบคุม

ทำให้ส่วนของภาครับสัญญาณสามารถรับสัญญาณได้หลายรูปแบบ โดยจะส่งสัญญาณต่อไปยังไมโครคอนโทรลเลอร์เพื่อประมวลผลต่อไป

3.3 ส่วนแสดงผลLCD

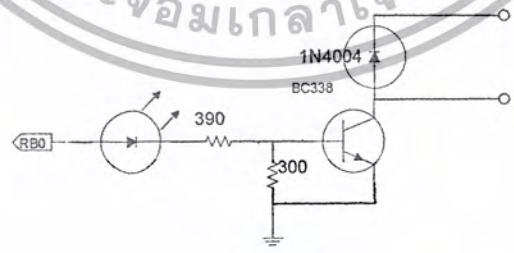
เป็นส่วนที่แสดงผลว่าขณะนี้ได้ส่งสัญญาณเอาท์พุตแบบใดออกไปยังอุปกรณ์ไฟฟ้าสามารถทำให้ผู้ใช้ได้ทราบว่ากำลังควบคุมอุปกรณ์ไฟฟ้าชนิดใด



รูปที่ 3.3 การต่อLCDกับไมโครคอนโทรลเลอร์

3.4 ส่วนควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า

การควบคุมการเปิดปิดอุปกรณ์ไฟฟ้าเป็นการรับคำสั่งจากไมโครคอนโทรเลอร์ ในโปรแกรมประมวลผลแบบควบคุมการเปิดปิด โดยมีLEDแสดงสถานะของอุปกรณ์และในวงจรมีทรานซิสเตอร์เพื่อช่วยเพิ่มกระแสในวงจรขับ ซึ่งจะให้เอาท์พุต12โวลต์ และสามารถนำเอาท์พุตที่ได้ไปใช้ในการขับรีเลย์เพื่อการเปิดปิดอุปกรณ์ไฟฟ้า



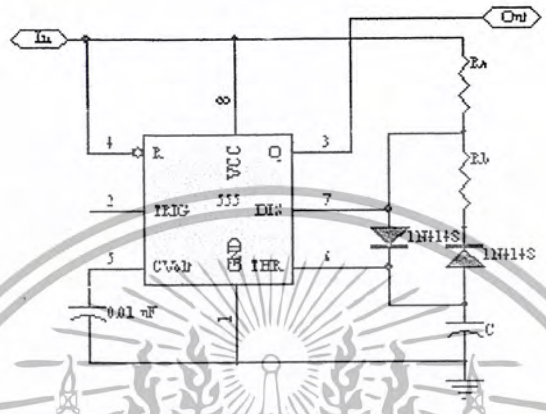
รูปที่ 3.4 วงจรควบคุมการเปิดปิดอุปกรณ์ไฟฟ้า

โดยทำการต่อเข้าทั้ง8บิตของ PORTB

3.5 ส่วนภาคส่งสัญญาณ

ส่วนนี้ประกอบไปด้วย

- ส่วนของแหล่งกำเนิดสัญญาณความถี่พาหะ เพื่อใช้ในการป้องกันสัญญาณรบกวนที่เกิดขึ้นในตัวรับและตัวส่งของอุปกรณ์ไฟฟ้าทั่วไป อันเนื่องมาจากเราใช้แสงอินฟราเรดในการส่งข้อมูล



รูปที่3.5 วงจรกำเนิดความถี่พาหะ

ระหว่างตัวรับและตัวส่ง ซึ่งแหล่งกำเนิดแสงอินฟราเรดมีอยู่ทั่วไป ทำให้แสงอินฟราเรดจากภายนอกระบบเข้ามารบกวนได้ ดังนั้นทางบริษัทที่ทำการสร้างอุปกรณ์ไฟฟ้าที่ใช้รีโมทคอนโทรลอินฟราเรดควบคุมจึงตั้งค่าความถี่พาหะเพื่อทำการมอดูเลตกับสัญญาณข้อมูลที่ต้องการส่ง และสร้างตัวรับสัญญาณที่สามารถรองรับการรับค่าได้เฉพาะความถี่นั้นเพื่อไม่ให้มีสัญญาณรบกวนผ่านเข้ามารวมกับข้อมูลได้ ในที่นี้ใช้วงจร 555 สร้างสัญญาณความถี่พาหะ 36 kHz. (ใช้ใน โปรโตคอลของ Sony) และ 40 kHz. (ใช้ใน โปรโตคอลของ Phillips) ดังรูป

โดยคำนวณ

$$T = T_1 + T_2 \qquad ; T_1 = 0.67 \times R_a \times C \, , T_2 = 0.67 \times R_b \times C$$

$$F = \frac{1}{T} = \frac{1.49}{(R_a + R_b)C}$$

$$Duty\ Cycle = \frac{T_1}{T_2}$$

T_1 : ค่าเวลาที่ส่งสัญญาณในเวลา 1 คาบ (pulse)

T_2 : ค่าเวลาที่ไม่มีสัญญาณในเวลา 1 คาบ (pause)

T : คาบของสัญญาณเอาต์พุต

F : ความถี่ของสัญญาณเอาต์พุต

Duty Cycle : ค่าอัตราส่วนของเวลา T_1 กับ T_2 (pulse / pause)

ที่ $Duty\ Cycle = \frac{1}{3}$:

ความถี่พาหะที่ $36k\Omega$ ใช้ $R_a = 13.36k\Omega,\ R_b = 26.72k\Omega,\ C = 1nF$

ความถี่พาหะที่ $40k\Omega$ ใช้ $R_a = 12.02k\Omega,\ R_b = 24.05k\Omega,\ C = 1nF$

ที่ $Duty\ Cycle = \frac{1}{4}$:

ความถี่พาหะที่ $36k\Omega$ ใช้ $R_a = 10.02k\Omega,\ R_b = 30.06k\Omega,\ C = 1nF$

ความถี่พาหะที่ $40k\Omega$ ใช้ $R_a = 9.014k\Omega,\ R_b = 27.05k\Omega,\ C = 1nF$

- ส่วนตัวส่งสัญญาณอินฟราเรด ทำการรับค่าข้อมูลจากแหล่งกำเนิดสัญญาณความถี่พาหะ เพื่อทำการส่งออกไปที่ตัวรับสัญญาณอินฟราเรดของอุปกรณ์ไฟฟ้าที่เราต้องการควบคุมโดยใช้วงจรดังรูป

รูปที่ 3.6 วงจรส่งสัญญาณอินฟราเรด

3.6 ส่วนประมวลผล

ในส่วนประมวลผลนี้ เป็นการออกแบบโปรแกรมให้ไมโครคอนโทรลเลอร์PIC สามารถรับค่าสัญญาณอินฟราเรดจากรีโมทอินฟราเรดใน 2 รูปแบบ คือแบบBi-Phase และแบบ Pulse Width จากนั้นนำข้อมูลที่ได้จากการรับสัญญาณมาประมวลผล เพื่อส่งสัญญาณเอาท์พุตในแบบควบคุมการเปิดปิดอุปกรณ์ไฟฟ้า หรือการส่งสัญญาณเอาท์พุตเป็นอินฟราเรดในแบบที่กำหนดไว้ออกไปยังอุปกรณ์ไฟฟ้าที่กำหนดไว้

การออกแบบโปรแกรมมี 3 ส่วนคือ

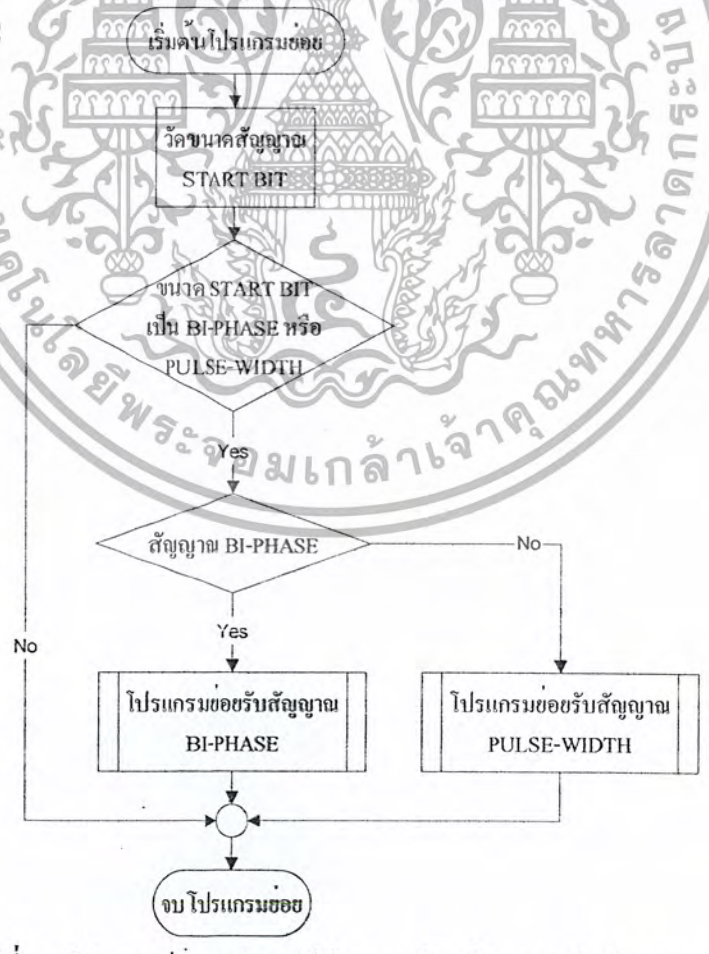
- โปรแกรมรับสัญญาณอินฟราเรด
- โปรแกรมประมวลผล
- โปรแกรมเอาท์พุต ในแบบการควบคุมการเปิดปิดอุปกรณ์ และการส่งสัญญาณอินฟราเรด

3.6.1 โปรแกรมรับสัญญาณอินฟราเรด

สัญญาณอินฟราเรดที่รับเข้ามานั้น จะใช้IRM 8601S แล้วต่อสัญญาณอินพุทเข้ากับขาRA2 ของPIC16F877 ซึ่งมีโปรแกรมการรับข้อมูลBI-PHASE และ PULSE-WIDTH ดังรูปโฟลวชาร์ตที่ 5.1 โดยการทำงานของโปรแกรมรับสัญญาณอินฟราเรดเริ่มจาก การวัดสัญญาณ Start Bit ที่ถูกส่งมาว่ามีขนาดเท่าไร แล้วนำมาประมวลผลว่าเป็นสัญญาณแบบ BI-PHASE หรือ PULSE-WIDTH แล้วเข้าสู่โปรแกรมย่อยการรับสัญญาณในแบบนั้นๆ โดยการตรวจสอบในการรับสัญญาณใช้การแสดงค่าในLCD

สำหรับโปรแกรมตรวจสอบ และประมวลผลค่าสัญญาณในแบบ BI-PHASE โปรแกรมจะทำการสุ่มค่าสัญญาณทุกๆ 800 uesc. แล้วเปรียบเทียบระดับสัญญาณของทั้งสองครั้ง แล้วนำมาประมวลผลแล้วนำเก็บไว้ในรีจิสเตอร์ADDRESS และCOMMAND

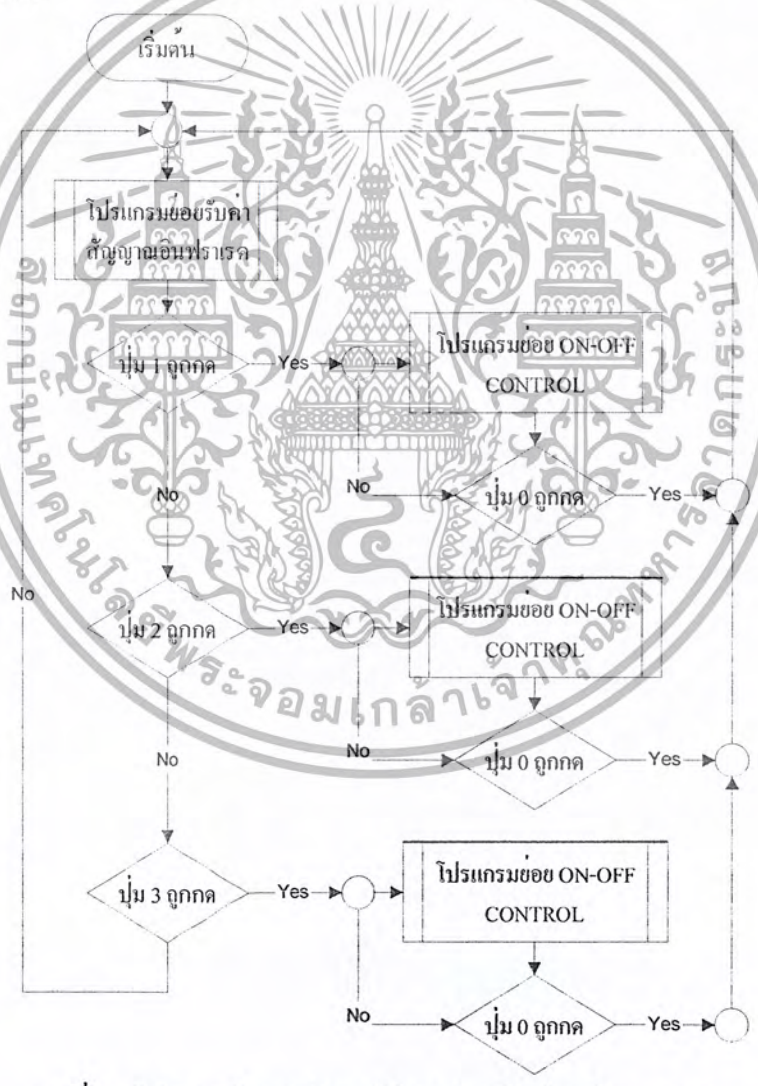
โปรแกรมตรวจสอบ และประมวลผลค่าสัญญาณในแบบ PULSE-WIDTH โปรแกรมจะทำการวัดความกว้างของสัญญาณลอจิก0 แล้วนำมาประมวลผลจากความกว้างของสัญญาณ แล้วเก็บไว้ในรีจิสเตอร์



รูปที่ 3.7 โฟลวชาร์ตการทำงานโปรแกรมย่อยส่วนภาครับสัญญาณ

3.6.2 โปรแกรมประมวลผล

ส่วนโปรแกรมประมวลผล หลังจากที่รีจิสเตอร์เก็บค่าสัญญาณอินฟราเรดจากส่วนภาครับ นำค่าที่เก็บไว้มาจัดลำดับการทำงานของค่าปุ่มกดต่างๆ โดยเริ่มจากการเปิดเครื่องทำงาน ต้องมีการกำหนดค่าสัญญาณเอาท์พุตจากผู้ใช้ โดยเลือกการทำงานได้3แบบคือ การควบคุมการทำงานแบบเปิด-ปิด, การส่งสัญญาณอินฟราเรดแบบBI-PHASE หรือการส่งสัญญาณอินฟราเรดแบบ PULSE WIDTH หลังจากนั้นไมโครคอนโทรลเลอร์จะทำงานในลักษณะที่เลือกสัญญาณเอาท์พุตนั้นๆใน และปุ่มที่สามารถเปลี่ยนแปลงค่าเอาท์พุตได้ตามผู้ใช้ต้องการ โดยการทำงานของโปรแกรมประมวลผลนั้น มีการแสดงสถานการณ์ทำงานว่าขณะนั้นทำงานอยู่ส่วนใด โดยการใช้ LCD มาแสดงผลการทำงาน



รูปที่ 3.8 โฟลวชาร์ตการทำงานโปรแกรมหลักส่วนประมวลผล

โดยส่วนของโปรแกรมย่อยในส่วนของโปรแกรมหลัก มีโปรแกรมย่อยในการรับสัญญาณอินฟราเรดอยู่ภายในโปรแกรมย่อย โดยส่วนของการควบคุมการเปิดปิดอุปกรณ์ไฟฟ้า เมื่อมีการกดครั้งแรกจะเป็นการเปิดอุปกรณ์ไฟฟ้า เมื่อมีการกดครั้งต่อไปเป็นการปิดอุปกรณ์ไฟฟ้าสลับไปเรื่อยๆ

การทำงานมีปุ่มที่สามารถกลับไปเริ่มต้นการเลือกการทำงานโดยการกดปุ่ม0ที่รีโมท จะสามารถเปลี่ยนการส่งสัญญาณเอาท์พุตได้ สำหรับโปรแกรมการส่งสัญญาณอินฟราเรดนั้นจะกล่าวในหัวข้อถัดไป

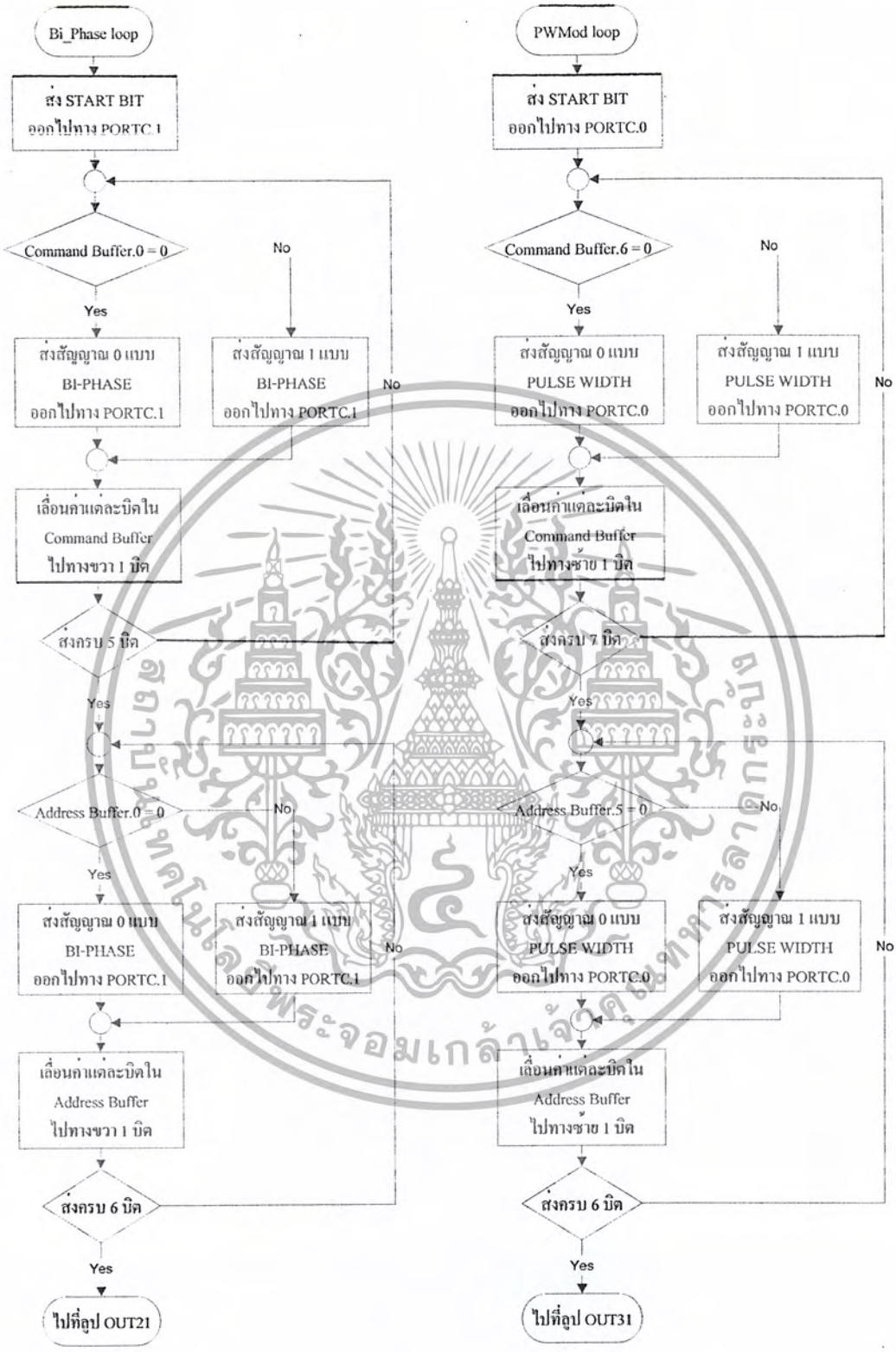
3.6.3โปรแกรมส่งสัญญาณอินฟราเรด

ในการส่งสัญญาณอินฟราเรดเป็นค่าแอดเดรสและคำสั่งออกไปควบคุมอุปกรณ์ สามารถส่งรูปแบบของสัญญาณได้ 2 แบบ คือ Bi-phase modulation และ Pulse-width modulation ดังนี้

Bi-phase modulation : ส่งสัญญาณออกมาทางขา RC0 ของพอร์ต C ออกไปที่วงจรภาคส่งสัญญาณ ค่าข้อมูลที่ต้องการส่งออกไปนั้นเป็น 0 จะส่งข้อมูลเป็น 10 และ 1 ส่ง 01 ออกไปโดยมีคาบของการส่ง 10 และ 01 เป็น 900 μs.

Pulse-width modulation : ส่งสัญญาณออกมาทางขา RC1 ของพอร์ต C โดยจะส่งข้อมูลเป็น 10 ทั้งหมดไม่ว่าข้อมูลจะเป็น 1 หรือ 0

ในการส่งสัญญาณทั้งสองแบบทำได้โดยการที่ภาคประมวลผลว่าคีย์ของรีโมทที่ถูกกดนั้นเป็นคีย์ใดและต้องการส่งค่าคำสั่งในโปรโตคอลใด เพื่อทำการเลือกรูปแบบและข้อมูลในการส่ง เป็นการเปลี่ยนโปรโตคอลในการส่ง โดยค่าข้อมูลเหล่านั้นถูกกำหนดไว้ที่ส่วนเริ่มต้นของโปรแกรม เมื่อมีการประมวลผลว่าต้องการส่งค่าข้อมูลเท่าไหร่ (เช่น %00000000) แบบใด (Bi-phase หรือ Pulse-width modulation) โปรแกรมจะกระโดดไปทำงานที่ลูป Bi_Phase หรือ ลูป PWMod เพื่อทำการส่งข้อมูลออกทางขา RC0 (Pulse width modulation) หรือ RC1 (Bi-phase modulation) โดยมีขั้นตอนการทำงานดังโฟลวชาร์ทต่อไปนี้



รูปที่ 3.9 (ก) โฟลวชาร์ตการทำงานโปรแกรมย่อย Bi_Phase

รูปที่ 3.9 (ข) โฟลวชาร์ตการทำงานโปรแกรมย่อย PWMod

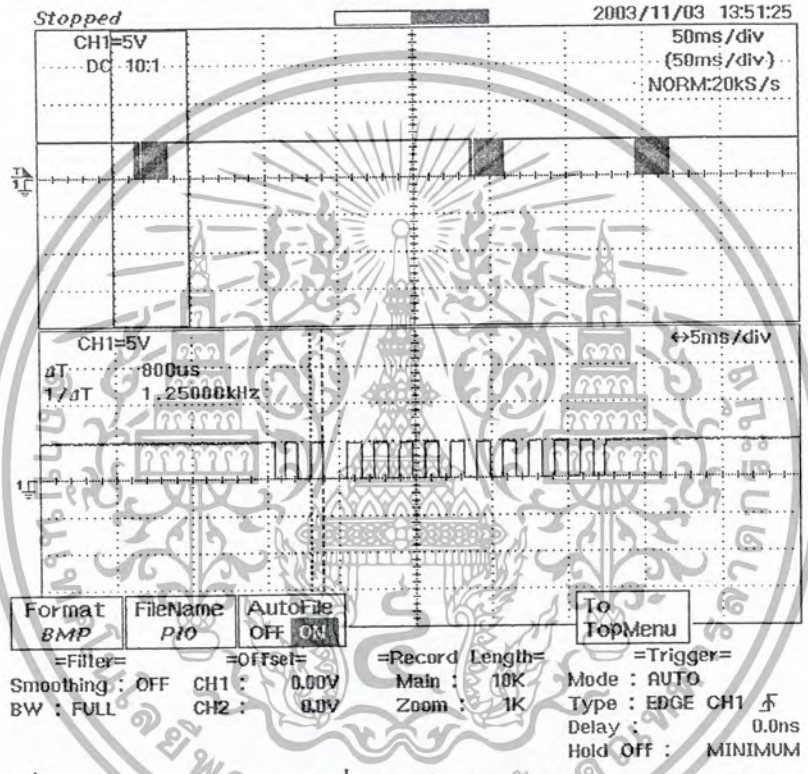
บทที่ 4

ผลการทดลอง

4.1 ด้านภาครับสัญญาณ

สัญญาณอินฟราเรดที่รับเข้ามานั้น จะรับโดยใช้ IRM 8601S แล้วต่อสัญญาณอินพุทเข้ากับขา RA2 ของ PIC16F877 ซึ่งมีโปรแกรมการรับข้อมูล BI-PHASE และ PULSE WIDTH

ทำการวัดสัญญาณโดยวัดที่ขาเอาท์พุตของตัวรับสัญญาณซึ่งสามารถจับสัญญาณได้ดังรูปที่ 4.1 ในการกดปุ่มค่าต่างๆ แล้วบันทึกค่าเพื่อเป็นข้อมูลในการประมวลผล



รูปที่4.1 สัญญาณอินฟราเรดวัดที่ขาเอาต์พุตของ IRM 8601S โดยดิจิตอลสโคป

4.2 ส่วนโปรแกรมประมวลผล

จากที่ได้ทำการทดลองรับค่าและเขียนโปรแกรมรับข้อมูล ส่วนภาครับสามารถรับสัญญาณอินฟราเรดได้ทั้งสองแบบ โดยไมโครคอนโทรลเลอร์สามารถทราบได้เองว่าสัญญาณที่ส่งเข้ามานั้นเป็นสัญญาณแบบใด เนื่องจากลักษณะของ Start Bit ของแต่ละโปรโตคอล หลังจากที่ทราบลักษณะการส่ง ไมโครคอนโทรลเลอร์จะเรียกโปรแกรมย่อยในการรับค่าแบบต่าง และเก็บข้อมูล ADDRESS และ COMMAND ไว้ในรีจิสเตอร์

ในการทดลองในครั้งแรก จะทำการทดสอบโดยการแสดงผลค่าที่ไมโครคอนโทรลเลอร์เก็บไว้ในรีจิสเตอร์มาแสดงผลทาง LCD ทำให้ทราบว่าไมโครคอนโทรลเลอร์ประมวลผลถูกหรือไม่

ในการทดลองจะใช้การแสดงผลของ LCD ในการตรวจสอบโปรแกรมว่าสามารถทำงานในส่วนใดได้บ้างโดยใช้การกำหนดการแสดงผลออกมาทาง LCD ซึ่งโปรแกรมสามารถประมวลผลให้ค่าเอาท์พุทที่ต้องการออกมาได้ตามที่กำหนด

ผลการทดลอง

- การทดลองกดสัญญาณรีโมททีวีแบบ BI-PHASE แสดงผลทาง LCD

KEY	ADDRESS	COMMAND	KEY	ADDRESS	COMMAND
0	0	0	8	0	8
1	0	1	9	0	9
2	0	2	VOL+	0	16
3	0	3	VOL -	0	17
4	0	4	CH+	0	32
5	0	5	CH-	0	33
6	0	6	POWER	0	12
7	0	7			

- การทดลองกดสัญญาณรีโมททีวีแบบ PULSE WIDTH แสดงผลทาง LCD

KEY	ADDRESS	COMMAND	KEY	ADDRESS	COMMAND
0	1	0	8	1	8
1	1	1	9	1	9
2	1	2	VOL+	1	18
3	1	3	VOL -	1	19
4	1	4	CH+	1	16
5	1	5	CH-	1	17
6	1	6	POWER	1	21
7	1	7			

เมื่อสามารถโปรแกรมให้ไมโครคอนโทรลเลอร์สามารถรับและเก็บค่าจากสัญญาณอินฟราเรด ได้ทำการทดลองโปรแกรมให้ไมโครคอนโทรลเลอร์สามารถกำหนดค่าเอาท์พุตตามที่ผู้ใช้ต้องการ โดยอาศัยการเปรียบเทียบกับค่าที่วัดได้ ว่ามีการกดปุ่มใดเกิดขึ้นและทำการวนลูปส่งสัญญาณเอาท์พุตในกรณีที่ต้องการ

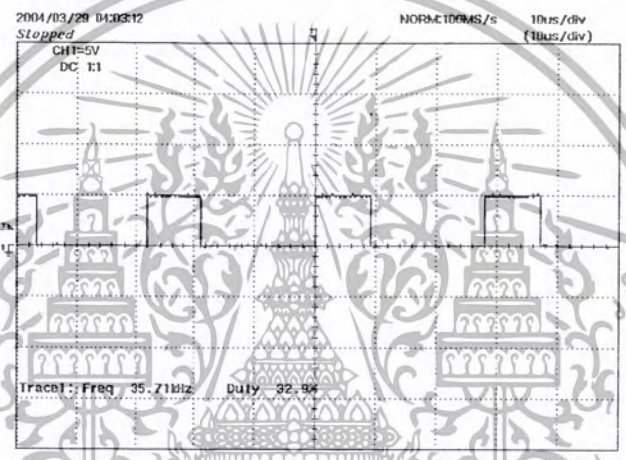
และมีปุ่มที่สามารถกลับมาเลือกสถานะเอาท์พุตใหม่อีกครั้ง โดยสถานะของเอาท์พุตจะแสดงผลทางจอ LCD ว่าขณะนี้ได้ทำงานในสถานะใด การทำงานแบบการควบคุมการเปิดปิดอุปกรณ์ไฟฟ้า เมื่อมีการกดปุ่มจากรีโมทคอนโทรลอินฟราเรดทำให้มีเกิดเปิดปิดอุปกรณ์ไฟฟ้า

4.3 ภาคส่งสัญญาณอินฟราเรด

จากการทดลองส่งสัญญาณความถี่พาหะด้วย IC 555 และส่งสัญญาณอินฟราเรดแล้วรับสัญญาณด้วยตัวรับสัญญาณอินฟราเรดได้ผลดังนี้

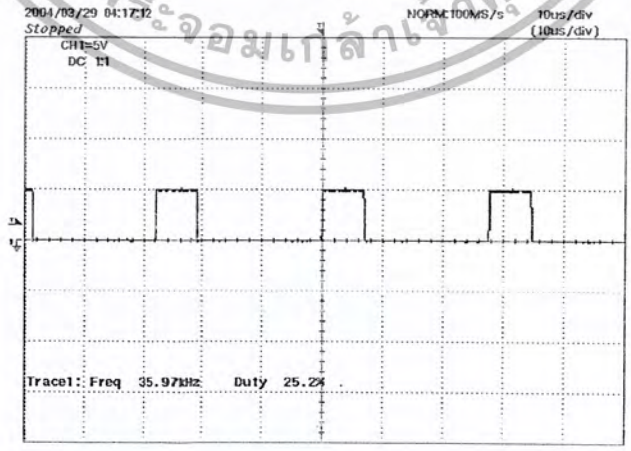
ผลการทดลอง

- ความถี่พาหะ 36 kHz. Duty Cycle เท่ากับ $\frac{1}{3}$



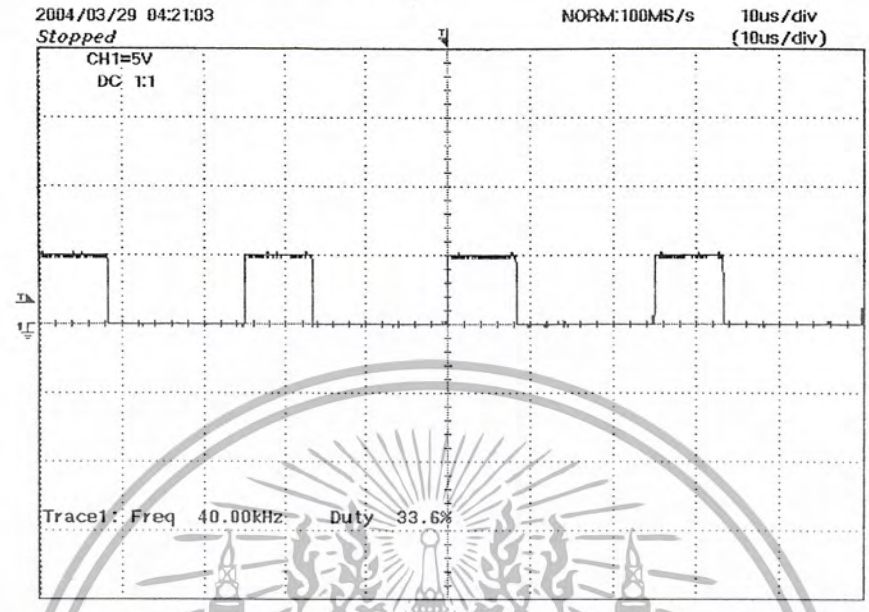
รูปที่4.2 สัญญาณวัดที่ขาเอาต์พุต (pin 3) ของ IC 555 ที่ ความถี่พาหะ 36 kHz. Duty cycle เท่ากับ 33.33%

- ความถี่พาหะ 36 kHz. Duty Cycle เท่ากับ $\frac{1}{4}$



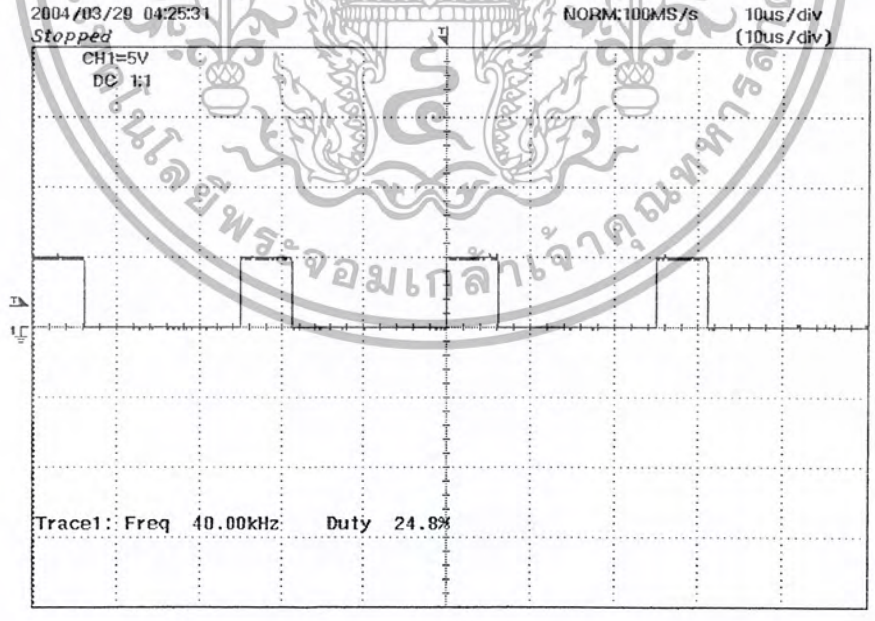
รูปที่4.3 สัญญาณวัดที่ขาเอาต์พุต (pin 3) ของ IC 555 ที่ ความถี่พาหะ 36 kHz. Duty cycle เท่ากับ 25%

- ความถี่พาหะ 40 kHz. Duty Cycle เท่ากับ $\frac{1}{3}$



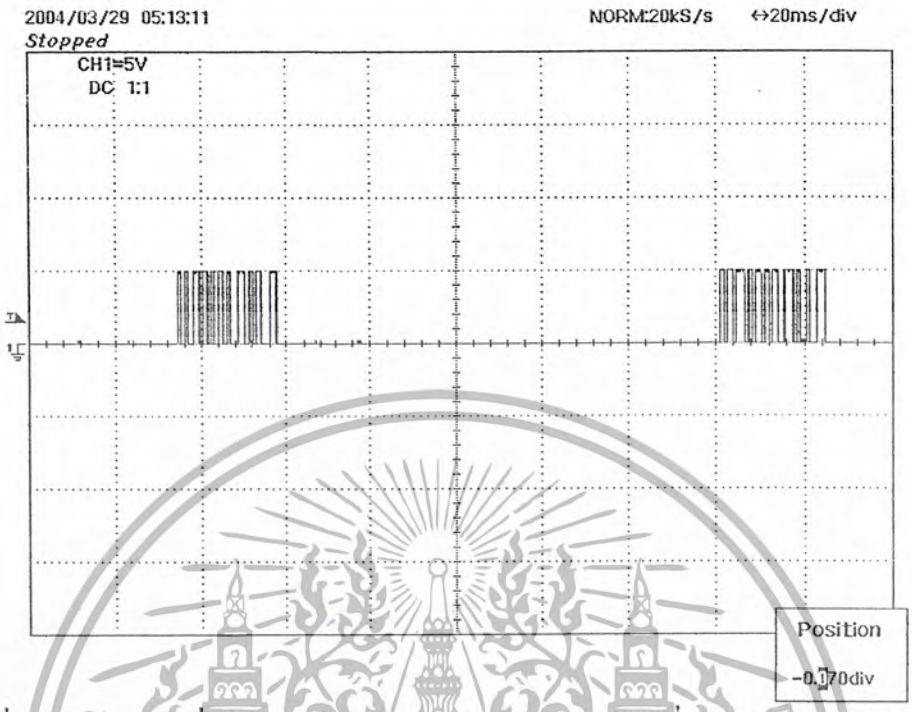
รูปที่4.4 สัญญาณวัดที่ขาเอาต์พุต (pin 3) ของ IC 555 ที่ ความถี่พาหะ 40 kHz. Duty cycle เท่ากับ 33.33%

- ความถี่พาหะ 40 kHz. Duty Cycle เท่ากับ $\frac{1}{4}$

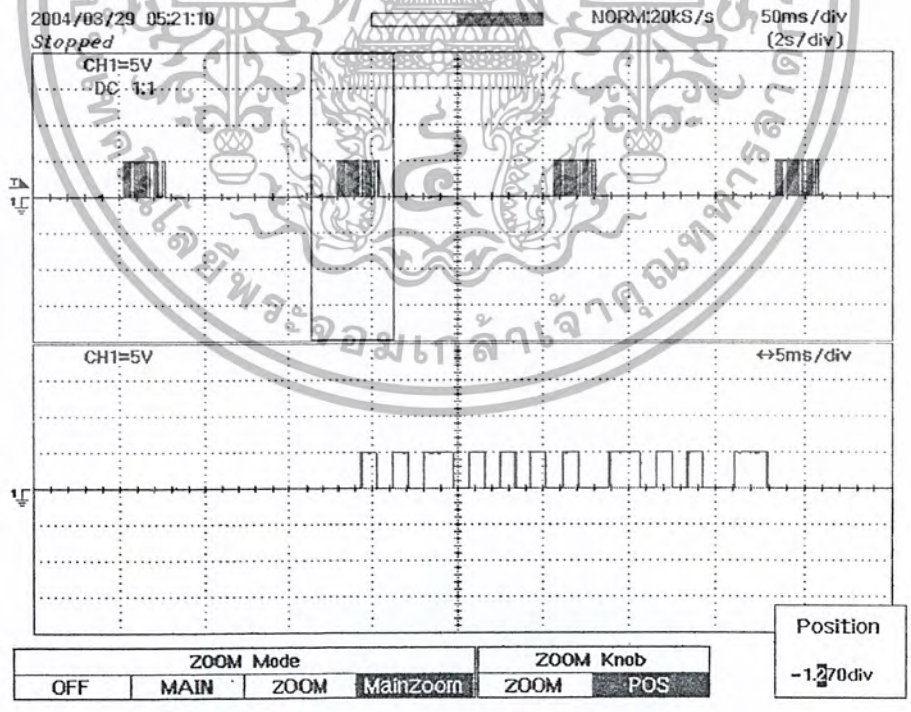


รูปที่4.5 สัญญาณวัดที่ขาเอาต์พุต (pin 3) ของ IC 555 ที่ ความถี่พาหะ 40 kHz. Duty cycle เท่ากับ 25%

- ผลสัญญาณอินฟราเรดส่งออกแบบ Bi-phase เมื่อกด CH-

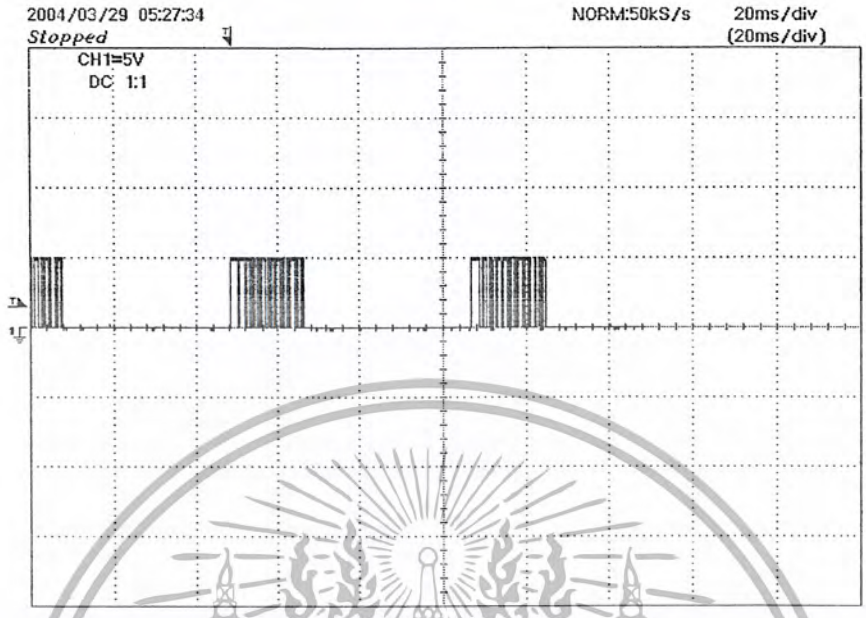


รูปที่4.6 สัญญาณวัดที่ขาเอาท์พุตของตัวรับสัญญาณอินฟราเรดเมื่อกด CH- แบบ Bi-phase

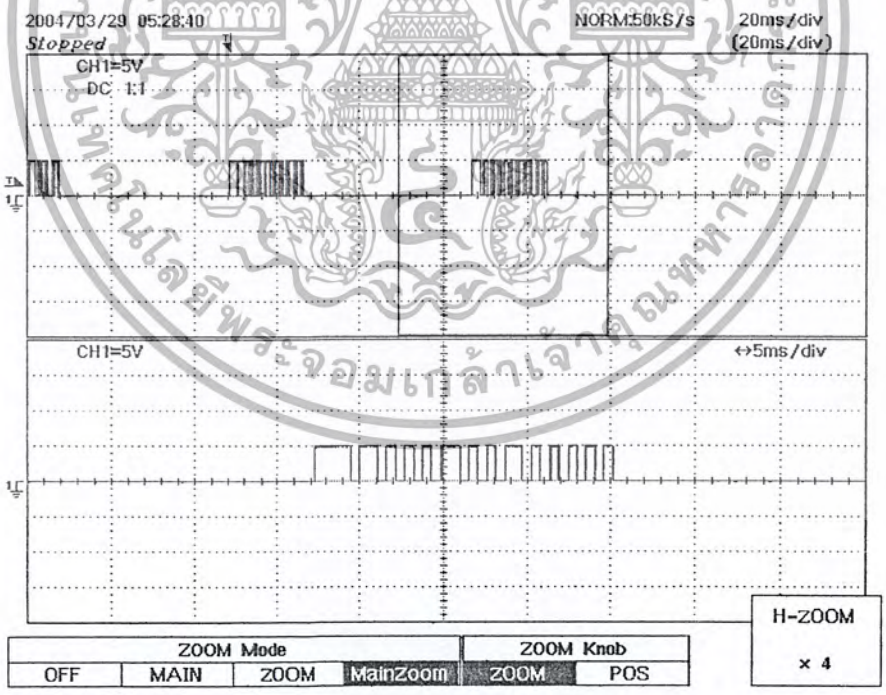


รูปที่4.7 สัญญาณวัดที่ขาเอาท์พุตของตัวรับสัญญาณอินฟราเรด เมื่อกด CH- แบบ Bi-phase

- ผลสัญญาณอินฟราเรดส่งออกแบบ Pulse width เมื่อกด CH-



รูปที่4.8 สัญญาณวัดที่ขาเอาท์พุตของตัวรับสัญญาณอินฟราเรด เมื่อกด CH- แบบ **Pulse-width modulation**



รูปที่4.9 สัญญาณวัดที่ขาเอาท์พุตของตัวรับสัญญาณอินฟราเรด เมื่อกด CH- แบบ **Pulse-width modulation**

จากการทดลองส่งสัญญาณจะเห็นได้ว่าสัญญาณทั้งแบบ Bi-phase modulation และ Pulse-width modulation มีการส่งค่าสัญญาณออกมาเป็น 0 และ 1 ตามค่า command และ address ตามตารางในบทที่ 2

บทที่ 5
บทสรุปและวิจารณ์

5.1 สรุปผลการทดลอง

จากการศึกษาการรับส่งสัญญาณอินฟราเรดในรูปแบบต่างๆ และได้ทำการออกแบบและสร้างอุปกรณ์รับส่งสัญญาณ และเขียนโปรแกรมการรับส่งข้อมูลในแบบ BI PHASE และ PULSE WIDTH ตามการออกแบบที่กำหนดไว้ ทำให้สามารถรับค่าสัญญาณอินฟราเรดและประมวลผลสัญญาณ นำมาใช้ประโยชน์ในการควบคุมอุปกรณ์ไฟฟ้า ทั้งแบบการควบคุมการเปิดปิดอุปกรณ์และการส่งสัญญาณอินฟราเรดออกไป แม้ว่าสัญญาณอินฟราเรดจะมีความเร็วสูงมาก แต่เนื่องจากการส่งสัญญาณซ้ำ ทำให้สามารถนำสัญญาณช่วงหนึ่งมาประมวลผลได้ สำหรับการส่งสัญญาณอินฟราเรดออกไปนั้นต้องส่งสัญญาณให้มีค่าความถี่พาหะตรงกันกับช่วงที่อุปกรณ์ไฟฟ้านั้นๆรับได้ อุปกรณ์ไฟฟ้าจึงจะรับรู้ว่ามีคำสั่งให้ทำงาน

5.2 วิจารณ์ผลการทดลอง

เนื่องจากสัญญาณอินฟราเรดมีความเร็วสูงมาก ทำให้วัดสัญญาณอินฟราเรดได้ยากต้องใช้การเขียนโปรแกรมวนลูปช่วยในการตรวจสอบสัญญาณซึ่งจะวัดค่าได้เพียงชั่วขณะหนึ่งเท่านั้น ทำให้ยากต่อการตรวจสอบค่าสัญญาณในพัลซ์ต่อๆไป แต่เนื่องจากมีการวัดสัญญาณไว้ล่วงหน้าสามารถทราบลักษณะสัญญาณก่อน แล้วนำมาเขียนโปรแกรมประมวลผลจึงทำให้สามารถตรวจสอบได้ระดับหนึ่ง แต่บางครั้งยังคงมีข้อผิดพลาดอยู่แต่ไม่สามารถแก้ข้อผิดพลาดโดยทันทีได้ทำให้ต้องแก้ไขโดยการสนใจสัญญาณในชุดต่อไป และสัญญาณที่ต้องการส่งออกไปควบคุมอุปกรณ์นั้น ในส่วนของการส่งสัญญาณความถี่พาหะต้องส่งสัญญาณด้วยค่าความถี่ที่แตกต่างกันไปตามอุปกรณ์ไฟฟ้าและผู้ผลิต ทำให้มีข้อจำกัดในการสร้างวงจรกำเนิดสัญญาณความถี่พาหะให้เพียงพอกับความถี่พาหะที่ใช้กันอยู่ตามท้องตลาด

ภาคผนวก

```
' PicBasic Pro program to

' Define LOADER_USED to allow use of the boot loader.
Include "modedefs.bas"
Define LOADER_USED 1
Define OSC 20
' Define LCD registers and bits
Define LCD_DREG PORTD
Define LCD_DBIT 4
Define LCD_RSREG PORTE
Define LCD_RSBIT 1
Define LCD_EREG PORTE
Define LCD_EBIT 0


W0 var word
P_val var word
ch1 var bit
ch2 var bit
add var word
add1 var word
com1 var word
com2 var word
t var byte
pb var byte
Bi_add VAR BYTE

Bi_com_dch VAR BYTE
Bi_com_uch VAR BYTE
Bi_com_dvo VAR BYTE
```

```
Bi_com_uvo      VAR BYTE
Bi_com_pow      VAR BYTE
Bi_com_one      VAR BYTE
Bi_com_two      VAR BYTE
Bi_com_thr      VAR BYTE
Bi_com_fou      VAR BYTE
Bi_com_fiv      VAR BYTE
Bi_com_six      VAR BYTE
Bi_com_sev      VAR BYTE
Bi_com_eig      VAR BYTE
Bi_com_nin      VAR BYTE

PW_add          VAR BYTE

PW_com_dch      VAR BYTE
PW_com_uch      VAR BYTE
PW_com_dvo      VAR BYTE
PW_com_uvo      VAR BYTE
PW_com_pow      VAR BYTE
PW_com_one      VAR BYTE
PW_com_two      VAR BYTE
PW_com_thr      VAR BYTE
PW_com_fou      VAR BYTE
PW_com_fiv      VAR BYTE
PW_com_six      VAR BYTE
PW_com_sev      VAR BYTE
PW_com_eig      VAR BYTE
PW_com_nin      VAR BYTE

Per_Bi_H        CON 900
```

```
Per_Bi_L      CON 900

Per_PW_H1     CON 1200
Per_PW_H0     CON 600
Per_PW_L      CON 600

Buff_Add      VAR BYTE
Buff_Com      VAR BYTE

Z1            VAR BYTE
Z2            VAR BYTE
Z3            VAR BYTE
Z4            VAR BYTE
Z5            VAR BYTE

    TRISC = %11111100
    Bi_com_dch = %00010001
    Bi_com_uch = %00010000
    Bi_com_dvo = %00110010
    Bi_com_uvo = %00110011
    Bi_com_pow = %00001010
    Bi_com_one = %00000001
    Bi_com_two = %00000010
    Bi_com_thr = %00000011
    Bi_com_fou = %00000100
    Bi_com_fiv = %00000101
    Bi_com_six = %00000110
    Bi_com_sev = %00000111
    Bi_com_eig = %00001000
    Bi_com_nin = %00001001
```

```
Bi_add = %00000000


PW_com_dch = %00010001
PW_com_uch = %00010000
PW_com_dvo = %00010001
PW_com_uvo = %00010010
PW_com_pow = %00010101
PW_com_one = %00000001
PW_com_two = %00000010
PW_com_thr = %00000011
PW_com_fou = %00000100
PW_com_fiv = %00000101
PW_com_six = %00000110
PW_com_sev = %00000111
PW_com_eig = %00001000
PW_com_nin = %00001001
PW_Add = %00000001

    Add = 0
    add1 = 0
    com1 = 0
    com2 = 0
ADCON1 = 7      ' Set PORTA and PORTE to digital
    TRISA  = %00001000 'SET PORT RA3=INPUT
    TRISB  = %00000000 'SET PORTB OUTPUT
    PB     = %00000000
    Pause 1000
loop:  Lcdout  $fe,1
    lcdout "Select m"
    Lcdout  $fe,172,"ode1 2 3"
```

```
loop1:  Call Input1

'

loop2:  If (Add = 1) Or (Add = 3) Or (add1 = 1) Then step1
            GoTo loop1


step1:  If (com2 = 0) Or (com1 = 1) Then out1
            If (com2 = 1) Or (com1 = 2) Then out2
            If (com2 = 2) Or (com1 = 3) Then out3
            GoTo loop1


'-------------------------------------------------------------

Input1: PulsIn  porta.3,0,W0            ' Measure pulse (in 2 uSec)
      If (W0 < 350) Or (W0 > 1250) Then Input2
            If (W0 > 350) And (W0 <= 450) Then GoTo rc5
            If (W0 > 1150) And (W0 <= 1250) Then GoTo sony
Input2: Return


sony:   add1 = 0
            com2 = 0
            Add = 255
            com1 = 255
            P_val = 0
p1:           PulsIn  portA.3,0,P_val
            If P_val = 0 Then p1
            If (P_val >= 500) Then
            com2 0# = 1
            Else
            com2 0# = 0
```

```
        End If

        P_val = 0
p2:      PulsIn  portA.3,0,P_val
        If P_val = 0 Then p2
        If (P_val >= 500) Then
        com2 0.1 = 1
        Else
        com2 0.1 = 0
        End If

        P_val = 0
p3:      PulsIn  portA.3,0,P_val
        If P_val = 0 Then p3
        If (P_val >= 500) Then
        com2 0.2 = 1
        Else
        com2 0.2 = 0
        End If

        P_val = 0
p4:      PulsIn  portA.3,0,P_val
        If P_val = 0 Then p4
        If (P_val >= 500) Then
        com2 0.3 = 1
        Else
        com2 0.3 = 0
        End If

        P_val = 0
```

```
p5:     PulsIn  portA.3,0,P_val
        If P_val = 0 Then p5
        If (P_val >= 500) Then
        com2 0.4 = 1
        Else
        com2 0.4 = 0
        End If

        P_val = 0
c6:     PulsIn  portA.3,0,P_val
        If P_val = 0 Then c6
        If (P_val >= 500) Then
        com2 0.5 = 1
        Else
        com2 0.5 = 0
        End If

        P_val = 0
c5:     PulsIn  portA.3,0,P_val
        If P_val = 0 Then c5
        If (P_val >= 500) Then
        com2 0.6 = 1
        Else
        com2 0.6 = 0
        End If

        P_val = 0
c4:     PulsIn  portA.3,0,P_val
        If P_val = 0 Then c4
        If (P_val >= 500) Then
```

```
        add1 0# = 1
        Else
        add1 0# = 0
        End If

        P_val = 0
c3:      PulsIn  portA.3,0,P_val
        If P_val = 0 Then c3
        If (P_val >= 500) Then
        add1 0.1 = 1
        Else
        add1 0.1 = 0
        End If

        P_val = 0
c2:      PulsIn  portA.3,0,P_val
        If P_val = 0 Then c2
        If (P_val >= 500) Then
        add1 0.2 = 1
        Else
        add1 0.2 = 0
        End If

        P_val = 0
c1:      PulsIn  portA.3,0,P_val
        If P_val = 0 Then c1
        If (P_val >= 500) Then
        add1 0.3 = 1
        Else
        add1 0.3 = 0
```

```
            End If
            P_val = 0
c0:          PulsIn  portA.3,0,P_val
            If P_val = 0 Then c0
            If (P_val >= 500) Then
            add1 0.4 = 1
            Else
            add1 0.4 = 0
            End If
            GoTo Input2
'-------------------------------------------------------------

rc5:   Add = 0
            com1 = 0
            add1 = 255
            com2 = 255
            t = 0
            ch1 = porta.3
            Pauseus 1000
            ch2 = porta.3
            If ch1 = ch2 Then
            GoTo loop9
            End If
            If ch1 < ch2 Then
            Add 0# = 0
            Else
            Add 0# = 1
            End If

            ch1 = 0
```

```
ch2 = 0
t = 1
Pauseus 800
ch1 = porta.3
Pauseus 800
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
Add 0.1 = 0
Else
Add 0.1 = 1
End If
ch1 = 0
ch2 = 0
t = 2
Pauseus 900
ch1 = porta.3
Pauseus 800
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
Add 0.2 = 0
Else
Add 0.2 = 1
End If
```

```
ch1 = 0
ch2 = 0
t = 3
Pauseus 800
ch1 = porta.3
Pauseus 800
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
Add 0.3 = 0
Else
Add 0.3 = 1
End If

ch1 = 0
ch2 = 0
t = 4
Pauseus 900
ch1 = porta.3
Pauseus 800
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
Add 0.4 = 0
Else
```

```
Add 0.4 = 1
End If

ch1 = 0
ch2 = 0
t = 5
Pauseus 900
ch1 = porta.3
Pauseus 800
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
Add 0.5 = 0
Else
Add 0.5 = 1
End If

ch1 = 0
ch2 = 0
t = 6
Pauseus 900
ch1 = porta.3
Pauseus 900
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
```

```
Add 0.6 = 0
Else
Add 0.6 = 1
End If

t = 11
Pauseus 900
ch1 = porta.3
Pauseus 900
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
com1 0.5 = 0
Else
com1 0.5 = 1
End If

t = 12
Pauseus 900
ch1 = porta.3
Pauseus 900
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
com1 0.4 = 0
Else
```

```
com1 0.4 = 1
End If

t = 13
Pauseus 900
ch1 = porta.3
Pauseus 900
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
com1 0.3 = 0
Else
com1 0.3 = 1
End If

t = 14
Pauseus 900
ch1 = porta.3
Pauseus 900
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
com1 0.2 = 0
Else
com1 0.2 = 1
End If
```

```
t = 15
Pauseus 900
ch1 = porta.3
Pauseus 900
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
com1 0.1 = 0
Else
com1 0.1 = 1
End If

t = 16
Pauseus 750
ch1 = porta.3
Pauseus 900
ch2 = porta.3
If ch1 = ch2 Then
GoTo loop9
End If
If ch1 < ch2 Then
com1 0# = 0
Else
com1 0# = 1
End If
loop9:  GoTo Input2
'-----------------------------------------------------------------
```

```
out1:  lcdout  $fe,1,"On-Off"
       com1 = 100
       com2 = 100
out10: Pause 1000
out11: com1 = 200
       com2 = 200
       Call Input1
       If (Add = 1) Or (Add = 3) Or (add1 = 1) Then out12
       GoTo out11
out12: If (com1 = 0) Or (com2 = 9) Then Loop
       If (com1 = 1) Or (com2 = 0) Then of1
       If (com1 = 2) Or (com2 = 1) Then of2
       If (com1 = 3) Or (com2 = 2) Then of3
       If (com1 = 4) Or (com2 = 3) Then of4
       If (com1 = 5) Or (com2 = 4) Then of5
       If (com1 = 6) Or (com2 = 5) Then of6
       If (com1 = 7) Or (com2 = 6) Then of7
       If (com1 = 8) Or (com2 = 7) Then of8
       If (com1 = 12) Or (com2 = 21) Then ofpo
       GoTo out10

of1:   lcdout  $fe,1,"On-Off"
       lcdout  $fe,172,"CH1^-^"
       IF  PortB.0 =0   Then
       High PortB.0
       Else
       Low PortB.0
       End If
       GoTo out10
```

```
of2:   lcdout $fe,1,"On-Off"
       lcdout $fe,172,"CH2^^;"
       IF PortB.1 =0 Then
       High PortB.1
       Else
       Low PortB.1
       End If
       GoTo out10
of3:   lcdout $fe,1,"On-Off"
       lcdout $fe,172,"CH3^3^"
       IF PortB.2 =0 Then
       High PortB.2
       Else
       Low PortB.2
       End If
       GoTo out10
of4:   lcdout $fe,1,"On-Off"
       lcdout $fe,172,"CH4- -"
       IF PortB.3 =0 Then
       High PortB.3
       Else
       Low PortB.3
       End If
       GoTo out10
of5:   lcdout $fe,1,"On-Off"
       lcdout $fe,172,"CH5 :p"
       IF PortB.4 =0 Then
       High PortB.4
       Else
       Low PortB.4
```

```
            End If
            GoTo out10
of6:    lcdout  $fe,1,"On-Off"
            lcdout  $fe,172,"CH6 :D"
            IF  PortB.5 =0   Then
            High PortB.5
            Else
            Low PortB.5
            End If
            GoTo out10
of7:    lcdout  $fe,1,"On-Off"
            lcdout  $fe,172,"CH7^^"
            IF  PortB.6 =0   Then
            High PortB.6
            Else
            Low PortB.6
            End If
            GoTo out10
of8:    lcdout  $fe,1,"On-Off"
            lcdout  $fe,172,"CH8^o^"
            IF  PortB.7 =0   Then
            High PortB.7
            Else
            Low PortB.7
            End If
            GoTo out10
ofpo:   lcdout  $fe,1,"On-Off"
            lcdout  $fe,172,"Clear"
            PortB = 0
            GoTo out10
```

```
'-------------------------------------------------------------

out2:  lcdout  $fe,1,"Bi-Phase"
          com1 = 100
          com2 = 100
          Pause 2000
out21:  Call Input1
          If (Add = 1) Or (Add = 3) Or (add1 = 1) Then out22
          GoTo out21
out22:  If (com1 = 0) Or (com2 = 9) Then Loop
          If (com1 = 1) Or (com2 = 0) Then Bi1
          If (com1 = 2) Or (com2 = 1) Then Bi2
          If (com1 = 3) Or (com2 = 2) Then Bi3
          If (com1 = 4) Or (com2 = 3) Then Bi4
          If (com1 = 5) Or (com2 = 4) Then Bi5
          If (com1 = 6) Or (com2 = 5) Then Bi6
          If (com1 = 7) Or (com2 = 6) Then Bi7
          If (com1 = 8) Or (com2 = 7) Then Bi8
          If (com1 = 9) Or (com2 = 8) Then Bi9
          If (com1 = 32) Or (com2 = 16) Then Bichp
          If (com1 = 33) Or (com2 = 17) Then Bichd
          If (com1 = 16) Or (com2 = 18) Then Bivop
          If (com1 = 17) Or (com2 = 19) Then Bivod
          If (com1 = 12) Or (com2 = 21) Then Bipo
 GoTo out21


 Bi1:   lcdout  $fe,172," CH1  "
       Buff_Com = Bi_com_one
       GoSub Bi_Phase
          GoTo out21
 Bi2:   lcdout  $fe,172," CH2  "
```

```
        Buff_Com = Bi_com_two
        GoSub Bi_Phase
            GoTo out21
Bi3:    lcdout $fe,172," CH3 "
        Buff_Com = Bi_com_thr
        GoSub Bi_Phase
            GoTo out21
Bi4:    lcdout $fe,172," CH4 "
        Buff_Com = Bi_com_fou
        GoSub Bi_Phase
            GoTo out21
Bi5:    lcdout $fe,172," CH5 "
        Buff_Com = Bi_com_fiv
        GoSub Bi_Phase
            GoTo out21
Bi6:    lcdout $fe,172," CH6 "
        Buff_Com = Bi_com_six
        GoSub Bi_Phase
            GoTo out21
Bi7:    lcdout $fe,172," CH7 "
        Buff_Com = Bi_com_sev
        GoSub Bi_Phase
            GoTo out21
Bi8:    lcdout $fe,172," CH8 "
        Buff_Com = Bi_com_eig
        GoSub Bi_Phase
            GoTo out21
Bi9:    lcdout $fe,172," CH9 "
        Buff_Com = Bi_com_nin
        GoSub Bi_Phase
```

```
            GoTo out21
Bichp:  lcdout  $fe,172,"  CH+  "
        Buff_Com = Bi_com_uch
        GoSub Bi_Phase
            GoTo out21
Bichd:  lcdout  $fe,172,"  CH-  "
        Buff_Com = Bi_com_dch
        GoSub Bi_Phase
            GoTo out21
Bivop:  lcdout  $fe,172,"  VOL+ "
        Buff_Com = Bi_com_uvo
        GoSub Bi_Phase
            GoTo out21
Bivod:  lcdout  $fe,172,"  VOL- "
        Buff_Com = Bi_com_dvo
        GoSub Bi_Phase
            GoTo out21
Bipo:   lcdout  $fe,172,"  Power"
        Buff_Com = Bi_com_pow
        GoSub Bi_Phase
            GoTo out21
'-------------------------------------------------------------
out3:   lcdout  $fe,1,"Pulse wi"
            lcdout  $fe,172,"dth"
            com1 = 100
            com2 = 100
            Pause 2000
out31:  Call Input1
            If (Add = 1) Or (Add = 3) Or (add1 = 1) Then out32
            GoTo out31
```

```
out32:  If (com1 = 0) Or (com2 = 9) Then Loop
        If (com1 = 1) Or (com2 = 0) Then Pw1
        If (com1 = 2) Or (com2 = 1) Then Pw2
        If (com1 = 3) Or (com2 = 2) Then Pw3
        If (com1 = 4) Or (com2 = 3) Then Pw4
        If (com1 = 5) Or (com2 = 4) Then Pw5
        If (com1 = 6) Or (com2 = 5) Then Pw6
        If (com1 = 7) Or (com2 = 6) Then Pw7
        If (com1 = 8) Or (com2 = 7) Then Pw8
        If (com1 = 9) Or (com2 = 8) Then Pw9
        If (com1 = 32) Or (com2 = 16) Then Pwchp
        If (com1 = 33) Or (com2 = 17) Then Pwchd
        If (com1 = 16) Or (com2 = 18) Then Pwvop
        If (com1 = 17) Or (com2 = 19) Then Pwvod
        If (com1 = 12) Or (com2 = 21) Then Pwpo

        GoTo out31

Pw1:    lcdout $fe,172,"dth CH1 "
    Buff_Com = PW_com_one
    GoSub PWMod
        GoTo out31
Pw2:    lcdout $fe,172,"dth CH2 "
    Buff_Com = PW_com_two
    GoSub PWMod
        GoTo out31
Pw3:    lcdout $fe,172,"dth CH3 "
    Buff_Com = PW_com_thr
    GoSub PWMod
        GoTo out31
```

```
Pw4:   lcdout  $fe,172,"dth CH4 "
       Buff_Com = PW_com_fou
       GoSub PWMod
            GoTo out31
Pw5:   lcdout  $fe,172,"dth CH5 "
       Buff_Com = PW_com_fiv
       GoSub PWMod
            GoTo out31
Pw6:   lcdout  $fe,172,"dth CH6 "
       Buff_Com = PW_com_six
       GoSub PWMod
            GoTo out31
Pw7:   lcdout  $fe,172,"dth CH7 "
       Buff_Com = PW_com_sev
       GoSub PWMod
            GoTo out31
Pw8:   lcdout  $fe,172,"dth CH8 "
       Buff_Com = PW_com_eig
       GoSub PWMod
            GoTo out31
Pw9:   lcdout  $fe,172,"dth CH9 "
       Buff_Com = PW_com_nin
       GoSub PWMod
            GoTo out31
Pwchp: lcdout  $fe,172,"dth CH+ "
       Buff_Com = PW_com_uch
       GoSub PWMod
            GoTo out31
Pwchd: lcdout  $fe,172,"dth CH- "
       Buff_Com = PW_com_dch
```

```
        GoSub PWMod
            GoTo out31
Pwvop:  lcdout  $fe,172,"dth VOL+"
        Buff_Com = PW_com_uvo
        GoSub PWMod
            GoTo out31
Pwvod:  lcdout  $fe,172,"dth VOL-"
        Buff_Com = PW_com_dvo
        GoSub PWMod
            GoTo out31
Pwpo:   lcdout  $fe,172,"dth Pow "
        Buff_Com = PW_com_pow
        GoSub PWMod
            GoTo out31
]
PWMod:
        High    PORTC.0
        Pause 2400
        Low     PORTC.0
        Pause 600

        For Z4 = 1 To 7
          If Buff_Com.0 = 0  Then
            GoSub Zero_PW
          Else
            GoSub One_PW
          End If
          Buff_Com.0 = Buff_Com.1
          Buff_Com.1 = Buff_Com.2
          Buff_Com.2 = Buff_Com.3
```

```
        Buff_Com.3 = Buff_Com.4
        Buff_Com.4 = Buff_Com.5
        Buff_Com.5 = Buff_Com.6
    Next Z4
    For Z5 = 1 To 5
        If Buff_Add.0 = 0  Then
            GoSub Zero_PW
        Else
            GoSub One_PW
        End If
        Buff_Add.0 = Buff_Add.1
        Buff_Add.1 = Buff_Add.2
        Buff_Add.2 = Buff_Add.3
        Buff_Add.3 = Buff_Add.4
    Next Z5
        PORTC=%00000000
Return
One_PW:
    High   PORTC.0
    Pauseus Per_PW_H1
    Low    PORTC.0
    Pauseus Per_PW_L
Return
Zero_PW:
    High   PORTC.0
    Pauseus Per_PW_H0
    Low    PORTC.0
    Pauseus Per_PW_L
Return
```

```
Bi_Phase:
    Buff_Add = Bi_add
    GoSub One_Bi
    GoSub One_Bi
    GoSub One_Bi
    For Z2 = 1 To 5
      If Buff_Add.0 = 0  Then
        GoSub Zero_Bi
      Else
        GoSub One_Bi
      End If
      Buff_Add.0 = Buff_Add.1
      Buff_Add.1 = Buff_Add.2
      Buff_Add.2 = Buff_Add.3
      Buff_Add.3 = Buff_Add.4
      Buff_Add.4 = Buff_Add.5
    Next Z2
    For Z3 = 1 To 6
      If Buff_Com.0 = 0  Then
        GoSub Zero_Bi
      Else
        GoSub One_Bi
      End If
      Buff_Com.0 = Buff_Com.1
      Buff_Com.1 = Buff_Com.2
      Buff_Com.2 = Buff_Com.3
      Buff_Com.3 = Buff_Com.4
      Buff_Com.4 = Buff_Com.5
      Buff_Com.5 = Buff_Com.6
```

```
    Next Z3
      PORTC=%00000000
Return
One_Bi:
    Low    PORTC.1
    Pauseus Per_Bi_L
    High   PORTC.1
    Pauseus Per_Bi_H
Return
Zero_Bi:
    High   PORTC.1
    Pauseus Per_Bi_H
    Low    PORTC.1
    Pauseus Per_Bi_L
Return
```

กิตติกรรมประกาศ

ขอขอบพระคุณบิดามารดาของเราที่ให้กำเนิด เลี้ยงดู และให้การศึกษาแก่เรา

ขอขอบพระคุณ อ.เกียรติศักดิ์ คมวัชระ ผู้เป็นอาจารย์ที่ปรึกษา สำหรับคำปรึกษา คำแนะนำ และความาช่วยเหลือแก่เราตลอดมา

ขอขอบพระคุณอาจารย์ภาคคอนโทรลทุกท่านที่ให้ความรู้แก่เรา

ขอขอบคุณพี่โอ๋ พี่เดียว พี่พอน เม้ง และเพื่อนภาคทุกคนที่ให้คำแนะนำ คำปรึกษา และช่วยให้กำลังใจ

ขอขอบคุณพี่ที่สโตร์และพี่ธุรการภาคที่อำนวยความสะดวกในการขอใช้ห้องและเบิกเครื่องมือ

สุดท้ายนี้ขอขอบคุณภาควิศวกรรมระบบควบคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณ

หนังสืออ้างอิง

(REFERENCE)

1. หนังสือเรื่อง "รีโมทคอนโทรลอินฟราเรด 10 ช่อง" รวบรวมจากนิตยสารเซมิคอนดัคเตอร์ 235 ฉบับเดือน พฤษภาคม 2545 หน้าที่ 161 – 167
2. หนังสือเรื่อง "รีโมตเครื่องควบคุมไร้สาย" บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน)
3. หนังสือ "ปฏิบัติการไมโครคอนโทรลเลอร์ PIC 16F87x" Innovative Experiment Co.,Ltd.
4. PicBasic Pro Compiler, micro Engineering Labs, Inc.
5. PIC 16F87x Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers, MICROCHIP 2001
6. Everlight Electronics Co.,Ltd. data sheet
7. Fairchild Semiconductor data sheet
8. sgs-thomson microelectronics data sheet
9. NE/SA/SE555/SE/555C data sheet, Phillips 2003