เครื่องทดสอบอินฟราเรดรีโมทคอนโทรล

INFRARED REMOTECONTROL TESTER

โดย

นายนพปฐล เฉยศิริ

นายศังสิต จามจุรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

INFRARED REMOTECONTROL TESTER

BY

Mr. NOPADOL CHOEISIRI

Mr. SUNGSIT CHAMCHUREE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMEN FOR THE DEGREE OF

BACHELER IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOGY LADKRABANG

2003

หัวข้อปริญญานิพนธ์	เครื่องทดสอบอินฟราเรดรีโมทคอนโทรล
TITLE	INFRARED REMOTECONTROL TESTER
นักศึกษา	นายนพปฎล เฉยศิริ รหัสประจำตัว 44015652
	นายศังสิต จามจุรี รหัสประจำตัว 44015670
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ผศ.ไพศาล สิทธิโยภาสกุล
ระดับการศึกษา	ปริญญาวิศวกรรมศาสตรบัณฑิต
	สาขาวิชาวิศวกรรมสารสนเทศ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2546

ปริญญานิพนธ์นี้ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(ผศ.ไพศาล สิทธิโยภาสกุล)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	เครื่องทดสอบอินฟราเรดรีโมทคอนโทรล
นักศึกษา	นายนพปฎล เฉยศิริ รหัสประจำตัว 44015652
	นายศังสิต จามจุรี รหัสประจำตัว 44015670
อาจารย์ที่ปรึกษา	ผศ.ไพศาล สิทธิโยภาสกุล
ภาควิชา	วิศวกรรมสารสนเทศ คณะวิศวกรรมศาสตร์
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา	2546

บทคัดย่อ

โครงงานนี้เป็นการออกแบบเครื่องทดสอบการรับ – ส่งสัญญาณ สำหรับอินฟราเรดรีโมทคอนโทรล โดยจะแบ่งการทำงานออกเป็น 2 ส่วน คือส่วนของฮาร์ดแวร์ ซึ่งจะใช้ไมโครคอนโทรลเลอร์ เป็นตัววิเคราะห์สัญญาณอินฟราเรดที่รับเข้ามาจากตัวรีโมทคอนโทรล ข้อมูลที่ได้จะถูกส่งต่อไปยังเครื่องคอมพิวเตอร์ซึ่งจะใช้การเชื่อมต่อผ่านทางพอร์ตอนุกรม (RS-232) ในส่วนของซอฟต์แวร์ได้ใช้ภาษา Visual Basic 6.0 ทำหน้าที่แสดงผลของสัญญาณซึ่งจะแสดงรหัสและรูปสัญญาณที่ส่งออกมา ทำการจัดเก็บสัญญาณที่รับเข้ามา และสามารถควบคุมการส่งสัญญาณกลับออกไปควบคุมอุปกรณ์ได้

PROJECT TITLE	INFRARED REMOTECONTROL TESTER
STUDENT	Mr. Nopadol Choeisiri ID. 44015652
	Mr. Sungsit Chamchuree ID. 44015670
ADVISOR	Asst.Prof.Paisan Sithiyopasakul
COURSE	Bachelor of Engineering in Information Engineering
DEPARTMENT	Information Engineering Faculty of Engineering
	King Mongkut's Institute of Technology Ladkrabang
YEAR	2003

ABSTRACT

This project presents to design an infrared remote control tester. By separate task 2 part , the hardware , uses microcontroller to capture the signal and signal analysis from remote control. The data will be transmit to computer via serial port interface (RS 232) .The software part, Uses visual basic 6.0 will show data code and signal form ,with store the IR signals from remote control and play signals back to control equipment.

กิตติกรรมประกาศ

สำหรับปริญญานิพนธ์ฉบับนี้ ที่สำเร็จลุล่วงมาได้ก็ต้องขอบคุณหลาย ๆ ฝ่ายที่ช่วยในการสนับสนุนจนประสบความสำเร็จ ขอขอบคุณ ผศ.ไพศาล สิทธิโยภาสกุล สำหรับคำปรึกษา คำแนะนำดี ๆ และให้การช่วยเหลือเป็นอย่างมาก และขอขอบคุณสำหรับผู้มีส่วนร่วม แต่มิได้เอ่ยนามในที่นี้ และที่สำคัญกำลังใจจาก บิดา มารดา และคนรอบข้างที่คอยเป็นกำลังใจมาตลอด

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

ปริญญานิพนธ์นี้มีชื่อว่า " เครื่องทดสอบอินฟราเรดรีโมทคอนโทรล " เนื่องจากในปัจจุบัน จะเห็นได้ว่าระบบการควบคุมแบบไร้สาย (RemoteControl) ได้เข้ามาเกี่ยวข้องในชีวิตประจำวันของคนเราเป็นอย่างมาก โดยเฉพาะการใช้อินฟราเรดรีโมทคอนโทรล เพราะสามารถควบคุมได้จากระยะไกล ปัจจุบันนี้ ผู้ผลิตเครื่องใช้ไฟฟ้าต่างมีเทคนิคการในส่งสัญญาณควบคุมที่เป็นลักษณะเฉพาะ หากต้องการนำไปประยุกต์ใช้งาน จะต้องรู้รหัสที่ส่งออกมา และจัดเก็บข้อมูลเพื่อใช้ในการทดสอบเปรียบเทียบ และใช้อ้างอิงในภายหลัง เครื่องทดสอบอินฟราเรดรีโมทคอนโทรล จะทดสอบสัญญาณรีโมทคอนโทรล จากนั้นจะตรวจสอบความถูกต้องของสัญญาณโดยนำมาเปรียบเทียบกับข้อมูลจริงที่มีอยู่ และเลียนแบบสัญญาณรีโมทคอนโทรลที่มีในฐานข้อมูลเพื่อที่จะนำเอาไปควบคุมแทนรีโมทคอนโทรลตัวเดิม โดยเครื่องนี้จะใช้ไมโครคอนโทรลเลอร์ตระกูล AVR ของบริษัท Atmel โดยใช้เบอร์ AT90S2313 เป็นไอซีประเภทซีมอส ชนิด 8 บิต เป็นตัววิเคราะห์สัญญาณอินฟราเรดที่รับมาจากรีโมทคอนโทรล และเชื่อมต่อไปยังคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม (RS 232) โปรแกรมจะจัดเก็บสัญญาณที่รับเข้ามา และทำการส่งสัญญาณไปควบคุมอุปกรณ์นั้น ๆ โดยตรงผ่านทางคอมพิวเตอร์

1.1 วัตถุประสงค์

1.1.1 เพื่อทำการหาข้อมูลที่ส่งออกมาจากรีโมทคอนโทรลว่านำไปใช้ควบคุมการทำงานส่วนในใดของอุปกรณ์เครื่องใช้ไฟฟ้า โดยการนำมาเปรียบเทียบกับข้อมูลที่มีอยู่

1.1.2 เพื่อนำไปใช้ประกอบในการทดสอบการเสียของอินฟราเรดรีโมทคอนโทรล

1.1.3 เพื่อเลียนแบบสัญญาณรีโมทคอนโทรลที่ได้มีการบันทึกเอาไว้ ให้สามารถนำไปควบคุมอุปกรณ์ชนิดนั้นๆ แทนรีโมทตัวเดิมได้

1.1.4 ศึกษาและวิเคราะห์ การส่งสัญญาณของอินฟราเรดรีโมทคอนโทรล เพื่อเป็นแนวทางในการนำไปออกแบบรีโมทคอนโทรล หรืออุปกรณ์ที่ต้องการจะควบคุม

1.2 ขอบเขตโครงงาน

ขอบเขตของโครงงานนี้แบ่งออกเป็น 3 ส่วนคือ

ส่วนแรกจะเป็นการศึกษาและเก็บรวบรวมข้อมูล รูปแบบการเข้ารหัส หรือโปรโตคอลของอินฟราเรดรีโมทคอนโทรลแต่ละยี่ห้อ เพื่อนำมาใช้สร้างโปรแกรมควบคุมไมโครคอนโทรลเลอร์ และ ศึกษาระบบฐานข้อมูลเพื่อนำมาใช้ในการเก็บรวบรวม เป็นฐานข้อมูลของอินฟราเรดรีโมทคอนโทรล

ส่วนที่สองเป็นการออกแบบสร้างวงจรในการรับส่งสัญญาณอินฟราเรดรีโมคอนโทรลและสร้างโปรแกรมควบคุมไมโครคอนโทรลเลอร์

ส่วนที่สามเป็นการสร้างโปรแกรมติดต่อกับฮาร์ดแวร์โดยใช้ Microsoft Visual Basic 6.0 และสร้างฐานข้อมูลเพื่อเก็บข้อมูลของอินฟราเรดรีโมทคอนโทรลแต่ละยี่ห้อ

1.3 เนื้อหาของแต่ละบท

บทที่ 1 เป็นการกล่าวถึงวัตถุประสงค์ ขอบเขตและผลที่คาดว่าจะได้รับของโครงงาน และเนื้อหาของแต่ละบท

บทที่ 2 เป็นทฤษฎีเกี่ยวกับโครงสร้างของไมโครคอนโทรลเลอร์ ตระกูล AVR ทฤษฎีการส่งสัญญาณของอินฟราเรดรีโมทคอนโทรล รูปแบบของสัญญาณ

บทที่ 3 กล่าวถึงการออกแบบทางด้านฮาร์ดแวร์(Hardware)และทางด้านซอฟแวร์(Software) โดยทางด้านฮาร์ดแวร์ อธิบายถึงการออกแบบวงจรในส่วนต่างๆ และหน้าที่ของแต่ละส่วนนั้น ส่วนทางด้านซอฟแวร์ อธิบายถึงการแสดงผลออกมาทางคอมพิวเตอร์ การเปรียบเทียบข้อมูลในฐานข้อมูล การออกแบบส่วนของฐานข้อมูล

บทที่ 4 การทดลองและผลลัพธ์ที่ได้ การใช้งานโปรแกรมแสดงผล

บทที่ 5 สรุปและวิจารณ์ผลการทดลอง

บทที่ 2
ทฤษฎีทั่วไป

2.1 ไมโครคอนโทรลเลอร์ตระกูล AVR

ความก้าวหน้าอย่างต่อเนื่องในด้านไมโครคอนโทรลเลอร์ และไมโครโปรเซสเซอร์ ได้สร้างอุปกรณ์ตัวใหม่ขึ้นมาและอยู่ในตัวถังไอซีมาตรฐาน ซึ่งมีประสิทธิภาพในการประมวลผลพอๆ กับเครื่องคอมพิวเตอร์รุ่นแรกๆ ในบทนี้จะกล่าวถึงไมโครคอนโทรลเลอร์ตระกูล AVR ของบริษัท Atmel มีเบอร์ ขึ้นต้นด้วย AT90Sxxxx เป็นไอซีประเภทซีมอส ชนิด 8 บิต ที่มีโครงสร้างพื้นฐานแบบ RISC (Reduce Instruction Set Computer) ซึ่งมีคุณสมบัติดังต่อไปนี้

คุณสมบัติของไมโครคอนโทรลเลอร์ ตระกูล AVR เบอร์ AT90S2313

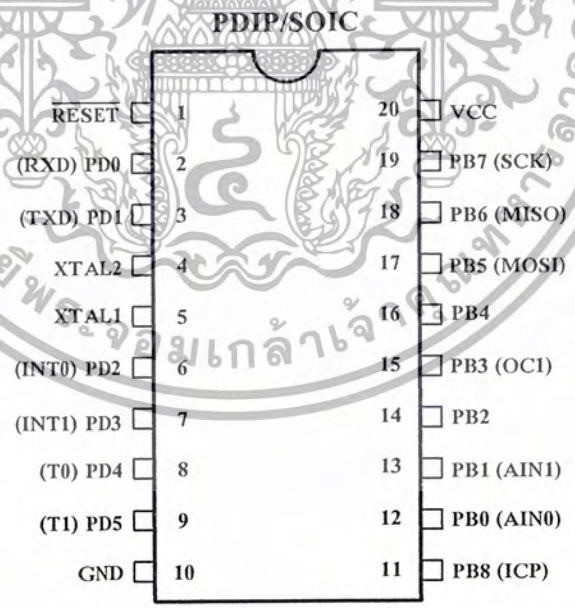
- สถาปัตยกรรมภายในถูกออกแบบให้ใช้สถาปัตยกรรมแบบ RISC (RISC คือ ทำให้การประมวลผลมีความเร็ว 1 คำสั่ง / 1 Clock หรือ CPU สามารถประมวลคำสั่งได้ 1 MIPS / MHz)
- มีคำสั่งในการควบคุมการทำงานไมโครคอนโทรลเลอร์จำนวน 118 คำสั่ง
- มีกลุ่มรีจิสเตอร์ (register) ที่ใช้งานทั่วไป ขนาด 8 บิต จำนวน 32 ตัว
- มีหน่วยความจำโปรแกรมชนิดแฟลช (Flash)ขนาด 2 กิโลไบต์ ที่สามารถเขียนข้อมูลซ้ำได้ 1,000 ครั้ง
- มีหน่วยความจำภายในชนิด EEPROM ขนาด 128 ไบต์ สามารถเขียนข้อมูลซ้ำได้ 100,000 ครั้ง
- หน่วยความจำชนิด SRAM ขนาด 128 ไบต์
- สามารถล็อก ป้องกันการอ่านข้อมูลจากหน่วยความจำแฟลชและ EEPROM ได้
- มีตัวนับ/จับเวลา ขนาด 8 บิตและ 16 บิต พร้อมตัวหาร (Prescaler)
- มีตัวเปรียบเทียบสัญญาณอะนาลอก
- มีวงจรวอตช์ด็อกพร้อมตัวกำเนิดสัญญาณจับเวลาให้วอตช์ด็อก
- มีตัวเชื่อมต่อแบบ SPI (Serial Programming Interface)
- มีโหมดประหยัดพลังงาน 2 โหมดคือ โหมดพัก (Idle mode) และโหมดลดพลังงาน (Power-down mode)
- มีแหล่งกำเนิดสัญญาณอินเตอร์รัปต์ได้ทั้งจากภายในและภายนอกไอซี

- สามารถเลือกใช้สัญญาณนาฬิกาจากภายในไอซีได้ ทำให้ไม่ต้องต่อสัญญาณนาฬิกาจากภายนอก

- การสิ้นเปลืองพลังงาน ที่สัญญาณนาฬิกา 4 เมกะเฮิรตซ์ แรงดันไฟเลี้ยง 3 โวลต์ อุณหภูมิ 25 องศาเซลเซียส

  - ขณะทำงาน : 2.8 มิลลิแอมป์
  - ขณะอยู่ในโหมดพัก : 0.8 มิลลิแอมป์
  - ขณะอยู่ในโหมดลดพลังงาน : น้อยกว่า 1 ไมโครแอมป์

- แรงดันไฟเลี้ยง
  - 2.7 ถึง 6.0 โวลต์ ในรุ่น AT90S2313-4
  - 4.0 ถึง 6.0 โวลต์ ในรุ่น AT90S2313-10
- ความถี่ที่ใช้ในการทำงาน
  - 0 ถึง 4 เมกะเฮิรต์ ในรุ่น AT90S2313-4
  - 0 ถึง 10 เมกะเฮิรต์ ในรุ่น AT90S2313-10



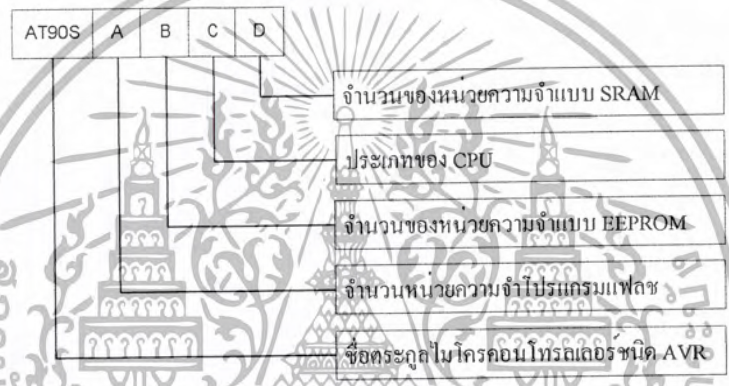
รูปที่ 2.1 ขาใช้งานต่าง

รายละเอียดขาต่าง

VCC	ขาจ่ายไฟให้กับ CPU
GND	ขากราวด์
Port B (PB7..PB0)	เป็นพอร์ตสองทิศทางขนาด 8 บิต สามารถกำหนดให้แต่ละขาของพอร์ตสามารถพูลอัป (pull-up) ภายในแยกจากกัน ที่แต่ละขาของพอร์ตสามารถจ่ายกระแสออกไปได้ถึง 20 mA ซึ่งสามารถขับ LED ได้โดยตรง นอกจากนั้นที่ขา PB0 และ PB1 สามารถใช้เป็นอินพุต AIN0 และ AIN1 สำหรับการเปรียบเทียบสัญญาณอะนาลอกอีกด้วย
Port D(PD6..PD0)	เป็นพอร์ตสองทิศทางขนาด 7 บิต มีคุณสมบัติคล้ายกันกับพอร์ต B นอกจากนี้พอร์ต D ยังใช้สำหรับทำหน้าที่อื่นได้อีกคือ - PD0 = RXD เป็นขาอินพุตสำหรับพอร์ตอนุกรม - PD1 = TXD เป็นขาเอาต์พุตสำหรับพอร์ตอนุกรม - PD2 = INT0 เป็นขาอินเตอร์รัปต์จากภายนอก - PD3 = INT1 เป็นขาอินเตอร์รัปต์จากภายนอก - PD4 = T0 เป็นขาสำหรับรับสัญญาณคล็อกจากภายนอกสำหรับไทม์เมอร์เคาร์เตอร์ 0(Timer/Counter 0) - PD5 = T1 เป็นขาสำหรับรับสัญญาณคล็อกจากภายนอกสำหรับไทม์เมอร์เคาร์เตอร์ 1 (Timer/Counter 1) - PD6 = ICP ใช้เป็นอินพุตสำหรับสั่งไทม์เมอร์เคาร์เตอร์ 1เก็บค่าที่กำลังนับไว้ในรีจิสเตอร์
RESET	ขารีเซต ซึ่งจะเกิดการรีเซตก็ต่อเมื่อมีลอจิก 0 เข้ามาเป็นเวลาไม่ต่ำกว่า 50 ns จะเกิดการรีเซตแม้ว่าจะไม่มีสัญญาณคล็อกก็ตาม
XTAL1	เป็นขาอินพุตสำหรับรับสัญญาณคล็อกจากภายนอก
XTAL2	เป็นขาเอาต์พุตของสัญญาณคล็อก

2.1.1 สถาปัตยกรรมภายใน

ไมโครคอนโทรลเลอร์ตระกูล AVR ได้แบ่ง หน่วยประมวลผล (CPU) ออกเป็น 2 ระดับ เพื่อความเหมาะสมในการใช้งาน โดยหมายเลขที่อยู่บนตัวไอซีจะเป็นตัวบอกข้อมูลที่เกี่ยวกับประเภทของ หน่วยประมวลผล จำนวนหน่วยความจำชนิดต่างๆ ดังในรูปที่ 2.2 แสดงรายละเอียดของหมายเลขแต่ละตัวที่อยู่บนตัวไอซีส่วนในตารางที่ 2.1 ได้อธิบายถึงรหัสของจำนวนหน่วยความจำข้อมูลแบบ EEPROM และแบบ SRAM



รูปที่ 2.2 รายละเอียดในแต่ละส่วนของหมายเลยที่อยู่บนตัวไอซี

ตารางที่ 2.1 รหัสของหน่วยความจำแบบ EEPROM และ SRAM

รหัส	0	1	2	3	4	5	6	7	8	9	A	B
จำนวนหน่วยความจำ	0	32	64	128	256	512	1K	2K	4K	8K	16K	32K

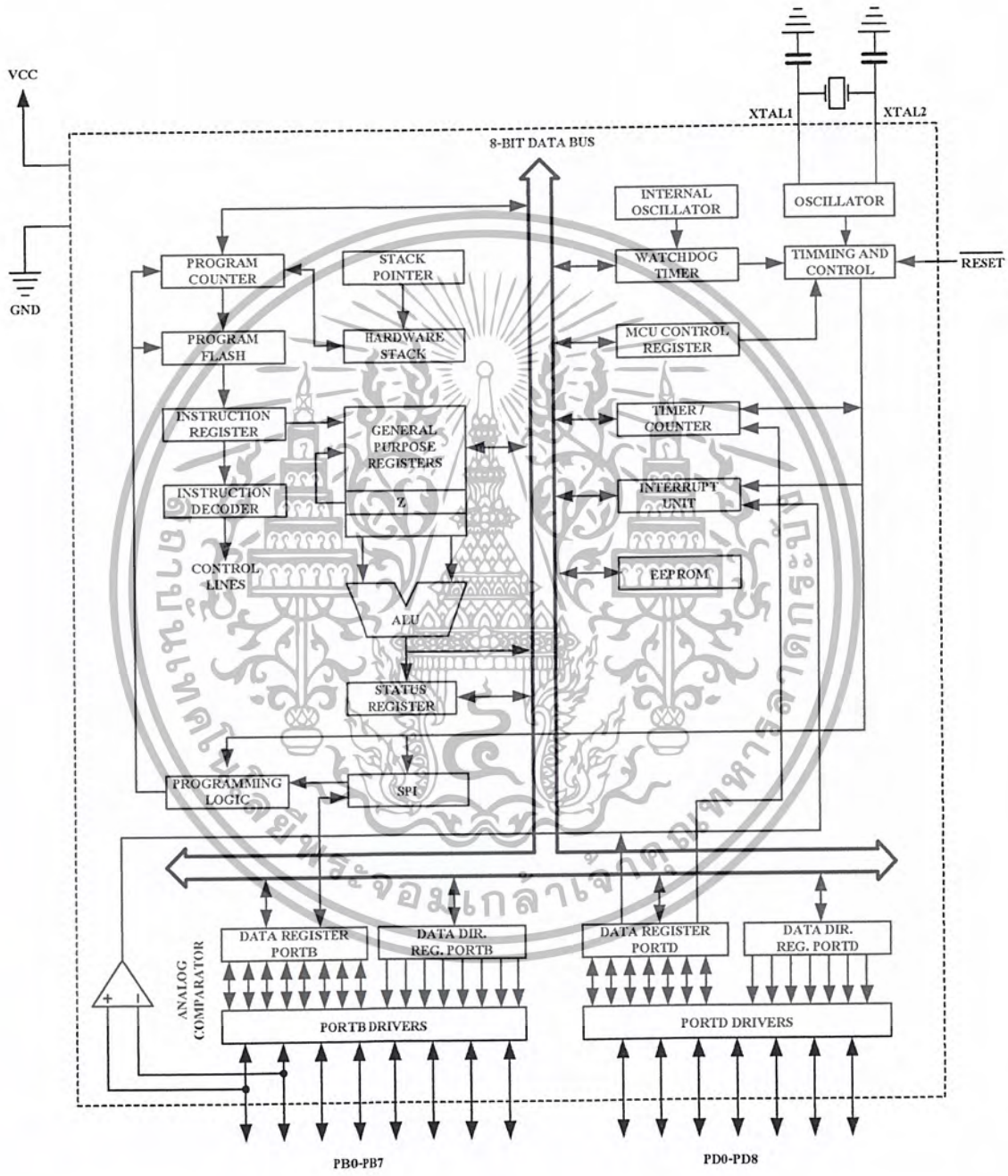
ตารางที่ 2.2 รายละเอียดคุณสมบัติของเบอร์ต่างๆ

Number	Flash (KB)	EEPROM (Byte)	CPU (Model)	SRAM (Byte)	Counter Timer	UART	ADC	Pins
AT90S1200	1	64	0	0	1	No	No	20
AT90S1220	1	64	2	0	1	No	No	8
AT90S2313	2	128	1	128	2	Yes	No	20
AT90S4414	4	256	1	256	3	Yes	No	40/44
AT90S4433	4	256	3	128	?	Yes	Yes	28
AT90S8518	8	512	1	512	3	Yes	No	40/44

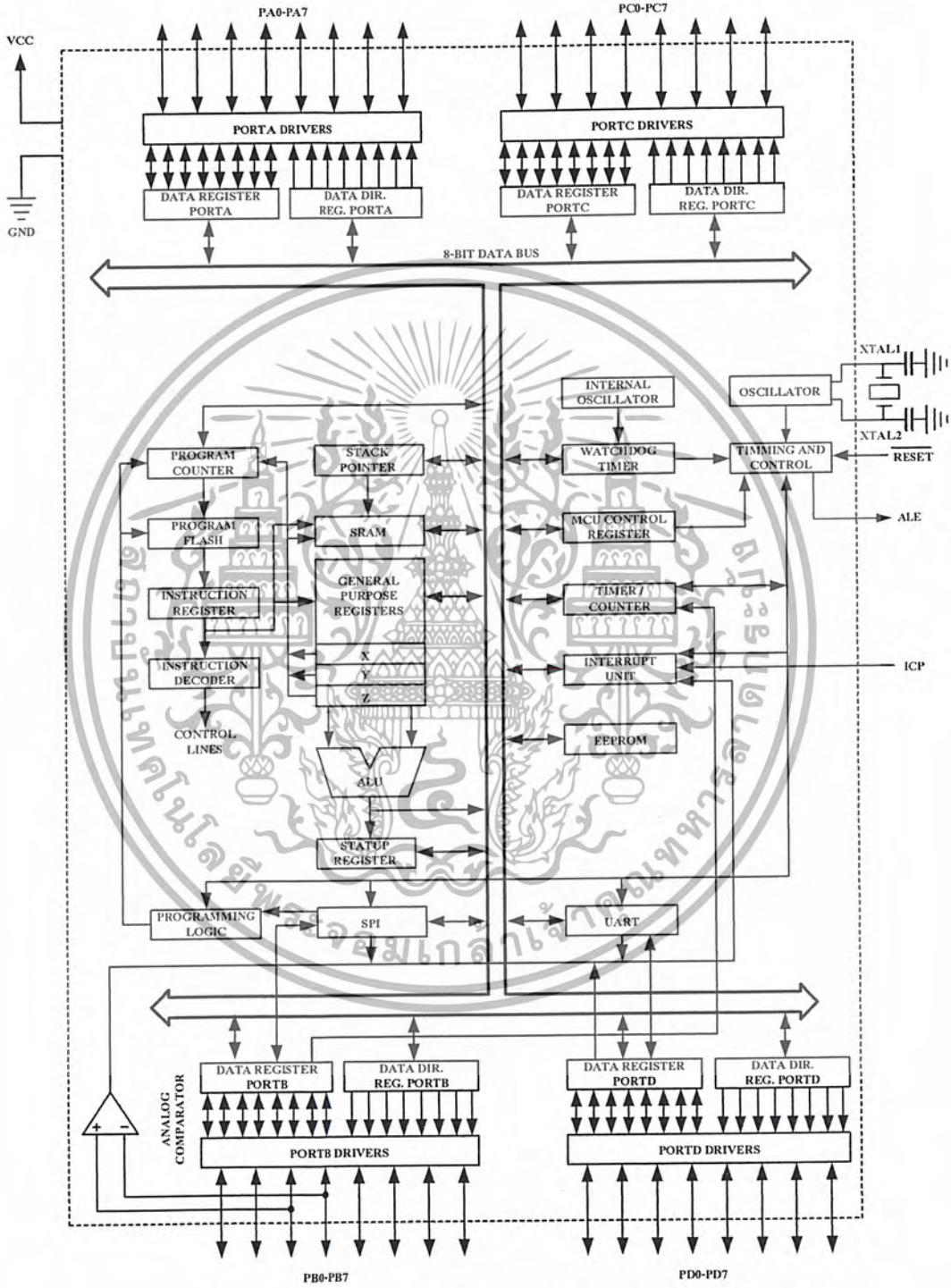
ในรูปที่ 2.3 และ 2.4 แสดงถึงบล็อกไดอะแกรมของไมโครคอนโทรลเลอร์เบอร์ AT90S1200 (ระดับต่ำ) และ AT90S8515 (ระดับสูง)

เมื่อเปรียบเทียบกันในรูปที่ 2.3 และ 2.4 เราสามารถพบองค์ประกอบที่เหมือนกัน และบางอย่างที่แตกต่างกัน ไอซีตระกูล AVR ทั้งหมด จะมีรีจิสเตอร์ทั่วไปขนาด 8 บิตจำนวน 32 ตัว ซึ่งสามารถเข้าถึงข้อมูลได้ ภายในเวลาในหนึ่งรอบของสัญญาณนาฬิกา ซึ่งหมายความว่า ไมโครคอนโทรลเลอร์ สามารถจัดการข้อมูลภายในรีจิสเตอร์ใช้งานทั่วไปได้เสร็จภายในหนึ่งรอบของสัญญาณนาฬิกา

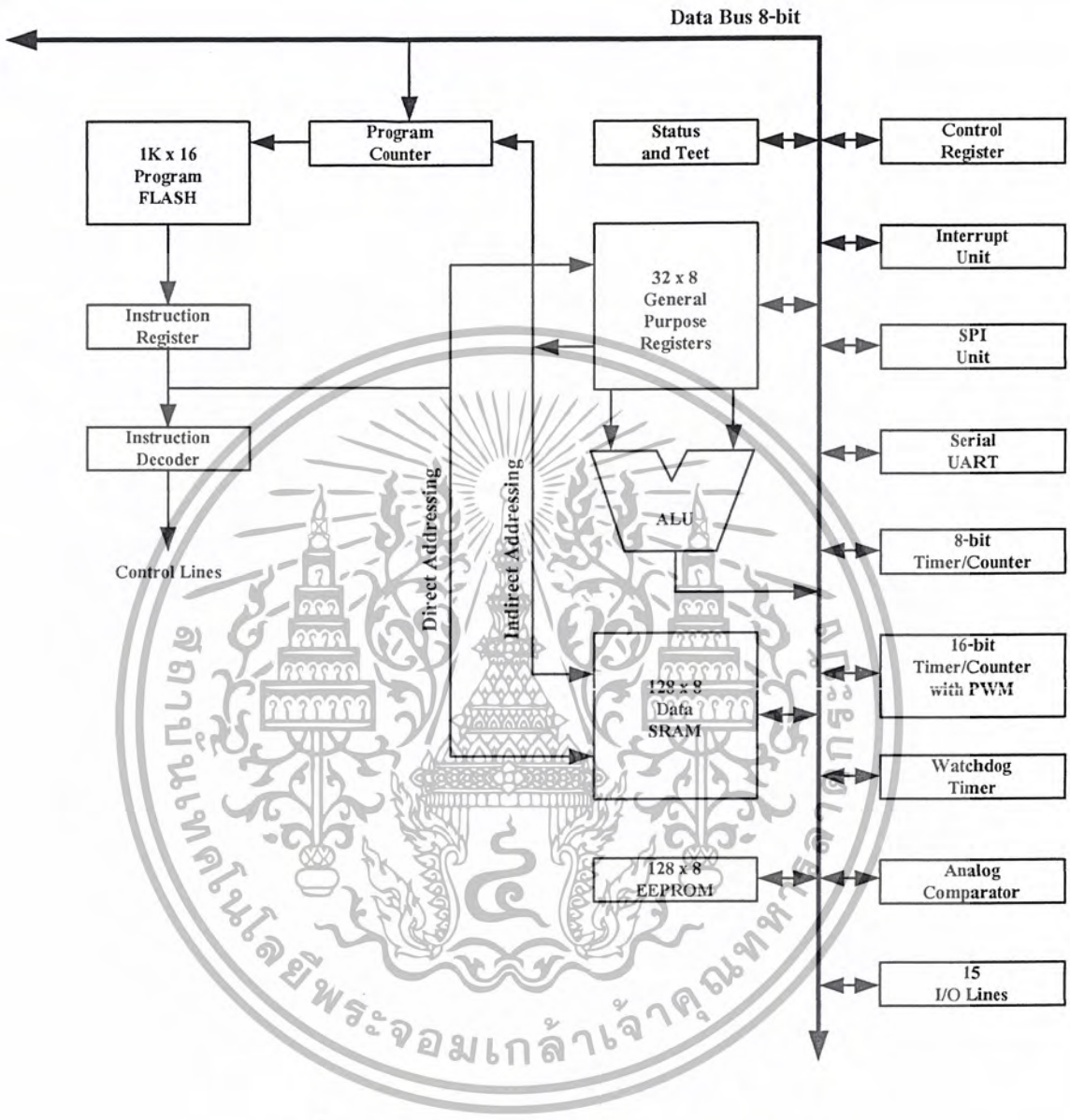
ในหน่วยประมวลผลแบบที่ 1 (model 1) จะมีรีจิสเตอร์ R26 ถึง R31 ซึ่งเป็นรีจิสเตอร์ขนาด 8 บิต จำนวน 6 ตัว สามารถนำมาจับคู่เพื่อใช้เป็นรีจิสเตอร์ขนาด 16 บิต ใช้เป็นตัวชี้ตำแหน่งในการเข้าถึงข้อมูลแบบอ้อมของ SRAM ส่วนในหน่วยประมวลผลแบบที่ 0 (model 0) จะใช้รีจิสเตอร์เพียง 2 ตัว นั้นก็หมายความว่าจะมีรีจิสเตอร์ 16 บิตไว้ใช้งานเพียงตัวเดียว ซึ่งจะใช้เป็นตัวชี้ตำแหน่งในการเปิดตารางข้อมูล โดยการอ้างถึงรีจิสเตอร์ 16 บิต ที่เพิ่มขึ้นมานี้จะใช้ชื่อเรียกว่า รีจิสเตอร์ X ,Y และ Z ตามลำดับ ในรูปที่ 2.5 จะแสดงสถาปัตยกรรมแบบ RISC ของ AT90S2313 ซึ่งใช้ในการทำโครงงานนี้



รูปที่ 2.3 บล็อกไดอะแกรมไมโครคอนโทรลเลอร์เบอร์ AT90S1200



รูปที่ 2.4 บล็อกไดอะแกรมไมโครคอนโทรลเลอร์เบอร์ AT90S8515



รูปที่ 2.5 สถาปัตยกรรมแบบ RISC ของ AT90S2313

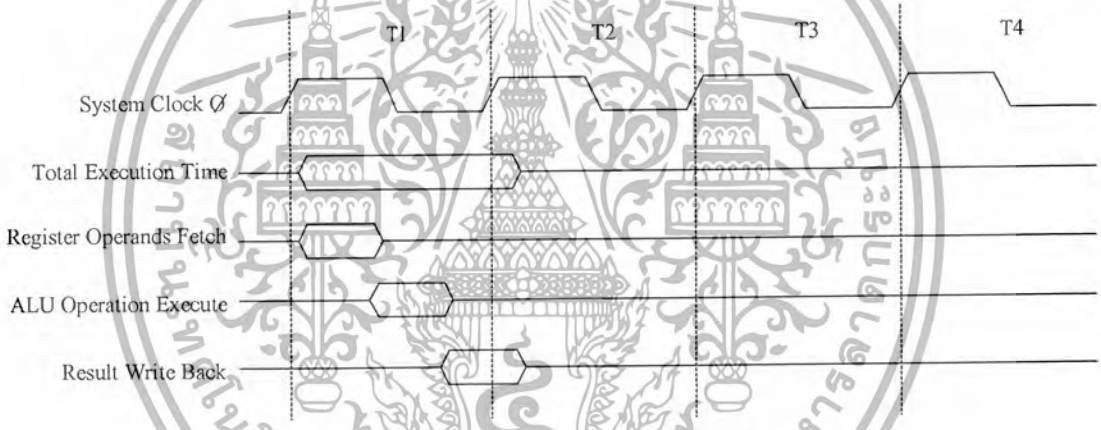
2.1.2 ส่วนประมวลผลทางคณิตศาสตร์

ส่วนประมวลผลของไมโครคอนโทรลเลอร์ ชนิด AVR จะไม่มีแอคคิวมูลเลเตอร์ (Accumulator register) สำหรับทำงานทางคณิตศาสตร์ แต่จะประยุกต์ใช้การทำงานทางคณิตศาสตร์ระหว่างรีจิสเตอร์กับรีจิสเตอร์แทน โดยจะใช้รีจิสเตอร์ที่ใช้งานทั่วไปขนาด 8 บิต จำนวน 32 ตัว ต่อเข้ากับส่วนคำนวณและเปรียบเทียบทางคณิตศาสตร์หรือ ALU (Arithmetic Logical Unit) โดยตรง

ทำให้สามารถใช้คำสั่งเพียงคำสั่งเดียวเข้าถึงรีจิสเตอร์ 2 ตัวได้ในหนึ่งรอบสัญญาณนาฬิกา ตราบใดที่มีรีจิสเตอร์เพียงพอมันจะช่วยลดจำนวนของคำสั่งในการทำงานทางคณิตศาสตร์ลงได้

ตัวอย่างเช่น คำสั่งการบวกจากสามคำสั่ง (คำสั่งโหลดข้อมูลเข้าไปใน แอกคิวมูเลเตอร์ , คำสั่งบวก , คำสั่งนำค่าผลลัพธ์ที่อยู่ในแอกคิวมูเลเตอร์ไปเก็บ) เหลือหนึ่งคำสั่งได้ ยิ่งไปกว่านั้น ส่วนประมวลผลนี้ยังรองรับคำสั่งชิปตระกูล MCS-51 ได้ด้วย และมีความสามารถทำงานตามคำสั่งคณิตศาสตร์ปกติได้ทุกคำสั่ง ในรูปที่ 2.6 แสดงถึงการทำงานของ ALU ระหว่างรีจิสเตอร์กับรีจิสเตอร์ภายในหนึ่งรอบสัญญาณนาฬิกา

การทำงานของ ALU ได้จัดแบ่งระบบการจัดการข้อมูลออกเป็น 3 ส่วนคือ ส่วนของการจัดการทางคณิตศาสตร์ ส่วนของการกระทำทางลอจิก และในส่วนของการกระทำกับบิต



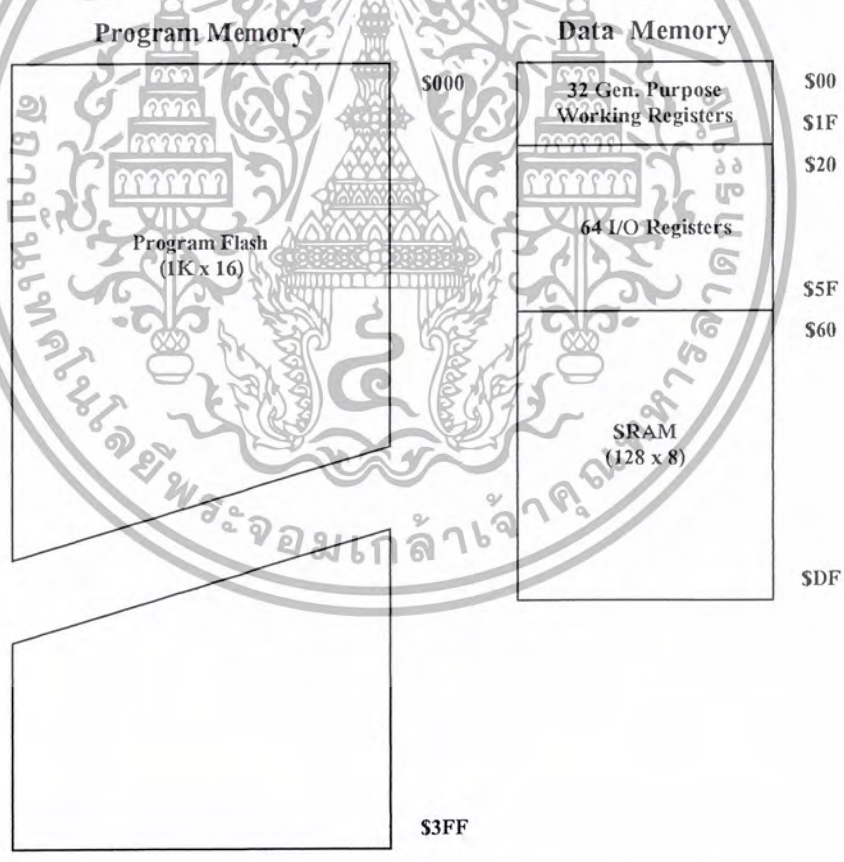
รูปที่ 2.6 การทำงานของ ALU ในหนึ่งคาบสัญญาณนาฬิกา

เทคโนโลยีไปป์ไลน์ (Pipeline) ช่วยให้ส่วนประมวลผลสามารถทำงาน 1 คำสั่ง ได้ภายในหนึ่งรอบของสัญญาณนาฬิกาเท่านั้น ซึ่งเป็นผลให้ชิป AVR นี้ทำงานได้ถึง 1 ล้านคำสั่งต่อวินาทีต่อสัญญาณนาฬิกา 1 เมกะเฮิรต์ (1 MIPS/MHz)

หน่วยประมวลผลนี้มีโครงสร้างแบบสแตติก(Static) ทำให้สามารถใช้ความถี่นาฬิกาได้ต่ำเท่าไรก็ได้ตามต้องการช่วยให้ ผู้ออกแบบวงจรสามารถลดสัดส่วนการใช้กระแสของชิปนี้ ตามต้องการได้โดยตรง

2.1.3 หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล

ระบบการทำงานของไมโครคอนโทรลเลอร์ แบบ AVR จะใช้ หลักการออกแบบสถาปัตยกรรมแบบฮาร์วาร์ด (Harvard memory architecture) ซึ่งหน่วยความจำที่ใช้เก็บโปรแกรม (Program memory) และหน่วยความจำที่ใช้เก็บข้อมูล (Data memory) จะใช้ระบบบัสที่แยกออกจากกัน ซึ่งการเข้าถึงข้อมูล ภายในรีจิสเตอร์จะใช้การอ้างตำแหน่งหน่วยความจำภายใน ที่ตำแหน่ง $00 ถึง $1F จำนวน 32 ตำแหน่ง และในไมโครคอนโทรลเลอร์ได้จัดแบ่งให้มีรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของหน่วยอินพุตและเอาต์พุตต่างๆ อีก 64 ตำแหน่ง ซึ่งสามารถเรียกใช้งานได้โดยการอ้างตำแหน่งหน่วยความจำที่ตำแหน่ง $20 ถึง $5F



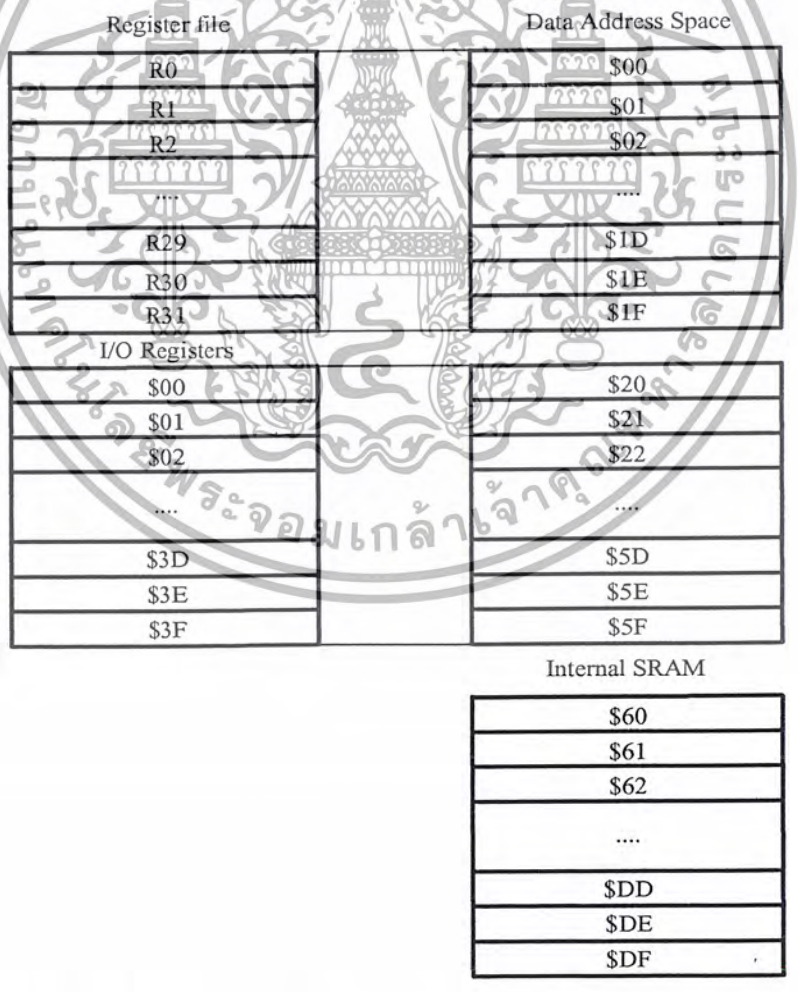
รูปที่ 2.7 การเปรียบเทียบหน่วยความจำ

หน่วยความจำโปรแกรมแบบแฟลต

ในเบอร์ AT90S2313 จะมีหน่วยความจำโปรแกรม ประเภทแฟลชขนาด 2 กิโลไบต์ บรรจุอยู่ภายใน ซึ่งเป็นแบบ ISP (In - System programmable) และสามารถลบแล้วเขียนใหม่ได้ประมาณ 1,000 ครั้ง ในส่วนของโปรแกรมเคาเตอร์ (PC) จะมีขนาด 10 บิต ทำให้สามารถที่จะถ้าถึงตำแหน่งได้ถึง 1,024 ตำแหน่ง

หน่วยความจำข้อมูลแบบ SRAM

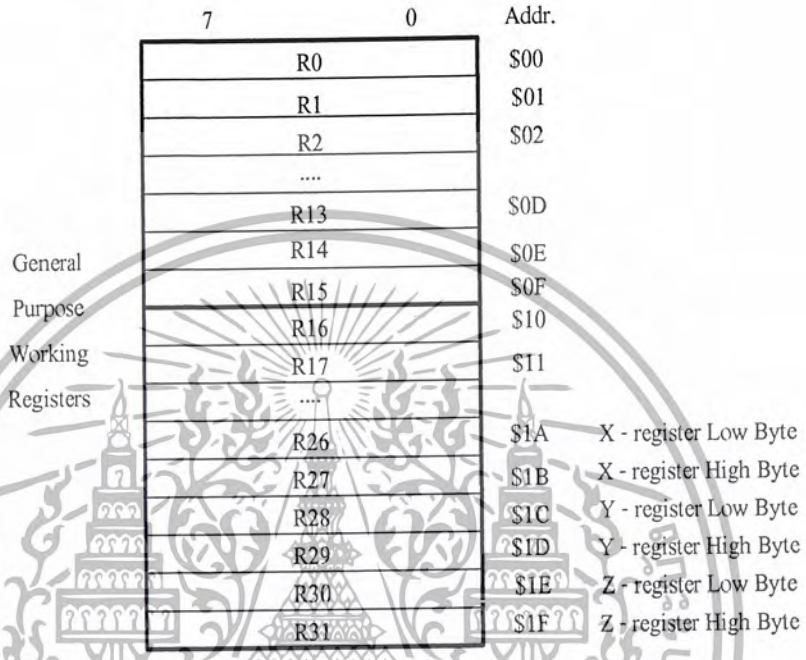
หน่วยความจะข้อมูลภายในจำนวน 224 ตำแหน่ง ได้ถูกจัดสรรคไว้สำหรับรีจิสเตอร์ใช้งานทั่วไป,รีจิสเตอร์ใช้งานอินพุต/เอาต์พุต และหน่วยความจำภายใน SRAM โดยที่ 96 ตำแหน่งแรก ($00-5F) จะถูกจัดสรรคไว้สำหรับรีจิสเตอร์ใช้งานทั่วไปกับรีจิสเตอร์ใช้งานอินพุต/เอาต์พุต ส่วน 128 ตำแหน่ง ($60-$DF) ถัดไปจะถูกจัดไว้เป็นของหน่วยความจำข้อมูล SRAM



รูปที่ 2.8 การจัดสรรหน่วยความจำข้อมูลภายในของเบอร์ AT90S2313

รีจิสเตอร์ใช้งานทั่วไป

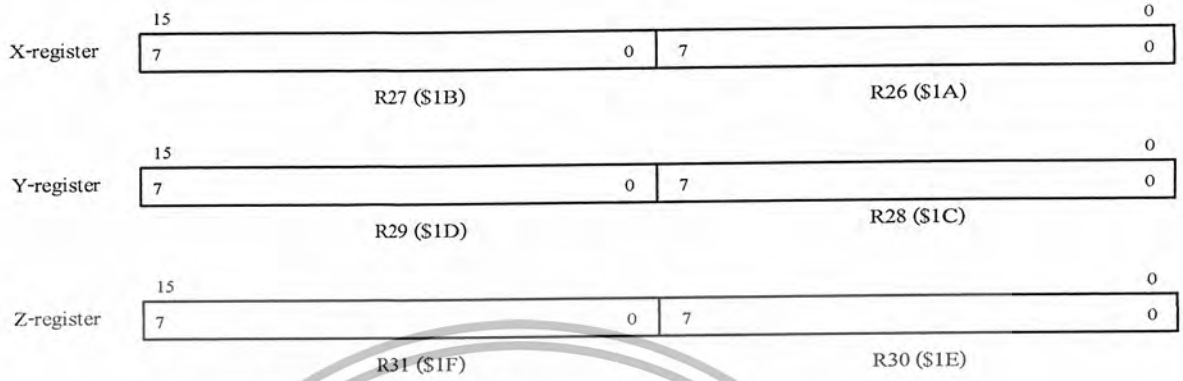
โครงสร้างรีจิสเตอร์ใช้งานทั่วไป



รูปที่ 2.9 ตำแหน่งของรีจิสเตอร์ใช้งานทั่วไป

รีจิสเตอร์ทั้งหมดสามารถใช้ชุดคำสั่ง เพื่อเข้าถึงได้โดยตรงและจะใช้ช่วงเวลาการเข้าถึงเพียง 1 คาบสัญญาณนาฬิกา โดยคำสั่ง SBCI , SUBI , CPI , ANDI และ ORI ซึ่งกระทำ ระหว่างรีจีสเตอร์กับค่าคงที่และรีจีสเตอร์กับรีจีสเตอร์และคำสั่ง LDI ที่ใช้โหลดค่าคงที่เข้าในรีจีสเตอร์ จะต้องใช้งานกับรีจีสเตอร์ R16 ถึง R31 ส่วนคำสั่ง SBC ,SUB,CP,AND และ OR และคำสั่งใช้งานอื่นๆ สามารถใช้งานได้กับรีจีสเตอร์ทุกตัว

รีจิสเตอร์ X , รีจิสเตอร์ Y และรีจิสเตอร์ Z รีจิสเตอร์ R26 ถึง R31 ที่อยู่ในรีจิสเตอร์ใช้งานทั่วไปสามารถนำมาต่อกันเพื่อทำเป็นคู่รีจิสเตอร์ เพื่อนำมาใช้งานเป็นตัวชี้ตำแหน่งข้อมูลแบบอ้อม ซึ่งจะทำให้ได้รีจิสเตอร์ชี้ตำแหน่งแบบอ้อมเป็นรีจิสเตอร์ X, Y และ Z ดังแสดงในรูปที่ 2.10



รูปที่ 2.10 รีจิสเตอร์ X,Y และ Z

อินพุต / เอาต์พุต รีจิสเตอร์

ไมโครคอนโทรลเลอร์เบอร์ AT90S2313 ส่วนของรีจิสเตอร์อินพุต/เอาต์พุตและรีจิสเตอร์อื่นๆที่เกี่ยวข้องเช่น รีจิสเตอร์สถานะ จะถูกกำหนดให้อยู่ในพื้นที่อินพุตเอาต์พุต ซึ่งจะสามารถเข้าถึงรีจิสเตอร์เหล่านี้ได้โดยการใช้คำสั่ง IN และคำสั่ง OUT ซึ่งเป็นคำสั่งที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่างรีจิสเตอร์ใช้งานทั่วไป กับรีจิสแตอร์อินพุต/เอาต์พุต รีจิสเตอร์ที่อยู่ในช่วงตำแหน่งตั้งแต่ $00 - $1F สามารถที่จะเข้าถึงได้ในระดับบิตได้โดยตรง โดยการใช้คำสั่ง SBI และ คำสั่ง CBI และยังสามารถตรวจสอบค่าในแต่ละบิตได้โดยการใช้คำสั่ง SBIS และ คำสั่ง SBIC

หน่วยประมวลผลจะติดต่อสื่อสารกับอุปกรณ์ภายนอกผ่านทางสายสัญญาณอินพุต/เอาต์พุต (I/O line) สายอินพุต/เอาต์พุต 8 เส้นจะต่อเข้ากับอินพุต/เอาต์พุตรีจิสเตอร์ 1 ตัว เนื่องจากตัวถังแบบ DIL ขนาด 20 ขา จะยอมให้มีอินพุต/เอาต์พุตได้มากที่สุด 15 ขา ดังนั้นในกรณีของชิป AT90S2313 จะมีอินพุต/เอาต์พุตรีจิสเตอร์ PB ที่มีสายสัญญาณ 8 เส้น และรีจิสเตอร์ PD จะมีสายสัญญาณเพียง 7 เส้น ส่วนอินพุต / เอาต์พุต PA และ PC จะมีอยู่ในชิปที่มีขาสัญญาณมากกว่านี้ (เช่น ชิป AT90S4414 หรือ AT90S8515)

ตารางที่ 2.3 แสดงตำแหน่งพื้นที่รีจิสเตอร์อินพุตเอาต์พุต

Address Hex	Name	Function
$3F ($5F)	SREG	Status Register
$3D ($5D)	SPL	Stack Pointer Low
$3B ($5B)	GIMSK	General Interrupt Mask Register
$3A ($5A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt Mask Register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag Register
$35 ($55)	MCUCR	MCU general Control Register
$33 ($53)	TCCR0	Timer/Counter 0 Control Register
$32 ($52)	TCNT0	Timer/Counter 0 (8-bit)
$2F ($4F)	TCNT0	Timer/Counter 1 Control Register A
$2E ($4E)	TCCR1B	Timer/Counter 1 Control Register B
$2D ($4D)	TCNT1H	Timer/Counter 1 High Byte
$2C ($4C)	TCNT1L	Timer/Counter 1 Low Byte
$2B ($4B)	OCR1AH	Output Compare Register 1 High Byte
$2A ($4A)	OCR1AL	Output Compare Register 1 Low Byte
$25 ($45)	ICR1H	T/C 1 Input Capture Register High Byte
$24 ($44)	ICR1L	T/C 1 Input Capture Register Low Byte
$21 ($41)	WDTCR	Watchdog Timer Control Register
$1E ($3E)	EEAR	EEPROM Address Register
$1D ($3D)	EEDR	EEPROM Data Register
$1C ($3C)	EECR	EEPROM Control Register

ตารางที่ 2.4 ตำแหน่งพื้นที่รีจิสเตอร์อินพุตเอาต์พุต (ต่อ)

Address Hex	Name	Function
$18 ($38)	PORTB	Data Register, Port B
$17 ($37)	DDRB	Data Direction Register, Port B
$16 ($36)	PINB	Input Pins, Port B
$12 ($32)	PORTD	Data Register, Port D
$11 ($31)	DDRD	Data Direction Register, Port D
$10 ($30)	PIND	Input Pins, Port D
$0C ($2C)	UDR	UART I/O Data Register
$0B ($2B)	USR	UART Status Register
$0A ($2A)	UCR	UART Control Register
$09 ($29)	UBRR	UART Baud Rate Register
$08 ($28)	ACSR	Analog Comparator Control and Status Register

หน่วยความจำข้อมูลแบบ EEPROM

ภายในไมโครคอนโทรลเลอร์ จะมีหน่วยความจำข้อมูลประเภท EEPROM บรรจุอยู่ ซึ่งจะมีขนาดแตกต่างกันตามเบอร์ไอซี โดยหน่วยความจำชนิดนี้จะถูกจัดให้แยกออกมาต่างหากจากพื้นที่ของหน่วยความจำทั่วไป และสามารถที่จะลบและเขียนใหม่ได้นับพันครั้ง

ในการเข้าถึงข้อมูลภายในรีจิสเตอร์ EEPROM ที่อยู่ในพื้นที่อินพุต/เอาต์พุต จะใช้คำสั่ง IN และ OUT ในการระบุรีจิสเตอร์ชี้ตำแหน่งของ EEPROM ,รีจิสเตอร์ข้อมูล EEPROM และรีจิสเตอร์ที่ใช้ควบคุมการทำงานของ EEPROM ในส่วนของรีจิสเตอร์ชี้ตำแหน่งของ EEPROM จะเป็นการระบุตำแหน่งพื้นที่ว่างของ EEPROM ที่จะทำการเขียน/อ่านข้อมูล ดังแสดงในรูปที่ 2.11 ในการอ้างถึงตำแหน่งข้อมูลภายใน EEPROM นี้จะมีลักษณะเป็นแบบเชิงเส้น ซึ่งจะมีค่าตั้งแต่ 0 จนถึง ตำแหน่งสุดท้ายของ EEPROM

EEPROM Address Register - EEAR

Bit	7	6	5	4	3	2	1	0	
$ 1E ($3E)	-	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
Read / Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

รูปที่ 2.11 รีจิสเตอร์ชี้ตำแหน่ง EEPROM

สำหรับขั้นตอนการเขียนข้อมูลลงใน EEPROM ทำได้โดยการเขียนข้อมูลที่ต้องการเก็บไว้ใน EEPROM ลงไปในรีจิสเตอร์ EERD จากนั้นทำการระบุตำแหน่งที่ต้องการเก็บข้อมูลลงในรีจิสเตอร์ EEAR สำหรับขั้นตอนการอ่านข้อมูลจาก EEPROM นั้น สามารถอ่านข้อมูลได้จากรีจิสเตอร์ EERD ซึ่งจะถูกอ่านออกมาจาก EEPROM จากการระบุตำแหน่งของรีจิสเตอร์ EEAR รูปที่ 2.12 แสดงรายละเอียดของรีจิสเตอร์ EERD

EEPROM Data Register - EEDR

Bit	7	6	5	4	3	2	1	0	
$ 1D ($3D)	MSB							LSB	EEDR
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

รูปที่ 2.12 รีจิสเตอร์ข้อมูลของ EEPROM

ในการอ่าน/เขียนข้อมูลจะมี 2 บิต ภายในรีจิสเตอร์ควบคุมที่เกิดการเปลี่ยนแปลง ในรูปที่ 2.13 แสดงถึงรายละเอียดของรีจิสเตอร์ควบคุม EECR เมื่อมีการโหลดข้อมูลเข้ามาที่รีจิสเตอร์ตำแหน่งและรีจิสเตอร์ข้อมูลอย่างถูกต้องแล้ว จะต้องเซตสัญญาณอีนาเบล (Enable) ที่บิต EEWE ให้เป็น 1 ตลอดช่วงของการเขียนข้อมูล (ใช้เวลา 2.5 ms ที่ Vcc = 5 V หรือ 4 ms ที่ Vcc = 2.7 V) เมื่อทำการเขียนข้อมูลเสร็จแล้วบิตนี้จะถูกเครียร์ เป็น 0 โดยฮาร์ดแวร์ ดังนั้นเราสามารถตรวจสอบที่บิตนี้ได้โดยกระบวนการทางซอฟแวร์โดยรอจนกว่าบิตจะถูกเคลียร์เป็นศูนย์สำหรับในการเขียนข้อมูลไบต์ถัดไป

EEPROM Control Register - EECR

Bit	7	6	5	4	3	2	1	0	
$ 1C ($3C)	-	-	-	-	-	EEMWE	EEWE	EERE	EECR
Read / Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

รูปที่ 2.13 รีจิสเตอร์ควบคุม EECR

ส่วนการอ่านข้อมูล เมื่อมีตำแหน่งแอดเดรสและข้อมูลโหลดเข้ามาจะต้องมีสัญญาณอีนาเบลในการอ่าน EERE เกิดขึ้น โดยจะถูกเซตเป็น 1 และเมื่อทำการอ่านข้อมูลเรียบร้อยแล้วบิตนี้จะถูกเครียร์เป็น 0 ด้วยกระบวนการทางฮาร์ดแวร์ ในการอ่านข้อมูลจะเกิดขึ้นที่รีจิสเตอร์ EEDR ซึ่งขั้นตอนในการอ่านข้อมูลจาก EEPROM นั้นจะใช้เวลาในการเข้าถึงข้อมูลภายในเวลาเพียง 1 คล็อกเท่านั้น จึงไม่จำเป็นที่จะต้องตรวจสอบที่บิต EERE ในส่วนของโปรแกรมตัวอย่าง ได้แสดงถึงการเขียนข้อมูลเข้าไปยังหน่วยความจำ EEPROM โดยช่วงเวลาในการเขียนข้อมูลจะขึ้นอยู่กับค่าของแห่งจ่ายไฟ ซึ่งจะอยู่ในช่วง 2.5-4 ms อย่างไรก็ตามเราสามารถที่จะเขียนโปรแกรมตรวจสอบได้โดยการตรวจสอบที่บิต EEWE จากโปรแกรมตัวอย่างจะต้องรอจนกว่าบิตนี้มีค่าเป็น 0 ถึงจะสามารถเขียนข้อมูลตัวถัดไปได้หรือไปทำคำสั่งอื่นๆ ต่อไป ข้อมูล $5A ควรจะถูกเก็บไว้ที่ตำแหน่งแอดเดรสที่ $20 ในหน่วยความจำ EEPROM

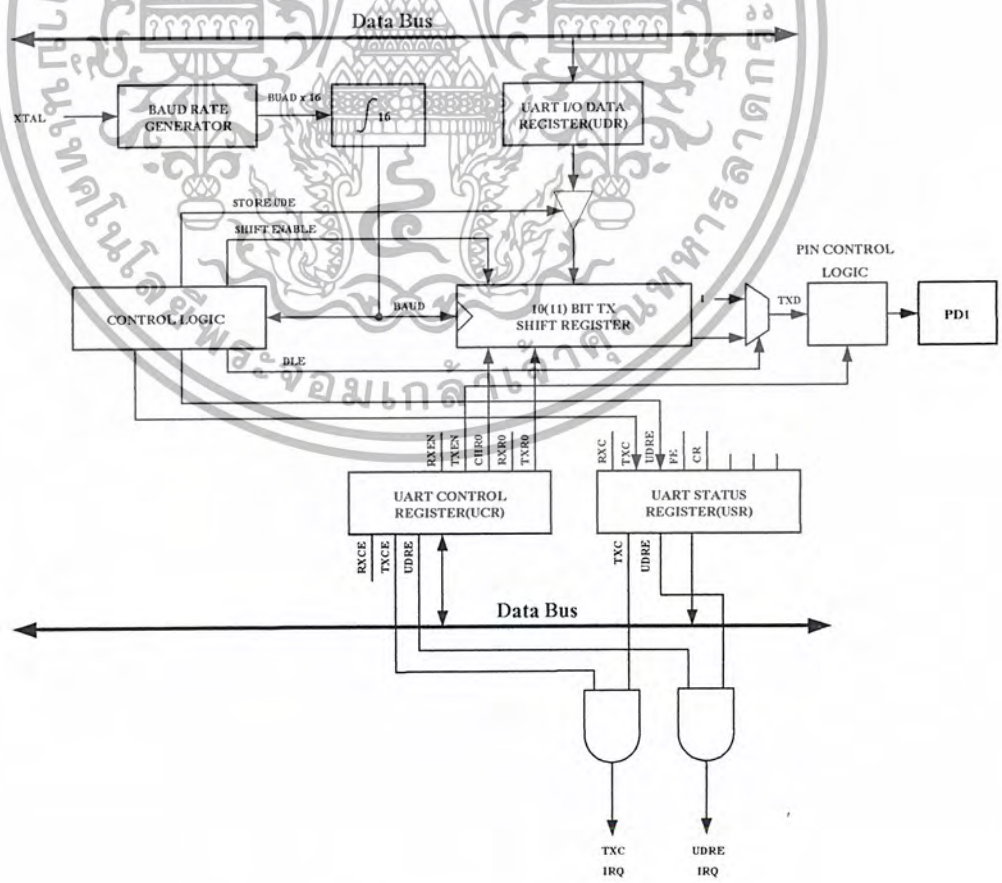
```
      ldi   temp,$20          ;Set EEPROM address $20
      out   EEAR,temp
      ldi   temp,$5A          ;Set EEPROM data $5A
      out   EEDR,temp
      ldi   temp,$02          ;Set EEPROM Write Enable
      out   EECR,temp
loop:in     temp,EECR         ;Read EEPROM Control Register
      sbrc  temp,1            ;Skip if EEWE bit is cleared
      rjmp  loop              ;Wait until EEWE is cleared
      nop                     ;Further instructions out of the
loop
```

2.1.4 การสื่อสารผ่านทางพอร์ตอนุกรม UART

การใช้งาน UART จะมีเฉพาะภายใน CPU model 1 มีลักษณะเป็นแบบ full- duplex ไมโครคอนโทรลเลอร์ AT90S4434/8535 ให้มีฟังก์ชั้น โดยมีคุณสมบัติดังนี้

1. สามารถเปลี่ยนแปลง BAUD RATE ของการสื่อสารได้หลาย BAUD RATE
2. สามารถสื่อสารได้ในอัตรา BAUD RATE ที่สูงในขณะที่ความถี่ XTAL ต่ำ
3. สื่อสารข้อมูลได้ทั้ง 8 บิต และ 9 บิต
4. มีส่วนของการกำจัดสัญญาณรบกวน
5. ตรวจสอบการผิดพลาดของการสื่อสารข้อมูล
6. ตรวจจับความผิดพลาดของบิตเริ่มต้น
7. จัดให้มีการแยกอินเตอร์รัพท์ของการสื่อสาร
8. มีบัฟเฟอร์ในการเก็บข้อมูล

การส่งข้อมูล จากรูปที่ 2.14 แสดงให้เห็นถึงส่วนประกอบต่างๆ ที่ใช้ในการส่งข้อมูล



รูปที่ 2.14 ภาคส่งของ UART

การส่งข้อมูลจะถูกเริ่มต้นโดยการเขียนข้อมูลที่ต้องการส่งไปยังรีจิสเตอร์ UDR (UART I/O Data register) ข้อมูลจาก UDR จะถูกส่งเข้าไปยังชิฟรีจิสเตอร์ เพื่อทำการส่งข้อมูลออกไปแบบอนุกรม ซึ่งชิฟรีจิสเตอร์นี้จะอยู่ในสถานะว่างก็ต่อเมื่อส่งบิตสุดท้ายของข้อมูลออกไปนั่นก็คือ stop bit ซึ่งจะสามารถโหลดข้อมูลใหม่เข้ามาได้ทันที

หลังจากที่ข้อมูลถูกส่งจาก UDR ผ่านชิฟรีจิสเตอร์ออกไป บิต UDRE (UART Data Register Empty) ที่อยู่ภายในรีจิสเตอร์สถานะ USR (UART Status register) จะถูกเซต นั่นหมายความว่าเมื่อบิตนี้ถูกเซตแสดงว่า UART พร้อมจะรับข้อมูลตัวถัดไป

2.2 การรับส่งข้อมูลแบบอนุกรม (RS - 232)

การรับส่งข้อมูลแบ่งเป็น 2 ประเภทใหญ่ ๆ คือการส่งข้อมูลแบบขนานและการส่งข้อมูลแบบอนุกรม

การส่งข้อมูลแบบขนาน คือการส่งข้อมูลที่ละไบต์ (8 บิต) เป็นวิธีที่เร็วแต่ถูกรบกวนได้ง่ายจากสัญญาณรบกวนต่าง ๆ และมีราคาแพง เนื่องจากใช้ปริมาณสายส่งมากจึงไม่นิยมใช้ส่งข้อมูลในระยะทางไกล ๆ

การส่งข้อมูลแบบอนุกรม คือ การส่งข้อมูลทีละ 1 บิต แต่สามารถรับส่งข้อมูลที่ละหลาย ๆ บิตได้ โดยจะต้องตกลงกันระหว่างตังส่งและตัวรับก่อนว่าจะรับส่งข้อมูลที่ละกี่บิต ตัวรับจะต้องรอข้อมูลให้ครบทุกบิตเสียก่อน จึงจะทำการประมวลผล ส่งผลให้การรับส่งข้อมูลแบบอนุกรมช้ากว่าแบบขนาน แต่มีผลกระทบจากสัญญาณรบกวนเพียงเล็กน้อยและประหยัดสายส่งเหมาะกับการส่งข้อมูลระยะไกล

การส่งข้อมูลแบบอนุกรมนั้นแบ่งออกเป็น แบบซิงโครนัสและแบบอะซิงโครนัส การสื่อสารแบบซิงโครนัสนั้นต้องอาศัยสัญญาณนาฬิการ่วมอยู่กับการรับส่งข้อมูลด้วย ดังนั้นการรับส่งข้อมูลแบบซิงโครนัส จะต้องอาศัยสายอย่างน้อย 3 เส้นคือ สัญญาณนาฬิกา สายข้อมูล และสายกราวด์

การสื่อสารข้อมูลแบบอะซิงโครนัส คือ การรับส่งโดยไม่ต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่านาฬิกา ทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในการกำหนดค่านี้ว่า “Baud Rate” มีหน่วยเป็นบิตต่อวินาที ค่า Baud Rate มาตรฐานที่ใช้ในการส่งแบบอนุกรมคือ 110 , 150 , 300 , 600 , 1200, 2400 , 4800 , 9600 และ 19200 Bit / Second และมีค่ามากขึ้นไปตามเทคโนโลยีที่เปลี่ยนไป

รูปแบบของข้อมูลที่ใช้ในการรับ-ส่ง แบบอะซิงโครนัส ประกอบด้วย 4 ส่วนคือ

1. บิตเริ่มต้น (Start Bit) ซึ่งจะมีขนาด 1บิต
2. บิตแบบอนุกรมซึ่งจะมีขนาด 8 บิต
3. พาริตี้บิต ขนาด 1 บิต (มีหรือไม่มีก็ได้)
4. บิตปิดท้าย (Stop Bit) จะมีขนาด 1หรือ 2 บิต

2.3 ทฤษฎีอินฟราเรดรีโมทคอนโทรล

การดำรงชีวิตของเรานั้น ต้องการสิ่งอำนวยความสะดวกในรูปแบบของเครื่องใช้ไฟฟ้าอันทันสมัยมากมาย ซึ่งการควบคุมเครื่องใช้ไฟฟ้าเหล่านั้นนิยมใช้รีโมทคอนโทรล เนื่องจากใช้งานง่าย เครื่องส่งใช้กำลังไฟน้อยและราคาไม่แพงเกินไป

แสงอินฟราเรด หมายถึง แสงที่มีความยาวคลื่น 950 nm. เป็นแสงปกติที่มีสีเฉพาะเจาะจงและสายตามนุษย์ไม่สามารถมองเห็นได้

อินฟราเรดรีโมทคอนโทรล หมายถึง การควบคุมอุปกรณ์ระยะไกล ในระยะที่มองเห็นโดยใช้ลำแสงอินฟราเรด สาเหตุที่ใช้อินฟราเรดสำหรับทำรีโมทคอนโทรล คือ ไม่สามารถมองเห็นลำแสงขณะใช้งานและอินฟราเรดมีราคาถูกและทำได้ง่าย

ปัจจุบันอินฟราเรดรีโมทคอนโทรล ซึ่งทำหน้าที่เสมือนแขนขาของมนุษย์ จึงทำให้เป็นที่ใช้กันอย่างแพร่หลาย และเริ่มเป็นสิ่งจำเป็นควบคู่กับเครื่องใช้ไฟฟ้าในปัจจุบัน จึงสมควรที่จะให้ความสนใจกันอย่างละเอียด

การทำงานของระบบดาต้าโค้ด

ตามที่ทราบมาแล้วว่า ระบบตัวเลขที่ใช้รีโมทคอนโทรลเป็นระบบตัวเลขฐาน 2 ซึ่งมีสภาพทางลอจิก “0” และ “1” หรือการพูดถึงระดับแรงไฟสภาพลอจิก 1 ก็คือสภาพที่แรงดันไฟใกล้เคียง 0 โวลต์ นั่นคือระบบสัญญาณพัลซ์นั่นเอง

กล่าวเฉพาะในส่วนของข้อมูลหลาย ๆ บิต การจะแยกออกมาอย่างชัดเจนว่าอันไหนเป็นลอจิก 0 อันไหนเป็นลอจิก 1 จำเป็นต้องใช้ฐานเวลาเข้ามาเป็นตัวแยก ในทางปฏิบัติเป็นเรื่องค่อนข้างยากอยู่พอประมาณ เนื่องจากบางครั้งผู้ใช้รีโมทอาจจะกดช้านานแต่ละครั้งอาจจะไม่เท่ากัน บางคนกดแช่ไว้ บางคนกดคีย์แล้วปล่อยเลย

ดังนั้นเราจึงเลือกเอาระบบสแกนพัลซ์ ซึ่งเป็นรูปแบบของการกวาดข้อมูลมาทดแทน เพื่อแก้ปัญหาเรื่องการที่จะต้องกำหนดฐานเวลา โดยหากเป็นลอจิก 0 เราจะให้พัลซ์ที่ออกมามีพัลซ์ชิดหรือแคบกว่า และหากเป็นลอจิก 1 เราจะให้ระยะของพัลซ์ทอดยาวออกไป การกำหนดฟังก์ชั่น

ทำได้โดยการใช้หลักการของคีย์แบบเมตริกซ์ (Matrix) เพื่อลดขาใช้งานของไอซีและสายเชื่อมโยงต่าง ๆ ให้น้อยลง

เมื่อเรากำหนดพัลซ์ในลักษณะที่กล่าวมา มีผลทำให้ฐานเวลาของข้อมูล (Data/Time) แต่ละตัวมีความแตกต่างกันออกไป ยกตัวอย่างเช่น ในกรณีที่ข้อมูลนั้น ๆ มากกว่าจะมีผลทำให้เวลาของข้อมูลยาวนานขึ้น ดังนั้นในยุคปัจจุบันที่มีการใช้ข้อมูลหลาย ๆ บิต จะเกิดปัญหาในเรื่องความเหลื่อมของเวลาขึ้น ปัญหานี้มีผลต่อการรับข้อมูลในเครื่องรับเป็นอันมาก รวมไปถึงหากเราจะใส่ข้อมูลอย่างอื่นฝากไปด้วยจะทำได้ลำบาก เราจะพบว่ารีโมทคอนโทรลในปัจจุบันนี้สามารถผ่านข้อมูลพิเศษเข้าไป
ได้มากทีเดียว เครื่องรับจะแยกได้อย่างไรหากเวลาไม่แน่นอน จึงต้องมีการอินเวิร์ต(Invert) ข้อมูล

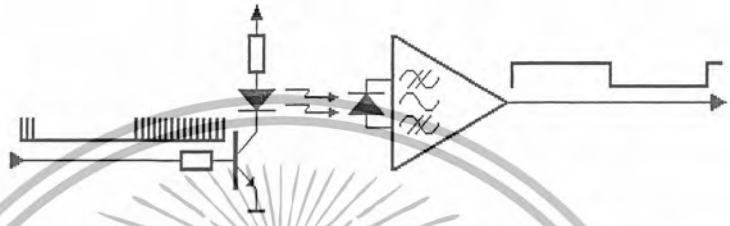
รูปที่ 2.15 ลักษณะทางลอจิกในระบบสแกนพัลซ์

ทั้งหมดให้กลายเป็นตรงกันข้ามเพื่อรักษาเวลาให้คงที่ และเพื่อไม่ต้องส่งสัญญาณซิงโครไนซ์ไปควบคุมเครื่องรับ ซึ่งนับเป็นความซับซ้อมยุ่งยาก

ยกตัวอย่างเช่น เราส่งข้อมูลหรือดาต้าโค้ดเป็นข้อมูลขนาด 6 บิต เป็นดังนี้ 000001 เราจะพบว่าเมื่อถูกแปรสภาพเป็นพัลซ์แล้ว ลอจิก 0 มีจำนวน 5 บิต ลอจิก 1 มีจำนวน 1 บิต จะพบว่าเวลารวมของพัลซ์สั้นมาก และหากเราส่งข้อมูลเป็น 111111 จะพบว่าเวลารวมของพัลซ์จะยาวมากทีเดียว แต่ถ้าหากเรานำมาอินเวิร์ต จะพบว่าเวลารวมทั้งหมดเท่ากัน นั่นคือ ข้อมูล 000001 เมื่ออินเวิร์ตแล้ว จะเป็น 111110 ข้อมูลรวมคือ 000001111110 (แยกเป็นลอจิก 0 รวม 6 บิต ลอจิก 1 รวม 6 บิต) พบว่าระยะเวลาของข้อมูลจะเท่ากันโดยอัตโนมัติ ดังนั้น วิธีการอินเวิร์ตข้อมูล คือการแก้ปัญหาฐานเวลาที่เกิดขึ้นกับเครื่องรับหรือตัวรับนั่นเอง

2.3.1 การมอดูเลต (Modulation)

การมอดูเลต คือการนำเอาสัญญาณขี่ไปบนสัญญารบกวน(Noise) โดยการมอดูเลตจะใช้ความถี่เฉพาะที่เกิดจากแสงของอินฟราเรด และตัวอินฟราเรดด้านรับ จะต้องปรับให้มีความถี่ตรงกัน ดังนั้นจึงสามารถไม่ยอมรับความถี่อื่น ๆ ได้



รูปที่ 2.16 การมอดูเลตและการตรวจจับสัญญาณ(detects)อินฟราเรด

การติดต่อสื่อสารของมนุษย์โดยทั่ว ๆ ไปมี 2 อย่างคือ การใช้คำพูด และ การไม่ใช้คำพูด แต่ในทางการสื่อสารคมนาคม จะมีลักษณะเป็น " mark " และ " space " โดย " space " คือสัญญาณปกติ มีสถานะการส่งปิด ซึ่งไม่มีการส่งสัญญาณแสงอินฟราเรดในสถานะนี้ และ " mark " เป็นสถานะการส่งสัญญาณแสงอินฟราเรด โดยส่งเป็นพัลซ์ " on " และ " of " ตามความถี่เฉพาะนั้น ๆ โดยทั่ว ๆ ไปของผู้ผลิตจะใช้ความถี่ 30 kHz และ 60 kHz

ทางด้านรับ " space " แสดงว่า เอาท์พุท เป็น Hight level และ "mark" แสดงว่า เอาท์พุท เป็น Low level แต่จำไว้ว่า " mark " และ " space " ไม่ใช่ 1 และ 0 ที่ต้องการส่ง แต่ความสัมพันธ์จริง ๆ ระหว่าง 1 และ 0 ถูกกำหนดโดยโปรโตคอลที่เราใช้

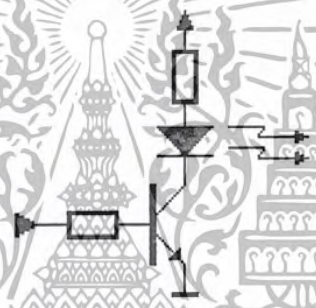
2.3.2 ตัวส่งสัญญาณ (Transmitter)

โดยปกติตังส่งจะใช้พลังงานแบตเตอรี่ ซึ่งมีกำลังไม่สูงมากนัก แต่สัญญาณอินฟราเรดก็มีความเข้มเพียงพอที่จะส่งเป็นระยะทางที่สามารถควบคุมอุปกรณ์นั้น ๆ โดยมีความทนทานต่อการสั่นสะเทือนได้เป็นอย่างดี

ชิป(chips) หลายตัวได้ถูกออกแบบมาใช้เป็นตัวส่งสัญญาณอินฟราเรด ชิปรุ่นเก่าใช้โปรโตคอลเพียงตัวเดียวจากหลายโปรโตคอลที่มี ในปัจจุบันไมโครคอนโทรลเลอร์ มีกำลังงานต่ำมาก ๆ ได้ถูกนำมาใช้เป็นตัวส่งสัญญาณอินฟราเรด เพราะมีความยืดหยุ่นในการใช้งานสูง ซึ่งเมื่อไม่มีการกดปุ่ม กำลังงานจะน้อยมาก ๆ แทบจะไม่มีกระแสเลยแต่เครื่องส่งจะทำงานเมื่อมีการกดปุ่มเกิดขึ้นเท่านั้น

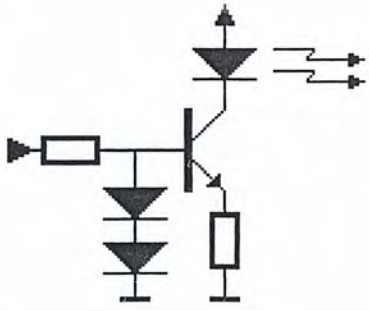
ผลึก Crystals นาน ๆ ครั้งจึงถูกใช้ทำงาน ซึ่งจะมีความบอบบางมาก ๆ และหยุดการทำงานได้ง่ายเมื่อเครื่องหยุดใช้งาน ดังนั้นเซรามิกส์จึงมีความเหมาะสมมากกว่า เพราะโดยทางกายภาพแล้วจะความทนทานต่อการสั่นสะเทือนสูง

กระแสที่ไหลผ่าน LED สามารถเปลี่ยนแปลงจาก 100 mA ถึง มากกว่า 1 A เพื่อที่จะส่งสัญญาณไปควบคุมได้ในระยะทางไกล ๆ กระแสของ LED ต้องสูง แต่ต้องแลกกันระหว่างตัว LED , อายุการใช้งานของแบตเตอรี่ กับระยะทางสูงสุดในการควบคุม กระแสของ LED สามารถทำให้สูงได้เพราะตัวขับพัลซ์ของ LED สั้นมาก ๆ กำลังงานเฉลี่ยของ LED ไม่ควรเกินค่าสูงสุดที่ผ่านได้ ซึ่งเห็นได้จากกระแสสูงสุดของ LED ไม่ได้สูงเกินไป โดยค่าต่าง ๆ เหล่านี้สามารถดูจากคุณสมบัติของ LED



รูปที่ 2.17 วงจรส่งสัญญาณ LED

วงจรโดยทั่วไปของทรานซิสเตอร์ สามารถใช้เป็นตัว ส่ง LED ได้ ทรานซิสเตอร์ซึ่งมีคุณสมบัติ HFE และความเร็วของสวิตชิ่งจึงสามารถใช้ได้ตามจุดประสงค์นี้ โดยค่าความต้านทานหาได้โดยใช้กฎของโอห์มและค่าแรงดันตกคร่อมต่ำสุดที่อินฟราเรด LED ประมาณ 1.1 V



รูปที่ 2.18 วงจรส่งสัญญาณ LED

จากวงจรรูปที่ 2.17 มีข้อเสียคือมีแรงดันตกคร่อมแบตเตอรี่ ทำให้กระแสที่ผ่าน LED ลดลง ทำให้ระยะทางในการควบคุมต่ำลง แก้ไขโดยการนำเอาไดโอด 2 ตัวมาอนุกรมกัน ดังรูปที่ 2.18 ทำให้ จำกัดจำนวนพัลซ์ที่ขาเบส(base) ของทรานซิสเตอร์ที่ 1.2 V. แรงดันที่ตกคร่อมที่ Vbe (base-emitter voltage) เท่ากับ 0.6 V. ผลคือทำให้ขาอีมิตเตอร์(emitter) มีแอมพลิจูด(amplitude)คงที่ 0.6 V. ผ่านรีซิสเตอร์(resister) ค่าคงที่ ผลที่ได้คือพัลซ์ที่แมกนิจูด(magnitudede) คงที่และค่ากระแสที่ผ่าน LED หาได้จากกฎของโอห์ม

2.3.3 ตัวรับสัญญาณ (Receiver)

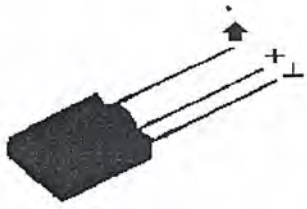
วงจรเครื่องรับมีมากมาย แตกต่างกันไป แต่กฎเกณฑ์ในการเลือกที่สำคัญที่สุดคือความถี่ที่ใช้ในการมอดูเลตและความต้องการในการนำไปใช้ประโยชน์นั้น ๆ

รูปที่ 2.19 Block Diagram ของภาครับสัญญาณอินฟราเรด

เมื่อรับสัญญาณอินฟราเรด โดยมี อินฟราเรดไดโอดเป็นตัวตรวจจับ(detect)สัญญาณและสัญญาณนี้จะถูกขยายและจำกัดโดยใช้วงจร AGC เพื่อให้ระดับพัลซ์คงที่โดยไม่คำนึงถึงระยะทาง

สัญญาณ AC จะถูกส่งผ่านวงจร Band Pass Fillter และวงจรจะปรับความถี่ในการมอดูเลต โดยทั่วไปความถี่ที่ผู้ผลิตใช้กันคือ 30 kHz. ถึง 60 kHz. ขั้นตอนต่อมาคือ detector , integrator และ comparator จุดประสงค์ของ 3 ขั้นตอนนี้คือต้องการตรวจจับความถี่ของการมอดูเลต ถ้าความถี่มอดูเลตนี้ออกที่เอาท์พุทของ comparator จะมีความถี่ต่ำ

จากที่กล่าวข้างต้น ทุกบล๊อกจะถูกรวมเอาไว้ในอุปกรณ์อิเลคทรอนิกส์ เพียงตัวเดียว เนื่องจากมีโรงงานผลิตอุปกรณ์ชนิดนี้มากมาย แตกต่างกันไปและส่วนใหญ่จะมีหลายเบอร์โดยมีการปรับความถี่มอดูเลตที่เฉพาะเจาะจง



รูปที่ 2.20 อุปกรณ์ Receiver

2.3.4 สัญญาณในส่วนต่างๆของรีโมทคอนโทรล

การที่เราจะให้วงจรภาครับของรีโมทคอนโทรลรับรู้ และทำการแยกสัญญาณหรือข้อมูลต่าง ๆ ได้อย่างถูกต้อง มีข้อมูลตัวอื่นเข้ามาเพื่อแก้ไขสิ่งซึ่งอาจผิดพลาด ดังนั้น นอกจากข้อมูล (Data) ซึ่งเป็นข้อมูลหลักที่เราส่งไปเป็นรหัสตัวเลขฐาน 2 ซึ่งตอนนี้อยู่ในรูปของ สแกนพัลซ์ พร้อมด้วยการอินเวิร์ตข้อมูล เพื่อรักษาค่าเวลาแล้วยังต้องมีข้อมูลอื่นเป็นส่วนประกอบ ซึ่งในกรณีนี้เราขอใช้ตัวอย่างของเครื่องเนชั่นแนลเป็นตัวอย่างข้อมูล (ซึ่งโดยหลักการจะเหมือนกันทุกยี่ห้อ เพียงแต่ว่าใครจะใช้ข้อมูลกี่บิต และใช้ฐานเวลาเท่าไรนั่นเอง) นี่คือการยกตัวอย่างเพื่อนำไปสู่การทำความเข้าใจ และทำการเปรียบเทียบกับยี่ห้ออื่น ๆ ในเวลาต่อไป

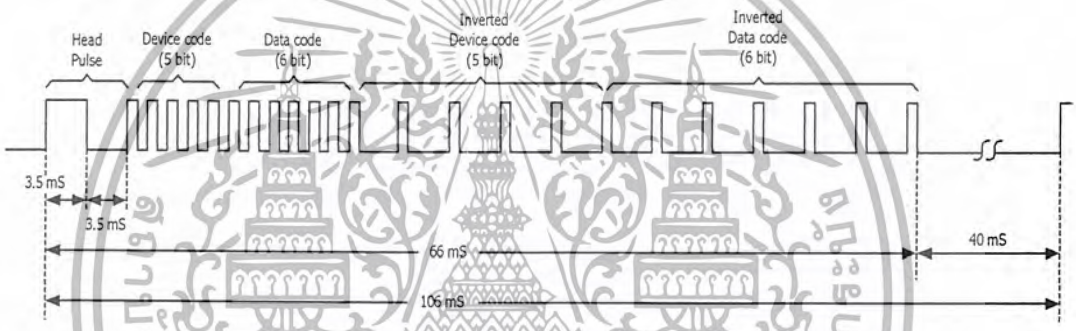
1. ดาต้าโค้ด (Data Code) ในกรณีของรีโมทคอนโทรล ซึ่งเป็นตัวส่งของเนชั่นแนลใช้ข้อมูลขนาด 6 บิต เป็นข้อมูลหลักที่จะส่งออกไปควบคุมวงจรในส่วนภาครับ โดยคีย์แมทตริกซ์จะเป็นตัวสั่งงานเข้าสู่ระบบการเข้ารหัสข้อมูล กำหนดความเป็นไปของแต่ละฟังก์ชั่น

2. อินเวิร์ตดาต้าโค้ด (Device Data Code) เป็นการกลับลอจิกของข้อมูลหลักเพื่อรักษาฐานเวลาให้คงที่ทุกข้อมูล ซึ่งเป็นข้อมูลขนาด 6 บิตเหมือนกัน

3. ดีไวส์โค้ด (Device Code) หรือบางครั้งใช้โทรศัพท์ "คัสตอมโค้ด" (Custom Code) เนื่องจากในปัจจุบันเครื่องใช้ไฟฟ้าต่าง ๆ ล้วนเป็นระบบรีโมทคอนโทรลแบบอินฟราเรดกันทั้งนั้น การสั่งงานจากรีโมทคอนโทรลอาจจะมีการคลื่นไปรบกวนอุปกรณ์อื่น ๆ ได้ อย่างเช่น เครื่องรับโทรทัศน์อาจจะพ่วงอยู่กับวีดีโอเทป จูนเนอร์พ่วงอยู่กับเครื่องขยายเสียงและคอมแพคดิสก์ เลเซอร์ดิสก์พ่วงอยู่กับเครื่องรับโทรทัศน์ หรืออื่น ๆ กรณีเช่นนี้หากเราสั่งเครื่องหนึ่งเครื่องใด เครื่องที่ต่อร่วมอยู่ด้วยก็สามารถรับเอาข้อมูลฟังก์ชั่นการทำงานเข้าไปด้วย จึงมีการสร้างข้อมูลของเครื่องเล่นแต่ละอย่างให้แตกต่างกันออกไป ตัวอย่างเช่น เครื่องรับโทรทัศน์ เราใช้ดีไวส์โค้ด (Device Code) ซึ่งเป็นโค้ดที่ใช้แยกประเภทของเครื่องใช้ด้วยระบบข้อมูล 00000 (5 บิต) ในขณะที่เครื่องเล่นวีดีโอเทป เราใช้ดีไวส์โค้ด 11111

4. อินเวิร์ตดีไวส์โค้ด (Inverse Device Code) เป็นการกลับข้อมูลดีไวส์โค้ด เพื่อรักษาเวลาเช่นเดียวกับระบบข้อมูลหลัก (Data Code) แน่นอนข้อมูลดังกล่าวต้องมี 5 บิต เหมือนดีไวส์โค้ด

5. เฮดพัลซ์ (Head Pulse) การกลับข้อมูลหรือการอินเวิร์ต เป็นเพียงการรักษาเวลาของข้อมูล แต่การใส่เฮดพัลซ์เป็นกรรมวิธีที่สามารถตรวจเช็คข้อมูลเพิ่มความแน่นอนของข้อมูลขึ้นอีก เพราะในบางครั้งเราอาจจะกดคีย์แช่ไว้นาน ๆ ความต่อเนื่องของข้อมูลจะมีตลอด นั่นหมายความว่าเราจะแยกแยะอย่างไรว่าอะไรคือข้อมูลหลัก อะไรคือข้อมูลรอง จึงมีเฮดพัลซ์ขึ้นมา โดยเฮดพัลซ์จะเป็นสัญญาณนำร่องก่อนจะมีข้อมูลต่าง ๆ ส่งออกมา และในขณะที่เราส่งข้อมูลอย่างต่อเนื่อง จะมีเฮดพัลซ์ส่งออกมาคั่นเป็นช่วง ๆ ให้เครื่องรับสามารถแยกกลุ่มข้อมูลออกมาได้



รูปที่ 2.21 ดาต้าสตรีม

สังเกตว่ารีโมทคอนโทรลจะส่งสัญญาณออกมาแต่ละคีย์ที่กดจะใช้เวลาที่เท่ากัน เนื่องจากมีการอินเวิร์ตดีไวส์โค้ด และอินเวิร์ตดาต้าโค้ด ลำดับดาต้าสตรีม (Data Stream) หรือระบบในการเรียงข้อมูลจะเป็นไปตามรูปที่ 2.21 โดยเฮดพัลซ์จะเข้ามาเป็นอันดับแรก เป็นการบอกว่าตอนนี้ออปชั่นภายในของตัวส่งรีโมทคอนโทรลแบบอินฟราเรด พร้อมที่จะทำงานแล้ว เป็นตัวตรวจสอบหรือตัวกระตุ้นความพร้อมของตัวรับว่า ต่อไปนี้จะมีการส่งข้อมูล

เมื่อเครื่องรับรู้การทำงาน (ซึ่งเมื่อเรากดคีย์ออกไป สัญญาณเฮดพัลซ์บางที่เราเรียกว่า ตัว “คอลล์” เครื่องรับจะแสดงการรับรู้การเรียกข้อมูล (Call) ด้วยการกระพริบของแอลอีดี และข้อมูลลำดับต่อมาก็คือดีไวส์โค้ด 5 บิต ตามด้วยดาต้าโค้ด 6 บิต อินเวิร์ตดีไวส์โค้ด 5 บิต และปิดท้ายด้วยอินเวิร์ตดาต้าโค้ด 6 บิต ก่อนทิ้งช่วงให้เกิดเฮดพัลซ์ครั้งต่อไป ให้ตัวรับสามารถรับรู้ความต่อเนื่องของข้อมูลอีกครั้งหรือหลาย ๆ ครั้งต่อไป

2.3.5 คุณสมบัติทางด้านเทคนิคของรีโมทคอนโทรลยี่ห้อต่างๆ

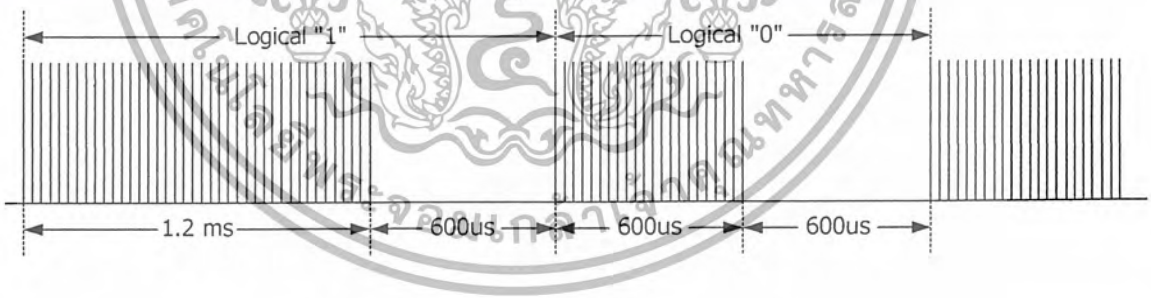
โปรโตคอล Sony

จากการศึกษาและเก็บข้อมูลของ Sony พบว่ามีโปรโตคอลอยู่ 3 รูปแบบ คือ 12 บิต , 15 บิต , 20 บิต ซึ่งในแต่ละรูปแบบจะแตกต่างกันที่จำนวนบิตที่ส่งต่อลำดับคำสั่ง ซึ่งในที่นี้จะกล่าวเฉพาะรูปแบบ 12 บิต

คุณสมบัติ

- มีโปรโตคอล 3 รปแบบ คือ 12 บิต , 15 บิต และ 20 บิต
- แอดเดรส 5 บิต และความยาวของคำสั่ง 7 บิต (โปรโตคอล 12 บิต)
- มอดูเลตตามความกว้างของพัลซ์ (Pulse width modulation)
- ความถี่ส่ง 40 kHz
- Bit time เท่ากับ 1.2 ms หรือ 0.6 ms

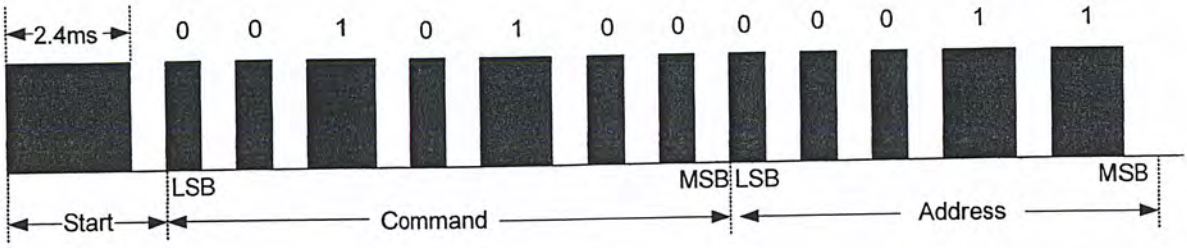
การมอดูเลต



รูปที่ 2.22 การมอดูเลตระหว่างลอจิก 1 และลอจิก 0

โปรโตคอล Sony ใช้ความกว้างของพัลซ์ในการเข้ารหัสบิต พัลซ์ที่แสดงถึง ลอจิก "1" จะใช้ระยะเวลา 12 ms จากความถี่ส่ง 40 kHz ขณะที่ลอจิก "0" จะใช้ระยะเวลา 0.6 ms ใช้การเปรียบเทียบของช่วงว่างของคาบเท่ากับ 0.6 ms ควรมี duty cycle ที่ 1/4 หรือ 1/3

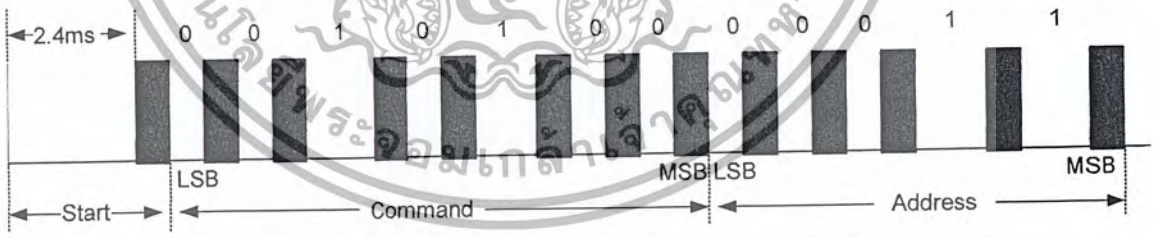
โปรโตคอล



รูปที่ 2.23 ตัวอย่างของการส่งข้อมูล

จากรูปแสดงถึงชุดพัลซ์ของโปรโตคอล Sony โปรโตคอลนี้เริ่มต้นการส่งที่ด้าน MSB ความกว้างของบิตเริ่มต้นเท่ากับ 2.4 ms ตามด้วยช่องว่างมาตรฐานขนาด 0.6 ms นอกจากที่สัญญาณของบิตเริ่มต้นจะเป็นการเริ่มต้นการส่งข่าวสารแล้ว ยังเป็นการปรับการขยายของอินฟราเรดด้านรับด้วย หลังจากบิตเริ่มต้นแล้ว จะเป็นแอดเดรสด้านส่งซึ่งมีขนาด 5 บิต ตามด้วยคำสั่งขนาด 7 บิต สำหรับกรณีนี้ แอดเดรส คือ 3 และส่งคำสั่ง 14 ออกไป คำสั่งจะส่งไปซ้ำ ๆ ทุก 45 ms (วัดจากเริ่มต้นถึงเริ่มต้น) ตราบเท่าที่ปุ่มของรีโมทคอนโทรลยังถูกกดอยู่

ถ้าหากเราใช้ตัวรับสัญญาณอินฟราเรดที่มีขายอยู่ตามท้องตลาดทั่วๆไป มาใช้ จะเห็นได้ว่ารูปคลื่นสัญญาณที่ปรากฏจะมีลักษณะที่กลับกัน (invert) กับสัญญาณจากภาคที่ส่งออกมาดังรูป



รูปที่ 2.24 สัญญาณที่ออกมาจากภาครับ

เราสามารถตรวจจับรหัสสัญญาณที่ส่งออกมาได้โดยใช้ไมโครคอนโทรลเลอร์ทำหน้าที่เป็นตัววิเคราะห์สัญญาณที่เข้ามา โดยทำตามขั้นตอนต่อไปนี้

1. กำหนดให้ var1 = 8 ,var2 =0
2. รอสัญญาณลงเป็น 0 - จะเป็นเฮดพัลซ์ (2.4ms)
3. รอสัญญาณขึ้นเป็น 1

4. รอสัญญาณลงเป็น 0 – รอดูข้อมูลว่าจะเป็นบิต 0 หรือ 1 โดย
5. หน่วงเวลา 800 us
6. วัดระดับของสัญญาณ
7. ถ้าสัญญาณเป็น "1" บิตที่รับมาเป็นลอจิก "0"
   - เซตบิต carry เป็นลอจิก 0
   - หมุนข้อมูล var1 ไปทางขวา (bit C --> MSB Var1)
   - หมุนข้อมูล var2 ไปทางขวา (bit C var1 --> MSB var2 , bit 0 var2 --> carry)
   - เช็ค carry bit : ถ้าเป็น "1" ไปทำข้อ (9) , ถ้าเป็น "0" ไปทำข้อ (4)

8. ถ้าสัญญาณเป็น "0" บิตที่รับมาเป็นลอจิก "1"
   - เซตบิต carry เป็นลอจิก 1
   - หมุนข้อมูล var1 ไปทางขวา (bit C --> MSB var1)
   - หมุนข้อมูล var2 ไปทางขวา (bit C Var1 --> MSB var2 , bit 0 var2 --> carry)
   - เช็คบิต carry : ถ้าเป็น "1" ไปทำข้อ (9) , ถ้าเป็น "0" ไปทำข้อ (3)

9. ตอนนี้มีข้อมูลครบทั้ง 12 บิต โดย
   var1 มี 8 บิต และ var2 มี 4 บิต
   บิตทางซ้ายมือ 5 บิตของ var1 เป็น แอดเดรส
   บิตทางขวา 3 บิต ของ var1 รวมกับ 4 บิตทางซ้ายของ var2 เป็นคำสั่ง

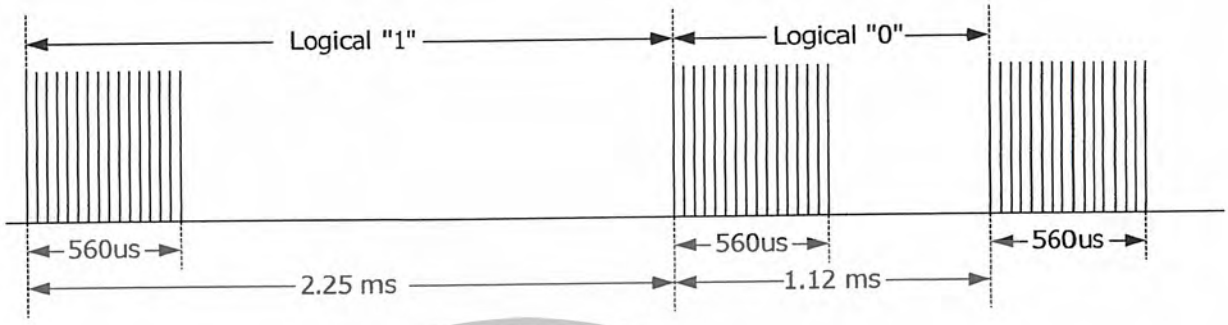
โปรโตคอลของ NEC

โปรโตคอลนี้ถูกพัฒนาโดยบริษัท NEC ซึ่งเป็นรูปแบบโปรโตคอลแบบญี่ปุ่น ซึ่งรูปแบบและคำอธิบายต่าง ๆ มาจากคู่มือของ VCR

คุณสมบัติ

- แอดเดรสขนาด 8 บิต และ คำสั่งขนาด 8 บิต
- เพื่อความถูกต้องและน่าเชื่อถือ แอดเดรสและคำสั่ง จะถูกส่งไป 2 ครั้ง
- มอดูเลตตามระยะห่างของพัลซ์ (Pulse distance modulation)
- ความถี่ส่ง 38 kHz
- Bit time เท่ากับ 1.12 ms หรือ 2.25 ms

การมอดูเลต



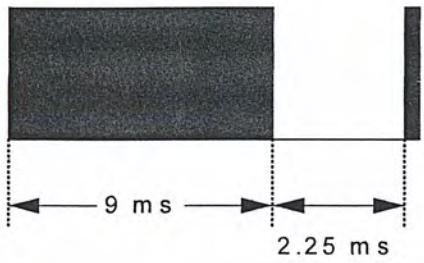
รูปที่ 2.25 การมอดูเลตระหว่างลอจิก 1 และลอจิก 0

โปรโตคอลของ NEC ใช้ระยะห่างของพัลซ์ในการเข้ารหัสของบิต แต่ละพัลซ์ยาว 560 us จากความถี่ส่ง 38 kHz (ประมาณ 21 รอบ) ลอจิก "1" ใช้เวลาส่ง 2.25 ms และลอจิก "0" ใช้เวลาส่ง1.12 ms ควรมี duty cycle ที่ 1/4 หรือ 1/3

โปรโตคอล

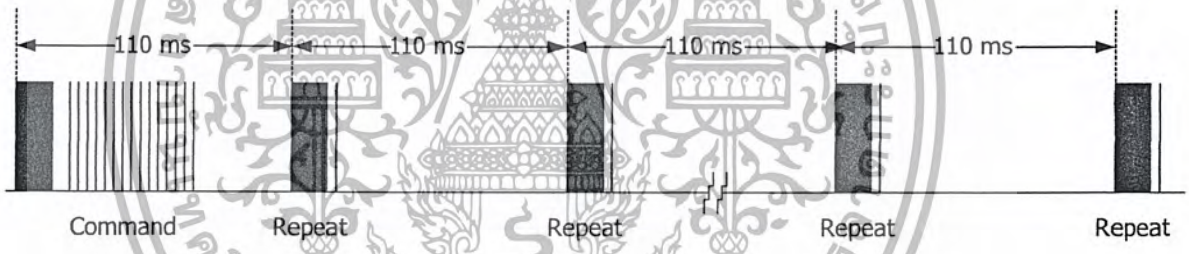
รูปที่ 2.26 ตัวอย่างการส่งข้อมูลของรีโมท NEC

จากรูปแสดงถึงชุดพัลซ์ของโปรโตคอล NEC โปรโตคอลนี้เริ่มต้นการส่งที่ด้าน LSB ในกรณีนี้ใช้แอดเดรสคือ $59 และคำสั่งคือ $16 ในการส่ง การส่งสัญญาณเริ่มต้น โดยใช้เวลา 9 ms ของ AGC burst ซึ่งถูกใช้เพื่อ เซตค่าการขยายของอินฟราเรดภาครับ AGC Burst นี้แล้วตามด้วยช่วงว่าง 4.5 ms หลังจากนั้นจะเป็นแอดเดรสและคำสั่ง ซึ่งแอดเดรสและคำสั่งจะถูกส่งไป 2 ครั้ง โดยครั้งที่ 2 จะเป็นค่าอินเวิร์ตและสามารถใช้เพื่อตรวจสอบความถูกต้องของข่าวสารที่ได้รับ เวลาโดยรวมของการส่งจะคงที่ เพราะว่าทุกบิตมีการส่งซ้ำตามความยาวของบิตอินเวิร์ต ถ้ารู้สึกว่าไม่น่าเชื่อถือ ก็ไม่ต้องสนใจค่าของอินเวิร์ตก็ได้ หรือสามารถขยายให้แต่ละแอดเดรสและคำสั่งมี 16 บิต



รูปที่ 2.27 เฮดพัลซ์ของรีโมท NEC

คำสั่งจะถูกส่งมาเพียงครั้งเดียวเท่านั้น ถึงแม้ว่ารีโมทคอนโทรลจะถูกกดอยู่ตลอดเวลา ในทุก ๆ 110 ms จะมีการส่งรหัสซ้ำ ถ้าปุ่มยังถูกกดอยู่ รหัสที่ส่งซ้ำเป็น พัลซ์ AGC 9 ms ตามด้วยช่อง 2.25 ms และ burst 560 us



รูปที่ 2.28 ช่วงเวลาของสัญญาณที่ส่งออกมาติดๆกัน

เราสามารถตรวจจับรหัสสัญญาณที่ส่งออกมาได้โดยใช้ไมโครคอนโทรลเลอร์ทำหน้าที่เป็นตัววิเคราะห์สัญญาณที่เข้ามา โดยทำตามขั้นตอนต่อไปนี้

1. เซตตัวแปร var1 = 128,var2 = 0,var3 = 0,var4 = 0
2. รอสัญญาณลง – เป็นการเริ่มส่งข้อมูล
3. รอสัญญาณขึ้น – ใช้เวลาประมาณ 9 ms
4. อยู่ในช่วงเวลาซิงค์ ใช้เวลาประมาณ 4.4 ms
5. รอสัญญาณลงเป็นลอจิก 0
6. เริ่มส่งบิตแรก (รอสัญญาณขึ้น)
7. หน่วงเวลา 800 us เพื่อดูว่าเป็นลอจิก 0 หรือ 1

8. วัดระดับสัญญาณ

- ถ้าระดับยังเป็น 0 อยู่ ข้อมูลที่รับมาเป็น 0

- ทำการเซต carry flag = 0

- หมุนตัวแปร var1 ไปทางขวา

- หมุนตัวแปร var2 ไปทางขวา

- หมุนตัวแปร var3 ไปทางขวา

- หมุนตัวแปร var4 ไปทางขวา

- เช็ค carry bit : ถ้าเป็น "1" ไปทำข้อ (10) , ถ้าเป็น "0" ไปทำข้อ (6)

- ถ้าระดับยังคงเป็น 1 อยู่ ข้อมูลที่รับมาเป็น 1

- ทำการเซต carry flag = 1

- หมุนตัวแปร var1 ไปทางขวา

- หมุนตัวแปร var2 ไปทางขวา

- หมุนตัวแปร var3 ไปทางขวา

- หมุนตัวแปร var4 ไปทางขวา

- เช็ค carry bit : ถ้าเป็น "1" ไปทำข้อ (9) , ถ้าเป็น "0" ไปทำข้อ (5)

9. ข้อมูลที่รับมามีทั้งหมด 32 บิต โดยแบ่งเป็น

var1 = Invert Command
var2 = Command
var3 = Invert Address
var4 = Address

โปรโตคอล Philips RC-5

รหัส RC-5 ของฟิลิปส์ เป็นโปรโตคอลที่ถูกนำมาใช้อย่างแพร่หลายของนักประดิษฐ์ เพราะสามารถควบคุมได้ระยะไกล โปรโตคอลนี้ได้ถูกกำหนดเป็นอย่างดีสำหรับอุปกรณ์ที่แตกต่างกันอย่างระบบเอ็นเตอร์เทนเม้นท์และเมื่อเร็ว ๆ นี้ฟิลิปส์เริ่มใช้โปรโตคอลตัวใหม่ชื่อ RC-6 ซึ่งมีความสามารถมากขึ้น

คุณสมบัติ

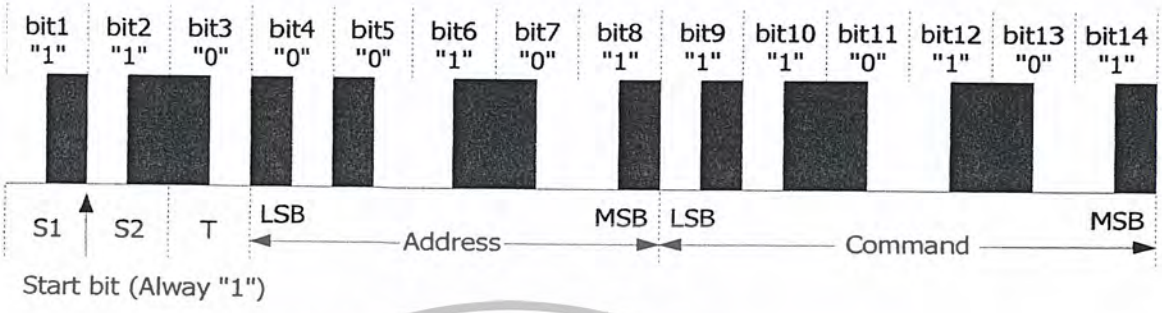
- แอดเดรสขนาด 5บิต และ คำสั่งขนาด 6 บิต
- เข้ารหัสแบบ Bi-Phase (Manchester Coding)
- ความถี่ส่ง 36 kHz
- Bit time เท่ากับ 1.8 ms
- ผลิตโดยโรงงานของฟิลิปส์

การมอดูเลต

รูปที่ 2.29 การมอดูเลตของลอจิก 1 และ ลอจิก 0

โปรโตคอลนี้ใช้การมอดูเลตแบบ Bi-phase (Manchester Coding) จากความถี่ส่ง 36 kHz และมีความยาวของบิตเท่ากันหมดคือ 1.8 ms ความถี่ส่ง 36 kHz จะแบ่งครึ่งกัน โดยครึ่งแรกแทนเป็นลอจิก “0” และ ครึ่งหลังแทนลอจิก “1” อัตรา pulse/pause ของความถี่ส่ง 36 kHz. คือ 1/3 หรือ 1/4 เพื่อลดการสิ้นเปลืองกำลังงาน

โปรโตคอล



รูปที่ 2.30 ตัวอย่างการส่งข้อมูล

จากรูปที่ 2.30 แสดงถึงชุดพัลซ์ของโปรโตคอล Philips RC-5 กรณีนี้เป็นการส่งคำสั่ง $2B แอดเดรสที่ $14 และ 2 พัลซ์แรกคือพัลซ์เริ่มต้น ซึ่งทั้ง 2 พัลซ์จะเป็นลอจิก "1" เสมอ และครึ่งหนึ่งของบิตเริ่มต้น จะถูกส่งไปเพื่อบอกให้ภาครับรู้ ก่อนที่จะส่งข่าวสารจริง ๆ RC-5ใช้บิตเริ่มต้นเพียงบิตเดียวเท่านั้น ส่วน S2 จะเป็นตัวแปลงคำสั่งทั้ง 6 บิต รวมแล้วทำให้มีคำสั่ง 7 บิต

บิตที่ 3 จะเป็น toggle bit ซึ่งบิตนี้จะเปลี่ยนทุกครั้งที่ปุ่มถูกกด วิธีนี้เครื่องรับสามารถรู้ลักษณะของปุ่มที่กดอยู่หรือกดซ้ำ 5 บิตถัดมาจะเป็นแอดเดรสซึ่งจะเริ่มต้นส่ง LSB ก่อน หลังจากแอดเดรสจะเป็นคำสั่งอีก 6 บิต

ข่าวสารทั้งหมดจะประกอบด้วย 14 บิตและมีความต่อเนื่องรวม 25.2 ms บางครั้งปรากฏว่าข่าวสารสั้นลงเพราะว่าครึ่งหนึ่งของบิตเริ่มต้น S1 ยังว่างอยู่และถ้าบิตสุดท้ายของข่าวสารเป็นลอจิก "0" นั่นคือครึ่งหนึ่งของบิตสุดท้ายว่างด้วย ตราบใดที่ปุ่มกดอยู่ ข่าวสารจะถูกส่งซ้ำไปทุก ๆ 114 ms ระหว่างนี้ toggle bit ยังคงเหมือนเดิมตลอดการส่งข่าวสารนี้ ซึ่งอยู่ที่ตัวซอฟแวร์ของเครื่องรับที่ตีความของการการส่งซ้ำโดยอัตโนมัติ

เราสามารถตรวจจับรหัสสัญญาณที่ส่งออกมาได้โดยใช้ไมโครคอนโทรลเลอร์ทำหน้าที่เป็นตัววิเคราะห์สัญญาณที่เข้ามา โดยทำตามขั้นตอนต่อไปนี้

1. กำหนดให้ Var1 = 4 , Var2 = 8
2. รอสัญญาณลงเป็น 0 ; จุดกึ่งกลางของ Start bit (s1)
3. รอสัญญาณขึ้นเป็น 1 ; จุดเริ่มต้นของ Start bit (s2)
4. รอสัญญาณลงเป็น 0 ; จุดกึ่งกลางของ Start bit (s2)
5. หน่วงเวลา 1100 uSec เพื่อทำการวัดระดับสัญญาณ

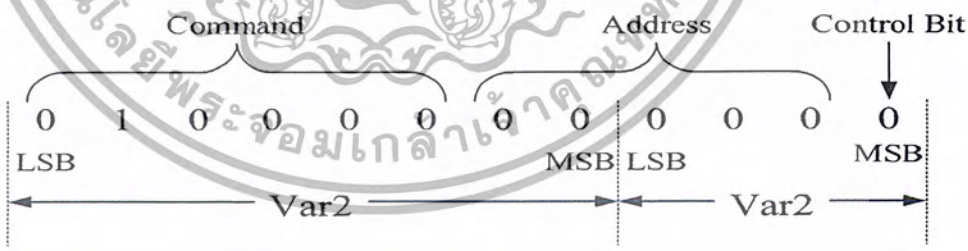
6. ถ้าสัญญาณเป็น “1” บิตที่รับมาเป็นลอจิก “1”
   - เซต Carry bit เป็น 1
   - หมุนข้อมูล Var1 ไปทางขวา
   - หมุนข้อมูล Var2 ไปทางขวา
   - เช็ค Carry bit ถ้าเป็น “1” ไปทำข้อ (8) ถ้าไม่
   - รอสัญญาณลงเป็น 0 กลับไปทำข้อ (5)
7. ถ้าสัญญาณเป็น “0” บิตที่รับมาเป็นลอจิก “0”
   - เซต Carry bit เป็น 0
   - หมุนข้อมูล Var1 ไปทางขวา
   - หมุนข้อมูล Var2 ไปทางขวา
   - เช็ค Carry bit ถ้าเป็น “1” ไปทำข้อ (8) ถ้าไม่
   - รอสัญญาณขึ้นเป็น 1 กลับไปทำข้อ (5)
8. ข้อมูลมีทั้งหมด 12 บิต ดังนี้

   Command : มี 6 บิต คือ Var1 เริ่มจากบิตที่ 7 (LSB) ถึงบิตที่ 2 (MSB)

   Address : มี 5 บิต คือ Var1 เริ่มจากบิตที่ 1 ถึงบิตที่ 0

   Var2 เริ่มจากบิตที่ 7 ถึงบิตที่ 5

   Control Bit : มี 1บิต คือ บิตที่ 4 ของ Var2



รูปที่ 2.31 รูปแบบการจัดแบ่งข้อมูล

2.4 หลักการของระบบฐานข้อมูล(Database)

ฐานข้อมูล หมายถึง กลุ่มของข้อมูลและความสัมพันธ์ (A collection of data and relationships) ความสัมพันธ์ของฐานข้อมูล สามารถแสดงในรูปของ โมเดลของข้อมูล โมเดลที่ได้รับความนิยมคือ รีเลชั่นนอลโมเดล (Relational model) เป็นโมเดลที่ใช้ในการอธิบายความสัมพันธ์ของข้อมูลที่ถูกเก็บด้วยระบบจัดการฐานข้อมูลแบบรีเลชั่นนอล (Relational Database Management System : RDBMS) ซึ่งการนำไปใช้งาน สามารถนำไปใช้กับเครื่องระดับตั้งแต่เมนเฟรมลงไปถึงเครื่องระดับไมโครคอมพิวเตอร์

การออกแบฐานข้อมูล มีความสำคัญต่อการจัดการระบบฐานข้อมูล ทั้งนี้เนื่องจากข้อมูลที่อยู่ภายในฐานข้อมูลจะต้องศึกษาความสัมพันธ์ของข้อมูล โครงสร้างของข้อมูล การเข้าถึงข้อมูล และกระบวนการที่โปรแกรมประยุกต์จะเรียกใช้ฐานข้อมูลดังนั้นเราสามารถแบ่งการสร้าฐานข้อมูลได้ดังนี้

1. รูปแบบข้อมูลแบบลำดับขั้น หรือ โครงสร้างแบบลำดับขั้น วิธีการสร้างข้อมูลแบบลำดับขั้นถูกพัฒนาโดย บริษัท ไอ บี เอ็ม จำกัด ได้รับความนิยมอย่างมากในการพัฒนาข้อมูลบนคอมพิวเตอร์ขนาดใหญ่ และขนาดกลางโดยโครงสร้างข้อมูลจะสร้างรูปแบบเหมือนต้นไม้ โดยมีความสัมพันธ์เป็นแบบ หนึ่ง ต่อหลาย (one-to-many) วิธีการจัดแบบลำดับขั้นเป็นการจัดกลุ่มข้อมูลที่มีความสัมพันธ์กัน และกำหนดให้เป็นเซกเมนต์ (segment) โดยมีการแยกประเภทของเซกเมนต์ว่าเป็นเซกเมนต์ราก หรือเป็นเซกเมนต์ตัวที่ปีนตัวพึ่ง
2. รูปแบบข้อมูลแบบเครือข่าย ฐานข้อมูลแบบเครือข่ายมีความคล้ายคลึงกับฐานข้อมูลแบบลำดับขั้น ต่างกันที่โครงสร้างแบบเครือข่าย อาจจะมีการต่อแบบ หนึ่งต่อหนึ่ง(one-to-one) หรือ หนึ่งต่อหลาย

   ข้อดีและข้อเสียของโครงสร้างแบบเครือข่ายคือ เรคคอร์ด (Record) แต่ละประเภทสามารถใช้เป็นเรคคอร์ดนำได้ โดยกล่าวถึงก่อน ส่วนการซ้ำซ้อนของฐานข้อมูลจะมีน้อยมาก เนื่องจากเรคคอร์ดสมาชิกสามารถใช้ร่วมกันได้
3. รูปแบบความสัมพันธ์ข้อมูล เป็นลักษณะการออกแบบฐานข้อมูลโดยจัดข้อมูลให้อยู่ในรูปของตาราง

บทที่ 3
การออกแบบ

การออกแบบเครื่องทดสอบเครื่องทดสอบอินฟราเรดรีโมทคอนโทรล ในโครงงานนี้แบ่งการทำงานออกเป็น 2 ส่วน คือ วงจรทางด้านฮาร์ดแวร์ และซอฟแวร์ที่ใช้แสดงผล ในวงจรทางด้านฮาร์ดแวร์นี้ โดยใช้ไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ AT90S2313 เป็นไอซีประเภทซีมอส ชนิด 8 บิต เป็นตัวควบคุมการทำงานของวงจร ทางด้านซอฟแวร์ที่ใช้ในการแสดงผลจะใช้ Visual Basic 6.0 เป็นตัวควบคุม

3.1 การออกแบบฮาร์ดแวร์

3.1.1 ส่วนประมวลผล

การออกแบบส่วนประมวลผลในโครงงานนี้ใช้ ไมโครคอนโทรลเลอร์ตระกูล AVR เบอร์ AT90S2313 ของบริษัท ATMEL ซึ่งมีพอร์ตใช้งานและมีหน่วยความจำเพียงพอสำหรับโครงงานนี้ สามารถโปรแกรมซ้ำได้ 1000 ครั้ง และมีการต่อขาใช้งานต่าง ๆ ดังนี้ต่อไปนี้

ขา 1 เป็นขารีเซตซึ่งจะเกิดการรีเซตก็ต่อเมื่อมีลอจิก 0 เข้ามาเป็นเวลาไม่ต่ำกว่า 50 ns จะเกิดการรีเซตแม้ว่าจะไม่มีสัญญาณคล็อกก็ตาม

ขา 3 คือพอร์ต PD1 และยังทำหน้าที่เป็นขา TXD ส่งสัญญาณออกสำหรับพอร์ตอนุกรม

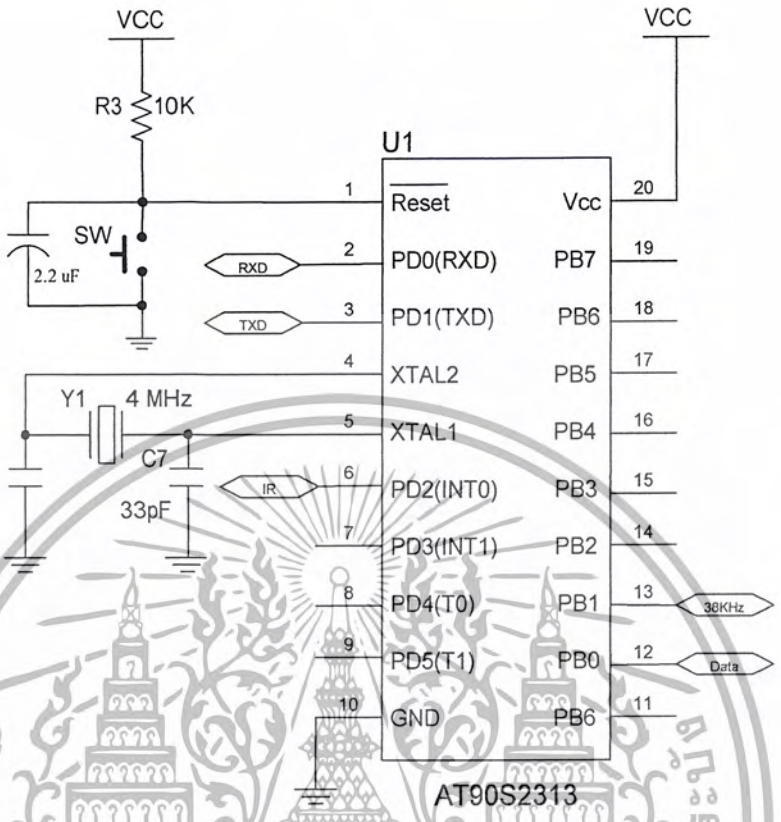
ขา 4 และ 5 ต่อกับคริสตอลเพื่อเป็นฐานเวลาให้กับไมโครคอนโทรลเลอร์โดยอาศัยความถี่ของคริสตอลเป็นตัวกำหนดฐานเวลา และฐานเวลานี้เองเป็นตัวบอกให้ทราบว่า ในคำสั่งหนึ่งคำสั่งใช้เวลาประมวลผลเท่าไร

ขา 6 คือพอร์ต PD2 ใช้เป็นขาอินเตอร์รัปต์ทำหน้าที่รับสัญญาณอินฟราเรด ที่ส่งมาจาตัวรับสัญญาณอินฟราเรด เพื่อส่งให้ไมโครคอนโทรลเลอร์ประมวลผล

ขา 12 พอร์ต PB0 ทำหน้าที่ข้อมูลไปมอดูเลตกับความถี่ 38 KHz

ขา 13 พอร์ต PB1 ทำหน้าที่ผลิตความถี่ 38 KHz จากไมโครคอนโทรลเลอร์

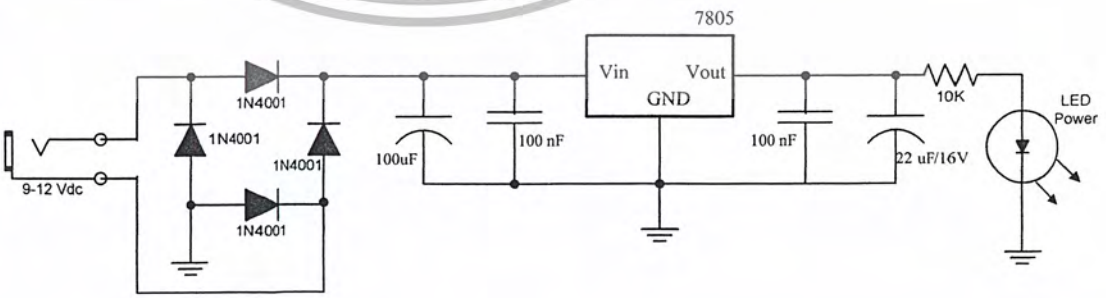
ขา 20 เป็นขา VCC ต่อกับแรงดันไฟ 5 โวลต์



รูปที่ 3.1 ขาต่อใช้งานของไมโครคอนโทรลเลอร์ AT90S231

3.1.2 ภาคจ่ายไฟ

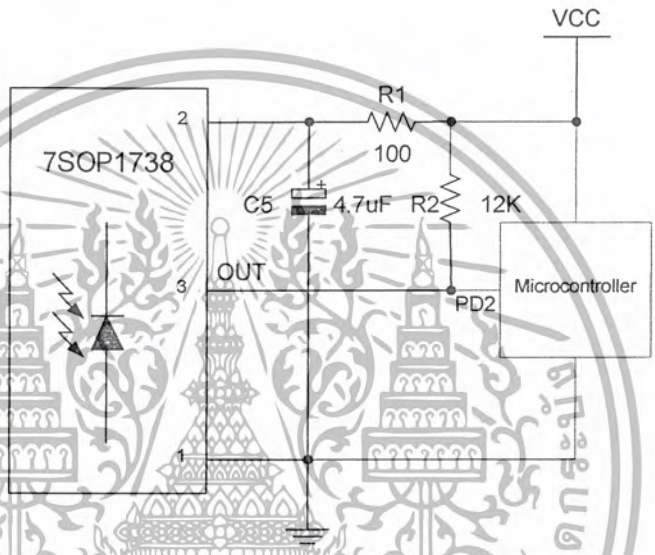
ใช้ไอซีเร็กกูเลต เบอร์ 7805 เพื่อแปลงแรงดันจากอะแดปเตอร์ 9 โวลต์ให้เหลือแรงดันคงที่ 5 โวลต์ และป้อนให้กับวงจรในส่วนต่างๆ ในวงจร ไดโอดทั้งสี่ตัวมีไว้เพื่อป้องกันการจ่ายแรงดันกลับขั้ว



รูปที่ 3.2 วงจรภาคจ่ายไฟ

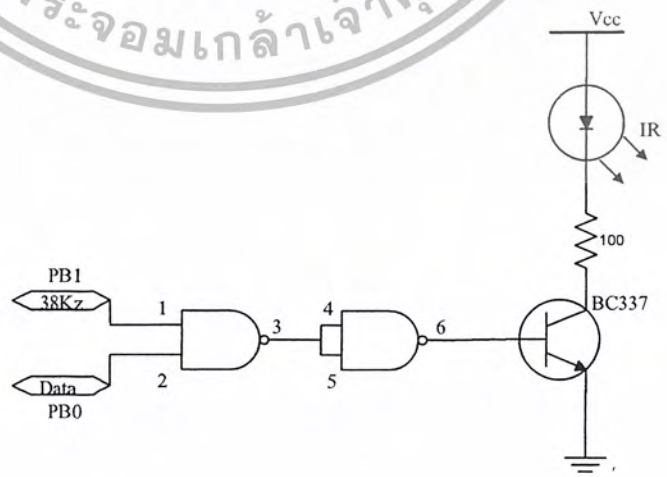
3.1.3 ส่วนของวงจรรับสัญญาณอินฟราเรดรีโมทคอนโทรล

ตัวรับสัญญาณอินฟราเรดรีโมทคอนโทรล ทำหน้าที่รับสัญญาณจากรีโมทคอนโทรลและส่งไปยังพอร์ต PD2 (ขา 6) ของไมโครคอนโทรลเลอร์ซึ่งจะทำให้เกิดการอินเตอร์รัปต์ขึ้น ซึ่งโปรแกรมที่อยู่ในบริการอินเตอร์รัปต์มี 2 โหมดคือ โหมดการทดสอบข้อมูลที่รับเข้ามา และโหมดแสดงผลสัญญาณ



รูปที่ 3.3 วงจรรับสัญญาณอินฟราเรดรีโมทคอนโทรล

3.1.4 ส่วนของภาคส่งสัญญาณอินฟราเรดรีโมทคอนโทรล



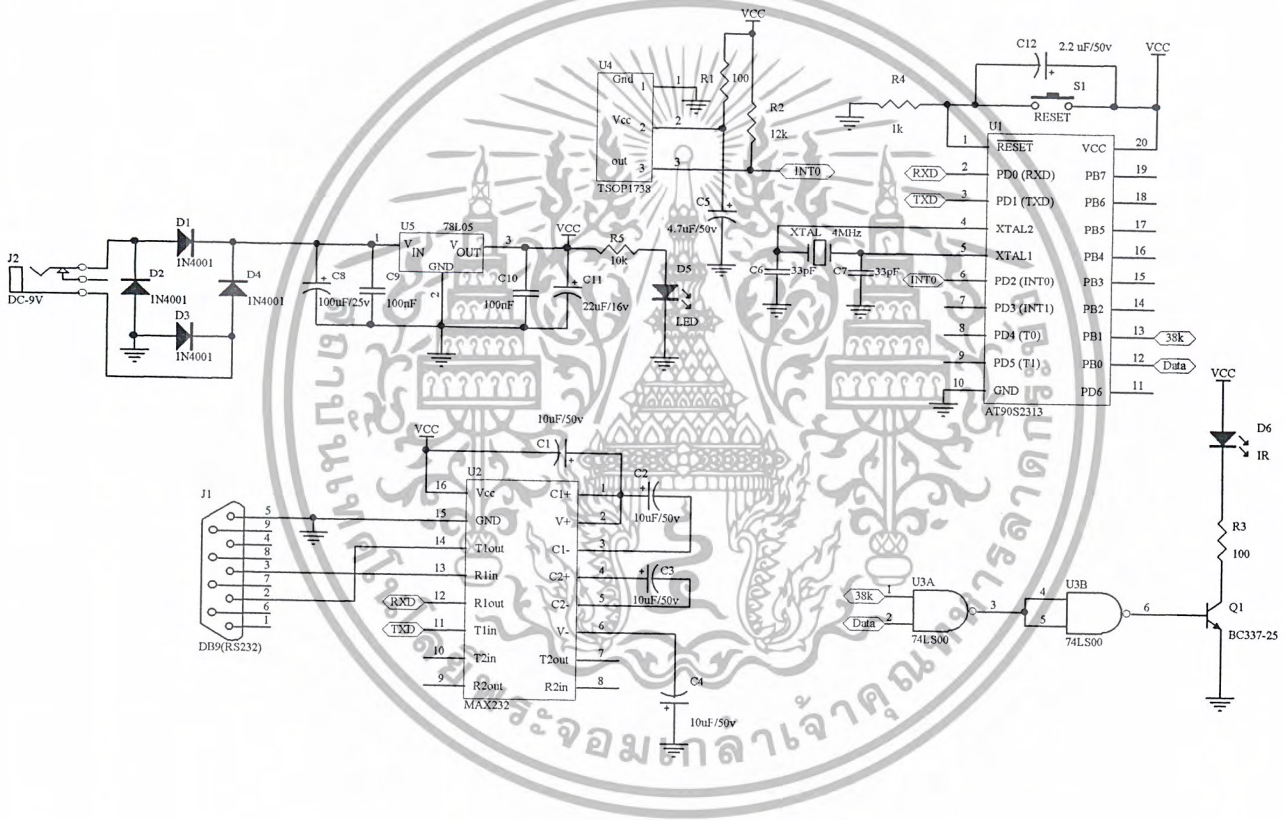
รูปที่ 3.4 วงจรภาคส่งสัญญาณรีโมทคอนโทรล

ในวงจรส่งสัญญาณรีโมทคอนโทรลจะใช้แนนเกต (NAND Gate) ทำหน้าที่ในการผสมสัญญาณคลื่นพาหะ 38 KHz กับสัญญาณข้อมูลที่ต้องการส่ง จากนั้นส่งต่อไปให้กับทรานซิสเตอร์ทำหน้าที่เหมือนกับสวิตช์เพื่อ เปิด-ปิด การจ่ายไฟ ทำให้หลอดอินฟราเรดส่งสัญญาณออกมาตามค่าที่ต้องการ

3.1.5 ส่วนของการติดต่อกับวิชวลเบสิกผ่านทางพอร์ต RS-232

รูปที่ 3.5 วงจรการเชื่อมต่อกับพอร์ต RS-232

สัญญาณข้อมูลที่ถูกประมวลผลด้วยไมโครคอนโทรลเลอร์จะถูกส่งออกทางพอร์ต TXD ของไมโครคอนโทรลเลอร์ เพื่อนำไปแสดงผลด้วยโปรแกรมวิชวลเบสิกต่อไป ส่วนการเปลี่ยนโหมดการทำงานของไมโครคอนโทรเลอร์คำสั่งจะถูกส่งจากโปรแกรมวิชวลเบสิกเข้ามาทาง พอร์ต RXD ซึ่งมขนาด 2 ไบต์ เช่น ถ้าต้องการเลือกโปรแกรมให้ทำงานในโหมดแสดงผลสัญญาณ จะส่งข้อมูล “7A” ออกไปให้กับคอนโทรลเลอร์ เป็นต้น



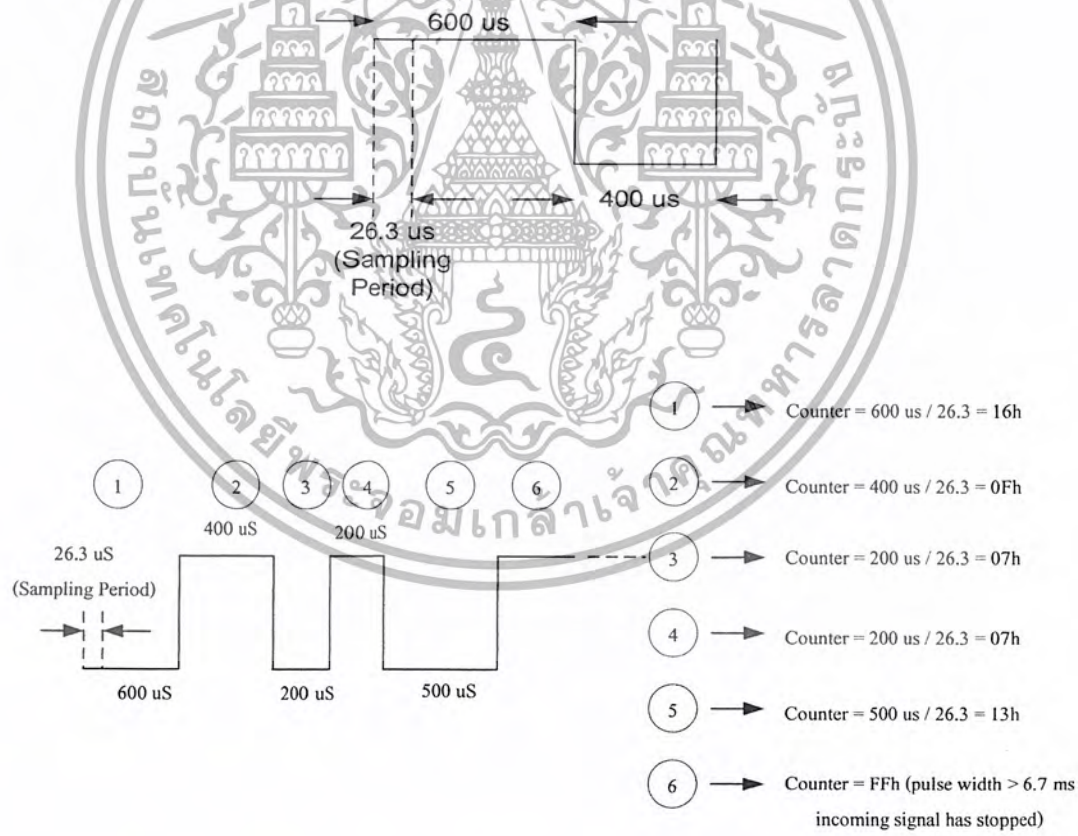
รูปที่ 3.6 วงจรรวม

3.2 การออกแบบซอฟแวร์

ในการออกแบบซอฟแวร์ควบคุมการทำงานและแสดงผลของโครงงานนี้แบ่งออกเป็นสองส่วนคือ ซอฟแวร์ที่เป็นภาษาแอสเซมบลี ซึ่งจะใช้ในการควบคุมการทำงานของ ไมโครคอนโทรล-เลอร์ และส่วนที่เป็นภาษาวิชวลเบสิกที่จะใช้ในการแสดงผลข้อมูล

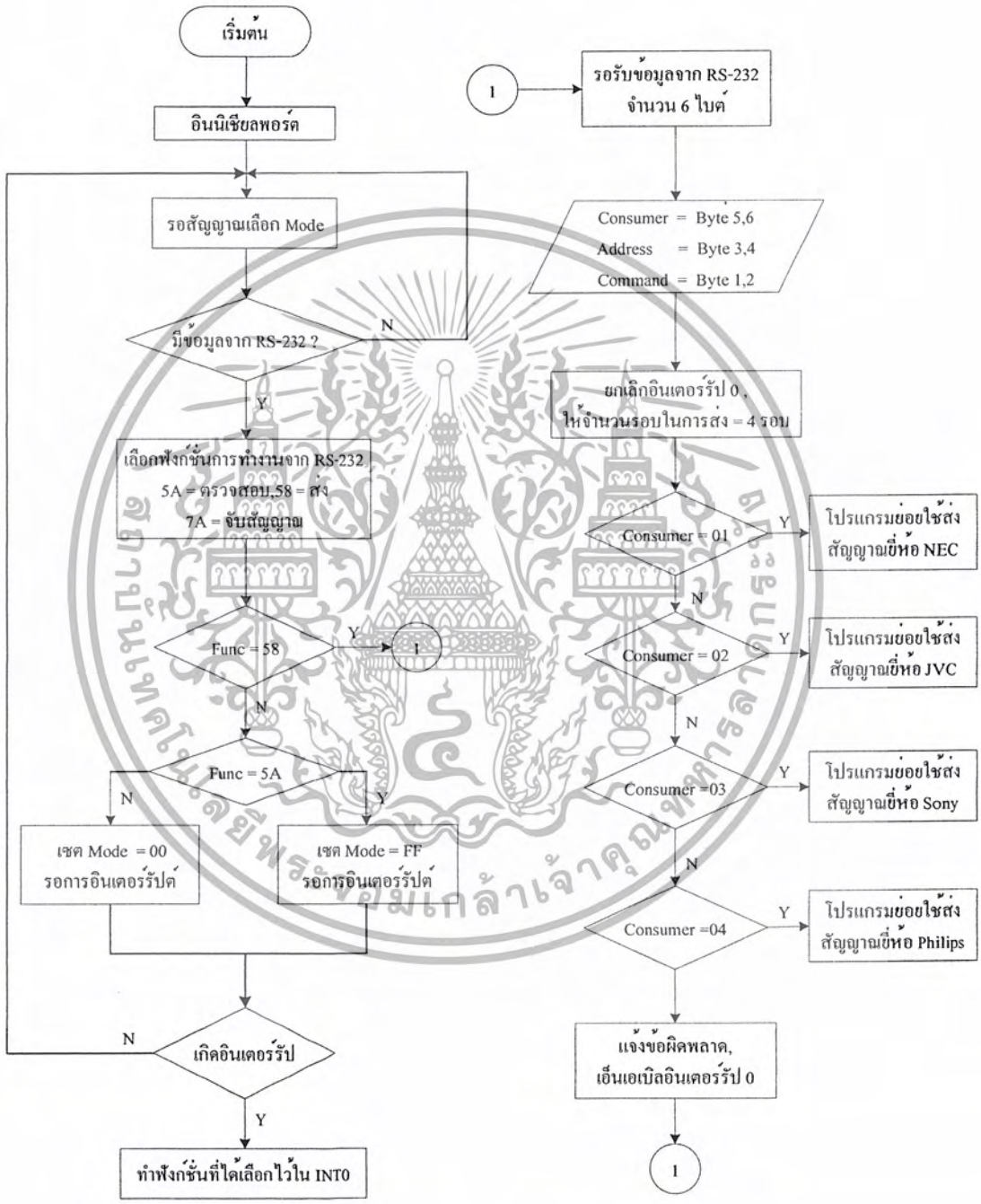
3.2.1 การออกแบบซอฟแวร์ส่วนของไมโครคอนโทรลเลอร์

การออกแบบในส่วนนี้จะเป็นการตรวจจับสัญญาณอินฟราเรดที่รับเข้ามา โดยในขั้นแรกเราจะต้องรู้ให้ได้ว่า สัญญาณที่รับเข้ามาเป็นของยี่ห้อใด โดยจะใช้การวัดค่าของเฮดพัลซ์ ซึ่งจะมีค่าต่างกันไปในแต่ละยี่ห้อ ในที่นี้จะใช้หลักการแซมปลิ้งสัญญาณของเฮดพัลซ์ จะใช้คาบเวลาของสัญญาณแซมปลิ้งประมาณ 1/38 kHz ของความถี่ ซึ่งจะประมาณ 26.3 us



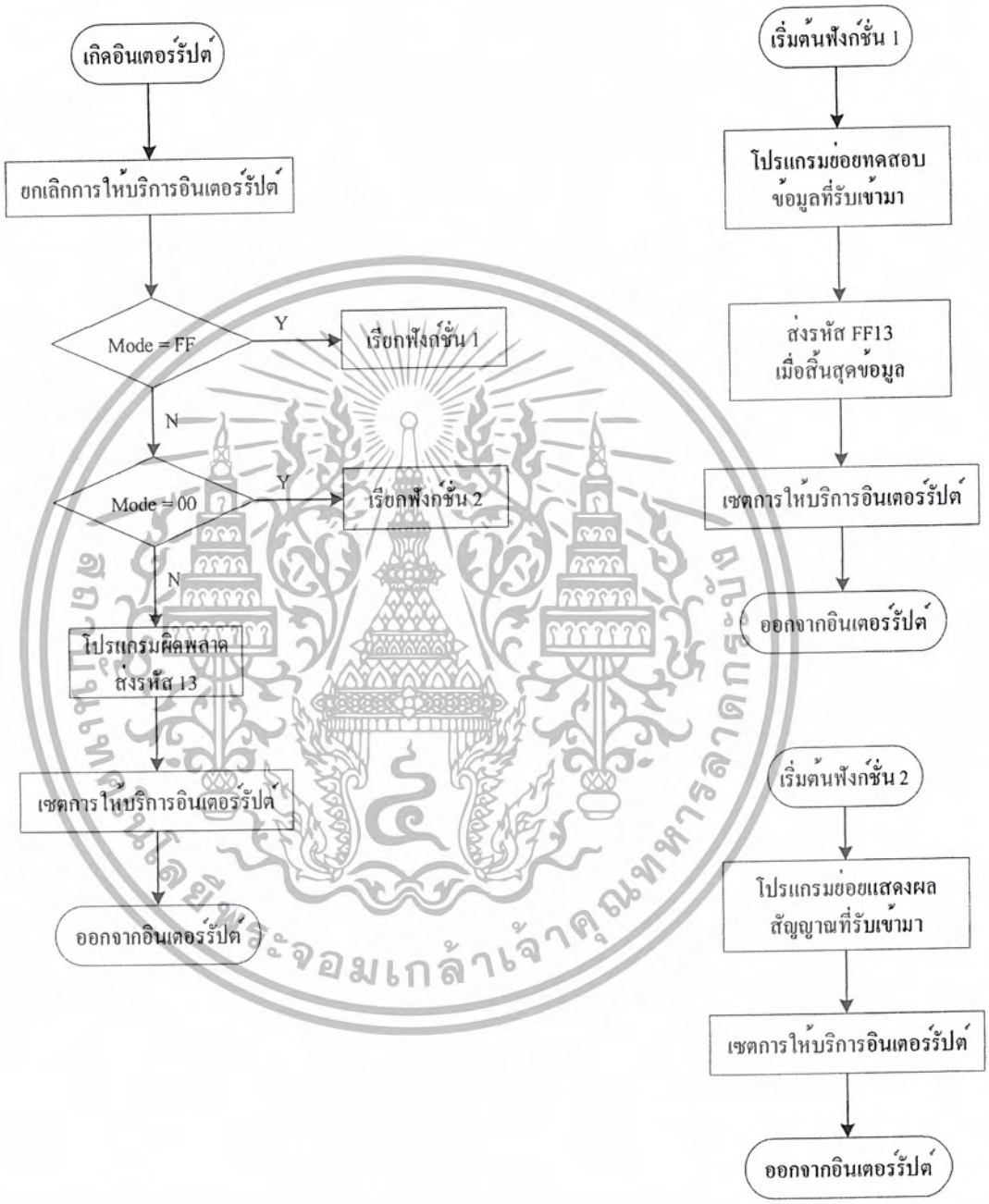
รูปที่ 3.7 ตัวอย่างการแซมปลิ้งสัญญาณ

โฟลวชาร์ตโปรแกรมหลัก



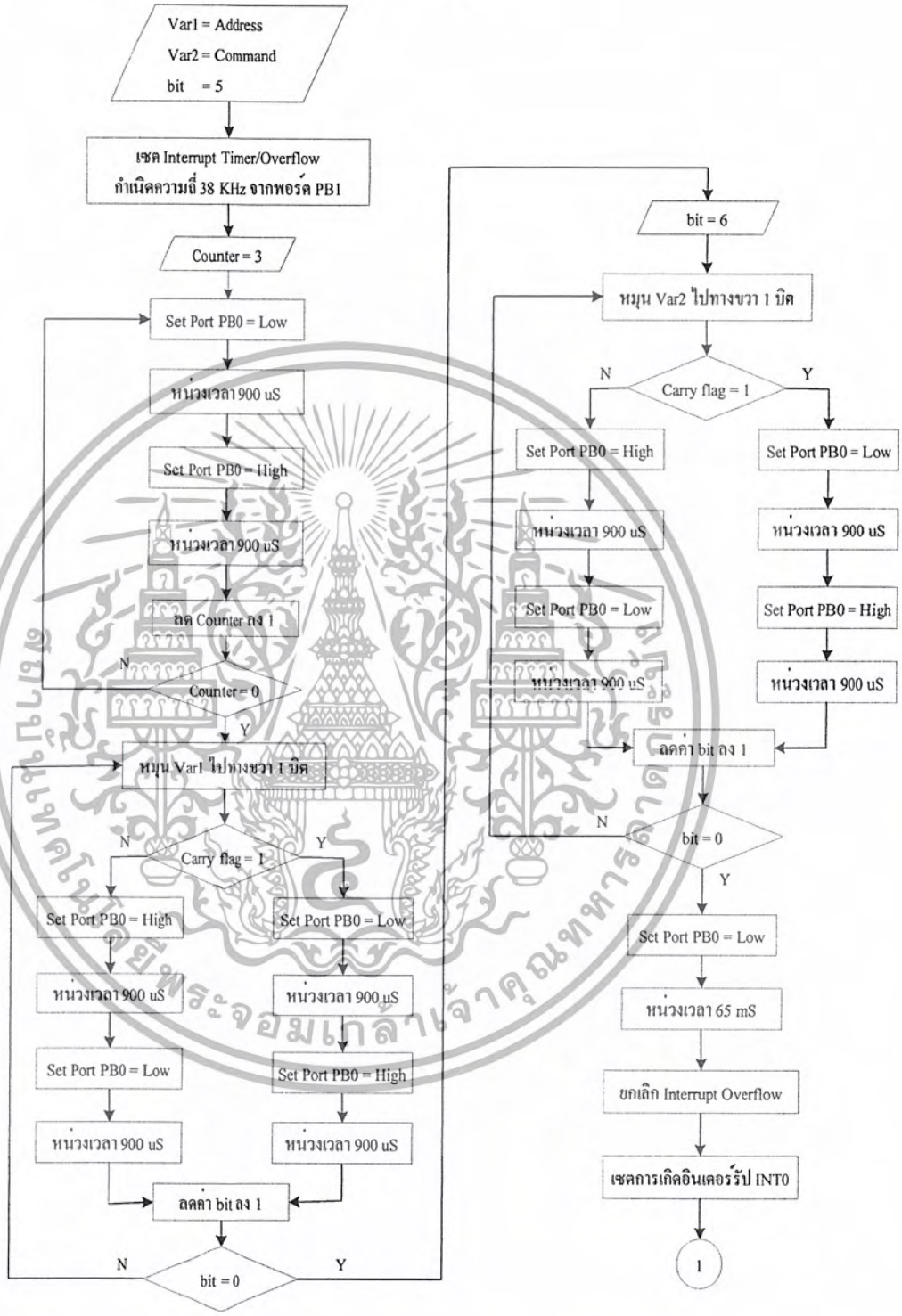
รูปที่ 3.8 โฟลวชาร์ตโปรแกรมหลัก

โปรแกรมย่อยอินเตอร์รัปต์ (INT0) รับสัญญาณอินฟราเรดที่รับเข้ามา



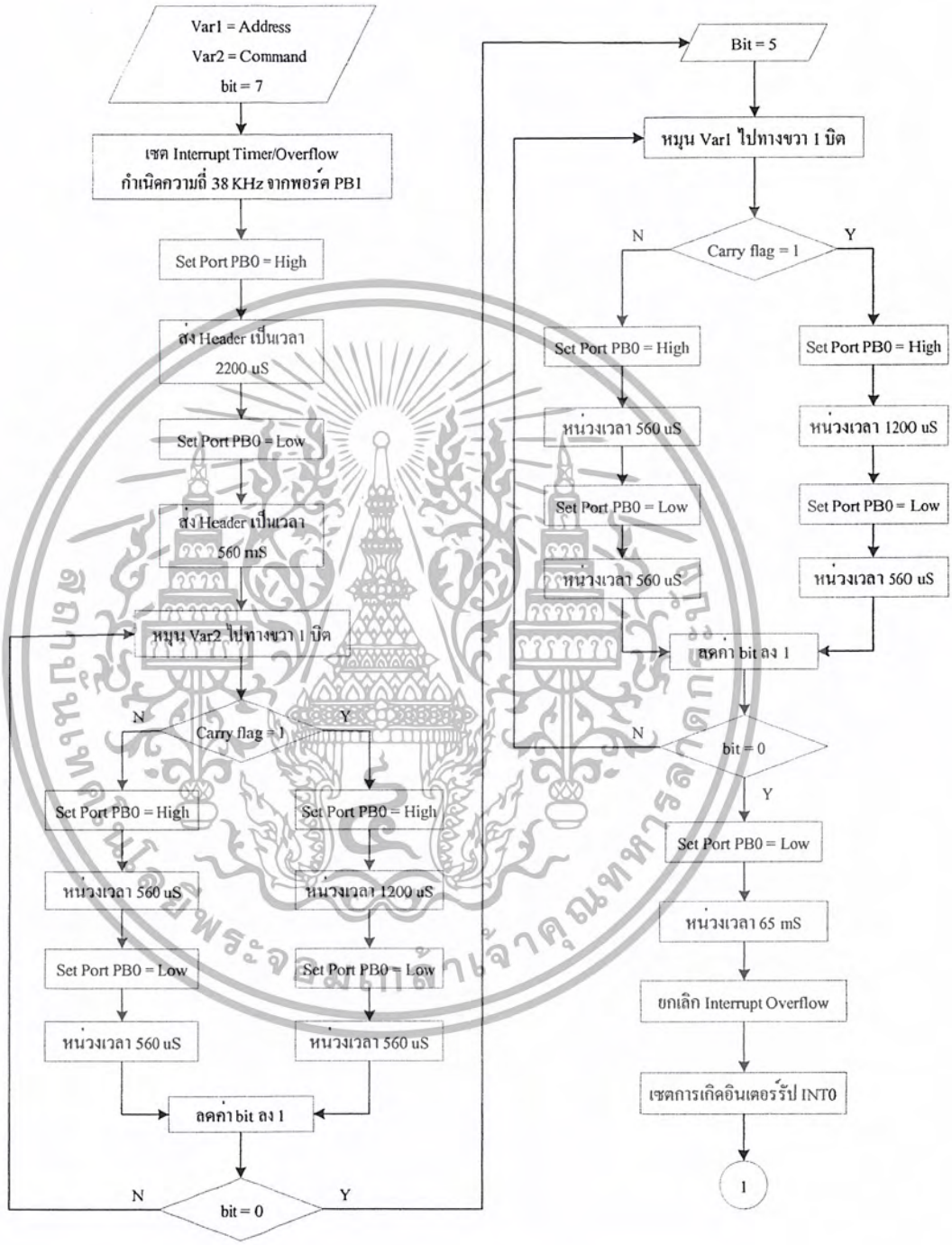
รูปที่ 3.9 โฟลวชาร์ตโปรแกรมย่อยอินเตอร์รัปต์ (INT0) รับสัญญาณอินฟราเรดที่รับเข้ามา

โปรแกรมย่อยในการส่งสัญญาณของ Philips



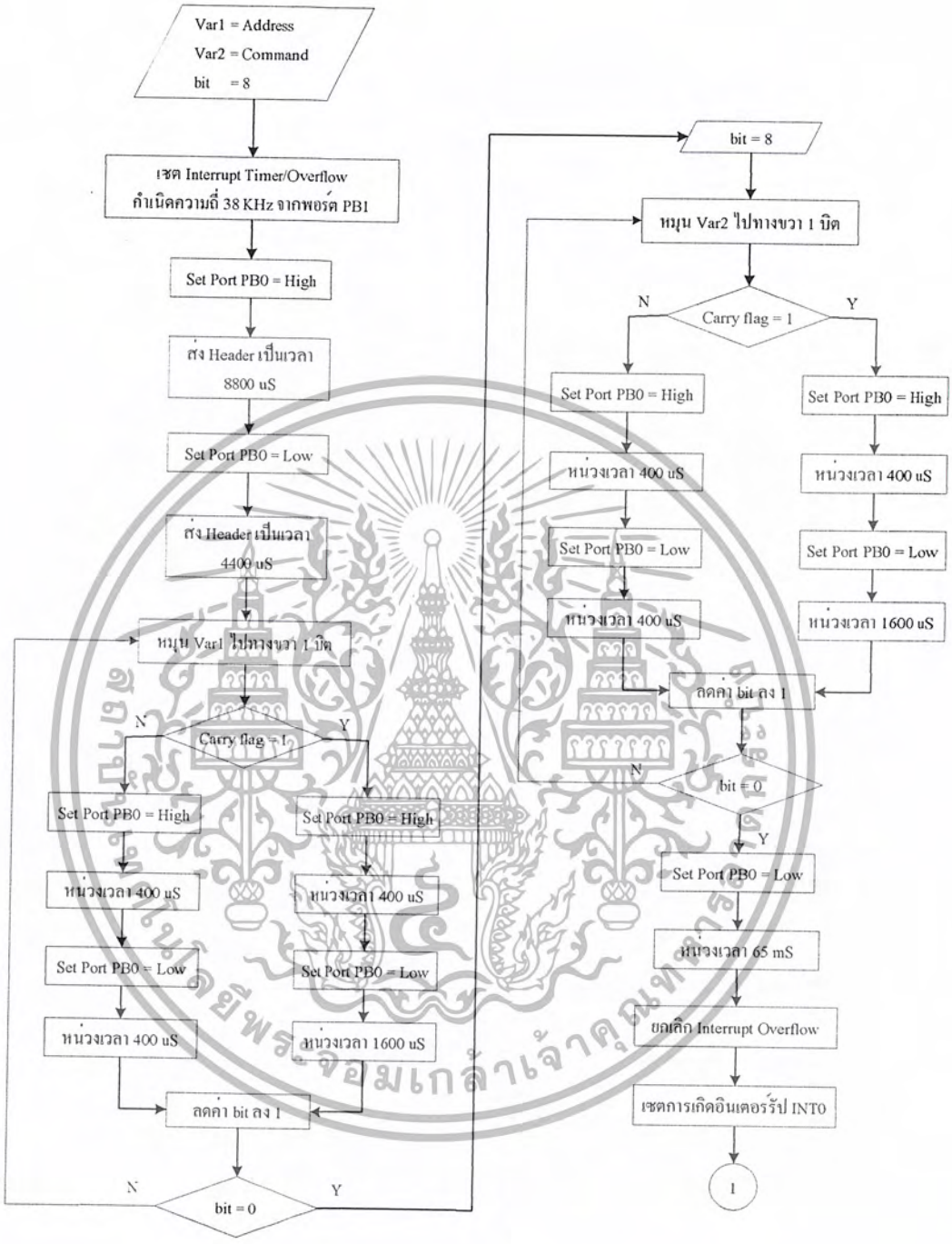
รูปที่ 3.10 โฟลวชาร์ตโปรแกรมย่อยในการส่งสัญญาณของ Philips

โปรแกรมย่อยในการส่งสัญญาณของ Sony

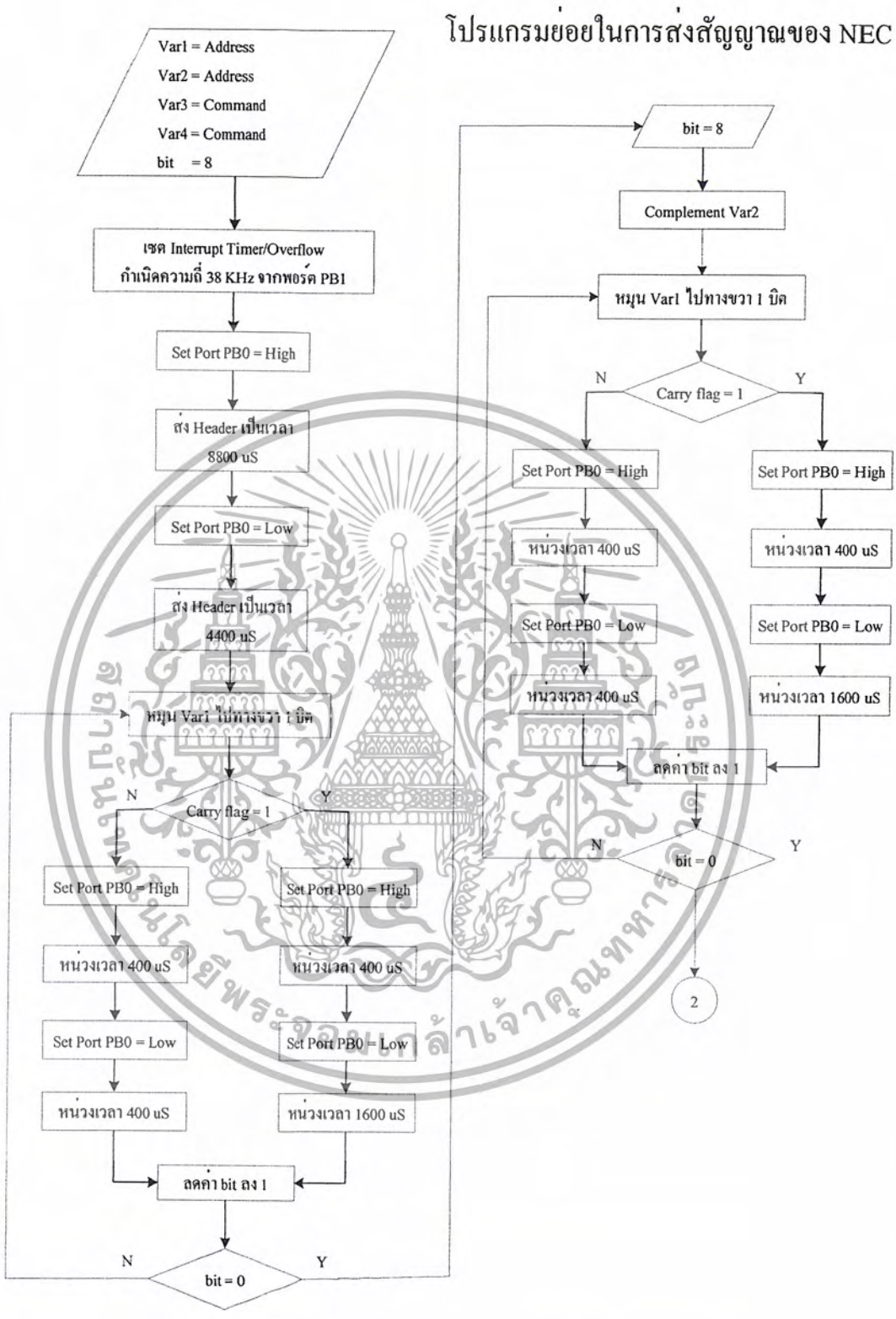


รูปที่ 3.11 โฟลวชาร์ตโปรแกรมย่อยในการส่งสัญญาณของ Sony

โปรแกรมย่อยในการส่งสัญญาณของ JVC

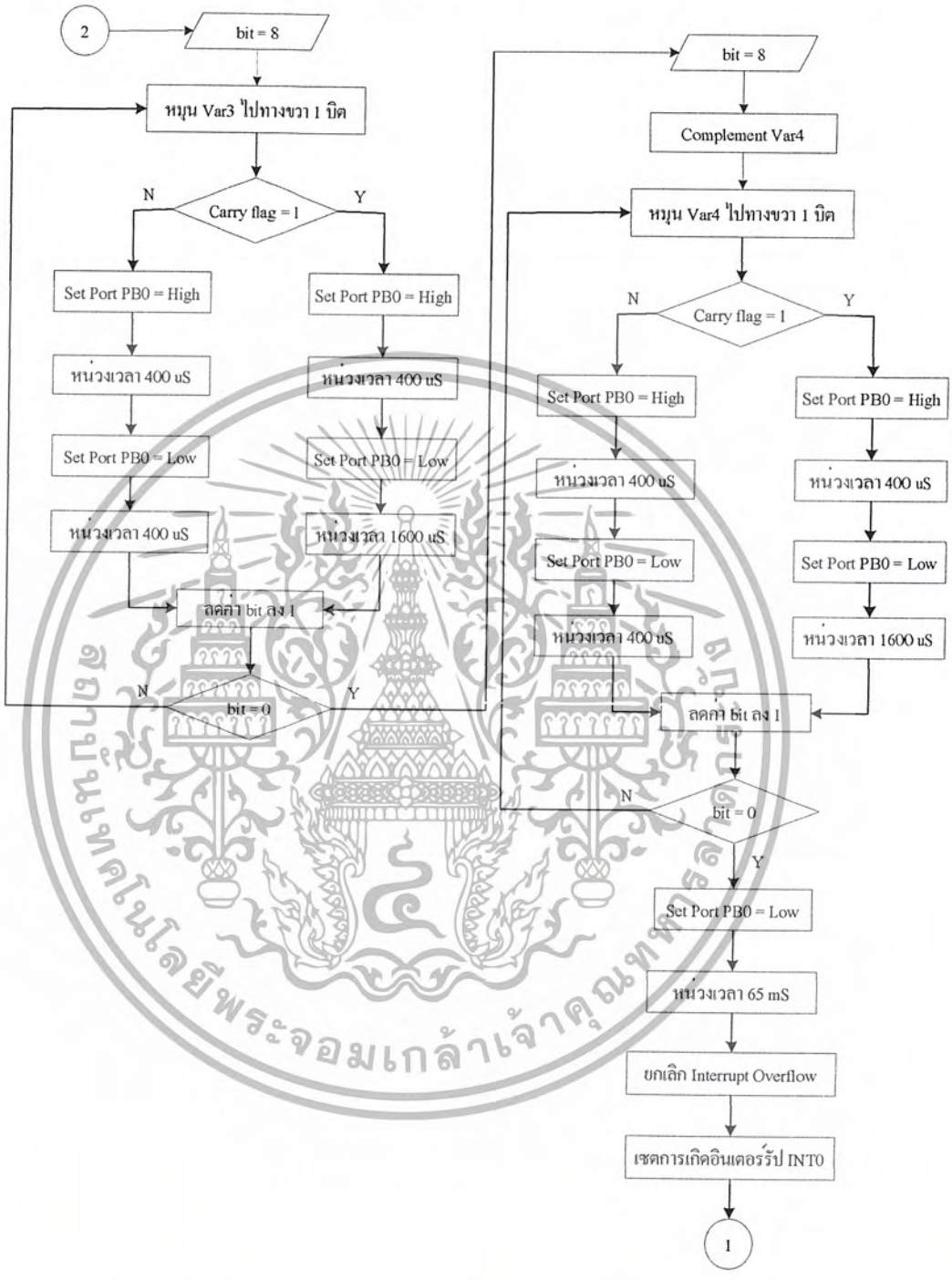


รูปที่ 3.12 โฟลวชาร์ตโปรแกรมย่อยในการส่งสัญญาณของ JVC



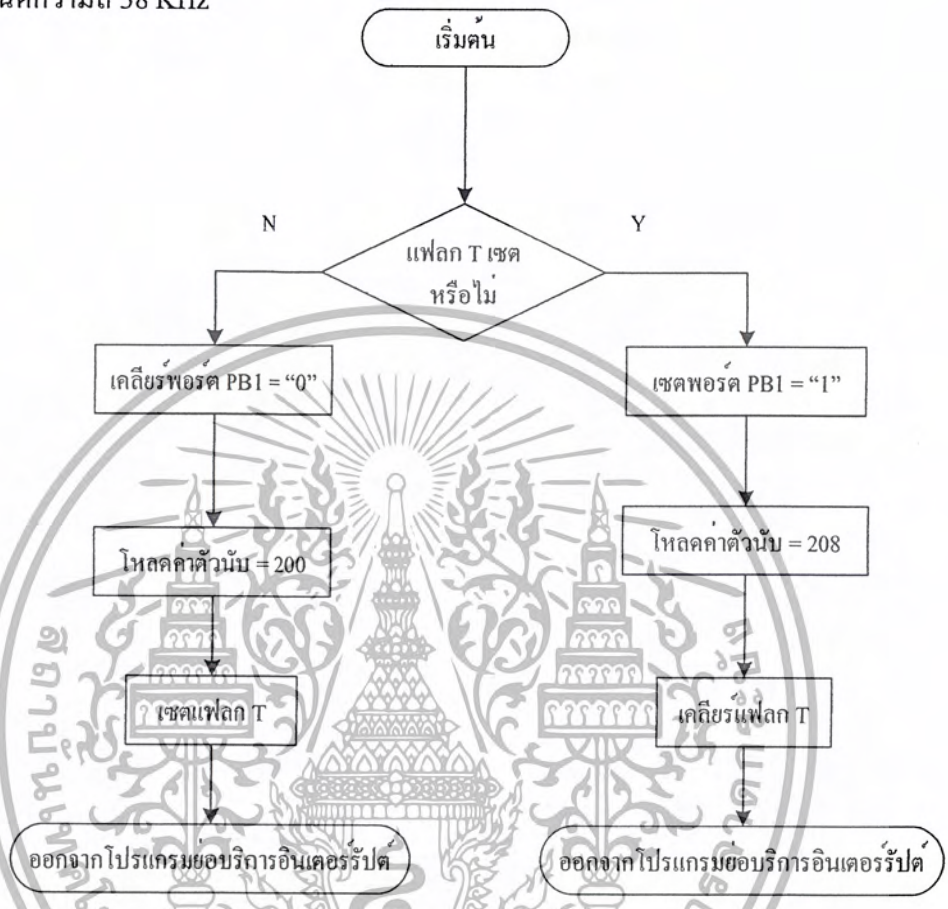
รูปที่ 3.13 โฟลวชาร์ตโปรแกรมย่อยในการส่งสัญญาณของ NEC

โปรแกรมย่อยในการส่งสัญญาณของ NEC (ต่อ)



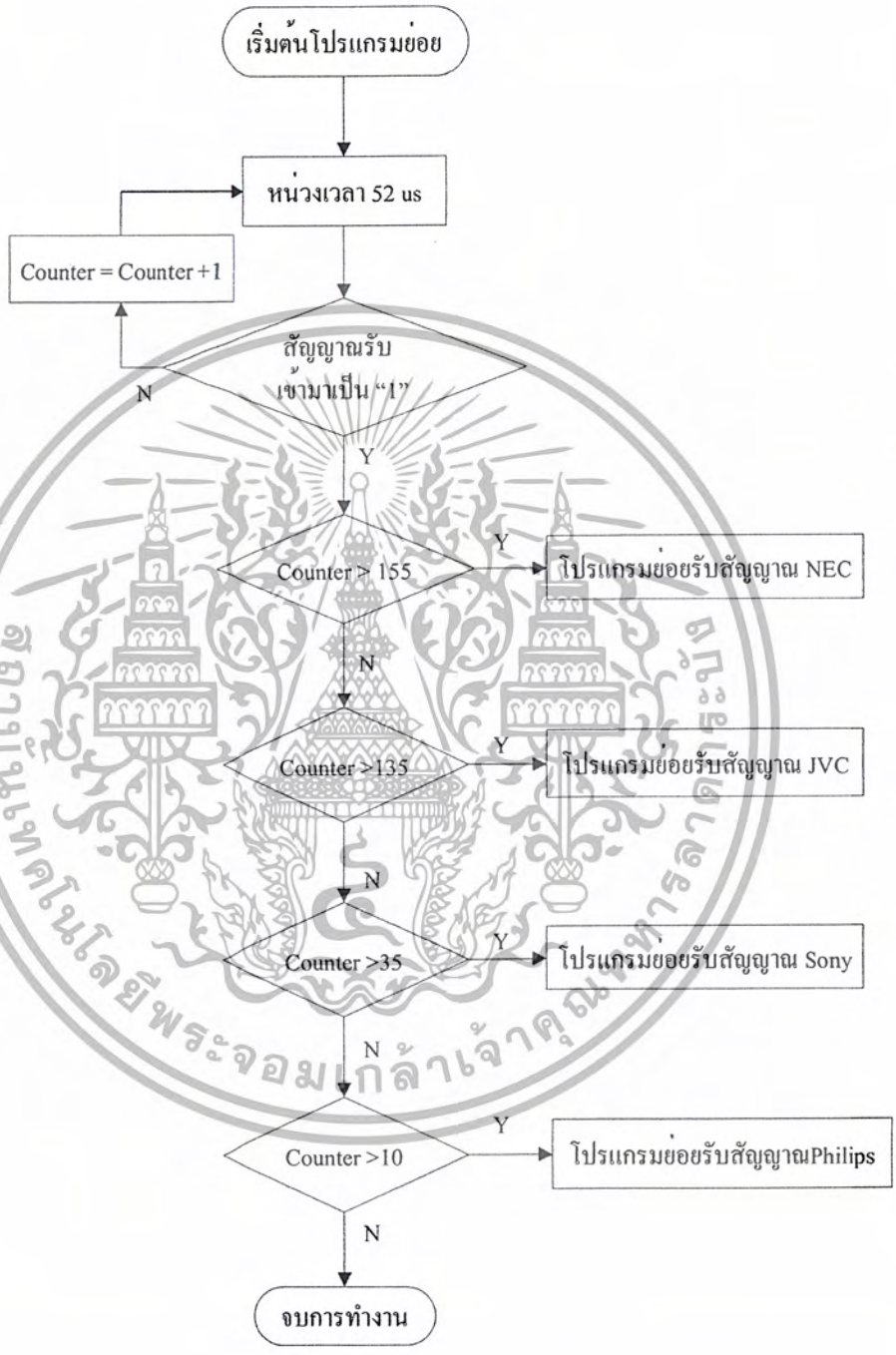
รูปที่ 3.14 โฟลวชาร์ตโปรแกรมย่อยในการส่งสัญญาณของ NEC (ต่อ)

โปรแกรมย่อยบริการอินเตอร์รัปต์
กำเนิดความถี่ 38 KHz

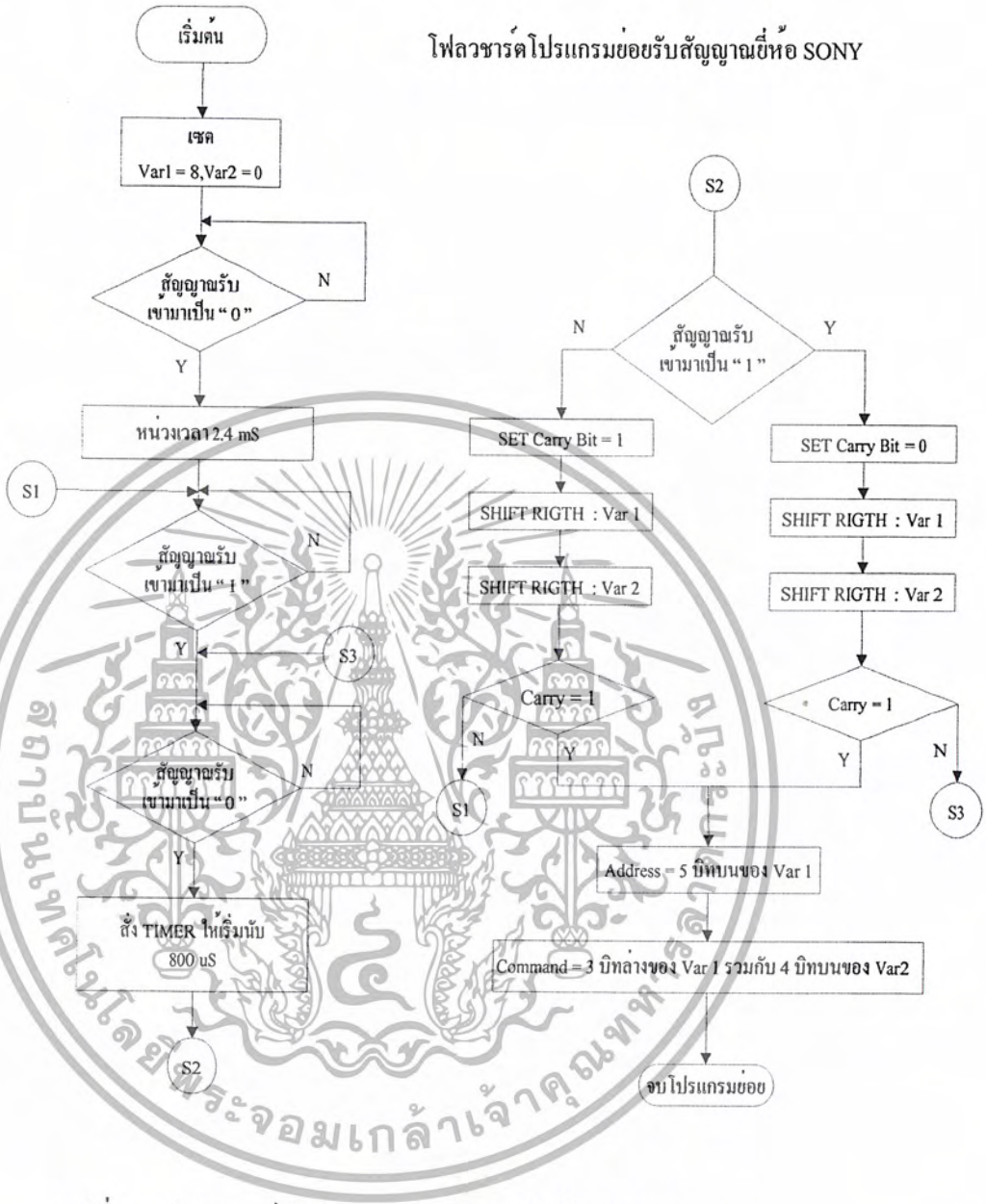


รูปที่ 3.15 โฟลวชาร์ตโปรแกรมย่อยบริการอินเตอร์รัปต์กำเนิดความถี่ 38 KHz

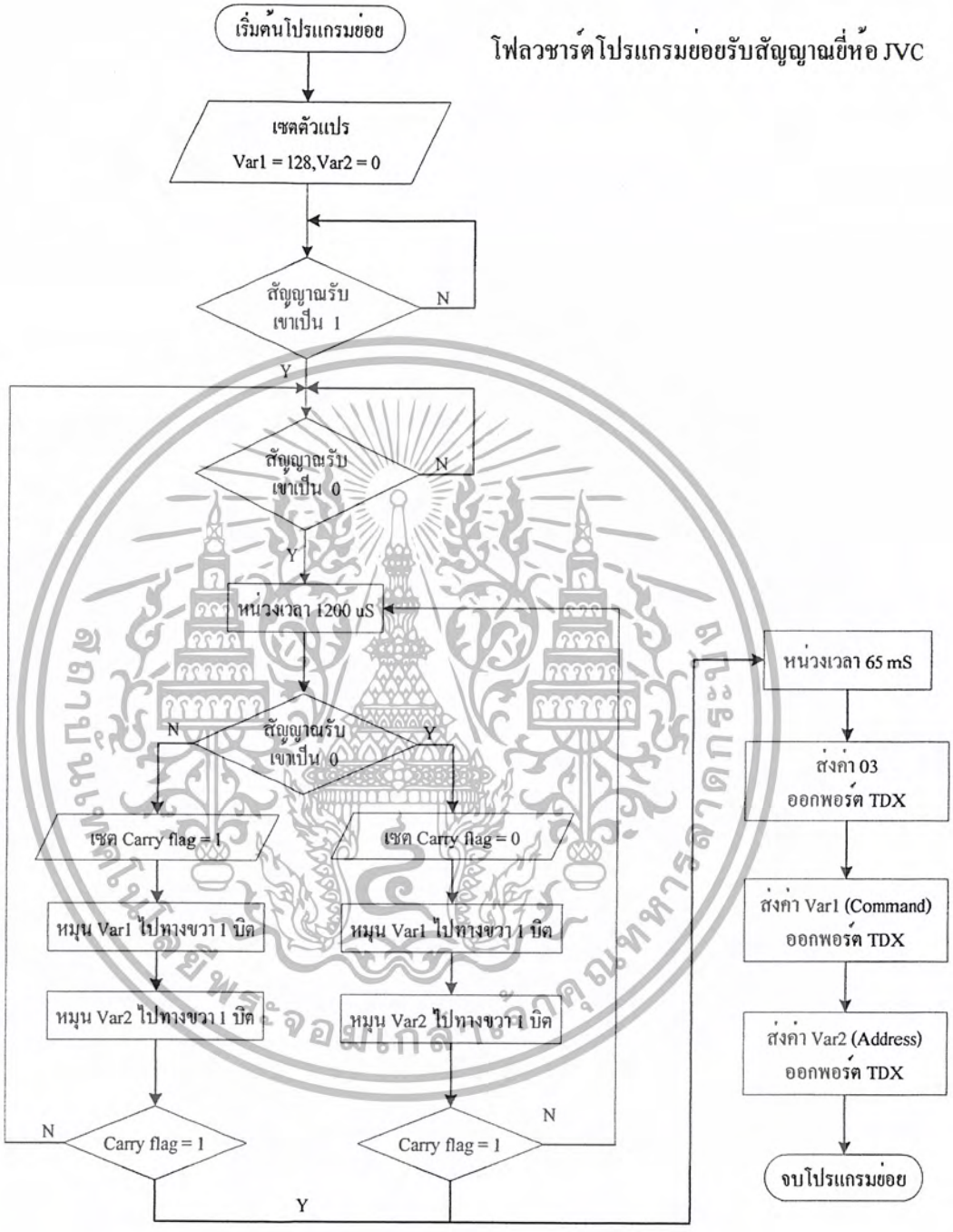
โปรแกรมย่อยรับสัญญาณเข้ามาทดสอบ



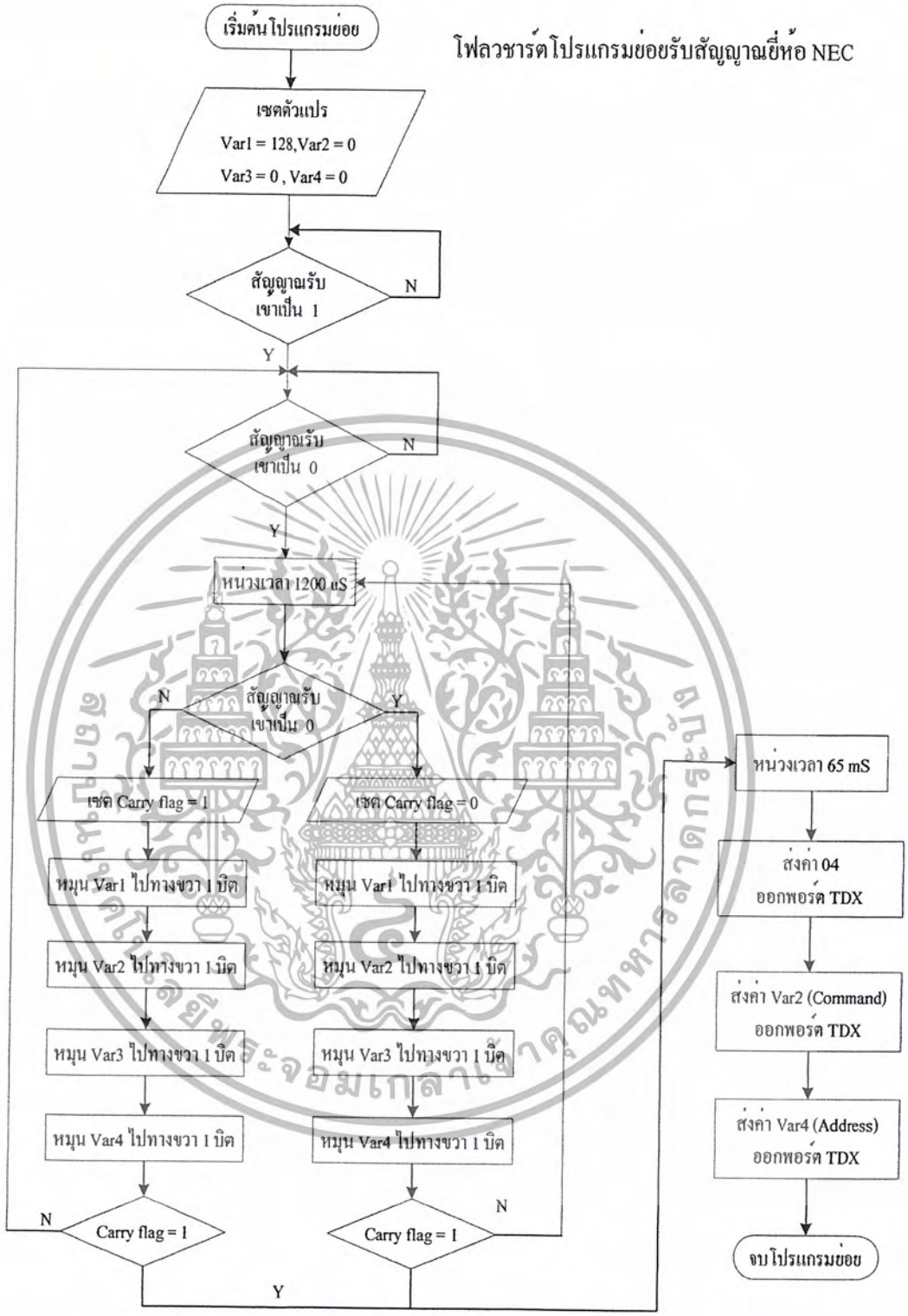
รูปที่ 3.16 โฟลวชาร์ตโปรแกรมย่อยรับสัญญาณเข้ามาทดสอบ



รูปที่ 3.17 โฟลวชาร์ตโปรแกรมย่อยในการรับสัญญาณของ SONY



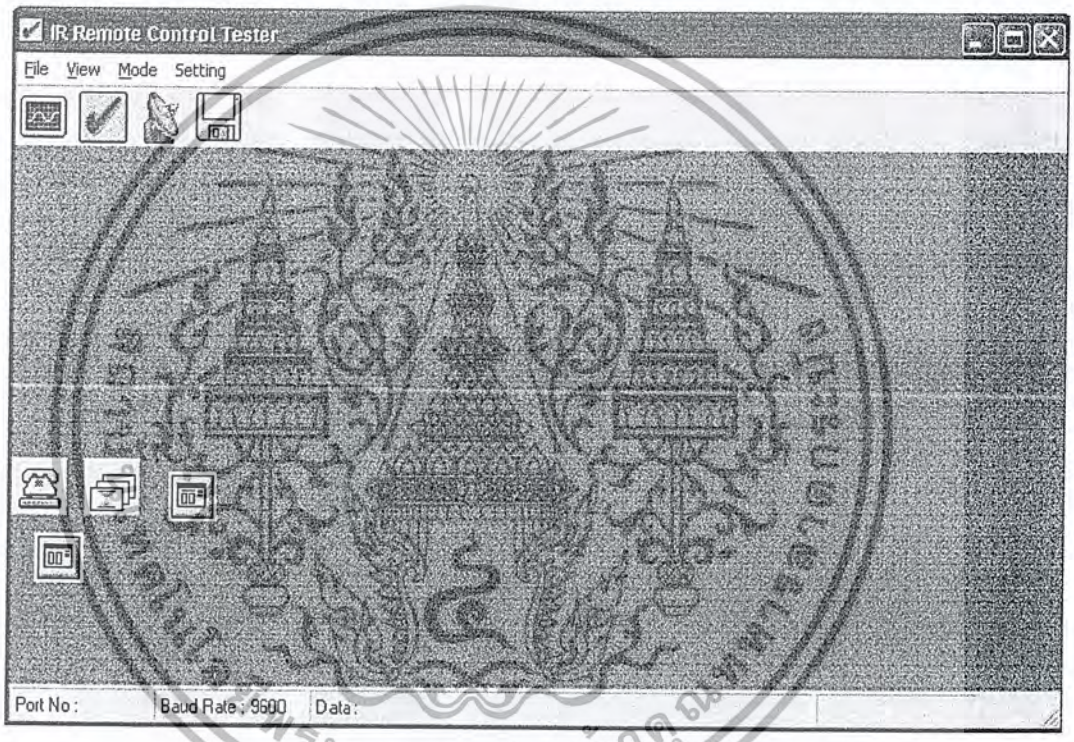
รูปที่ 3.18 โฟลวชาร์ตโปรแกรมย่อยในการรับสัญญาณของ JVC



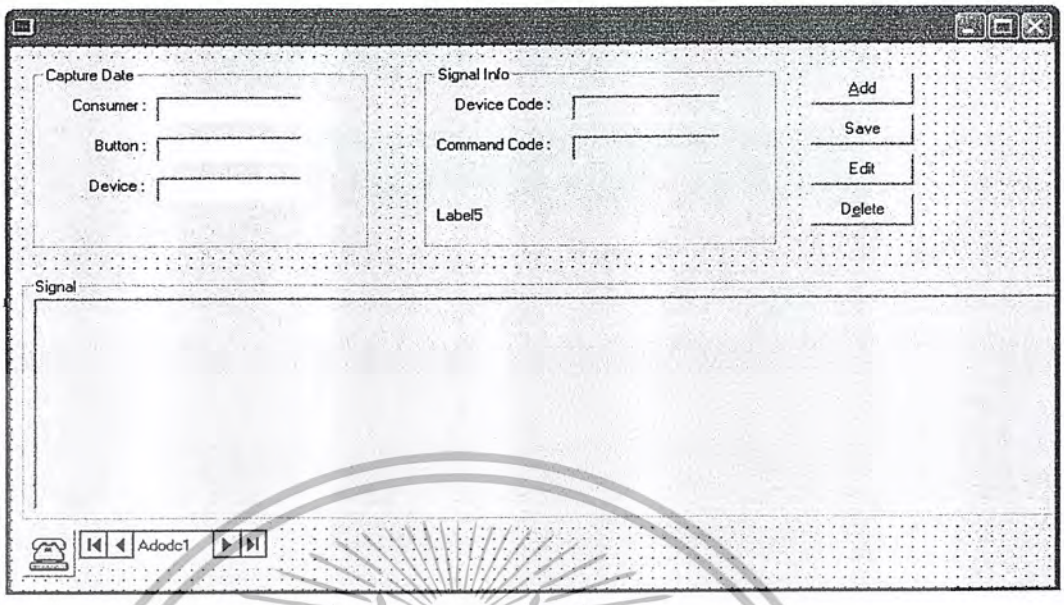
รูปที่ 3.19 โฟลวชาร์ตโปรแกรมย่อยในการรับสัญญาณของ NEC

3.2.1 การออกแบบซอฟแวร์ส่วนของการแสดงผลทางคอมพิวเตอร์

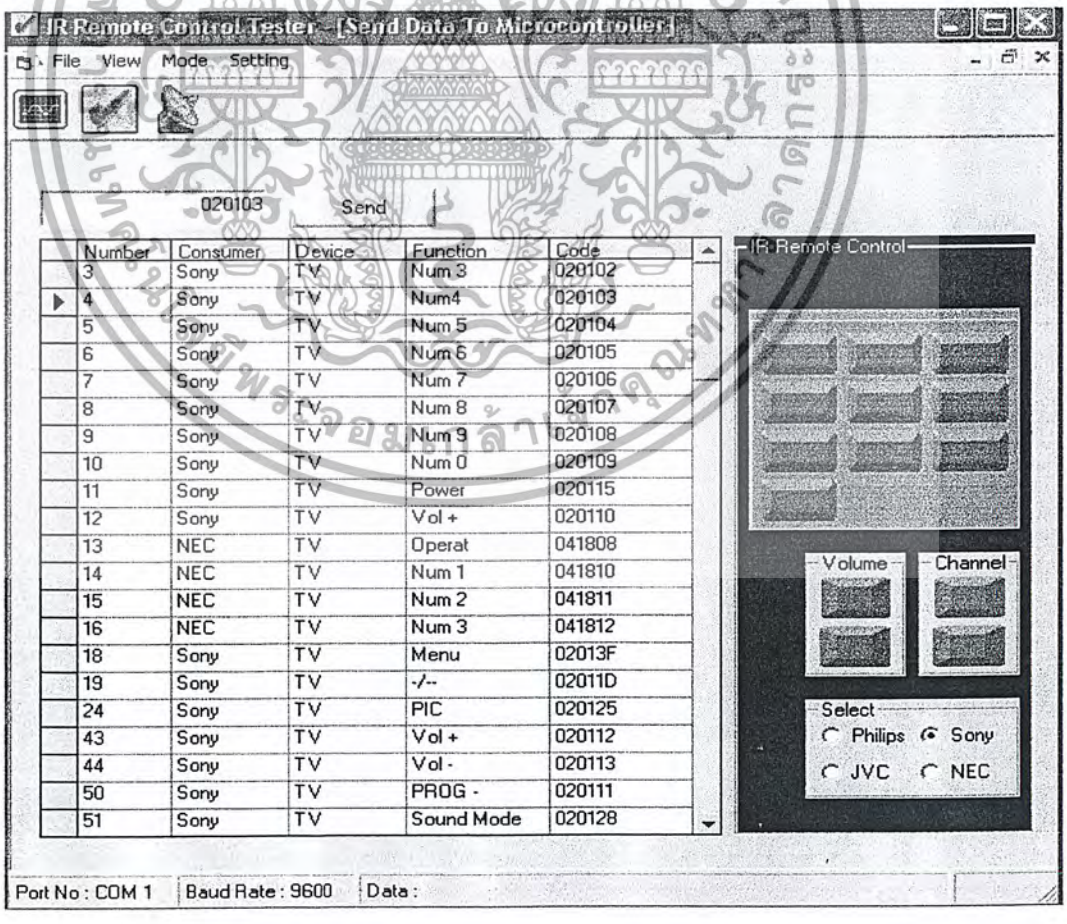
การออกแบบในส่วนของการแสดงผลทางคอมพิวเตอร์ จะใช้โปรแกรม Microsoft Visual Basic 6 ซึ่งการออกแบบในส่วนนี้ จะเป็นการนำผลที่ได้จากการประมวลผลในส่วนของไมโครคอนโทรลเลอร์ ที่รับผลผ่านทางพอร์ต RS-232 ของคอมพิวเตอร์ ซึ่งโปรแกรมที่ออกแบบจะแสดงผลในรูปของคลื่นสัญญาณ และรายละเอียดอื่น ๆ และโปรแกรมจะจัดเก็บข้อมูลที่ได้รับไว้ในรูปของฐานข้อมูล(Database)



รูปที่ 3.20 แสดงการออกแบบส่วนของโปรแกรมหลัก



รูปที่ 3.21 แสดงการออกแบบส่วนของการรับ



Number	Consumer	Device	Function	Code
3	Sony	TV	Num 3	020102
4	Sony	TV	Num4	020103
5	Sony	TV	Num 5	020104
6	Sony	TV	Num 6	020105
7	Sony	TV	Num 7	020106
8	Sony	TV	Num 8	020107
9	Sony	TV	Num 9	020108
10	Sony	TV	Num 0	020109
11	Sony	TV	Power	020115
12	Sony	TV	Vol +	020110
13	NEC	TV	Operat	041808
14	NEC	TV	Num 1	041810
15	NEC	TV	Num 2	041811
16	NEC	TV	Num 3	041812
18	Sony	TV	Menu	02013F
19	Sony	TV	-/--	02011D
24	Sony	TV	PIC	020125
43	Sony	TV	Vol +	020112
44	Sony	TV	Vol -	020113
50	Sony	TV	PROG -	020111
51	Sony	TV	Sound Mode	020128

รูปที่ 3.22 แสดงการออกแบบส่วนของการส่ง

3.2.3 การออกแบบส่วนของฐานข้อมูล (Database Design)

เนื่องจากฐานข้อมูลที่จะสร้าง เป็นฐานข้อมูลขนาดเล็กและมีความซับซ้อนน้อย ดังนั้นในส่วนของการสร้างฐานข้อมูลจึงใช้ โปรแกรม Microsoft Access 2000 สร้างฐานข้อมูลเพื่อเก็บข้อมูลของรีโมทคอนโทรยี่ห้อต่าง ๆและสร้างตารางได้ 4 ตารางดังต่อไปนี้

ตารางที่ 1 ตาราง Consumer

Con_ID	Con_Name

ตารางที่ 2 ตาราง Function

ID_Fun	Function

ตารางที่ 3 ตาราง Capture

Number	Consumer	Button	Data

ตารางที่ 4 ตาราง Remote

Number	Con_Name	Device	Function	Code	Comment

ConID	Con_Name
1	Sony
2	Philips
3	JVC
4	NEC
5	Panasonic
6	National
7	Pioneer
8	Hitachi
9	Apex

รูปที่ 3.23 ตาราง Consumer

Number	Consumer	Button	Data
1			
2			
3			
4			

รูปที่ 3.24 ตาราง Remote

ID_Fun	Function
1	Num 1
2	Num 2
3	Num 3
4	Num 4
5	Num 5
6	Num 6
7	Num 7
8	Num 8
9	Num 9
10	Num 0
11	Power
12	Vol+
13	Vol-
14	Ch+
15	Ch-
16	TV/Video
17	Picture +
18	Picture -
19	Mute
20	Select
21	PROGR +
22	PROGR -
23	PIC Mode
24	Stop
25	Play
26	Rew
27	FF
28	Pause
29	Sound Mode
30	-/--
31	A/B
32	Rec
33	Eject
34	Menu
35	Stereo
36	AV
37	DBB
38	Spatial
39	System
40	Zoom

ID_Fun	Function
41	Standard
42	Position Call
43	Timer
44	Clear
45	Control +
46	Control -
47	ZoomAdj
48	Operate
49	Display
50	Sleep
51	PRG1/2
52	Open/Close
53	Scan+
54	Scan -
55	Tuning/Preset
56	Memory
57	Band
58	Stereo/Mono

รูปที่ 3.25 ตาราง Function

Number	Con_Name	Device	Function	Code	Comment
1	Sony	TV	Num 1	020100	
2	Sony	TV	Num 2	020101	
3	Sony	TV	Num 3	020102	
4	Sony	TV	Num4	020103	
5	Sony	TV	Num 5	020104	
6	Sony	TV	Num 6	020105	
7	Sony	TV	Num 7	020106	
8	Sony	TV	Num 8	020107	
9	Sony	TV	Num 9	020108	
10	Sony	TV	Num 0	020109	
11	Sony	TV	Power	020115	
12	Sony	TV	Vol +	020110	
13	NEC	TV	Operat	041808	
14	NEC	TV	Num 1	041810	
15	NEC	TV	Num 2	041811	
16	NEC	TV	Num 3	041812	
18	Sony	TV	Menu	02013F	
19	Sony	TV	-/--	02011D	
24	Sony	TV	PIC	020125	
43	Sony	TV	Vol +	020112	
44	Sony	TV	Vol -	020113	
50	Sony	TV	PROG -	020111	
51	Sony	TV	Sound Mode	020128	
54	Sony	TV	A/B	020117	
71	Sony	TV	Timer Info	02013C	
79	Sony	TV	Timer	020136	
81	Sony	TV	Color -	020175	
82	Sony	TV	PIC Mode	020164	
83	Sony	TV	Select	02017C	
84	Sony	TV	Mute	020114	
85	NEC	TV	Num 4	041813	
86	NEC	TV	Num 5	041814	
87	NEC	TV	Num 6	041815	
88	NEC	TV	Num 7	041816	

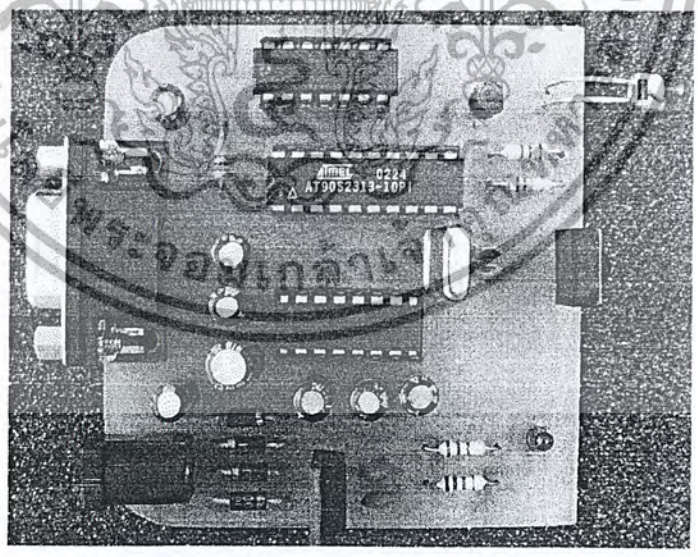
รูปที่ 3.26 ตาราง Remote

บทที่ 4
การทดลองและใช้งาน

4.1 คุณสมบัติเครื่องทดสอบอินฟราเรดรีโมทคอนโทรล

- รับและส่งสัญญาณอินฟราเรดได้
- เก็บรหัสสัญญาณอินฟราเรดในรูปของฐานข้อมูลได้
- แสดงผลการทำงานในรูปคลื่นและรหัสของสัญญาณ
- สามารถส่งสัญญาณผ่านทางคอมพิวเตอร์ไปควบคุมเครื่องใช้ไฟฟ้าได้
- ใช้แรงดัน 9 โวลท์ในการทำงาน โดยเร็กกูเลทเปลี่ยนแรงดันเหลือ 5 โวลท์

4.2 โครงสร้างภายในเครื่องทดสอบอินฟราเรดรีโมทคอนโทรล



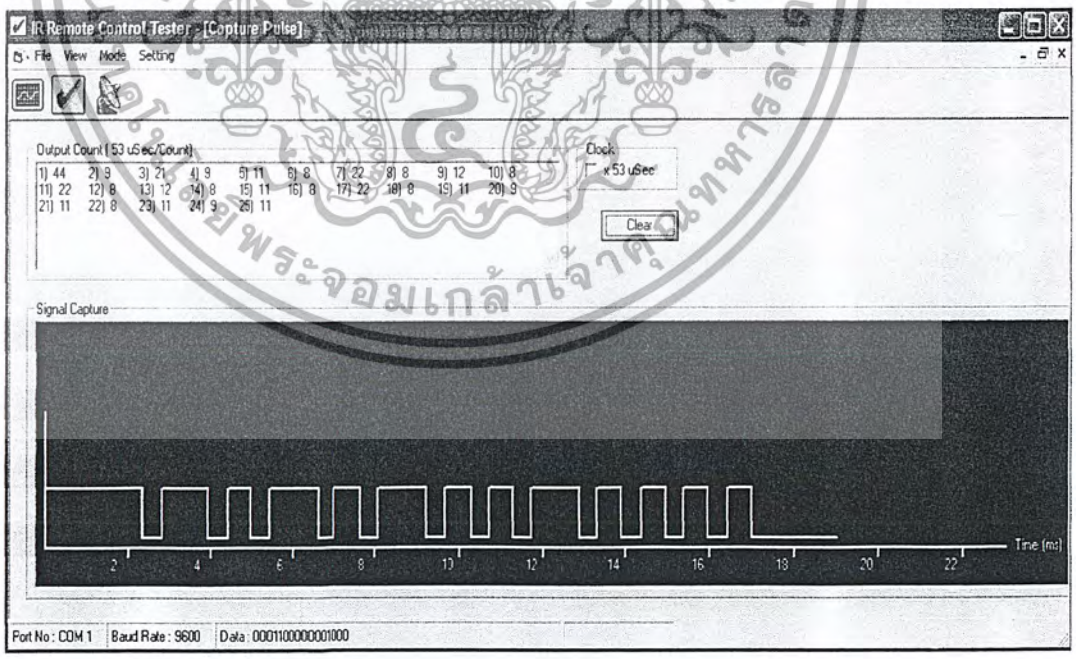
รูปที่ 4.1 โครงสร้างภายในเครื่องทดสอบอินฟราเรดรีโมทคอนโทรล

อุปกรณ์ที่ใช้ในการทดลอง

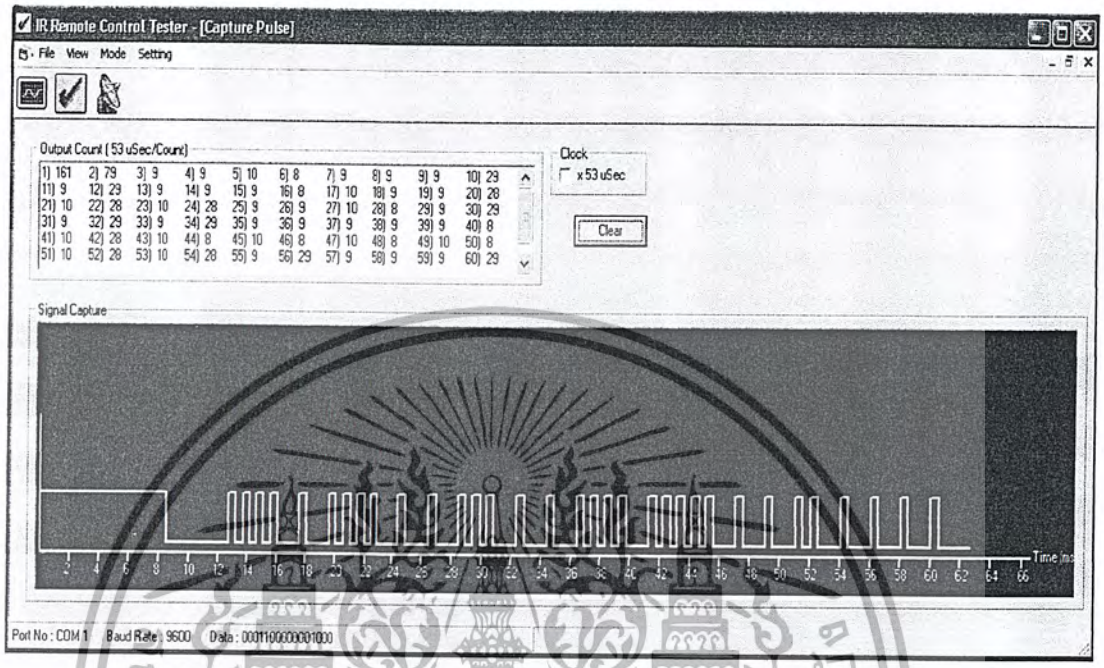
1. ชุดรับสัญญาณอินฟราเรดรีโมทคอนโทรล
2. คอมพิวเตอร์
3. โปรแกรมแสดงรหัสคำสั่ง
4. สายส่งสัญญาณพอร์ต RS 232
5. อินฟราเรดรีโมทคอนโทรลโทรทัศน์ยี่ห้อ SONY

ขั้นตอนการทดลอง

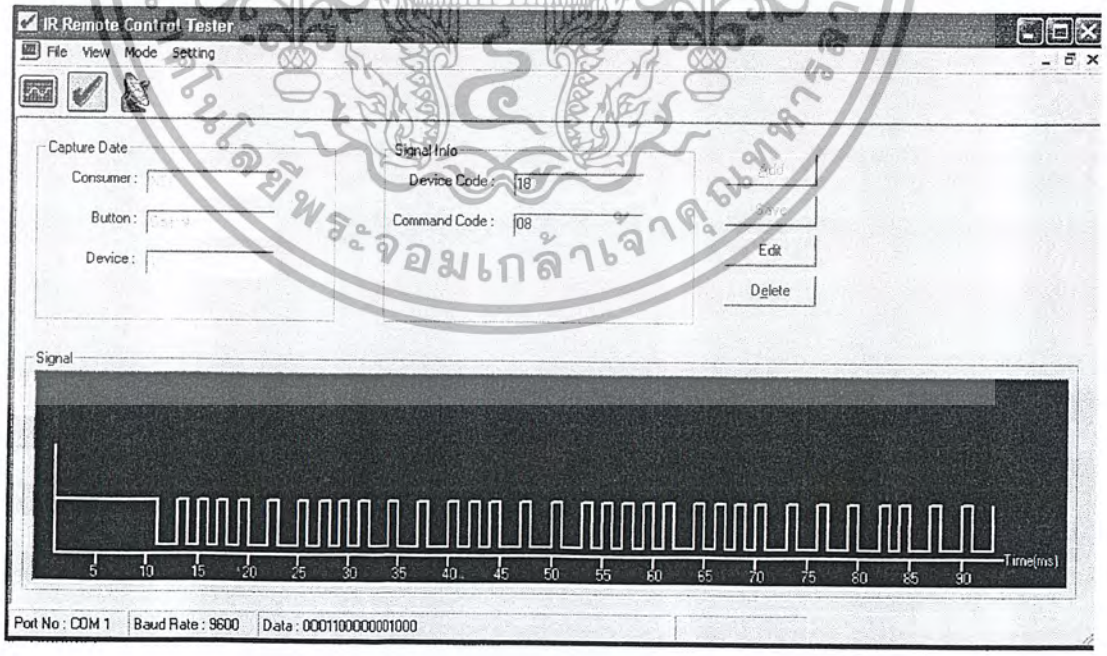
1. นำชุดรับสัญญาณอินฟราเรดรีโมทคอนโทรลมาติดตั้ง โดยต่อพอร์ต RS232 กับพอร์ตอนุกรมของคอมพิวเตอร์
2. นำอินฟราเรดรีโมทคอนโทรล กดส่งสัญญาณตามตารางรหัสคำสั่งยี่ห้อ SONY และ JVC
3. ศึกษาและเปรียบผลการทดลอง จากผลที่แสดงในโปรแกรมแสดงรหัสคำสั่งที่ได้กับตารางรหัสคำสั่งโทรทัศน์ยี่ห้อ SONY และ JVC



รูปที่ 4.2 รูปคลื่นที่ได้จากการรับสัญญาณของยี่ห้อ Sony



รูปที่ 4.3 รูปคลื่นที่ได้จากการรับสัญญาณของยี่ห้อ NEC



รูปที่ 4.4 รูปคลื่นที่ได้ตรวจสอบสัญญาณของยี่ห้อ NEC

สรุปผลการทดลอง

จากทดลองกดปุ่มการสั่งงานของรีโมทคอนโทรลยี่ห้อ SONY และ JVC แต่ละปุ่ม และเมื่อเปรียบเทียบตารางรหัสคำสั่ง ผลปรากฏว่ารหัสคำสั่งที่ได้จากโปรแกรมแสดงรหัสคำสั่ง ตรงกับตารางรหัสคำสั่ง ของโทรทัศน์ยี่ห้อ SONY แสดงว่าผลการออกแบบ เพื่อตรวจจับสัญญาณอินฟราเรดรีโมทคอนโทรลยี่ห้อ SONY สามารถตรวจจับสัญญาณได้อย่างถูกต้อง

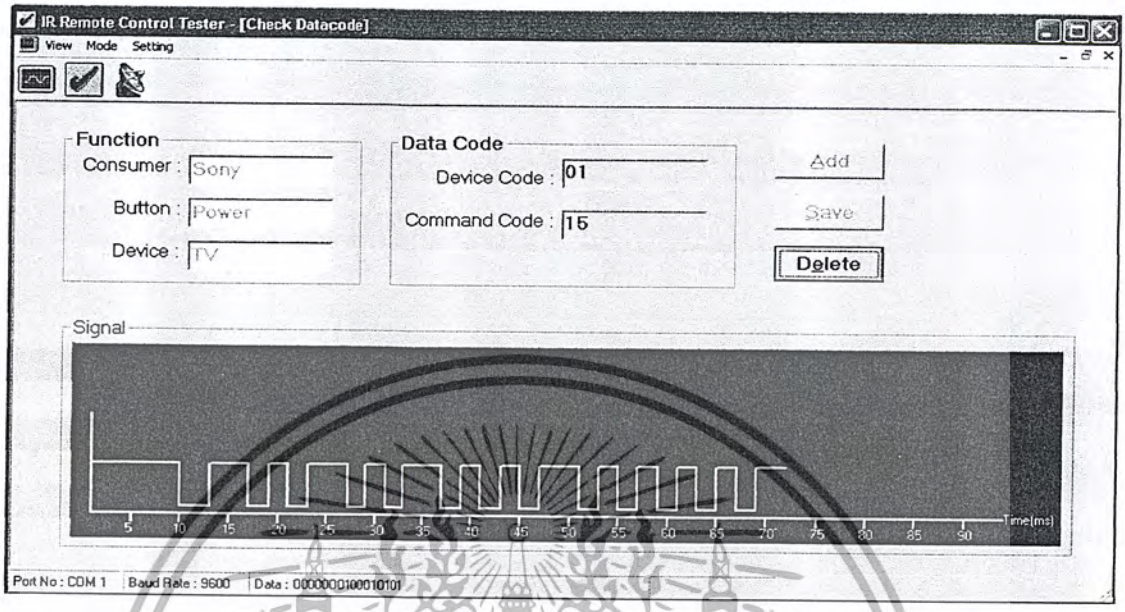
4.3 การใช้งานโปรแกรมแสดงผล

โปรแกรมที่ใช้แสดงผลการทำงาน ซึ่งเขียนด้วยภาษาวิชวลเบสิก มีหน้าต่างที่ใช้งานแบ่งออกเป็น 3 ส่วนคือ

1. การตรวจสอบรหัส
2. การตรวจจับสัญญาณ
3. การส่งสัญญาณ

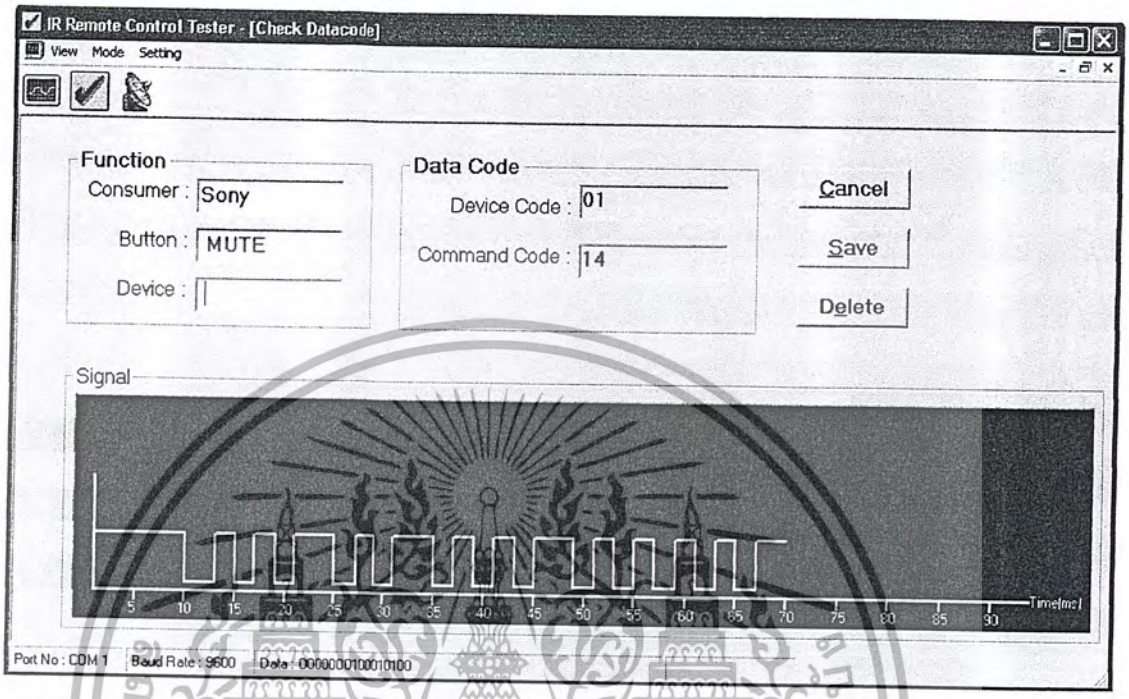
1. โหมดตรวจสอบรหัส

เมื่อผู้ใช้งานเรียกโปรแกรมขึ้นมา หน้าจอแรกที่ผู้ใช้จะพบคือโหมดตรวจสอบรหัสของรีโมทคอนโทรลที่รับเข้ามา ซึ่งสามารถถอดรหัสได้ 4 ยี่ห้อ คือ NEC , JVC , Sony และ Philips โดยโปรแกรมจะรับค่ารหัสที่ได้จากไมโครคอนโทรลเลอร์ซึ่งแสดงอยู่ในเฟรมของดาต้าโค๊ด แล้วจากนั้นจะไปทำการค้นหาจากฐานข้อมูล และแสดงออกมาในเฟรมของฟังก์ชั่น ซึ่งจะบอกถึงยี่ห้อปุ่มที่กดและอุปกรณ์ที่ใช้ควบคุม



รูปที่ 4.5 การใช้งานในโหมดตรวจสอบรหัส

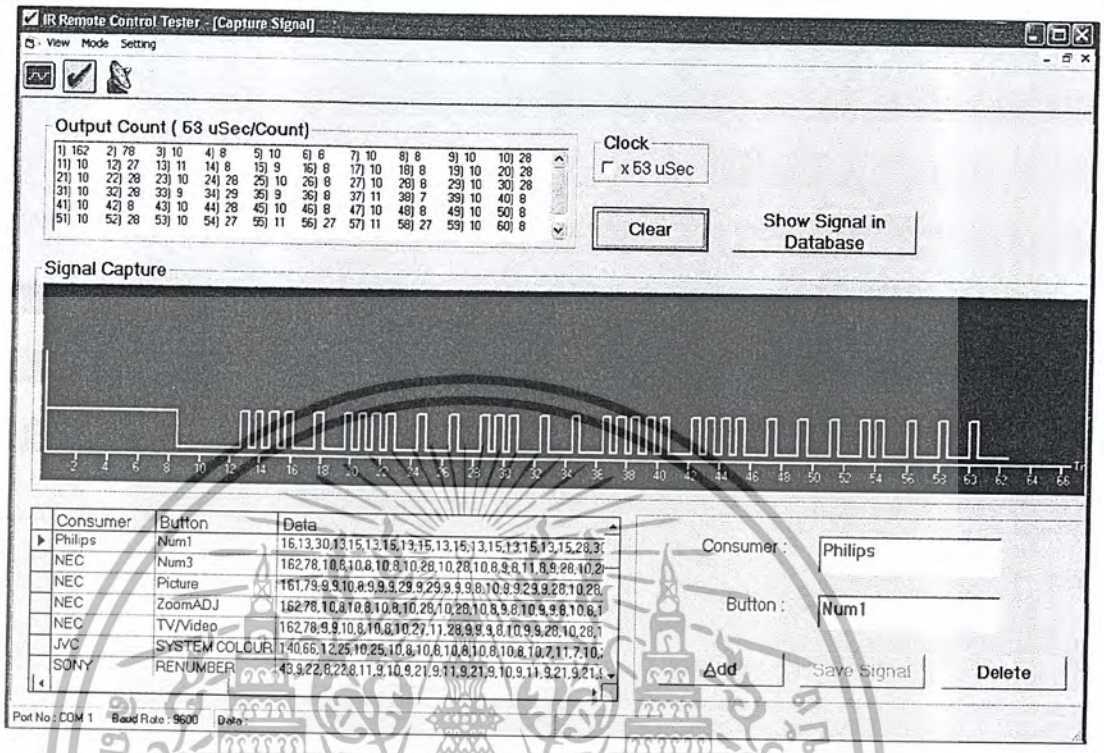
ถ้าหากว่าตัวไมโครคอนโทรลเลอร์สามารถถอดรหัสได้ แต่ไม่มีอยู่ในฐานข้อมูล ผู้ใช้จะสามารถเพิ่มรหัสและฟังก์ชันนั้นเข้าไปในฐานข้อมูลได้ โดยคลิกที่ปุ่ม Add และทำการ Save ข้อมูลก็จะถูกบันทึกลงไปในฐานข้อมูล ในขณะที่ทำการบันทึกนั้นถ้าหากผู้ใช้งานเปลี่ยนใจก็สามารถยกเลิกการบันทึกข้อมูลได้



รูปที่ 4.6 การบันทึกข้อมูลลงในฐานข้อมูล

2. การใช้งานในโหมดตรวจจับสัญญาณ

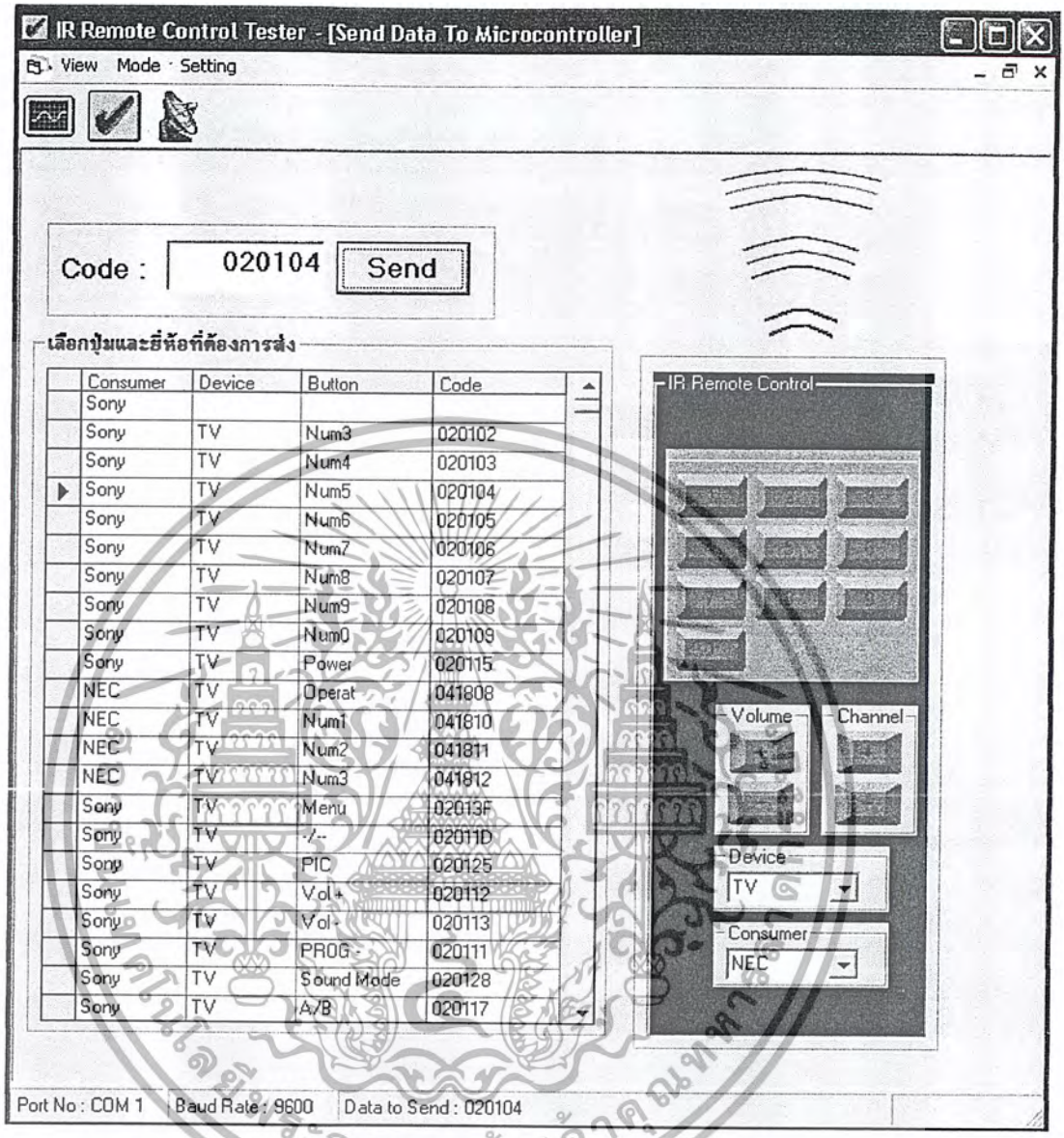
ในกรณีที่ไมโครคอนโทรลเลอร์ไม่สามารถจะถอดรหัสออกมาได้ โปรแกรมจะทำการเปลี่ยนหน้าต่างมายังโหมดนี้ทันที เพื่อแสดงให้เห็นถึงลักษณะของรูปคลื่นสัญญาณที่รับได้ โดยจะแสดงเป็นกราฟในรูปแบบของฐานเวลา และจำนวนสัญญาณพัลส์ที่ถูกนับด้วยคาบเวลาละ 53 uSec โดยแสดงออกทางเท็กซ์บ็อกซ์ ซึ่งสามารถที่จะจัดเก็บไว้ในฐานข้อมูลได้ โดยการคลิกที่ปุ่ม Add เพื่อทำการเพิ่ม และคลิกที่ปุ่ม Save เพื่อทำการบันทึก กราฟที่แสดงออกมานั้นเป็นค่าที่รับได้จริงจากตัวรีโมทคอนโทรล ซึ่งค่าของสเกลจะเปลี่ยนไปตามค่าเวลาทั้งหมดที่รับเข้ามา



รูปที่ 4.7 การใช้งานในส่วนของการจับสัญญาณ

3. การใช้งานในโหมดส่งสัญญาณ

การทำงานในส่วนนี้สัญญาณข้อมูลที่จะส่งออกไปจากตัวโปรแกรม จะได้มาจากฐานข้อมูลตัวเดียวกันกับที่ใช้ในโหมดตรวจสอบสัญญาณ โดยผู้ใช้งานจะทำการเลือกฟังก์ชั่นที่ต้องการส่งออกไปควบคุมอุปกรณ์จากตารางฐานข้อมูล จากนั้นให้กดปุ่ม Send เพื่อส่งข้อมูลออกไปให้กับไมโครคอนโทรลเลอร์ หรืออีกวิธีหนึ่งก็คือผู้ใช้สามารถกดปุ่มได้โดยตรงจากรูปภาพที่เป็นรีโมตคอนโทรล ซึ่งสามารถเลือกยี่ห้อและอุปกรณ์ที่ต้องการควบคุมได้จากลิสต์บล็อก ดังแสดงในรูป4.8



รูปที่ 4.8 การทำงานในโหมดส่งสัญญาณ

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

จากการนำเครื่องทดสอบอินฟราเรดรีโมทคอนโทรลไปทดสอบอ่านค่า ของรีโมทยี่ห้อต่างๆ ปรากฏว่าสามารถที่จะถอดรหัสสัญญาณได้ 4 ยี่ห้อ คือ Philips , Sony , JVC , NEC เนื่องจากหารูปแบบของสัญญาณได้เพียงเท่านี้ ซึ่งสามารถอ่านค่าได้ถูกต้อง ส่วนรีโมทยี่ห้ออื่นๆนั้นสามารถที่จะใช้โหมดแสดงรูปคลื่นสัญญาณอออกมาได้โดยมีการอ่านค่าออกมาเป็นคาบเวลาจริง ส่วนการส่งกลับออกไปนั้นพบว่าข้อมูลที่ส่งออกผสมกับความถี่ 38 KHz ระยะทางในการส่งและความไวที่เครื่องรับสามารถที่จะรับสัญญาณได้นั้นจะขึ้นอยู่กับค่าดิวตี้ไซเคิลของความถี่พาหะ 38 KHz นี้ทดสอบได้จากการเปลี่ยนค่ารีโหลดภายในไมโครคอนโทรลเลอร์ ปัญหาอีกอย่างหนึ่งก็คือเรื่องคาบเวลาในการส่งสัญญาณอาจจะเพี้ยนไปเล็กน้อยเนื่องจากใช้โปรแกรมในการหน่วงเวลา

ในเรื่องของการรับสัญญาณนั้นไม่ค่อยเกิดปัญหาเท่าใดนักเพราะถ้าหากไม่พบข้อมูลของทั้ง 4 ยี่ห้อ โปรแกรมก็จะทำการสวิตช์ไปยังโหมดจับรูปคลื่นสัญญาณโดยอัตโนมัติ ด้านการส่งสัญญาณจะขึ้นอยู่กับความถี่พาหะ 38 KHz ถ้าหากเครื่องใช้ไฟฟ้ามีความไวในการรับสัญญาณไม่ตรงกันแล้วก็จะไม่สามารถควบคุมอุปกรณ์ไฟฟ้าชนิดนั้น ๆ ได้

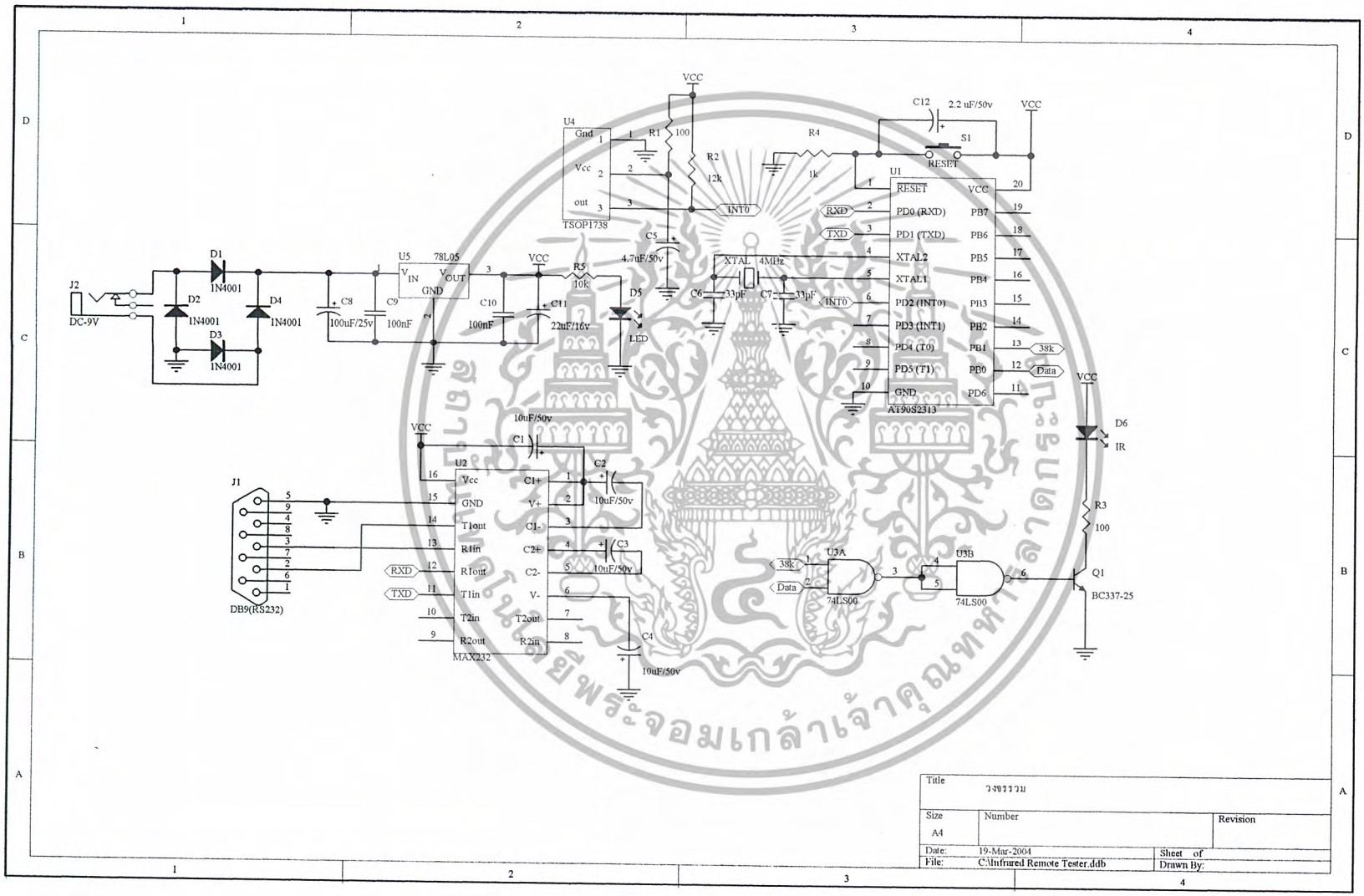
แนวทางในการพัฒนาต่อ

สำหรับเครื่องทดสอบอินฟราเรดรีโมทคอนโทรล ควรมีการพัฒนาต่อในด้าน ฮาร์ดแวร์ (Hardware) ให้สามารถรับสัญญาณให้ได้มากกว่าเดิม ส่วนทางด้านซอฟแวร์(Software)ควรพัฒนาให้มีรูปแบบในการประมวลผลที่มากขึ้น หรือเพิ่มแอปพริเคชั่น (Applicaton) ให้สามารถที่จะนำผลของสัญญาณมาออกแบบการส่งสัญญาณอินฟราเรดรีโมทคอนโทรล หรือการสร้างรูปแบบการส่งที่มีระบบความปลอดภัยสูง ตลอดจนพัฒนาส่วนแสดงผลให้มีรูปแบบที่ดีขึ้น

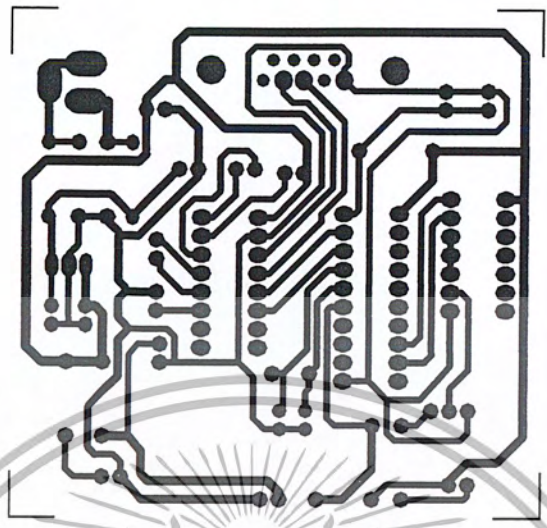
บรรณานุกรม

1. Claus Kuhnel , " AVR RISE Microcontroller Handbook " , Boston Oxford
2. www.AVRfreaks.net
3. www.epanorama.net
4. www.xs4all.nl
5. www.cypressmicro.com

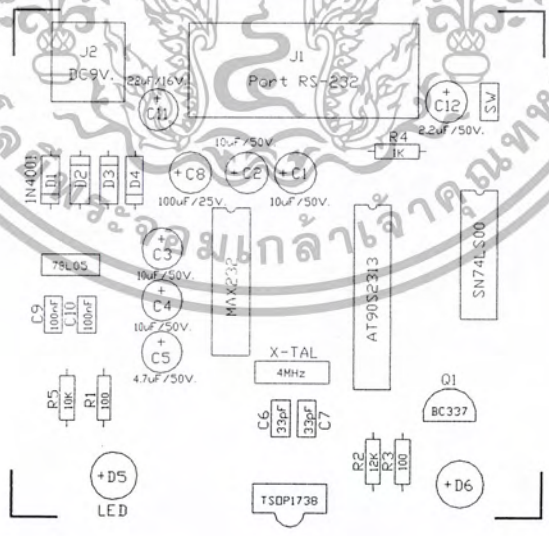
ภาคผนวก ก



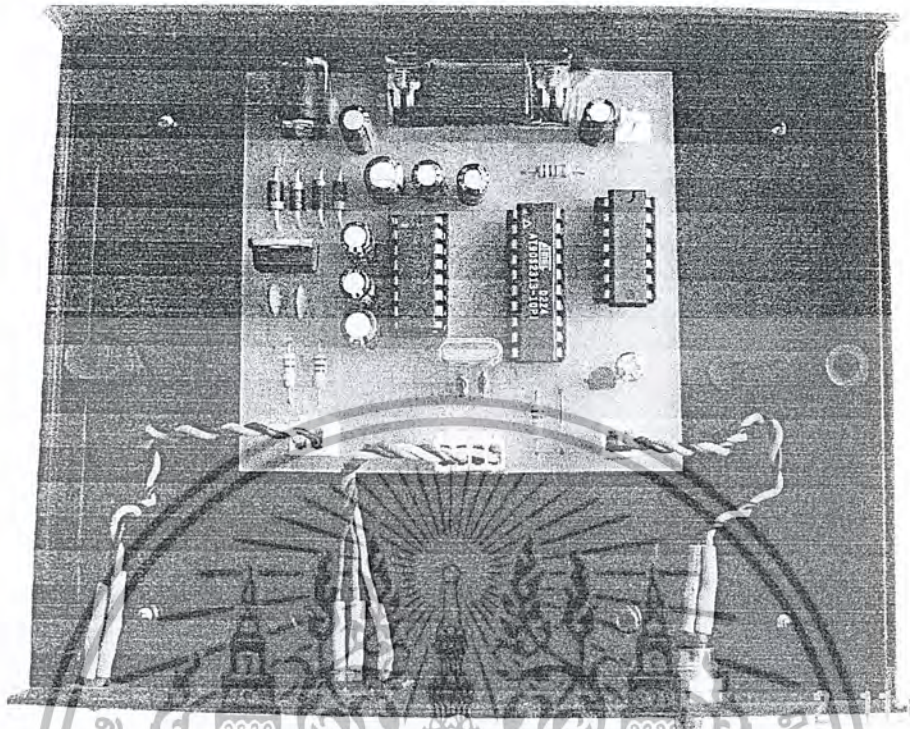
รูปที่ ก-1 วงจรรวม



รูปที่ ก-2 แสดงลายวงจร

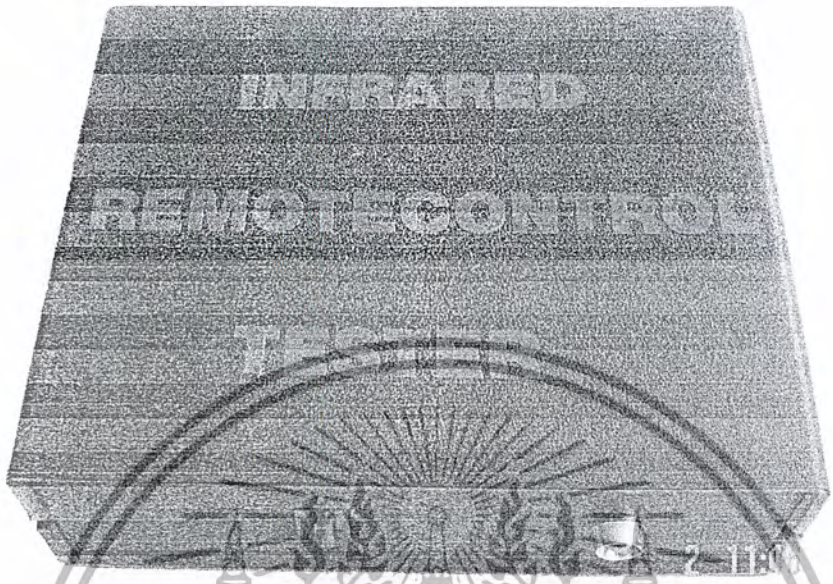


รูปที่ ก-3 แสดงการวางอุปกรณ์



รูปที่ ก-4 การติดตั้งภายในเครื่อง

รูปที่ ก-5 ด้านหลังเครื่อง



รูปที่ ก-6 ด้านหน้าเครื่อง

ภาคผนวก ข

Source Code of Microcotroller

```
.include        "2313def.inc"
.equ    IR      = PD2
.def    Byte    = r7
.def    Bit     = r8
.def    dlycnt1 = r9
.def    dlycnt2 = r10
.def    time    = r11
.def    Regmode     = r12
.def    temp    = r16
.def    Ascii   = r17
.def    var1    = r18
.def    var2    = r19
.def    var3    = r20
.def    var4    = r21
.def    counter= r22
.def    reload  = r23
.def    address = r24
.def    command     = r25

;************ Interrupt vector table***************
                rjmp    RESET           ;000
                rjmp    EXT_INT0        ;IRQ0 Handle
                reti                    ;IRQ1
                reti                    ;Timer1 capture
                reti                    ;Timer1 Compare
                reti                    ;Timer1 Overflow
                rjmp    TIM0_OVF        ;Timer0 Overflow
                reti                    ;UART RX
                reti                    ;UART UDR
                reti                    ;UART TX
                reti                    ;Analog Comparator

;************ Initial Register **********************
RESET:          ldi     temp,LOW(RAMEND)
                out     SPL,temp
                cbi     DDRD,IR             ;IR - is input
                sbi     PORTD,IR
                clr     temp
```

```
                out     DDRD,temp   ;Use pullup
                clr     temp
                out     PORTB,temp  ;Clear Output Port
                ser     temp
                out     DDRB,temp   ;Set Direction Port
                ldi     temp,0b00001010 ;Setup Interrupt for trigging on falling
edge
                out     MCUCR,temp
                ldi     temp,(1<<INT0)          ;Enable External Interrupt0
                out     GIMSK,temp
                ldi     temp,25         ;UART Baudrate 9600 @ 4 MHz Clock
                out     UBRR,temp
                clr     Byte            ;Clear Input counter(3
Bytye:Consumer,Adderss,Command)
                clr     r27
                ldi     r26,$60
                ldi     reload,202      ;256-54=202
                ldi     temp,0b00000001
                out     TCCR0,temp  ;Prescler CK  (0.25us)
                out     TCNT0,reload
                sei

;******************** Main Program **********************
;*      Receive Data from RS-232                        *
;*      Select Mode                                     *
;*******************************************************
Mode:           sbi     UCR,RXEN        ;Rx Enable
                sbis    USR,RXC         ;Wait until Data Register is empty
                rjmp    Mode
                in      temp,UDR
                cbi     UCR,RXEN
                clr     Regmode
                cpi     temp,$58        ;Jump to Mode Send
                brne    Chang
                rcall   getc
                ret

Chang:          cpi     temp,$5A
                breq    Mode1
                ldi     temp,$00
                mov     Regmode,temp    ;Set Mode
                rjmp    Mode

Mode1:          ldi     temp,$FF
                mov     Regmode,temp
```

```
                rjmp    Mode                ;Set mode Receive = FF

;********************** Mode Send ************************

getc:           sbi     UCR,RXEN
                sbis    USR,RXC
                rjmp    getc
                in      Ascii,UDR           ;Read character
                cbi     UCR,RXEN            ;Rx Disable
                mov     temp,Ascii
                cpi     Ascii,$5A
                breq    Chang
                cpi     Ascii,$7A
                breq    Chang

                subi    Ascii,$30           ;Convert Ascii to Binary
                cpi     Ascii,$10
                brge    Sub_bi
                st      X+,Ascii

                inc     Byte                ;Count Data 6 Byte
                mov     temp,Byte
                cpi     temp,$06
                breq    Data_Comp           ;Brane if Receive Complete
                rjmp    getc

Sub_bi:         subi    Ascii,$07           ;Convert Ascii to Binary
                st      X+,Ascii

                inc     Byte                ;Count Data 6 Byte
                mov     temp,Byte
                cpi     temp,$06
                breq    Data_Comp           ;Brane if Receive Complete
                rjmp    getc

Data_Comp:      ldi     r26,$60             ;get data from memory and compare
                ld      r1,X+
                ld      r2,X+
                ld      r3,X+
                ld      r4,X+
                ld      r5,X+
                ld      r6,X
```

```
        swap    r1              ;Swap Nibbles
        bst     r2,0            ;Bit Store from Register to T
        bld     r1,0            ;Bit load from T to Register
        bst     r2,1
        bld     r1,1
        bst     r2,2
        bld     r1,2
        bst     r2,3
        bld     r1,3            ;r1 is Register Consumer

        swap    r3
        bst     r4,0            ;Bit Store from Register to T
        bld     r3,0            ;Bit load from T to Register
        bst     r4,1
        bld     r3,1
        bst     r4,2
        bld     r3,2
        bst     r4,3
        bld     r3,3            ;Register Address

        swap    r5
        bst     r6,0            ;Bit Store from Register to T
        bld     r5,0            ;Bit load from T to Register
        bst     r6,1
        bld     r5,1
        bst     r6,2
        bld     r5,2
        bst     r6,3
        bld     r5,3                    ;r5 is Register Command

        clr     Byte
        mov     temp,r1

        cpi     temp,4
        brne    Comp_1          ;Compare find Comsumer
        rcall   Send_NEC
        ret

Comp_1:     cpi     temp,3
        brne    Comp_2
        rcall   Send_JVC
        ret
```

```
Comp_2:         cpi     temp,2
                brne    Comp_3
                rcall   Send_Sony
                ret

Comp_3:         cpi     temp,1
                brne    Error
                rcall   Send_Philips
                ret

Error:          nop
                ret


;********************************** Send Data NEC 32 bit
**********************************
;*
*
;*  | Hp | Hs | Address 8 Bit | Invert Address 8 Bit | Command 8 Bit | Invert Command 8
Bit  *
;*
*
;*****************************************************************************
************************
Send_NEC:       mov     var1,r3             ;Set Address
                mov     var2,r3             ;set Invert Adderss
                mov     var3,r5             ;Set Command
                mov     var4,r5             ;Set Invert Command
                ldi     temp,(0<<INT0)      ;Disable External Interrupt0
                out     GIMSK,temp
                ldi     temp,TOV0<<1
                out     TIMSK,temp          ;T/C0 Interrupt Enable

                sbi     PORTB,PB0
                rcall   delay_88ms          ;Header +
                cbi     PORTB,PB0
                rcall   delay_44ms          ;Header -
                clc
                ldi     temp,8
                mov     bit,temp
Tx_NECBy1:      ror     var1                ;Send Address Code
                brcs    NEC_1               ;Branch if carry is set
                rcall   TxN_0
```

```
                dec     bit
                brne    Tx_NECBy1
                rjmp    Tx_NECBy2           ;Next Byte
NEC_1:          rcall   TxN_1
                dec     bit
                brne    Tx_NECBy1

Tx_NECBy2:      ldi     temp,8
                mov     bit,temp
                com     var2
Tx_NEC22:       ror     var2                ;Send Invert Address
                brcs    NEC_2
                rcall   TxN_0
                dec     bit
                brne    Tx_NEC22
                rjmp    Tx_NECBy3           ;Next Byte

NEC_2:          rcall   TxN_1
                dec     bit
                brne    Tx_NEC22

Tx_NECBy3:      ldi     temp,8
                mov     bit,temp
Tx_NEC33:       ror     var3                ;Send Command Code
                brcs    NEC_3
                rcall   TxN_0
                dec     bit
                brne    Tx_NEC33
                rjmp    Tx_NECBy4           ;Next Byte
NEC_3:          rcall   TxN_1
                dec     bit
                brne    Tx_NEC33

Tx_NECBy4:      ldi     temp,8
                mov     bit,temp
                com     var4
Tx_NEC44:       ror     var4                ;Send Invert Command Code
                brcs    NEC_4
                rcall   TxN_0
                dec     bit
                brne    Tx_NEC44
                rjmp    Tx_NEC_End          ;Next Byte
NEC_4:          rcall   TxN_1
```

```
                dec     bit
                brne    Tx_NEC44

Tx_NEC_End:cbi  PORTB,PB1
                cbi     PORTB,PB0
                rcall   delay
                ldi     temp,TOV0<<0
                out     TIMSK,temp              ;T/C0 Interrupt Enable
                cli
                ldi     temp,(1<<INT0)          ;Enable External Interrupt0
                out     GIMSK,temp
                sei
                ret

;*********************************
TxN_0:          sbi     PORTB,PB0
                rcall   delay_400us
                cbi     PORTB,PB0
                rcall   delay_400us
                ret

TxN_1:          sbi     PORTB,PB0
                rcall   delay_400us
                cbi     PORTB,PB0
                rcall   delay1100us
                rcall   delay_400us
                ret

;************************ Send Data JVC 16 bit **********************
;*                                                                  *
;*       | Hp | Hs | Address 8 Bit | Command 8 Bit |               *
;*                                                                  *
;********************************************************************

Send_JVC:       mov     var1,r3                 ;Set Address
                mov     var2,r5                 ;set Command
                ldi     temp,(0<<INT0)          ;Disable External Interrupt0
                out     GIMSK,temp
                ldi     temp,TOV0<<1
                out     TIMSK,temp              ;T/C0 Interrupt Enable


                sbi     PORTB,PB0
```

```
                rcall   delay_72ms              ;Header +
                cbi     PORTB,PB0
                rcall   delay_33ms              ;Header -

                ldi     temp,8
                mov     bit,temp
                clc
JVC_Add:        ror     var1                    ;Send Address Code
                brcs    JVC_1                   ;Branch if carry is set
                rcall   TxJ_0
                dec     bit
                brne    JVC_Add
                rjmp    JVC_Next                ;Next Byte

JVC_1:          rcall   TxJ_1
                dec     bit
                brne    JVC_Add

JVC_Next:       ldi     temp,8
                mov     bit,temp
JVC_Comm:       ror     var2                    ;Send Command
                brcs    JVC_1n
                rcall   TxJ_0
                dec     bit
                brne    JVC_Comm
                rjmp    Tx_JVC_End

JVC_1n:         rcall   TxJ_1
                dec     bit
                brne    JVC_Comm

Tx_JVC_End:     cbi     PORTB,PB1
                cbi     PORTB,PB0
                rcall   delay
                ldi     temp,TOV0<<0
                out     TIMSK,temp              ;T/C0 Interrupt Disable
                cli
                ldi     temp,(1<<INT0)          ;Enable External Interrupt0
                out     GIMSK,temp
                sei
                ret
;*********************************
TxJ_0:          sbi     PORTB,PB0
                rcall   delay_400us
```

```
                cbi     PORTB,PB0
                rcall   delay_400us
                ret

TxJ_1:          sbi     PORTB,PB0
                rcall   delay_400us
                cbi     PORTB,PB0
                rcall   delay1100us
                rcall   delay_400us
                ret
;************************ Sony Send Data 12 bit
*****************************
;*                                                  *
;*  | Hp | Hs | C6 | C5 | C4 | C3 | C2 | C1 | C0 | A4 | A3 | A2 | A1 | A0 | *
;*                                                  *
;****************************************************************************
********
Send_Sony:      mov     var1,r3
                mov     var2,r5
                ldi     temp,507
                mov     bit,temp                ;Count Command Bit
                ldi     temp,(0<<INT0)          ;Enable External Interrupt0
                out     GIMSK,temp
                ldi     temp,TOV0<<1
                out     TIMSK,temp              ;T/C0 Interrupt Enable

                sbi     PORTB,PB0
                rcall   delay_2200us            ;Header
                cbi     PORTB,PB0
                rcall   delay_480us
                clc
Sony_Comm:      ror     var2
                brcs    Send_1                  ;Branch if carry is set
                sbi     PORTB,PB0
                rcall   delay_480us
                cbi     PORTB,PB0
                rcall   delay_480us
                dec     bit
                brne    Sony_Comm
                rjmp    So_Next
```

```
Send_1:         sbi     PORTB,PB0
                rcall   delay_480us
                rcall   delay_480us
                cbi     PORTB,PB0
                rcall   delay_480us
                dec     bit
                brne    Sony_Comm
                rjmp    So_Next

So_Next:        ldi     temp,$05
                mov     bit,temp
Sony_Add:       ror     var1
                brcs    Send_1i                 ;Branch if carry is set
                sbi     PORTB,PB0
                rcall   delay_480us
                cbi     PORTB,PB0
                rcall   delay_480us
                dec     bit
                brne    Sony_Add
                rjmp    Tx_Sony_End

Send_1i:        sbi     PORTB,PB0               ;PB0 = output
                rcall   delay_480us
                rcall   delay_480us
                cbi     PORTB,PB0
                rcall   delay_480us
                dec     bit
                brne    Sony_Add

Tx_Sony_End:    cbi     PORTB,PB1
                cbi     PORTB,PB0
                rcall   delay
                ldi     temp,TOV0<<0
                out     TIMSK,temp              ;T/C0 Interrupt Disable
                cli
                ldi     temp,(1<<INT0)          ;Enable External Interrupt0
                out     GIMSK,temp
                sei
                ret
```

```
;****************** Philips Send Data 14 bit ***************************
;*
                                                              *
;*  | S | S | T | A4 | A3 | A2 | A1 | A0 | C5 | C4 | C3 | C2 | C1 | C0 | *
;*
                                                              *
;*****************************************************************
*****
Send_Philips: mov   var1,r3
              mov   var2,r5
              ldi   temp,$5
              mov   bit,temp
              ldi   temp,(0<<INT0)        ;Disable External Interrupt0
              out   GIMSK,temp
              ldi   temp,TOV0<<1
              out   TIMSK,temp            ;T/C0 Interrupt Enable
              ldi   temp,3
Ph_St:        cbi   PORTB,PB0
              rcall delay_800us           ;Header
              sbi   PORTB,PB0
              rcall delay_800us
              dec   temp
              brne  Ph_St
              clc
Ph_Add:       ror   var1
              brcs  Ph_1                  ;Branch if carry is set
              sbi   PORTB,PB0
              rcall delay_800us
              cbi   PORTB,PB0
              rcall delay_800us
              dec   bit
              brne  Ph_Add
              rjmp  Ph_Next

Ph_1:         cbi   PORTB,PB0             ;PB0 = output
              rcall delay_800us
              sbi   PORTB,PB0
              rcall delay_800us
              dec   bit
              brne  Ph_Add
Ph_Next:      ldi   temp,$6
              mov   bit,temp              ;Count Address 5 Bit
```

```
Ph_Comm:    ror     var2
            brcs    Ph_1n               ;Branch if carry is set
            sbi     PORTB,PB0
            rcall   delay_800us
            cbi     PORTB,PB0
            rcall   delay_800us
            dec     bit
            brne    Ph_Comm
            rjmp    Tx_Ph_End
Ph_1n:      cbi     PORTB,PB0
            rcall   delay_800us
            sbi     PORTB,PB0
            rcall   delay_800us
            dec     bit
            brne    Ph_Comm

Tx_Ph_End:  cbi     PORTB,PB1
            cbi     PORTB,PB0
            rcall   delay
            ldi     temp,TOV0<<0
            out     TIMSK,temp          ;T/C0 Interrupt Disable
            cli
            ldi     temp,(1<<INT0)      ;Enable External Interrupt0
            out     GIMSK,temp
            sei
            ret

;***************** Interrupt Service Routen 0 **************
;*          Select Function                                *
;*          1) Register_Mode = FF --> Check Data           *
;*          2) Register_Mode = 00 --> Capture Signal       *
;*****************************************************************
EXT_INT0:   cli
            mov     temp,Regmode
            cpi     temp,$FF
            breq    Func1               ;Check Data
            cpi     temp,$00
            breq    Func2               ;Capture
            ldi     temp,$13            ;Send Error
            rcall   putc
            sei
            reti
```

```
Func1:      rcall   Check
END_Data:   ldi     temp,$FF            ;End of Data Send FF13
            rcall   putc
            ldi     temp,$13
            rcall   putc
            rcall   delay
            rcall   delay
            sei
            reti

Func2:      rcall   Capture
            rcall   delay
            sei
            reti                        ;Return to Main

;********************** Interrupt Overflow ****************
;*          Generate Frequency 38 KHz to port PB1          *
;*          for Modurate Signal to send data               *
;*          T = 1/38KHz = 26 uS                            *
;*          Counter Timer Resolution 0.25 uS               *
;*          26uS/0.25 = 104 time                           *
;*                                                         *
;***********************************************************
TIM0_OVF:   brts    High                ;Test T Flag
            cbi     PORTB,PB1           ;Clear PB1
            ldi     reload,208          ;202    195:60
            out     TCNT0,reload
            set
            reti

High:       sbi     PORTB,PB1           ;206
            ldi     reload,198          ;217    211:44
            out     TCNT0,reload
            clt
            reti

;********** Initial Register **************
Check:      clr     counter             ;Clear Register Conuter
            clc
Sampling:   rcall   delay_53us          ;Check Head Pulse
            sbis    PIND,IR
            rjmp    Add_count
            rjmp    Case
```

```
Add_count:  inc   counter
            rjmp  Sampling

Case:       cpi   counter,155
            brlo  Case1
            rjmp  Rec_NEC

Case1:      cpi   counter,135
            brlo  Case2
            rjmp  Rec_JVC

Case2:      cpi   counter,35
            brlo  Case3
            rjmp  Rec_Sony

Case3:      cpi   counter,10
            brlo  Case4

            rjmp  Rec_Philips

Case4:      ret

;****************** Receive Philips ************************

Rec_Philips: clc
            ldi   var1,s08          ;Set Commmand 6 bit left
            ldi   var2,s00          ;Set Address (2 bit Var1 + 3 bit Var2)

Ph_St2:     sbis  PIND,IR           ;Start bit of second bit
            rjmp  Ph_St2

Ph_Md2:           sbic  PIND,IR     ;Middle bit
            rjmp  Ph_Md2
Ph_Measure: rcall delay_1200us      ;Measurement level of Signal

            sbis  PIND,IR           ;Received Bit is ZERO
            rjmp  Ph_Rec0           ;Received Bit is ONE
            sec                     ;Set Carry Bit =0
            ror   var1              ;Rotate Right Var1
            ror   var2              ;Rotate Right Var2
            brcs  Ph_Complete       ;Check Carry bit
            rjmp  Ph_Md2
Ph_Rec0:    clc                     ;Set Carry Bit =0
```

```
                ror     var1
                ror     var2
                brcs    Ph_Complete
Ph_Md0:         sbis    PIND,IR         ;Wait middle Bit Down
                rjmp    Ph_Md0
                rjmp    Ph_Measure

Ph_Complete:    rcall   delay
                rcall   delay
                ldi     temp,$01        ;Type Philips
                rcall   putc
                clr     temp            ;Find Address from var2
                bst     var2,5
                bld     temp,0          ;bit 0
                bst     var2,4
                bld     temp,1          ;bit 1
                bst     var2,3
                bld     temp,2          ;bit 2
                bst     var2,2
                bld     temp,3          ;bit 3
                bst     var2,1
                bld     temp,4          ;bit 4
                rcall   putc            ;Send Address to RS-232

                clr     temp            ;Find Command from var1
                bst     var2,7
                bld     temp,4
                bst     var2,6
                bld     temp,5
                bst     var1,0
                bld     temp,3
                bst     var1,1
                bld     temp,2
                bst     var1,2
                bld     temp,1
                bst     var1,3
                bld     temp,0
                rcall   putc            ;Send Command to Rs-232
                ret
```

```
;********** Sub Program Sony Receive ********************
Rec_Sony:   clc
            ldi     var1,$08
            ldi     var2,$00
                                        ;start of bit
Sony_Up:    sbis    PIND,IR
            rjmp    Sony_Up

Sony_Bit:   sbic    PIND,IR
            rjmp    Sony_Bit

            rcall   delay_800us

            sbis    PIND,IR
            rjmp    Sony_Rec1
            clc                         ;Receive bit 0
            ror     var1
            ror     var2
            brcs    Sony_Comp           ;Branc if carry flag set
            rjmp    Sony_Bit

Sony_Rec1:  sec                         ;Receive bit 1
            ror     var1
            ror     var2
            brcs    Sony_Comp
            rjmp    Sony_up

Sony_Comp:  rcall   delay
            rcall   delay
            ldi     temp,$02            ;Send to RS-232 Type Sony
            rcall   putc
            mov     temp,var1           ;compleat 12 bit
            lsr     temp                ;Logical Shift Right
            lsr     temp
            lsr     temp
            rcall   putc                ;Send Address

            mov     temp,var2
            lsr     temp
            lsr     temp
            lsr     temp
            lsr     temp
            bst     var1,0
            bld     temp,4
            bst     var1,1
```

```
                bld     temp,5
                bst     var1,2
                bld     temp,6
                rcall   putc                    ;Send Command

                ret

;***************** Sub Program JVC Receive ************************
Rec_JVC:        clc
                ldi     var1,0b10000000
                ldi     var2,$00

JVC_Up:                 sbis    PIND,IR
                rjmp    JVC_Up

J_Start:        sbic    PIND,IR                 ;start bit
                rjmp    J_Start

J_Mid:          sbis    PIND,IR
                rjmp    J_Mid

J_Measure:      rcall   delay_800us             ;measure level

                sbic    PIND,IR
                rjmp    J_Rec1
                clc                             ;receive bit 0
                ror     var1
                ror     var2
                brcs    J_Comp
                rjmp    J_Mid                   ;J_Measure

J_Rec1:         sec
                ror     var1
                ror     var2
                brcs    J_Comp
                rjmp    J_Start

J_Comp:                 rcall   delay           ;delay 110 ms (65+65)
                rcall   delay
                ldi     temp,$03                ;Type JVC
                rcall   putc
                mov     temp,var2               ;Send Address
                rcall   putc
                mov     temp,var1               ;Send Command
```

```
                rcall   putc
                ret

;************ Sub Program NEC Receive ********************
Rec_NEC:        clc
                ldi     var1,0b10000000
                ldi     var2,$00
                ldi     var3,$00
                ldi     var4,$00

NEC_StUp:       sbis    PIND,IR
                rjmp    NEC_StUp

NEC_Bit:        sbic    PIND,IR                 ;start bit
                rjmp    NEC_Bit

NEC_Measure:    rcall   delay_1200us            ;measure level

                sbic    PIND,IR
                rjmp    NEC_Rec1
                clc                             ;receive bit 0
                ror     var1
                ror     var2
                ror     var3
                ror     var4
                brcs    NEC_Comp
                rjmp    NEC_Measure

NEC_Rec1:       sec
                ror     var1
                ror     var2
                ror     var3
                ror     var4
                brcs    NEC_Comp
                rjmp    NEC_Bit

NEC_Comp:       rcall   delay                   ;delay 65 ms
                ldi     temp,$04
                rcall   putc
                mov     temp,var4               ;Address
                rcall   putc
                mov     temp,var2               ;Command
                rcall   putc
                ret
```

```
;******************** Interup Mode Capture Signel ************************
Capture:    clr     r27
            ldi     r26,$60
            clr     counter

Chk_Low:    rcall   delay_53us
            sbis    PIND,IR
            rjmp    Add_L
            rjmp    Store_L
Add_L:      inc     counter
            cpi     counter,170
            breq    Cap_End
            rjmp    Chk_Low

Store_L:    st      X+,counter
            clr     counter

Chk_Hi:     rcall   delay_53us
            sbic    PIND,IR
            rjmp    Add_H
            rjmp    Store_H
Add_H:      inc     counter
            cpi     counter,170
            breq    Cap_End
            rjmp    Chk_Hi

Store_H:    st      X+,counter
            clr     counter
            rjmp    Chk_Low

Cap_End:    st      X+,counter
            rcall   delay
            ldi     r26,$60
out_232:ld  temp,X+
            rcall   putc
            cpi     temp,170
            brne    out_232
            ret

;************* Send Data to RS- 232 ********************
putc:       sbi     UCR,TXEN
            sbis    USR,UDRE
            rjmp    putc
            out     UDR,temp
            cbi     UCR,TXEN
```

```
                ret
;********** Sampling Period 53us **********************
delay_53us:     ldi     temp,26
                mov     dlycnt1,temp            ;Delay 53.0 us
del_53:         dec     dlycnt1
                nop
                nop
                nop
                nop
                nop
                brne    del_53
                ret
;*****************************************************

delay_800us:    ldi     temp,502
                mov     dlycnt1,temp
dly_800us_1:    ldi     temp,SC8
                mov     dlycnt2,temp
loop2:          dec     dlycnt2                 ; 8*.25=2us
                nop
                nop
                nop
                nop
                nop
                brne    loop2
                dec     dlycnt1
                brne    dly_800us_1
                ret

;*********Delay 1100 us for mesual level******
delay1100us:    ldi     temp,11
                mov     dlycnt1,temp
delay1100us_1:  ldi     temp,70                 ;time 100us
                mov     dlycnt2,temp
delay1100us_2:  dec     dlycnt2
                nop
                brne    delay1100us_2
                dec     dlycnt1
                brne    delay1100us_1
                ret
```

```
;*********Delay 1200 us for mesual level******
delay_1200us:  ldi    temp,$03
               mov    dlycnt1,temp
delay_1200us_1:ldi    temp,$c8          ;time 400us
               mov    dlycnt2,temp
delay_1200us_2:dec    dlycnt2
               nop
               nop
               nop
               nop
               nop
               brne   delay_1200us_2
               dec    dlycnt1
               brne   delay_1200us_1
               ret

;*********Delay 1500 us for mesual level******
delay_1500us:  ldi    temp,$04
               mov    dlycnt1,temp
delay_1500us_1:ldi    temp,210          ;time 400us (150*2uSec)
               mov    dlycnt2,temp
delay_1500us_2:dec    dlycnt2
               nop
               nop
               nop
               nop
               nop
               brne   delay_1500us_2
               dec    dlycnt1
               brne   delay_1500us_1
               ret

;*****************Pluse T 's Sony ***********************
delay_480us:   ldi    temp,$02          ;0.25
               mov    dlycnt1,temp
d480us_1:      ldi    temp,120          ;0.25 (2*120=240us)
               mov    dlycnt2,temp
d480us_2:      dec    dlycnt2           ;0.25 (8*0.25=2us)
               nop
               nop
               nop
               nop
               nop
```

```
                brne    d480us_2                ;0.5 if true
                dec     dlycnt1
                brne    d480us_1
                ret
;******************** Header Sony (Send) ********************************
delay_2200us:   ldi     temp,11
                mov     dlycnt1,temp
d2200_1: ldi    temp,95                         ;do 100time
                mov     dlycnt2,temp
d2200_2: dec    dlycnt2
                nop
                nop
                nop
                nop
                nop
                brne    d2200_2                 ;0.5 if true
                dec     dlycnt1
                brne    d2200_1
                ret
;******************** Header NEC (Send)
*********************************
delay_88ms:     ldi     temp,21                 ;delay 8800 us
                mov     dlycnt1,temp
d88_1:          ldi     temp,200                ;do 200 time
                mov     dlycnt2,temp
d88_2:          dec     dlycnt2
                nop
                nop
                nop
                nop
                nop
                brne    d88_2                   ;0.5 if true
                dec     dlycnt1
                brne    d88_1
                ret
;******************** Header JVC (Send)
*********************************
delay_72ms:     ldi     temp,15                 ;delay 7200 us
                mov     dlycnt1,temp
d72_1:          ldi     temp,200                ;do 200 time
                mov     dlycnt2,temp
d72_2:          dec     dlycnt2
                nop
```

```
                nop
                nop
                nop
                nop
                brne    d72_2                   ;0.5 if true
                dec     dlycnt1
                brne    d72_1
                ret
;****************** Header JVC (Send)
*********************************
delay_33ms:     ldi     temp,8                  ;delay 3300 us
                mov     dlycnt1,temp
d33_1:          ldi     temp,200                ;do 200 time
                mov     dlycnt2,temp
d33_2:          dec     dlycnt2
                nop
                nop
                nop
                nop
                nop
                brne    d33_2                   ;0.5 if true
                dec     dlycnt1
                brne    d33_1
                ret
;****************** Header NEC (Send)
*********************************
delay_44ms:     ldi     temp,11                 ;delay 8800 us
                mov     dlycnt1,temp
d44_1:          ldi     temp,198                ;do 198 time
                mov     dlycnt2,temp
d44_2:          dec     dlycnt2
                nop
                nop
                nop
                nop
                nop
                brne    d44_2                   ;0.5 if true
                dec     dlycnt1
                brne    d44_1
                ret
;******************Pluse T 's NEC ***********************
delay_400us:    ldi     temp,$02        ;0.25
                mov     dlycnt1,temp
d400us_1:       ldi     temp,100                ;0.25 (2*100=200us)
```

```
                mov     dlycnt2,temp
d400us_2:       dec     dlycnt2             ;0.25 (8*0.25=2us)
                nop
                nop
                nop
                nop
                nop
                brne    d400us_2            ;0.5 if true
                dec     dlycnt1
                brne    d400us_1
                ret
;***********Delay 65 ms *****************
delay:          clr     dlycnt1
                ldi     temp,$FF
                mov     dlycnt2,temp
delay_11:       dec     dlycnt1
                nop
                nop
                nop
                nop
                nop
                brne    delay_11
                dec     dlycnt2
                brne    delay_11
                ret
```

Source Code of Microsoft Visual Basic 6.0

**********FrmMain**********

```
Option Explicit
    Dim PortNum As Integer

Private Sub MDIForm_Activate()
    PortNum = mintComPort
    sbr1.Panels(1).Text = "Port No : COM " & PortNum
End Sub

Private Sub MDIForm_Load()
    dlgCommon.Color = vbWhite
    mintComPort = 1
    PortNum = mintComPort
    sbr1.Panels(1).Text = "Port No : COM " & PortNum
End Sub

Private Sub mnuCheck_Click()
    Unload frmCapture
    Unload frmSendData
    frmReceive.Show
End Sub

Private Sub mnuModeCapture_Click()
    Unload frmReceive
    Unload frmSendData
    frmCapture.Show
    Me.WindowState = 2
End Sub

Private Sub mnuModeSend_Click()
    Unload frmReceive
    Unload frmCapture
    frmSendData.Show
End Sub
Private Sub mnuSetPort_Click()
    frmComPort.Show
End Sub
Private Sub munDisBG_Click()
    With dlgBack
        .Flags = cdlCCRGBInit
        .ShowColor
    End With
End Sub
Private Sub munDisLine_Click()
    With dlgCommon
```

```
            .Flags = cdlCCRGBInit
            .ShowColor
        End With
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)

    Select Case Button.Key

        Case "Capture"
                Unload frmReceive
                Unload frmSendData
                frmCapture.Show
                Me.WindowState = 2
        Case "Check"
                Unload frmCapture
                Unload frmSendData
                frmReceive.Show
        Case "Send"
                Unload frmReceive
                Unload frmCapture
                frmSendData.Show
        Case "Save"
                dlgCommon.ShowSave

    End Select
End Sub
```

**********FrmCapture**********

```
Option Explicit
Dim mintData(1 To 200) As Integer
Dim mblchkFF As Boolean
Dim mintIndex As Integer
Dim mintSum As Long
Dim mstrData As String
Private Sub cmdAdd_Click()

    On Error GoTo HandleAddErrors
     If cmdAdd.Caption = "&Add" Then
        adoCapture.Recordset.AddNew
        txtData.Text = mstrData
        cmdAdd.Caption = "&Cancel"
        cmdSave.Enabled = True
    Else
        adoCapture.Recordset.CancelUpdate
        cmdAdd.Caption = "&Add"
    End If
cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim strMessage As String
    strMessage = "การทำงานไม่สมบูรณ์" & vbCrLf & vbCrLf _
                & Err.Description
    MsgBox strMessage, vbExclamation, "Database Error"
    Resume cmdAdd_Click_Exit
End Sub
Private Sub cmdClear_Click()
    picShow.Cls
    txtShow.Text = ""
End Sub
Private Sub cmdDelete_Click()
    On Error GoTo HandleDeleteError
    With adoCapture.Recordset
    If MsgBox("คุณแน่ใจว่าต้องการลบข้อมูลนี้", vbYesNo, "คำเตือน") = vbYes Then
        .Delete
        .MoveNext
        If .EOF Then
            .MovePrevious
            If .BOF Then
                MsgBox "ไม่มีข้อมูลภายในฐานข้อมูล", vbInformation, "ไม่มีข้อมูล"
            End If
        End If
    End If
    End With
cmdDelete_Click_Exit:
```

```
        Exit Sub

HandleDeleteError:
        Dim strMessage As String
        strMessage = "การทำงานไม่สมบูรณ์" & vbCrLf & vbCrLf _
                        & Err.Description
        MsgBox strMessage, vbExclamation, "Database Error"
        On Error GoTo 0
End Sub

Private Sub cmdSave_Click()

        On Error GoTo HandleSaveErrors
        If txtConsumer.Text = "" Or txtButton.Text = "" Then
            MsgBox "ไม่มีข้อมูลที่จะบันทึก โปรดกรอกข้อมูลให้ครบถ้วน", vbOKOnly, "ร้องขอข้อมูล"
        Else
            txtConsumer.Text = UCase(txtConsumer.Text)
            txtButton.Text = UCase(txtButton.Text)
            adoCapture.Recordset.Update
            cmdAdd.Caption = "&Add"
            cmdAdd.Enabled = False
            cmdSave.Enabled = False
        End If

cmdSave_Click_Exit:
        Exit Sub

HandleSaveErrors:
        Dim strMessage As String
        strMessage = "ไม่สามารถทำการบันทึกข้อมูลได้" & vbCrLf & vbCrLf _
                        & Err.Description
        MsgBox strMessage, vbExclamation, "Database Error"
        Resume cmdSave_Click_Exit
End Sub

Private Sub cmdShow_Click()
        Dim intLen As Integer
        Dim strSig As String
        Dim intByte As Integer
        Dim strChar As String
        Dim strData(1 To 200) As String
        Dim a As Integer

        MSComm1.PortOpen = False
        txtShow.Text = ""
        intLen = Len(txtData.Text)
        For a = 1 To intLen
            strChar = Mid(txtData.Text, a, 1)
            If strChar <> "," Then
                intByte = Val(intByte & strChar)
```

```
            Else
                mintIndex = mintIndex + 1
                mintData(mintIndex) = intByte
                mintSum = intByte + mintSum

            End If

        Next a
        Call ShowData(0)
        mintSum = 0
        mintIndex = 0

        cmdAdd.Enabled = False
End Sub

Private Sub Form_Load()
        Dim PortNum As Integer
        PortNum = mintComPort
        MSComm1.CommPort = PortNum
        MSComm1.PortOpen = True
        MSComm1.Settings = "9600,N,8,1"
        MSComm1.RThreshold = 1
        MSComm1.InputLen = 1
        MSComm1.InputMode = comInputModeText
        MSComm1.Handshaking = comNone
        MSComm1.Output = "z"    'Send Number Mode Capture (Hex:7A)
        mintIndex = 0
        mintSum = 0
        cmdSave.Enabled = False
        cmdAdd.Enabled = False
        picShow.Cls
        picShow.Scale (-100, -7)-(70000, 20)
        picShow.BackColor = frmMain.dlgBack.Color
        picShow.ForeColor = frmMain.dlgCommon.Color
        picShow.DrawWidth = 2
End Sub

Private Sub Form_Unload(Cancel As Integer)
    If MSComm1.PortOpen = True Then
            MSComm1.PortOpen = False
        End If
End Sub

Private Sub MSComm1_OnComm()
        Dim Buffer As String
        Dim intGetstr As Integer
       Select Case MSComm1.CommEvent
            Case comEvReceive
                Buffer = MSComm1.Input
                intGetstr = Asc(Buffer)
```

```
                If intGetstr = 0 Then
                    mblchkFF = True
                End If

                If mblchkFF = True Then
                    MSComm1.PortOpen = False
                    Call ShowData(0)
                Else
                    mintIndex = mintIndex + 1
                    mintData(mintIndex) = intGetstr
                    mintSum = intGetstr + mintSum
                End If
        End Select
End Sub

Private Sub ShowData(index As Integer)
        Dim i As Integer
        Dim j As Long
        Dim X As Long
        Dim Pulse As Currency, Xp As Currency
        Dim ScaleX As Long
        picShow.Cls
        picShow.BackColor = frmMain.dlgBack.Color
        picShow.ForeColor = vbWhite
        ScaleX = mintSum * 53                          '53 uSec/Count
        picShow.Scale (-200, -7)-(ScaleX, 20)
        picShow.Line (0, 16)-(ScaleX - 3000, 16)       'Set Sceal AxisX
        picShow.Line (0, 2)-(0, 16)                    'Set Sceal Axis Y
       txtShow.Text = ""

    ' /*******      Set Scale      *******/
        For j = 1 To (ScaleX / 2000) - 2
            X = j * 2000
            picShow.Line (X, 16)-(X, 17)
            picShow.CurrentX = X - 500
            picShow.Print X / 1000
        Next j
        picShow.CurrentX = ScaleX - 2800
        picShow.CurrentY = 15
        picShow.Print "Time (ms)"
        i = 1
        j = 0
        picShow.ForeColor = frmMain.dlgCommon.Color
        Do Until i > mintIndex - 1
            If chk53.Value = Checked Then
                j = j + 1
                If j = 10 Then
                    txtShow.Text = txtShow.Text & i & ")  " & mintData(i) * 53 &
vbCrLf
                    j = 0
```

```
                Else
                    txtShow.Text = txtShow.Text & i & ")  " & mintData(i) * 53 &
vbTab
                End If
            Else
                j = j + 1
                If j = 10 Then
                    txtShow.Text = txtShow.Text & i & ")  " & mintData(i) & vbCrLf
                    j = 0
                Else
                    txtShow.Text = txtShow.Text & i & ")  " & mintData(i) & vbTab
                End If
            End If

            ' /***** Plot Signal     ******/
            If i Mod 2 = 1 Then
                Pulse = (mintData(i) * 53) + Xp
                picShow.Line (Xp, 10)-(Pulse, 10)
                picShow.Line (Pulse, 10)-(Pulse, 15)
                Xp = Pulse
            Else
                Pulse = (mintData(i) * 53) + Xp
                picShow.Line (Xp, 15)-(Pulse, 15)
                picShow.Line (Pulse, 10)-(Pulse, 15)
                Xp = Pulse
            End If
            i = i + 1
        Loop
        picShow.Line (Xp, 15)-(Xp + 2000, 15)

        ' /******** get data from mintData() *******/
        mstrData = mintData(1)
        i = 2
        Do Until i > mintIndex - 1
            mstrData = mstrData & "," & mintData(i)
            i = i + 1
        Loop
            '  txtData.Text = mstrData

        mintSum = 0
        mintIndex = 0
        mblchkFF = False
        cmdAdd.Enabled = True
        MSComm1.PortOpen = True
End Sub

Private Sub Frame1_DragDrop(Source As Control, X As Single, Y As Single)
End Sub
```

**********FrmReceive**********

```
Option Explicit
Dim IndexLst As Integer
Dim mintData(1 To 10) As String
Dim mblchkFF    As Boolean, mblchk19 As Boolean
Dim mintIndex     As Integer, mintCurrentX As Currency, mintNextX As Currency
Dim mstrData     As String, mstrBCD As String
Dim mstrConsumer      As String
Dim mstrAddress       As String
Dim mstrCommand      As String
Private Sub cmdAdd_Click()
    On Error GoTo HandleAddErrors
    If cmdAdd.Caption = "&Add" Then
        adoRemote.Recordset.AddNew
        EnableButtons
        cmdAdd.Caption = "&Cancel"
         If mstrConsumer = "01" And mstrAddress <> "00" And mstrCommand <>
"00" Then
            txtConsumer.Text = "Philips"
            lblCode.Caption = "01" & mstrAddress & mstrCommand
        ElseIf mstrConsumer = "02" And mstrAddress <> "00" And mstrCommand
<> "00" Then
            txtConsumer.Text = "Sony"
            lblCode.Caption = "02" & mstrAddress & mstrCommand
        ElseIf mstrConsumer = "03" And mstrAddress <> "00" And mstrCommand
<> "00" Then
            txtConsumer.Text = "JVC"
            lblCode.Caption = "03" & mstrAddress & mstrCommand

        ElseIf mstrConsumer = "04" And mstrAddress <> "00" And mstrCommand
<> "00" Then
            txtConsumer.Text = "NEC"
            lblCode.Caption = "04" & mstrAddress & mstrCommand
        End If

    Else
        cmdAdd.Caption = "&Add"
        adoRemote.Recordset.CancelUpdate
    End If

cmdAdd_Click_Exit:
    Exit Sub

HandleAddErrors:
    Dim strMessage As String
    strMessage = "การทำงานไม่สมบูรณ์" & vbCrLf & vbCrLf _
```

```
                    & Err.Description
    MsgBox strMessage, vbExclamation, "Database Error"
    cmdAdd.Caption = "&Add"
    Resume cmdAdd_Click_Exit
End Sub
Private Sub cmdDelete_Click()
    'Delete the current record
    On Error GoTo HandleDeleteError
    With adoRemote.Recordset
    If MsgBox("คุณแน่ใจว่าต้องการลบข้อมูลนี้", vbYesNo, "คำเตือน") = vbYes Then
        .Delete
        .MoveNext
        If .EOF Then
            .MovePrevious
            If .BOF Then
                MsgBox "ไม่มีข้อมูลภายในฐานข้อมูล", vbInformation, "ไม่พบข้อมูล"
            End If
        End If
    End If
    End With
cmdDelete_Click_Exit:
    Exit Sub

HandleDeleteError:
    Dim strMessage As String
    strMessage = "การทำงานไม่สมบูรณ์" & vbCrLf & vbCrLf _
                & Err.Description
    MsgBox strMessage, vbExclamation, "Database Error"
    On Error GoTo 0

End Sub
Private Sub cmdSave_Click()
    On Error GoTo HandleSaveErrors
    If txtDevice.Text = "" Or txtFunction.Text = "" Then
        MsgBox "ไม่มีข้อมูลที่จะบันทึก โปรดกรอกข้อมูลให้ครบถ้วน", vbOKOnly, "ร้องขอข้อมูล"
    Else
        txtDevice.Text = UCase(txtDevice.Text)
        txtFunction.Text = UCase(txtFunction.Text)
        adoRemote.Recordset.Update
        cmdAdd.Caption = "&Add"

        DisableButtons
    End If
cmdSave_Click_Exit:
    Exit Sub

HandleSaveErrors:
    Dim strMessage
    strMessage = "เรคคอร์ดไม่สามารถทำการบันทึกข้อมูลได้" & vbCrLf _
```

```
                & Err.Description      'Display system error message
        MsgBox strMessage, vbExclamation, "ฐานข้อมูลผิดพลาด"
        SetUpAddRecord          'Allow user to change Code and complete Add
        Resume cmdSave_Click_Exit

End Sub

Private Sub Form_Load()
    adoRemote.Refresh
    Static PortNum As Integer
     frmMain.sbr1.Panels(3).Text = "Data :"
    PortNum = mintComPort
    frmReceive.MSComm1.CommPort = PortNum
    frmReceive.MSComm1.PortOpen = True
    frmReceive.MSComm1.Settings = "9600,N,8,1"
    MSComm1.RThreshold = 1
    MSComm1.InputLen = 1
    MSComm1.InputMode = comInputModeText
    MSComm1.Handshaking = comNone
    picShow.Scale (-1, -3)-(100, 20)
    picShow.BackColor = frmMain.dlgBack.Color
    picShow.ForeColor = frmMain.dlgCommon.Color
    picShow.DrawWidth = 2
    MSComm1.Output = "Z"  'Send Number  Select Mode  Receive : 5A

End Sub

Private Sub Form_Unload(Cancel As Integer)
    If MSComm1.PortOpen = True Then
        MSComm1.PortOpen = False
    End If
End Sub

Private Sub Frame2_DragDrop(Source As Control, X As Single, Y As Single)

End Sub

Private Sub MSComm1_OnComm()
    Dim Buffer As String
    Dim intGetstr As Integer
    On Error Resume Next
    Select Case MSComm1.CommEvent

        Case comEvReceive
            Buffer = MSComm1.Input
            intGetstr = Asc(Buffer)
            If intGetstr = 255 Then
                mblchkFF = True
            ElseIf intGetstr = 19 And mblchkFF = True Then
                mblchk19 = True
```

```
                End If
                If mblchkFF = True Then
                    If mblchk19 = True Then                    'End Frame data from
remote control
                        cmdAdd.Enabled = False
                        cmdSave.Enabled = False
                        Call ShowData(0)
                    End If
                Else
                    mintIndex = mintIndex + 1
                    mintData(mintIndex) = intGetstr
                End If
            End Select
End Sub

Private Sub ShowData(index As Integer)

    Dim HexNum As String
    Dim blnNot    As Boolean
    Dim Com As String
    Dim Fun As String

     If Len(mintData(1)) = 1 Then
        mstrConsumer = "0" & Hex(mintData(1))
     Else
        mstrConsumer = Hex(mintData(1))
     End If

     If Len(mintData(2)) = 1 Then
        mstrAddress = "0" & Hex(mintData(2))
     Else
        mstrAddress = Hex(mintData(2))
     End If

     If Len(mintData(3)) = 1 Then
        mstrCommand = "0" & Hex(mintData(3))
     Else
        mstrCommand = Hex(mintData(3))
     End If
     mstrData = mstrConsumer & mstrAddress & mstrCommand
     lblAddress.Caption = mstrAddress
     lblCommand.Caption = mstrCommand
     ' /******      Find Data  in Database    ******/
     With adoRemote.Recordset
        .MoveFirst
        .Find "Code = '" & mstrData & "' "
       Call ShowSignal
        ' /*******     Not Found Data        ******/
        If .EOF Then
```

```
                If (mstrData = "010000") Or (mstrData = "020000") Or (mstrData
= "030000") Or (mstrData = "040000") Then
                    Unload Me
                    Load frmCapture
                    frmMain.WindowState = 2
                Else
                    Call ShowSignal
                    cmdAdd.Enabled = True
                    cmdSave.Enabled = True
                End If
            End If
        End With
        HexNum = mstrAddress & mstrCommand
        mstrBCD = BCD(HexNum)
        frmMain.sbr1.Panels(3).Text = "Data : " & mstrBCD
        mintIndex = index
        mblchkFF = False
        mblchk19 = False
End Sub

Private Sub ShowSignal()
    Dim Digi As String, InvAdd As String, InvCom As String, InvDigi As String
    Dim i  As Integer, n  As Integer, X As Integer
    picShow.Cls
    picShow.BackColor = frmMain.dlgBack.Color
    picShow.ForeColor = vbWhite
    picShow.Line (1, 17)-(94, 17)       'Set Sceal AxisX
    picShow.Line (1, 5)-(1, 17)         'Set Sceal Axis Y
    picShow.CurrentX = 94
    picShow.CurrentY = 16
    picShow.Print "Time(ms)"
    For n = 1 To 18
        X = n * 5
        picShow.Line (X, 17)-(X, 18)
        picShow.CurrentX = X - 1
        picShow.Print X
    Next n

    picShow.ForeColor = frmMain.dlgCommon.Color
     If mstrConsumer = "01" Then
        mintCurrentX = 13
        picShow.Line (1, 11)-(10, 11)
        picShow.Line (10, 11)-(10, 16)
        picShow.Line (10, 16)-(13, 16)
        picShow.Line (13, 11)-(13, 16)

    ElseIf mstrConsumer = "02" Then
        mintCurrentX = 13
        picShow.Line (1, 11)-(10, 11)
        picShow.Line (10, 11)-(10, 16)
```

```
        picShow.Line (10, 16)-(13, 16)
        picShow.Line (13, 11)-(13, 16)

        For i = 16 To 10 Step -1
            Digi = Mid(mstrBCD, i, 1)
            If Digi = "1" Then
                Call SonyData1
            ElseIf Digi = "0" Then
                Call SonyData0
            End If
        Next i
        For i = 8 To 4 Step -1
            Digi = Mid(mstrBCD, i, 1)
            If Digi = "1" Then
                Call SonyData1
            Else
                Call SonyData0
            End If
        Next i
        picShow.Line (mintCurrentX, 11)-(mintCurrentX + 3, 11)
    ElseIf mstrConsumer = "03" Then
        mintCurrentX = 13
        picShow.Line (1, 11)-(10, 11)
        picShow.Line (10, 11)-(10, 16)
        picShow.Line (10, 16)-(13, 16)
        picShow.Line (13, 11)-(13, 16)
        For i = 8 To 1 Step -1
            Digi = Mid(mstrBCD, i, 1)
            If Digi = "1" Then
                Call JVCData1
            Else
                Call JVCData0
            End If
        Next i
        For i = 16 To 9 Step -1
            Digi = Mid(mstrBCD, i, 1)
            If Digi = "1" Then
                Call JVCData1
            Else
                Call JVCData0
            End If
        Next i
    ElseIf mstrConsumer = "04" Then
        mintCurrentX = 13
        picShow.Line (1, 11)-(11, 11)
        picShow.Line (11, 11)-(11, 16)
        picShow.Line (11, 16)-(13, 16)
        picShow.Line (13, 11)-(13, 16)
        For i = 1 To 8
            Digi = Mid(mstrBCD, i, 1)
```

```
                If Digi = "1" Then
                    Call NECData1
                Else
                    Call NECData0
                End If
            Next i
            For i = 1 To 8
                Digi = Mid(mstrBCD, i, 1)
                If Digi = "1" Then
                    Call NECData0
                Else
                    Call NECData1
                End If
            Next i

            For i = 9 To 16
                Digi = Mid(mstrBCD, i, 1)
                If Digi = "1" Then
                    Call NECData1
                Else
                    Call NECData0
                End If
            Next i
            For i = 9 To 16
                Digi = Mid(mstrBCD, i, 1)
                If Digi = "1" Then
                    Call NECData0
                Else
                    Call NECData1
                End If
            Next i
        End If
End Sub
Private Sub SonyData1()
    mintNextX = mintCurrentX + 4
    picShow.Line (mintCurrentX, 11)-(mintNextX, 11)      ' Up Level
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
    mintNextX = mintCurrentX + 2
    picShow.Line (mintCurrentX, 16)-(mintNextX, 16)
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
     mintCurrentX = mintNextX
End Sub

Private Sub SonyData0()
    mintNextX = mintCurrentX + 2
    picShow.Line (mintCurrentX, 11)-(mintNextX, 11)      ' Up Level
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
    mintNextX = mintCurrentX + 2
```

```
        picShow.Line (mintCurrentX, 16)-(mintNextX, 16)
        picShow.Line (mintNextX, 11)-(mintNextX, 16)
        mintCurrentX = mintNextX
End Sub

Private Function BCD(Base As String) As String
        Dim Num(10) As String
        Dim Leng As Integer, i As Integer
        Dim Str(10) As String
        i = 1
        Leng = Len(Base)
        Do Until i > Leng
                Str(i) = Mid(Base, i, 1)
                i = i + 1
        Loop
        For i = 1 To Leng
                Select Case Str(i)
                Case "0"
                        Num(i) = "0000"
                Case "1"
                        Num(i) = "0001"
                Case "2"
                        Num(i) = "0010"
                Case "3"
                        Num(i) = "0011"
                Case "4"
                        Num(i) = "0100"
                Case "5"
                        Num(i) = "0101"
                Case "6"
                        Num(i) = "0110"
                Case "7"
                        Num(i) = "0111"
                Case "8"
                        Num(i) = "1000"
                Case "9"
                        Num(i) = "1001"
                Case "A"
                        Num(i) = "1010"
                Case "B"
                        Num(i) = "1011"
                Case "C"
                        Num(i) = "1100"
                Case "D"
                        Num(i) = "1101"
                Case "E"
                        Num(i) = "1110"
                Case "F"
                        Num(i) = "1111"
```

```
            End Select
        Next i

        If Leng = 1 Then
            BCD = Num(1)
        ElseIf Leng = 2 Then
            BCD = Num(1) & Num(2)
        ElseIf Leng = 3 Then
            BCD = Num(1) & Num(2) & Num(3)
        ElseIf Leng = 4 Then
            BCD = Num(1) & Num(2) & Num(3) & Num(4)
        End If

End Function

Private Sub EnableButtons()

        txtConsumer.Enabled = True
        txtFunction.Enabled = True
        txtDevice.Enabled = True
        cmdSave.Enabled = True
        cmdAdd.Enabled = True
End Sub

Private Sub DisableButtons()
        txtConsumer.Enabled = False
        txtFunction.Enabled = False
        txtDevice.Enabled = False
        cmdSave.Enabled = False
        cmdAdd.Enabled = False
End Sub

Private Sub SetUpAddRecord()
        Dim Device As String
        Dim Key As String
        Dim Consumer As String

        On Error Resume Next
        Consumer = txtConsumer.Text
        Device = txtDevice.Text
        Key = txtFunction.Text

        'Start a new Add
        adoRemote.Recordset.AddNew

        'Place saved data back in form controls
        txtConsumer.Text = Consumer
        txtFunction.Text = Key
        txtDevice.Text = Device
```

```
End Sub
Private Sub NECData1()
    mintNextX = mintCurrentX + 1
    picShow.Line (mintCurrentX, 11)-(mintNextX, 11)      ' Up Level
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
    mintNextX = mintCurrentX + 2
    picShow.Line (mintCurrentX, 16)-(mintNextX, 16)

    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
End Sub

Private Sub NECData0()
    mintNextX = mintCurrentX + 1
    picShow.Line (mintCurrentX, 11)-(mintNextX, 11)      ' Up Level
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
    mintNextX = mintCurrentX + 1
    picShow.Line (mintCurrentX, 16)-(mintNextX, 16)
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
End Sub

Private Sub JVCData1()
    mintNextX = mintCurrentX + 2
    picShow.Line (mintCurrentX, 11)-(mintNextX, 11)      ' Up Level
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
    mintNextX = mintCurrentX + 4
    picShow.Line (mintCurrentX, 16)-(mintNextX, 16)

    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
End Sub

Private Sub JVCData0()
    mintNextX = mintCurrentX + 2
    picShow.Line (mintCurrentX, 11)-(mintNextX, 11)      ' Up Level
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
    mintNextX = mintCurrentX + 2
    picShow.Line (mintCurrentX, 16)-(mintNextX, 16)
    picShow.Line (mintNextX, 11)-(mintNextX, 16)
    mintCurrentX = mintNextX
End Sub
```

**********FrmComport**********

```
Option Explicit

Private Sub cmdPortCancel_Click()
    Unload Me
End Sub

Private Sub cmdPortOk_Click()
    Dim strPort As String

        strPort = cboPort.List(cboPort.ListIndex)
        If frmReceive.MSComm1.PortOpen = True Then
            frmReceive.MSComm1.PortOpen = False
        ElseIf frmCapture.MSComm1.PortOpen = True Then
            frmCapture.MSComm1.PortOpen = False
        Else: frmSendData.MSComm1.PortOpen = True
            frmSendData.MSComm1.PortOpen = False
        End If
            Select Case strPort
                Case "COM 1"
                    mintComPort = 1
                Case "COM 2"
                    mintComPort = 2
                Case "COM 3"
                    mintComPort = 3
                Case "COM 4"
                    mintComPort = 4
                End Select
        Print mintComPort
        frmMain.sbr1.Panels(1).Text = "Port No : COM " & mintComPort
        Unload Me

End Sub

Private Sub Form_Load()

End Sub
```

**********FrmSendData**********

```
Option Explicit
Dim mVarSend As String
Dim mButton As String
Private Sub Search()
'*************************************************************************
*****************************
'Function search code from database ,when user select consumer and equipment   *
'*************************************************************************
*****************************
    Dim strDevice As String
    Dim strConsumer As String
    Dim Sqlcmd     As String
    Dim intCount  As Integer
    strConsumer = cboConsumer.Text
    strDevice = cboDevice.Text
    MSComm1.Output = "X"                    'send code to AVR change mode Send

    'send data to RS-232
    For intCount = 1 To Len(lblOut.Caption)
        mVarSend = Mid(lblOut.Caption, intCount, 1)
        MSComm1.Output = mVarSend
    Next intCount

    adoRemote.Refresh
    On Error GoTo HandleSearch
    With adoRemote.Recordset
        .MoveFirst
        .Filter = "Con_name = '" & strConsumer & "' and Device = '" & strDevice
& "' and Function  = '" & mButton & "'"
        If .EOF Then
            MsgBox "ไม่มีข้อมูลที่ตรงกับที่ท่านเลือก" & vbCrLf & "ท่านได้ลืมยี่ห้อรีโมทและอุปกรณ์ที่
ต้องการควบคุมแล้วหรือยัง", vbCritical, "ไม่ปรากฏข้อมูล"
        Else
            frmMain.sbr1.Panels(3).Text = "Data to Send : " & lblOut.Caption
            Call Delay
        End If
    End With

search_Exit:
    Exit Sub

HandleSearch:
    Dim strMessage  As String
    strMessage = "การทำงานไม่สมบูรณ์" & vbCrLf & vbCrLf _
                & Err.Description
```

```
        MsgBox strMessage, vbExclamation, "Database Error"
        On Error GoTo 0
    End Sub
    Private Sub cmdSend_Click()
        Dim intCount  As Integer
        Dim i As Integer
        MSComm1.Output = "X"
        For intCount = 1 To Len(txtInput.Text)
            mVarSend = Mid(txtInput.Text, intCount, 1)
            MSComm1.Output = mVarSend
        Next intCount
        frmMain.sbr1.Panels(3).Text = "Data to Send : " & txtInput.Text
        Do
            imgSignal.Visible = True
            hpCounter.Delay_ms (400)
            imgSignal.Visible = False
            hpCounter.Delay_ms (300)
            imgSignal.Visible = True
            hpCounter.Delay_ms (400)
            imgSignal.Visible = False
            hpCounter.Delay_ms (300)
            i = i + 1
        Loop While i > 500
    End Sub

    Private Sub dbgRemote_Click()

    End Sub

    Private Sub Form_Load()
        Dim PortNum As Integer
        PortNum = mintComPort
        MSComm1.CommPort = mintComPort
        MSComm1.PortOpen = True
        MSComm1.Settings = "9600,N,8,1"
        MSComm1.RThreshold = 1
        MSComm1.InputLen = 1
        MSComm1.InputMode = comInputModeText
        MSComm1.Handshaking = comNone
        txtInput.Text = ""
    End Sub
    Private Sub Form_Unload(Cancel As Integer)
        If MSComm1.PortOpen = True Then
            MSComm1.PortOpen = False
        End If
    End Sub
    Private Sub img0_Click()
        mButton = "Num0"
        Call Search
    End Sub
```

```
Private Sub img1_Click()
    mButton = "Num1"
    Call Search
End Sub
Private Sub img2_Click()
    mButton = "Num2"
    adoRemote.Refresh
    Call Search
End Sub
Private Sub img3_Click()
    mButton = "Num3"
    Call Search
End Sub
Private Sub img4_Click()
    mButton = "Num4"
    Call Search
End Sub
Private Sub img5_Click()
    mButton = "Num5"
    Call Search
End Sub
Private Sub img6_Click()
    mButton = "Num6"
    Call Search
End Sub
Private Sub img7_Click()
    mButton = "Num7"
    Call Search
End Sub
Private Sub img8_Click()
    mButton = "Num8"
    Call Search
End Sub
Private Sub img9_Click()
    mButton = "Num9"
    Call Search
End Sub
Private Sub imgCHDec_Click()
    mButton = "Ch-"
    Call Search
End Sub
Private Sub imgCHInc_Click()
    mButton = "Ch+"
    Call Search
End Sub
Private Sub imgVolDec_Click()
    mButton = "Vol-"
    Call Search
End Sub
Private Sub imgVolInc_Click()
```

```
        mButton = "Vol+"
        Call Search
End Sub
Private Sub MSComm1_OnComm()
        Dim Buffer As String
                Select Case MSComm1.CommEvent
                Case comEvReceive
                Case comEvSend
                End Select
End Sub
Private Sub Delay()
        Dim i As Integer
        Do
                imgSignal.Visible = True
                hpCounter.Delay_ms (300)
                imgSignal.Visible = False
                hpCounter.Delay_ms (300)
                imgSignal.Visible = True
                hpCounter.Delay_ms (400)
                imgSignal.Visible = False
                hpCounter.Delay_ms (300)
                i = i + 1
        Loop While i > 500
End Sub
```

ภาคผนวก ค

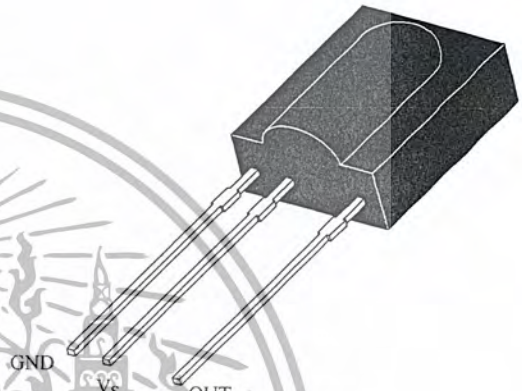
Photo Modules for PCM Remote Control Systems

Available types for different carrier frequencies

Type	fo	Type	fo
TSOP1730	30 kHz	TSOP1733	33 kHz
TSOP1736	36 kHz	TSOP1737	36.7 kHz
TSOP1738	38 kHz	TSOP1740	40 kHz
TSOP1756	56 kHz		

Description

The TSOP17.. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.
The demodulated output signal can directly be decoded by a microprocessor. TSOP17.. is the standard IR remote control receiver series, supporting all major transmission codes.

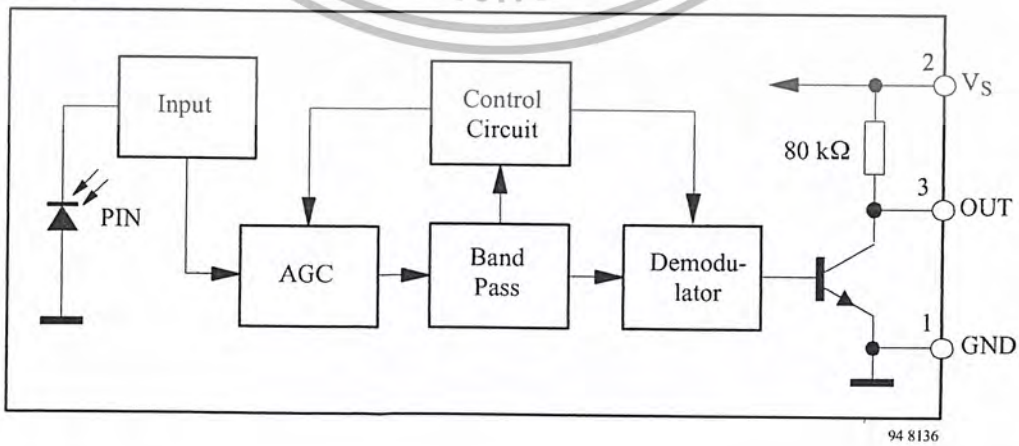


94 8691

Features

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Improved shielding against electrical field disturbance
- TTL and CMOS compatibility
- Output active low
- Low power consumption
- High immunity against ambient light
- Continuous data transmission possible (up to 2400 bps)
- Suitable burst length ≥10 cycles/burst

Block Diagram



94 8136

Absolute Maximum Ratings

T_{amb} = 25°C

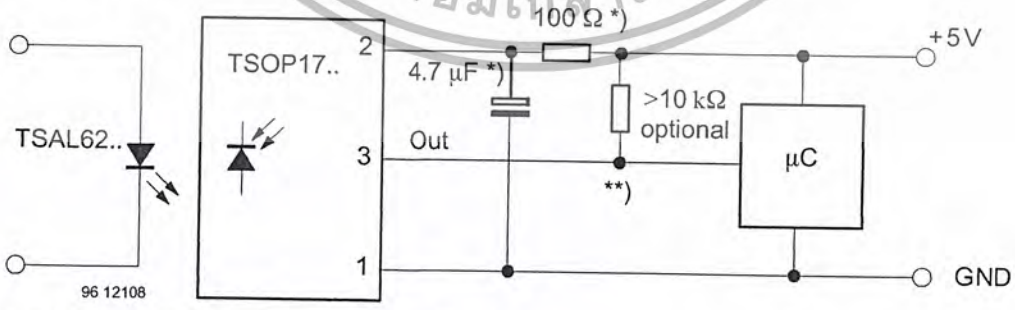
Parameter	Test Conditions	Symbol	Value	Unit
Supply Voltage	(Pin 2)	V_S	–0.3...6.0	V
Supply Current	(Pin 2)	I_S	5	mA
Output Voltage	(Pin 3)	V_O	–0.3...6.0	V
Output Current	(Pin 3)	I_O	5	mA
Junction Temperature		T_j	100	°C
Storage Temperature Range		T_{stg}	–25...+85	°C
Operating Temperature Range		T_{amb}	–25...+85	°C
Power Consumption	(T_{amb} ≦ 85 °C)	P_{tot}	50	mW
Soldering Temperature	t ≦ 10 s, 1 mm from case	T_{sd}	260	°C

Basic Characteristics

T_{amb} = 25°C

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Current (Pin 2)	V_S = 5 V, E_v = 0	I_{SD}	0.4	0.6	1.5	mA
	V_S = 5 V, E_v = 40 klx, sunlight	I_{SH}		1.0		mA
Supply Voltage (Pin 2)		V_S	4.5		5.5	V
Transmission Distance	E_v = 0, test signal see fig.7, IR diode TSAL6200, I_F = 400 mA	d		35		m
Output Voltage Low (Pin 3)	I_{OSL} = 0.5 mA, E_e = 0.7 mW/m², f = f_o, t_p/T = 0.4	V_{OSL}			250	mV
Irradiance (30 – 40 kHz)	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal (see fig.7)	$E_{e\,min}$		0.35	0.5	mW/m²
Irradiance (56 kHz)	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal (see fig.7)	$E_{e\,min}$		0.4	0.6	mW/m²
Irradiance	$t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$	$E_{e\,max}$	30			W/m²
Directivity	Angle of half transmission distance	$\varphi_{1/2}$		±45		deg

Application Circuit



*) recommended to suppress power supply disturbances

**) The output voltage should not be hold continuously at a voltage below 3.3V by the external circuit.

Suitable Data Format

The circuit of the TSOP17.. is designed in that way that unexpected output pulses due to noise or disturbance signals are avoided. A bandpassfilter, an integrator stage and an automatic gain control are used to suppress such disturbances.
The distinguishing mark between data signal and disturbance signal are carrier frequency, burst length and duty cycle.

The data signal should fullfill the following condition:

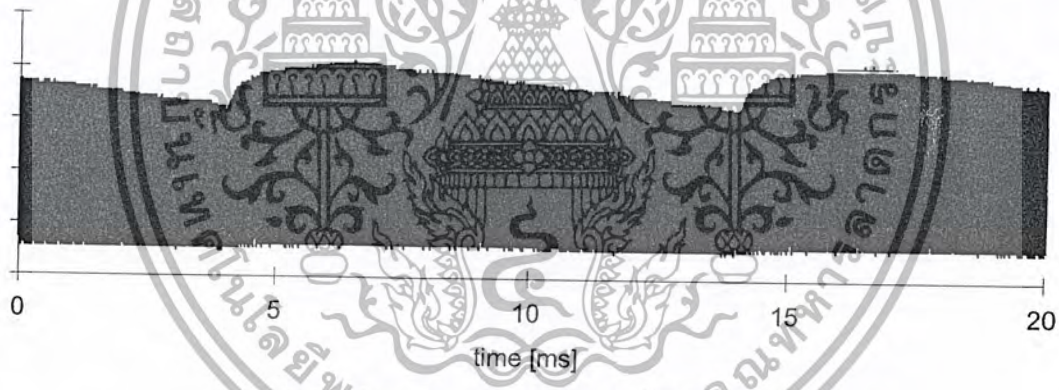
- Carrier frequency should be close to center frequency of the bandpass (e.g. 38kHz).
- Burst length should be 10 cycles/burst or longer.
- After each burst which is between 10 cycles and 70 cycles a gap time of at least 14 cycles is neccessary.
- For each burst which is longer than 1.8ms a corresponding gap time is necessary at some time in the data stream. This gap time should have at least same length as the burst.
- Up to 1400 short bursts per second can be received continuously.

Some examples for suitable data format are:
NEC Code, Toshiba Micom Format, Sharp Code, RC5 Code, RC6 Code, R–2000 Code, Sony Format (SIRCS).

When a disturbance signal is applied to the TSOP17.. it can still receive the data signal. However the sensitivity is reduced to that level that no unexpected pulses will occure.

Some examples for such disturbance signals which are suppressed by the TSOP17.. are:

- DC light (e.g. from tungsten bulb or sunlight)
- Continuous signal at 38kHz or at any other frequency
- Signals from fluorescent lamps with electronic ballast (an example of the signal modulation is in the figure below).



IR Signal from Fluorescent Lamp with low Modulation

Typical Characteristics (T_{amb} = 25°C unless otherwise specified)

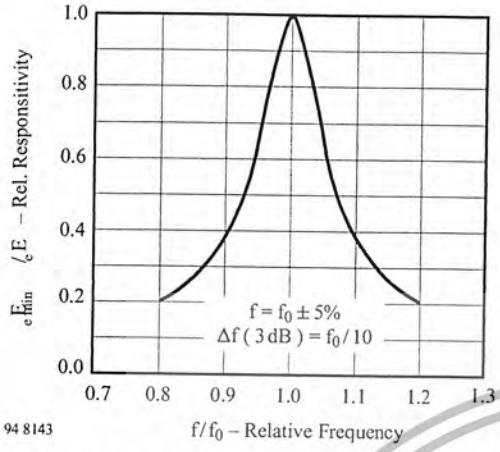


Figure 1. Frequency Dependence of Responsivity

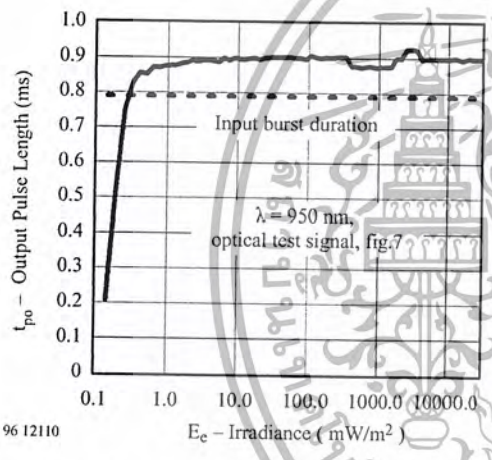


Figure 2. Sensitivity in Dark Ambient

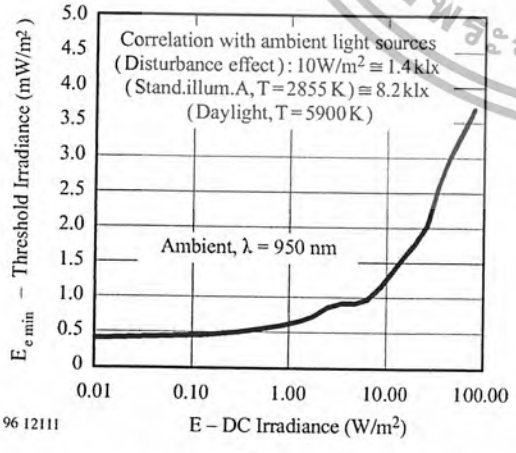


Figure 3. Sensitivity in Bright Ambient

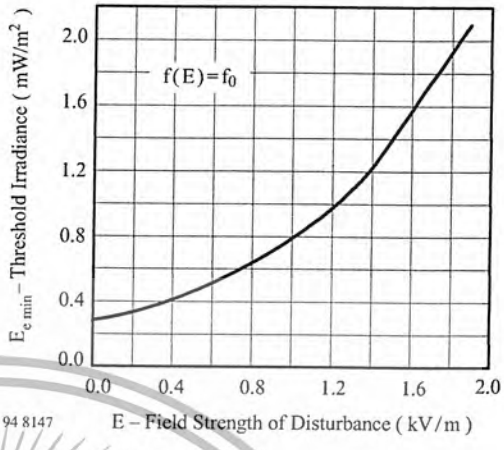


Figure 4. Sensitivity vs. Electric Field Disturbances

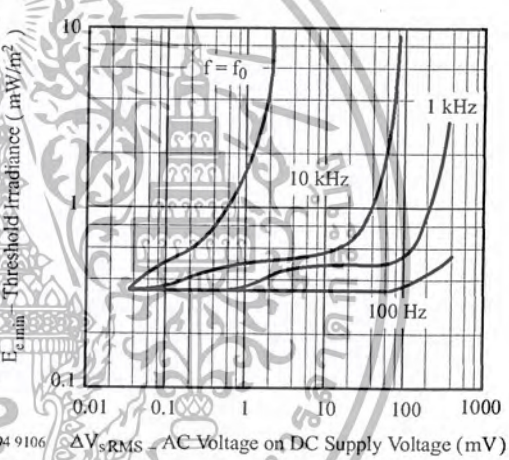


Figure 5. Sensitivity vs. Supply Voltage Disturbances

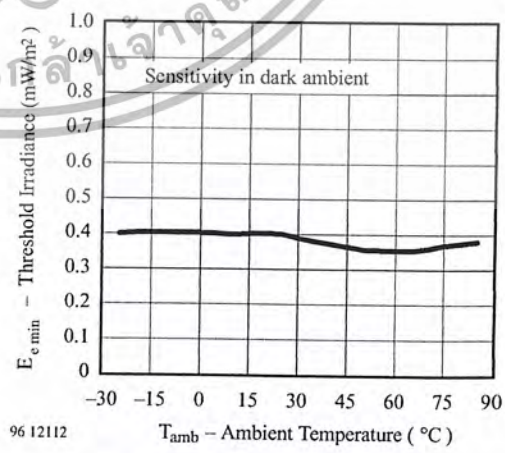
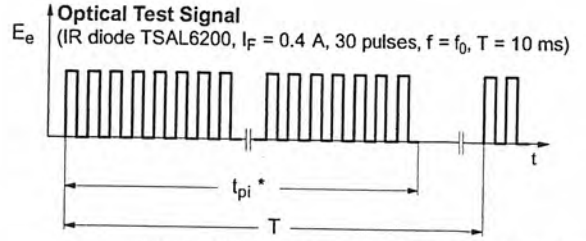


Figure 6. Sensitivity vs. Ambient Temperature



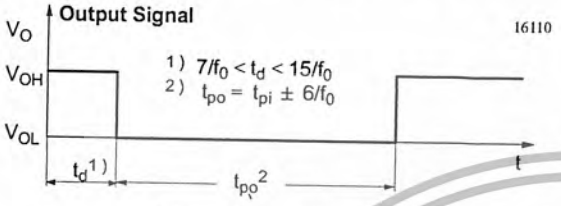


Figure 7. Output Function

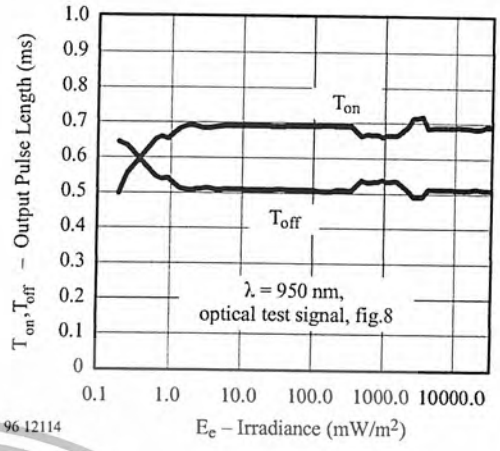


Figure 10. Output Pulse Diagram

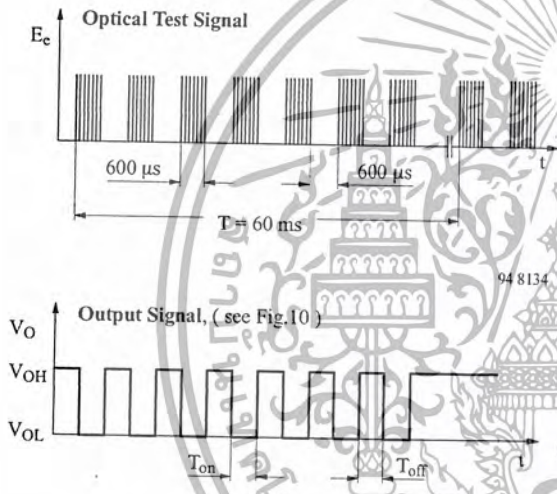


Figure 8. Output Function

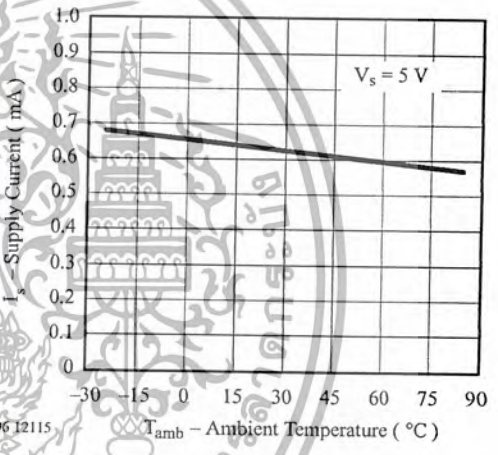


Figure 11. Supply Current vs. Ambient Temperature

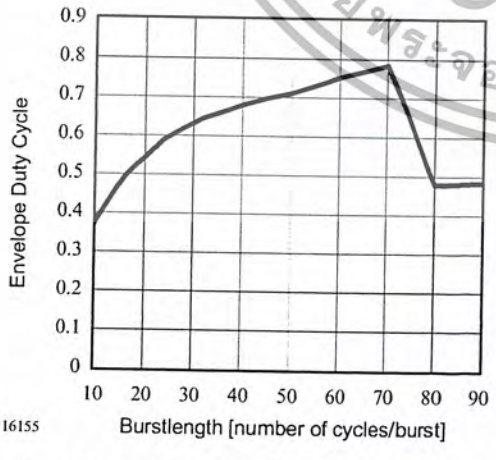


Figure 9. Max. Envelope Duty Cycle vs. Burstlength

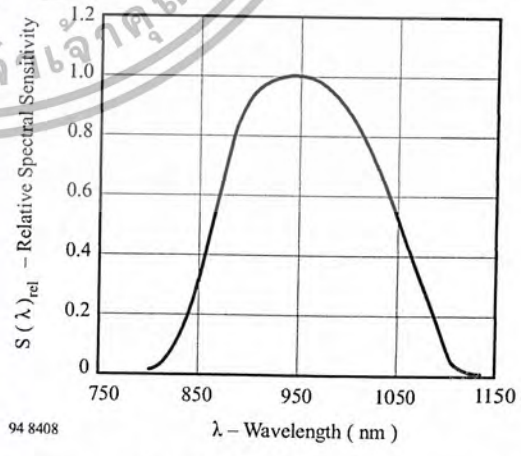


Figure 12. Relative Spectral Sensitivity vs. Wavelength

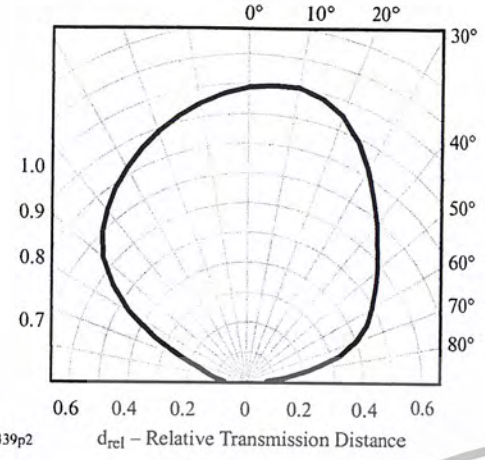


Figure 13. Vertical Directivity φ_y

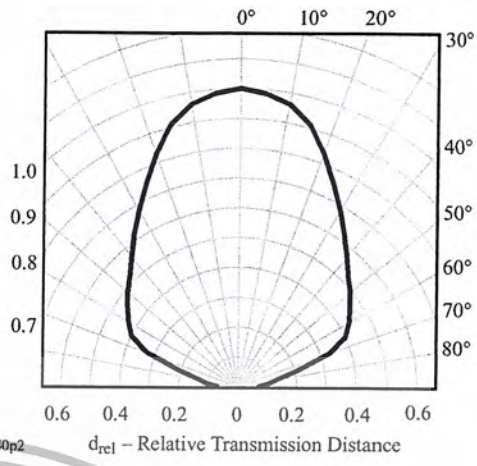
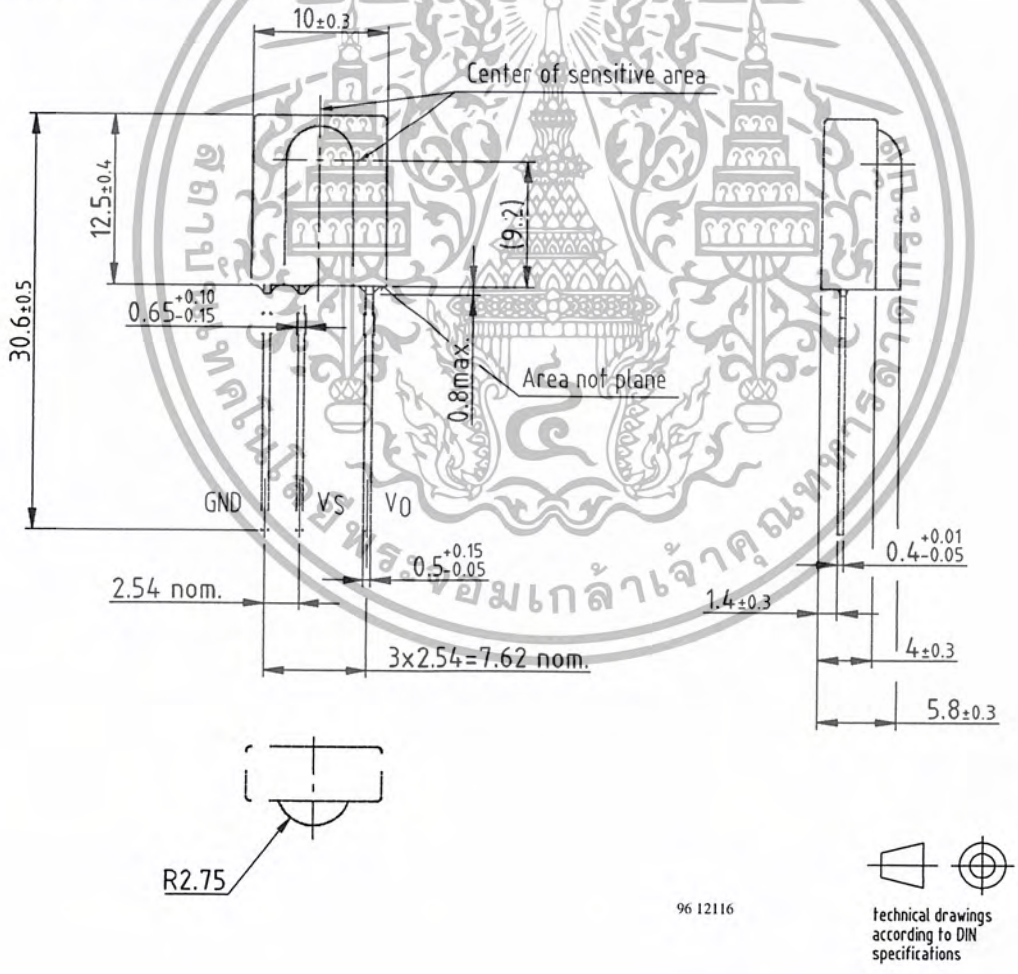


Figure 14. Horizontal Directivity φ_x

Dimensions in mm



Ozone Depleting Substances Policy Statement

It is the policy of **Vishay Semiconductor GmbH** to

1. Meet all present and future national and international statutory requirements.

2. Regularly and continuously improve the performance of our products, processes, distribution and operating systems with respect to their impact on the health and safety of our employees and the public, as well as their impact on the environment.

It is particular concern to control or eliminate releases of those substances into the atmosphere which are known as ozone depleting substances (ODSs).

The Montreal Protocol (1987) and its London Amendments (1990) intend to severely restrict the use of ODSs and forbid their use within the next ten years. Various national and international initiatives are pressing for an earlier ban on these substances.

Vishay Semiconductor GmbH has been able to use its policy of continuous improvements to eliminate the use of ODSs listed in the following documents.

1. Annex A, B and list of transitional substances of the Montreal Protocol and the London Amendments respectively

2. Class I and II ozone depleting substances in the Clean Air Act Amendments of 1990 by the Environmental Protection Agency (EPA) in the USA

3. Council Decision 88/540/EEC and 91/690/EEC Annex A, B and C (transitional substances) respectively.

Vishay Semiconductor GmbH can certify that our semiconductors are not manufactured with ozone depleting substances and do not contain such substances.

Vishay Semiconductor GmbH, P.O.B. 3535, D-74025 Heilbronn, Germany
Telephone: 49 (0)7131 67 2831, Fax number: 49 (0)7131 67 2423