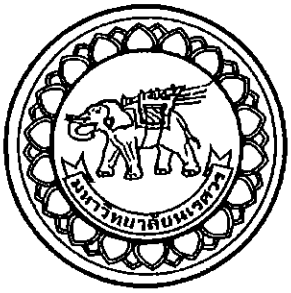


ระบบพ่นละอองน้ำอัตโนมัติเพื่อลดความร้อนให้กับคอนเดนเซอร์ของเครื่องปรับอากาศ

AUTOMATIC WATER SPRAYING SYSTEM FOR COOLING AIR CONDITIONER CONDENSER

นายพิเชษฐ์ เป็งขัน รหัส 50380447

นายบุญญฤทธิ์ กล้วยน้อย รหัส 50381086

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้าภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์มหาวิทยาลัยนเรศวร

ปีการศึกษา 2557

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน ระบบพ่นละอองน้ำอัตโนมัติเพื่อลดความร้อนให้กับคอนเดนเซอร์ของเครื่องปรับอากาศ

ผู้ดำเนินโครงงาน นายพิเชษฐ์ เป็งขัน รหัส 50380447

นายบุญญฤทธิ์ กล้วยน้อย รหัส 50381086

ที่ปรึกษาโครงงาน ผู้ช่วยศาสตราจารย์ ดร. อัครพันธ์ วงศ์กังแห

สาขาวิชา วิศวกรรมไฟฟ้า

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2557

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....ที่ปรึกษาโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร. อัครพันธ์ วงศ์กังแห)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ศุภวรรณ พลพิทักษ์ชัย)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ศิษฐ์ภัณฑ์ แคนลา)

ชื่อหัวข้อโครงงาน	ระบบพ่นละอองน้ำอัตโนมัติเพื่อลดความร้อนให้กับคอนเดนเซอร์ของเครื่องปรับอากาศ
ผู้ดำเนินโครงงาน	นายพิเชษฐ์ เป็งขัน รหัส 50380447
	นายบุญญฤทธิ์ กล้วยน้อย รหัส 50381086
ที่ปรึกษาโครงงาน	ผู้ช่วยศาสตราจารย์ ดร. อัครพันธ์ วงศ์กังแห
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2557

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอ การประยุกต์ใช้ ไมโครคอนโทรลเลอร์ในการควบคุมการสเปรย์น้ำเพื่อการระบายความร้อนที่ คอนเดนเซอร์แอร์ โดยคอนเดนเซอร์มักจะมีการแลกเปลี่ยนระบายความร้อนโดยรอบ เราจึงใช้ไมโครคอนโทรลเลอร์อ่านค่าอุณหภูมิจากเซ็นเซอร์วัดอุณหภูมิ (DS1820) และถ้าอุณหภูมิเกินกว่าค่าที่ตั้งไว้จะมีการสเปรย์น้ำลงในพื้นที่โดยรอบและแลกเปลี่ยนความร้อน ผลของโครงงานนี้คือการประหยัดพลังงานในบางระยะเวลาของวัน

Project title	Automatic Water Spraying System for Cooling Air Conditioner Condenser
Name	Mr. Pichet Pengkhan ID. 50380447
	Mr. Bunyarit Gulaynoi ID. 50381086
Project advisor	Asst. Prof. Dr. Akaraphunt Vongkunghae
Major	Electrical Engineering
Department	Electrical and Computer Engineering
Academic year	2014

Abstract

This thesis presents application of a microcontroller to control water spray for cooling the air conditioner condenser. The condenser usually uses the surrounding for cooling its heat exchanger. Microcontroller reads a temperature from the temperature sensor (DS1820), and if the temperature is greater than a setting point , the water is spraying for cooling down the surrounding area and the air conditioner heat exchanger .The result of this project is the energy saving on a some period of day time.

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุลวงไปได้ด้วยความกรุณาเป็นอย่างยิ่งจาก ผศ.ดร. อัครพันธ์ วงศ์กังแห ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน และให้ความกรุณาในการตรวจทานปริญญานิพนธ์ คณะผู้ดำเนินโครงงานขอกราบขอบพระคุณเป็นอย่างสูงและขอระลึกถึงความกรุณาของทานไว้ตลอดไป

นอกจากนี้ยังต้องขอขอบคุณภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ มหาวิทยาลัยนเรศวร ที่ให้ยืมอุปกรณ์และเครื่องมือวัดมาใช้งาน จนทำให้โครงงานนี้สำเร็จลุลวงไปได้

เหนือสิ่งอื่นใด คณะผู้ดำเนินโครงงานขอกราบขอบพระคุณคุณพ่อ คุณแม่ ผู้มอบความรักความเมตตา สติปัญญา รวมทั้งเป็นผู้ให้ทุกสิ่งทุกอย่างตั้งแต่วัยเยาว์จวบจนถึงปัจจุบัน คอยเป็นกำลังใจทำให้ได้รับความสำเร็จอย่างทุกวันนี้ และขอขอบคุณทุกๆคนในครอบครัวของ คณะผู้ดำเนินโครงงานที่ไม่ได้กล่าวไว้ ณ ที่นี้ด้วย

นายพิเชษฐ์ เป็งขัน

นายบุญญฤทธิ์ กล้วยน้อย

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในการทำงานปรับอุณหภูมิของระบบปรับอากาศแบบแยกส่วน ความต้องการพลังงานไฟฟ้าส่วนใหญ่เกิดจากการทำงานของคอนเดนเซอร์ (Condenser) ซึ่งทำหน้าที่แลกเปลี่ยนความร้อน โดยถูกติดตั้งอยู่ภายนอกห้องมีมอเตอร์พัดลมเพื่อระบายความร้อนให้กับคอยล์ร้อน เพื่อให้น้ำยาแอร์มีอุณหภูมิลดลง จากการทำงานที่หนักและนานของมอเตอร์พัดลมประกอบกับปัจจัยที่สำคัญคือ อุณหภูมิที่สูงโดยรอบคอนเดนเซอร์ ณ ขณะนั้นจึงส่งผลกระทบทำให้การระบายความร้อนเป็นไปได้ช้าและทำให้ต้องใช้พลังงานไฟฟ้ามากขึ้น

เพื่อให้การทำงานของคอนเดนเซอร์มีประสิทธิภาพสูงขึ้น ทางกลุ่มผู้จัดทำโครงงานจึงมีแนวคิดที่จะทำให้อากาศโดยรอบคอนเดนเซอร์มีอุณหภูมิที่ต่ำลงโดยการพ่นละอองน้ำเข้าไปในอากาศก่อนที่จะเข้าสู่ตัวคอนเดนเซอร์ แต่การพ่นละอองน้ำเข้าสู่ตัวคอนเดนเซอร์โดยตรงอาจทำให้เกิดการเปียกแฉะและเกิดตะกรันจับที่บริเวณครีบระบายความร้อนส่งผลให้เกิดการอุดตันและถ่ายเทความร้อนได้ไม่ดี จึงสามารถแก้ไขโดยการนำแผ่นบัฟเฟอร์มาใช้ในการดักจับละอองน้ำทำให้มีแต่ลมเย็นผ่านเข้าสู่ตัวคอนเดนเซอร์อย่างเดียว โดยได้ออกแบบสร้างชุดควบคุมระบบการพ่นละอองน้ำให้เปิดปิดเป็นไปอย่างอัตโนมัติ โดยใช้ดิจิตอลเทอร์มิสเตอร์ds1820 มาเป็นตัวเซนเซอร์ตรวจจับอุณหภูมิ เพื่อเป็นเงื่อนไขในการพ่นละอองน้ำ และเชื่อมต่อกับชุดไมโครคอนโทรลเลอร์เพื่อใช้เป็นตัวประมวลผลและควบคุมการทำงานของรีเลย์เพื่อเปิดปิดโซลินอยด์วาล์วสำหรับการพ่นละอองน้ำให้เป็นไปตามเงื่อนไขของอุณหภูมิที่กำหนดไว้ ส่งผลทำให้คอนเดนเซอร์ระบายความร้อนได้ดีและไวขึ้นช่วยลดการใช้พลังงานไฟฟ้าลงอีกทั้งยังช่วยยืดอายุการทำงานให้กับคอนเดนเซอร์อีกด้วย

1.2 วัตถุประสงค์ของโครงงาน

1) ออกแบบชุดควบคุมการพ่นละอองน้ำด้วยไมโครคอนโทรลเลอร์ โดยใช้ค่าอุณหภูมิที่วัดได้จากดิจิตอลเทอร์มิสเตอร์ds1820 เป็นเงื่อนไขในการเปิดปิดโซลินอยด์วาล์ว
2) เพื่อเปรียบเทียบประสิทธิภาพการทำงานของคอนเดนเซอร์ก่อนและหลังการติดตั้งอุปกรณ์พ่นน้ำ

1.3 ขอบเขตของโครงงาน

1) สร้างชุดควบคุมโดยใช้ไมโครคอนโทรลเลอร์ควบคุมการทำงานของรีเลย์เพื่อเปิดปิดโซลินอยด์วาล์วสำหรับการพ่นละอองน้ำ
2) การใช้ดิจิตอลเทอร์มิสเตอร์ds1820 ในการตรวจวัดอุณหภูมิสำหรับเป็นเงื่อนไขในการพ่นละอองน้ำ
3) ทดสอบการทำงานของชุดควบคุมโดยการติดตั้งกับคอนเดนเซอร์ยูนิต

1.4 ขั้นตอนและแผนการดำเนินงาน

รายละเอียด	ปี 2557				ปี 2558		
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
1) ศึกษาและรวบรวมข้อมูล							
2) ออกแบบวงจรและเลือกอุปกรณ์ในโครงงาน							
3) สร้างและทดสอบวงจรควบคุม							
4) ออกแบบและบันทึกผลการทดลอง							
5) สรุปผลการดำเนินงาน							
6) จัดทำรูปเล่มปริญญานิพนธ์							

1.5 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน

1) เข้าใจและสามารถสร้างวงจรควบคุมโดยใช้ไมโครคอนโทรลเลอร์ได้
2) เพิ่มประสิทธิภาพการทำงานของคอนเดนเซอร์ ทำให้ใช้พลังงานไฟฟ้าลดลงและยืดอายุการบำรุงรักษา
3) สามารถนำความรู้ที่ได้ไปต่อยอดในการทำงานต่อไป

1.6) งบประมาณ

1) อุปกรณ์สร้างบอร์ดไมโครคอนโทรลเลอร์	400 บาท
2) อุปกรณ์สร้างแหล่งจ่ายแรงดัน	100 บาท
3) อุปกรณ์สื่อสารข้อมูลRS232	300 บาท
4) จอแสดงผลแอลซีดี	200 บาท
5) ดิจิตอลเทอร์มิสเตอร์DS1820	50 บาท
6) สวิตช์ควบคุม	20 บาท
7) อุปกรณ์รีเลย์	100 บาท
8) โซลินอยด์วาล์ว	130 บาท
9) หัวสเปรย์น้ำ,สายยาง,แผ่นบัฟเฟอร์	100 บาท
10) ค่าถ่ายเอกสารและเข้าเล่มปริญญานิพนธ์	1,000 บาท
รวมเป็นเงินทั้งสิ้น (สองพันสี่ร้อยบาทถ้วน)	2,400 บาท

หมายเหตุ: ถัวเฉลี่ยทุกรายการ

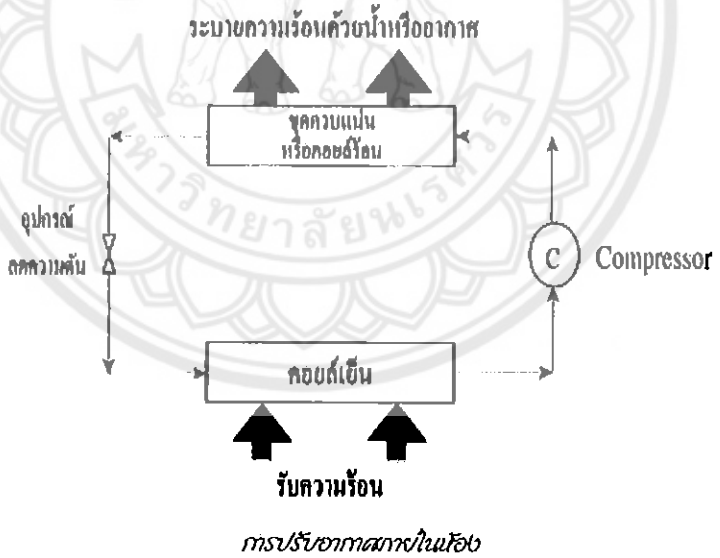
บทที่ 2

หลักการและทฤษฎี

ในบทนี้จะกล่าวถึงหลักการทำงานของระบบปรับอากาศและอธิบายทฤษฎีที่เกี่ยวของกับอุปกรณ์ต่างๆเพื่อใช้ในการทำโครงงาน ซึ่งประกอบด้วย วงจรควบคุมค่าแรงดัน ระบบส่งผ่านข้อมูล เซนเซอร์วัดอุณหภูมิ วงจรรีเลย์ วงจรแสดงผลแอลซีดี และได้นำไมโครคอนโทรลเลอร์มาใช้ในการประมวลผล พร้อมทั้งหลักการของภาษาซี ซึ่งเป็นภาษาที่เลือกมาใช้เขียนโปรแกรมเพื่อควบคุมการทำงานของระบบ

2.1 หลักการทำงานของระบบปรับอากาศ[1]

หลักการคือการนำเอาความร้อนจากที่ที่ต้องการทำความเย็นถ่ายเทไปทิ้งยังที่ที่ไม่ต้องการทำความเย็นโดยมีตัวกลางคือสารทำความเย็นหรือน้ำยาแอร์



รูปที่ 2.1 แสดงลักษณะการทำงานของระบบปรับอากาศ

การทำงานเริ่มต้นที่คอมเพรสเซอร์ (Compressor) จะดูดน้ำยาที่เป็นไอจากอีวาโปเรเตอร์ (Evaporator) หรือคอยล์เย็นไอน้ำยาจะมีความดันและอุณหภูมิต่ำจากนั้นไอน้ำยาที่ถูกดูดเข้ามา

คอมเพรสเซอร์จะอัดน้ำยาให้มีความดันและอุณหภูมิสูงขึ้นจากนั้นน้ำยาจะถูกดันและส่งผ่านไปยังคอนเดนเซอร์ (Condenser) หรือคอยล์ร้อนคอนเดนเซอร์มีหน้าที่ระบายความร้อนจากน้ำยาผ่านตัวกลางคืออากาศไอน้ำยาจะมีอุณหภูมิต่ำลงจนควบแน่นเป็นของเหลวแต่ความดันและอุณหภูมิยังสูงอยู่น้ำยาที่เป็นของเหลวจะถูกส่งไปยังอุปกรณ์ลดความดัน (Expansion Valve) ซึ่งทำหน้าที่ลดความดันก่อนเข้าอีวาโปเรเตอร์ทำให้น้ำยามีความดันและอุณหภูมิต่ำเมื่อไหลเข้าอีวาโปเรเตอร์ก็จะรับความร้อนผ่านตัวกลางคืออากาศทำให้สารทำความเย็นเดือดกลายเป็นไอของสารทำความเย็นที่ออกจากอีวาโปเรเตอร์จะมีความดันและอุณหภูมิต่ำและไหลกลับเข้าคอมเพรสเซอร์เพื่อทำการเพิ่มความดันต่อไประบบการทำความเย็นของเครื่องปรับอากาศจะทำงานวนเวียนเป็นวัฏจักรตลอดเวลาที่คอมเพรสเซอร์ยังคงทำงานอยู่และน้ำยาที่มีอยู่ในระบบจะไม่มีการสูญเสียไปไหนเลยนอกจากเกิดการรั่วซึม

ในระบบทำความเย็นเบื้องต้นนี้มีทั้งน้ำยาที่อยู่ในสภาพความดันและอุณหภูมิสูงกับแรงดันและอุณหภูมิต่ำจึงมีการแบ่งภาคออกเป็น 2 ภาค

1) ทางด้านความดันสูงซึ่งจะเริ่มจากทางอัดของคอมเพรสเซอร์ผ่านคอนเดนเซอร์จนถึงทางเข้าของอุปกรณ์ลดความดันส่วนนี้สารทำความเย็นจะมีทั้งความดันและอุณหภูมิสูงจึงเรียกว่าทาง High Side

2) ทางด้านความดันต่ำซึ่งจะเริ่มตั้งแต่ทางออกของอุปกรณ์ลดความดันผ่านอีวาโปเรเตอร์จนถึงทางเข้าของคอมเพรสเซอร์ส่วนนี้จะมีทั้งความดันและอุณหภูมิต่ำจึงเรียกว่าทาง Low Side

2.1.1 ส่วนประกอบของเครื่องปรับอากาศเบื้องต้น

เครื่องปรับอากาศแยกตามระบบการทำงานแบ่งออกเป็น 2 ส่วนตามการติดตั้ง คือ

1) ส่วนที่ติดตั้งภายในบ้าน (Indoor Unit) ทำหน้าที่ดูดซับความร้อนทำให้อากาศภายในห้องเย็นลงหรือเรียกว่า คอยล์เย็น (Evaporator) ประกอบด้วยท่อแผงคอยเย็น พัดลม และอุปกรณ์วัดและควบคุมอุณหภูมิห้องซึ่งส่วนคอยล์เย็นนี้จะถูกติดตั้งภายในห้องที่เราสังเกตเห็นทั่วไป

2) ส่วนที่ติดตั้งภายนอกบ้าน (Outdoor Unit) ประกอบด้วย 3 ส่วน คือส่วนที่ทำหน้าที่ในการระบายความร้อนหรือที่เรียกว่า คอยล์ร้อน (Condensing) ส่วนที่ทำหน้าที่ควบคุมการไหลเวียนของน้ำยาทำความเย็นหรือที่เรียกว่าคอมเพรสเซอร์ และน้ำยาแอร์ และอีกส่วนทำหน้าที่ในการลดความดันและอุณหภูมิของน้ำยาแอร์หรือที่เรียกว่า อุปกรณ์ลดความดัน (Throttling Device) ซึ่งส่วนคอยล์ร้อนนี้จะถูกติดตั้งภายนอกห้องที่เราสังเกตเห็นทั่วไปแต่ทั้งนี้คอยล์ร้อนมักไม่ติดตั้งให้ห่าง

จากห้องมากหรือพยายามให้ใกล้กับคอยล์เย็นที่ติดตั้งในห้องมากที่สุดเพื่อประหยัดไฟ และน้ำยาแอร์ที่ต้องเดินทางไกล

หากแบ่งตามหน้าที่การทำงาน สามารถแบ่งส่วนประกอบของแอร์ออกเป็น 3 ส่วน ดังนี้

1) คอยล์เย็น (Evaporator) เป็นชุดของแอร์ที่ถูกติดตั้งภายในห้องประกอบด้วยแผงคอยล์เย็นที่ติดกับส่วนท่อน้ำยาแอร์ที่บรรจุน้ำยาแอร์ไหลเวียนภายในส่งต่อไปยังคอมเพรสเซอร์โดยขณะไหลเวียนผ่านท่อจะมีพัดลมคอยดูดอากาศภายในห้องจากด้านล่างเครื่องผ่านท่อและแผงคอยล์เย็นเพื่อให้เกิดการแลกเปลี่ยนความร้อนและผ่านอากาศเย็นออกมายังช่องแอร์ในด้านปลายมีดังนี้

1.1 แผงคอยล์เย็นมีรูปร่างเป็นเส้นท่อขดไปมาตามความยาวของเครื่องและจะมีแผ่นครีบอลูมิเนียมบางๆ หุ้มขดท่อเหล่านั้นอยู่แผงขดท่อจะมองเห็นได้อย่างชัดเจนเมื่อถอดหน้ากากส่งลมหรือหน้ากากรับลมกลับของเครื่องออกแผงคอยล์เย็นนี้จะน้ำยาแอร์ไหลเวียนภายในท่อเพื่อรับถ่ายเทความร้อนที่อยู่ในอากาศภายในห้องให้มีอุณหภูมิที่เย็นลง

1.2 มอเตอร์พัดลมคอยล์เย็นมีใบพัดลมคอยล์เย็นเป็นตัวขับเคลื่อนให้เกิดการเคลื่อนที่ของลมโดยได้กำลังมาจากมอเตอร์ไฟฟ้าเหนี่ยวนำชนิด 1 เฟสทำหน้าที่กระจายลมเย็นให้กับพื้นที่ที่ต้องการและส่วนสำคัญของพัดลมคือหน้าที่ในการนำส่งลมผ่านแผ่นกรองอากาศเพื่อกรองฝุ่นที่ปะปนอยู่ในห้องให้สะอาดและยังเป็นส่วนสำคัญในการนำพาอากาศเข้าแลกเปลี่ยนความเย็นและถ่ายเทความร้อนให้แก่น้ำยาแอร์ในแผงคอยล์เย็น

1.3 ชุดควบคุมระบบปรับอากาศเป็นอุปกรณ์ที่ใช้สำหรับควบคุมการทำงานของเครื่องปรับอากาศโดยทั่วไประบบปรับอากาศรุ่นใหม่ๆจะใช้อุปกรณ์อิเล็กทรอนิกส์มาควบคุมระบบการทำงานซึ่งการควบคุมแบบอิเล็กทรอนิกส์จะมีความเที่ยงตรงสูงกว่าแบบเก่าที่ใช้กลไกทางกลในการควบคุม อุปกรณ์นี้จะทำหน้าที่ในการปรับตั้งอุณหภูมิภายในห้องพร้อมทำหน้าที่ในการตัดและเปิดระบบการทำงานเมื่ออุณหภูมิภายในห้องได้ตามที่เราตั้งไว้

1.4 อุปกรณ์ควบคุมแรงดันสารทำความเย็น การควบคุมปริมาณสารความเย็นในระบบปรับอากาศแบบแยกส่วนใช้การควบคุมโดยท่อรูเข็ม (Capillary Tube – Cap. Tube) โดยติดตั้งระหว่างคอนเดนเซอร์กับอีวาโปเรเตอร์ทำหน้าที่ลดแรงดันและควบคุมปริมาณสารทำความเย็นก่อนเข้าคอยล์เย็น

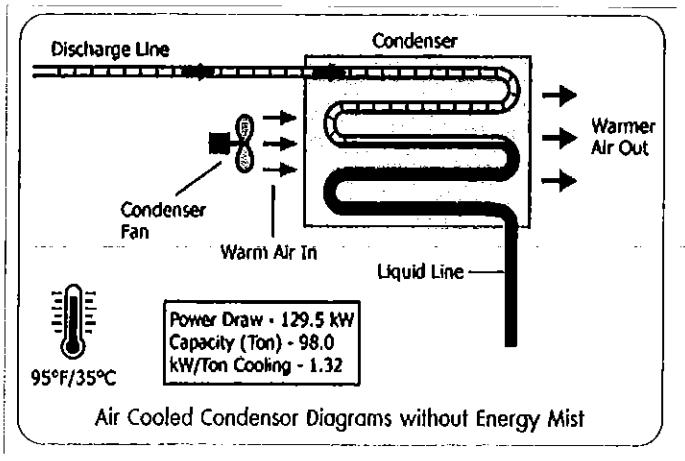
2) คอยล์ร้อน (Condensing) เป็นชุดของแอร์ที่ถูกติดตั้งภายนอกอาคารเหมือนกัน ประกอบด้วยแผงคอยล์ร้อนหรือที่เรียกว่า อุปกรณ์ลดความดัน (Throttling Device) มีลักษณะเป็นท่อทองแดงขนาดเล็กขดเป็นแผงไปมาอยู่ด้านหลังถัดจากใบพัดลมและอีกส่วนจะเป็นพัดลมที่ช่วยในการระบายความร้อนของน้ำยาแอร์ทำให้น้ำยาแอร์มีอุณหภูมิลดลงแต่ความดันคงที่ก่อนไหลเวียนเข้าสู่แผงคอยล์เย็นเพื่อลดอุณหภูมิในห้องต่อไป

2.1 คอมเพรสเซอร์ และน้ำยาแอร์ (Evaporator) เป็นส่วนอุปกรณ์ที่ถูกติดตั้งนอกอาคารด้านข้างพัดลมระบายความร้อนทำหน้าที่ในการเก็บ และควบคุมการไหลเวียนของน้ำยาแอร์ในระบบและทำให้น้ำยาแอร์มีอุณหภูมิและความดันสูงขึ้นก่อนส่งผ่านไปยังตามท่อของส่วนคอยล์ร้อนต่อไป

2.2 แผงคอยล์ร้อนมีลักษณะเป็นท่ออลูมิเนียมหรือทองแดงขนาดเล็กขดไปมาเป็นแผงอยู่ภายใน ครีบอลูมิเนียมแผ่นเล็กๆเรียงซ้อนกันทำหน้าที่เป็นพื้นที่สำหรับระบายความร้อนของสารทำความเย็นที่ส่งผ่านมาจากคอมเพรสเซอร์ โดยอาศัยการนำพาอากาศผ่านพัดลม

2.3 มอเตอร์พัดลม และพัดลมคอยล์ร้อนเป็นมอเตอร์เหนี่ยวนำ 1 เฟสทำหน้าที่เป็นตัวระบายความร้อนให้กับสารทำความเย็นที่มีสถานะเป็นก๊าซให้มีสถานะเป็นของเหลวเพื่อส่งผ่านเข้าสู่อุปกรณ์ควบคุมแรงดันสารทำความเย็นการติดตั้งคอยล์ร้อนในตำแหน่งที่มีอากาศเย็นหรือไม่มีแดดส่องถึงจะช่วยให้ประสิทธิภาพการระบายอากาศดีขึ้น

2.1.2 คอนเดนเซอร์ระบายความร้อนด้วยอากาศ (Air Cooled Condensers)



รูปที่ 2.2 การระบายความร้อนด้วยอากาศของคอนเดนเซอร์

สำหรับคอนเดนเซอร์ที่ระบายความร้อนด้วยอากาศนั้น การเคลื่อนที่ของอากาศเป็นการเสริมประสิทธิผลของการถ่ายเทความร้อน อาจเป็นได้ทั้งโดยธรรมชาติหรือโดยวิธีกล คือใช้พัดลมพัดผ่านหรือดูดผ่านคอนเดนเซอร์ หากโดยอาศัยแรงลมตามธรรมชาติ ปริมาณลมที่พัดผ่านคอนเดนเซอร์เพื่อพาความร้อนจากคอนเดนเซอร์ไปด้วยมีน้อย ขนาดของคอนเดนเซอร์จึงต้องโตเพื่อเพิ่มพื้นที่ผิวถ่ายเทความร้อนออกสู่บรรยากาศด้วยเหตุที่คอนเดนเซอร์ที่อาศัยแรงลงธรรมชาติต้องมีขนาดใหญ่ เพราะประสิทธิภาพการถ่ายเทความร้อนมีน้อย การใช้งานคอนเดนเซอร์แบบนี้จึงใช้กับงานเครื่องทำความเย็นขนาดเล็ก เช่น ตู้เย็นขนาดเล็ก เครื่องทำน้ำเย็น เป็นต้น รูปแบบของคอนเดนเซอร์อาจเป็นแผง (Plate surface) หรือเป็นแบบท่อครีบ (finned tubing)

การติดตั้งเครื่องทำความเย็นที่ใช้คอนเดนเซอร์แบบแผง สำหรับตู้เย็นจะติดที่ด้านหลังของตัวตู้ในลักษณะที่ให้มีช่องอากาศระหว่างตัว ตู้กับคอนเดนเซอร์เพื่อให้อากาศไหลผ่านแผงถ่ายเทความร้อนได้สะดวก มีคอนเดนเซอร์แบบแผงบางชนิดที่ติดตั้งที่ส่วนใต้ของตู้ ซึ่งการติดตั้งจะเป็นที่ส่วนใดของตู้ก็ตาม ความต้องการสำหรับคอนเดนเซอร์แบบนี้ก็คือ ช่องว่างที่มากพอสำหรับระบายอากาศ การตั้งตู้เย็นเพื่อใช้งานจึงควรคำนึงถึงสิ่งเหล่านี้คือ หากแผงคอนเดนเซอร์อยู่ด้านหลังของตัวตู้ การวางตู้เย็นไม่ควรให้ชิดผนังมากเกินไป ควรให้ห่างจากผนังประมาณ 30 เซนติเมตร (1 ฟุต) เพื่อให้อากาศถ่ายเทได้ และไม่ควรวางอยู่ที่ซอกที่มีวัสดุอื่นปิดทางกระแสลม และไม่ควรวางตู้เย็นไว้ใกล้เตาไฟหรืออุปกรณ์ทำความร้อนอื่นๆ เพราะสิ่งเหล่านี้มีส่วนลดประสิทธิภาพการทำความเย็นของตู้เย็น

คอนเดนเซอร์แบบท่อครีบ โครงสร้างจะเป็นแบบท่อทองแดงไม่มีตะเข็บสอดในแผ่นครีบคล้ายรังผึ้งหม้อน้ำรถยนต์ โอกาสที่จะสกปรกมีเศษวัสดุปิดกั้นกระแสลมจึงมีมาก โดยทั่วไปคอนเดนเซอร์แบบท่อครีบจะมีพัดลมประกอบอยู่ในระบบ พัดลมจะสร้างกระแสลม (forced – air) ให้ไหลผ่านคอนเดนเซอร์เพื่อเพิ่มและสิทธิภาพการถ่ายเทความร้อนการที่พัดลมประกอบในคอนเดนเซอร์แบบท่อครีบแบ่งได้ 2 ประเภทคือ

1) แบบติดตั้งรวมกับเครื่องอัด โดยคอนเดนเซอร์จะติดตั้งบนฐานเดียวกับเครื่องอัด และต้นกำลังของชุดอัด (Compressor) กรณีชุดอัดเป็นแบบกึ่งปิดสนิท (Semi- hermetic Compressor) จะไม่รวมอีเวปอเรเตอร์ และตัวควบคุมปริมาณสารความเย็นไว้ด้วย อุปกรณ์ชุดทำความเย็นลักษณะนี้มีชื่อเรียกอีกอย่างหนึ่งว่า ชุดควบแน่น (Condensing Unit)

2) แบบติดตั้งแยกกับเครื่องอัดการติดตั้งจะแยกจะกันระหว่างคอนเดนเซอร์และเครื่องอัด คอนเดนเซอร์แบบนี้สามารถติดตั้งได้ทั้งในอาคารและนอกอาคาร หากติดตั้งในอาคาร เช่น ในห้องที่มีผนังรอบด้านจะต้องมีช่องลมเพื่อให้อากาศภายนอกเข้ามาพาความร้อนที่คายออกจากคอนเดนเซอร์ออกไปนอกห้อง อากาศภายนอกจะเข้าทางช่องหน้าต่าง ถูกพัดลมดูดมาสู่คอนเดนเซอร์และเป่าผ่านคอนเดนเซอร์สู่ภายนอก ในขณะอากาศผ่านคอนเดนเซอร์ออกสู่ภายนอกนี้จะพาความความร้อนซึ่งมากับไอสารความเย็นออกด้วยทางหน้าต่าง

2.1.3 สมรรถนะของคอนเดนเซอร์โดยทั่วไป

สมรรถนะของคอนเดนเซอร์ หมายถึง ความสามารถที่คอนเดนเซอร์สามารถถ่ายเทความร้อนออกจากไอสารความเย็นจนสารความเย็นนั้นเปลี่ยนสภาพจากไอเป็นของเหลวซึ่งประกอบด้วยหลายข้อที่ควรพิจารณา คือ

1) พื้นที่ผิวของคอนเดนเซอร์ ปกติเล่นทางที่ไอสารความเย็นเคลื่อนที่ไปเป็นเพียงท่อทองแดงตามขนาดที่เหมาะสมและมีความยาวของท่อจำนวนหนึ่ง เพื่อความสะดวกในการใช้งานจึงนำท่อที่ใช้มาขดให้เป็นรูปร่างที่กะทัดรัดและเพิ่มประสิทธิภาพการถ่ายเทความร้อนด้วยการเพิ่มครีบ (Fins)

2) พื้นที่ผิวสัมผัสของไอสารความเย็นกับผิวด้านในของท่อ หากผิวสัมผัสด้านในของท่อมากการถ่ายเทความร้อนจากไอสารความเย็นมาสู่ท่อจะมาก แต่ในทางปฏิบัติ ต้องคำนึงถึงราคาของผลผลิตด้วย หากใช้ท่อโต ราคาจะสูง จึงต้องหาขนาดที่เหมาะสมโดยพิจารณาทั้งประสิทธิภาพของการถ่ายเทความร้อนและราคา

3) ความแตกต่างของอุณหภูมิของไอสารความเย็นกับอุณหภูมิของตัวกลางที่ใช้ในการพาความร้อนไป เช่น น้ำหรืออากาศ หากตัวกลางมีอุณหภูมิต่ำขณะที่ไอสารความเย็นมีอุณหภูมิสูง การถ่ายเทความร้อนย่อมมาก ในทางตรงข้ามหากทั้งไอสารความเย็นและตัวกลางมีอุณหภูมิใกล้เคียงกัน การถ่ายเทความร้อนจะมีน้อย

4) อัตราความเร็วของตัวกลางที่ไหลผ่านคอนเดนเซอร์ เช่นอากาศที่พัดผ่านผิวท่อด้านนอกของคอนเดนเซอร์ หากมีความเร็วมากจะมีโอกาสพาความร้อนให้หลุดออกไปจากผิวท่อได้มาก จึง

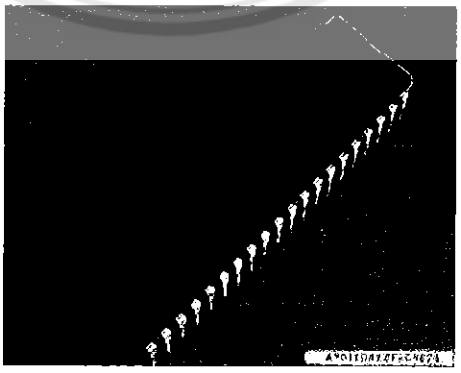
มีการใช้พัดลมดูดลมหรือเป่าลมผ่านท่อครีบคอนเดนเซอร์ เพื่อเพิ่มประสิทธิภาพการถ่ายเทความร้อน หรือน้ำที่ใช้เป็นตัวกลางพาความร้อนออกไปจากคอนเดนเซอร์ หากมีปริมาณมาก หมายถึงเร็วและแรง การถ่ายเทความร้อนออกจากคอนเดนเซอร์จะมีมากด้วย

5) ความสะอาดของผิวท่อคอนเดนเซอร์ที่มีผลต่อการถ่ายเทความร้อน คราบหรือฝุ่นละอองที่จับผิวท่อคอนเดนเซอร์จะเป็นเหมือนฉนวนของการถ่ายเทความร้อน เศษวัสดุเช่นฟางหรือหญ้าแห้งที่นกนำมาจะเป็นตัวกั้นกระแสลมที่พัดผ่านคอนเดนเซอร์ ตะไคร่น้ำจับตามท่อคอนเดนเซอร์จะทำให้ประสิทธิภาพการระบายความร้อนด้วยน้ำของคอนเดนเซอร์ลดลง

ในการออกแบบสร้างคอนเดนเซอร์แบบต่างๆ จะพิจารณาเพียงอุณหภูมิและความดันอันเป็นผลของคอนเดนเซอร์โดยสิ่งต่างๆ ที่เป็นปัจจัยให้เกิดปัญหากับคอนเดนเซอร์มิได้พิจารณาถึง ฉะนั้นปัจจัยที่จะนำไปสู่การเสื่อมประสิทธิภาพของคอนเดนเซอร์จึงควรทำให้ไม่มีเลยหรือมีน้อยที่สุดเพื่อประสิทธิภาพสูงสุดของเครื่องทำความเย็น

2.2 ไมโครคอนโทรลเลอร์[2]

ไมโครคอนโทรลเลอร์ (อังกฤษ: microcontroller μC, uCหรือ MCU) คือ อุปกรณ์ควบคุมขนาดเล็ก ซึ่งบรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์โดยในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู, หน่วยความจำและพอร์ตซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน โดยทำการบรรจุเข้าไว้ในตัวถังเดียวกัน



รูปที่ 2.3 ไมโครคอนโทรลเลอร์ตระกูล AVR แบบ PIDP

โครงสร้างโดยทั่วไป ของไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ๆ ดังต่อไปนี้

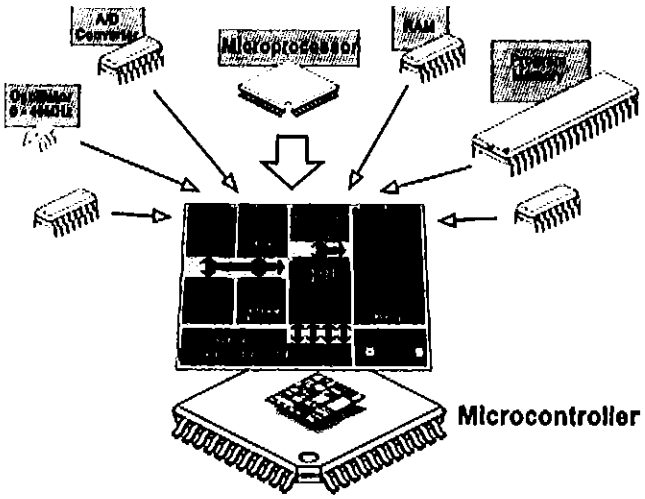
1) หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit) ทำหน้าที่เป็นศูนย์กลางควบคุมการทำงานของระบบคอมพิวเตอร์ทั้งหมดโดยนำข้อมูลจากอุปกรณ์รับข้อมูลมาทำงาน ประมวลผลข้อมูลตามคำสั่งของโปรแกรมและส่งผลลัพธ์ออกไปหน่วยแสดงผล

2) หน่วยความจำ (Memory)สามารถแบ่งออกเป็น ส่วน คือ หน่วยความจำที่มีไว้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของ เครื่องคอมพิวเตอร์ตั้งโต๊ะคือข้อมูลใดๆ ที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยงอีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกกระดาษทดในการคำนวณของซีพียูและเป็นที่พักข้อมูลชั่วคราวขณะทำงาน แต่หากไม่มีไฟเลี้ยงข้อมูลก็จะหายไปคล้ายกับหน่วยความแรม (RAM) ในเครื่องคอมพิวเตอร์ทั่วๆ ไปแต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแรมซึ่งข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Erasable Electrically Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยง

3) ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port)มี 2 ลักษณะคือ พอร์ตอินพุต (Input-Port) และพอร์ตส่งสัญญาณหรือพอร์ตเอาต์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอก ถือว่าเป็นส่วนที่สำคัญมากใช้ร่วมกันระหว่างพอร์ตอินพุต เพื่อรับสัญญาณ อาจจะด้วยการกดสวิตช์เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาต์พุต เพื่อแสดงผลเช่นการติดสว่างของหลอดไฟ เป็นต้น

4) ช่องทางเดินของสัญญาณ หรือบัส (BUS)คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง ซีพียู หน่วยความจำและพอร์ตเป็นลักษณะของสายสัญญาณ จำนวนมากอยู่ภายในตัวไมโครคอนโทรลเลอร์โดยแบ่งเป็นบัสข้อมูล (Data Bus) ,บัสแอดเดรส (Address Bus) และบัสควบคุม (Control Bus)

5) วงจรกำเนิดสัญญาณนาฬิกานับเป็นส่วนประกอบที่สำคัญมากอีกส่วนหนึ่งเนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์จะขึ้นอยู่กับการกำหนดจังหวะ หากสัญญาณนาฬิกามีความถี่สูงจังหวะการทำงานก็จะสามารถทำได้ถี่ขึ้นส่งผลให้ไมโครคอนโทรลเลอร์จังหวะนั้นมีความเร็วในการประมวลผลสูงตามไปด้วย



รูปที่ 2.4 โครงสร้างและส่วนประกอบหลักของไมโครคอนโทรลเลอร์

นอกจากนี้ยังมีส่วนพิเศษอื่นๆ จะขึ้นอยู่กับกระบวนการผลิตของแต่ละบริษัทที่จะผลิตขึ้นมาใส่คุณสมบัติพิเศษลงไปเช่น

- ADC (Analog to Digital) ส่วนภาครับสัญญาณอนาล็อกแปลงไปเป็นสัญญาณดิจิตอล

- DAC (Digital to Analog) ส่วนภาคส่งสัญญาณดิจิตอลแปลงไปเป็นสัญญาณอนาล็อก

- I2C (Inter Integrate Circuit Bus) เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) เพื่อใช้ติดต่อสื่อสาร ระหว่าง ไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอกซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือ serial data (SDA) และสาย serial clock (SCL) ซึ่งสามารถ เชื่อมต่ออุปกรณ์ จำนวนหลายๆตัว เข้าด้วยกันได้ ทำให้ MCU ใช้พอร์ตเพียง 2 พอร์ตเท่านั้น

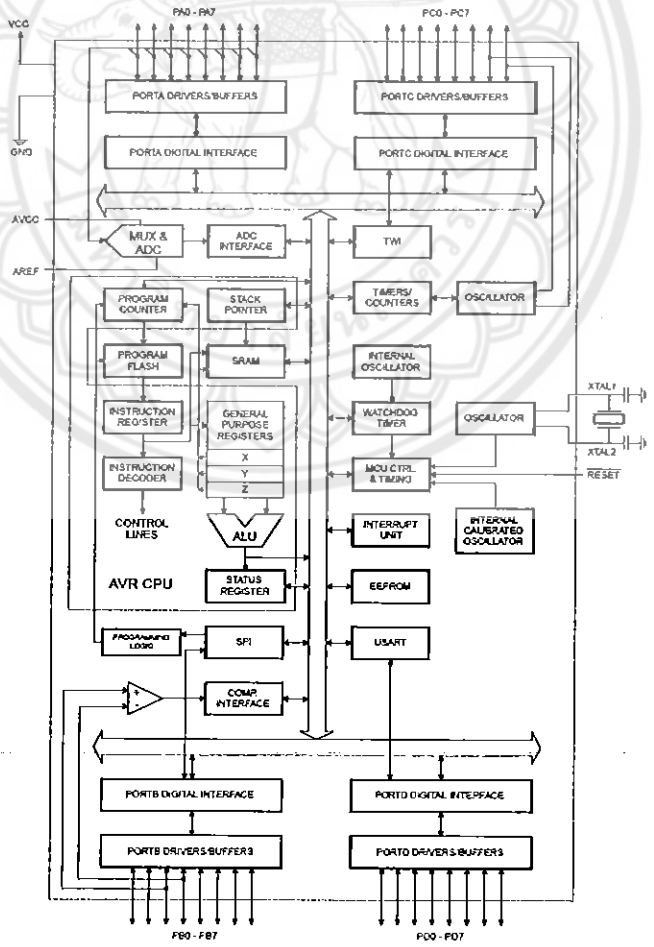
- SPI (Serial Peripheral Interface) เป็นการเชื่อมต่อกับอุปกรณ์เพื่อรับส่งข้อมูลแบบซิงโครนัส (Synchronize) มีสัญญาณนาฬิกาเข้ามาเกี่ยวข้องระหว่างไมโครคอนโทรลเลอร์ (Microcontroller) หรือจะเป็นอุปกรณ์ภายนอกที่มีการรับส่งข้อมูลแบบ SPI อุปกรณ์ที่ทำหน้าที่เป็นมาสเตอร์ (Master) โดยปกติแล้วจะเป็นไมโครคอนโทรลเลอร์ หรืออาจกล่าวได้ว่าอุปกรณ์ Master จะต้องควบคุมอุปกรณ์ Slave ได้ โดยปกติตัว Slave มักจะเป็นไอซี (IC) หน้าที่พิเศษต่างๆ เช่น ไอซีอุณหภูมิ, ไอซีฐานเวลานาฬิกาจริง (Real-Time Clock) หรืออาจเป็นไมโครคอนโทรลเลอร์ที่ทำหน้าที่ในโหมด Slave ก็ได้เช่นกัน

- PWM (Pulse Width Modulation) การสร้างสัญญาณพัลส์แบบสแควร์เวฟที่สามารถปรับเปลี่ยนความถี่และ Duty Cycle ได้เพื่อนำไปควบอุปกรณ์ต่างๆเช่น มอเตอร์

- UART (Universal Asynchronous Receiver Transmitter) ทำหน้าที่รับส่งข้อมูลแบบอะซิงโครนัสสำหรับมาตรฐานการรับส่งข้อมูลแบบ RS-232

2.2.1 ไมโครคอนโทรลเลอร์ตระกูล AVR ATmega16

ATmega16เป็นไมโครคอนโทรลเลอร์สมรรถนะสูงที่ใช้สถาปัตยกรรม AVR แบบ 8 บิต ของบริษัท Atmel ซึ่งเป็นสถาปัตยกรรมแบบ RISC (Reduced Instruction Set Computer) ที่ถูกออกแบบมาให้โปรแกรมมีขนาดเล็กใช้เนื้อที่ในหน่วยความจำน้อยและกินไฟต่ำ ไมโครคอนโทรลเลอร์ตัวนี้ทำหน้าที่เป็นศูนย์กลางการประมวลผลในแผงวงจรหลักของ IPST-MicroBOX



รูปที่ 2.5 รายละเอียดภายในไมโครคอนโทรลเลอร์ตระกูล AVR รุ่น ATmega16

คุณสมบัติพื้นฐานของ ATmega16

สถาปัตยกรรมขั้นสูงแบบ RISC

- ชุดคำสั่งภาษาแอสเซมบลี 131 คำสั่งซึ่งส่วนใหญ่ทำงานที่ 1 รอบสัญญาณนาฬิกา (clock cycle)
- รีจีสเตอร์ขนาด 8 บิต 32 ตัว
- ความเร็วในการประมวลผล 16 ล้านคำสั่งต่อวินาที(Million Instructions Per Second : MIPS) ที่สัญญาณนาฬิกา 16 เมกะเฮริตซ์(MHz)

หน่วยความจำ

- แบบ Flash ขนาด 16 Kbytes สามารถเขียน/ลบโปรแกรมได้ 10,000 ครั้ง
- แบบ EEPROM ขนาด 512 bytes สามารถเขียน/ลบข้อมูลได้ 100,000 ครั้ง
- แบบ SRAM ขนาด 1 กิโลไบต์ (Kbytes)

อินเตอร์เฟส Joint Test Action Gropu : JTAG (IEEE std. 1149.1 Compliant)

- ใช้ในการตรวจสอบวงจรแบบ Boundary scan
- ใช้ในการโปรแกรมชิป flash และ EEPROM
- ใช้ในการตรวจสอบสถานะของชิปตัวเอง

ความเร็วสัญญาณนาฬิกา

-0 ถึง 16 MHz

ไฟเลี้ยง

-ระหว่าง 4.5 ถึง 5.5 VDC

การรองรับอุปกรณ์ต่อพ่วง

- พอร์ตอินพุตและเอาท์พุตขนาด 8 บิตจานวน 4 พอร์ต
- ตัวจับเวลาและตัวนับ (timer/counter) ขนาด 8 บิต 2 ตัวและขนาด 16 บิต 1 ตัว
- ระบบการเปลี่ยนสัญญาณแอนะล็อกเป็นดิจิทัลขนาด 10 บิตจำนวน 8 ช่อง
- ตัวเปรียบเทียบสัญญาณแอนะล็อกภายในชิป

อื่นๆ

- ระบบตรวจจับการทำงานผิดพลาดของซีพียู (watchdog timer)
- ระบบการขัดจังหวะจากภายในและภายนอก (internal and external interrupt)

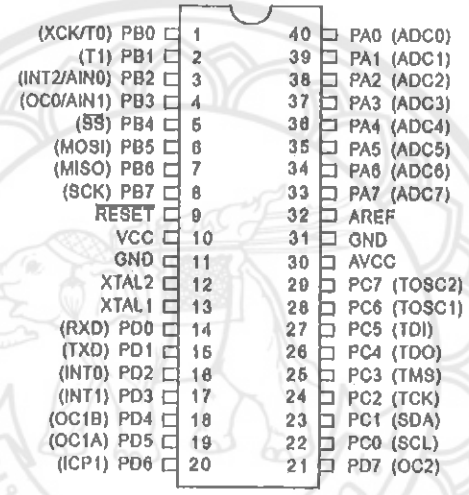
- โหมดการอนุรักษ์พลังงาน 6 โหมดได้แก่ idle, ADC noise reduction, power save, power down, standby และ extended standby

-ระบบการรีเซ็ตแบบอัตโนมัติเมื่อเริ่มจ่ายกระแสไฟฟ้าเข้าไมโครคอนโทรลเลอร์ (power on reset)

- ศึกษารายละเอียดเพิ่มเติมหรือ www.atmel.com

2.2.2 รายละเอียดขาสัญญาณไมโครคอนโทรลเลอร์ ATmega16

มีการจัดเรียงขาดังรูปด้านล่าง



รูปที่ 2.6 การจัดเรียงขาของไมโครคอนโทรลเลอร์ ATmega16

ไฟเลี้ยงและแรงดันอ้างอิง

- VCC เป็นขาจ่ายไฟให้กับตัวไมโครคอนโทรลเลอร์

- GND เป็นขากราวด์

- AREF เป็นขาแรงดันอ้างอิงที่ใช้งานในส่วนของวงจรแปลงสัญญาณแอนะล็อกเป็นดิจิทัลปกติจะต่อกับขา VCC

- AVCC ใช้จ่ายไฟให้กับวงจรแปลงสัญญาณแอนะล็อกเป็นดิจิทัลมักต่อเข้ากับขา VCC

Port A (PA0..PA7)

- ทำหน้าที่เป็นพอร์ตรับข้อมูลของสัญญาณแอนะล็อกเพื่อแปลงเป็นสัญญาณดิจิทัล

- เป็นพอร์ต 2 ทิศทางขนาด 8 บิตโดยสามารถกำหนดให้แต่ละขาของพอร์ตต่อกับ pull-up resistor ภายในซึ่งแยกจากกันเพื่อดึงแรงดันของลอจิก 1 ให้เท่ากับ 5 โวลต์สามารถรับกระแส sink 20 มิลลิแอมแปร์ (mA)

Port B (PB0..PB7)

- เป็นพอร์ต 2 ทิศทางขนาด 8 บิตโดยสามารถกำหนดให้แต่ละขาของพอร์ตต่อกับ pull-up resistor ภายในซึ่งแยกจากกันสามารถรับกระแส sink 20 mA

- สามารถใช้งานพิเศษตามความต้องการของATmega 16

Port C (PC0..PC7)

- เป็นพอร์ต 2 ทิศทางขนาด 8 บิตโดยสามารถกำหนดให้แต่ละขาของพอร์ตต่อกับ pull-up resistor ภายในซึ่งแยกจากกันสามารถรับกระแส sink 20 mA

- สามารถทำหน้าที่เป็น JTAG Interface ได้

Port D (PD0..PD7)

- เป็นพอร์ต 2 ทิศทางขนาด 8 บิตโดยสามารถกำหนดให้แต่ละขาของพอร์ตต่อกับ pull-up resistor ภายในซึ่งแยกจากกันสามารถรับกระแส sink 20 mA

- สามารถใช้งานพิเศษตามความต้องการของATmega 16

กลุ่มขาสาหรับติดต่อกับอุปกรณ์ภายนอกแบบอนุกรม (SPI – Serial Peripheral Interface pin group)

- SCK (SPI Bus Serial Clock) เป็นสัญญาณนาฬิกาสำหรับ SPI

- MISO (SPI Bus Master Input/Slave Output) เป็นสัญญาณอินพุตสำหรับ SPI

- MOSI (SPI Bus Master Output/Slave Input) เป็นสัญญาณเอาต์พุตสำหรับ SPI

- SS (SPI Slave Select Input) ใช้เลือกสัญญาณของ SPI ในกรณีที่มีหลาย slave

กลุ่มขาเกี่ยวกับ JTAG หรือ (IEEE 1149.1)

- TDI (JTAG Test Data In)

- TDO (JTAG Test Data Out)

- TMS (JTAG Test Mode Select)

- TCK (JTAG Test Clock) ขาเหล่านี้ใช้สำหรับตรวจสอบ (debug) วงจรและสามารถใช้ในการโปรแกรมชิปได้

กลุ่มขาเกี่ยวกับตัวจับเวลาและตัวนับ (timer –counter pin group)

- T0 (Timer/Counter0 External Counter Input)

- T1 (Timer/Counter1 External Counter Input)

นอกจากนี้หากไม่ต้องการใช้สัญญาณนาฬิกาจากตัวไมโครคอนโทรลเลอร์เรายังสามารถป้อนสัญญาณนาฬิกาภายนอกผ่านทางขาต่อไปนี้ได้

- TOSC1 (Timer Oscillator Pin 1)

- TOSC2 (Timer Oscillator Pin 2)

กลุ่มขาสำหรับรับสัญญาณขัดจังหวะจากภายนอก (external interrupt pin group)

- INT0 (External Interrupt 0 Input)

- INT1 (External Interrupt 1 Input)

- INT2 (External Interrupt 2 Input) การเปลี่ยนแปลงระดับแรงดันที่ขาเหล่านี้จะมีผลทำให้โปรแกรมกระโดดไปทำงานในส่วนของชุดคำสั่งให้บริการการขัดจังหวะ (interrupt service routine)

กลุ่มขาสำหรับติดต่อกับพอร์ตอนุกรม (USART pin group)

-TXD (USART Output Pin) ขาส่งข้อมูลออก

- RXD (USART Input Pin) ขารับข้อมูลเข้า

- XCK (USART External Clock Input/Output) ขาสัญญาณนาฬิกาใช้สำหรับโหมด synchronous

2.2.3 ภาษาที่ใช้ในการเขียนโปรแกรมการสั่งงานในไมโครคอนโทรลเลอร์

Codevisionคือ สภาพแวดล้อมและเครื่องมือที่ช่วยในการพัฒนาการเขียนโปรแกรมด้วยภาษาซีเพื่อช่วยให้การพัฒนางานด้านเขียนโค๊ดเพื่อ Burn หรืออัดลงบนไมโครคอนโทรลเลอร์ตระกูล AVR ของบริษัท ATMEL

Codevisionเป็นโปรแกรมที่พัฒนาขึ้นโดยบริษัท HP Info Tech ซึ่งทำมาเพื่อจำหน่ายโปรแกรมสำหรับผู้ที่ต้องการพัฒนางานด้านไมโครคอนโทรลเลอร์ตระกูล AVR จะต้องเสียเงินซื้อ แต่ถ้าคุณต้องการทดลองใช้งานทางบริษัท ก็มีให้ดาวน์โหลดไปทดลองใช้งาน โดยสามารถใช้งานได้ แต่จำกัดที่ขนาดของ code size อยู่ที่ 3kb โปรแกรม codevisionเป็นโปรแกรมที่เอาไว้เขียน

ภาษาซี ซึ่งตัวโปรแกรมอ้างอิงตามมาตรฐาน ANSI C เพราะฉะนั้นจึงทำให้เราสามารถนำความรู้พื้นฐานของการใช้ภาษาซีของเราสามารถนำมาใช้กับการเขียนโค๊ดภาษาซี บน codevision ได้เลย

คุณสมบัติของโปรแกรม codevision รุ่นมาตรฐาน

- สามารถติดตั้งได้ทั้งwindows 2000, xp ,vista และ windows 7 ทั้งแบบ 32 บิต และ 64 บิต
- IDE ถูกออกแบบให้สามารถใช้งานได้ง่าย และเข้ากันได้กับการเขียนภาษาซี ตามมาตรฐาน ANSI C
- ตัวโปรแกรมสามารถย่อหน้าให้เราอัตโนมัติ ไวยกรณ์ของภาษาซี และ ASSEMBLY ถูกระบายสี เพื่อการสังเกตได้ชัดเจน
- รองรับการประกาศตัวแปร ชนิด bit, bool, char, int, short, long, float
- มีไลบรารี รองรับการทำงานกับข้อมูลที่เป็น เลขทศนิยม กับไมโครคอนโทรลเลอร์ที่รองรับคำสั่งเหล่านี้

ในตอนนี้ เราจะมาเริ่มเข้าเนื้อหากฎทั่วๆ ไปของการเขียนภาษาซี กับ CodeVisionAVR C Complier โดยเน้นเฉพาะที่ต้องใช้กับ Complier ซึ่งเราอาจจะไม่ได้ลงไปในรายละเอียดทุกๆ ส่วนของภาษาซีแต่จะเน้นเฉพาะที่ต้องใช้บ่อยๆ ในการทำงานกับไมโครคอนโทรลเลอร์เท่านั้นซึ่งรายละเอียดทั้งหมดของภาษาซี เพื่อนๆสามารถหาอ่านได้ตามหนังสือภาษาซีที่มีขายอยู่ทั่วไป

The C Preprocessor directives ยอมอนุญาต ให้คุณกระทำการได้ดังนี้

- นำไฟล์อื่นเข้ามาร่วมในการคอมไพล์ได้ โดยไฟล์นั้น อาจจะเป็นไฟล์ไลบรารี หรือ ไฟล์ที่ประกาศเฉพาะแต่ function prototype ก็ได้
- ยอมให้เรานิยามตัวแปร Macro ซึ่งช่วยลดความยุ่งยากในการเขียนโปรแกรมมิ่ง และช่วยทำให้ไล่โค๊ดได้ง่าย
- ช่วยให้การตั้งค่าการ Debug และช่วยปรับปรุงโค๊ดให้มีความสามารถโยกย้ายไป complier ค่ายอื่นๆได้ง่าย
- สามารถระบุ complier ที่ใช้ในการคอมไพล์ได้

โดย Preprocessor จะประกาศไว้ก่อน function main(void) ทุกครั้ง Preprocessor ที่สำคัญ ได้แก่

#include ทำหน้าที่รวมไฟล์อื่นๆ เข้ามาในไฟล์ซอร์สโค๊ดของคุณ เราสามารถรวมไฟล์ด้วยคำสั่ง #include ได้มากที่สุด 300 ไฟล์

#define ใช้สำหรับในการกำหนด macro (macro คือคำสั่งที่ผู้ใช้เขียนขึ้นเองหรือคำสั่งที่ประกอบด้วยคำสั่งย่อยๆ คำสั่งซึ่งแทนด้วยสัญลักษณ์เพียงสัญลักษณ์เดียว) ถ้ามีการใช้ # ร่วมกับ parameter ของมาโคร จะเป็นแปลง parameter ของมาโคร ให้กลายเป็น ตัวอักษร หากว่ามาโครของเรามีจำนวนบรรทัดมากกว่า 1 บรรทัด เราสามารถขึ้นบรรทัดใหม่ได้ โดยการใช้ \ ก่อนจบบรรทัดเก่า เราสามารถยกเลิกการใช้มาโครได้ด้วยใช้คำสั่ง #undef ALFA เป็นการยกเลิกมาโครALFA

Comments เหมือนกันกับภาษาซีCodeVisionก็มีรูปแบบการเขียนคอมเมนท์ หรือ หมายเหตุ ของโปรแกรมเพื่อเอาไว้เขียนคำอธิบายโปรแกรมหรือสิ่งที่เราต้องการสื่อสารให้ผู้ที่มาอ่านโค๊ดของเราในภายหลังหรือเอาไว้ให้เราอ่านเอง แต่ไม่ต้องการให้ Complier หรือตัวแปรภาษาโปรแกรมสนใจในส่วนนี้

Reversed Keywords หรือคำสงวน คำสงวนพวกนี้ เป็นคำที่เป็นคำสั่งพิเศษ ที่โปรแกรมต้องการใช้ห้ามให้เรานำคำเหล่านี้ ไปตั้งเป็นชื่อตัวแปร ซึ่งเราจะได้พบเห็นในตอนต่อๆไป คำสงวนเหล่านี้ ได้แก่

__eeprom	enum	static	__flash	extern	struct
__interrupt	flash	switch	__task	float	typedef
_bool	for	union	break	goto	unsigned
bit	if	void	bool	inline	volatile
case	int	while	char	interrupt	sfrw
const	long	continue	register	default	return
defined	short	do	signed	double	sizeof
eeprom	sfrb	else			

Identifiers ในการตั้งชื่อตัวแปร ตัวใหญ่ กับตัวเล็ก ให้ความหมายที่ต่างกันเราสามารถที่จะตั้งชื่อตัวแปรขึ้นต้นด้วย ตัวอักษรA...Z หรือ a.....z หรือใช้ตัว __ เครื่องหมาย underscore ในการนำหน้าชื่อตัวแปร ก็ได้

Data Types ชนิดของข้อมูลแต่ละประเภท ใช้จำนวนหน่วยความจำที่แตกต่างกัน และให้ขอบเขตของค่าข้อมูลที่แตกต่างกันดังตารางด้านล่างนี้

ตารางที่ 2.1 หน่วยความจำและขอบเขตของข้อมูลแต่ละประเภท

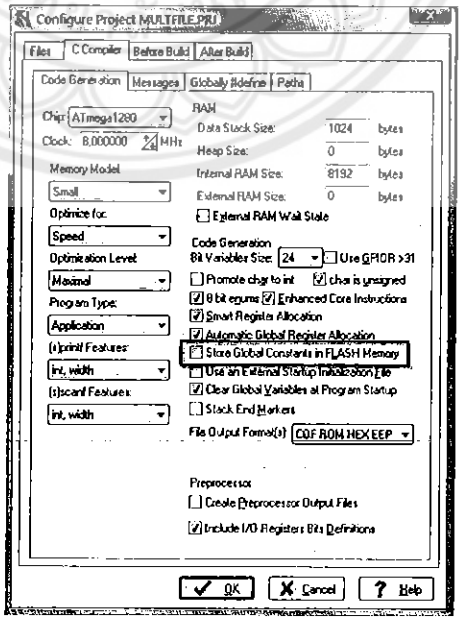
Type	Size(Bits)	Range
bit	1	0, 1
bool, _Bool	8	0, 1
char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127
int	16	-32768 to 32767
unsigned int	16	0 to 65535
signed int	16	-32768 to 32767
long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
signed long int	32	-2147483648 to 2147483647
float	32	+/- 1.175e-38 to +/-3.402e38
double	32	+/- 1.175e-38 to +/-3.402e38

Constants

- ค่าคงที่ เป็นค่าที่เราประกาศแล้ว เป็นตัวเลขอาจจะเขียนอยู่ในรูปเลขฐานสิบ (เช่น 1234)
- ถ้าต้องการเขียนให้อยู่ในรูปเลขฐานสอง เราจะเขียนได้เป็น 0b นำหน้าเลขฐานสอง เช่น0b101001
- ในเลขฐานสิบหก เราจะให้ 0x นำหน้าตัวเลขฐานสิบหก เช่น 0xff
- หรือ ถ้าเป็นเลขฐานแปด เราจะใช้ 0 นำหน้า เช่น 077
- ถ้าเราต้องการบอกให้ complier ทราบว่าเลขต่อไปนี้ เป็นข้อมูลชนิด unsigned integer เราอาจจะใส่ U ตามหลังเลข เช่น 1000U
- ถ้าเป็นข้อมูลที่เป็น Long integer เราจะใช้ L ตามหลังตัวเลข เช่น 99L

- สำหรับข้อมูลที่เป็น Unsigned long integer เราจะใช้ผสมกันระหว่าง U และ L เช่น 99UL
- จำนวนที่เป็นเลขทศนิยม Floating point เราจะใช้ F ตามหลังเลขทศนิยม เช่น 1.234F
- ค่าคงที่ ที่เป็นตัวอักขระ เราจะใช้เครื่องหมาย single quote เช่น 'a'
- แต่ถ้าเป็นค่าคงที่ ทีเป็นข้อความ หรือ string เราจะใช้ double quote เช่น "Hello World"
- เราสามารถประกาศตัวแปร constant ได้ทั้ง Global (เห็นทุกฟังก์ชัน)หรือแบบ Local (เห็นเฉพาะ ในฟังก์ชันที่ประกาศ) โดยใช้คำว่า cost นำหน้าตัวแปรที่เราประกาศ
- นอกจากนี้ เราสามารถประกาศตัวแปรค่าคงที่ ที่เป็น array ได้ด้วย โดยที่ index ตัวเลขมีลำดับเป็น 0
- แต่ถ้าเราประกาศแบบนี้ จะทำให้ 2 ลำดับแรก มีค่าเป็น 1 และ 2 ลำดับที่เหลือ จะกลายเป็นเลขศูนย์ทั้งหมดหรือจะสร้างเป็น array หลายมิติก็ได้

ถ้าหากโปรแกรม CodeVision ถูกกำหนดที่เมนู Project|Configuration|C Complier|Code Generator|Store Global| Constant in FLASH Memory ถูกกำหนดเป็น Enable แล้ว ตัวแปร global constant เมื่อถูกประกาศแล้ว (มีคำว่า constหน้าตัวแปร) จะถูกเก็บไว้ที่ FLASH Memory เมื่อถูกคอมไพล์ แต่ถ้าที่เมนู ถูกกำหนดไว้เป็น disabled ตัวแปรที่ประกาศเป็น constจะถูกเก็บไว้ที่ RAM memory และตัวแปรที่เป็น Local constant จะถูกเก็บไว้ที่ RAM memory.



รูปที่ 2.7 การตั้งค่าใช้งานโปรแกรมCodeVisionAVR

คำว่า **flash** หรือ**_flash** เมื่อนำมาใช้ในการประกาศตัวแปรที่เป็นค่าคงที่ จะเป็นการนำค่าของตัวแปรนั้นไปเก็บไว้ที่ FLASH memory ไม่ว่าเราจะไปตั้งค่าที่ Store Global Constant in FLASH memory หรือไม่ก็ตาม

รูปแบบ

flash<type definition><identifier> = constant expression;

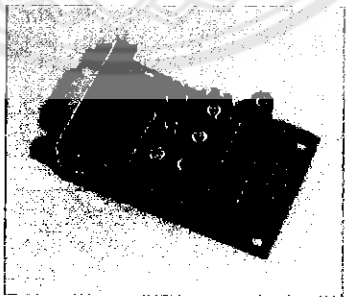
_flash <type definition><identifier> = constant expression;

ในกรณีที่เราต้องส่งค่าที่เป็น string เข้าไปในฟังก์ชันเราจะต้องส่งค่าแบบ pointer เท่านั้น ในการส่งค่าเข้าไปนี้เราสามารถที่จะสั่งให้ทำการเก็บค่าตัวแปรที่ส่งเข้าไปไว้ที่ memory โดยใช้ keyword constant ที่เราได้กล่าวไว้แล้วได้

2.3 อุปกรณ์ที่ใช้เชื่อมต่อกับไมโครคอนโทรลเลอร์[3]

เมื่อเราได้ศึกษาการทำงานของไมโครคอนโทรลเลอร์และการเขียนภาษาซีในการสั่งงานแล้วต่อไปจะมาศึกษาในส่วนขององค์ประกอบที่ใช้เชื่อมต่อในระบบการพ่นละอองน้ำโดยมี 5 ระบบ คือ ระบบส่งผ่านข้อมูล ระบบแหล่งจ่ายแรงดันไฟ ระบบแสดงผลแบบแอลซีดี ระบบตรวจจับอุณหภูมิ ระบบรีเลย์และโซลินอยด์วาล์ว

2.3.1 ระบบส่งผ่านข้อมูล



รูปที่ 2.8 MAX232 RS232 ใช้สื่อสารข้อมูลระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์

ระบบการ รับ-ส่ง ข้อมูลแบบ RS232 นี้ ถือกำเนิดขึ้นครั้งแรก ในปี ค.ศ.1969 โดย สมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์ แห่งสหรัฐอเมริกา ซึ่งข้อกำหนดของมาตรฐานระบบนี้จะครอบคลุมทั้งข้อกำหนดที่เป็นลักษณะทางกล เช่น ลักษณะของขั้วต่อสัญญาณที่ใช้ การจัดเรียงขาสัญญาณ

ต่างๆ รวมไปถึงข้อกำหนดที่เป็นคุณสมบัติทางไฟฟ้าของสัญญาณที่ใช้ในการ รับ-ส่ง ข้อมูลกันระหว่างอุปกรณ์ต้นทางกับปลายทาง เพื่อเป็นจุดอ้างอิงให้กับบริษัทผู้ผลิตต่างๆ ที่จะสร้างอุปกรณ์ที่ใช้ในการสื่อสารอนุกรม Asynchronous ได้ใช้ เป็นมาตรฐานเดียวกันในการผลิตสินค้าออกจำหน่าย โดยมาตรฐาน RS232 นี้ได้กำหนดย่านแรงดันไฟฟ้าที่จะรับ-ส่งในสายสัญญาณ เป็นดังนี้

ตารางที่ 2.2 แสดงการแทนค่าลอจิก ด้วยระดับแรงดันของระบบ RS232

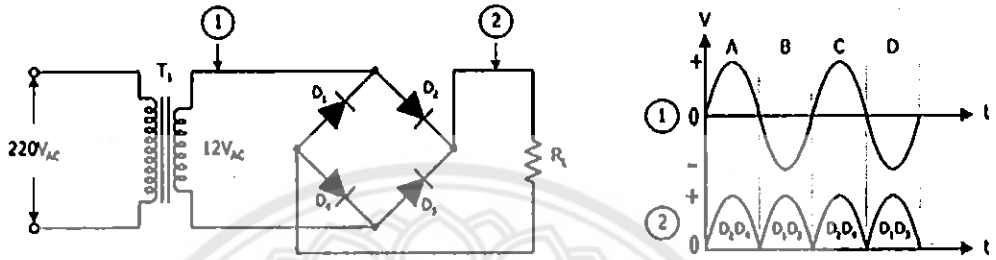
TTL Logic	ระดับ แรงดัน
"0"	+3V ถึง +15V
"1"	–3V ถึง –15V

แต่อย่างไรก็ตามเนื่องจากเครื่องคอมพิวเตอร์ PC ส่วนมากมักมีแหล่งจ่ายไฟ ซึ่งมีขนาด +5V และ +12V ซึ่งใช้ สำหรับเลี้ยงวงจรและอุปกรณ์อื่นๆ ในระบบ ดังนั้นวงจร Line Driver ของระบบ RS232 ที่มีใช้ อยู่ในคอมพิวเตอร์ PC จึงมักกำหนดไว้เป็น +12V ด้วยเสมอ โดยสัญญาณที่เป็นลอจิก "0" นั้น วงจร ภาค Line Driver จะเปลี่ยนเป็น +12V ส่วนสัญญาณลอจิก "1" นั้น Line Driver จะเปลี่ยนเป็น -12V ด้วย ดังนั้นอุปกรณ์ Line Driver แบบมาตรฐาน RS232 นั้นในด้านของภาคส่งต้องสามารถเปลี่ยนสัญญาณ Logic ให้ เป็นระดับแรงดันตามค่าที่กำหนดไว้ในตารางได้ สำหรับในส่วนของวงจรด้านภาครับเองก็ต้องสามารถตรวจจับระดับแรงดันที่รับเข้ามาแล้วเปลี่ยนกลับให้เป็นสัญญาณลอจิกได้อย่างถูกต้องด้วยเช่นกันสำหรับวงจร Linc Diver ส่วนมากมัก ใช้ชิพไอซีสำเร็จรูปกันเป็นส่วนใหญ่ เช่น ไอซี Line Driver RS232 สำเร็จรูปเบอร์ MAX232 ซึ่งทำหน้าที่เปลี่ยนสัญญาณ TTL ลอจิกให้ เป็นสัญญาณในระดับข้อกำหนดของ RS232 และยังทำ หน้าที่ แปลงสัญญาณของ RS232 ให้กลับมาเป็นสัญญาณ TTL ได้ในตัวเดียวกัน

2.3.2 ระบบแหล่งจ่ายแรงดันไฟ

การแปลงแรงดันไฟสลับเป็นแรงดันไฟตรงวงจรอิเล็กทรอนิกส์ต่าง ๆ จะต้องใช้แรงดันเลี้ยงวงจรเป็นแรงดันไฟตรง (DC) โดยทำการแปลงแรงดันไฟสลับ (AC) ให้เป็นแรงดันไฟตรง (DC) วงจรที่ทำหน้าที่ดังกล่าวนี้เรียกว่าวงจรเร็กติไฟเออร์ (Rectifier Circuit) หรืออาจเรียกว่าวงจรเรียงกระแส อุปกรณ์ที่ทำหน้าที่นี้คือไดโอด ไดโอดที่นิยมนำมาใช้งานในวงจรเร็กติไฟเออร์เป็นไดโอดชนิดซิลิกอน

เร็กติไฟเออร์เต็มคลื่นแบบ บริดจ์ คือวงจรเร็กติไฟเออร์แบบเต็มคลื่นนั่นเอง เพียงแต่การจัดวงจรเร็กติไฟเออร์มีความแตกต่างไปจากวงจรเร็กติไฟเออร์เต็ม คลื่นใช้หม้อแปลงมีแทปกลาง วงจรเร็กติไฟเออร์เต็มคลื่นแบบบริดจ์ประกอบด้วยหม้อแปลงใช้ชนิดทางขดทุติย ภูมิมี 2 ขั้วต่อไม่ต้องมีแทปกลาง (CT) ใช้ไดโอดในการเร็กติไฟเออร์4 ตัว การทำงานแต่ละครั้งไดโอดทำงานเป็นชุด 2 ตัว ลักษณะวงจรและแรงดันที่ได้แสดงดังรูปที่ 2.9



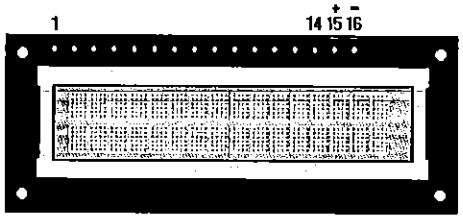
(ก) วงจร (ข) แรงดันจุดต่าง ๆ

รูปที่ 2.9 วงจรเร็กติไฟเออร์เต็มคลื่นแบบบริดจ์

จากรูปที่ 2.7 แสดงวงจรเร็กติไฟเออร์เต็มคลื่นแบบบริดจ์รูปที่ 2.7 (ก) เป็นวงจรเร็กติไฟเออร์เต็มคลื่นแบบบริดจ์ มีไดโอด D1-D4 เป็นวงจรเร็กติไฟเออร์ หม้อแปลง T1 เป็นชนิดธรรมดาไม่มีแทปกลาง (CT) วัดสัญญาณที่จุด 1 และจุด 2 ออกมาได้เหมือนกับเร็กติไฟเออร์เต็มคลื่นใช้หม้อแปลงแทปกลางทุกประการการทำงานของวงจรตามรูปที่ 2.7 อธิบายได้ดังนี้ ที่จุด 1 เมื่อมีแรงดันไฟสลับซีกบวกตำแหน่ง A ป้อนเข้ามา ไดโอด D2, D4 ได้รับไบอัสตรงนำกระแส มีกระแสไหลผ่าน D2, RL และผ่าน D4 ครบวงจร ได้แรงดันตกคร่อม RL ตามจุด 2 ที่ตำแหน่ง A ส่วนไดโอด D1, D3 ได้รับไบอัสกลับไม่นำกระแสเมื่อมีแรงดันไฟสลับซีกลบตำแหน่ง B ของจุด 1 ป้อนเข้ามาไดโอด D1, D3 ได้รับไบอัสตรงนำกระแสมีกระแสไหลผ่าน D3, RL และผ่าน D1 ครบวงจร ได้แรงดันตกคร่อม RL ตามจุด 2 ที่ตำแหน่ง B ส่วนไดโอด D2, D4 ได้รับไบอัสกลับไม่นำกระแสเมื่อมีแรงดันไฟสลับซีกบวกตำแหน่ง C ของจุด 1 ป้อนเข้ามาอีกครั้ง ไดโอด D2, D4 ได้รับไบอัสตรงนำกระแส เป็นการทำงานเหมือนกับที่ตำแหน่ง A ทุกประการ ได้แรงดันตกคร่อม RL ตามจุด 2 ที่ตำแหน่ง C และเมื่อมีแรงดันไฟสลับซีกลบตำแหน่ง D ของจุด 1 ป้อนเข้ามาอีกครั้ง ไดโอด D1, D3 ได้รับไบอัสตรงนำกระแส เป็นการทำงานซ้ำเหมือนกับที่ตำแหน่ง B ทุกประการ ได้แรงดันตกคร่อม RL ตามจุด 2 ที่ตำแหน่ง D

2.3.3 ระบบแสดงผลแบบแอลซีดี

การใช้งาน LCD โมดูล



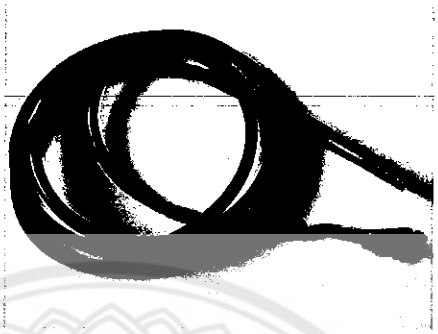
รูปที่ 2.10LCD 16x2 Line

ตารางที่ 2.3 ตำแหน่งของขาและหน้าที่การใช้งานของ LCD โมดูล

Pin No.	Symbol	Description	Level	Function	
1	VSS	Ground	-	0V	Ground
2	VDD	Power Supply	-	+5V	ต่อกับแรงดันไฟเลี้ยง +5V
3	VO	LCD Contr	-	-	ต่อกับแรงดันเพื่อปรับความเข้มของการแสดงผล
4	RS	Register Select	H/L	RS = 0 หมายถึงต้องการติดต่อกับรีจิสเตอร์คำสั่ง (Instruction Register) RS = 1 หมายถึงต้องการติดต่อกับรีจิสเตอร์ข้อมูล (Data Register)	
5	R/W	Read/Write	H/L	R/W = 0 หมายถึงต้องการเขียนข้อมูลไปยัง LCD โมดูล R/W = 1 หมายถึงต้องการอ่านข้อมูลจาก LCD โมดูล	
6	E	Enable	H, H->L	Enable Signal	
7 - 14	DB0-DB7	Data Bus	H/L	Data Bus Line	
15	A	Back Light A	-	BackLight +5V (สำหรับรุ่นที่มี Back Light)	
16	K	Back Light K	-	Back Light 0V (สำหรับรุ่นที่มี Back Light)	

2.3.4 ระบบตรวจวัดอุณหภูมิ

ดิจิตอลเทอร์มิสเตอร์ds1820



รูปที่ 2.11 ดิจิตอลเทอร์มิสเตอร์ds1820

การวัดอุณหภูมิด้วยไอซี DS1820 Digital Thermistor

หลักการเบื้องต้นของไอซีDS1820

ไอซี DS1820 เป็นไอซีที่มีระบบการสื่อสารข้อมูลอนุกรมแบบหนึ่งสายซึ่งถือได้ว่าเป็นระบบที่มีความชาญฉลาดและใช้จำนวนสายสัญญาณเพียง 1 เส้นเท่านั้นโดยไม่ต้องมีสายสัญญาณนาฬิกามาควบคุมจังหวะการถ่ายทอดข้อมูลเหมือนกับระบบสื่อสารข้อมูลอนุกรมในแบบอื่นสายข้อมูลจะทำหน้าที่เสมือนเป็นสายสัญญาณนาฬิกาในตัวส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญญาณที่ปรากฏบนสายสัญญาณในแต่ละช่องของเวลาซึ่งเรียกว่าไทม์สล็อต (Time Slot) โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่างๆในการสื่อสารข้อมูลในแต่ละไทม์สล็อตมีการกำหนดขอบเขตไว้อย่างชัดเจนการถ่ายทอดข้อมูลจะเกิดขึ้นในแต่ละไทม์สล็อตนั้นรูปแบบการถ่ายทอดข้อมูลจะเป็นแบบอะซิงโครนัสในระดับบิตไม่มีการกำหนดความยาวของข้อมูลเป็นระดับไบต์ระบบสื่อสารแบบนี้เหมาะที่จะใช้ในการสื่อสารข้อมูลระหว่างไอซีแผงวงจรเดียวกัน

รูปที่ 2.12 ไอซี DS1820

ในปัจจุบัน มีไอซีสำหรับวัดอุณหภูมิให้เลือกใช้อยู่หลายแบบแบ่งตามรูปแบบของเอาต์พุต ได้เป็นสองประเภทคือไอซีที่ให้เอาต์พุตแบบแอนะล็อก และแบบดิจิทัล โดยทั่วไปไอซีแบบดิจิทัลจะมีวงจรประเภท ADC (Analog to Digital Converter) รวมอยู่ภายใน บางตระกูลหรือบางรุ่นสามารถโปรแกรมหรืออ่านค่ารีจิสเตอร์ภายในได้ เช่นเพื่อกำหนดค่าที่เกี่ยวข้องกับการทำงานของไอซีในส่วนการเชื่อมต่อกับไมโครคอนโทรลเลอร์ มักจะเป็นการสื่อสารข้อมูลแบบ SPI หรือ I2C เพื่อประหยัดขา I/O ในการเชื่อมต่อ

การอ่านค่าอุณหภูมิจากไอซี DS1820 ซึ่งเป็นตัวแปลงอุณหภูมิ ให้เป็นค่าดิจิตอลสำหรับการอ่านค่าอุณหภูมิ ใช้สายสัญญาณเพียง 1 เส้น (1-Wire?) เท่านั้น ถ้ารวม VCC, GND เข้าไปด้วยก็จะมีขาใช้งานเพียง 3 ขาเท่านั้นรูปร่างหน้าตาก็จะคล้ายกับทรานซิสเตอร์ตัวถัง TO-92 จริงๆแล้วมันไม่ใช่แค่เซนเซอร์อุณหภูมิเท่านั้น แต่ยังมี LASERED ROM ขนาด 64 บิตโดย DS1820 ในแต่ละตัวจะมีค่า LASERED ROM ไม่ซ้ำกัน ทำให้สามารถต่อ DS1820 (1-Wire?) หลายๆตัวในสายสัญญาณเส้นเดียวกันได้ ย่านการวัดอุณหภูมิจะอยู่ในช่วง -55 C ถึง +125 C ยังไม่หมดแค่นั้นครับ ยังมี Alarm Triger TH, TL ไว้คอยเตือนเราด้วยว่าอุณหภูมิเกิด เกิด range ที่เราต้องการควบคุม

ข้อมูลเชิงเทคนิคเกี่ยวกับไอซี DS1820

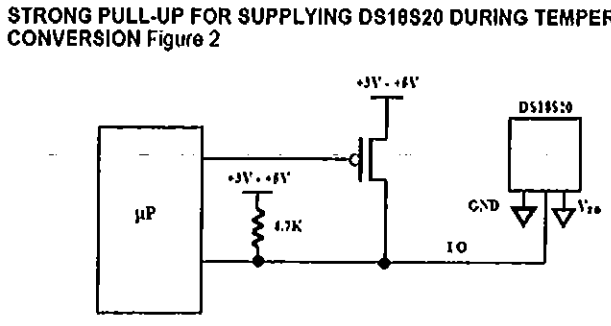
- ใช้แรงดันไฟเลี้ยง Vdd (หรือ Vcc) ได้ในช่วง 3.0V ถึง 5.5V
- มี 3 ขา (สำหรับตัวถัง TO-92) คือGnd (Pin 1), DQ (Pin 2), Vdd (Pin 3)
- ใช้งานได้สองแบบ: normal mode (ใช้ทั้ง 3 ขา)และ parasite Power mode (ใช้เพียง 2 ขา คือ DQ และ GND ในขณะที่ขา Vddจะต่อกับขา Gnd)
- สามารถนำไอซีมาพ่วงต่อกันในบัสเดียว (เส้นสัญญาณ DQ) ได้หลายอุปกรณ์

- ในการใช้งาน จะต้องต่อ pull-up 4.7kΩ (หรือน้อยกว่าได้เล็กน้อย) ที่ขา DQ กับแรงดันไฟเลี้ยง
- วัดอุณหภูมิได้ในช่วง -55 °C ถึง +125 °C
- มีความแม่นยำ ±0.5 °C สำหรับอุณหภูมิในช่วง -10°C ถึง +85°C
- มีความละเอียดของค่าที่อ่านได้ 12 บิต (Resolution)
- ใช้เวลาในการแปลงข้อมูลสำหรับ ADC ไม่เกิน 750 msec (มิลลิวินาที) สำหรับข้อมูล 12 บิต
- ไอซีแต่ละตัวมีหมายเลขเฉพาะตัว ขนาด 64 บิต (64-bit serial code)
- สำหรับตระกูล DS18B20 มีค่าไบต์สำหรับ 8-bit family code ตรงกับ 28h (0x28) เป็นไบต์แรกของหมายเลขอุปกรณ์

ภายในไอซี DS18B20 มีหน่วยความจำแบบ SRAM ขนาดความจุ 9 ไบต์ (Byte 0 ถึง Byte 9) และเรียกว่า Scratchpad ส่วนหนึ่งของหน่วยความจำนี้จะใช้สำหรับเก็บค่าอุณหภูมิที่ได้จากการอ่านและแปลงเป็นข้อมูลดิจิทัลในแต่ละครั้ง (ใช้ 2 ไบต์ และเก็บไว้ใน Byte 0 และ Byte 1) และยังมีการคำนวณค่า CRC (checksum) ขนาดหนึ่งไบต์ด้วย (เก็บไว้ใน Byte 8)

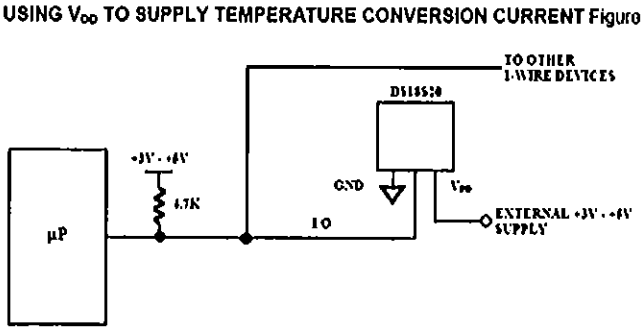
ในการต่อใช้งานจะมีอยู่ 2 วิธีด้วยกัน

1) ใช้ไฟเลี้ยงจาก R Pull-up (PARASITE POWER) วิธีนี้ขา VDD จะต้องต่อลง GND ทำให้ต่อสายเพียง 2 เส้นเท่านั้น



รูปที่ 2.13 การต่อแบบใช้ไฟเลี้ยง R Pull-up

2) ต่อไฟเลี้ยงให้กับขา VDD (External power supply) วิธีนี้จะเป็นที่นิยมใช้กันมากกว่า



รูปที่ 2.14 การต่อแบบจ่ายไฟเลี้ยงให้กับขา VDD

ค่าอุณหภูมิที่อ่านได้จาก DS1820 จะมีความละเอียดสเตปละ 0.5 C ขนาด 9 บิต

ตารางที่ 2.4 ความสัมพันธ์ระหว่างอุณหภูมิกับค่าที่อ่านได้

2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	LSB

MSb (unit = °C) LSb

S	S	S	S	S	S	S	S	MSB

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+85°C	0000 0101 0101 0000	0550h*
+125°C	0000 0000 1111 1010	00FAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55°C	1111 1111 1001 0010	FF92h

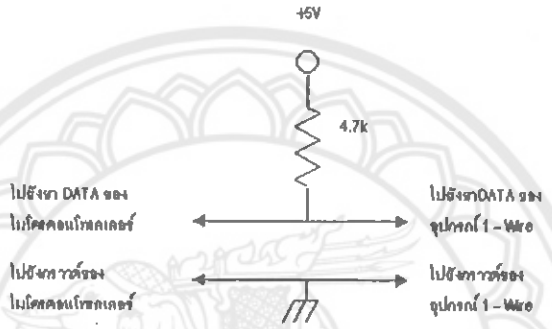
*The power on reset register value is +85°C

การอินเตอร์เฟสผ่านสายเส้นเดียว

การเชื่อมต่อหรือการอินเตอร์เฟสระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอกโดยใช้จำนวนสายสัญญาณให้น้อยที่สุดได้มีการพัฒนามาอย่างต่อเนื่องจากหลายบริษัทผู้ผลิตเช่นการเชื่อมต่ออุปกรณ์ต่อพ่วงแบบอนุกรม (Serial Peripheral Interface, SPI) ในไมโคร-คอนโทรลเลอร์ 68HC1 ของโมโตโรลาการเชื่อมต่อแบบ SPI นี้ช่วยให้ไมโครคอนโทรลเลอร์แลกเปลี่ยนข้อมูลกับ

อุปกรณ์ต่อพ่วงได้ด้วยความเร็วถึง 1 ล้านบิตต่อวินาทีโดยใช้สายรับส่งสัญญาณเพียง 3 หรือ 4 เส้น รวมกับสายกราวด์อีกหนึ่งเส้น

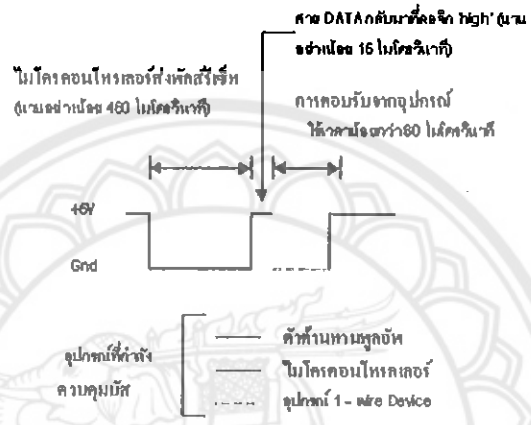
อุปกรณ์ที่สนับสนุนระบบบัสเพียงเส้นเดียวจะมีสายสัญญาณเพียง 2 เส้นเท่านั้นคือสายกราวด์และสายสัญญาณซึ่งเรียกอีกอย่างว่าสาย DATA สายนี้จะจัดการเกี่ยวกับทั้งสัญญาณข้อมูลและสัญญาณนาฬิกาที่ใช้ในการแลกเปลี่ยนข้อมูลสาย DATA นี้จะเป็นชนิด Open Drain ดังนั้นในการออกแบบวงจรจะต้องออกแบบให้มีตัวต้านทานมาพูลอัพสาย DATA นี้ด้วยให้ดูรูปแผนผังแสดงการต่อระบบบัสของ 1 – Wire Bus ดังรูป



รูปที่ 2.15 แผนผังของระบบบัสแบบ 1- Wire Bus

จากการแลกเปลี่ยนข้อมูลระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ที่ใช้ 1 – Wire Bus นี้ไม่ได้ง่ายๆเหมือนกับการส่งข้อมูลผ่านทางบัสแบบ SPI เพราะในระบบ 1 – Wire Bus นั้นสาย DATA จะต้องจัดการเกี่ยวกับจังหวะเวลาระดับสัญญาณและทิศทางของข้อมูลทั้งหมดทำให้การเขียนซอฟท์แวร์ของไมโครคอนโทรลเลอร์ที่ติดต่อกับอุปกรณ์พวกนี้ต้องมีความซับซ้อนมากขึ้น ในสภาวะพักอุปกรณ์ที่ใช้บัสแบบ 1 – Wire Bus จะทำให้สาย DATA อยู่ในสภาวะลอยทำให้ขานี้มีแรงดันเท่ากับแรงดันพูลอัพซึ่งปกติก็คือ 5 โวลต์นั้นเองส่วนไมโครคอนโทรลเลอร์ก็จะปล่อยให้ขาเอาต์พุตที่ใช้ติดต่อกับขา DATA นี้อยู่ในสภาวะความต้านทานสูง (High-Impedance State) เช่นกันเมื่ออุปกรณ์ทั้งสองชนิดนี้ปล่อยให้ขา DATA อยู่ในสภาวะลอยแล้วนี้ความต้านทานพูลอัพก็จะช่วยรักษาระดับแรงดันไฟเลี้ยงที่จ่ายให้กับอุปกรณ์ที่ใช้บัสแบบ 1 – Wire Bus นี้ได้อย่างสม่ำเสมอเพราะอุปกรณ์ที่ใช้บัสแบบ 1 – Wire Bus นี้จะใช้พลังงานในการทำงานน้อยมากในการแลกเปลี่ยนข้อมูลกันไมโครคอนโทรลเลอร์และอุปกรณ์ที่ใช้บัสแบบ 1 – Wire Bus จะต้องดำเนินการอย่างระมัดระวังตามลำดับขั้นตอนในการทำให้สาย DATA มีลอจิกเป็น 'low' ปล่อยให้สาย DATA กลับมามีลอจิกเป็น 'high' และตรวจจับการตอบรับกับอุปกรณ์อีกด้านหนึ่งช่วงจังหวะ

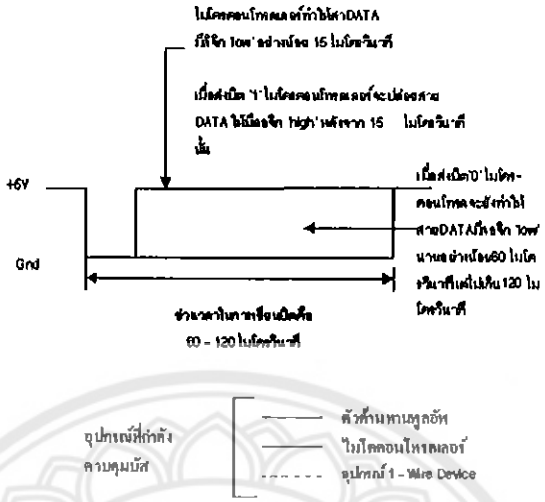
เวลาที่ใช้ในกระบวนการนี้จะถูกกำหนดโดยข้อกำหนดเฉพาะของระบบ 1 –Wire Bus นี้ซึ่งต้องใช้ไมโครคอนโทรลเลอร์ที่มีการตอบสนองได้อย่างรวดเร็วด้วยดังนั้นเราจะต้องตรวจสอบความสามารถของระบบให้ดีเสียก่อนที่จะใช้ระบบบัสแบบ 1 – Wire Bus นี้จากสภาวะพักข้างต้นการแลกเปลี่ยนข้อมูลจะเริ่มขึ้นด้วยการที่ไมโครคอนโทรลเลอร์กระทำกระบวนการรีเซ็ตซึ่งทำได้โดยการทำให้สาย DATA มีลอจิก'low' เป็นเวลาอย่างน้อย 480 ไมโครวินาทีแล้วจึงปล่อยให้กลับมาอยู่ในสภาวะ 'high' อีกครั้งหนึ่งดังรูป



รูปที่ 2.16 จังหวะเวลาในการทำกระบวนการตรวจสอบว่ามีอุปกรณ์อยู่บนบัส

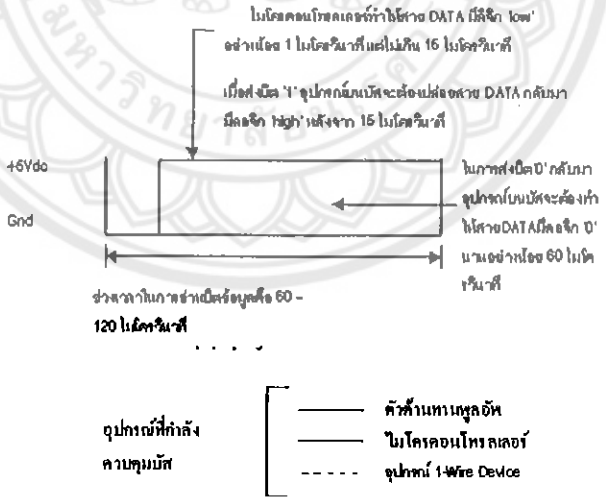
เมื่ออุปกรณ์ที่ใช้บัสแบบ 1 – Wire Bus นี้ตรวจพบสภาวะ RESET นี้มันก็จะตอบสนองด้วยการส่งพัลส์กลับไปเพื่อบอกให้ไมโครคอนโทรลเลอร์รู้ว่าบนสายบัสนี้มีอุปกรณ์แบบ 1 – Wire Device กำลังทำงานอยู่โดยอุปกรณ์แบบ 1 – Wire Device จะปล่อยให้สาย DATA อยู่ในลอจิก 'high' อย่างน้อย 15 ไมโครวินาทีแต่ไม่เกิน60 ไมโครวินาทีจากนั้นมันก็จะให้สาย DATA ลงมามีลอจิกเป็น 'low' ในช่วงเวลาประมาณ 60-240 ไมโครวินาทีแล้วจึงปล่อยให้กลับไปมีลอจิก 'high' เช่นเดิมช่วงเวลานี้มีชื่อเรียกกันหลายชื่อเช่นช่วงเวลาเริ่มติดต่อหลังจากอุปกรณ์ 1 – Wire Device ปล่อยให้สาย DATA กลับมาอยู่ในลอจิก 'high' แล้วไมโครคอนโทรลเลอร์จะต้องปล่อยให้สาย DATA อยู่ในลอจิกนี้นานอย่างน้อย 240 ไมโครวินาทีต่อไปจากนั้นไมโครคอนโทรลเลอร์ก็จะส่งคำสั่งเริ่มต้นขนาด 1 ไบต์ไปยังอุปกรณ์ 1 – Wire Bus ซึ่งคำสั่งนี้อาจจะเป็นคำสั่งอะไรก็ได้ไมโครคอนโทรลเลอร์จะส่งบิตของคำสั่งนั้นออกไปโดยการเปลี่ยนสถานะของสาย DATA กลับไปกลับมาโดยตอนแรกจะให้เป็นลอจิก 'low'แล้วจึงกลับมาเป็น 'high' ตามช่วงจังหวะเวลาที่เหมาะสมช่วงเวลานานที่ไมโครคอนโทรลเลอร์ทำให้สาย DATA มีลอจิก 'low' จะเป็นตัวแยกแยะ

ว่าบิตไหนที่มีลอจิกเป็น ‘1’ บิตไหนมีลอจิกเป็น ‘0’ ให้ดูแผนภูมิเวลาในการเขียนบิต ‘1’ หรือ ‘0’ ดังรูป



รูปที่ 2.17 จังหวะเวลาที่ไมโครคอนโทรลเลอร์ใช้เขียนข้อมูล ‘0’ หรือ ’1’ ไปยังอุปกรณ์บนบัส

ช่วงเวลาระหว่าง 15-120 ไมโครวินาทีหลังไมโครคอนโทรลเลอร์ทำให้สาย DATA มีลอจิก ‘low’ ในตอนนี้จะเรียกว่าช่วงการอ่านสถานะบิตข้อมูล (Sampling Window)



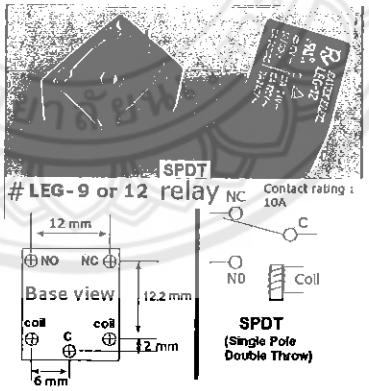
รูปที่ 2.18 จังหวะเวลาที่ไมโครคอนโทรลเลอร์ใช้อ่านบิตข้อมูล ‘0’ หรือ ’1’ จากอุปกรณ์บนบัส

ในการอ่านบิตข้อมูลจากบัสแบบ 1-Wire Bus ตอนแรกไมโครคอนโทรลเลอร์จะต้องทำให้สาย DATA มีลอจิก ‘low’ เป็นเวลานานไม่เกิน 15 ไมโครวินาทีแล้วจึงปล่อยให้สาย DATA กลับมามีลอจิก ‘high’ เช่นเดิมจากนั้นอุปกรณ์ 1 – Wire Device ก็จะเข้าควบคุมสาย DATA แทนโดยจะส่ง

บิต '0' โดยการทำให้สาย DATA มีลอจิกเป็น 'low' และส่งบิต '1' โดยการปล่อยให้สาย DATA กลับมามีลอจิก 'high' ตามเดิมเมื่อส่งข้อมูลออกไปยังไมโครคอนโทรลเลอร์เรียบร้อยแล้ว จะมีการพักอยู่ชั่วขณะหนึ่งจากนั้นอุปกรณ์ 1 – Wire Device จะปล่อยการควบคุมจากสาย DATA ให้เป็นอิสระและรอรับคำสั่งการอ่านของข้อมูลครั้งต่อไปถ้าอุปกรณ์ 1 – Wire Deviceส่งบิต '0' ออกไปไมโครคอนโทรลเลอร์ก็จะสามารถตรวจสอบจุดสิ้นสุดของลอจิก '0' นี้ง่าย เพราะสาย DATA จะกลับมาอยู่ที่ลอจิก 'high' ตามเดิมแต่ถ้าตรวจสอบจุดสิ้นสุดของการส่งบิต '1' ของอุปกรณ์ 1 – Wire Deviceจะต้องใช้เทคนิคมากกว่านี้เพราะสาย DATA จะอยู่ที่ลอจิก 'high' อยู่แล้วนี่ก็เป็นเหตุผลว่าทำไมจังหวะเวลาในการอ่านเขียนข้อมูลจึงเป็นเรื่องที่สำคัญมากเมื่อจะอ่านบิต '1' จากอุปกรณ์ 1 – Wire Device ไมโครคอนโทรลเลอร์จะต้องทำตามช่วงเวลาที่แสดงในแผนภูมิเวลาอย่างเคร่งครัดและต้องไม่ทำการอ่านลอจิกของบิตถัดมาจนกว่าจะผ่านเวลาไปแล้วอย่างน้อย 60 ไมโครวินาที

2.3.5 ระบบรีเลย์และโซลินอยด์วาล์ว

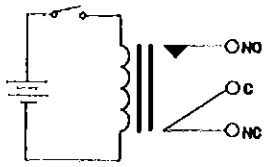
การทำงานของรีเลย์

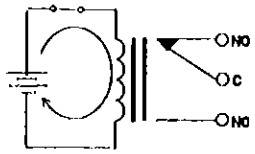


รูปที่ 2.19 Relay 12VDC SPDT

รีเลย์เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการตัด-ต่อวงจรคล้ายกับสวิตช์โดยทั่วไปจะเป็นแบบ Electromagnetic Relay หรือเรียกว่าแบบหน้าสัมผัสประกอบด้วยชุดหน้าสัมผัส (Contacts) ที่ต่อกับแท่งอาร์เมเจอร์ (Armature) และคอยล์ (Coil) ที่ถูกพันด้วยขดลวด เมื่อมีการจ่ายแรงดันไฟฟ้าให้กับคอยล์ (Energize) จะทำให้เกิดสนามแม่เหล็กแท่งอาร์เมเจอร์ที่ต่อกับหน้าสัมผัสจะถูกดูดทำ

ให้หน้าสัมผัสเปลี่ยนการเชื่อมต่อเป็นตรงกันข้ามกล่าวคือ ปกติเปิด (NO-Normally Open) เป็นปิด หรือปกติปิด (NC-Normally Closed) เป็นเปิดและเมื่อตัดไฟที่จ่ายให้คอยล์ (Deenergize) จะทำให้รีเลย์กลับสู่สถานะปกติ กล่าวคือ หน้าสัมผัสต่างๆจะกลับสู่สภาวะแรกก่อนการจ่ายไฟด้วยแรงจากสปริง





รูปที่ 2.20 เมื่อไม่ใส่กระแสไฟจะเป็นวงจรเปิด **รูปที่ 2.21** เมื่อใส่กระแสไฟจะเป็นวงจรปิด

ส่วนประกอบของ RELAY

รีเลย์เป็นอุปกรณ์ที่ทำหน้าที่เป็นสวิตช์มีส่วนประกอบที่สำคัญดังนี้ คือ

1) คอล์ยแม่เหล็ก (Magnetic coil)
2) หน้าสัมผัส (Contact)
3) แกนเหล็ก (Armature)
4) สปริงดันอาร์เมเจอร์ และสปริงดันหน้าสัมผัส (Spring)
5) โครงยึดอุปกรณ์ (Mounting)

ยังมีรีเลย์อีกประเภทที่เป็นที่นิยมคือรีเลย์ที่ไม่ใช้หน้าสัมผัสที่เรียกว่าแบบ Solid State Relay ซึ่งใช้เทคโนโลยีของ Semiconductor ทำให้ไม่มีชิ้นส่วนที่เคลื่อนที่เพื่อลดเสียงรบกวนที่เกิดขึ้นจากรีเลย์แบบหน้าสัมผัสและเพิ่มประสิทธิภาพในการใช้งานระยะยาว

ปกติแล้วรีเลย์จะเป็นสวิตช์ตัดต่อวงจรที่ใหญ่กว่าวงจรที่ใช้ควบคุมเสมอตัวอย่างเช่น คอยล์ต้องการใช้เพียงระดับมิลลิแอมป์แต่หน้าสัมผัสสามารถตัดต่อวงจรได้ถึงระดับสิบแอมป์ เป็นต้น สำหรับรีเลย์ที่สามารถควบคุมวงจรที่มีขนาดกระแสสูงๆ จะถูกเรียกว่า คอนแทคเตอร์ (Contactor)

Resistive Load คือ Load ที่เปลี่ยนกระแสให้เป็นรูปแบบอื่นๆของพลังงานโดยจะมีตัวต้านทานไฟฟ้า (Resistor) เป็นส่วนประกอบสำคัญตัวอย่างของ Resistive Load ได้แก่ ฮีทเตอร์ไฟฟ้า หลอดไฟ วงจรที่ประกอบด้วย Resistive Load อย่างเดียวจะมีค่าPower Factor เท่ากับ 1.0

Inductive Load คือ Load ที่ใช้หลักการของสนามแม่เหล็กโดยมีคอยล์ที่ถูกพันด้วยสายไฟเป็นส่วนประกอบ ตัวอย่างของ Inductive Load ได้แก่ มอร์เตอร์ โซลินอล์ยต่างๆรีเลย์ บัลลาสต์ วงจรที่มี Inductive Load เป็นส่วนประกอบจะมีค่า Power Factor น้อยกว่า 1.0 ยิ่งมี Inductive Load มาก ค่า Power Factor ยิ่งน้อยมาก

Contact Capacity กระแสที่หน้าคอนแทคทนได้ที่ความต่างศักย์นั้นๆถ้านำไปใช้กับความต่างศักย์ที่ต่ำกว่าจะใช้ได้ไม่มีปัญหาแต่ไม่เหมาะกับการนำไปใช้กับความต่างศักย์ที่สูงกว่าเนื่องจากกระแสที่ทนได้จะน้อยกว่าที่กำหนดContact Capacity ปกติจะระบุกับการงานใช้กับ Resistive Load หรือระบบที่มีค่า Power Factor เท่ากับ 1.0 แต่หากนำไปใช้กับ Inductive Load หรือระบบที่มีค่า Power Factor น้อยกว่า 1.0 จะทำให้ค่าของ Contact Capacity ลดลง ซึ่งก็คือกระแสที่หน้าคอนแทคทนได้จะลดลงเมื่อนำไปใช้กับ Inductive Load

Protection Module เมื่อหยุดจ่ายแรงดันไฟฟ้ากับกับคอยล์ (Deenergize) สนามแม่เหล็กที่ลดลงกระทันหันจะเหนี่ยวนำให้เกิดแรงดันไฟฟ้าสูงชั่วขณะ (Voltage Spike) อาจจะมีค่าสูงถึงหลายพันโวลต์และความถี่หลายเมกกะเฮิร์ตซึ่งจะสามารถทำลายTransistorหรือ ICs ของอุปกรณ์อิเล็กทรอนิกส์อื่นๆที่ต่ออยู่ในระบบเดียวกับคอยล์ได้ดังนั้น หากระบบมีการใช้อุปกรณ์อิเล็กทรอนิกส์ จึงควรใช้ Protection Module หรือที่เรียกว่า Coil Transient Suppressionเพื่อลดการเกิด Voltage Spike ดังกล่าวที่นิยมใช้มี 2 ชนิด ได้แก่

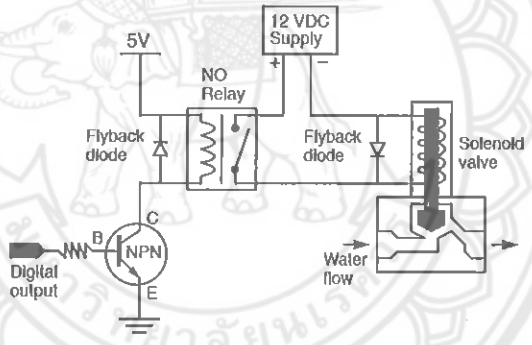
1) Diode Circuit Module: นิยมใช้กับ DC Coil โดยจะเพิ่มระยะเวลาของการ Deenergizeทำให้สนามแม่เหล็กค่อยๆลดลง จึงไม่เกิด Voltage Spike

2) Varistor Circuit Module: นิยมใช้กับ AC Coil ใช้หลักการของการจำกัด Voltage Spike และเพิ่มระยะเวลาในการ Deenergize เล็กน้อย

Metal Maintaining Clamp สำหรับการใช้งานในระบบที่มีอุณหภูมิหรือความชื้นสูงหรือระบบที่มีการสั่นสะเทือน Metal Maintaining Clamp ที่ทำจากสแตนเลสจะยึด Relay ไว้กับ Socket เพื่อคงรูปทรงและรักษาการเชื่อมต่อระหว่างกันไว้

การทำงานของโซลินอยด์วาล์ว

โซลินอยด์ (Solenoid) เป็นอุปกรณ์แม่เหล็กไฟฟ้าชนิดหนึ่งที่มีหลักการทำงานคล้ายกับรีเลย์(Relay) ภายในโครงสร้างของโซลินอยด์จะประกอบด้วยขดลวดที่พันอยู่รอบแท่งเหล็กที่ภายในประกอบด้วยแม่เหล็กชุดบนกับชุดล่างเมื่อมีกระแสไฟฟ้าไหลผ่านขดลวดที่พันรอบแท่งเหล็กทำให้แท่งเหล็กชุดล่างมีอำนาจแม่เหล็กดึงแท่งเหล็กชุดบนลงมาสัมผัสกันทำให้ครบวงจรทำงานเมื่อวงจรถูกตัดกระแสไฟฟ้าทำให้แท่งเหล็กส่วนล่างหมดอำนาจแม่เหล็กสปริงก็จะดันแท่งเหล็กส่วนบนกลับสู่ตำแหน่งปกติจากหลักการดังกล่าวของโซลินอยด์ก็จะนำมาใช้ในการเลื่อนลิ้นวาล์วของระบบนิวแมติกส์ไฟฟ้าโครงสร้างของSolenoid Valveโดยทั่วไปแบ่งออกเป็น 2 ชนิดคือ เลื่อนวาล์วด้วยโซลินอยด์วาล์วกลับด้วยสปริง (Single Solenoid Valve) และเลื่อนวาล์วด้วยโซลินอยด์วาล์วกลับด้วยโซลินอยด์วาล์ว (Double Solenoid Valve) ที่ใช้อยู่ในปัจจุบัน เช่น โซลินอยด์วาล์ว 2/2, 3/2, 4/2 , 5/2 เป็นต้น



รูปที่ 2.22การต่อใช้งานโซลินอยด์วาล์ว

โซลินอยด์วาล์ว เป็นโซลินอยด์วาล์วที่มีคุณภาพทำงานได้อย่างรวดเร็ว โซลินอยด์วาล์วใช้ไฟ 220V AC ใช้ได้กับข้อต่อเกลียวท่อขนาด3/4นิ้ว(ขนาด6หุน)เหมาะสำหรับการใช้งานกับน้ำและของเหลวความหนืดต่ำ

รูปที่ 2.23 โซลินอยด์วาล์ว ปิดเปิดน้ำ

บทที่ 3

วิธีดำเนินโครงงาน

จากบทที่ 2 ทำให้เราทราบถึงหลักการและทฤษฎีที่เกี่ยวข้องกับการดำเนินโครงงาน ซึ่งบทนี้จะพูดถึงขั้นตอนการดำเนินงานและการออกแบบระบบพ่นละอองน้ำอัตโนมัติโดยควบคุมด้วยไมโครคอนโทรลเลอร์ซึ่งจะมีเนื้อหารายละเอียดดังนี้

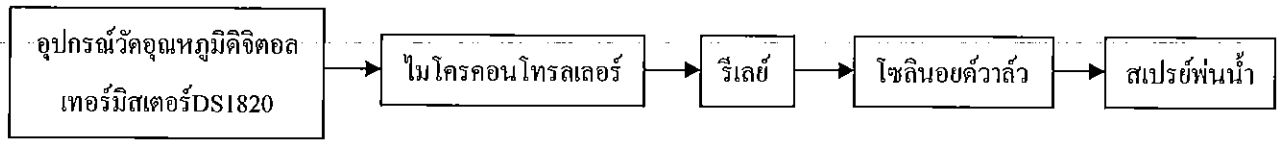
1) แผนผังการดำเนินงาน
2) ระบบชุดควบคุมการพ่นละอองน้ำอัตโนมัติ
3) โครงสร้างชุดควบคุมการพ่นละอองน้ำอัตโนมัติ
4) การเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์

3.1 แผนผังการดำเนินงาน

เพื่อเป็นการวางแผนการจัดการที่มีระบบ เราจำเป็นต้องทำการออกแบบระบบชุดควบคุมการพ่นละอองน้ำอัตโนมัติ ซึ่งจะทำให้ผู้ดำเนินโครงงานทราบถึงระบบต่างๆในชุดควบคุมความสะดวกและเกิดความรวดเร็วในการสร้างชิ้นงาน

3.1.1 การทำงานของชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

ผู้ดำเนินโครงงานได้ทำการออกแบบระบบต่างๆของชุดการพ่นละอองน้ำอัตโนมัติ ซึ่งประกอบด้วยไมโครคอนโทรลเลอร์ อุปกรณ์ตรวจจับอุณหภูมิดิจิตอลเทอร์มิสเตอร์ DS1820รีเลย์ โซลินอยด์วาล์วและสเปรย์พ่นน้ำ ดังแสดงในรูปที่ 3.1



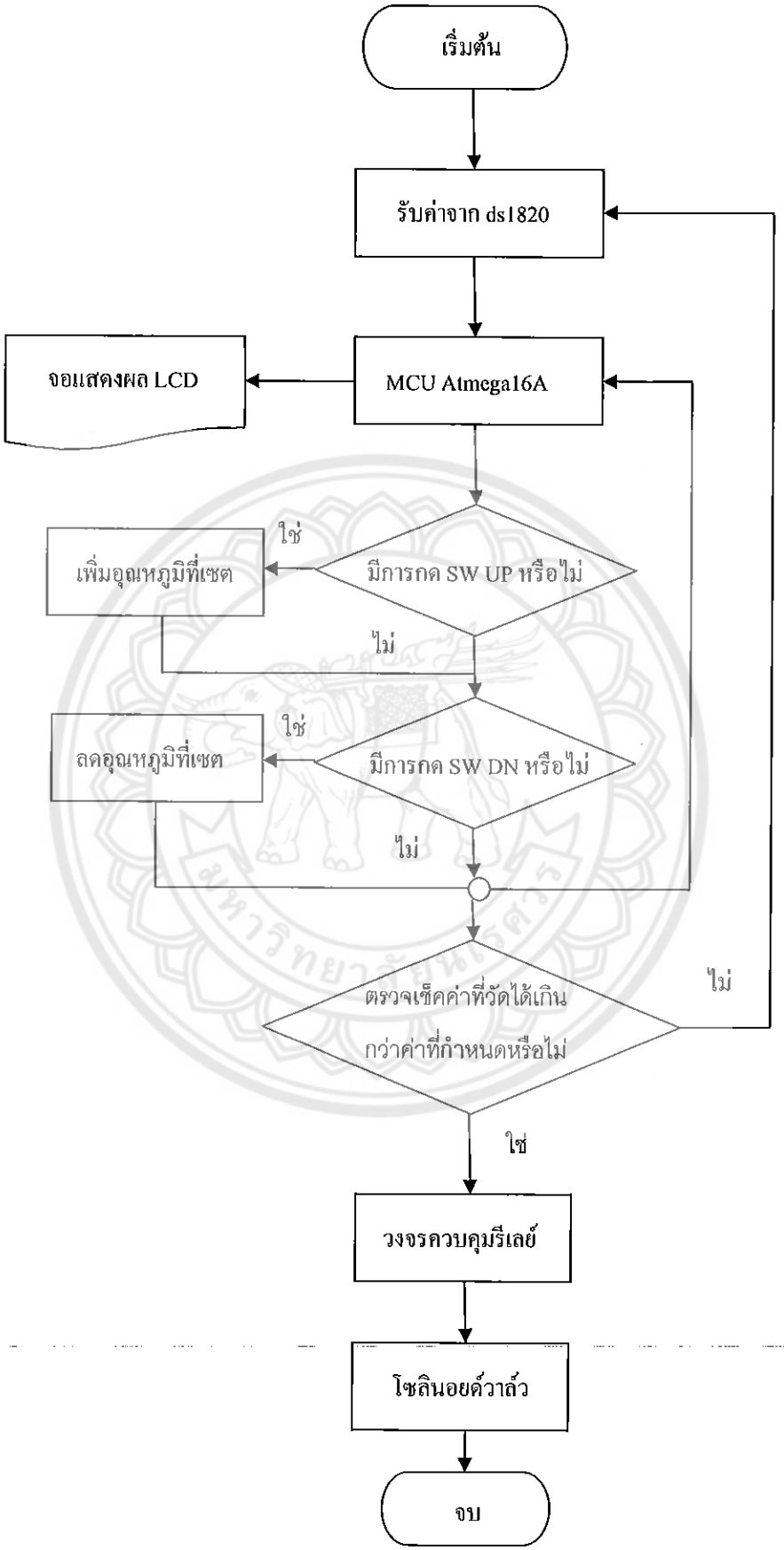
รูปที่ 3.1 แผนผังระบบชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

จากรูปที่ 3.1 เป็นการออกแบบระบบของชุดควบคุมการพ่นละอองน้ำอัตโนมัติ โดยจะทำการรับข้อมูลจากอุปกรณ์ตรวจจับอุณหภูมิดิจิตอลเทอร์มิสเตอร์ DS1820โดยไมโครคอนโทรลเลอร์จะทำการประมวลผลตามค่าอุณหภูมิที่กำหนดซึ่งจะทำให้รีเลย์ทำงานเป็นสวิตช์เปิดปิด รีเลย์จะส่งสัญญาณไปให้โซลินอยด์วาล์วทำงานซึ่งจะทำให้น้ำไหลผ่านสเปรย์พ่นน้ำจนกว่าอุณหภูมิจะลดลงระบบจึงสั่งรีเลย์หยุดทำงาน

3.1.2 หลักการทำงานชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

เมื่อได้ทำการออกแบบระบบชุดควบคุมการพ่นละอองน้ำอัตโนมัติ ในหัวข้อนี้ผู้ดำเนินโครงงานได้ทำการกำหนดเงื่อนไขในการทำงานโดยจะมีลักษณะการทำงาน ดังแสดงในรูปที่ 3.2

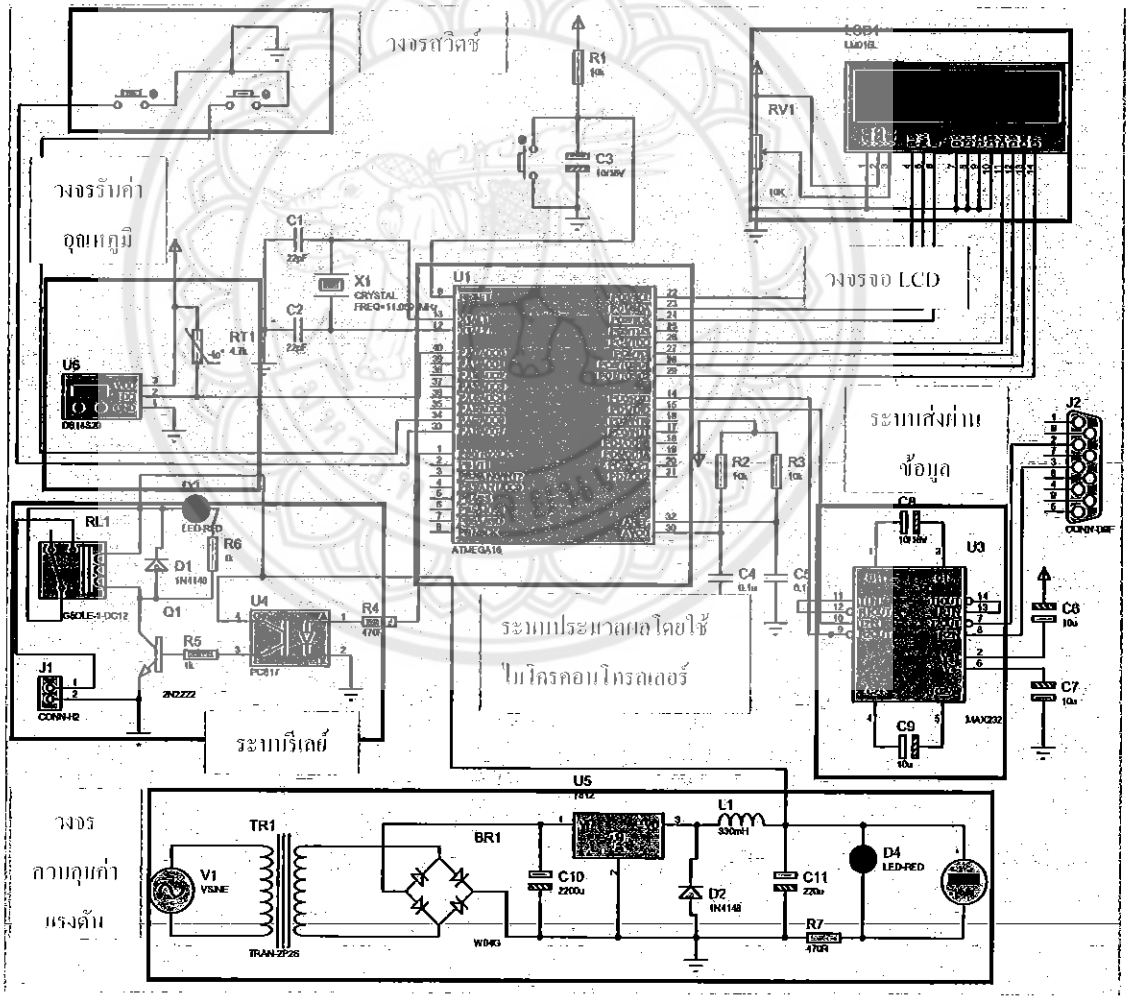
เริ่มต้นโปรแกรมไมโครคอนโทรลเลอร์รับค่าอุณหภูมิจากds1820 และนำไปแสดงผลทางจอแอลซีดีเพื่อแสดงค่าอุณหภูมิที่อ่านได้ขณะนั้นหลังจากนั้นโปรแกรมจะตรวจสอบว่ามีการกดสวิตช์เพิ่มอุณหภูมิหรือไม่ ถ้ามีการกดสวิตช์อุณหภูมิที่ตั้งค่าไว้ก็จะเพิ่มทีละ 0.5 องศาเซลเซียสต่อการกดสวิตช์หนึ่งครั้ง แต่ถ้าไม่มีการกดสวิตช์เพิ่มอุณหภูมิโปรแกรมก็จะทำงานในขั้นต่อไป คือตรวจสอบว่ามีการกดสวิตช์ลดอุณหภูมิหรือไม่ถ้ามีการกดสวิตช์อุณหภูมิที่ตั้งค่าไว้ก็จะลดทีละ 0.5 องศาเซลเซียสต่อการกดสวิตช์หนึ่งครั้ง แต่ถ้าไม่มีการกดสวิตช์ลดอุณหภูมิโปรแกรมก็จะทำงานในขั้นต่อไป โดยตรวจสอบว่าค่าอุณหภูมิที่วัดได้เกินกว่าค่าที่กำหนดหรือไม่ ถ้าใช่ก็จะสั่งงานให้วงจรควบคุมรีเลย์ทำงาน และวนกลับไปตรวจรับค่าอุณหภูมิจนกว่าค่าอุณหภูมิที่วัดได้ต่ำกว่าค่าที่กำหนดก็จะสั่งงานให้วงจรควบคุมรีเลย์หยุดทำงาน จึงจบการทำงาน



รูปที่ 3.2 แผนผังชุดควบคุมไมโครคอนโทรลเลอร์

3.2 ระบบชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

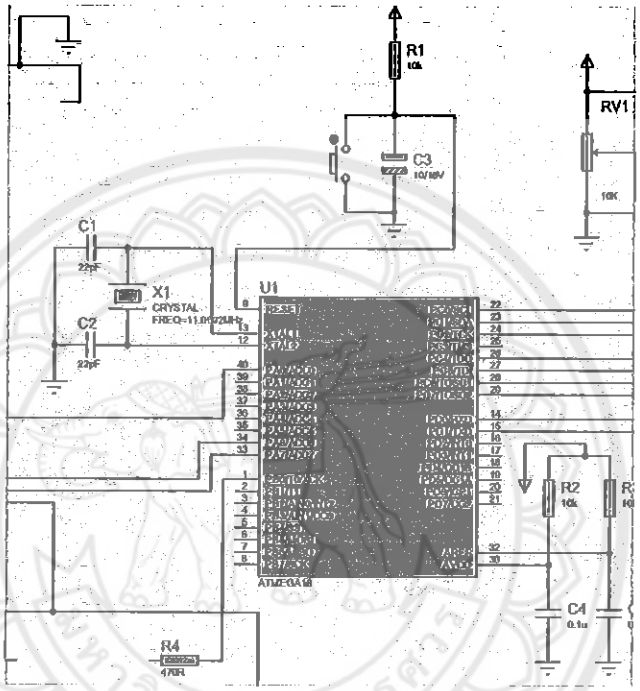
เพื่อช่วยให้การสร้างชุดควบคุมการพ่นละอองน้ำอัตโนมัติ เป็นไปอย่างถูกต้องแม่นยำและช่วยให้ประหยัดเวลาจึงได้มีการนำโปรแกรมออกแบบจำลองวงจรมาช่วยสร้างวงจรชุดควบคุม เพื่อใช้ในการเช็คว่าวงจรที่ออกแบบมานั้นมีการทำงานที่ถูกต้องหรือไม่และสามารถทดลองนำโปรแกรม codevisionavrที่ใช้ในการเขียนภาษาซีในชุดควบคุมไมโครคอนโทรลเลอร์มาอัดลงในวงจรเพื่อทดสอบการทำงานของโปรแกรมได้ด้วย โดยเราได้แบ่งการออกแบบวงจรเป็น 7 วงจร ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 ระบบชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

3.2.1 ระบบประมวลผลโดยใช้ไมโครคอนโทรลเลอร์ตระกูล AVR รุ่น ATMEGA16A

ส่วนประกอบภายในไมโครคอนโทรลเลอร์ตระกูล AVR รุ่น ATMEGA16A ประกอบด้วยขาทั้งหมด 40 ขาโดยมีพอร์ต PA-PD ซึ่งสามารถเป็นได้ทั้งขาอินพุต/เอาท์พุต โดยที่มีการใช้ค่าคริสตอลเท่ากับ 11.0592 กิโลเฮิรตซ์ แรงดันไฟเลี้ยง 5 โวลต์ ดังแสดงในรูปที่ 3.4



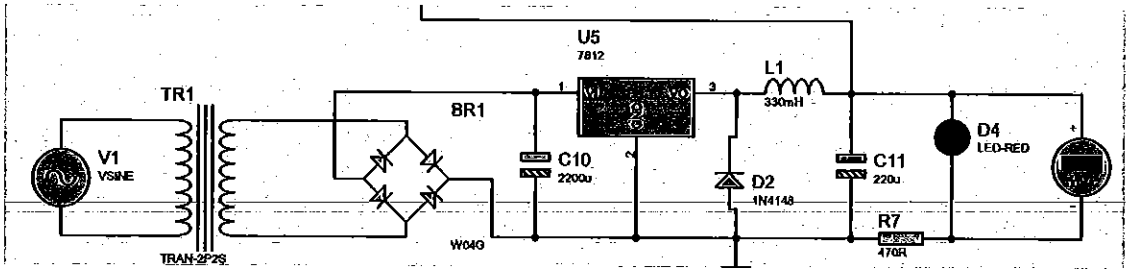
รูปที่ 3.4 ระบบประมวลผลโดยใช้ไมโครคอนโทรลเลอร์

ตำแหน่งและรายละเอียดของพอร์ตอินพุต/เอาท์พุตสำหรับติดต่อสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์ตระกูล AVR รุ่น ATMEGA16A แสดงดังในตารางที่ 3.1

ตารางที่ 3.1 พอร์ตอินพุต/เอาท์พุต ของไมโครคอนโทรลเลอร์

พอร์ต	ลักษณะการทำงาน
PA0	รับข้อมูลจากอุปกรณ์ตรวจวัดอุณหภูมิ DS1820
PA6-PA7	รับสัญญาณจากสวิตช์ SW UP/SW DN
PB0	สั่งงานออกไปให้รีเลย์ทำงาน
PC0-PC7	นำข้อมูลอุณหภูมิแสดงออกทางจอแอลซีดี
PD0-PD1	ใช้รับส่งข้อมูลในการเชื่อมต่อทางคอมพิวเตอร์

3.2.2วงจรควบคุมค่าแรงดัน

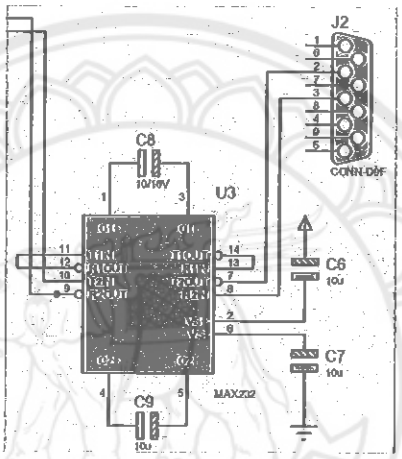


รูปที่ 3.5วงจรควบคุมค่าแรงดันสำหรับไมโครคอนโทรลเลอร์และรีเลย์

- เริ่มต้นรับแรงดันจากหม้อแปลงไฟกระแสสลับสำหรับต่อไฟเลี้ยงเข้าตั้งแต่ 7-40 โวลต์ DC หรือ AC ก็ได้เพราะมีบริดจ์ เป็นตัวแปลงกระแสไฟ
- บริดจ์ มีหน้าที่แปลงไฟจากไฟกระแสสลับให้เป็นกระแสตรงเมื่อต่อไฟ AC เข้า แต่เมื่อต่อไฟ DC เข้า บริดจ์จะมีหน้าที่กลับไฟลบให้เป็นไฟบวกหากต่อผิดขั้ว
- C10 คือตัวเก็บประจุมีหน้าที่กรองแรงดันไฟให้เรียบเมื่อผ่าน บริดจ์ในกรณีที่ต่อไฟ AC เข้า แต่ถ้าต่อไฟ DC เข้าไม่จำเป็นต้องต่อ C10 ก็ได้
- LM2575-5 โวลต์ มีหน้าที่รักษาระดับแรงดันไฟฟ้าเมื่อเข้ามา 7-40 โวลต์ จะทำให้เอาท์พุตออกมา 5 โวลต์สำหรับไมโครคอนโทรลเลอร์
- LM7812-12 โวลต์ มีหน้าที่รักษาระดับแรงดันไฟฟ้าเมื่อเข้ามา 7-40 โวลต์ จะทำให้เอาท์พุตออกมา 12 โวลต์สำหรับรีเลย์
- D2 คือไดโอดมีหน้าที่ป้องกันกระแสย้อนกลับจาก L1 ที่เป็นตัวเหนี่ยวนำ
- L1 คือตัวเหนี่ยวนำซึ่งทำหน้าที่เป็นตัวหน่วงกระแสเพื่อไม่ให้แรงดันตกขณะที่ไฟ 5 โวลต์จ่ายโหลดมากเกินความต้องการ
- C11 คือตัวเก็บประจุมีหน้าที่กรองแรงดันเอาท์พุตของ 5 โวลต์ ที่ออกมาจาก LM2575
- LED4 เป็นแอลอีดีแสดงสถานะเพื่อให้รู้ว่ามีไฟ 5 โวลต์ ออกมาจาก LM2575 หรือไม่ส่วน R7 (ตัวต้านทาน) มีหน้าที่เป็นตัวต้านทานกระแสไม่ให้กระแสไหลผ่านหลอดแอลอีดีมากเกินไป ซึ่งจะทำให้หลอดแอลอีดีขาดได้

3.2.3 ระบบส่งผ่านข้อมูล

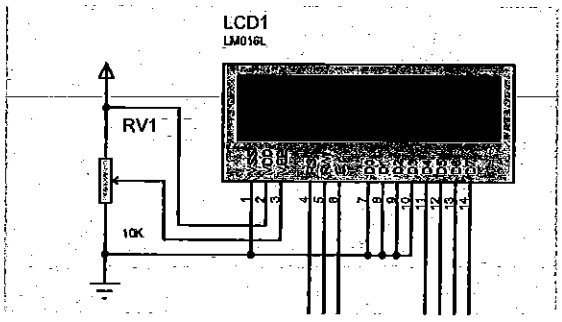
เมื่อเขียนโปรแกรมเพื่อควบคุมการทำงานด้วยภาษาซีในคอมพิวเตอร์ ขั้นตอนต่อมาคือการนำข้อมูลนั้นส่งไปที่ไมโครคอนโทรลเลอร์ โดยข้อมูลที่ส่งมาจากคอมพิวเตอร์จะมีการเรียงระดับสัญญาณแบบ RS232 ผ่านไอซี MAX 232เพื่อเรียงสัญญาณใหม่ให้เป็นระดับสัญญาณลอจิก (TTL) แล้วส่งข้อมูลเข้าไปยังไมโครคอนโทรลเลอร์ และยังสามารถแสดงผลการทำงานของไมโครคอนโทรลเลอร์ผ่านทางโปรแกรม Hyperterminal ดังแสดงในรูปที่ 3.6



รูปที่ 3.6 ระบบส่งผ่านข้อมูล RS232 ผ่านทางไอซี MAX 232

3.2.4 การเชื่อมต่อ จอแอลซีดี 16x2

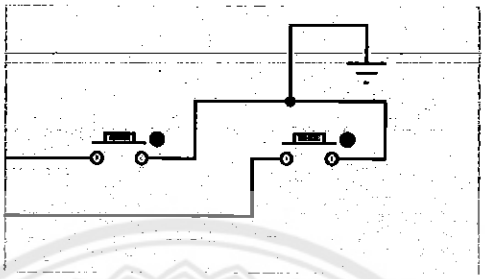
จอแอลซีดีที่นำมาใช้งานเป็นจอแอลซีดีขนาด 16x2 คือ 16 ตัวอักษร 2 บรรทัดในการต่อกับไมโครคอนโทรลเลอร์ ที่เป็นการต่อแบบ 8 บิต ดังแสดงในรูปที่ 3.7



รูปที่ 3.7 วงจรจอแอลซีดี

3.2.5 การต่อใช้งานสวิตช์

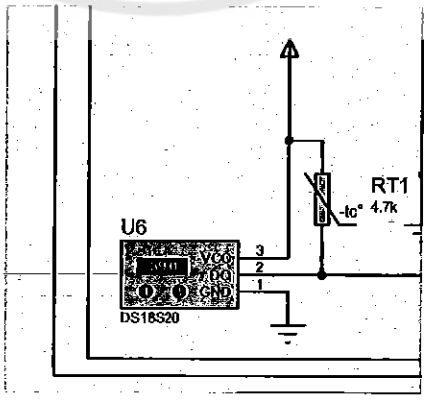
สวิตช์ เป็นสวิตช์แบบกดติดปล่อยดับ มีสวิตช์ SW UP และ SW DN หลักการคือ เมื่อมีการกด SW UP อุณหภูมิจะเพิ่มทีละ 0.5 องศาเซลเซียสต่อการกดสวิตช์หนึ่งครั้ง และเมื่อมีการกด SW DN อุณหภูมิก็จะลดทีละ 0.5 องศาเซลเซียสต่อการกดสวิตช์หนึ่งครั้งดังแสดงในรูปที่ 3.8



รูปที่ 3.8 วงจรสวิตช์

3.2.6 การต่อใช้งานไอซีวัดอุณหภูมิ DS1820

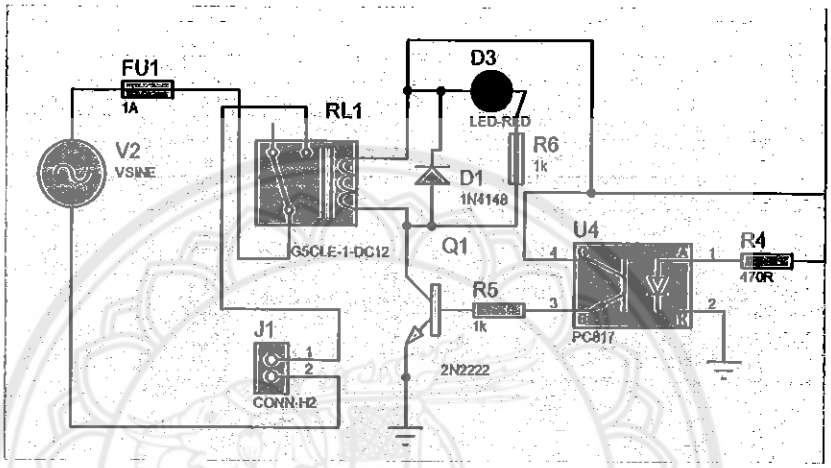
ในโครงงานนี้เราได้ใช้การตรวจจับอุณหภูมิโดยดิจิตอลเทอร์มิสเตอร์ DS1820เป็นตัวแปลงค่าอุณหภูมิให้เป็นสัญญาณดิจิตอลเข้าสู่ไมโครคอนโทรลเลอร์ซึ่งเป็นขารับส่งข้อมูลสายเดียว โดยตัวของไอซี DS1820 มีสามขาคือ ขาแหล่งจ่ายไฟ ขากราวด์ และขารับข้อมูลส่งข้อมูลไปยังไมโครคอนโทรลเลอร์ ดังแสดงในรูปที่ 3.9



รูปที่ 3.9 วงจร DS1820

3.2.7 ระบบสวิตช์เปิดปิดโดยใช้รีเลย์

เป็นส่วนควบคุมการเปิดปิดวงจรให้กับโซลินอยด์วาล์วโดยต่อผ่าน J1 ซึ่งภายในประกอบด้วยทรานซิสเตอร์ ไดโอด หลอดแอลอีดีและวงจรควบคุมแรงดันขนาด 12 โวลต์ ดังแสดงในรูปที่ 3.10

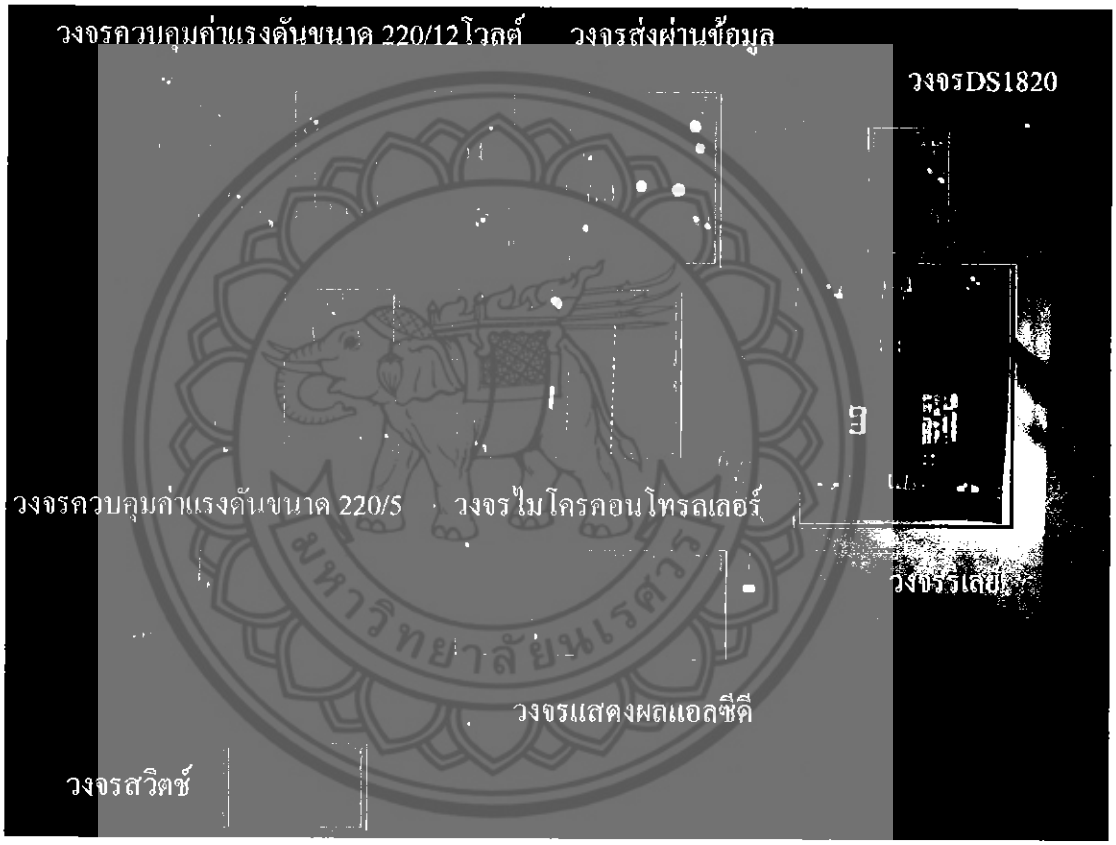


รูปที่ 3.10 วงจรสวิตช์เปิดปิดโดยใช้รีเลย์

จากรูปที่ 3.10 เป็นระบบสวิตช์เปิดปิดโดยใช้รีเลย์ โดยจะได้รับสัญญาณมาจากไมโครคอนโทรลเลอร์ซึ่งเป็นค่าแรงดันไฟเมื่อผ่านตัวต้านทาน R4 จะทำหน้าที่จำกัดกระแสที่ป้อนให้กับไอซี PC817 ซึ่งเป็นไอซีออปโต้คัปเปลอร์(Optocoupler) มีคุณสมบัติแบ่งแรงดันไฟสูงออกจากแรงดันไฟต่ำถ้าเกิดส่วนใดส่วนหนึ่งเกิดลัดวงจรจะไม่ทำให้เกิดความเสียหายกับส่วนที่เหลือ โดยแรงดันไฟที่ได้เมื่อไหลผ่านตัวต้านทาน R5 จะทำหน้าที่จำกัดกระแสที่ป้อนให้กับทรานซิสเตอร์ 2N2222 ทำหน้าที่เป็นสวิตช์ตัดต่อให้กับรีเลย์ มีไดโอด D1 ทำหน้าที่ป้องกันแรงดันไฟไหลย้อนกลับ มีตัวต้านทาน R6 ทำหน้าที่จำกัดกระแสที่ป้อนให้หลอดแอลอีดีและรีเลย์จะมีการเปลี่ยนสถานะจากหน้าสัมผัสเปิดเป็นหน้าสัมผัสปิดทำให้โซลินอยด์วาล์วทำงาน

3.3 โครงสร้างชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

ในหัวข้อนี้เป็นการแนะนำโครงสร้างชุดควบคุมการพ่นละอองน้ำอัตโนมัติ โดยมีส่วนประกอบดังนี้ วงจรไมโครคอนโทรลเลอร์ วงจรส่งผ่านข้อมูล วงจรควบคุมค่าแรงดันขนาด 220/12โวลต์และ 220/5โวลต์ วงจรแสดงผลแอลซีดี วงจรDS1820 วงจรรีเลย์ และวงจรสวิตช์ ดังแสดงในรูปที่ 3.11

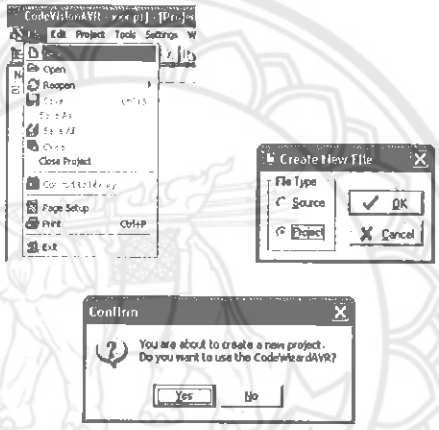


รูปที่ 3.11 ส่วนประกอบชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

3.4 การเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรเลอร์

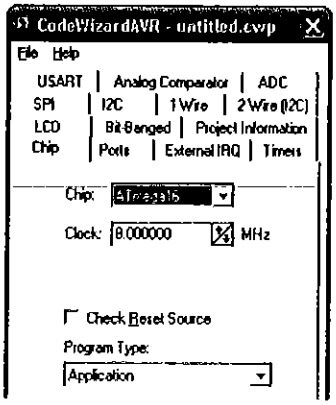
3.4.1 ขั้นตอนการเริ่มต้นใช้โปรแกรม CodevisionAVRในการเขียนโปรแกรมภาษาซีในโครงงาน

1. เปิดใช้งานโปรแกรมCodevisionAVR เมื่อหน้าต่างโปรแกรมถูกเปิดขึ้นมาเรียบร้อยแล้วให้คลิกที่เมนู file เลือก new จากนั้นเลือก Project แล้วคลิกปุ่ม OK เพื่อเริ่มต้นสร้างโค้ดโปรแกรม แล้วกดปุ่ม Yes เพื่อเลือกใช้ CodevisionAVRดังแสดงในรูปที่ 3.12



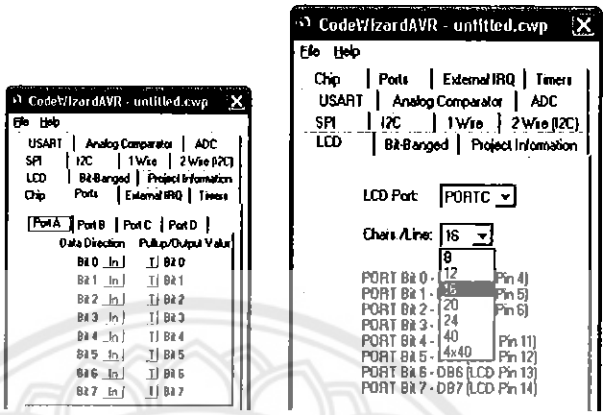
รูปที่ 3.12 การเปิดใช้งานโปรแกรม CodevisionAVR

2. เมื่อหน้าต่าง CodevisionAVR แสดงขึ้นมาให้เลือกแถบ Chip แล้วเลือกเบอร์ไมโครคอนโทรลเลอร์และความถี่ของคริสตอลดังรูปที่ 3.13



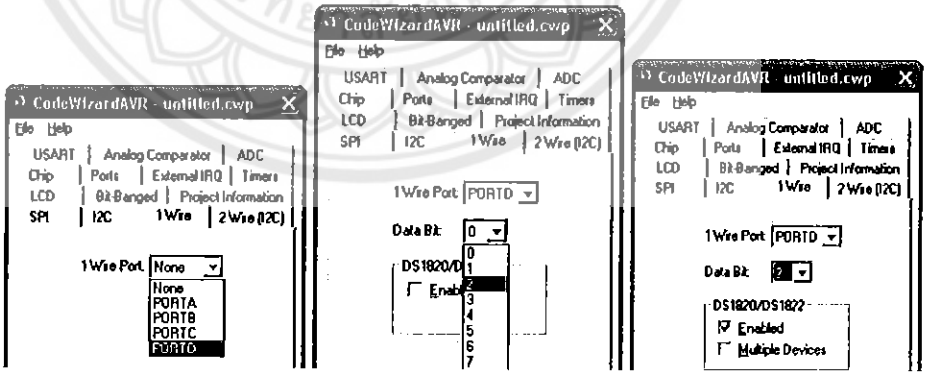
รูปที่ 3.13 การเลือกเบอร์ไมโครคอนโทรลเลอร์และความถี่ของคริสตอล

3. เลือกแถบ Port เพื่อกำหนดว่าจะให้บิตใดใน Port เป็นอินพุตและเอาต์พุต จากนั้นเลือกแถบ LCD แล้วเลือก PortC สำหรับการใช้งานโมดูลแอลซีดีดังรูปที่ 3.14



รูปที่ 3.14 การตั้งค่า Port โมดูลแอลซีดี

4. เลือกแถบ 1 Wire เพื่อกำหนดการใช้งานของไอซีวัดอุณหภูมิ DS1820 และค่าต่างๆดังรูปที่ 3.15



รูปที่ 3.15 การตั้งค่าการใช้งานไอซีวัดอุณหภูมิ DS1820

5. เลือกแถบ Timers แล้วคลิกแถบ Timer1 แล้วตั้งค่าต่างๆดังรูปที่ 3.16

Clock source: System Clock

Clock value: 1000.000 kHz

Mode: Normal top=FFFFh

OC1A output: Discon.

OC1B output: Discon.

Noise Canceler: Off

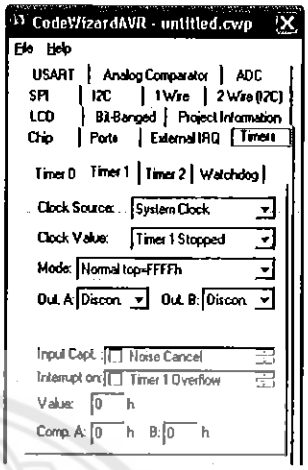
Input Capture on Falling Edge

Timer 1 Overflow Interrupt: On

Input Capture Interrupt: Off

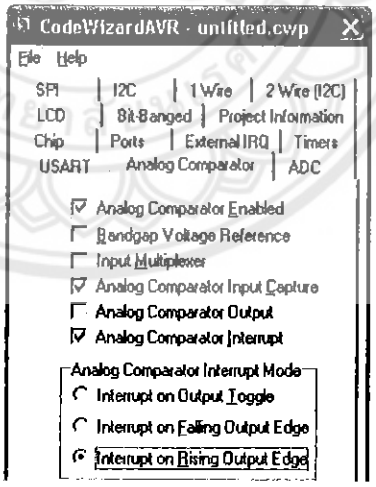
Compare A Match Interrupt: Off

Compare B Match Interrupt: Off



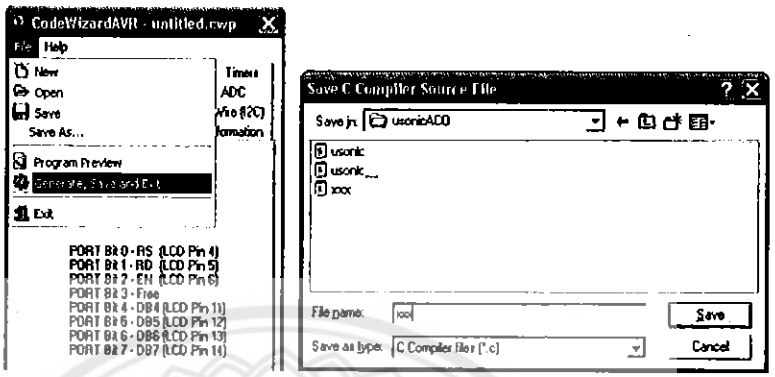
รูปที่ 3.16 การตั้งค่า Timers

6. เลือกแถบ Analog Comparator แล้วตั้งค่าต่างๆดังรูปที่ 3.17



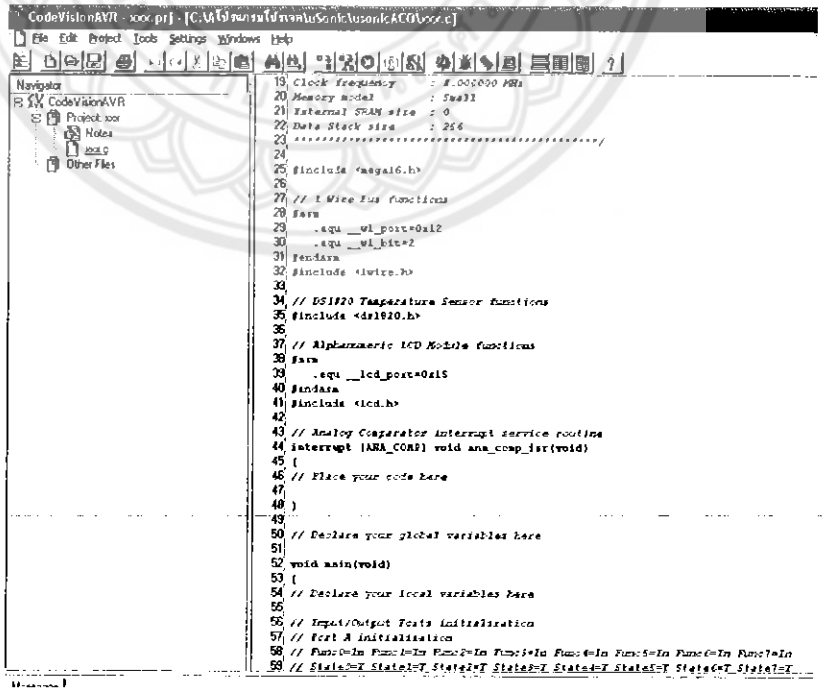
รูปที่ 3.17 การตั้งค่า Analog Comparator

7. คลิก File เลือก Generat,Save and Exit แล้วเลือกที่สำหรับ SAVE FILE จากนั้นตั้งชื่อ File เป็นชื่อเดียวกัน เช่น xxx ทั้งสามนามสกุล รูปที่ 3.18



รูปที่ 3.18 การ Save File

8. เมื่อ Save File เรียบร้อยแล้ว จะมีหน้าต่างปรากฏโค้ดที่โปรแกรม CodeWizardAVRสร้างขึ้นมาให้ จากนั้นให้เข้าไปแก้ไขโค้ด ในส่วนของการประกาศตัวแปรต่างๆ การกำหนดค่าในรีจิสเตอร์ และเงื่อนไขการทำงานของโปรแกรม ดังรูปที่

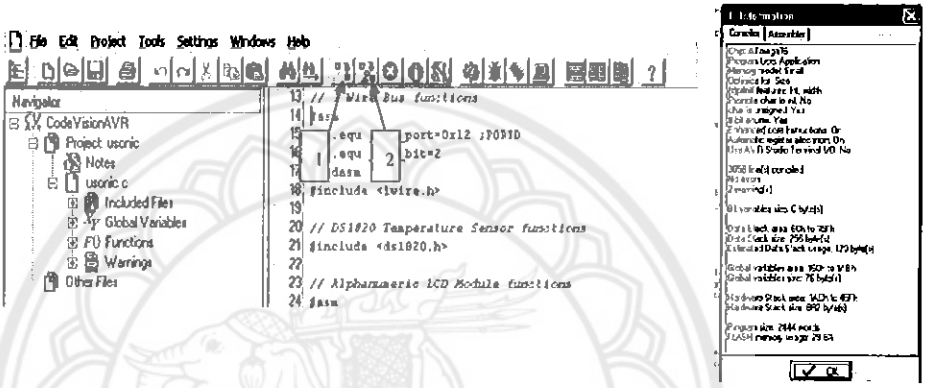


รูปที่ 3.19 หน้าต่างปรากฏโค้ดที่โปรแกรม

3.4.2 ขั้นตอนในการอัดโปรแกรมลงในไมโครคอนโทรลเลอร์มีดังนี้

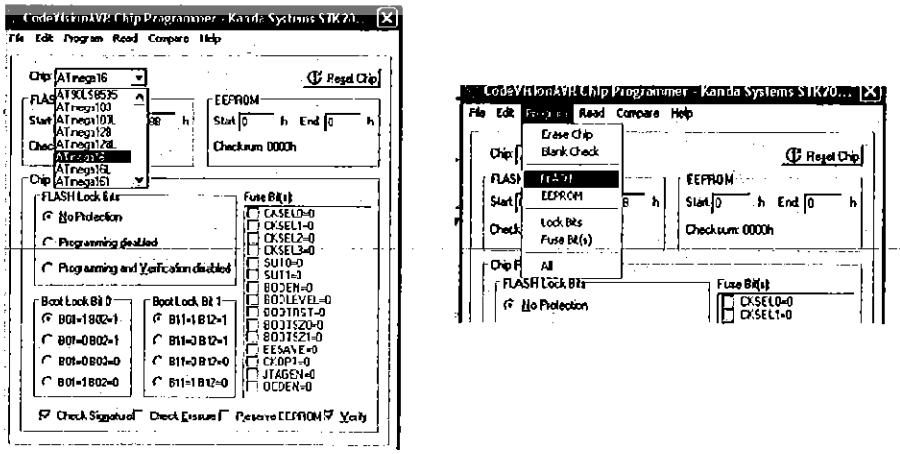
1. เปิดใช้งานโปรแกรม CodevisionAVR เมื่อเขียนโปรแกรมเรียบร้อยแล้ว คลิกไอคอน  เพื่อคอมไพล์โปรแกรม

2. เมื่อคอมไพล์ผ่าน คลิกที่ไอคอน  จากนั้นคลิก OK จะได้ File นามสกุล .HEX ดังรูปที่ 3.20



รูปที่ 3.20 การคอมไพล์โปรแกรม

3. คลิกที่ไอคอน  เพื่ออัดโปรแกรม คลิกเครื่องหมายถูกหน้า Check Signature, Check Erasure และ Verify เลือกเบอร์ไมโครคอนโทรลเลอร์ ที่ช่อง Chip เช่น ATmega16A จากนั้นไปที่เมนู Program คลิก FLASH เพื่อเริ่มอัดโปรแกรมแบบ FLASH ดังรูปที่ 3.21



รูปที่ 3.21 การอัดโปรแกรมลงในไมโครคอนโทรลเลอร์

บทที่ 4

ผลการทดสอบ

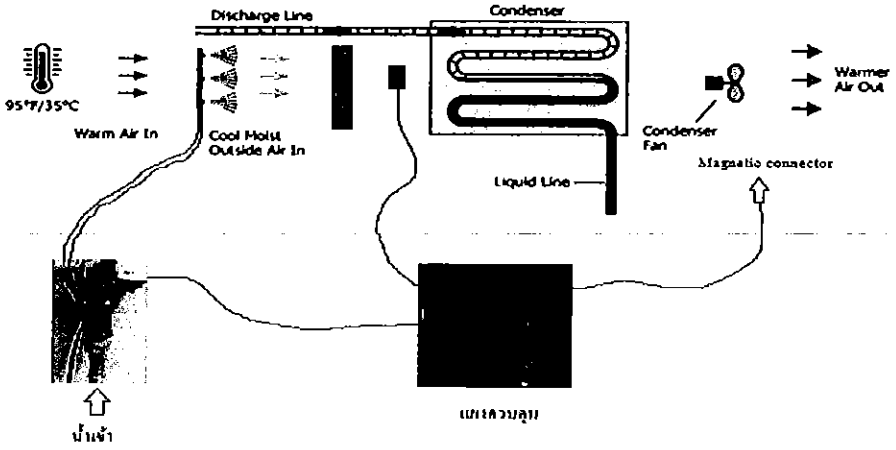
ในบทนี้จะเป็นการทดสอบชุดควบคุมการพ่นละอองน้ำอัตโนมัติโดยใช้ ไมโครคอนโทรลเลอร์หลังจากที่ได้สร้างวงจรควบคุมและอัดโปรแกรมการทำงานพร้อมทั้งการนำไปติดตั้งทดสอบร่วมกับคอนเดนเซอร์ซึ่งแบ่งการทดสอบออกเป็น 2 ส่วนดังนี้

1) ทดสอบความสามารถในการทำงานของชุดควบคุมการพ่นละอองน้ำอัตโนมัติโดยใช้ไมโครคอนโทรลเลอร์ ว่าเป็นไปตามเงื่อนไขที่กำหนดเอาไว้หรือไม่
2) ทดสอบบันทึกผลอุณหภูมิและค่าการใช้ไฟฟ้าทั้งก่อนและหลังการติดตั้งชุดควบคุมการพ่นละอองน้ำอัตโนมัติโดยใช้ไมโครคอนโทรลเลอร์

4.1 ความสามารถในการทำงานของชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

โดยปกติการทำงานของคอนเดนเซอร์จะใช้อากาศโดยรอบพัดผ่านเข้าสู่ตัวคอนเดนเซอร์เพื่อระบายความร้อนให้อุณหภูมิของสารทำความเย็นมีอุณหภูมิลดลงและต้องใช้พลังงานไฟฟ้ามากขึ้นหากการระบายความร้อนเป็นไปได้ช้าโดยในโครงงานนี้ได้กำหนดเงื่อนไขในการทำงานของชุดควบคุมการพ่นละอองน้ำอัตโนมัติโดยใช้ไมโครคอนโทรลเลอร์ไว้ที่อุณหภูมิ 35 องศาเซลเซียสหรือมากกว่าให้ทำงานในการสั่งรีเลย์ควบคุมการเปิด โซลินอยด์วาล์วทำการพ่นละอองน้ำ

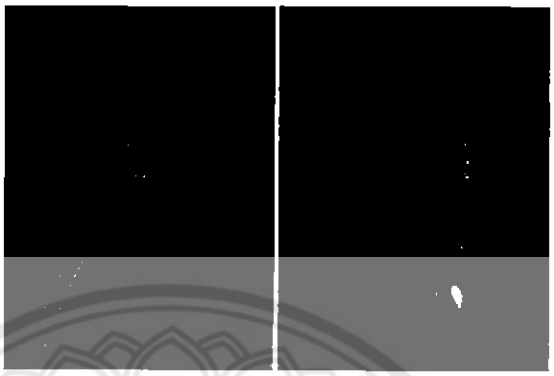
4.1.1 การติดตั้งแบบจำลองการพ่นละอองน้ำอัตโนมัติ



รูปที่ 4.1 ภาพรวมการติดตั้งชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

4.1.2 การทำงานของชุดควบคุมการพ่นละอองน้ำอัตโนมัติโดยใช้ไมโครคอนโทรลเลอร์

ในหัวข้อนี้ทางผู้ดำเนินโครงงานได้ทำการกำหนดเงื่อนไขในการทำงานของชุดควบคุมการพ่นละอองน้ำอัตโนมัติโดยใช้ไมโครคอนโทรลเลอร์



(ก) (ข)

รูปที่ 4.2 ไมโครคอนโทรลเลอร์รับค่าอุณหภูมิจากds1820 และนำไปแสดงผลทางจอแอลซีดี

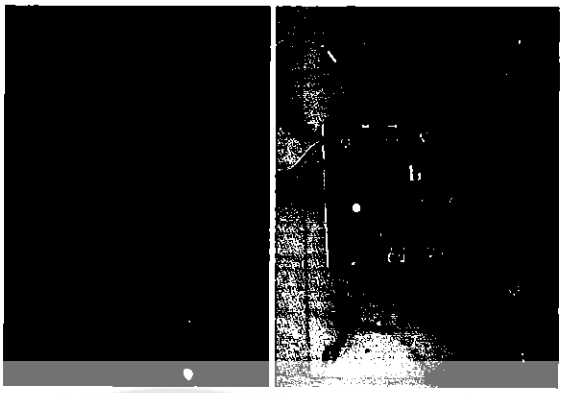
เริ่มต้นจากรูปที่ 4.2 (ก) โปรแกรมไมโครคอนโทรลเลอร์รับค่าอุณหภูมิจากds1820 และนำไปแสดงผลทางจอแอลซีดีเพื่อแสดงค่าอุณหภูมิที่อ่านได้ในรูปที่ 4.2 (ข)

(ก) (ข)

รูปที่ 4.3 การกดสวิตช์เพิ่ม ลดค่าอุณหภูมิ

จากรูปที่ 4.3 (ก) โปรแกรมจะตรวจสอบว่ามีการกดสวิตช์เพิ่มอุณหภูมิหรือไม่ ถ้ามีการกดสวิตช์อุณหภูมิที่ตั้งค่าไว้ก็จะเพิ่มทีละ 0.5 องศาเซลเซียสต่อการกดสวิตช์หนึ่งครั้ง ดังรูปที่ 4.3 (ข) แต่ถ้าไม่มีการกดสวิตช์เพิ่มอุณหภูมิโปรแกรมก็จะทำงานในขั้นต่อไป คือตรวจสอบว่ามีการกด

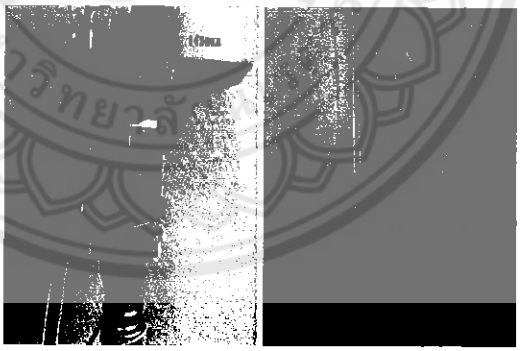
สวิตช์ลดอุณหภูมิหรือไม่ถ้ามีการกดสวิตช์อุณหภูมิที่ตั้งค่าไว้ก็จะลดทีละ 0.5 องศาเซลเซียสต่อการกดสวิตช์หนึ่งครั้ง



(ก)　　(ข)

รูปที่ 4.4 การทำงานของรีเลย์

จากรูปที่ 4.4 (ก) เมื่อค่าอุณหภูมิที่วัดได้มีค่ามากกว่าหรือเท่ากับค่าอุณหภูมิที่เซ็ตไว้คือ 35 องศาเซลเซียส ไมโครคอนโทรลเลอร์จะสั่งงานให้วงจรควบคุมรีเลย์ทำงานดังรูปที่ 4.4 (ข)



(ก)　　(ข)

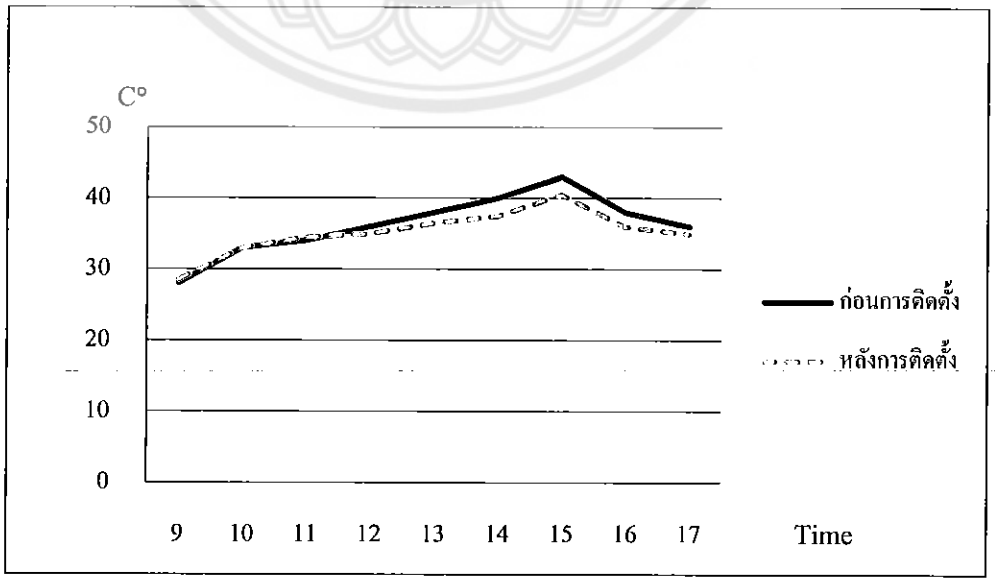
รูปที่ 4.5 การทำงานของโซลินอยด์วาล์ว

เมื่อไมโครคอนโทรลเลอร์สั่งงานให้วงจรควบคุมรีเลย์ทำงาน ก็จะเกิดกระแสไหลผ่านชุดโซลินอยด์วาล์ว ดังรูปที่ 4.5 (ก) ส่งผลให้มีการสเปรย์น้ำดังรูปที่ 4.5 (ข) ไปยังแผงคอนเดนเซอร์ โดยผ่านชุดดักจับละอองน้ำและเมื่ออุณหภูมิลดต่ำลงกว่าค่าที่เซ็ตไว้ก็จะหยุดสเปรย์น้ำ

4.2 บันทึกผลอุณหภูมิและค่าพลังงานไฟฟ้าทั้งก่อนและหลังการติดตั้งชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

ตารางที่ 4.1 บันทึกผลอุณหภูมิก่อนและหลังการติดตั้งชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

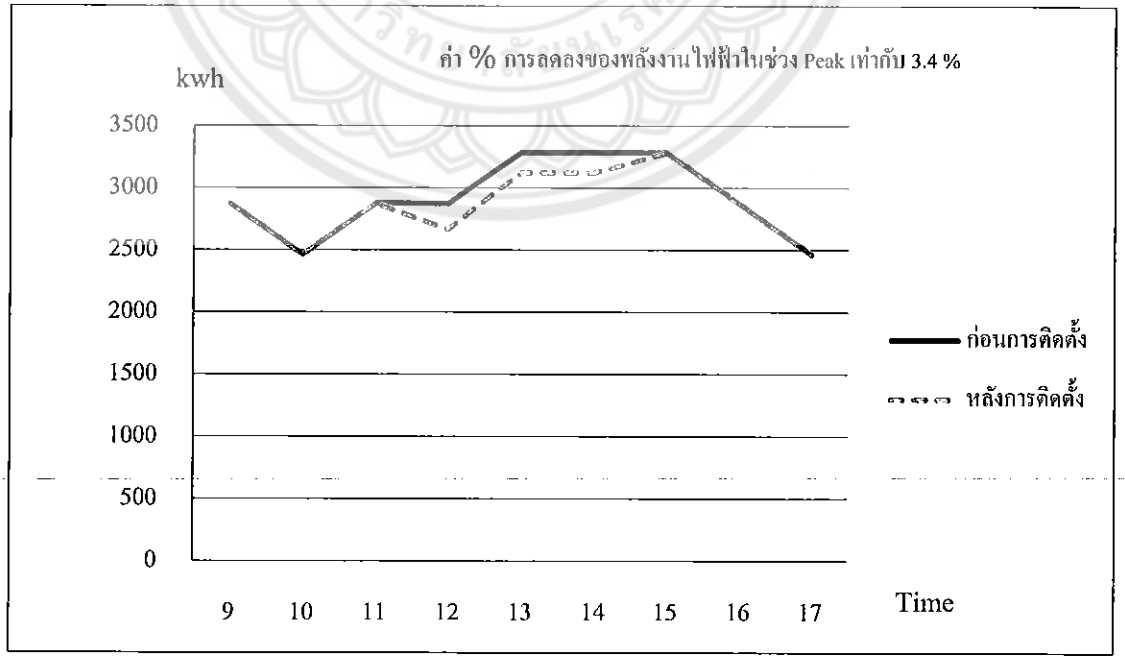
ช่วงเวลาที่วัด	อุณหภูมิที่ผ่าน Evaporator	
	ก่อนการติดตั้งชุดควบคุม	หลังการติดตั้งชุดควบคุม
(ชั่วโมง)	C°	C°
9.00 น.	28	28.5
10.00 น.	33	33
11.00 น.	34	34.5
12.00 น.	36	35
13.00 น.	38	36.5
14.00 น.	40	37.5
15.00 น.	43	40.5
16.00 น.	38	36
17.00 น.	36	35



รูปที่ 4.6 กราฟแสดงอุณหภูมิก่อนและหลังการติดตั้งชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

ตารางที่ 4.2 บันทึกผลค่าพลังงานไฟฟ้าก่อนและหลังการติดตั้งชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

ช่วงเวลาที่วัด	ค่าพลังงานไฟฟ้าที่ใช้ในการทำงานของ condenser unit	
	ก่อนการติดตั้งชุดควบคุม	หลังการติดตั้งชุดควบคุม
(ชั่วโมง)	kwh	kwh
9.00 น.	2872	2872
10.00 น.	2462	2462
11.00 น.	2872	2872
12.00 น.	2872	2667
13.00 น.	3282	3122
14.00 น.	3282	3122
15.00 น.	3282	3282
16.00 น.	2872	2872
17.00 น.	2462	2462



รูปที่ 4.7 กราฟแสดงค่าพลังงานไฟฟ้าก่อนและหลังการติดตั้งชุดควบคุมการพ่นละอองน้ำอัตโนมัติ

บทที่ 5

สรุปผลและข้อเสนอแนะ

จากการดำเนินโครงงานสามารถสรุปผล ชี้แจงปัญหาที่เกิดขึ้นในระหว่างการดำเนินงาน รวมทั้งเสนอแนวทางแก้ปัญหา พร้อมให้ข้อเสนอแนะในการทำโครงงานไปพัฒนาต่อไป

5.1 สรุปผลการดำเนินโครงงาน

ทางกลุ่มผู้จัดทำโครงงานได้ออกแบบสร้างชุดควบคุมระบบการพ่นละอองน้ำให้เปิดปิดเป็นไปอย่างอัตโนมัติ โดยใช้การทดสอบร่วมกับการทำงานของคอนเดนเซอร์แอร์ ซึ่งได้ผลดังต่อไปนี้ ในตอนเช้าช่วงเวลา 9.00 น. ถึง 11.00 น. อยู่ในช่วงที่มีอุณหภูมิต่ำกว่า 35 องศาเซลเซียส ส่งผลให้ไม่มีการทำงานของระบบพ่นละอองน้ำ เมื่อเข้าสู่ช่วงเวลา 12.00 น.จนถึง 17.00 น.อุณหภูมิจะมากกว่า 35 องศาเซลเซียส จึงทำให้มีการทำงานของระบบพ่นละอองน้ำเพื่อลดอุณหภูมิโดยรอบจนกว่าอุณหภูมิจะลดต่ำกว่าค่าที่เซ็ตไว้คือ 35 องศาเซลเซียสระบบจึงหยุดสเปรย์น้ำ และเมื่อวิเคราะห์ถึงประสิทธิภาพของพลังงานไฟฟ้าที่ใช้ของคอนเดนเซอร์แอร์พบว่ามีการใช้พลังงานไฟฟ้าที่ลดลง

5.2 ปัญหาและแนวทางการแก้ไข

1) ควรทำการทดลองในขณะที่อุณหภูมิภายนอกห้องคงที่หรือมีค่าใกล้เคียงกันมากที่สุดในแต่ละครั้งเพื่อให้เครื่องปรับอากาศได้ทำงานในสภาวะที่ใกล้เคียงกัน
2) ในการทดลองหากภาระความร้อนภายในห้องทดลองไม่คงที่จะส่งผลให้เครื่องปรับอากาศทำงานตามสภาวะที่เปลี่ยนแปลงไปเรื่อยๆซึ่งจะทำให้ค่าการทำความเย็นเปลี่ยนแปลงเรื่อยๆเช่นกัน
3) ควรศึกษาค่าอัตราส่วนประสิทธิภาพพลังงานตามมาตรฐานของเครื่องปรับอากาศที่ทำการทดลองเพื่อนำมาใช้เปรียบเทียบผลที่ได้จากการทดลอง

5.3 แนวทางการพัฒนาต่อไป

1) มีการพัฒนาวงจรให้มีขนาดเล็กลงเพื่อสะดวกในการติดตั้งใช้จริง
2) นำไปประยุกต์ใช้กับระบบอุตสาหกรรมที่ใหญ่ขึ้น

เอกสารอ้างอิง

[1] สมศักดิ์ สุโมตยกุล,เครื่องทำความเย็นและเครื่องปรับอากาศ

[2] ประจิน พลังสันติสุข,2549, การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ AVR ด้วยภาษา C กับ WinAVR(Ccomplier) เล่ม 1,กรุงเทพฯ:แอพซอฟต์เทค.

[3] มงคลทองสงครามและสุวิชาทับดี. (2545). **อิเล็กทรอนิกส์ 2**.กรุงเทพฯ : ห้างหุ้นส่วนจำกัด วี.เจ. พริ้นติ้ง.

[4] **ยืน**ภู่วรวรรณ, ทฤษฎีและการใช้งานอิเลคทรอนิกส์, ซีเอ็ดยูเคชั่น, 2521

ภาคผนวก ก

รหัสต้นฉบับของโปรแกรมควบคุมการพ่นน้ำ

```
#include <mega16.h>
//=============================================================
#include <delay.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <stdio.h>

#define MAX_DEVICES 8
unsigned char devices=0;
unsigned char rom_codes[MAX_DEVICES][9];
int  temp_ds18B20_Buff1=0;
charlcd[20];
char Fag1,Fag2;

int set=350;
eepromintset_eep = 350;

#define  SW_UP    PIND.6
#define  SW_DOWN   PIND.7
#define  Relay    PORTB.0
//=============================================================
// 1 Wire Bus functions
#asm
  .equ __w1_port=0x1B ;PORTA
  .equ __w1_bit=0
#endasm
#include <1wire.h>

// DS1820 Temperature Sensor functions
#include <ds1820.h>
```

```
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
charrx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsignedintrx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bitrx_buffer_overflow;
```

```
// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
charstatus,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
rx_buffer[rx_wr_index]=data;
if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
if (++rx_counter == RX_BUFFER_SIZE)
    {
rx_counter=0;
rx_buffer_overflow=1;
    };
  };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
chargetchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
```

```
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
chartx_buffer[TX_BUFFER_SIZE];

#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsignedinttx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
  {
  --tx_counter;
  UDR=tx_buffer[tx_rd_index];
if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
  };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
voidputchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
```

```
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
  {
tx_buffer[tx_wr_index]=c;
if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
  ++tx_counter;
  }
else
  UDR=c;
#asm("sei")
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
//=============================================
if(SW_UP)    Fag1 = 1;
if(SW_DOWN)  Fag2 = 1;
```

ตั้งค่าสถานะเริ่มต้นของสวิตช์โดยให้ SW_UP เท่ากับ Fag1=1 และ SW_DOWN เท่ากับ Fag2=1

```
if(!SW_UP)
        {
delay_ms(10);
if(!SW_UP && set<1000 && Fag1 == 1)
          {
```

ถ้ามีการกดสวิตช์ SW_UP โดยหน่วงเวลาการทำงาน 10 ms โดยมีเงื่อนไขว่าอุณหภูมิจะต้องน้อยกว่า 100°ที่ Fag1=1 และเมื่อปล่อยสวิตช์ SW_UP จะเปลี่ยนเป็น Fag1=0 ค่าอุณหภูมิจะเพิ่มขึ้น 0.5 องศาเซลเซียส จากเดิมและเก็บค่าไว้ที่ eep

```
set += 5;
set_eep = set;
            Fag1 = 0;
          }
}else if(!SW_DOWN)
        {
delay_ms(10);
if(!SW_DOWN && set>0 && Fag2 == 1)
          {
set -= 5;
set_eep = set;
            Fag2 = 0;
          }
        }
//=============================================================

}

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=0
```

ถ้ามีการกดสวิตช์ SW_DOWN โดยหน่วงเวลาการทำงาน 10 ms โดยมีเงื่อนไขว่าอุณหภูมิจะต้องมากกว่า 0°ที่ Fag2=1 และเมื่อปล่อยสวิตช์ SW_DOWN จะเปลี่ยนเป็น Fag2=0 ค่าอุณหภูมิจะลดลง 0.5 องศาเซลเซียส จากเดิมและเก็บค่าไว้ที่ eep

```
PORTB=0x00;
DDRB=0x01;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0xC0;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 10.800 kHz
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x05;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
```

```
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
```

```
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x01;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// 1 Wire Bus initialization
w1_init();

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")
lcd_clear();
lcd_gotoxy(0,0);
sprintf(lcd,"  Temp Control  ");
```

```
lcd_puts(lcd);
lcd_gotoxy(0,1);
sprintf(lcd,"              ");
lcd_puts(lcd);
delay_ms(1000);
set = set_eep;

while (1)
    {
// Place your code here
    //=============================================================
 X:devices=w1_search(0xf0,rom_codes);   // มีการต่อ ds1820 เพียง 1 ตัว
 //temp_ds18B20_Buff1=ds18b20_temperature(&rom_codes[0][0]);
temp_ds18B20_Buff1=ds1820_temperature_10(&rom_codes[0][0]);
if(temp_ds18B20_Buff1 < 0) goto X;  //ถ้า DS1820 อ่านค่าอุณหภูมิน้อยกว่า 0 ให้กลับไปอ่านค่าใหม่ที่ X
lcd_gotoxy(0,0);  //ให้บรรทัดที่ 1 ของแอลซีดีแสดงค่าอุณหภูมิจริง ( Temp )
sprintf(lcd," Temp ===> %2i.%iC",temp_ds18B20_Buff1/10,temp_ds18B20_Buff1%10);
lcd_puts(lcd);  //กำหนดให้แสดงเป็นจำนวนเต็มกับจุดทศนิยม
lcd_gotoxy(0,1);  //ให้บรรทัดที่ 2ของแอลซีดีแสดงค่าอุณหภูมิที่เซ็ตไว้( SET )
sprintf(lcd," SET  ===> %2i.%iC",set/10,set%10);
lcd_puts(lcd);  //กำหนดให้แสดงเป็นจำนวนเต็มกับจุดทศนิยมซึ่งรับค่ามาจากสวิตช์
//delay_ms(100);

if(temp_ds18B20_Buff1 >= set)   Relay = 1;
else                            Relay = 0;

    //=============================================================

    };
    }
```

ก๊อปการใช้งาน DS1820มา จาก HELP ของโปรแกรม

ถ้าเป็นไปตามเงื่อนไขที่เซ็ตไว้ >= จะทำให้ Reray ทำงาน และถ้าไม่จะสั่ง Rerayหยุดทำงาน

ภาคผนวก ข

รายละเอียดของไอซีหมายเลข ATMEGA16A

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 16K Bytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1K Byte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels in TQFP Package Only
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.6V for ATmega16A
- Speed Grades
  - 0 - 16 MHz for ATmega16A
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16A
  - Active: 0.6 mA
  - Idle Mode: 0.2 mA
  - Power-down Mode: < 1µA

8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

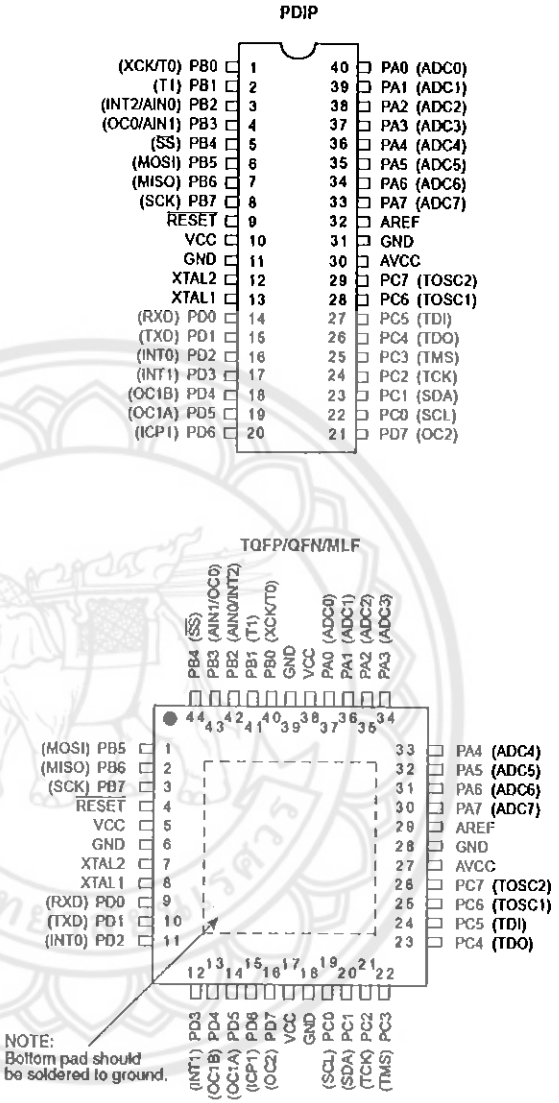
ATmega16A

Summary

8154BS–AVR–07/09

1. Pin Configurations

Figure 1-1. Pinout ATmega16A

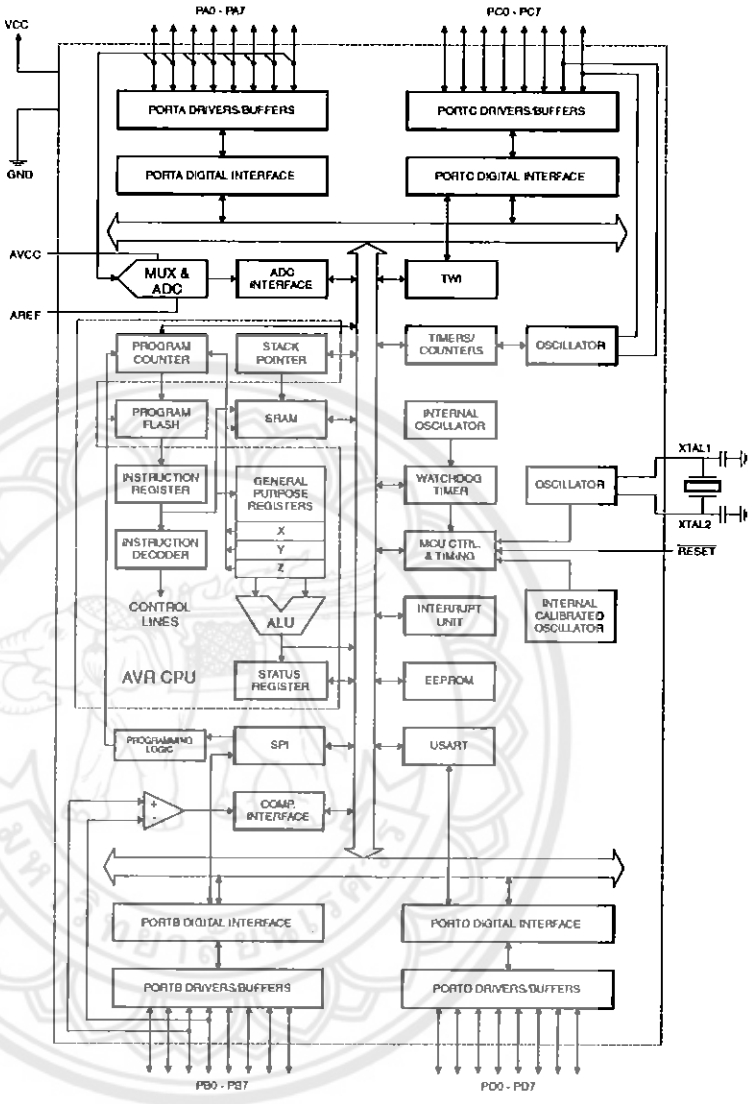


2. Overview

The ATmega16A is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16A achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16A provides the following features: 16K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1K byte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16A is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16A AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-circuit emulators, and evaluation kits.

2.2 Pin Descriptions

2.2.1 VCC

Digital supply voltage.

2.2.2 GND

Ground.

2.2.3 Port A (PA7:PA0)

Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

2.2.4 Port B (PB7:PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega16A as listed on page 57.

2.2.5 Port C (PC7:PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Port C also serves the functions of the JTAG interface and other special features of the ATmega16A as listed on page 60.

2.2.6 Port D (PD7:PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega16A as listed on page 62.

2.2.7 $\overline{\text{RESET}}$

Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 27-2 on page 296. Shorter pulses are not guaranteed to generate a reset.

2.2.8 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

2.2.9 XTAL2

Output from the inverting Oscillator amplifier.

2.2.10 AVCC

AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

2.2.11 AREF

AREF is the analog reference pin for the A/D Converter.

3. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

4. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

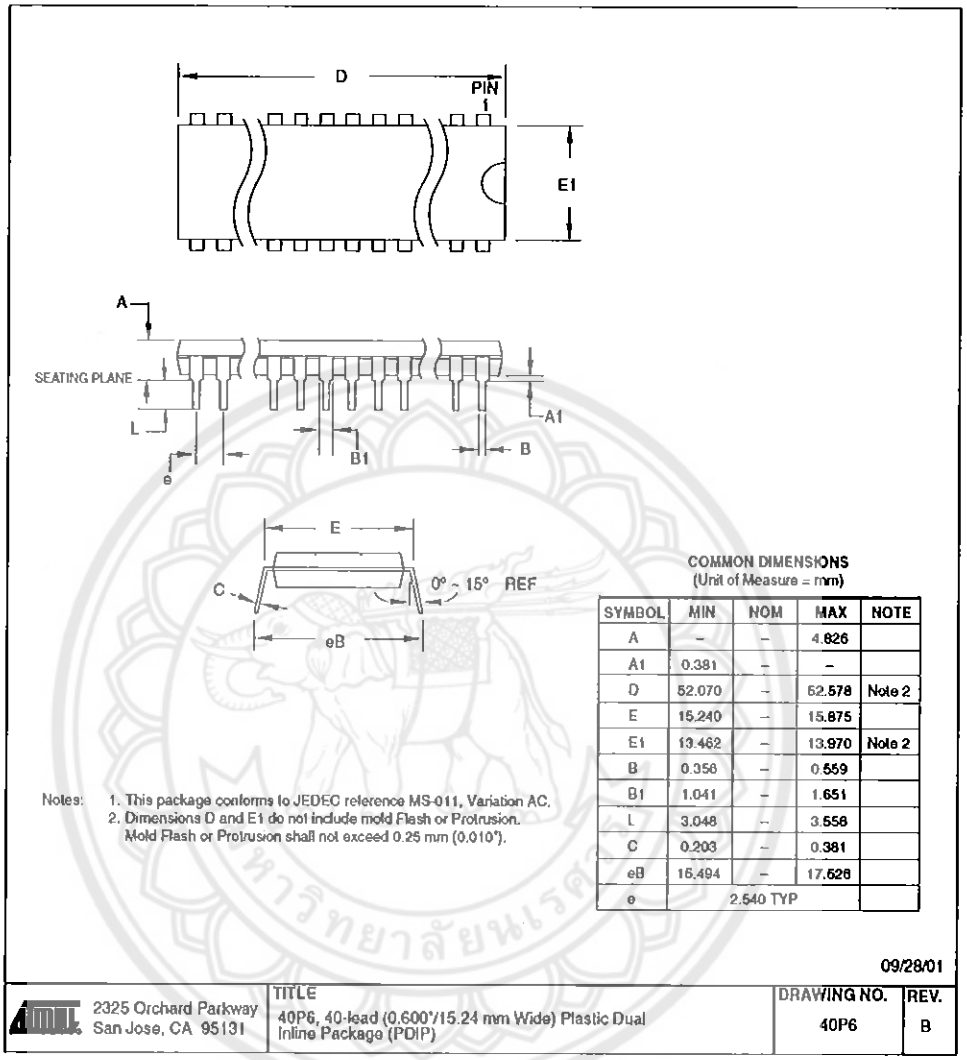
7. Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
16	2.7 - 5.5V	ATmega16A-AU[1] ATmega16A-PU[1] ATmega16A-MU[1]	44A 40P6 44M1	Industrial (-40°C to 85°C)

Note: 1. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

Package Type	
44A	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
40P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44M1	44-pad, 7 x 7 x 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

40P6



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

Notes: 1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

09/28/01

2325 Orchard Parkway San Jose, CA 95131	TITLE 40P6, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	DRAWING NO. 40P6	REV. B

ภาคผนวก ค

รายละเอียดของอุปกรณ์แปลงค่าอุณหภูมิDS1820

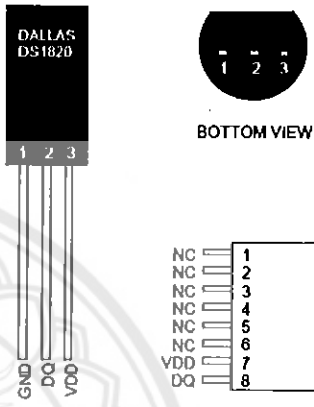
DALLAS SEMICONDUCTOR

DS1820
1–Wire™ Digital Thermometer

FEATURES

- Unique 1–Wire™ interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line
- Zero standby power required
- Measures temperatures from –55°C to +125°C in 0.5°C increments. Fahrenheit equivalent is –67°F to +257°F in 0.9°F increments
- Temperature is read as a 9–bit digital value.
- Converts temperature to digital word in 200 ms (typ.)
- User–definable, nonvolatile temperature alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

PIN ASSIGNMENT



DS1820
PR35 PACKAGE
See Mech. Drawings Section

DS1820S
16–PIN SSOP
See Mech. Drawings Section

PIN DESCRIPTION

GND	– Ground
DQ	– Data In/Out
V_{DD}	– Optional V_{DD}
NC	– No Connect

DESCRIPTION

The DS1820 Digital Thermometer provides 9–bit temperature readings which indicate the temperature of the device.

Information is sent to/from the DS1820 over a 1–Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS1820. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS1820 contains a unique silicon serial number, multiple DS1820s can exist on the same 1–Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and in process monitoring and control.

DETAILED PIN DESCRIPTION

PIN 16-PIN SSOP	PIN PR35	SYMBOL	DESCRIPTION
9	1	GND	Ground.
8	2	DQ	Data Input/Output pin. For 1-Wire operation: Open drain. (See "Parasite Power" section.)
7	3	V_{DD}	Optional V_{DD} pin. See "Parasite Power" section for details of connection.

DS1820S (16-pin SSOP): All pins not specified in this table are not to be connected.

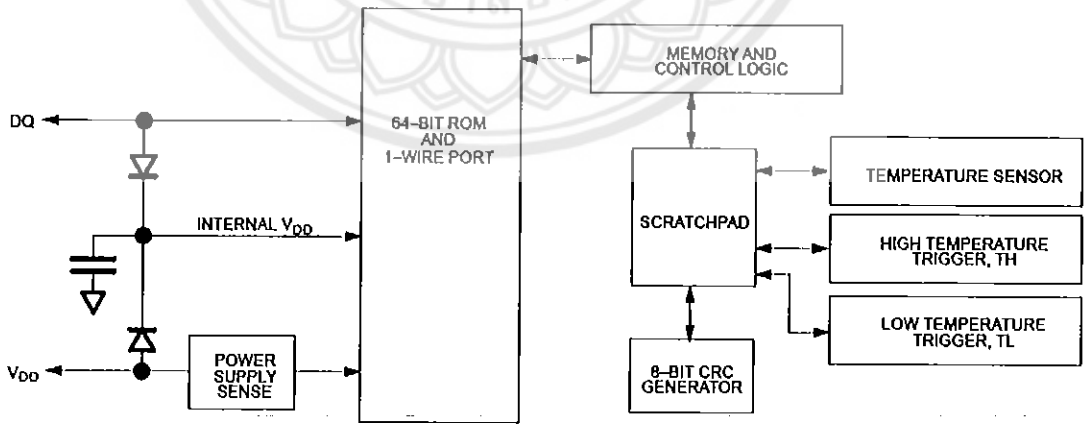
OVERVIEW

The block diagram of Figure 1 shows the major components of the DS1820. The DS1820 has three main data components: 1) 64-bit lasered ROM, 2) temperature sensor, and 3) nonvolatile temperature alarm triggers TH and TL. The device derives its power from the 1-Wire communication line by storing energy on an internal capacitor during periods of time when the signal line is high and continues to operate off this power source during the low times of the 1-Wire line until it returns high to replenish the parasite (capacitor) supply. As an alternative, the DS1820 may also be powered from an external 5 volts supply.

Communication to the DS1820 is via a 1-Wire port. With the 1-Wire port, the memory and control functions will not be available before the ROM function protocol has been established. The master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. These commands operate on the 64-bit lasered ROM portion of each device and can single out a specific device if many are present on the 1-Wire line as well as indicate to the Bus Master how many and what types of devices are present. After a ROM function sequence has been successfully executed, the memory and control functions are accessible and the master may then provide any one of the six memory and control function commands.

One control function command instructs the DS1820 to perform a temperature measurement. The result of this measurement will be placed in the DS1820's scratchpad memory, and may be read by issuing a memory function command which reads the contents of the scratchpad memory. The temperature alarm triggers TH and TL consist of one byte EEPROM each. If the alarm search command is not applied to the DS1820, these registers may be used as general purpose user memory. Writing TH and TL is done using a memory function command. Read access to these registers is through the scratchpad. All data is read and written least significant bit first.

DS1820 BLOCK DIAGRAM Figure 1



PARASITE POWER

The block diagram (Figure 1) shows the parasite powered circuitry. This circuitry "steals" power whenever the I/O or V_{DD} pins are high. I/O will provide sufficient power as long as the specified timing and voltage requirements are met (see the section titled "1–Wire Bus System"). The advantages of parasite power are two–fold: 1) by parasiting off this pin, no local power source is needed for remote sensing of temperature, and 2) the ROM may be read in absence of normal power.

In order for the DS1820 to be able to perform accurate temperature conversions, sufficient power must be provided over the I/O line when a temperature conversion is taking place. Since the operating current of the DS1820 is up to 1 mA, the I/O line will not have sufficient drive due to the 5K pull–up resistor. This problem is particularly acute if several DS1820's are on the same I/O and attempting to convert simultaneously.

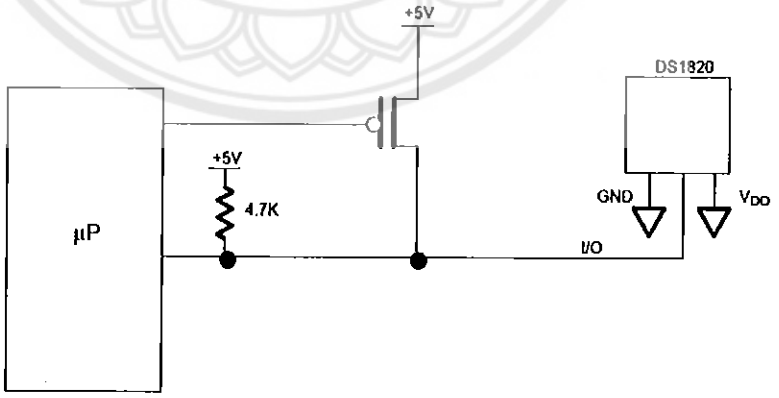
There are two ways to assure that the DS1820 has sufficient supply current during its active conversion cycle. The first is to provide a strong pull–up on the I/O line whenever temperature conversions or copies to the E^2 memory are taking place. This may be accomplished by using a MOSFET to pull the I/O line directly to the power supply as shown in Figure 2. The I/O line must be switched over to the strong pull–up within 10 µs maximum after issuing any protocol that involves copying to the E^2 memory or initiates temperature conversions. When using the parasite power mode, the V_{DD} pin must be tied to ground.

Another method of supplying current to the DS1820 is through the use of an external power supply tied to the V_{DD} pin, as shown in Figure 3. The advantage to this is that the strong pull–up is not required on the I/O line, and the bus master need not be tied up holding that line high during temperature conversions. This allows other data traffic on the 1–Wire bus during the conversion time. In addition, any number of DS1820's may be placed on the 1–Wire bus, and if they all use external power, they may all simultaneously perform temperature conversions by issuing the Skip ROM command and then issuing the Convert T command. Note that as long as the external power supply is active, the GND pin may not be floating.

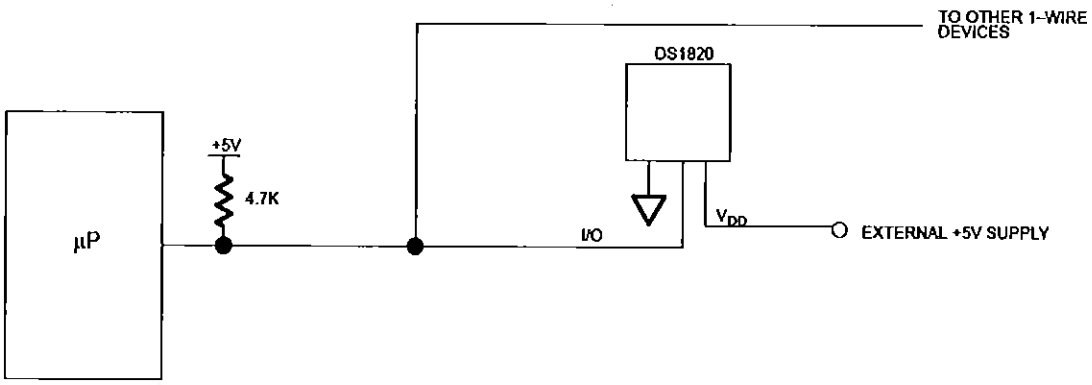
The use of parasite power is not recommended above 100°C, since it may not be able to sustain communications given the higher leakage currents the DS1820 exhibits at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that V_{DD} be applied to the DS1820.

For situations where the bus master does not know whether the DS1820's on the bus are parasite powered or supplied with external V_{DD}, a provision is made in the DS1820 to signal the power supply scheme used. The bus master can determine if any DS1820's are on the bus which require the strong pull–up by sending a Skip ROM protocol, then issuing the read power supply command. After this command is issued, the master then issues read time slots. The DS1820 will send back "0" on the 1–Wire bus if it is parasite powered; it will send back a "1" if it is powered from the V_{DD} pin. If the master receives a "0", it knows that it must supply the strong pull–up on the I/O line during temperature conversions. See "Memory Command Functions" section for more detail on this command protocol.

STRONG PULL–UP FOR SUPPLYING DS1820 DURING TEMPERATURE CONVERSION Figure 2



USING V_{DD} TO SUPPLY TEMPERATURE CONVERSION CURRENT Figure 3



OPERATION – MEASURING TEMPERATURE

The DS1820 measures temperature through the use of an on–board proprietary temperature measurement technique. A block diagram of the temperature measurement circuitry is shown in Figure 4.

The DS1820 measures temperature by counting the number of clock cycles that an oscillator with a low temperature coefficient goes through during a gate period determined by a high temperature coefficient oscillator. The counter is preset with a base count that corresponds to –55°C. If the counter reaches zero before the gate period is over, the temperature register, which is also preset to the –55°C value, is incremented, indicating that the temperature is higher than –55°C.

At the same time, the counter is then preset with a value determined by the slope accumulator circuitry. This circuitry is needed to compensate for the parabolic behavior of the oscillators over temperature. The counter is then clocked again until it reaches zero. If the gate period is still not finished, then this process repeats.

The slope accumulator is used to compensate for the non–linear behavior of the oscillators over temperature, yielding a high resolution temperature measurement. This is done by changing the number of counts necessary for the counter to go through for each incremental degree in temperature. To obtain the desired resolution, therefore, both the value of the counter and the number of counts per degree C (the value of the slope accumulator) at a given temperature must be known.

Internally, this calculation is done inside the DS1820 to provide 0.5°C resolution. The temperature reading is provided in a 16–bit, sign–extended two's complement reading. Table 1 describes the exact relationship of output data to measured temperature. The data is transmitted serially over the 1–Wire interface. The DS1820 can measure temperature over the range of –55°C to +125°C in 0.5°C increments. For Fahrenheit usage, a lookup table or conversion factor must be used.

Note that temperature is represented in the DS1820 in terms of a $^1/_2$°C LSB, yielding the following 9–bit format:

MSB								LSB
1	1	1	0	0	1	1	1	0

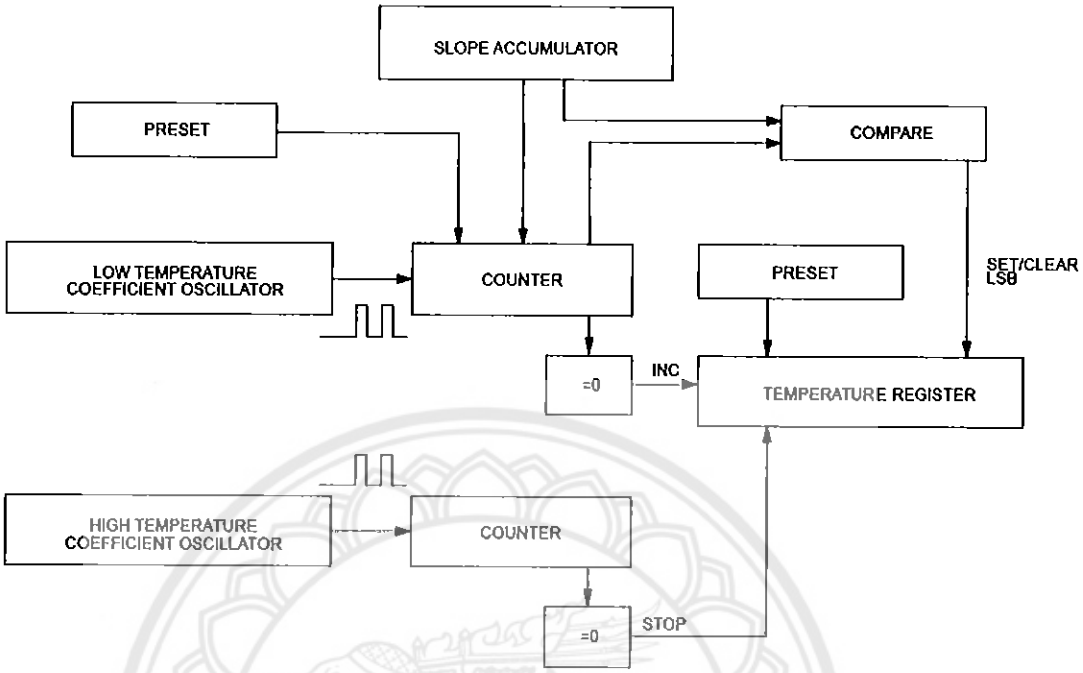
= –25°C

The most significant (sign) bit is duplicated into all of the bits in the upper MSB of the two–byte temperature register in memory. This "sign–extension" yields the 16–bit temperature readings as shown in Table 1.

Higher resolutions may be obtained by the following procedure. First, read the temperature, and truncate the 0.5°C bit (the LSB) from the read value. This value is TEMP_READ. The value left in the counter may then be read. This value is the count remaining (COUNT_REMAIN) after the gate period has ceased. The last value needed is the number of counts per degree C (COUNT_PER_C) at that temperature. The actual temperature may be then be calculated by the user using the following:

$$\text{TEMPERATURE} = \text{TEMP\_READ} - 0.25 + \frac{(\text{COUNT\_PER\_C} - \text{COUNT\_REMAIN})}{\text{COUNT\_PER\_C}}$$

TEMPERATURE MEASURING CIRCUITRY Figure 4



TEMPERATURE/DATA RELATIONSHIPS Table 1

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	00000000 11111010	00FA
+25°C	00000000 00110010	0032h
+1/2°C	00000000 00000001	0001h
+0°C	00000000 00000000	0000h
–1/2°C	11111111 11111111	FFFFh
–25°C	11111111 11001110	FFCEh
–55°C	11111111 10010010	FF92h

OPERATION – ALARM SIGNALING

After the DS1820 has performed a temperature conversion, the temperature value is compared to the trigger values stored in TH and TL. Since these registers are 8–bit only, the 0.5°C bit is ignored for comparison. The most significant bit of TH or TL directly corresponds to the sign bit of the 16–bit temperature register. If the result of a temperature measurement is higher than TH or lower than TL, an alarm flag inside the device is set. This flag is updated with every temperature measurement. As long as the alarm flag is set, the DS1820 will respond to the alarm search command. This allows many DS1820s to be connected in parallel doing simultaneous temperature measurements. If somewhere the temperature exceeds the limits, the alarming device(s) can be identified and read immediately without having to read non–alarming devices.

64–BIT LASERED ROM

Each DS1820 contains a unique ROM code that is 64–bits long. The first eight bits are a 1–Wire family code (DS1820 code is 10h). The next 48 bits are a unique serial number. The last eight bits are a CRC of the first 56 bits. (See Figure 5.) The 64–bit ROM and ROM Function Control section allow the DS1820 to operate as a 1–Wire device and follow the 1–Wire protocol detailed in the section "1–Wire Bus System". The functions required to control sections of the DS1820 are not accessible until the ROM function protocol has been satisfied. This protocol is described in the ROM function protocol flowchart (Figure 6). The 1–Wire bus master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. After a ROM functions sequence has been successfully executed, the functions specific to the DS1820 are accessible and the bus master may then provide and one of the six memory and control function commands.

CRC GENERATION

The DS1820 has an 8–bit CRC stored in the most significant byte of the 64–bit ROM. The bus master can compute a CRC value from the first 56–bits of the 64–bit ROM and compare it to the value stored within the DS1820 to determine if the ROM data has been received error–free by the bus master. The equivalent polynomial function of this CRC is:

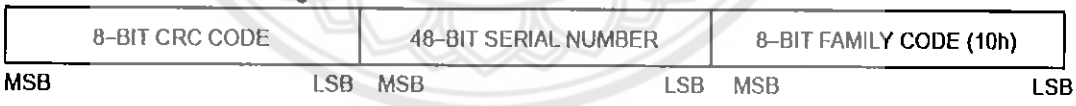
$$CRC = X^8 + X^5 + X^4 + 1$$

The DS1820 also generates an 8–bit CRC value using the same polynomial function shown above and provides this value to the bus master to validate the transfer of data bytes. In each case where a CRC is used for data transfer validation, the bus master must calculate a CRC value using the polynomial function given above and compare the calculated value to either the 8–bit CRC value stored in the 64–bit ROM portion of the DS1820 (for ROM reads) or the 8–bit CRC value computed within the DS1820 (which is read as a ninth byte when the scratchpad is read). The comparison of CRC values and decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS1820 that prevents a command sequence from proceeding if the CRC stored in or calculated by the DS1820 does not match the value generated by the bus master.

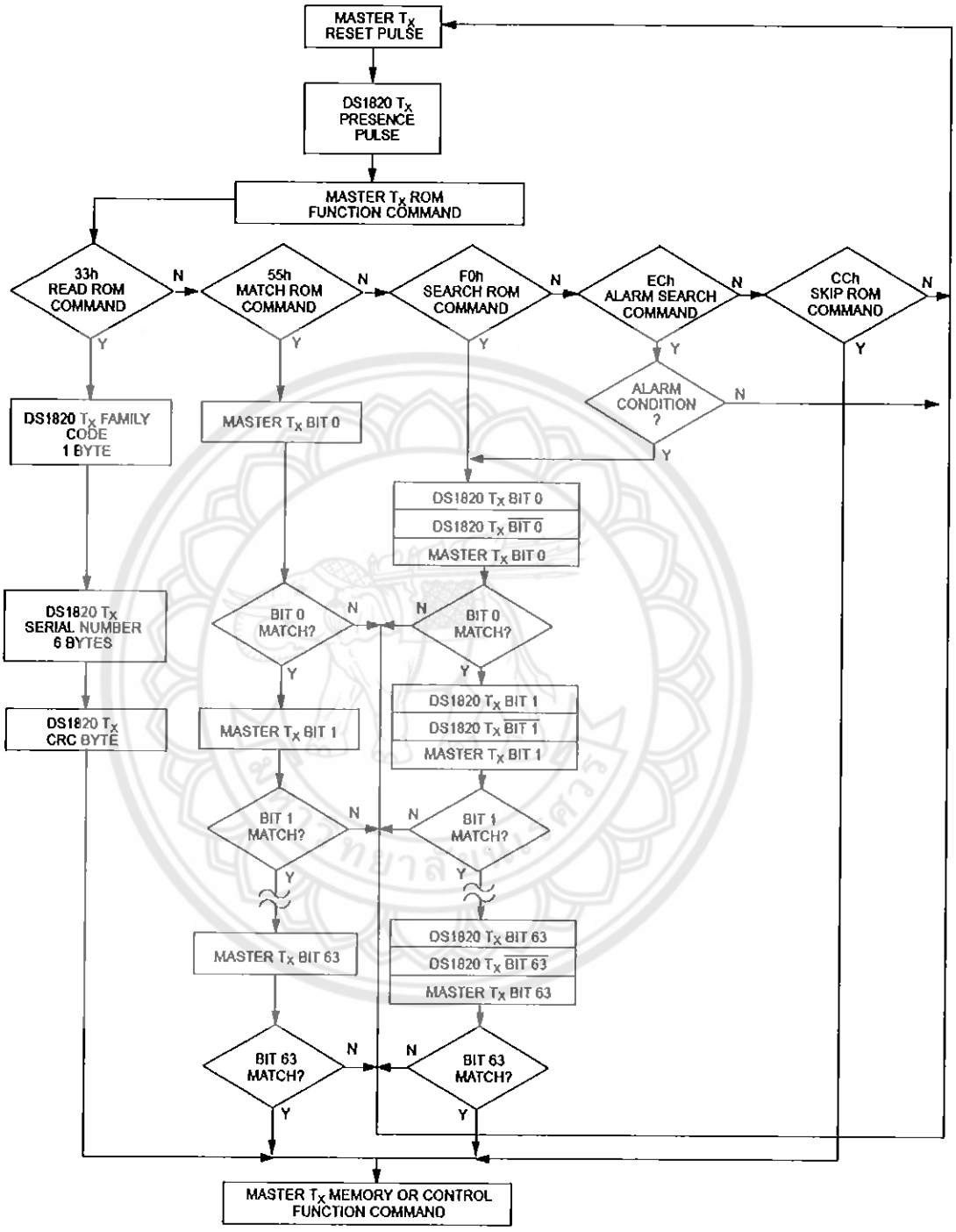
The 1–Wire CRC can be generated using a polynomial generator consisting of a shift register and XOR gates as shown in Figure 7. Additional information about the Dallas 1–Wire Cyclic Redundancy Check is available in Application Note 27 entitled "Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Products".

The shift register bits are initialized to zero. Then starting with the least significant bit of the family code, one bit at a time is shifted in. After the 8th bit of the family code has been entered, then the serial number is entered. After the 48th bit of the serial number has been entered, the shift register contains the CRC value. Shifting in the eight bits of CRC should return the shift register to all zeros.

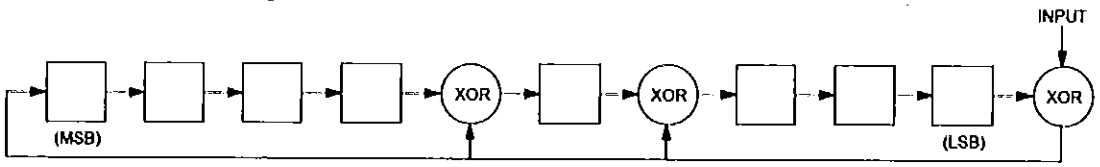
64–BIT LASERED ROM Figure 5



ROM FUNCTIONS FLOW CHART Figure 6



1–WIRE CRC CODE Figure 7



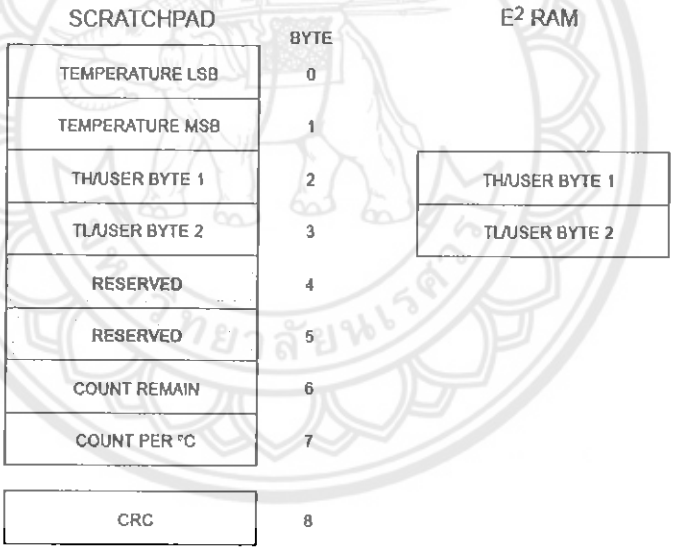
MEMORY

The DS1820's memory is organized as shown in Figure 8. The memory consists of a scratchpad RAM and a nonvolatile, electrically erasable (E^2) RAM, which stores the high and low temperature triggers TH and TL. The scratchpad helps insure data integrity when communicating over the 1–Wire bus. Data is first written to the scratchpad where it can be read back. After the data has been verified, a copy scratchpad command will transfer the data to the nonvolatile (E^2) RAM. This process insures data integrity when modifying the memory.

The scratchpad is organized as eight bytes of memory. The first two bytes contain the measured temperature information. The third and fourth bytes are volatile copies of TH and TL and are refreshed with every power–on reset. The next two bytes are not used; upon reading back, however, they will appear as all logic 1's. The seventh and eighth bytes are count registers, which may be used in obtaining higher temperature resolution (see "Operation–measuring Temperature" section).

There is a ninth byte which may be read with a Read Scratchpad command. This byte contains a cyclic redundancy check (CRC) byte which is the CRC over all of the eight previous bytes. This CRC is implemented in the fashion described in the section titled "CRC Generation".

DS1820 MEMORY MAP Figure 8



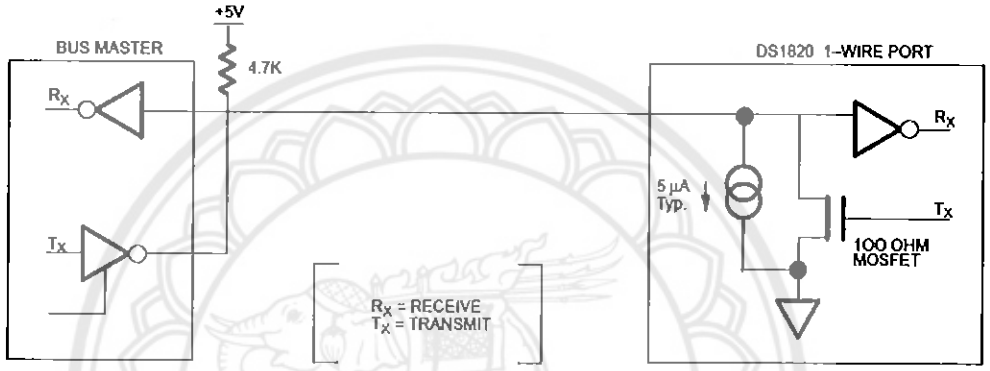
1–WIRE BUS SYSTEM

The 1–Wire bus is a system which has a single bus master and one or more slaves. The DS1820 behaves as a slave. The discussion of this bus system is broken down into three topics: hardware configuration, transaction sequence, and 1–Wire signaling (signal types and timing).

HARDWARE CONFIGURATION

The 1–Wire bus has only a single line by definition; it is important that each device on the bus be able to drive it at the appropriate time. To facilitate this, each device attached to the 1–Wire bus must have open drain or 3–state outputs. The 1–Wire port of the DS1820 (I/O pin) is open drain with an internal circuit equivalent to that shown in Figure 9. A multidrop bus consists of a 1–Wire bus with multiple slaves attached. The 1–Wire bus requires a pullup resistor of approximately 5KΩ.

HARDWARE CONFIGURATION Figure 9



The idle state for the 1–Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1–Wire bus is in the inactive (high) state during the recovery period. If this does not occur and the bus is left low for more than 480 µs, all components on the bus will be reset.

TRANSACTION SEQUENCE

The protocol for accessing the DS1820 via the 1–Wire port is as follows:

- Initialization
- ROM Function Command
- Memory Function Command
- Transaction/Data

INITIALIZATION

All transactions on the 1–Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s).

The presence pulse lets the bus master know that the DS1820 is on the bus and is ready to operate. For more details, see the "1–Wire Signaling" section.

ROM FUNCTION COMMANDS

Once the bus master has detected a presence, it can issue one of the five ROM function commands. All ROM function commands are 8–bits long. A list of these commands follows (refer to flowchart in Figure 6):

Read ROM [33h]

This command allows the bus master to read the DS1820's 8–bit family code, unique 48–bit serial number, and 8–bit CRC. This command can only be used if there is a single DS1820 on the bus. If more than one slave is present on the bus, a data collision will occur when all slaves try to transmit at the same time (open drain will produce a wired AND result).

Match ROM [55h]

The match ROM command, followed by a 64–bit ROM sequence, allows the bus master to address a specific DS1820 on a multidrop bus. Only the DS1820 that exactly matches the 64–bit ROM sequence will respond to the following memory function command. All slaves that do not match the 64–bit ROM sequence will wait for a reset pulse. This command can be used with a single or multiple devices on the bus.

Skip ROM [CCh]

This command can save time in a single drop bus system by allowing the bus master to access the memory functions without providing the 64–bit ROM code. If more than one slave is present on the bus and a read command is issued following the Skip ROM command, data collision will occur on the bus as multiple slaves transmit simultaneously (open drain pulldowns will produce a wired AND result).

Search ROM [F0h]

When a system is initially brought up, the bus master might not know the number of devices on the 1–Wire bus or their 64–bit ROM codes. The search ROM command allows the bus master to use a process of elimination to identify the 64–bit ROM codes of all slave devices on the bus.

Alarm Search [ECh]

The flowchart of this command is identical to the Search ROM command. However, the DS1820 will respond to this command only if an alarm condition has been encountered at the last temperature measurement. An alarm condition is defined as a temperature higher than TH or lower than TL. The alarm condition remains set as long as the DS1820 is powered up, or until another temperature measurement reveals a non–alarming value. For alarming, the trigger values stored in EEPROM are taken into account. If an alarm condition exists and the TH or TL settings are changed, another temperature conversion should be done to validate any alarm conditions.

Example of a ROM Search

The ROM search process is the repetition of a simple 3–step routine: read a bit, read the complement of the bit, then write the desired value of that bit. The bus master performs this simple, 3–step routine on each bit of the ROM. After one complete pass, the bus master knows the contents of the ROM in one device. The remaining number of devices and their ROM codes may be identified by additional passes.

The following example of the ROM search process assumes four different devices are connected to the same 1–Wire bus. The ROM data of the four devices is as shown:

ROM1	00110101...
ROM2	10101010...
ROM3	11110101...
ROM4	00010001...

The search process is as follows:

1. The bus master begins the initialization sequence by issuing a reset pulse. The slave devices respond by issuing simultaneous presence pulses.
2. The bus master will then issue the Search ROM command on the 1–Wire bus.
3. The bus master reads a bit from the 1–Wire bus. Each device will respond by placing the value of the first bit of their respective ROM data onto the 1–Wire bus. ROM1 and ROM4 will place a 0 onto the 1–Wire bus, i.e., pull it low. ROM2 and ROM3 will place a 1 onto the 1–Wire bus by allowing the line to stay high. The result is the logical AND of all devices on the line, therefore the bus master sees a 0. The bus master reads another bit. Since the Search ROM data command is being executed, all of the devices on the 1–Wire bus respond to this second read by placing the complement of the first bit of their respective ROM data onto the 1–Wire bus. ROM1 and ROM4 will place a 1 onto the 1–Wire, allowing the line to stay high. ROM2 and ROM3 will place a 0 onto the 1–Wire, thus it will be pulled low. The bus master again observes a 0 for the complement of the first ROM data bit. The bus master has determined that there are some devices on the 1–Wire bus that have a 0 in the first position and others that have a 1.

The data obtained from the two reads of the 3–step routine have the following interpretations:

00 There are still devices attached which have conflicting bits in this position.

01 All devices still coupled have a 0–bit in this bit position.

10 All devices still coupled have a 1–bit in this bit position.

11 There are no devices attached to the 1–Wire bus.

4. The bus master writes a 0. This deselects ROM2 and ROM3 for the remainder of this search pass, leaving only ROM1 and ROM4 connected to the 1–Wire bus.
5. The bus master performs two more reads and receives a 0–bit followed by a 1–bit. This indicates that all devices still coupled to the bus have 0's as their second ROM data bit.
6. The bus master then writes a 0 to keep both ROM1 and ROM4 coupled.
7. The bus master executes two reads and receives two 0–bits. This indicates that both 1–bits and 0–bits exist as the third bit of the ROM data of the attached devices.
8. The bus master writes a 0–bit. This deselects ROM1 leaving ROM4 as the only device still connected.
9. The bus master reads the remainder of the ROM bits for ROM4 and continues to access the part if desired. This completes the first pass and uniquely identifies one part on the 1–Wire bus.
10. The bus master starts a new ROM search sequence by repeating steps 1 through 7.
11. The bus master writes a 1–bit. This decouples ROM4, leaving only ROM1 still coupled.
12. The bus master reads the remainder of the ROM bits for ROM1 and communicates to the underlying logic if desired. This completes the second ROM search pass, in which another of the ROMs was found.
13. The bus master starts a new ROM search by repeating steps 1 through 3.
14. The bus master writes a 1–bit. This deselects ROM1 and ROM4 for the remainder of this search pass, leaving only ROM2 and ROM3 coupled to the system.
15. The bus master executes two read time slots and receives two zeros.
16. The bus master writes a 0–bit. This decouples ROM3, and leaving only ROM2.
17. The bus master reads the remainder of the ROM bits for ROM2 and communicates to the underlying logic if desired. This completes the third ROM search pass, in which another of the ROMs was found.
18. The bus master starts a new ROM search by repeating steps 13 through 15.
19. The bus master writes a 1–bit. This decouples ROM2, leaving only ROM3.
20. The bus master reads the remainder of the ROM bits for ROM3 and communicates to the underlying logic if desired. This completes the fourth ROM search pass, in which another of the ROMs was found.

Note the following:
The bus master learns the unique ID number (ROM data pattern) of one 1–Wire device on each ROM Search operation. The time required to derive the part's unique ROM code is:

$$960\ \mu s + (8 + 3 \times 64)\ 61\ \mu s = 13.16\ ms$$

The bus master is therefore capable of identifying 75 different 1–Wire devices per second.

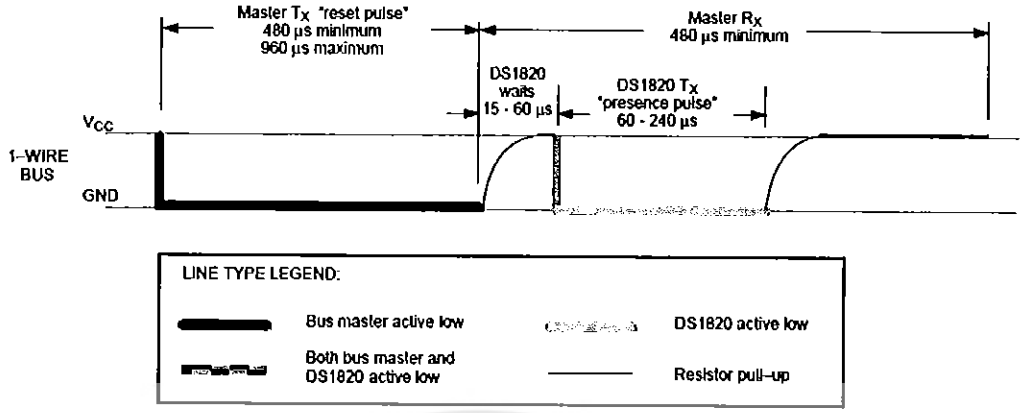
I/O SIGNALING

The DS1820 requires strict protocols to insure data integrity. The protocol consists of several types of signaling on one line: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. All of these signals, with the exception of the presence pulse, are initiated by the bus master.

The initialization sequence required to begin any communication with the DS1820 is shown in Figure 11. A reset pulse followed by a presence pulse indicates the DS1820 is ready to send or receive data given the correct ROM command and memory function command.

The bus master transmits (TX) a reset pulse (a low signal for a minimum of 480 µs). The bus master then releases the line and goes into a receive mode (RX). The 1–Wire bus is pulled to a high state via the 5K pull–up resistor . After detecting the rising edge on the

INITIALIZATION PROCEDURE "RESET AND PRESENCE PULSES" Figure 11



DS1820 COMMAND SET Table 2

INSTRUCTION	DESCRIPTION	PROTOCOL	1-WIRE BUS AFTER ISSUING PROTOCOL	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	<read temperature busy status>	1
MEMORY COMMANDS				
Read Scratchpad	Reads bytes from scratchpad and reads CRC byte.	BEh	<read data up to 9 bytes>	
Write Scratchpad	Writes bytes into scratchpad at addresses 2 and 3 (TH and TL temperature triggers).	4Eh	<write data into 2 bytes at addr. 2 and addr. 3>	
Copy Scratchpad	Copies scratchpad into nonvolatile memory (addresses 2 and 3 only).	48h	<read copy status>	2
Recall E^2	Recalls values stored in nonvolatile memory into scratchpad (temperature triggers).	B8h	<read temperature busy status>	
Read Power Supply	Signals the mode of DS1820 power supply to the master.	B4h	<read supply status>	

NOTES:

1. Temperature conversion takes up to 500 ms. After receiving the Convert T protocol, if the part does not receive power from the V_{DD} pin, the I/O line for the DS1820 must be held high for at least 500 ms to provide power during the conversion process. As such, no other activity may take place on the 1-Wire bus for at least this period after a Convert T command has been issued.
2. After receiving the Copy Scratchpad protocol, if the part does not receive power from the V_{DD} pin, the I/O line for the DS1820 must be held high for at least 10 ms to provide power during the copy process. As such, no other activity may take place on the 1-Wire bus for at least this period after a Copy Scratchpad command has been issued.

Read Scratchpad [BEh]
This command reads the contents of the scratchpad. Reading will commence at byte 0, and will continue through the scratchpad until the 9th (byte–8, CRC) byte is read. If not all locations are to be read, the master may issue a reset to terminate reading at any time.

Copy Scratchpad [48h]
This command copies the scratchpad into the E^2 memory of the DS1820, storing the temperature trigger bytes in nonvolatile memory. If the bus master issues read time slots following this command, the DS1820 will output "0" on the bus as long as it is busy copying the scratchpad to E^2; it will return a "1" when the copy process is complete. If parasite powered, the bus master has to enable a strong pull–up for at least 10 ms immediately after issuing this command.

Convert T [44h]
This command begins a temperature conversion. No further data is required. The temperature conversion will be performed and then the DS1820 will remain idle. If the bus master issues read time slots following this command, the DS1820 will output "0" on the bus as long as it is busy making a temperature conversion; it will return a "1" when the temperature conversion is complete. If parasite powered, the bus master has to enable a strong pullup for 500 ms immediately after issuing this command.

Recall E2 [B8h]
This command recalls the temperature trigger values stored in E^2 to the scratchpad. This recall operation happens automatically upon power–up to the DS1820 as well, so valid data is available in the scratchpad as soon as the device has power applied. With every read data time slot issued after this command has been sent, the device will output its temperature converter busy flag "0"=busy, "1"=ready.

Read Power Supply [B4h]
With every read data time slot issued after this command has been sent to the DS1820, the device will signal its power mode: "0"=parasite power, "1"=external power supply provided.

READ/WRITE TIME SLOTS
DS1820 data is read and written through the use of time slots to manipulate bits and a command word to specify the transaction.

Write Time Slots
A write time slot is initiated when the host pulls the data line from a high logic level to a low logic level. There are two types of write time slots: Write One time slots and Write Zero time slots. All write time slots must be a minimum of 60 µs in duration with a minimum of a one µs recovery time between individual write cycles.

The DS1820 samples the I/O line in a window of 15 µs to 60 µs after the I/O line falls. If the line is high, a Write One occurs. If the line is low, a Write Zero occurs (see Figure 12).

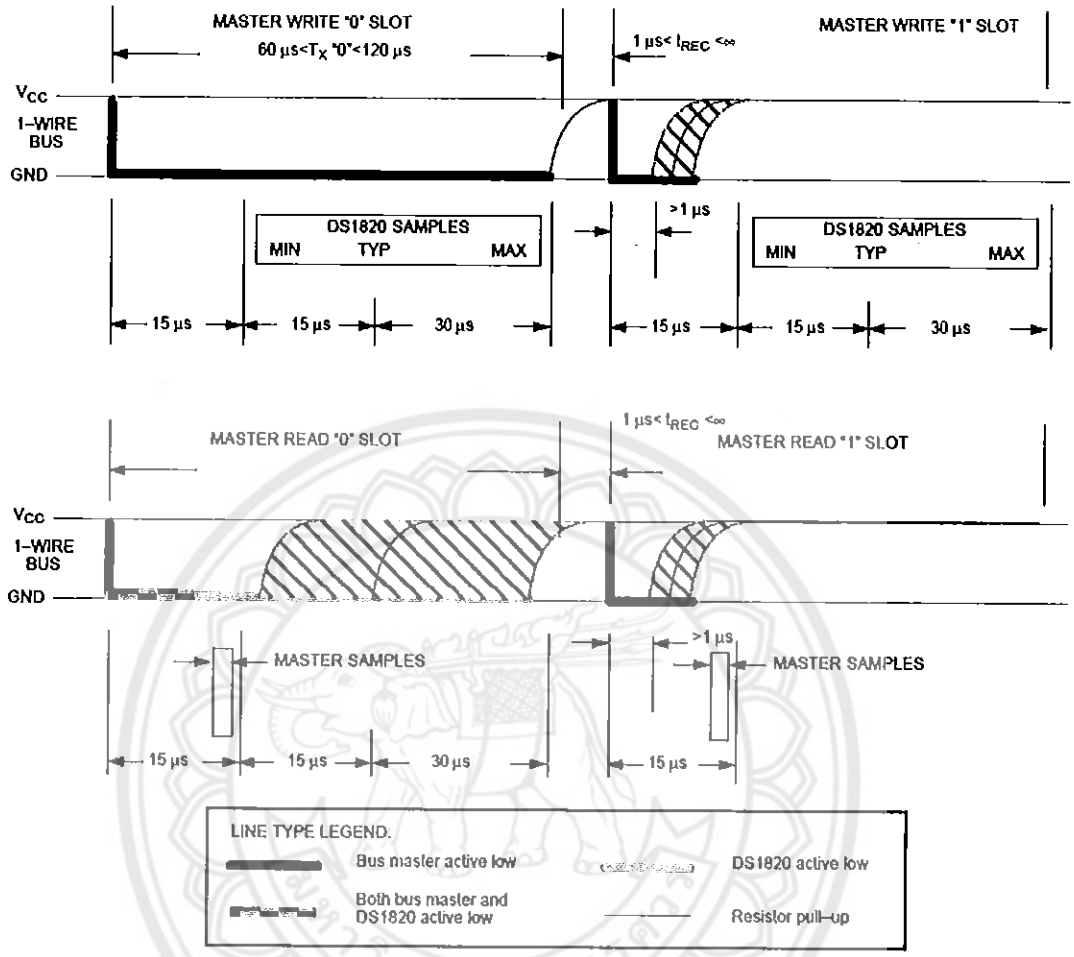
For the host to generate a Write One time slot, the data line must be pulled to a logic low level and then released, allowing the data line to pull up to a high level within 15 µs after the start of the write time slot.

For the host to generate a Write Zero time slot, the data line must be pulled to a logic low level and remain low for 60 µs.

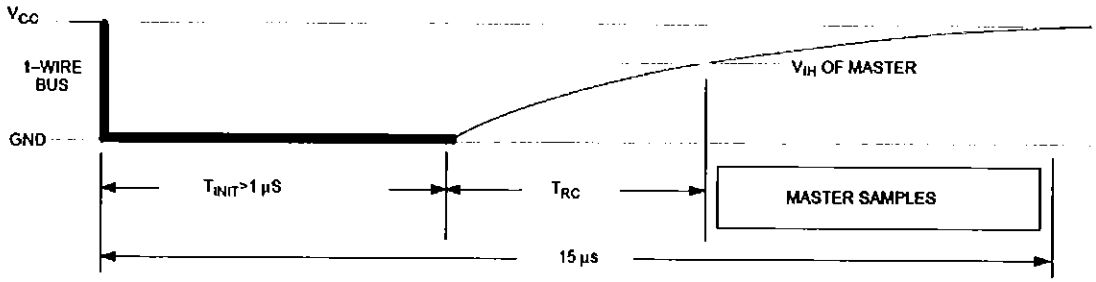
Read Time Slots
The host generates read time slots when data is to be read from the DS1820. A read time slot is initiated when the host pulls the data line from a logic high level to logic low level. The data line must remain at a low logic level for a minimum of one µs; output data from the DS1820 is valid for 15 µs after the falling edge of the read time slot. The host therefore must stop driving the I/O pin low in order to read its state 15 µs from the start of the read slot (see Figure 12). By the end of the read time slot, the I/O pin will pull back high via the external pull–up resistor. All read time slots must be a minimum of 60 µs in duration with a minimum of a one µs recovery time between individual read slots.

Figure 13 shows that the sum of T_{INIT}, T_{RC}, and T_{SAMPLE} must be less than 15 µs. Figure 14 shows that system timing margin is maximized by keeping T_{INIT} and T_{RC} as small as possible and by locating the master sample time towards the end of the 15 µs period.

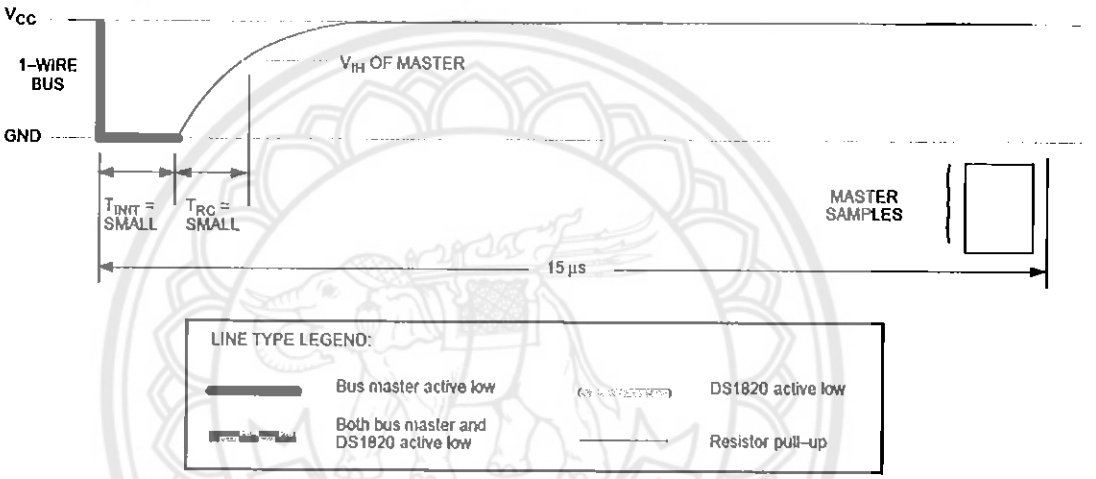
READ/WRITE TIMING DIAGRAM Figure 12



DETAILED MASTER READ "1" TIMING Figure 13



RECOMMENDED MASTER READ "1" TIMING Figure 14



MEMORY FUNCTION EXAMPLE Table 3
Example: Bus Master initiates temperature conversion, then reads temperature (parasite power assumed).

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Reset pulse (480–960 µs).
RX	Presence	Presence pulse.
TX	55h	Issue "Match ROM" command.
TX	<64–bit ROM code>	Issue address for DS1820.
TX	44h	Issue "Convert T" command.
TX	<I/O LINE HIGH>	I/O line is held high for at least 500 ms by bus master to allow conversion to complete.
TX	Reset	Reset pulse.
RX	Presence	Presence pulse.
TX	55h	Issue "Match ROM" command.
TX	<64–bit ROM code>	Issue address for DS1820.
TX	BEh	Issue "Read Scratchpad" command.
RX	<9 data bytes>	Read entire scratchpad plus CRC; the master now recalculates the CRC of the eight data bytes received from the scratchpad, compares the CRC calculated and the CRC read. If they match, the master continues; if not, this read operation is repeated.
TX	Reset	Reset Pulse.
RX	Presence	Presence pulse, done.

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.5V to +7.0V
Operating Temperature	–55°C to +125°C
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

RECOMMENDED DC OPERATING CONDITIONS

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{DD}	I/O Functions	2.8	5.0	5.5	V	1, 2
		±½°C Accurate Temperature Conversions	4.3		5.5		
Data Pin	I/O		–0.5		+5.5	V	2
Logic 1	V_{IH}		2.0		V_{CC}+0.3	V	2, 3
Logic 0	V_{IL}		–0.3		+0.8	V	2, 4

DC ELECTRICAL CHARACTERISTICS

(–55°C to +125°C; V_{DD}=3.6V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Thermometer Error	t_{ERR}	–0°C to +70°C			±½	°C	
		–55°C to 0°C and +70°C to +125°C		See Typical Curve			1, 9, 10
Input Logic High	V_{IH}		2.2		5.5	V	2, 3
Input Logic Low	V_{IL}		–0.3		+0.8	V	2, 4
Sink Current	I_L	$V_{I/O}$=0.4V	–4.0			mA	2
Standby Current	I_Q			200	350	nA	8
Active Current	I_{DD}			1	1.5	mA	5, 6
Input Load Current	I_L			5		µA	7

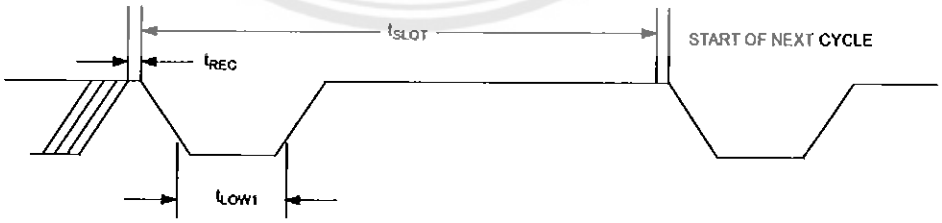
AC ELECTRICAL CHARACTERISTICS: (–55°C to +125°C; V_{DD}=3.6V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Temperature Conversion Time	t_{CONV}		200	500	ms	
Time Slot	t_{SLOT}	60		120	µs	
Recovery Time	t_{REC}	1			µs	
Write 0 Low Time	t_{LOW0}	60		120	µs	
Write 1 Low Time	t_{LOW1}	1		15	µs	
Read Data Valid	t_{RDV}			15	µs	
Reset Time High	t_{RSTH}	480			µs	
Reset Time Low	t_{RSTL}	480		4800	µs	
Presence Detect High	t_{PDHIGH}	15		60	µs	
Presence Detect Low	t_{PDLOW}	60		240	µs	
Capacitance	$C_{IN/OUT}$			25	pF	

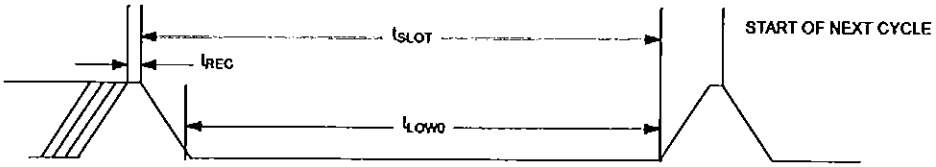
NOTES:

1. Temperature conversion will work with ±2°C accuracy down to V_{DD} = 3.4 volts.
2. All voltages are referenced to ground.
3. Logic one voltages are specified at a source current of 1 mA.
4. Logic zero voltages are specified at a sink current of 4 mA.
5. I_{DD} specified with V_{CC} at 5.0 volts.
6. Active current refers to either temperature conversion or writing to the E^2 memory. Writing to E^2 memory consumes approximately 200 µA for up to 10 ms.
7. Input load is to ground.
8. Standby current specified up to 70°C. Standby current typically is 5 µA at 125°C.
9. See Typical Curve for specification limits outside the 0°C to 70°C range. Thermometer error reflects sensor accuracy as tested during calibration.
10. Typical accuracy curve valid for 4.3V ≤ V_{DD} ≤ 5.5V.

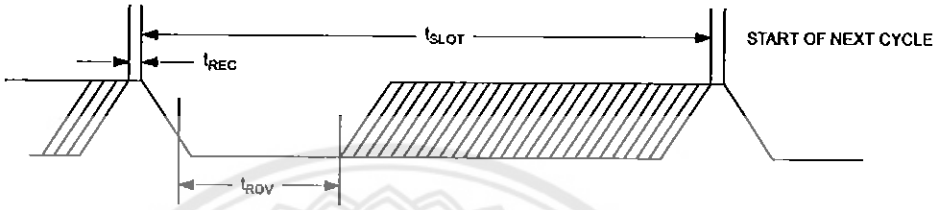
1–WIRE WRITE ONE TIME SLOT



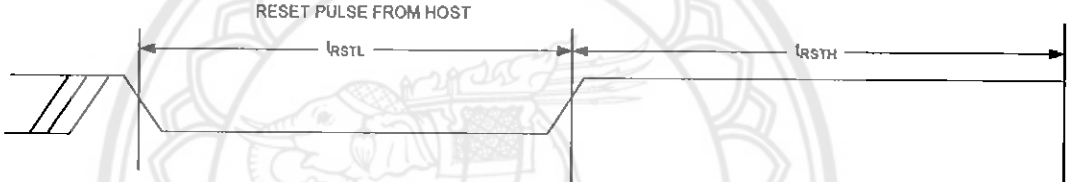
1–WIRE WRITE ZERO TIME SLOT



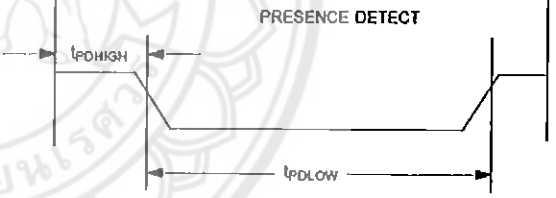
1–WIRE READ ZERO TIME SLOT



1–WIRE RESET PULSE

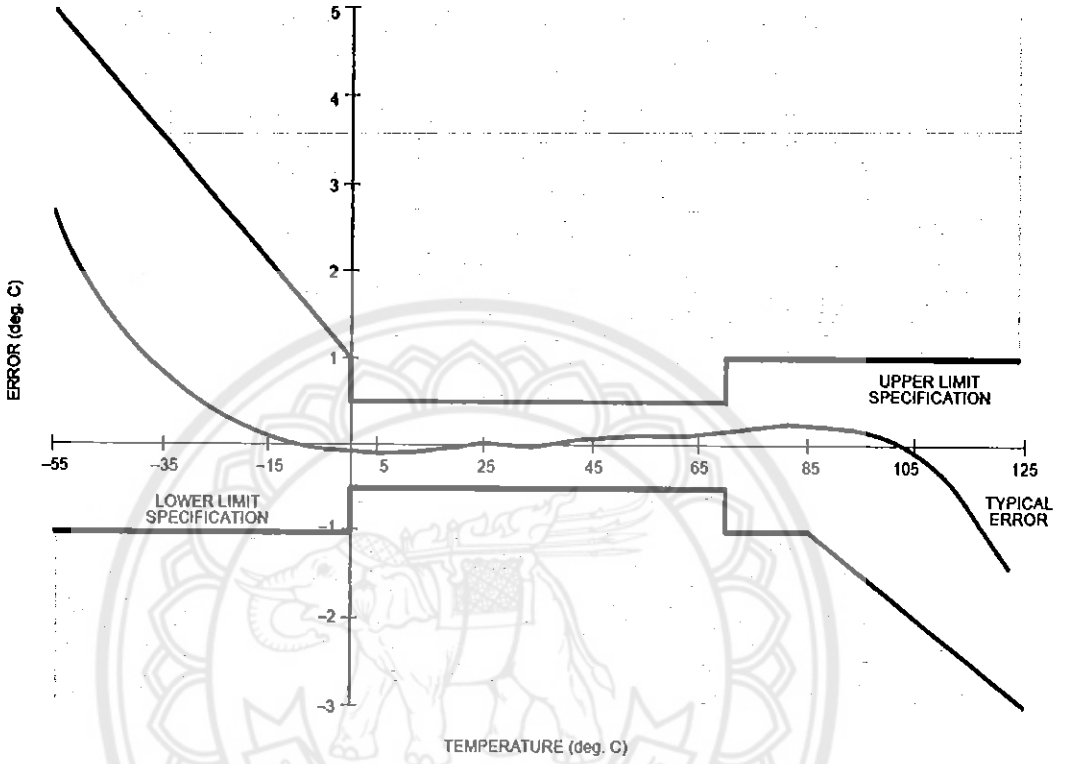


1–WIRE PRESENCE DETECT



TYPICAL PERFORMANCE CURVE

DS1820 DIGITAL TERMOMETER AND THERMOSTAT
TEMPERATURE READING ERROR



ประวัติผู้ดำเนินโครงงาน

ชื่อ นายพิเชษฐ์ เป็งขัน

ภูมิลำเนา 148 หมู่ 4 ต.บ้านหลวง อ.จอมทอง จ.เชียงใหม่

ประวัติการศึกษา

- จบระดับมัธยมศึกษาตอนปลายจากโรงเรียนจอมทอง จ.เชียงใหม่
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 8 สาขาวิศวกรรมไฟฟ้าคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail: **pichet.ee50@gmail.com**

ชื่อ นายบุญญฤทธิ์ กล้วยน้อย

ภูมิลำเนา 389/3 ต.แม่ต๋ำ อ.เมือง จ.พะเยา

ประวัติการศึกษา

- จบระดับมัธยมศึกษาตอนปลายจากโรงเรียนพุขามครุฑมณีอุทิศ จ.เพชรบูรณ์
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี ชั้นปีที่ 8 สาขาวิศวกรรมไฟฟ้าคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail: **stunmaster@hotmail.co.th**