การควบคุมการทำงานเครื่องปรับอากาศ โดยใช้ไมโครโปรเซสเซอร์

AIR – CONDITIONER CONTROL CIRCUIT BY MICROPROCESSOR

โดย

นาย	วีระ	จุฑารัตน์	32.1321
นาย	สุเธียร	ศิริกุลยดี	32.1401

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ปริญญานิพนธ์ปีการศึกษา 2535

ภาควิชา วิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมการทำงานเครื่องปรับอากาศโดยใช้ไมโครโปรเซสเซอร์

ผู้จัดทำ

1. นาย วีระ จุฑารัตน์
2. นาย สุเธียร ศิริกุลบดี

(อาจารย์ที่ปรึกษา)

(อาจารย์ เกียรติวรรณ ทรงสัตย์)

การควบคุมการทำงานของเครื่องปรับอากาศโดยใช้ไมโครโปรเซสเซอร์

นาย วีระ จุฑารัตน์ 32.1321

นาย สุเธียร ศิริกุลบดี 32.1401

อาจารย์ที่ปรึกษา

อาจารย์ เกียรติวรรณ ทรงสัตย์

บทคัดย่อ

โครงงานนี้เป็นการศึกษาเกี่ยวกับวงจรควบคุมการทำงานของเครื่องปรับอากาศแบบ on-off control โดยจะเน้นไปที่การควบคุมการทำงานของคอมเพรสเซอร์ เพื่อปรับระดับของอุณหภูมิตามที่ผู้ใช้ต้องการ และเพื่อป้องกันความเสียหายของตัวคอมเพรสเซอร์เอง ซึ่งวงจรควบคุมนี้จะถูกแบ่งเป็นวงจรย่อย ๆ ซึ่งทำหน้าที่ตรวจสอบสภาวะต่าง ๆ เช่น ระดับอุณหภูมิห้องแรงดันไฟบ้าน โดยจะใช้ไมโครโปรเซสเซอร์ 8751 เป็นตัวรับสถานที่วงจรย่อยส่งมาเพื่อเป็นข้อมูลในการทำงานตามโปรแกรมที่ได้ถูกบรรจุไว้ภายใน จะเป็นผลให้ตัวคอมเพรสเซอร์ทำงานเพื่อปรับระดับอุณหภูมิได้ตามต้องการ และเป็นการป้องกันความเสียหายที่อาจเกิดขึ้นกับตัวคอมเพรสเซอร์เองได้

AIR-CONDITIONER CONTROL CIRCUIT BY MICROPROCESSOR

BY

MR. WEERA JUTARUTNA

MR. SUTEIN SIRIKULBODEE

ADVISOR

MR. KEATTIWAN SONGSATYA

1992

Abstract

This project is to study the function of an on-off control Air-Conditioner control curcuit, in particular, to study the way it controls the operation of the compressor so that temperature is kept at the point the user desires and the way it protect the compressor as well. The control circuit composed of subcircuits which are assigned to inspect such conditions as room temperature level, household voltage, etc, uses a microprocessor 8751 to act as the main part to receive deta and react as is programmed in itself. A seperate circuit is used to constantly check the status of the microprocessor, So that the compressor will regulate the temperature and be protected from being damaged as mentioned above.

กิตติกรรมประกาศ

ขอขอบคุณ อาจารย์ เกียรติวรรณ ทรงสัตย์ เป็นอย่างยิ่งที่ท่านได้ให้คำปรึกษาและคำแนะนำต่าง ๆ ในการทำงาน อีกทั้งสนับสนุนในด้านเครื่องมือ วัสดุอุปกรณ์ เพื่อให้การทำงานได้รับความสะดวก ขอขอบคุณ อาจารย์ ปรเมษฐ์ ประนยานันทน์ ที่ได้ให้คำปรึกษาเกี่ยวกับโปรแกรมการทำงานของ 8031 ซึ่งทำให้ผู้เขียนมีความคิดในการนำมาแก้ปัญหาในการทำงานได้อย่างดียิ่งขอขอบคุณ คุณพ่อ คุณแม่ ที่คอยเป็นห่วงลูก ขอบคุณเพื่อน ๆ น้อง ๆ ที่ให้กำลังใจ

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1

บทนำ

เครื่องปรับอากาศเป็นอุปกรณ์ไฟฟ้าเพื่อการอำนวยความสะดวกสบาย มีความจำเป็นมากในปัจจุบัน เนื่องจากประเทศไทยเป็นประเทศร้อน เครื่องปรับอากาศจะช่วยให้สภาพอากาศภายในที่ทำงานหรือที่อยู่อาศัยมีความเย็นสบายไม่ร้อนอบอ้าว ในบางสถานที่ที่มีความจำเป็นอย่างยิ่งที่จะต้องใช้เครื่องปรับอากาศ เช่น ในศูนย์คอมพิวเตอร์ ซึ่งมีคอมพิวเตอร์จำนวนมาก ซึ่งถ้าขาดระบบปรับอากาศเสียแล้ว เครื่องคอมพิวเตอร์เหล่านี้อาจจะทำงานได้ไม่ดีเท่าที่ควร เครื่องปรับอากาศโดยทั่ว ๆ ไปจะประกอบด้วยส่วนสำคัญ 4 ส่วนคือ

1. คอยล์เย็น (EVAPORATOR) มีลักษณะเป็นท่อขดอยู่ภายในห้อง ทำหน้าที่เป็นตัวความเย็น

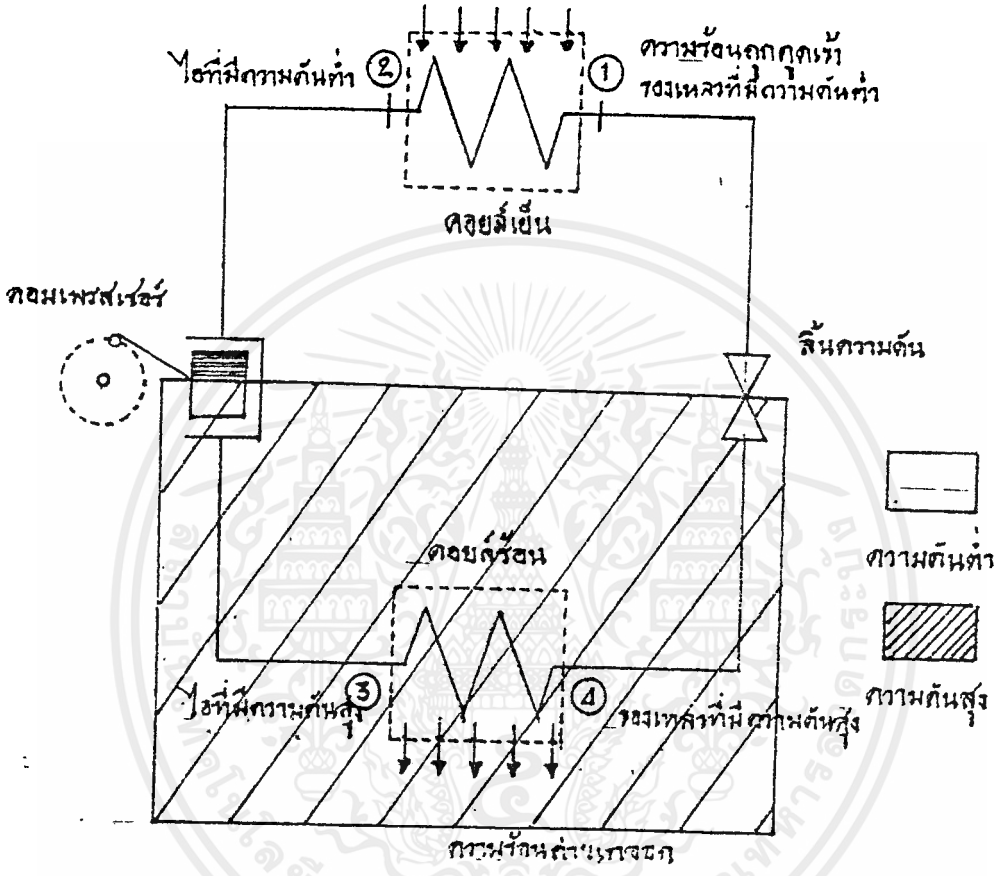
2. คอยล์ร้อน (CONDENSOR) มีลักษณะเป็นท่อขดอยู่ภายนอกห้อง ทำหน้าที่เป็นตัวระบายความร้อน

3. ลิ้นลดความดัน (EXPANSION VALVE) ทำหน้าที่ในการปรับระดับความดันของของเหลวภายในท่อให้ลดลง

4. คอมเพรสเซอร์ (COMPRESSOR) ทำหน้าที่ เป็นตัวดูดและอัดของเหลวภายในท่อให้ได้ความดันตามต้องการ และ เป็นตัวทำให้ของเหลวภายในท่อไหลวนในปริมาณที่ต้องการ

ในส่วนประกอบที่สำคัญของเครื่องปรับอากาศทั้ง 4 ส่วน ตัวคอมเพรสเซอร์ถือเป็นส่วนที่สำคัญที่สุดเนื่องจากคุณภาพการทำงานและอายุการใช้งานของเครื่องปรับอากาศขึ้นอยู่กับส่วนสำคัญส่วนนี้เป็นส่วนใหญ่ ในขณะที่ตัวคอมเพรสเซอร์ทำงาน บางครั้งอาจจะมีปัจจัย

ภายนอกที่มีผลรกะทบทำให้ตัวคอมเพรสเซอร์เสื่อมสภาพเนื่องจากลักษณะการทำงานที่ผิดปกติ เช่น ลักษณะที่เกิดการไฟตกไฟเกินหรือลักษณะของไฟที่ตับและเปิดขึ้นมาใหม่กระทันหันในช่วงเวลาสั้น ๆ สิ่งเหล่านี้ทำให้ตัวคอมเพรสเซอร์เกิดความเสียหายหรือเสื่อมสภาพได้ทั้งสิ้น



รูปที่ 1.1 ส่วนประกอบที่สำคัญของเครื่องปรับอากาศ

เพื่อป้องกันและรักษาตัวคอมเพรสเซอร์ไว้ให้มีอายุการใช้งานที่ยืนนาน จึงมีความจำเป็นที่ เครื่องปรับอากาศควรจะมีวงจรเพื่อควบคุมการทำงานของคอมเพรสเซอร์ นอกจากนี้วงจรนี้ยังสามารถที่จะควบคุมระดับ speed ของพัดลมเพื่อให้อุณหภูมิของห้องเป็นไปตามที่ผู้ใช้ต้องการ ซึ่งจะทำให้ได้รับความสะดวกสบายยิ่งขึ้นในการใช้เครื่องปรับอากาศ ดังนั้นสามารถที่จะสรุปประโยชน์ของวงจรควบคุมในการทำโครงงานได้หลัก ๆ ดังนี้คือ

1. เพื่อป้องกันและรักษาสภาพของตัวคอมเพรสเซอร์ให้มีอายุการใช้งานยาวนาน

2. เพื่อควบคุมระดับของอุณหภูมิภายในห้องให้เป็นไปตามที่เราต้องการ

โดยวงจรควบคุมในโครงงานนี้จึงออกแบบมาเพื่อใช้งานกับเครื่องปรับอากาศที่ทำงานแบบ on-off control ซึ่งเราจะได้ศึกษาตัววงจรและทำความเข้าใจหลักการทำงานของวงจรต่างๆ ในบทต่อ ๆ ไป

บทที่ 2

วงจรควบคุมการทำงานของเครื่องปรับอากาศ

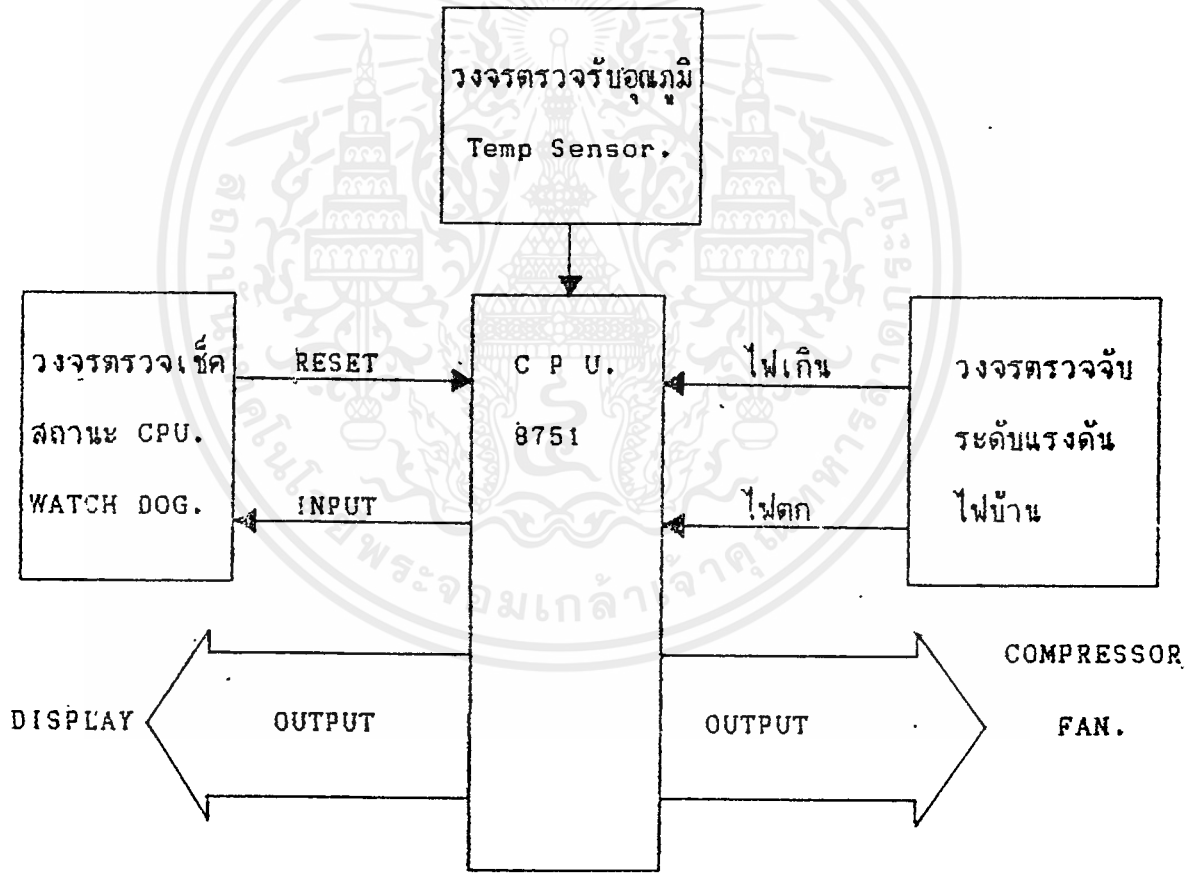
ส่วนประกอบของวงจรควบคุมการทำงานของเครื่องปรับอากาศ ประกอบไปด้วยวงจรย่อยซึ่งทำหน้าที่ในแต่ละอย่างแยกกันออกไป โดยมีศูนย์กลางการควบคุมการทำงานอยู่ที่ C.P.U.8751 วงจรย่อยเหล่านี้จะมีหน้าที่ในการปรับสถานะทางกายภาพเป็นสัญญาณเพื่อมาป้อนเป็น input ให้กับ C.P.U. เพื่อดำเนินงานตามโปรแกรมที่ได้บรรจุไว้ภายใน เช่นสถานะของระดับอุณหภูมิจริงของห้อง สถานะของระดับแรงดันไฟบ้าน เป็นต้น C.P.U. จะรับรู้สถานะดังกล่าวและดำเนินตามโปรแกรมและจะส่งสัญญาณ output ออกไปควบคุมการทำงานของคอมเพรสเซอร์หรือ พัดลม ให้เป็นไปตามที่เราต้องการ นอกจากนี้ยังสามารถที่จะส่ง output ออกไปยัง Display ซึ่งจะสามารถที่จะบอกถึงสถานะในขณะนั้นได้ทำให้ผู้ใช้ทราบถึงสภาวะต่าง ๆ ได้อย่างง่ายได้ วงจรควบคุมเครื่องปรับอากาศสามารถพิจารณาได้ดังรูปที่ 2.1

จากรูปที่ 2.1 จะเห็นว่าวงจรจะประกอบไปด้วย

1. C.P.U 8751 ทำหน้าที่เป็นศูนย์กลางควบคุมการทำงานทั้งหมด
2. วงจร WATCH DOG. เป็นวงจรที่ใช้ในการตรวจเช็คสภาวะการทำงานของ C P U. และทำหน้าที่ในการ Reset cpu. ในกรณีเกิดการผิดปกติของการทำงาน
3. วงจร TEMPERATURE SENSOR (วงจรตรวจจับอุณหภูมิ) ทำหน้าที่ในการตรวจวัดอุณหภูมิห้องและส่งเป็นสัญญาณเป็นข้อมูลให้กับ cpu.

4. วงจรตรวจจับระดับความดันไฟบ้าน เป็นวงจรที่ใช้ในการวัดระดับไฟตกไฟเกินของไฟบ้าน เมื่อเกิดไฟตกหรือไฟเกิน วงจรที่จะส่งสัญญาณไปให้C.P.U. ทำการติด COMPRESSOR

5. DISPLAY เป็นวงจรที่ใช้ในการแสดงสถานะต่าง ๆ โดย C.P.U. จะเป็นตัวส่ง OUTPUT ไปให้



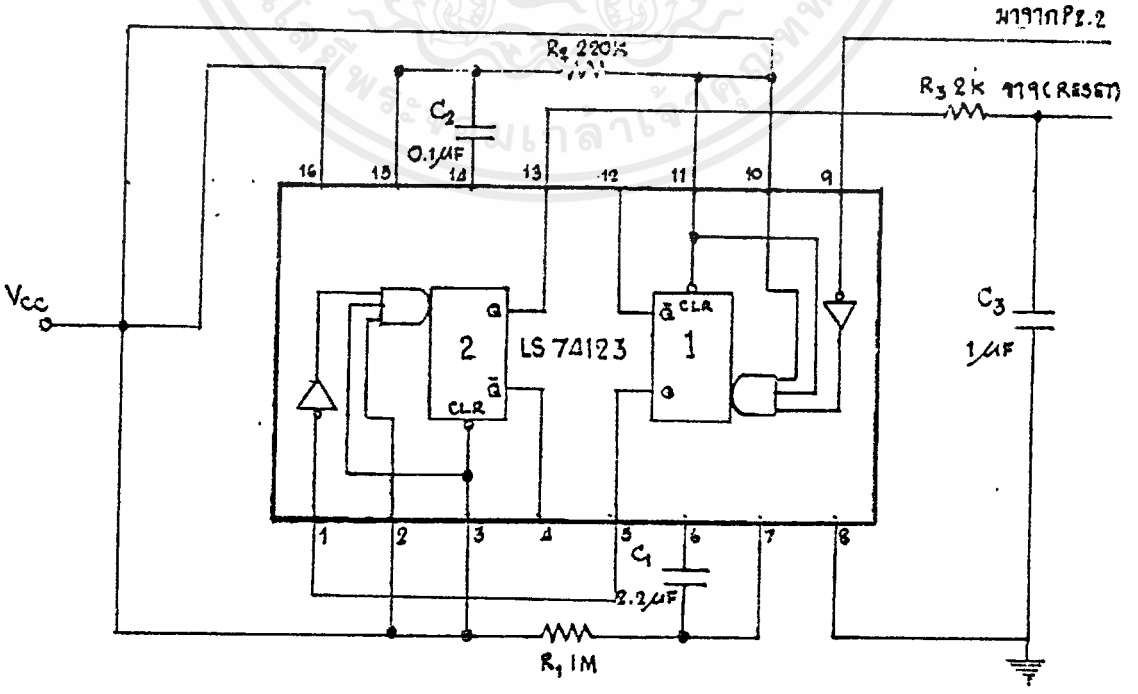
รูปที่ 2.1 แสดงส่วนประกอบของวงจรควบคุมเครื่องปรับอากาศ

วงจรย่อยเหล่านี้จะถูกต่อเข้ากับ C.P.U. 8751 ซึ่งใช้เป็นตัวควบคุมการทำงานทั้งหมด (SINGLE SHIP CONTROLO ของวงจร คุณสมบัติสำคัญของ SHIP 8751 เป็น SHIP ที่มี EPROM ในตัว มีคำสั่งพิเศษสามารถที่จะพัฒนาโปรแกรมได้ง่ายและกว้างขวาง ไม่จำเป็นต้องใช้ MEMORY ภายนอก จึงเหมาะกับวงจรที่ต้องการความกระทัดรัด

บทที่ 3

วงจรตรวจเช็คสถานะ C.P.U. (WATCH DOG)

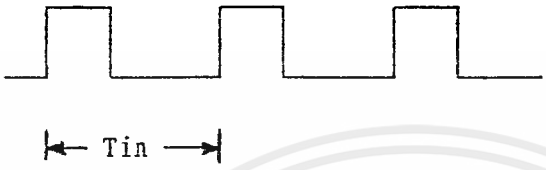
เป็นวงจรตรวจสอบสถานะการทำงานของ C.P.U. เนื่องจากในขณะที่วงจรทำงานอยู่อาจเกิดผิดพลาดในการทำงานของวงจร หรือ อาจเกิดการ HANG ของโปรแกรมขึ้น ทำให้ C.P.U ไม่สามารถควบคุมการทำงานได้อย่างถูกต้อง ซึ่งอาจทำให้ตัวคอมเพรสเซอร์ทำงานผิดปกติอาจจะก่อให้เกิดความเสียหายขึ้นได้ ดังนั้นวงจรควบคุมการทำงานของเครื่องปรับอากาศนี้จึงจำเป็นต้องมีวงจร WATCH DOG. เพื่อทำหน้าที่ในการตรวจสอบสถานะของ 8751 ระหว่างการดำเนินงานตามโปรแกรม เมื่อเกิดข้อผิดพลาด หรือ เกิดการ HAHG ของโปรแกรม วงจร WATCH DOG. สามารถที่จะรับสัญญาณผิดปกติจาก 8751 ได้และจะสร้างสัญญาณขึ้นเพื่อไป RESET 8751 ให้เริ่มทำงานในสถานะเริ่มต้น (INITIAL CONDITION) ใหม่ จึงทำให้โปรแกรมสามารถที่จะดำเนินต่อไปอย่างต่อเนื่องวงจร WATCH DOG. ในที่นี้จะใช้ IC 74123 ในการตรวจสอบสัญญาณ วงจร WATCH DOG. นี้จะมีลักษณะดังรูปที่ 3.1



รูปที่ 3.1 วงจรตรวจเช็คสถานะ C P U (WATCH DOG)

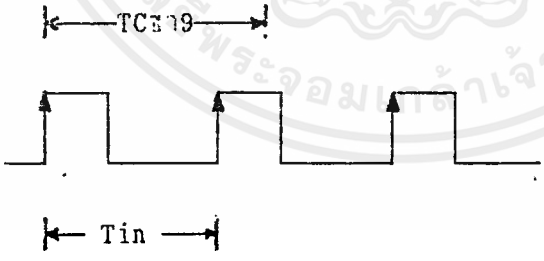
หลักการทำงานของ WATCH DOG.

ใช้ IC เบอร์ 74123 ภายในประกอบด้วย MONOSTABLE 2 ตัว ใช้แบบ Retriggerable Monostable หลักการทำงานจะมีการส่งสัญญาณมาจาก CPU. เข้ามาที่ขา 9 ซึ่งเป็น INPUT ของ monostable ตัวที่ 1 ดังรูปที่ 3.2



รูปที่ 3.2 แสดงลักษณะสัญญาณที่ส่งมาจาก CPU.

การรับสัญญาณนั้นจะสร้างค่า TIME CONSTANT (TC) จากวงจรจะได้ค่า TC = 0.37 R_1C_1 ซึ่งมีค่าเท่ากับ 0.814 sec. ในการรับสัญญาณจะให้คาบเวลามีค่ามากกว่า Tin (คาบเวลาของสัญญาณที่ส่งจาก CPU) เพื่อรับสัญญาณ Trig ขาลงของสัญญาณจาก CPU ดังรูปที่ 3.3

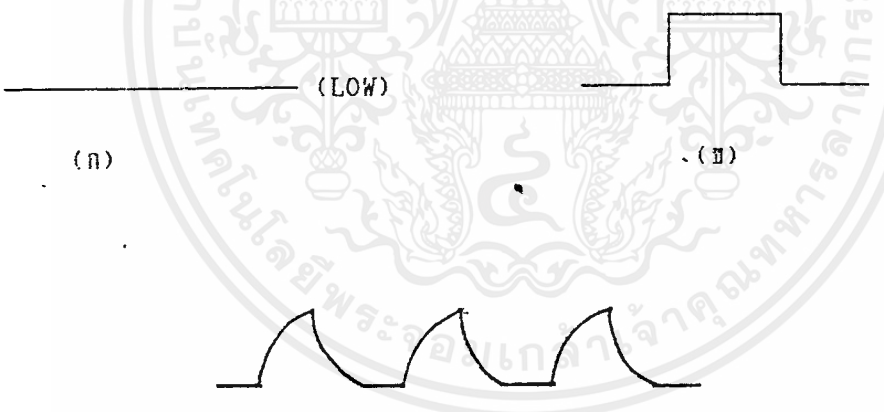


รูปที่ 3.3 แสดง TIME CONSTANT ของขา 9

เมื่อรับสัญญาณ TRIG ขาขึ้นจาก CPU MONOSTABLE ตัวแรกจะส่งค่าออกที่ขา 5 (OUTPUT Q) เป็นสถานะ high ไปยัง MONOSTABLE ตัวที่ 2 ที่ขา 1

ซึ่งเป็น INPUT โดยจะมี TIME CONSTANT = TC 0.37 R_2C_2 ซึ่งมีค่าเท่ากับ 0.00814 sec จนกระทั่งเมื่อเกิดสภาวะผิดปกติขึ้นที่ cpu ทำสัญญาณที่ส่งไปยังขา 9 ขาดหายไปเป็นเวลามากกว่า TIME CONSTANS ที่กำหนดไว้ MONOSTABLE ตัวแรก จะส่งสถานะ low ไปยัง MONOSTABLE ตัวที่ 2 ที่ขา 1 โดยจะมีลักษณะของสัญญาณดังรูปที่ 3.4

รูปที่ 3.4 ลักษณะสัญญาณที่ออกมา Trig ขา 1



(ค)

รูปที่ 3.5 ลักษณะที่ส่งไป reset cpu จากขา 13

ก. สัญญาณขณะ cpu อยู่ในสภาวะปกติ ข. สัญญาณที่ขา 13 ขณะ cpu ทำงานผิดปกติ

ค. สัญญาณจากขา 13 ที่ผ่าน RC แล้ว

เนื่องจากขา 1 ต้องการสัญญาณ Trig ขอบขาลง (↓) เพื่อจะสร้างสัญญาณออกไป reset cpu ทางขา 13 ผ่าน R_3 และ C_3 ไปยังขา 9 (reset) ของ cpu มีลักษณะของสัญญาณดังกล่าว ตามรูปที่ 3.5

บทที่ 4

วงจรตรวจจับค่าของอุณหภูมิ (TEMPERATURE SENSOR)

เป็นวงจรสำหรับจับค่าอุณหภูมิที่แท้จริงของห้องในขณะนั้น เพื่อเป็นข้อมูลสำหรับโปรแกรมของ 8751 ในการที่จะควบคุมการทำงานของคอมเพรสเซอร์และพัดลม เพื่อให้มีค่าอุณหภูมิตามที่เราได้ตั้งไว้ วงจร SENSOR จะมีเทอร์มิสเตอร์เป็นส่วนสำคัญ เทอร์มิสเตอร์ที่ใช้เป็นแบบ NEGATIVE COEFFICIENT คือเมื่ออุณหภูมิสูงความต้านทานจะต่ำ และเมื่อเมื่ออุณหภูมิต่ำความต้านทานจะสูง เราใช้เทอร์มิสเตอร์เป็นตัวตรวจจับอุณหภูมิ โดยใช้ IC 555 ประกอบเป็นวงจร MONOSTABLE โดยค่า TIME CONSTANT จะมีค่าตัวเดียวกับเทอร์มิสเตอร์ R_T, R_t, C_t โดยที่สัญญาณ TRIG MONOSTABLE จะถูกส่งมาจาก cpu อย่างต่อเนื่อง ทำให้ความกว้างของสัญญาณ MONOSTABLE จะแปรผันตามอุณหภูมิที่อยู่รอบตัวเทอร์มิสเตอร์ จากนั้นเราจะใช้ cpu ในการส่งวัดความกว้างของสัญญาณพัลซ์ เพื่อแปลงค่าดังกล่าวออกมาเป็นอุณหภูมิเพื่อจะใช้ในการควบคุมอุณหภูมิ และแสดงผลให้ผู้ใช้ทราบ

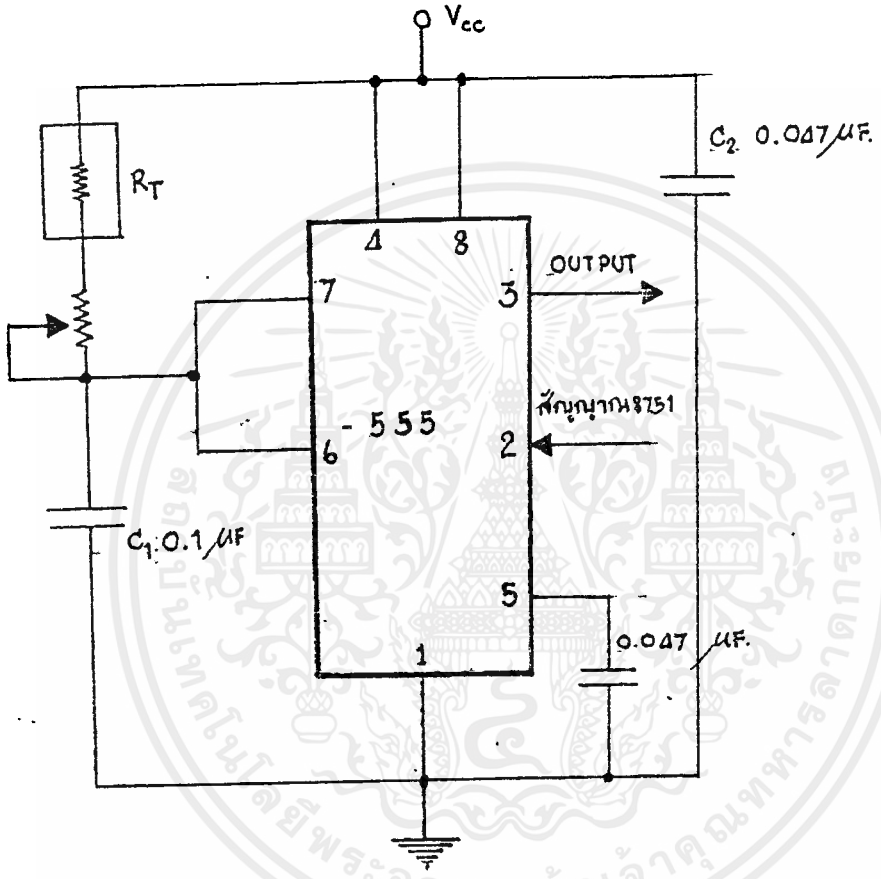
หลักการทำงานของวงจร

วงจร TEMPERATURE SENSOR สามารถแบ่งออกได้เป็น 2 ส่วน คือ ส่วนของวงจรและส่วนของโปรแกรม

1. ส่วนวงจร จะใช้วงจรโมโนเสตเบิ้ลของ IC 555 มาสร้างพัลซ์เพื่อนำไปเปรียบเทียบแล้วเปลี่ยนค่าเป็นอุณหภูมิ โดยใช้ความกว้างของพัลซ์เป็นตัวเปรียบเทียบ ซึ่งความกว้างของพัลซ์หาได้จาก

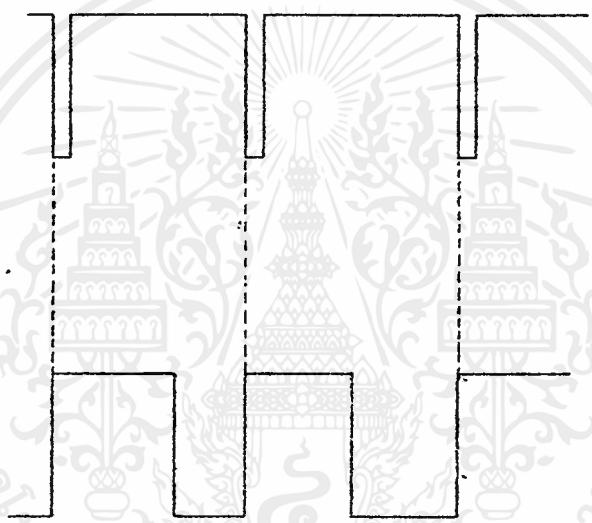
$$T = 1.1\ RC.$$

โดย T = ความกว้างของพัลซ์, C (ตัวเก็บประจุ) มีค่าคงที่ ความกว้างของพัลซ์จะขึ้นอยู่กับความต้านทาน (R) เมื่อเราน่าเทอร์มิสเตอร์ซึ่งเป็นอุปกรณ์ตรวจค่าอุณหภูมิ โดยค่าความต้านจะเปลี่ยนไปตามอุณหภูมิ มาต่อเป็น R ในวงจรดังรูปที่ 4.1 ความกว้างของพัลซ์จะขึ้นกับอุณหภูมิ



รูปที่ 4.1 วงจร 555

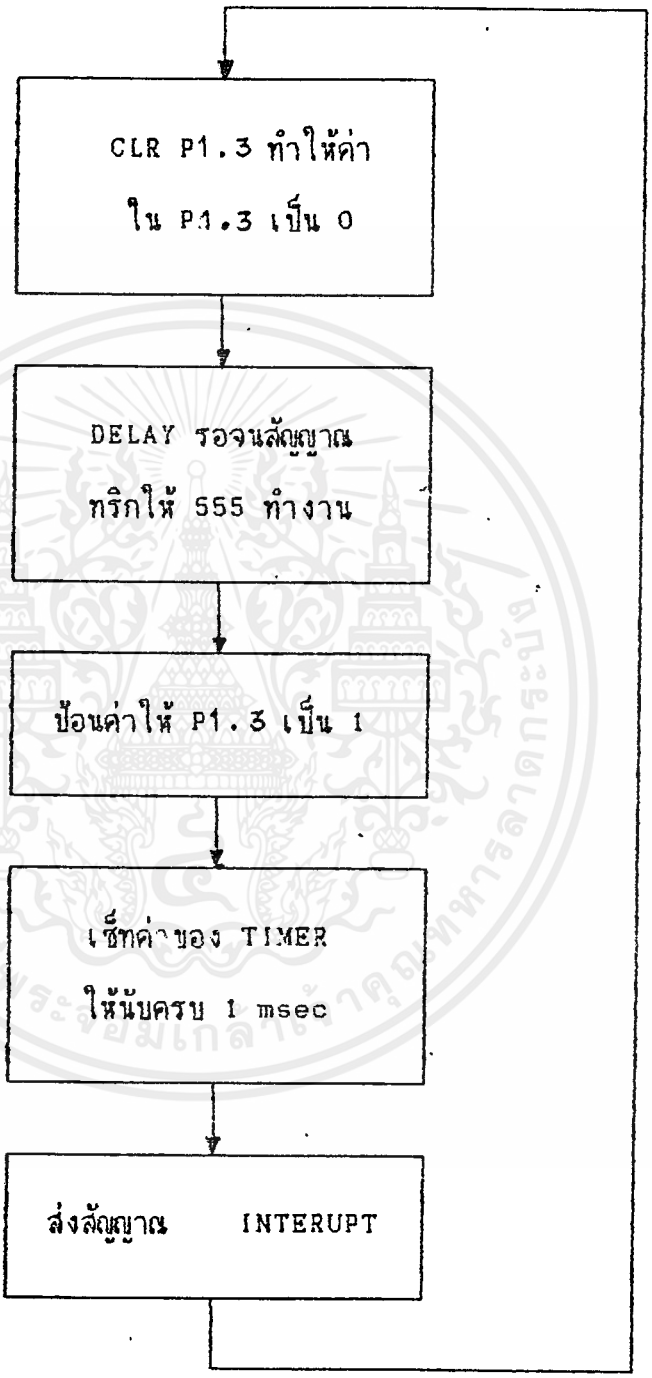
วงจรของ IC 555 จะทำงานเมื่อสัญญาณทริก ที่เปลี่ยนจาก 1 เป็น 0 มาทริก ถ้าเราสร้างสัญญาณทริก เป็นความถี่ ที่ความกว้างของความคงที่จะทริกให้ IC 555 ทำงานดังรูปที่ 4.2



รูปที่ 4.2 ลักษณะของสัญญาณพัลซ์

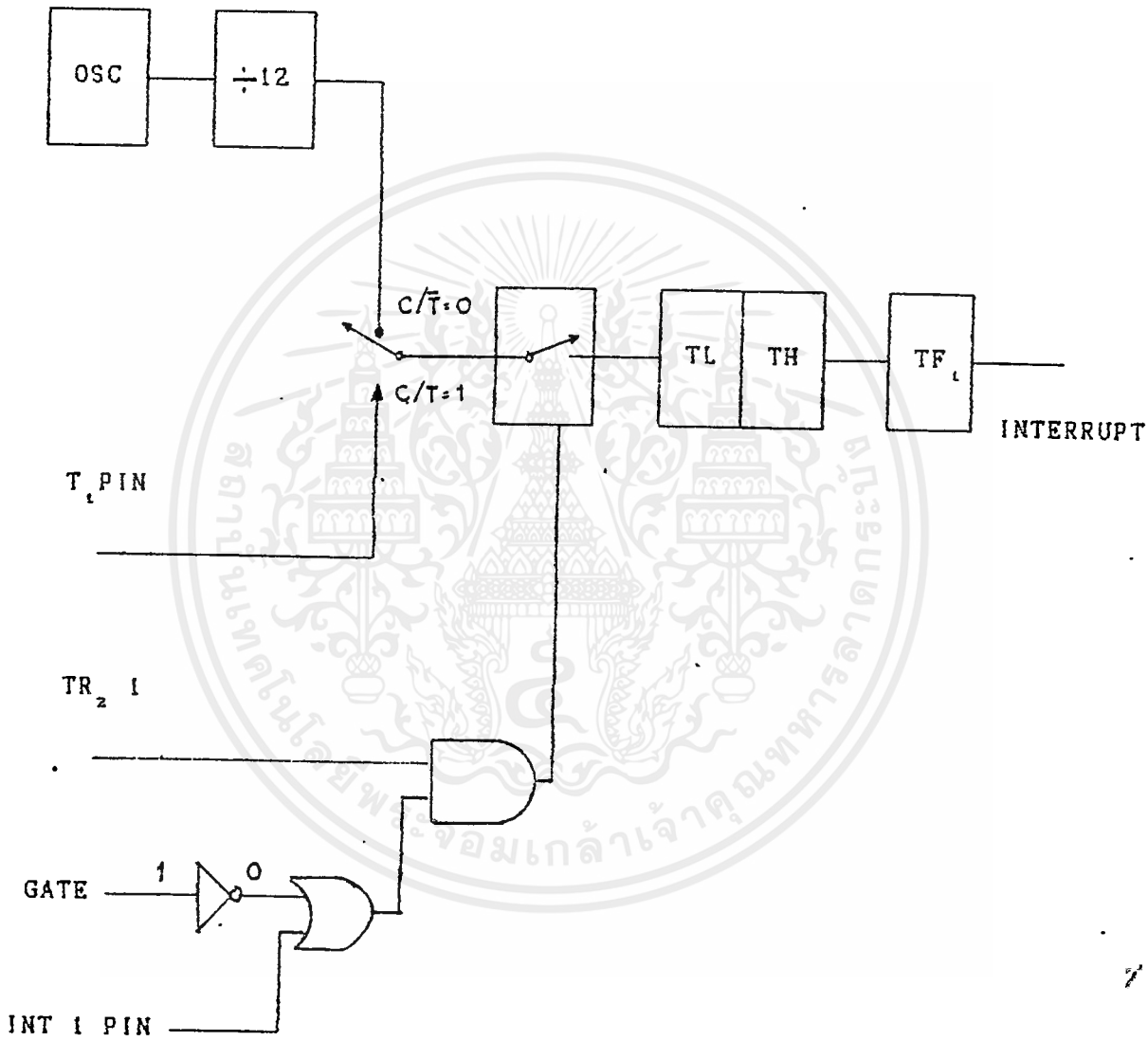
2. ส่วนของโปรแกรมจะแยกได้ 2 โปรแกรมดังนี้

2.1 โปรแกรมการสร้างสัญญาณทริก เราใช้ MCS - 51 ตัว 8751 เป็นตัวควบคุมดังนั้นโปรแกรมที่ใช้จึงเป็นโปรแกรมของ 8751 ก่อนอื่นต้องเลือกพอร์ทที่จะใช้สัญญาณทริก ซึ่งจะเลือก P 2.0 เมื่อเปลี่ยนค่าจาก 1----->0 และ 0----->1 แล้วเราต้องการให้เป็น 1 นาน 1 MSEC ก่อนที่จะเป็น 0 อีกครั้ง จึงใช้ Timer 1 เป็นตัวนับ เมื่อครบเวลา 1 msec จะส่งสัญญาณ INTERRUPT ไปทำงาน เปลี่ยนค่า 1----0 ทำอย่างนี้เรื่อยไป แสดงดัง FLOW CHART ของการสร้างสัญญาณทริก ดังรูปที่ 4.3



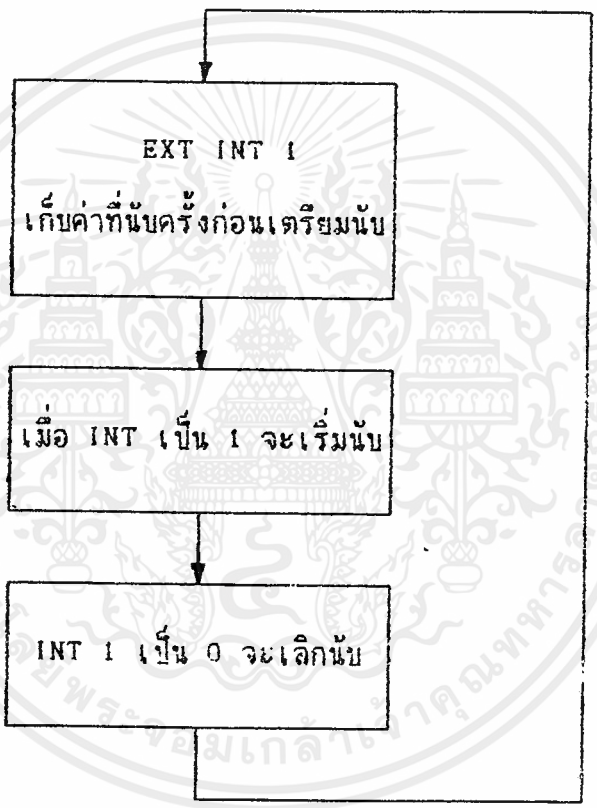
รูปที่ 4.3 FLOW CHART แสดงโปรแกรมการสร้างสัญญาณทริก

2.2 โปรแกรมวัดความกว้างของพัลซ์ จะใช้ Timer counter 1 ในการวัด จากรูปที่ 4.4 เป็นวงจรการทำงานของ Timer Counter 1



รูปที่ 4.4 วงจรการทำงานของ Timer Counter 1

จากรูปถ้าเรากำหนดให้ TR 1 มีค่าเป็น 1 GATE มีค่าเป็น 1 จะทำให้สัญญาณจากขา INT 1 เป็นสัญญาณควบคุมการนับ ถ้าสัญญาณที่ขา INT 1 เป็น 0 จะไม่มีการนับเกิดขึ้นจนเมื่อสัญญาณเป็น 1 จึงจะนับได้ ดัง FLOW CHART รูปที่ 4.5



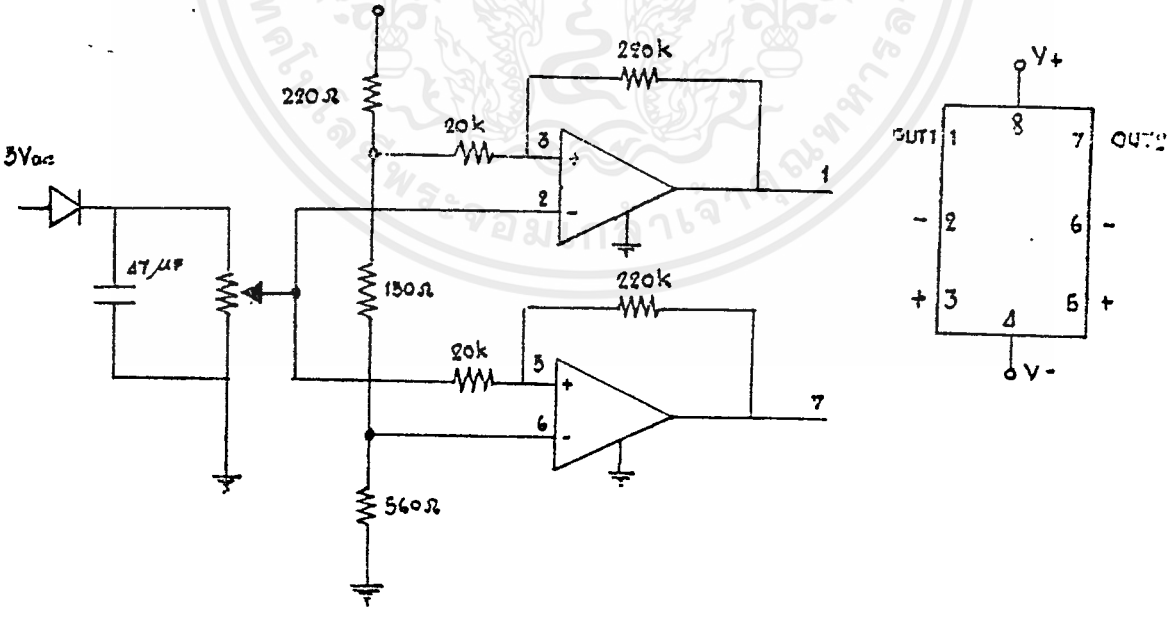
รูปที่ 4.5 FLOW CHART แสดงการนับ

เนื่องจากภาพที่แสดงอุณหภูมิแต่ละคาบใช้เวลาน้อยมาก คือ 1 MSEC ดังนั้นจึงต้องนับหลาย ๆ ครั้ง แล้วนำค่าเฉลี่ยเป็นอุณหภูมิไปเปรียบเทียบ

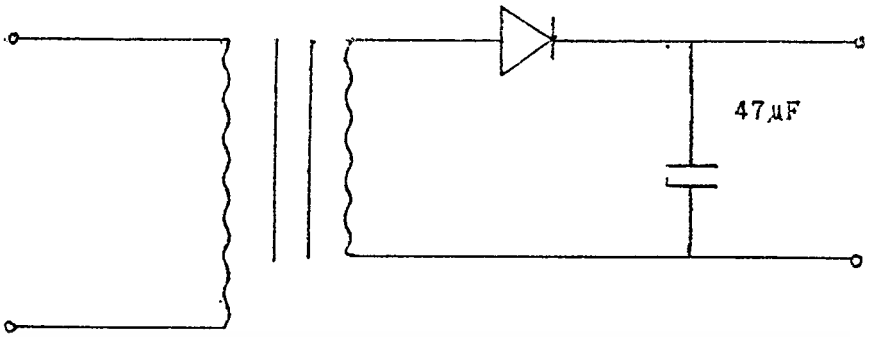
บทที่ 5

วงจรตรวจจับระดับแรงดันไฟฟ้า

เนื่องจากแรงดันไฟฟ้ากระแสสลับ 220 V AC ในแต่ละย่านที่อยู่อาศัย อาจจะมีความไม่คงคที่ในระดับแรงดัน โดยจะขึ้นอยู่กับปัจจัยหลาย ๆ อย่าง เช่นสภาพของสายไฟ ความหนาแน่นของการใช้ไฟฟ้าในแต่ละช่วงเวลา เป็นต้น ซึ่งถ้าแรงดันไฟสลับดังกล่าวมีการเปลี่ยนแปลง สูงขึ้นหรือต่ำลงเกินกว่าที่คอมเพรสเซอร์จะทำงานในสภาวะปกติ (189-250 VAC) เพื่อทำให้คอมเพรสเซอร์มีอายุการใช้งานมากขึ้น หรือลดความเสียหายที่อาจเกิดขึ้นได้ ดังนั้นจึงจำเป็นที่จะต้องมีวงจรเพื่อทำงานตรวจวัดระดับแรงดัน ว่าอยู่ในขอบเขตของแรงดันที่ทำให้คอมเพรสเซอร์ทำงานในสภาวะปกติหรือเป็นอันตรายต่อคอมเพรสเซอร์ โดยวงจรนี้จะใช้ OP AMP เบอร์ LF 353 ประกอบเป็นวงจร WINDOW COMPARATOR ซึ่งมีลักษณะของวงจรดังรูปที่ 5.1



รูปที่ 5.1 ลักษณะวงจรตรวจวัดแรงดันไฟฟ้า และ IC LF 353



รูปที่ 5.2 วงจรแปลงสัญญาณ AC เป็น DC

หลักการทำงานของวงจร

เราจะกำหนดให้ค่าแรงดันไฟฟ้า ที่ตัวคอมเพรสเซอร์สามารถทำงานได้ตามปกติให้อยู่ในช่วง 189V-250V ถ้าแรงดันไฟบ้านอยู่ภายนอกช่วงนี้วงจรจะส่งสัญญาณให้CPU. ทำการตัดการทำงานทันที ในการตรวจเช็คระดับแรงดันไฟฟ้าจะทำโดย ผ่านไฟบ้านสู่หม้อแปลงให้เหลือ5volt AC และแปลง AC ให้เป็น DC โดยวงจร ดังรูปที่ 5.2

เราจะ set ค่าเพื่อทำการเปรียบเทียบระดับแรงดันไฟฟ้าในวงจรคอมพาราเตอร์ โดยแรงดันปกติที่ 220 volt เมื่อถูกแปลงลงมาและผ่านการแปลงเป็น DC จะได้ ซึ่งจะนำไปผ่านความต้านทานปรับค่าได้ โดยเราจะปรับให้ได้ volt เท่ากับ 3.4 volt ฉะนั้นเราจะได้ค่า voltage สูงสุดที่คอมเพรสเซอร์ทำงานได้ตามปกติ (250 volt) ในวงจรคอมพาราเตอร์เท่ากับ 3.86 volt โดยเราจะประมาณให้มีค่าเท่ากับ 3.8 volt และค่า voltage ต่ำสุดที่คอมเพรสเซอร์ทำงานได้ตามปกติ (189 volt) ในวงจรคอมพาราเตอร์เท่ากับ 2.92 volt ซึ่งเราจะประมาณให้มีค่าเท่ากับ 3.0 volt voltage 2 ค่านี้ (3.0,3.8) จะเป็นตัวกำหนดช่วงแรงดันไฟปกติ

ในวงจรคอมพาราเตอร์ ฉะนั้นเมื่อให้ได้ค่าของ voltage ดังกล่าวในวงจร เราจะต่อความต้านทานเป็น votage divider และคำนวณหาค่าความต้านทานที่จะนำไปต่อในวงจรให้ได้ voltage นั้น โดยเลือกตัวความต้านทานที่มีเมื่อให้ได้ค่า voltage มีค่าใกล้เคียงกับ voltage ที่คำนวณออกมามากที่สุด ซึ่งในวงจรนี้หลังจากเลือกตัวความต้านทานที่มีค่าเหมาะสมที่สุดแล้ว จะได้ช่วงของแรงดันไฟปกติในวงจรคอมพาราเตอร์มีค่าเท่ากับ 3.01 volt ถึง 3.81 volt จากค่านี้ทำให้วงจรจริงมีช่วงแรงดันไฟปกติอยู่ระหว่าง 195 volt ถึง 247 volt เมื่อเกิดไฟเกิน 247 volt จะได้ค่า voltage ที่วงจรคอมพาราเตอร์เกิน 3.81 volt กรณีเช่นนี้จะทำให้สัญญาณ OUTPUT ที่ขา 1 เป็น LOW เมื่อเกิดไฟตกต่ำกว่า 195 volt จะได้ค่า voltage ที่วงจรคอมพาราเตอร์ต่ำกว่า 3.01 volt จะทำให้สัญญาณ OUTPUT ที่ขา 7 เป็น LOW ซึ่งในทั้ง 2 กรณีนี้สัญญาณ LOW ที่ส่งไป จะทำให้ C.P.U ทำการตัดการทำงานของคอมเพรสเซอร์ทันที และเมื่อไฟบ้านอยู่ในช่วงปกติ (195 volt-247 volt หรือ 3.01 volt - 3.81 volt ในวงจรคอมพาราเตอร์) ขา 1 และ ขา 7 จะเกิดสัญญาณของขา 1 และ ขา 7 จะแสดงดังตารางที่ 5.1 ต่อไปนี้

	ปกติ	UNDER	OVER	ขา
P1.0	1	0	1	7
P1.1	1	1	0	1

ตารางที่ 5.1 แสดงสัญญาณที่ขา 1 และขา 7 ในสภาวะแรงดันไฟต่าง ๆ

บทที่ 6

ส่วนแสดงผล (DISPLAY)

ส่วนแสดงผลทำให้ผู้ใช้มีความสะดวกสบายใน set ค่าที่ต้องการและแสดงให้เห็นสถานะต่าง ๆ ของการทำงานของเครื่องปรับอากาศ ซึ่ง DISPLAY นี้จะสามารถใช้ประโยชน์ได้ดังต่อไปนี้

1. POWER เปิด/ปิด

เพียงกดปุ่ม POWER เครื่องจะเริ่มทำงาน โดยแสดงสัญญาณไฟที่ตำแหน่ง "NORM"กรณีที่แรงดันไฟฟ้าเกิน (OVER) หรือต่ำ (UNDER) กว่ามาตรฐานซึ่งอาจเป็นอันตรายต่อเครื่องปรับอากาศ ไมโครโพรเซเซอร์จะหยุดการทำงานของเครื่องและแสดงเครื่องปรับอากาศ ไมโครโพรเซสเซอร์จะหยุดการทำงานของเครื่องและแสดงสัญญาณไฟให้เห็นที่ตำแหน่ง "OVER" หรือ "UNDER" จนกว่าแรงดันไฟฟ้าจะกลับมาสู่สภาพปกติเครื่องปรับอากาศจะเริ่มทำงานต่อไป

2. FAN พัดลม

สามารถควบคุมพัดลมให้ทำงานได้ทั้งระบบอัตโนมัติและระบบธรรมดา ระบบอัตโนมัติจะมีสัญญาณไฟที่ตำแหน่ง "AUTO" ซึ่งความเร็วของพัดลมจะถูกปรับโดยอัตโนมัติให้สอดคล้องกับความแตกต่างของอุณหภูมิห้องกับอุณหภูมิที่ตั้งไว้ หากแตกต่างมากพัดลมจะถูกปรับให้มีความเร็วสูงสุด เพื่อเร่งกระจายความเย็นให้ทั่วถึงทั้งห้องอย่างรวดเร็วและแสดงสัญญาไฟกระพริบที่ตำแหน่ง "HI" เมื่ออุณหภูมิห้องลดลงพัดลมจะถูกปรับความเร็ว

จาก "HI" เป็น "MED." และ "LOW" โดยอัตโนมัติ พร้อมทั้งแสดงให้เห็นด้วยสัญญาณไฟกระพริบที่ตำแหน่งความเร็วนั้นระบบอัตโนมัตินอกจากจะให้ความรู้สึกที่สบายแล้วยังช่วยประหยัดกระแสไฟฟ้าอีกด้วย

ระบบธรรมดา เมื่อกดปุ่ม "FAN" สัญญาณไฟ "AUTO" จะดับและพัดลมจะพัดที่ความเร็ว "HI" ซึ่งแสดงด้วยสัญญาณไฟที่ติดตลอดเวลาที่ตำแหน่ง "HI" เมื่อต้อบการให้พัดลมพัดที่ความเร็วอื่น "MED" หรือ "LOW" หรือกลับไปสู่ระบบอัตโนมัติก็เพียงแต่กดปุ่ม "FAN" ซ้ำ ๆ กัน ก็จะเปลี่ยนได้ตามต้องการโดยมีสัญญาณไฟเปลี่ยนตามไปด้วย

3. TEMPERATURE ปรับอุณหภูมิ

เมื่อกดปุ่ม POWER เครื่องจะทำงานตามอุณหภูมิสุดท้ายที่ตั้งไว้ซึ่งแสดงสัญญาณไฟที่ติดตลอดเวล การเปลี่ยนระดับอุณหภูมิทำได้โดยกดปุ่ม Δ หรือ ∇ เท่านั้น นอกจากนี้ยังแสดงอุณหภูมิห้องในขณะนั้นไฟกระพริบอีกด้วย

4. SPEEP ควบคุมอุณหภูมิเวลานอน

เพียงกดปุ่ม "SLEEP" อุณหภูมิที่ตั้งไว้จะเลื่อนขึ้นไป $1^{\circ}C$ เมื่อเวลาผ่านไป 1/2 ชั่วโมง และอีก 1/2 ชั่วโมงจากนั้นจะปรับให้สูงขึ้นอีก $1^{\circ}C$ ทั้งนี้เพราะขณะหลับคนเราจะใช้พลังงานลดลงทำให้รู้สึกหนาว และยังช่วยประหยัดกระแสไฟฟ้าได้อีกด้วย

5. TIMER ตั้งเวลาเปิด-ปิด

สามารถตั้งเวลาให้เปิดหรือปิดเครื่องปรับอากาศล่วงหน้าได้ถึง 12 ชั่วโมง เปิดเครื่องอัตโนมัติ เพียงกดปุ่ม "TIMER 1" จะปรากฏสัญญาณไฟแสดงที่ตำแหน่ง "ON" หากเครื่องกำลังใช้งานอยู่จะหยุดทำงานเอง พร้อมทั้งของตัวเลขชั่วโมงจะแสดงที่เลข 1

กดปุ่ม หรือ เพื่อเปลี่ยนแปลงจำนวนชั่วโมงให้เพิ่มขึ้นหรือลดลง หากต้องการยกเลิกคำสั่ง ทำได้โดยกดปุ่ม "TIMER 1" ซ้ำอีกครั้งหนึ่งปิดเครื่องอัตโนมัติ เพียงกดปุ่ม "TIMER 0" จะปรากฏสัญญาณไฟแสดงที่ตำแหน่ง "OFF" พร้อมทั้งช่องตัวเลขชั่วโมงจะขึ้นที่เลข 1 กดปุ่ม ▲ หรือ เพื่อเปลี่ยนแปลงจำนวนชั่วโมงให้เพิ่มขึ้นหรือลดลง หากต้องการยกเลิกคำสั่งทำได้โดยการกดปุ่ม "TIMER 0" ซ้ำอีกครั้งหนึ่ง

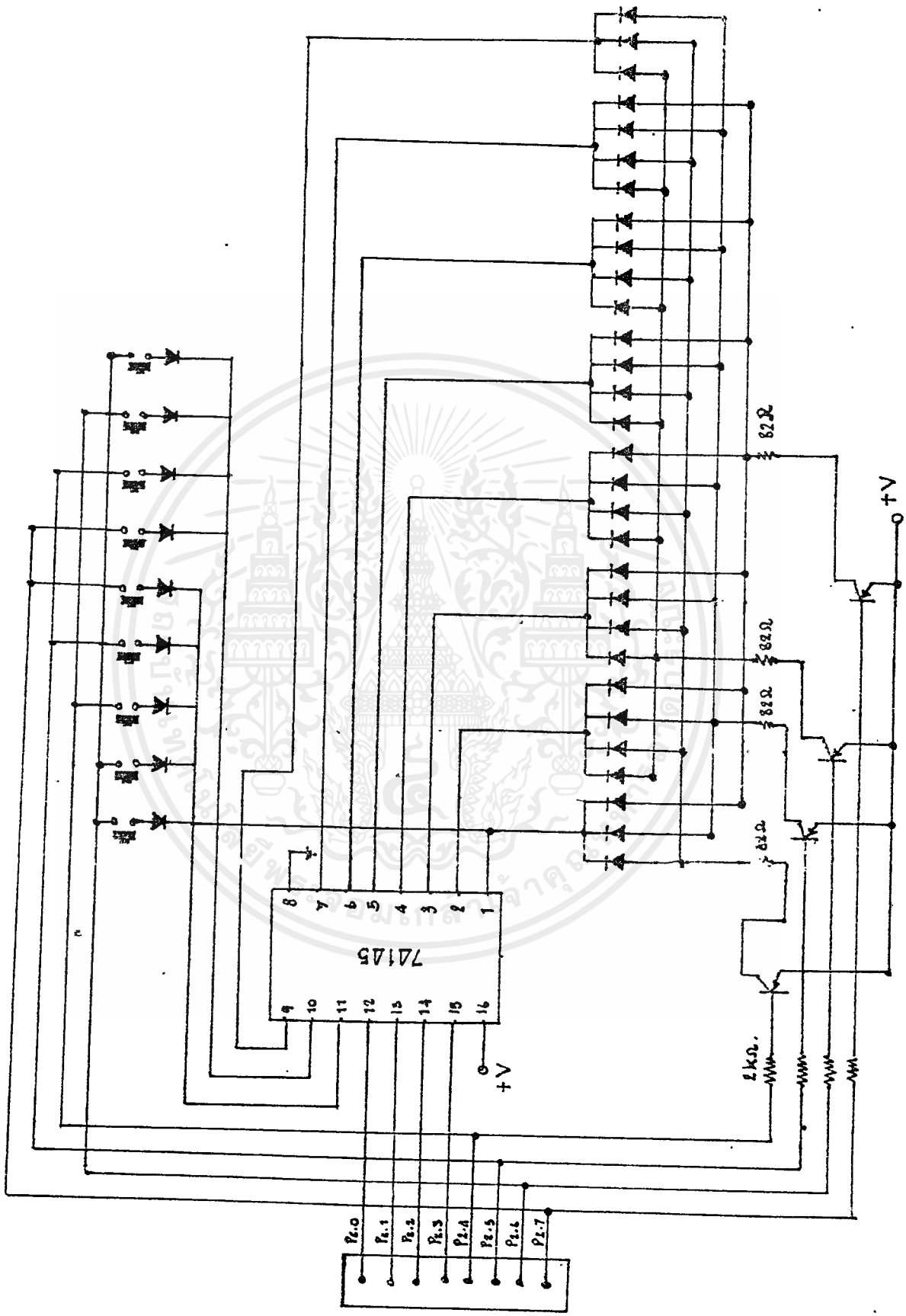
คุณสมบัติพิเศษ

DELAY หน่วงเวลา 3 นาที

บางครั้งกระแสไฟฟ้าอาจดับในช่วงเวลาสั้น ๆ แล้วกลับติดขึ้นมาใหม่ซึ่งอาจทำความเสียหายต่อคอมเพรสเซอร์ได้ ไมโครโพรเซสเซอร์จะหน่วงเวลาการเริ่มทำงานของคอมเพรสเซอร์ออกไป 3 นาที เพื่อป้องกันความเสียหายที่อาจเกิดขึ้นแต่ในการเปิดการทำงานของเครื่องปกติจะไม่มีการหน่วงเวลา เพื่อให้คอมเพรสเซอร์เริ่มทำงานในทันที

หลักการในการแสดงผลและ set ค่า

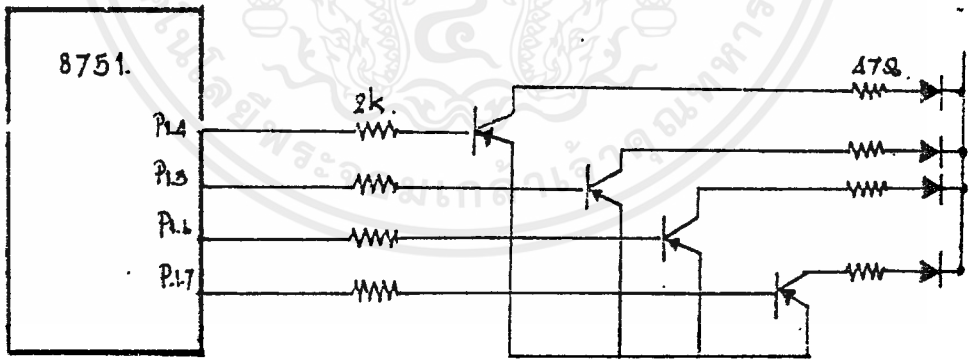
วงจร display จะประกอบด้วยชุด LED ทั้งหมด 8 ชุด เพื่อแสดงผลในสถานะต่าง ๆ SWITCH เพื่อใช้ในการ set ค่าที่ผู้ใช้ต้องการ และ IC 74145 ทำหน้าที่ในการสร้างสัญญาณไป SCAN ชุดของ LED ร่วมกับข้อมูลที่เคย set ไว้จาก STACK เพื่อให้ LED แสดงผลและ SCAN ชุดของ SWITCH เพื่อให้ผู้ใช้ input ค่าที่ต้องการเข้าไปเก็บเป็นข้อมูลใน stack วงจร display จะมีรูปแบบดังรูปที่ 6.1 LED ในวงจรจะถูกแบ่งออกเป็น 8 ชุดด้วยกัน ดังนั้น 74145 จะทำการ SCAN (ให้สัญญาณ LOW) ไปทีละชุด โดย LED ในแต่ละชุดจะถูกเลือกให้แสดงผลในสถานะนั้น ๆ จากข้อมูลที่ได้ set ไว้ใน stack โดยข้อมูลจาก stack จะเป็นตัวเลือก transistor ตัวใดตัวหนึ่งให้ทำงาน คือ



รูปที่ 6.1 วงจร DISPLAY.

ข้อมูลที่ส่งสู่ Tranistor ที่ถูกเลือกนั้นจะเป็น low ที่ขา collector จะมีผลทำให้มีกระแสไหลผ่าน LED แสดงผลในสภาวะนั้น ๆ ตามข้อมูล 74145 จะทำการ scan ให้สัญญาณ low ตั้งแต่ขาที่ 1 ถึงขาที่ 9 เป็นลำดับ พร้อม ๆ กับข้อมูลที่ถูกส่งออกมาทำการเลือก Transistor ให้ LED แสดงผลตามสถานะเป็นลำดับไปด้วย จากนั้นก็จะ scan ต่อโดย ใช้สัญญาณ LOE ที่ขา 10 และขา 11 เป็นลำดับ ในที่จะเป็นการเลือกชุดของ SWICH เพื่อที่จะให้ผู้ใช้ทำการ input ค่าที่ต้องการและไปเก็บใน stack เพื่อที่จะเป็นข้อมูลในการแสดงผลต่อไป การ scan ของ 74145 จะมีความเร็วสูงทำให้ดูเหมือน LED ติดอยู่ตลอดเวลา

ในการส่งสัญญาณของ cpu เพื่อทำการขับ compressor และ พัดลม โดยจะส่งสัญญาณจาก P1.4 เพื่อทำการเปิด-ปิด compressor และจาก P1.5, P1.6,P1.7 เพื่อทำการขับพัดลมโดยมีระดับ speed จาก high medium low ตามลำดับ โดยวงจรการขับจะแสดงรูปที่ 6.2



รูปที่ 6.2 วงจรขับ compressor และ พัดลม

บทที่ 7

โปรแกรมการทำงาน

ฟังก์ชั่นการทำงานของวงจรควบคุมการทำงานของเครื่องปรับอากาศจะสามารถทำหน้าที่ต่าง ๆ ได้ดังต่อไปนี้

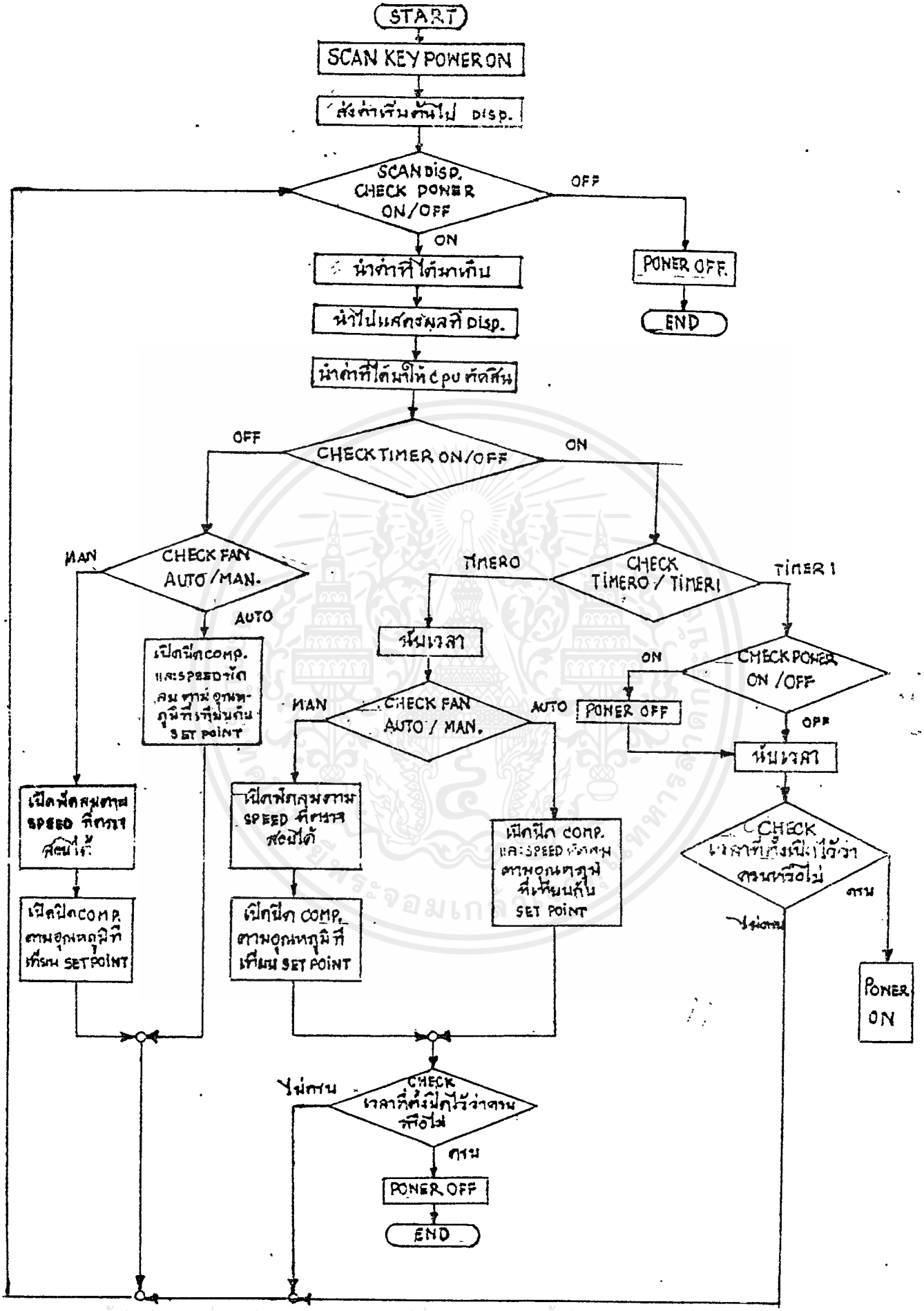
1. ป้องกันความเสียหายที่อาจเกิดขึ้นกับตัวคอมเพรสเซอร์ เนื่องจากภาวะไฟตก-ไฟเกิน

2. เลือกให้พัดลมสามารถปรับระดับ speed ของตัวมันเองให้สอดคล้องกับอุณหภูมิห้อง (MODE AUTO) หรือจะให้ผู้ใช้ปรับระดับ speed ของพัดลมตามความพอใจของผู้ใช้ของ (MODE MANUAL) ซึ่งจะมี speed 3 ระดับคือ HIGH,MEDIUM,LOW

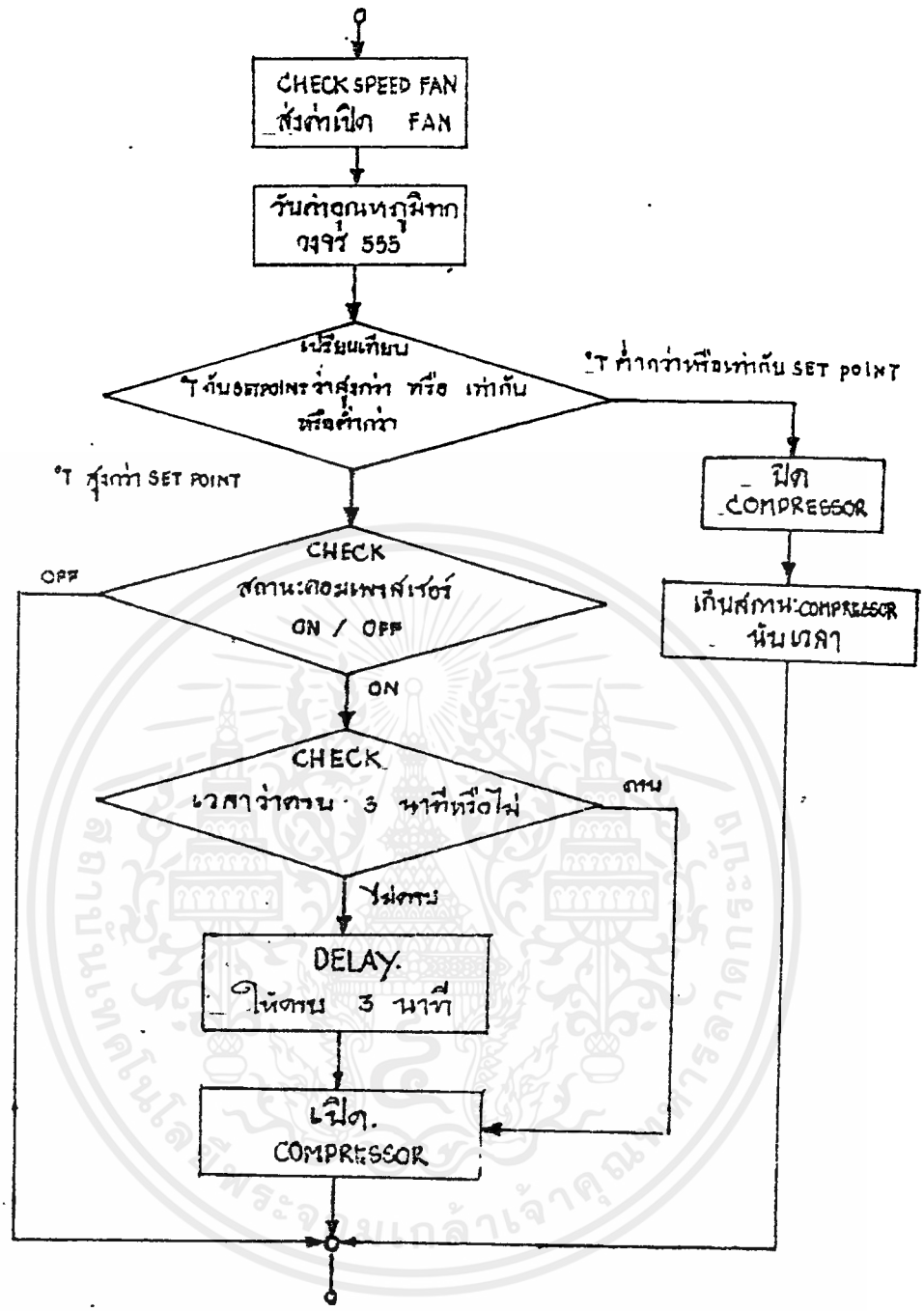
3. เพื่อให้ผู้ใช้ตั้งระดับอุณหภูมิสูงต่ำตามความต้องการ

4. เพื่อให้ผู้ใช้สามารถที่จะตั้งเวลาเปิดปิดได้เองตามต้องการ

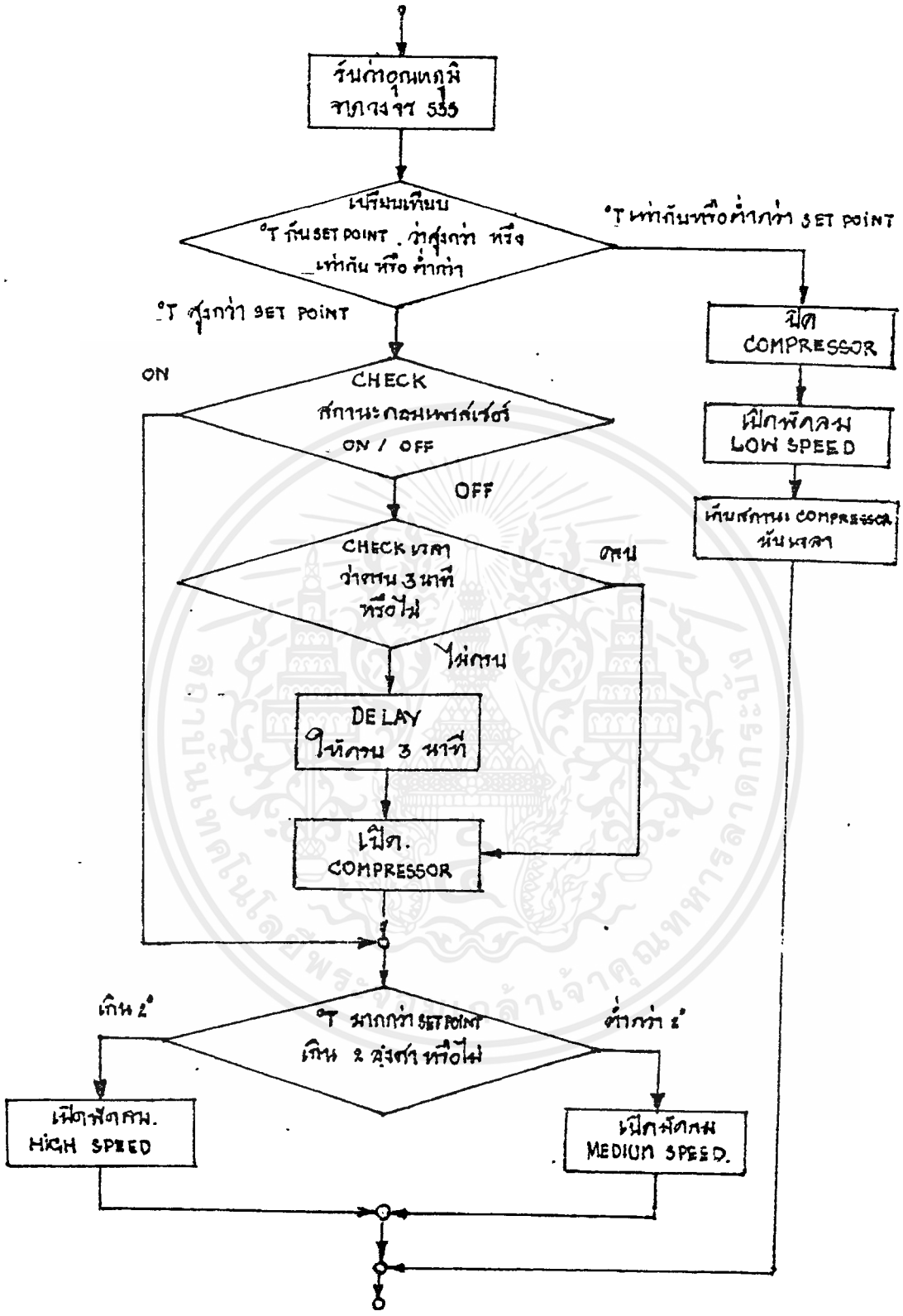
5. มีคุณสมบัติในการหน่วงเวลา 3 นาที เพื่อให้ความดันของเหลวภายในต่อแอร์ปรับตัวอยู่ในสภาวะสมดุลย์จึงเริ่มทำงานใหม่ เราจะสามารถศึกษาการทำงานของโปรแกรมได้ตาม flow chart ต่อไปนี้



รูปที่ 7.1 FLOW CHART แสดงการทำงานของโปรแกรม



รูปที่ 7.2 FLOW CHART แสดง MODE MANUAL



รูปที่ 7.3 Flow Chart แสดง MODE AUTO

จาก flow chart จะสามารถอธิบายการทำงานของโปรแกรมได้ดังนี้

เมื่อเริ่มเปิดเครื่อง (SWITCH ON) CPU จะส่งค่าสถานะเริ่มต้นไปแสดงที่ DISPLAY และจะทำการ CHECK แรงดันไฟฟ้าว่าอยู่ในช่วงปกติหรือไม่ (190 volt-250volt) ถ้าแรงดันไฟฟ้าไม่อยู่ในช่วงนี้ ก็จะยังไม่ทำการเปิดคอมเพรสเซอร์ และจะส่งสัญญาณไปแสดงที่ DISPLAY ว่าเกิดสภาวะไฟตกหรือไฟเกิน ถ้าแรงดันไฟฟ้าอยู่ในช่วงปกติก็จะทำตามโปรแกรมต่อไป โดยนำค่าที่ผู้ใช้ set ใหม่มาเก็บและแสดงผลออกไปยัง DISPLAY นำค่าที่ได้มาให้ CPU ตัดสิน หลังจากนั้นจะ check ว่าผู้ใช้เลือกใช้ฟังก์ชั่น TIMER หรือปล่าว ถ้าเลือกใช้ check ดูว่า ผู้ใช้เลือก TIMER 0 (ตั้งปิดอัตโนมัติ) หรือ TIMER 1 (ตั้งเปิดอัตโนมัติ) ถ้าผู้ใช้เลือกตั้งปิดอัตโนมัติ ก็จะเริ่มนับเวลา และทำการ CHECK ดูว่าอยู่ใน MODE AUTO หรือ MANUAL หลังจากนั้นก็จะทำการเปิดหรือปิดคอมเพรสเซอร์ตามการเปรียบอุณหภูมิที่ได้ set ไว้กับอุณหภูมิห้อง ซึ่งจะมีการหน่วงเวลาการเปิดคอมเพรสเซอร์ถ้าการปิดยังไม่ครบ 3 นาที โดยใน MODE AUTO SPEED ของพัดลมจะปรับตามระดับของอุณหภูมิห้องโดยอัตโนมัติ ส่วน MODE MANUAL SPEED ของพัดลมจะปรับตามความพอใจของผู้ใช้ จนกว่าเวลาที่ตั้งไว้ใน TIMER 0 จะหมด ก็จะหยุดการทำงานทั้งหมด ส่วน TIMER 1 หรือตั้งเปิดอัตโนมัติ โปรแกรมจะ check ว่าเครื่องเปิดหรือปิดอยู่ก็จะปิด ก็จะเริ่มนับเวลาและ check ว่าถึงเวลาหรือยังก็จะเปิดได้ เมื่อถึงแล้วก็จะทำการเปิดเครื่อง (POWER ON) ส่วนถ้าผู้ใช้ไม่ต้องการตั้งเวลาเปิดหรือปิดโปรแกรมก็จะดำเนินการ check ว่าอยู่ใน MODE AUTO หรือ MODE MANUAL และก็จะทำการเปิดปิดคอมเพรสเซอร์ตามการเปรียบเทียบอุณหภูมิที่ได้ set ระดับแรงดันไฟ TIMER MANUAL หรือ AUTO เป็น LOOP ตลอดไปจนกว่าจะมีการกด SWITCH OFF เครื่องก็จะหยุดดำเนินงาน

บทที่ 8

สรุปผลและวิจารณ์

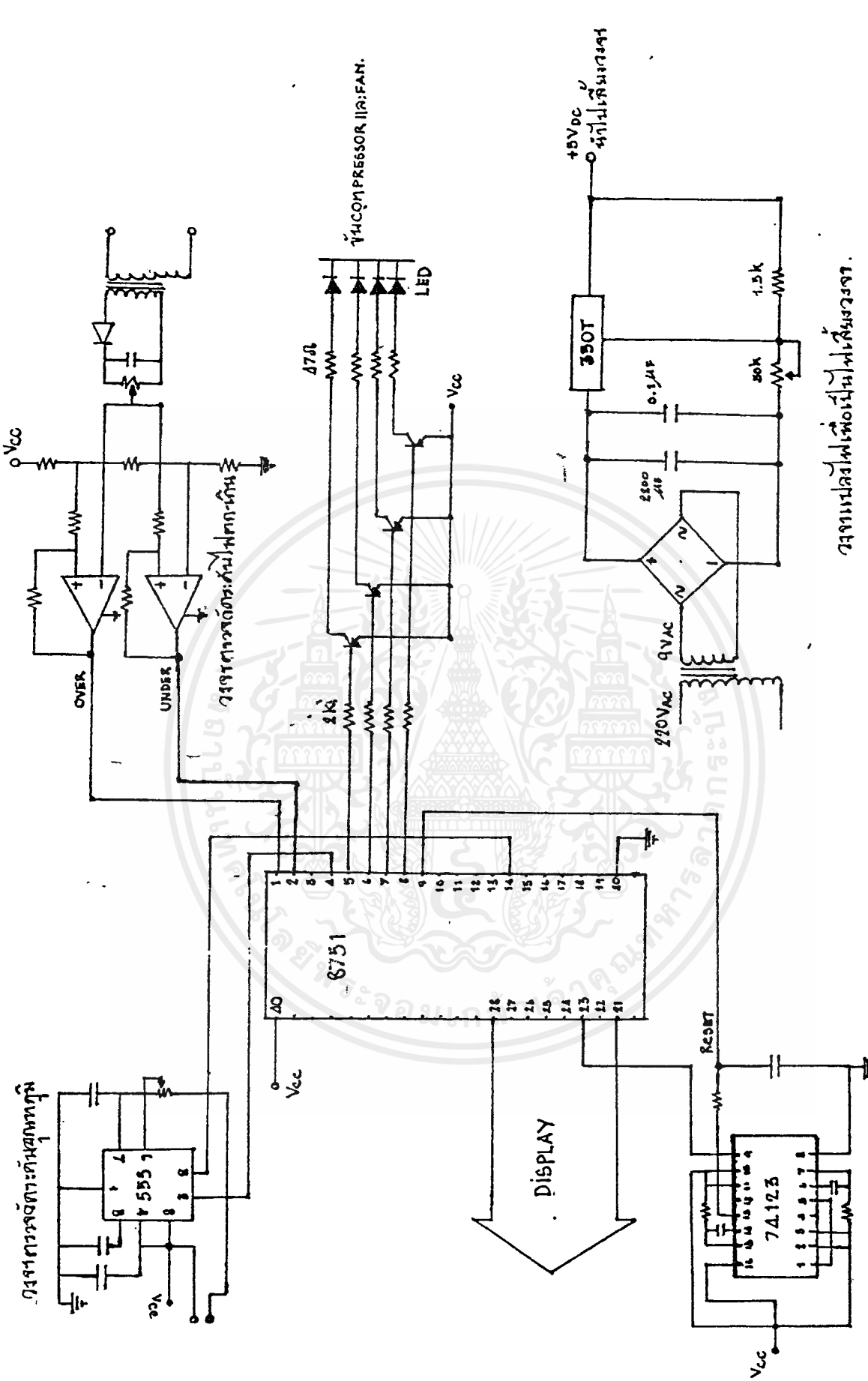
เราได้ศึกษาและทำความเข้าใจเกี่ยวกับหลักการทำงานของวงจรต่าง ๆ ซึ่งประกอบกันขึ้นเป็นวงจรเพื่อการควบคุมการทำงานของเครื่องปรับอากาศซึ่งมีลักษณะการทำงานเป็นแบบ ON-OFF CONTROL โดยมี MICROPROCESSOR 8751 เป็น CPU ทำการควบคุมการทำงานทั้งหมด โดยจะสามารถแสดงเป็นวงจรรวมได้ดังรูปที่ 8.1 ซึ่งสามารถสรุปหน้าที่และหลักการทำงานของวงจร ได้ดังนี้

1. ควบคุมการทำงานของคอมเพรสเซอร์ เพื่อ

1.1 ทำการปรับระดับของอุณหภูมิตามที่ผู้ใช้ต้องการ โดยจะมีวงจรตรวจวัดระดับอุณหภูมิ (TEMPERATURE SENSOR) ทำการวัดระดับอุณหภูมิตามที่ผู้ใช้ต้องการ โดยจะมีวงจรตรวจวัดระดับของอุณหภูมิที่ผู้ใช้ตั้งไว้ เพื่อเป็นข้อมูลในการให้ cpu ทำการตัดสินใจในการเปิดหรือปิดคอมเพรสเซอร์ คือเมื่อระดับอุณหภูมิถึงค่าที่ตั้งไว้ก็จะทำการปิดและเพื่ออุณหภูมิสูงกว่าก็จะทำการเปิดคอมเพรสเซอร์

1.2 เพื่อป้องกันความเสียหายอันเกิดจากแรงดันไฟตก-ไฟเกิน โดยจะมีวงจรตรวจวัดระดับแรงดันไฟฟ้า ทำหน้าที่ในการส่งสัญญาณข้อมูลให้กับ cpu ในกรณีที่ระดับแรงดันไฟฟ้าไม่อยู่ในช่วงที่ คอมเพรสเซอร์สามารถทำงานได้อย่างปลอดภัย โดย cpu จะทำการปิดคอมเพรสเซอร์ทันทีที่ได้รับสัญญาณ

ในการเปิดคอมเพรสเซอร์ทุกครั้ง จะทำการเช็คเวลาว่าได้ทำการปิดมาแล้วครบ 3 นาทีหรือยังถ้ายังจะทำการหน่วงเวลาให้ครบ 3 นาทีแล้วจึงเปิด ทั้งนี้เพื่อเป็นการรอให้ระดับความดันของ ๆ เหลวภายในท่อเครื่องปรับอากาศอยู่ในสภาวะสมดุลย์ก่อน



รูปที่ 8.1 วงจรควบคุมการทำงานของเครื่องปรับอากาศ

2. ควบคุมระดับ speed ของพัดลม โดยจะแบ่งลักษณะการควบคุม ออกเป็น

2.1 ควบคุม speed ของพัดลมโดยผู้ใช้ (mode manual) โดย cpu จะเป็นตัวรับข้อมูลโดยตรงจาก display ซึ่งถูกเลือกโดยผู้ใช้ และจะส่งสัญญาณไปปรับ speed ของพัดลมตามข้อมูลที่ได้รับมา โดยจะมี speed 3 ระดับคือ low medium high

2.2 ควบคุม speed ของพัดลมโดยอัตโนมัติ (mode auto) ใน mode นี้ ระดับ speed ของพัดลม จะปรับเองอัตโนมัติ ตามระดับอุณหภูมิของห้อง คือเมื่อระดับอุณหภูมิจริงของห้องมีค่ามากกว่า อุณหภูมิที่ผู้ใช้ตั้งไว้เกินกว่า 2 องศาเซลเซียส speed ของพัดลมจะปรับในระดับ high เมื่ออุณหภูมิจริงของห้องมีค่ามากกว่าอุณหภูมิที่ผู้ใช้ตั้งไว้ต่ำกว่า 2 องศาเซลเซียส พัดลมจะปรับในระดับ medium และเมื่ออุณหภูมิจริงเท่ากับอุณหภูมิที่ตั้งไว้ พัดลมจะปรับระดับเป็น low ทั้งนี้ cpu จะต้องใช้ข้อมูลจากวงจรตรวจวัดระดับอุณหภูมิ (TEMPERATURE SENSOR) นำมาเปรียบเทียบค่าตามโปรแกรมที่ตั้งไว้และทำการส่งสัญญาณไปปรับระดับ speed ของพัดลม ตามค่าที่เปรียบเทียบได้

3. ตั้งเวลาเปิด-ปิด อัตโนมัติในฟังก์ชั่นการทำงานนี้ cpu จะเป็นตัวรับค่าของเวลาที่ผู้ใช้ได้ตั้งไว้โดยตรงจาก display เพื่อทำการนับเวลาเมื่อครบตามที่ได้ตั้งไว้ cpu ก็จะทำการเปิดหรือปิด โดยอัตโนมัติ

4. เช็คสถานะการทำงานของ cpu เพื่อให้วงจรสามารถทำงานตามปกติโดยจะมีวงจร watch dog ทำการตรวจวัดสัญญาณที่ส่งมาจาก cpu เมื่อ cpu ทำงานผิดปกติ ก็สามารถที่จะตรวจสอบสัญญาณ ซึ่งผิดปกติ ทำให้วงจร watch dog รับรู้และส่งสัญญาณ เพื่อไป reset cpu ให้ทำงานในสถานะเริ่มต้นใหม่ และสามารถทำงานตามปกติได้ต่อไป

5. แสดงผลและ set ค่าตามที่ผู้ใช้ต้องการโดยวงจร display

จากการศึกษาและการเรียนรู้ที่ผ่านมาทำให้ได้มีโอกาส พัฒนาวงจรการทำงานจริง ๆ ทำให้เข้าใจเทคนิคการทำงานของวงจรต่างๆและเข้าใจการพัฒนาโปรแกรมที่ใช้กับ microprocessor 8751 เพื่อควบคุมการทำงานของวงจร ประสบการณ์และความรู้ที่ได้ จะได้นำไปใช้ประโยชน์ในภายภาคหน้าต่อไป

ภาคผนวก

```
                    1       $include"head.asm"
007C=           +   2       tmp             equ     7ch     ;7c,7d,7e,7f
007B=           +   3       entry           equ     7bh     ;7b
007A=           +   4       col             equ     7ah     ;7a
0079=           +   5       code            equ     79h     ;79
0030=           +   6       stack           equ     30h     ;30-5f
                +   7       ;
0020=           +   8       led_data        equ     20h     ;20,21,22,23
                +   9       ;
0000=           +  10       data.0          bit     00      ;
0001=           +  11       pw_over         bit     01      ;0e0h
0002=           +  12       pw_norm         bit     02      ;0d0h
0003=           +  13       pw_under        bit     03      ;070h
0004=           +  14       fan_auto        bit     04      ;0b8h
0005=           +  15       fan_high        bit     05      ;0e8h
0006=           +  16       fan_medium      bit     06      ;0d8h
0007=           +  17       fan_low         bit     07      ;078h
                +  18       ;
0008=           +  19       sleep           bit     08      ;0b4h
0009=           +  20       temp_28         bit     09      ;0e4h
000A=           +  21       temp_27         bit     10      ;0d4h
000B=           +  22       temp_26         bit     11      ;074h
000C=           +  23       temp_25         bit     12      ;0bch
000D=           +  24       temp_24         bit     13      ;0ech
000E=           +  25       temp_23         bit     14      ;0dch
000F=           +  26       temp_22         bit     15      ;07ch
                +  27       ;
0010=           +  28       temp_21         bit     16      ;0b2h
0011=           +  29       timer_on        bit     17      ;0e2h
0012=           +  30       timer_off       bit     18      ;0d2h
0013=           +  31       hour_12         bit     19      ;072h
0014=           +  32       hour_11         bit     20      ;0bah
0015=           +  33       hour_10         bit     21      ;0eah
0016=           +  34       hour_9          bit     22      ;0dah
0017=           +  35       hour_8          bit     23      ;07ah
                +  36       ;
0018=           +  37       hour_7          bit     24      ;0b6h
0019=           +  38       hour_6          bit     25      ;0e6h
001A=           +  39       hour_5          bit     26      ;0d6h
001B=           +  40       hour_4          bit     27      ;076h
001C=           +  41       hour_3          bit     28      ;0beh
001D=           +  42       hour_2          bit     29      ;0eeh
001E=           +  43       hour_1          bit     30      ;0deh
001F=           +  44       data.31         bit     31      ;
                +  45       ;
0024=           +  46       status          equ     24h     ;status of timer
                +  47       ;
0020=           +  48       timer_off_status bit    24h.0
0021=           +  49       timer_on_status bit     24h.1
                +  50       ;########  key code  ######
                +  51       ;key_col_1
00B0=           +  52       key_pw_on       equ     0b0h    ;0f0h and 0b0h :(p2.6 = 0)
                +  53       ;
                +  54       ;key_col_2                      ;0f9h
00B9=           +  55       key_fan         equ     0b9h    ;0f9h and 0b9h :(p2.6 = 0)
00E9=           +  56       key_sleep       equ     0e9h    ;0f9h and 0e9h :(p2.4 = 0)
00D9=           +  57       key_inc_temp    equ     0d9h    ;0f9h and 0d9h :(p2.5 = 0)
0079=           +  58       key_dec_temp    equ     079h    ;0f9h and 079h :(p2.7 = 0)
```

```
                +  59        ;
                +  60        ;key_col_3                      ;0f1h
0071=           +  61        key_timer_on    equ     071h    ;0f1h and 071h :(p2.7 = 0)
00D1=           +  62        key_timer_off   equ     0d1h    ;0f1h and 0d1h :(p2.5 = 0)
00E1=           +  63        key_inc_hour    equ     0e1h    ;0f1h and 0e1h :(p2.4 = 0)
00B1=           +  64        key_dec_hour    equ     0b1h    ;0f1h and 0b1h :(p2.6 = 0)
                +  65        ;
                +  66        ; buffers of values for decision
                +  67        ; power : 1 = clear, 2 = under, 3 = norm, 4 = over
0078=           +  68        power           equ     78h     ;78
                +  69        ; fan   : 1 = low, 2 = medium, 3 = high
0077=           +  70        fan             equ     77h     ;77
                +  71        ; temp_set  : 21,...,28 (decimal)
0076=           +  72        temp_set        equ     76h     ;76
                +  73        ; timer on, timer off, off timer checked from status
                +  74        ;
                +  75        ; hour  : 1,...,12 (decimal)
0075=           +  76        hour            equ     75h     ;75
                +  77
0074=           +  78        count_gen       equ     74h
0073=           +  79        tmph            equ     73h
0072=           +  80        tmpl            equ     72h
0071=           +  81        cmph            equ     71h
0070=           +  82        cmpl            equ     70h
006F=           +  83        temp            equ     6fh
006E=           +  84        disp_count_h    equ     6eh
006D=           +  85        disp_count_l    equ     6dh
006C=           +  86        delay_h         equ     6ch
006B=           +  87        delay_l         equ     6bh
                +  88        ;
007F=           +  89        key_flag        bit     7fh
007E=           +  90        releaseflag     bit     7eh
007D=           +  91        equal           bit     7dh
007C=           +  92        freq            bit     7ch
007B=           +  93        delay_ok        bit     7bh
007A=           +  94        equal_          bit     7ah
0079=           +  95        timer_mask      bit     79h
                   96        $include"init.asm"
0200            +  97                        org     0200h
0200 75812F     +  98        init:           mov     sp,#stack-1     ;set stack pointer to stack area
0203 757A01     +  99                        mov     col,#1          ;col : 1,2,3
0206 C27E       + 100                        clr     releaseflag     ;first it's released ( = 0)
0208 756F1C     + 101                        mov     temp,#28        ;init temperature
020B 757619     + 102                        mov     temp_set,#25    ;init temperature setting
020E 757803     + 103                        mov     power,#3        ;norm
0211 757701     + 104                        mov     fan,#1          ;low
                + 105        ;
0214 515C       + 106                        acall   clr_data
0216 D202       + 107                        setb    pw_norm
0218 D204       + 108                        setb    fan_auto
021A D207       + 109                        setb    fan_low
021C D20C       + 110                        setb    temp_25
021E 217A       + 111                        ajmp    end_init
                + 112        ;
0220 D201       + 113                        setb    pw_over
0222 D202       + 114                        setb    pw_norm
0224 D203       + 115                        setb    pw_under
0226 D204       + 116                        setb    fan_auto
0228 D205       + 117                        setb    fan_high
```

```
022A D206     + 118                     setb    fan_medium
022C D207     + 119                     setb    fan_low
022E D208     + 120                     setb    sleep
0230 D209     + 121                     setb    temp_28
0232 D20A     + 122                     setb    temp_27
0234 D20B     + 123                     setb    temp_26
0236 D20C     + 124                     setb    temp_25
0238 D20D     + 125                     setb    temp_24
023A D20E     + 126                     setb    temp_23
023C D20F     + 127                     setb    temp_22
023E D210     + 128                     setb    temp_21
0240 D211     + 129                     setb    timer_on
0242 D212     + 130                     setb    timer_off
0244 D213     + 131                     setb    hour_12
0246 D214     + 132                     setb    hour_11
0248 D215     + 133                     setb    hour_10
024A D216     + 134                     setb    hour_9
024C D217     + 135                     setb    hour_8
024E D218     + 136                     setb    hour_7
0250 D219     + 137                     setb    hour_6
0252 D21A     + 138                     setb    hour_5
0254 D21B     + 139                     setb    hour_4
0256 D21C     + 140                     setb    hour_3
0258 D21D     + 141                     setb    hour_2
025A D21E     + 142                     setb    hour_1
              + 143
                144           $include"scan.asm"
              + 145           $title"scan display and keyboard"
              + 146           ;#################################################
025C C0E0     + 147           clr_data:       push    acc
025E E4       + 148                           clr     a
025F F520     + 149                           mov     led_data,a
0261 F521     + 150                           mov     led_data+1,a
0263 F522     + 151                           mov     led_data+2,a
0265 F523     + 152                           mov     led_data+3,a
0267 D0E0     + 153                           pop     acc
0269 22       + 154                           ret
              + 155
              + 156           ;#################################################
026A C0E0     + 157           scandisp:       push    acc
026C C0F0     + 158                           push    b
026E C0D0     + 159                           push    psw
0270 757B00   + 160           wait_released:  mov     entry,#0
0273 85207C   + 161                           mov     tmp,led_data
0276 85217D   + 162                           mov     tmp+1,led_data+1
0279 85227E   + 163                           mov     tmp+2,led_data+2
027C 85237F   + 164                           mov     tmp+3,led_data+3
              + 165
027F 307C18   + 166                           jnb     freq,sc_1
0282 E56F     + 167                           mov     a,temp          ;display receive temperature
0284 713B     + 168                           acall   temp_table
0286 F5F0     + 169                           mov     b,a
0288 E57E     + 170                           mov     a,tmp+2
028A 13       + 171                           rrc     a
028B F57E     + 172                           mov     tmp+2,a
028D E57D     + 173                           mov     a,tmp+1
028F 13       + 174                           rrc     a
0290 45F0     + 175                           orl     a,b
0292 33       + 176                           rlc     a
```

```
0293 F57D     + 177                     mov     tmp+1,a
0295 E57E     + 178                     mov     a,tmp+2
0297 33       + 179                     rlc     a
0298 F57E     + 180                     mov     tmp+2,a
              + 181
029A C3       + 182       sc_1:         clr     c
029B E57F     + 183                     mov     a,tmp+3
029D 13       + 184                     rrc     a
029E F57F     + 185                     mov     tmp+3,a
02A0 E57E     + 186                     mov     a,tmp+2
02A2 13       + 187                     rrc     a
02A3 F57E     + 188                     mov     tmp+2,a
02A5 E57D     + 189                     mov     a,tmp+1
02A7 13       + 190                     rrc     a
02A8 F57D     + 191                     mov     tmp+1,a
02AA E57C     + 192                     mov     a,tmp
02AC 13       + 193                     rrc     a
02AD F57C     + 194                     mov     tmp,a
02AF 057B     + 195                     inc     entry
02B1 E57B     + 196                     mov     a,entry
02B3 710A     + 197                     acall   table
02B5 5003     + 198                     jnc     off_led
02B7 00       + 199       on_led:       nop
02B8 8004     + 200                     sjmp    sc_next
02BA 44F0     + 201       off_led:      orl     a,#0f0h
02BC 8000     + 202                     sjmp    sc_next
02BE F5A0     + 203       sc_next:      mov     p2,a
02C0 E57B     + 204                     mov     a,entry
02C2 B420D5   + 205                     cjne    a,#32,sc_1
02C5 307E16   + 206                     jnb     releaseflag,released
02C8 E57A     + 207                     mov     a,col
02CA 7105     + 208                     acall   col_table
02CC F5A0     + 209                     mov     p2,a            ;out
02CE E5A0     + 210                     mov     a,p2            ;in
02D0 30E49D   + 211                     jnb     acc.4,wait_released
02D3 30E59A   + 212                     jnb     acc.5,wait_released
02D6 30E697   + 213                     jnb     acc.6,wait_released
02D9 30E794   + 214                     jnb     acc.7,wait_released
02DC C27E     + 215                     clr     releaseflag     ;released it yet
02DE 712C     + 216       released:     acall   inc_col         ;inc column
02E0 E57A     + 217                     mov     a,col
02E2 7105     + 218                     acall   col_table
02E4 F5A0     + 219       scankey:      mov     p2,a            ;out
02E6 E5A0     + 220                     mov     a,p2            ;in
02E8 F579     + 221                     mov     code,a
02EA C27F     + 222                     clr     key_flag
02EC 30E40B   + 223                     jnb     acc.4,set_key_flag
02EF 30E508   + 224                     jnb     acc.5,set_key_flag
02F2 30E605   + 225                     jnb     acc.6,set_key_flag
02F5 30E702   + 226                     jnb     acc.7,set_key_flag
02F8 8004     + 227                     sjmp    scan_end
02FA D27E     + 228       set_key_flag: setb    releaseflag
02FC D27F     + 229                     setb    key_flag
02FE D0D0     + 230       scan_end:     pop     psw
0300 D0F0     + 231                     pop     b
0302 D0E0     + 232                     pop     acc
0304 22       + 233                     ret
0305 83       + 234       col_table:    movc    a,@a+pc
0306 22       + 235                     ret     ;1   ,2   ,3
```

```
0307 F0F9F1   + 236                    db    0f0h,0f9h,0f1h
030A 83       + 237       table:       movc  a,@a+pc
030B 22       + 238                    ret
              + 239       ;                    1    2    3    4    5    6    7    8
030C F0E0D070 + 240                    db    0f0h,0e0h,0d0h,070h,0b8h,0e8h,0d8h,078h
0310 B8E8D878
0314 B4E4D474 + 241                    db    0b4h,0e4h,0d4h,074h,0bch,0ech,0dch,07ch
0318 BCECDC7C
031C B2E2D272 + 242                    db    0b2h,0e2h,0d2h,072h,0bah,0eah,0dah,07ah
0320 BAEADA7A
0324 B6E6D676 + 243                    db    0b6h,0e6h,0d6h,076h,0beh,0eeh,0deh,0feh
0328 BEEEDEFE
              + 244       ;##################################################
032C C0E0     + 245       inc_col:     push  acc
032E E57A     + 246                    mov   a,col
0330 04       + 247                    inc   a
0331 B40402   + 248                    cjne  a,#4,inc_col_1
0334 7401     + 249                    mov   a,#1
0336 F57A     + 250       inc_col_1:   mov   col,a
0338 D0E0     + 251                    pop   acc
033A 22       + 252                    ret
              + 253
033B 83       + 254       temp_table:  movc  a,@a+pc
033C 22       + 255                    ret
              + 256       ;
033D 31323334 + 257                    db    '12345678901234567890'  ;degree
0341 35363738
0345 39303132
0349 33343536
034D 37383930
0351 80       + 258                    db    10000000b               ;21
0352 40       + 259                    db    01000000b               ;22
0353 20       + 260                    db    00100000b               ;23
0354 10       + 261                    db    00010000b               ;24
0355 08       + 262                    db    00001000b               ;25
0356 04       + 263                    db    00000100b               ;26
0357 02       + 264                    db    00000010b               ;27
0358 01       + 265                    db    00000001b               ;28
              + 266
                267       $include"inc_temp.asm"
              + 268       $title"inc_temp ,dec_temp"
0359 C0E0     + 269       inc_temp:    push  acc
035B C0D0     + 270                    push  psw
035D 20090D   + 271                    jb    temp_28,inc_temp_end
0360 A210     + 272                    mov   c,temp_21
0362 C210     + 273                    clr   temp_21
0364 E521     + 274                    mov   a,led_data+1
0366 13       + 275                    rrc   a
0367 F521     + 276                    mov   led_data+1,a
0369 9208     + 277                    mov   sleep,c
036B 0576     + 278                    inc   temp_set
036D D0D0     + 279       inc_temp_end: pop  psw
036F D0E0     + 280                    pop   acc
0371 22       + 281                    ret
0372 C0E0     + 282       dec_temp:    push  acc
0374 C0D0     + 283                    push  psw
0376 20100D   + 284                    jb    temp_21,dec_temp_end
0379 A208     + 285                    mov   c,sleep
037B C208     + 286                    clr   sleep
```

```
037D E521     + 287                     mov     a,led_data+1
037F 33       + 288                     rlc     a
0380 F521     + 289                     mov     led_data+1,a
0382 9210     + 290                     mov     temp_21,c
0384 1576     + 291                     dec     temp_set
0386 D0D0     + 292     dec_temp_end:   pop     psw
0388 D0E0     + 293                     pop     acc
038A 22       + 294                     ret
                295     $include"inc_hour.asm"
              + 296     $title"inc_hour ,dec_hour"
038B C0E0     + 297     inc_hour:       push    acc
038D C0D0     + 298                     push    psw
038F 20131C   + 299                     jb      hour_12,inc_hour_end
0392 C21F     + 300                     clr     data.31
0394 E523     + 301                     mov     a,led_data+3
0396 13       + 302                     rrc     a
0397 F523     + 303                     mov     led_data+3,a
0399 E522     + 304                     mov     a,led_data+2
039B 13       + 305                     rrc     a
039C F522     + 306                     mov     led_data+2,a
039E 92D1     + 307                     mov     psw.1,c         ;save carry flag
03A0 A211     + 308                     mov     c,timer_on      ;timer_on => timer_off
03A2 9212     + 309                     mov     timer_off,c
03A4 A210     + 310                     mov     c,temp_21       ;temp_21 => timer_on
03A6 9211     + 311                     mov     timer_on,c
03A8 A2D1     + 312                     mov     c,psw.1         ;restore carry flag
03AA 9210     + 313                     mov     temp_21,c       ;carry => temp_21
03AC 0575     + 314                     inc     hour
03AE D0D0     + 315     inc_hour_end:   pop     psw
03B0 D0E0     + 316                     pop     acc
03B2 22       + 317                     ret
03B3 C0E0     + 318     dec_hour:       push    acc
03B5 C0D0     + 319                     push    psw
03B7 201E1A   + 320                     jb      hour_1,dec_hour_end
03BA E522     + 321                     mov     a,led_data+2
03BC 33       + 322                     rlc     a
03BD F522     + 323                     mov     led_data+2,a
03BF E523     + 324                     mov     a,led_data+3
03C1 33       + 325                     rlc     a
03C2 F523     + 326                     mov     led_data+3,a
03C4 A211     + 327                     mov     c,timer_on      ;timer_on => temp_21
03C6 9210     + 328                     mov     temp_21,c
03C8 A212     + 329                     mov     c,timer_off     ;timer_off => timer_on
03CA 9211     + 330                     mov     timer_on,c
03CC A213     + 331                     mov     c,hour_12       ;hour_12 => timer_off
03CE 9212     + 332                     mov     timer_off,c
03D0 C213     + 333                     clr     hour_12
03D2 1575     + 334                     dec     hour
03D4 D0D0     + 335     dec_hour_end:   pop     psw
03D6 D0E0     + 336                     pop     acc
03D8 22       + 337                     ret
                338     $include"timer_on.asm"
03D9 C211     + 339     off_timer:      clr     timer_on
03DB C212     + 340                     clr     timer_off
03DD 9100     + 341                     acall   clr_hour
03DF 22       + 342                     ret
03E0 D211     + 343     on_timer_on:    setb    timer_on
03E2 C220     + 344                     clr     timer_off_status
03E4 C212     + 345                     clr     timer_off
```

```
03E6 9100     + 346                     acall   clr_hour
03E8 D21E     + 347                     setb    hour_1
03EA 757501   + 348                     mov     hour,#1
03ED 911F     + 349                     acall   temp_ctr        ;xxxxxxxxxxxxxxxxxxxx
03EF 22       + 350                     ret
03F0 D212     + 351     on_timer_off:   setb    timer_off
03F2 C221     + 352                     clr     timer_on_status
03F4 C211     + 353                     clr     timer_on
03F6 9100     + 354                     acall   clr_hour
03F8 D21E     + 355                     setb    hour_1
03FA 757501   + 356                     mov     hour,#1
03FD 911F     + 357                     acall   temp_ctr        ;xxxxxxxxxxxxxxxxxxxx
03FF 22       + 358                     ret
0400 C213     + 359     clr_hour:       clr     hour_12
0402 C214     + 360                     clr     hour_11
0404 C215     + 361                     clr     hour_10
0406 C216     + 362                     clr     hour_9
0408 C217     + 363                     clr     hour_8
040A C218     + 364                     clr     hour_7
040C C219     + 365                     clr     hour_6
040E C21A     + 366                     clr     hour_5
0410 C21B     + 367                     clr     hour_4
0412 C21C     + 368                     clr     hour_3
0414 C21D     + 369                     clr     hour_2
0416 C21E     + 370                     clr     hour_1
0418 22       + 371                     ret
                372     $include"cmp_temp.asm"
              + 373     ; temp_set >= temp ==> c = 0
              + 374     ; temp_set < temp ==> c = 1
0419 C3       + 375     cmp_temp:       clr     c
041A E576     + 376                     mov     a,temp_set
041C 956F     + 377                     subb    a,temp
041E 22       + 378                     ret
              + 379     ;################################################
041F C0D0     + 380     temp_ctr:       push    psw
0421 C0E0     + 381                     push    acc
0423 9119     + 382                     acall   cmp_temp
0425 4006     + 383                     jc      temp_ctr_1
0427 915F     + 384                     acall   off_comp
0429 9162     + 385                     acall   fan_low_on
042B 8018     + 386                     sjmp    temp_ctr_end
042D 900435   + 387     temp_ctr_1:     mov     dptr,#temp_ctr_table
0430 F4       + 388                     cpl     a
0431 2401     + 389                     add     a,#1
0433 23       + 390                     rl      a
0434 73       + 391                     jmp     @a+dptr
0435 0000     + 392     temp_ctr_table: db      00h,00h         ;0
0437 B14A     + 393                     ajmp    lo__o           ;1
0439 B150     + 394                     ajmp    me__d           ;2
043B B156     + 395                     ajmp    hi__i           ;3
043D B156     + 396                     ajmp    hi__i           ;4
043F B156     + 397                     ajmp    hi__i           ;5
0441 B156     + 398                     ajmp    hi__i           ;6
0443 B156     + 399                     ajmp    hi__i           ;7
              + 400
0445 D0E0     + 401     temp_ctr_end:   pop     acc
0447 D0D0     + 402                     pop     psw
0449 22       + 403                     ret
044A 915C     + 404     lo__o:          acall   on_comp
```

```
044C 9162     + 405                    acall   fan_low_on
044E 80F5     + 406                    sjmp    temp_ctr_end
0450 915C     + 407       me__d:       acall   on_comp
0452 9169     + 408                    acall   fan_med_on
0454 80EF     + 409                    sjmp    temp_ctr_end
0456 915C     + 410       hi__i:       acall   on_comp
0458 9170     + 411                    acall   fan_hi_on
045A 80E9     + 412                    sjmp    temp_ctr_end
              + 413       ;
045C C294     + 414       on_comp:     clr     p1.4                ;open
045E 22       + 415                    ret
045F D294     + 416       off_comp:    setb    p1.4                ;off
0461 22       + 417                    ret
0462 D295     + 418       fan_low_on:  setb    p1.5
0464 D296     + 419       '            setb    p1.6
0466 C297     + 420                    clr     p1.7
0468 22       + 421       .            ret
0469 D295     + 422       fan_med_on:  setb    p1.5
046B C296     + 423                    clr     p1.6
046D D297     + 424                    setb    p1.7
046F 22       + 425                    ret
0470 C295     + 426       fan_hi_on:   clr     p1.5
0472 D296     + 427                    setb    p1.6
0474 D297     + 428                    setb    p1.7
0476 22       + 429                    ret
              + 430       ;################################################
0477 C082     + 431       inc_delay:   push    dpl
0479 C083     + 432                    push    dph
047B 856B82   + 433                    mov     dpl,delay_l
047E 856C83   + 434                    mov     dph,delay_h
0481 A3       + 435                    inc     dptr
0482 85826B   + 436                    mov     delay_l,dpl
0485 85836C   + 437                    mov     delay_h,dph
0488 D083     + 438                    pop     dph
048A D082     + 439                    pop     dpl
048C 22       + 440                    ret
              + 441       ;################################################
048D C0E0     + 442       cmp_delay:   push    acc
048F C27A     + 443                    clr     equal_
0491 E56B     + 444                    mov     a,delay_l
0493 B46807   + 445                    cjne    a,#low(360),cmp_del_end
0496 E56C     + 446                    mov     a,delay_h
0498 B40102   + 447                    cjne    a,#high(360),cmp_del_end
049B D27A     + 448                    setb    equal_
049D D0E0     + 449       cmp_del_end: pop     acc
049F 22       + 450                    ret
              + 451
0000            452                    org     0000h
0000 8063       453       rstv:        sjmp    cld_start
                454       ; timer 0 interrupt service routine
                455       ; used for IC-555 triggering
000B            456                    org     000bh
000B C0E0       457                    push    acc
000D C0D0       458                    push    psw
000F 011B       459                    ajmp    trig_555
                460
                461       ; external interrupt 1 service routine
                462       ; processor come here as the falling edge of signal at int1 pin
                463       ; and at that time timer 1 stop
```

```
0013           464                     org     00013h
0013 B58B72    465                     mov     tmpl,tl1        ;save value of previous pulsewidth
0016 B58D73    466                     mov     tmph,th1
0019 0183      467                     ajmp    get_temp
               468
001B 0574      469     trig_555:       inc     count_gen
001D E574      470                     mov     a,count_gen
001F B40007    471                     cjne    a,#low(256),retrn       ;63.9 ms
0022 C293      472                     clr     p1.3
0024 757400    473                     mov     count_gen,#0
0027 D293      474                     setb    p1.3
0029 113C      475     retrn:          acall   inc_disp_count
002B 1152      476                     acall   cmp_disp_count
002D 307D07    477                     jnb     equal,retrn1
0030 E4        478                     clr     a
0031 F56D      479                     mov     disp_count_l,a
0033 F56E      480                     mov     disp_count_h,a
0035 B27C      481                     cpl     freq
0037 D0D0      482     retrn1:         pop     psw
0039 D0E0      483                     pop     acc
003B 32        484     return:         reti
               485
003C C082      486     inc_disp_count: push    dpl
003E C083      487                     push    dph
0040 856D82    488                     mov     dpl,disp_count_l
0043 856E83    489                     mov     dph,disp_count_h
0046 A3        490                     inc     dptr
0047 85826D    491                     mov     disp_count_l,dpl
004A 85836E    492                     mov     disp_count_h,dph
004D D083      493                     pop     dph
004F D082      494                     pop     dpl
0051 22        495                     ret
               496
0052 C0E0      497     cmp_disp_count: push    acc
0054 C27D      498                     clr     equal
0056 E56D      499                     mov     a,disp_count_l
0058 B4E807    500                     cjne    a,#low(1000),cmp_disp_end
005B E56E      501                     mov     a,disp_count_h
005D B40302    502                     cjne    a,#high(1000),cmp_disp_end
0060 D27D      503                     setb    equal
0062 D0E0      504     cmp_disp_end:   pop     acc
0064 22        505                     ret
               506
0065 D2AF      507     cld_start:      setb    ea              ;enable all interrupts
0067 75B800    508                     mov     ip,#0           ;set interrupt priority to 0
006A 75D000    509                     mov     psw,#0          ;select RB0
               510     ;
006D E4        511                     clr     a               ;init disp_count
006E F56D      512                     mov     disp_count_l,a  ; (used for .5 Hz freq gen.)
0070 F56E      513                     mov     disp_count_h,a  ;1000x230x1.085 us = .250 s
               514     ;
0072 F574      515                     mov     count_gen,a     ;init count_gen of values [0,1,2,3]
0074 11C9      516                     acall   trig_gen        ;generate 555-trigged signal
               517     ;
0076 20B3FD    518                     jb      int1,$          ;wait until int1 pin go low
0079 11DA      519                     acall   pulse_measure   ;start timer 1 for pulse measuring
               520     ;
007B D28A      521                     setb    it1             ;int1 is edge trigger
007D C28B      522                     clr     ie1             ;clear request
```

```
007F D2AA       523                     setb    ex1             ;enable external interrupt 1
0081 4100       524                     ajmp    init            ;wait existing interrupt
                525
0083 C0E0       526     get_temp:       push    acc
0085 C0D0       527                     push    psw
0087 758D00     528                     mov     th1,#0          ;timer will start again after
008A 758B00     529                     mov     tl1,#0          ;int1 pin go high
008D 741C       530                     mov     a,#28           ;start from 28 degree
008F C0E0       531     get_temp_1:     push    acc             ;-------------[
0091 3102       532                     acall   temp_l_upper
0093 F570       533                     mov     cmpl,a
0095 D0E0       534                     pop     acc             ;-------------]
0097 C0E0       535                     push    acc             ;-------------[
0099 3120       536                     acall   temp_h_upper
009B F571       537                     mov     cmph,a
009D D0E0       538                     pop     acc             ;-------------]
009F 11E9       539                     acall   compare         ;compare with upper limit
00A1 4002       540                     jc      less_upper
00A3 8016       541                     sjmp    next_check      ;pulse >= upper
00A5 C0E0       542     less_upper:     push    acc             ;-------------[ ; pulse < upper
00A7 313E       543                     acall   temp_l_lower
00A9 F570       544                     mov     cmpl,a
00AB D0E0       545                     pop     acc             ;-------------]
00AD C0E0       546                     push    acc             ;-------------[
00AF 315C       547                     acall   temp_h_lower
00B1 F571       548                     mov     cmph,a
00B3 D0E0       549                     pop     acc             ;-------------]
00B5 11E9       550                     acall   compare         ;compare with lower limit
00B7 4002       551                     jc      less_low
00B9 8007       552                     sjmp    temperature     ;lower<= pulse < upper
00BB            553     less_low:                               ;pulse < lower
00BB 15E0       554     next_check:     dec     acc
00BD B414CF     555                     cjne    a,#21-1,get_temp_1
00C0 8002       556                     sjmp    get_temp_2
00C2 F56F       557     temperature:    mov     temp,a
00C4 D0D0       558     get_temp_2:     pop     psw
00C6 D0E0       559                     pop     acc
00C8 32         560                     reti
                561     ; 555-trigged signal generated every 1 us
                562     ; timer 0 mode 2 : 8_bit auto_reload timer
00C9 D2A9       563     trig_gen:       setb    et0             ;enable timer 0 interrupt
00CB 5389F0     564                     anl     tmod,#0f0h      ;clear low nibble
00CE 438902     565                     orl     tmod,#02h       ;set low nibble to 2h
00D1 758A1A     566                     mov     tl0,#-230       ;230x256 cycles = 58.88k x 1.085 us
00D4 758C1A     567                     mov     th0,#-230       ;                = 63.9 ms
00D7 D28C       568                     setb    tr0             ;timer 0 run
00D9 22         569                     ret
                570     ; pulse-width measurement thru int1 pin
                571     ; timer 1 mode 1 : 16_bit timer EXTERNAL CONTROL
00DA            572     pulse_measure:  ;setb    et1             ;enable timer 1 interrupt for error
00DA 53890F     573                     anl     tmod,#0fh       ;clear high nibble
00DD 438990     574                     orl     tmod,#90h       ;set high nibble to 9h
00E0 758B00     575                     mov     tl1,#0          ;init timer 1 value
00E3 758D00     576                     mov     th1,#0
00E6 D28E       577                     setb    tr1             ;timer 1 run
                578     ;               (wait for logic 1 at int1 pin)
00E8 22         579                     ret
                580     ; compare value in tmph:tmpl with cmph:cmpl
                581     ; if tmph:tmpl > cmph:cmpl ==> equal = 0, c = 0
```

```
                    582         ; if tmph:tmpl = cmph:cmpl ==> equal = 1, c = 0
                    583         ; if tmph:tmpl < cmph:cmpl ==> equal = 0, c = 1
00E9 C0E0           584         compare:        push    acc
00EB C27D           585                         clr     equal           ;default equal = 0
00ED E573           586                         mov     a,tmph
00EF B57109         587         cmp_high:       cjne    a,cmph,high_not_eq
00F2 E572           588         high_eq:        mov     a,tmpl          ;tmph = cmph
00F4 B57004         589         cmp_low:        cjne    a,cmpl,low_not_eq
00F7 D27D           590         all_eq:         setb    equal           ;tmph = cmph, tmpl = cmpl
00F9 8004           591                         sjmp    cmp_end         ;c = 0, equal = 1
00FB                592         low_not_eq:     ;[tmph <> cmph] or [tmph = cmph but tmpl <> cmpl]
00FB 4002           593         high_not_eq:    jc      less_than
00FD 8000           594         greater_than:   sjmp    cmp_end         ;c = 0, equal = 0
00FF                595         less_than:                              ;c = 1, equal = 0
00FF D0E0           596         cmp_end:        pop     acc
0101 22             597                         ret
                    598         ; temperature table
0102 83             599         temp_l_upper:   movc    a,@a+pc
0103 22             600                         ret
                    601         ;
0104 31323334       602                         db      '12345678901234567890'  ;degree
0108 35363738
010C 39303132
0110 33343536
0114 37383930
0118 2A             603                         db      low(298)                ;21
0119 24             604                         db      low(292)                ;22
011A 1E             605                         db      low(286)                ;23
011B 18             606                         db      low(280)                ;24
011C 12             607                         db      low(274)                ;25
011D 0C             608                         db      low(268)                ;26
011E 06             609                         db      low(262)                ;27
011F 00             610                         db      low(256)                ;28
0120 83             611         temp_h_upper;   movc    a,@a+pc
0121 22             612                         ret
                    613         ;
0122 31323334       614                         db      '12345678901234567890'  ;degree
0126 35363738
012A 39303132
012E 33343536
0132 37383930
0136 01             615                         db      high(298)               ;21
0137 01             616                         db      high(292)               ;22
0138 01             617                         db      high(286)               ;23
0139 01             618                         db      high(280)               ;24
013A 01             619                         db      high(274)               ;25
013B 01             620                         db      high(268)               ;26
013C 01             621                         db      high(262)               ;27
013D 01             622                         db      high(256)               ;28
013E 83             623         temp_l_lower:   movc    a,@a+pc
013F 22             624                         ret
                    625         ;
0140 31323334       626                         db      '12345678901234567890'  ;degree
0144 35363738
0148 39303132
014C 33343536
0150 37383930
0154 27             627                         db      low(295)                ;21
0155 21             628                         db      low(289)                ;22
```

```
0156 1B          629                    db      low(283)                ;23
0157 15          630                    db      low(277)                ;24
0158 0F          631                    db      low(271)                ;25
0159 09          632                    db      low(265)                ;26
015A 03          633                    db      low(259)                ;27
015B FD          634                    db      low(253)                ;28
015C 83          635        temp_h_lower:  movc    a,@a+pc
015D 22          636                    ret
                 637        ;
015E 31323334    638                    db      '12345678901234567890'  ;degree
0162 35363738
0166 39303132
016A 33343536
016E 37383930
0172 01          639                    db      high(295)               ;21
0173 01          640                    db      high(289)               ;22
0174 01          641                    db      high(283)               ;23
0175 01          642                    db      high(277)               ;24
0176 01          643                    db      high(271)               ;25
0177 01          644                    db      high(265)               ;26
0178 01          645                    db      high(259)               ;27
0179 00          646                    db      high(253)               ;28
                 647        ;
017A             648        end_init:
017A 911F        649        l_1:        acall   temp_ctr
017C 516A        650                    acall   scandisp
017E 207F02      651                    jb      key_flag,check
0181 80F7        652                    sjmp    l_1
0183 E579        653        check:      mov     a,code
0185 B4B015      654        ch_pw_on:   cjne    a,#key_pw_on,ch_fan
0188 B202        655                    cpl     pw_norm
018A 200204      656                    jb      pw_norm,restart
018D 515C        657                    acall   clr_data
018F 80E9        658                    sjmp    l_1
0191 515C        659        restart:    acall   clr_data
0193 D202        660                    setb    pw_norm
0195 D204        661                    setb    fan_auto
0197 D207        662                    setb    fan_low
0199 D20C        663                    setb    temp_25
019B 80DD        664                    sjmp    l_1
019D B4B904      665        ch_fan:     cjne    a,#key_fan,ch_sleep
01A0 D204        666                    setb    fan_auto
01A2 80D6        667                    sjmp    l_1
01A4 B4E904      668        ch_sleep:   cjne    a,#key_sleep,ch_inc_temp
01A7 B208        669                    cpl     sleep
01A9 80CF        670                    sjmp    l_1
01AB B4D904      671        ch_inc_temp: cjne   a,#key_inc_temp,ch_dec_temp
01AE 7159        672                    acall   inc_temp
01B0 80C8        673                    sjmp    l_1
01B2 B47904      674        ch_dec_temp: cjne   a,#key_dec_temp,ch_timer_on
01B5 7172        675                    acall   dec_temp
01B7 80C1        676                    sjmp    l_1
01B9 B47118      677        ch_timer_on: cjne   a,#key_timer_on,ch_timer_off
01BC B221        678                    cpl     timer_on_status
01BE E524        679                    mov     a,status
01C0 5403        680                    anl     a,#00000011b
01C2 B40004      681                    cjne    a,#00b,ch_tm_on_1
01C5 71D9        682                    acall   off_timer
01C7 80B1        683                    sjmp    l_1
```

```
01C9 B40104     684       ch_tm_on_1:     cjne    a,#01b,ch_tm_on_2
01CC 71F0       685                       acall   on_timer_off
01CE 80AA       686                       sjmp    l_I
01D0 71E0       687       ch_tm_on_2:     acall   on_timer_on
01D2 80A6       688                       sjmp    l_I
01D4 B4D118     689       ch_timer_off:   cjne    a,#key_timer_off,ch_inc_hour
01D7 B220       690                       cpl     timer_off_status
01D9 E524       691                       mov     a,status
01DB 5403       692                       anl     a,#00000011b
01DD B40004     693                       cjne    a,#00b,ch_tm_off_1
01E0 71D9       694                       acall   off_timer
01E2 8096       695                       sjmp    l_1
01E4 B40204     696       ch_tm_off_1:    cjne    a,#10b,ch_tm_off_2
01E7 71E0       697                       acall   on_timer_on
01E9 808F       698                       sjmp    l_I
01EB 71F0       699       ch_tm_off_2:    acall   on_timer_off
01ED 8088       700                       sjmp    l_I
01EF B4E104     701       ch_inc_hour:    cjne    a,#key_inc_hour,ch_dec_hour
01F2 7188       702                       acall   inc_hour
01F4 8084       703                       sjmp    l_1
01F6 B4B181     704       ch_dec_hour:    cjne    a,#key_dec_hour,l_1
01F9 71B3       705                       acall   dec_hour
01FB 217A       706                       ajmp    l_1
                707
```

all_eq = 00F7	ch_dec_hour = 01F6	ch_dec_temp = 01B2	ch_fan = 019D	ch_inc_hour = 01EF
ch_inc_temp = 01AB	ch_pw_on = 01B5	ch_sleep = 01A4	ch_timer_off = 01D4	ch_timer_on = 01B9
ch_tm_off_1 = 01E4	ch_tm_off_2 = 01EB	ch_tm_on_1 = 01C9	ch_tm_on_2 = 01D0	check = 01B3
cld_start = 0065	clr_data = 025C	clr_hour = 0400	cmp_del_end = 049D	cmp_delay = 048D
cmp_disp_count = 0052	cmp_disp_end = 0062	cmp_end = 00FF	cmp_high = 00EF	cmp_low = 00F4
cmp_temp = 0419	cmph = 0071	cmpl = 0070	code = 0079	col = 007A
col_table = 0305	compare = 00E9	count_gen = 0074	data.0 = 0000	data.31 = 001F
dec_hour = 03B3	dec_hour_end = 03D4	dec_temp = 0372	dec_temp_end = 0386	delay_h = 006C
delay_l = 006B	delay_ok = 007B	disp_count_h = 006E	disp_count_l = 006D	end_init = 017A
entry = 007B	equal = 007D	equal_ = 007A	fan = 0077	fan_auto = 0004
fan_hi_on = 0470	fan_high = 0005	fan_low = 0007	fan_low_on = 0462	fan_med_on = 0469
fan_medium = 0006	freq = 007C	get_temp = 0083	get_temp_1 = 008F	get_temp_2 = 00C4
greater_than = 00FD	hi_i = 0456	high_eq = 00F2	high_not_eq = 00FB	hour = 0075
hour_1 = 001E	hour_10 = 0015	hour_11 = 0014	hour_12 = 0013	hour_2 = 001D
hour_3 = 001C	hour_4 = 001B	hour_5 = 001A	hour_6 = 0019	hour_7 = 0018
hour_8 = 0017	hour_9 = 0016	inc_col = 032C	inc_col_1 = 0336	inc_delay = 0477
inc_disp_count = 003C	inc_hour = 038B	inc_hour_end = 03AE	inc_temp = 0359	inc_temp_end = 036D
init = 0200	key_dec_hour = 00B1	key_dec_temp = 0079	key_fan = 00B9	key_flag = 007F
key_inc_hour = 00E1	key_inc_temp = 00D9	key_pw_on = 00B0	key_sleep = 00E9	key_timer_off = 00D1
key_timer_on = 0071	l_1 = 017A	led_data = 0020	less_low = 00BB	less_than = 00FF
less_upper = 00A5	lo_o = 044A	low_not_eq = 00FB	me_d = 0450	next_check = 00BB
off_comp = 045F	off_led = 02BA	off_timer = 03D9	on_comp = 045C	on_led = 02B7
on_timer_off = 03F0	on_timer_on = 03E0	power = 007B	pulse_measure = 00DA	pw_norm = 0002
pw_over = 0001	pw_under = 0003	released = 02DE	releaseflag = 007E	restart = 0191
retrn = 0029	retrn1 = 0037	return = 003B	rstv = 0000	sc_1 = 029A
sc_next = 02BE	scan_end = 02FE	scandisp = 026A	scankey = 02E4	set_key_flag = 02FA
sleep = 0008	stack = 0030	status = 0024	table = 030A	temp = 006F
temp_21 = 0010	temp_22 = 000F	temp_23 = 000E	temp_24 = 000D	temp_25 = 000C
temp_26 = 000B	temp_27 = 000A	temp_28 = 0009	temp_ctr = 041F	temp_ctr_1 = 042D
temp_ctr_end = 0445	temp_ctr_table = 0435	temp_h_lower = 015C	temp_h_upper = 0120	temp_l_lower = 013E
temp_l_upper = 0102	temp_set = 0076	temp_table = 033B	temperature = 00C2	timer_mask = 0079
timer_off = 0012	timer_off_status = 0020	timer_on = 0011	timer_on_status = 0021	tmp = 007C
tmph = 0073	tmpl = 0072	trig_555 = 001B	trig_gen = 00C9	wait_released = 0270

อ้างอิง

- ไอซี TTL

  THE WORLD TTL, ICDATA & CROSS - REFERENCE GUIDE

- คู่มือไอซี ไมโครโปรเซสเซอร์

  MICROPROCESSOR MCS-51 MICROCONTROLLER

- THE 8051 MICROCONTROLLER

  ARCHITECHTURE, PROGRAMMING,

  AND APPLICATIONS

  KENNETH J. AYALA

  WESTERN CAROLINA UNIVERSITY.