เครื่องวัดอุณหภูมิ 6 ช่อง

6 CHANNEL THERMOMETER

โดย

นาย ทรงสมศักดิ์ บุญมิ่ง

นาย วีระ บุญไพโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีอิเล็กทรอนิคส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

หัวข้อปริญญานิพนธ์ เครื่องวัดอุณหภูมิ 6 ช่อง

โดย นาย ทรงสมศักดิ์ บุญมิ่ง 37.013336

นาย วีระ บุญไพโรจน์ 37.013359

อาจารย์ที่ปรึกษา อ.มนชนก ศรีเสือขาม

อ.ไพศาล สิทธิโยภาสกุล

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2539

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับส่วนหนึ่งของการศึกษาตามหลักสูตร ปริญญาอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

..... ประธานกรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

เครื่องวัดอุณหภูมิ 6 ช่อง

6 CHANNEL THERMOMETER

โดย นาย วีระ บุญไพโรจน์
นาย ทรงสมศักดิ์ บุญมิ่ง

อาจารย์ที่ปรึกษา อ. มนชนก ศรีเสือขาม
อ.ไพศาล สิทธิโยภาสกุล

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นเรื่องเกี่ยวกับเครื่องวัดอุณหภูมิ 6 ช่อง ซึ่งเครื่องวัดอุณหภูมิ 6 ช่องเป็นเครื่องที่ใช้สำหรับวัดอุณหภูมิซึ่งสามารถวัดอุณหภูมิได้พร้อมกัน 6 ช่อง โดยการแสดงผลได้ทั้งจอ LCD และแสดงออกทางจอ COMPUTER โดยใช้ 68HC11 เป็นตัวควบคุม

ABSTRACT

THIS PROJECT IS CONCERNED ABOUT 6 CHANNEL THERMOMETER. 6 CHANNEL THERMOMETER IS MACHINE WHICH USED FOR MEASURE TEMPERATURE. IT CAN MEASURE TEMPERATURE IN SIX CHANNELS BY DISPLAY IN BOTH LCD MONITER OR COMPUTER. AND IT IS CONTROLLED BY 68HC11 .

สารบัญ

บทที่ 1
ความมุ่งหมายของปริญญานิพนธ์

1.1 บทนำ

การผลิต ผลิตภัณฑ์ต่างๆ ภายในโรงงานซึ่งต้องเกี่ยวข้องกับความร้อนและต้องรักษาค่าอุณหภูมิ ในขั้นตอนการผลิต ดังนั้นจึงมีความจำเป็นที่จะต้องมีอุปกรณ์ที่ใช้ในการตรวจเฝ้าค่าอุณหภูมิของขั้น ตอนการผลิตขั้นตอนต่างๆ เพื่อตรวจดูความผิดพลาดของอุณหภูมิที่จะเกิดขึ้นในขั้นตอนการผลิต

เครื่องวัดอุณหภูมิ 6 ช่อง มีความเหมาะสมในการใช้งานดังที่ได้กล่าวมาเนื่องจากเป็นโครงงานที่สามารถตรวจวัดค่าแสดงผลได้พร้อมๆกันถึง 6 ช่อง ซึ่งทำให้สามารถตรวจวัดค่าอุณหภูมิในขั้นตอนการผลิตในเวลาเดียวกันได้หลายขั้นตอน นอกจากนี้แล้วโครงงานนี้ยังสามารถแสดงผลได้บนจอไมโครคอมพิวเตอร์ซึ่งง่ายแก่การอ่านและเฝ้าตรวจวัดได้ตลอดเวลา

นอกจากนี้แล้วโครงงานนี้ยังสามารถแสดงผลได้บนจอผลึกเหลว จึงนำไปใช้งานได้กว้างขวางมากกว่าที่กล่าวมา และมีความสะดวกในการนำไปใช้งานนอกสถานที่ได้ดี

1.2 วัตถุประสงค์ของการทำปริญญานิพนธ์

1. เพื่อศึกษาไมโครคอนโทรลเลอร์ (Microtroller) เบอร์ 68HC11A1FN
2. เพื่อศึกษาวงจรไมโครคอมพิวเตอร์ชิพเดี่ยว (Single Chip Microcomputer)
3. เพื่อศึกษาวงจรเปลี่ยนค่าอุณหภูมิเป็นแรงดัน
4. เพื่อศึกษาและใช้งานโปรแกรมคอมพิวเตอร์ภาษา C
5. เพื่อศึกษาและใช้งานโปรแกรมคอมพิวเตอร์ภาษาแอสเซมบลี้ (Assembly)
6. เพื่อนำเอาวงจรเครื่องวัดอุณหภูมิ 6 ช่องไปใช้งานจริง

1.3 ขอบเขตของโครงงานปริญญานิพนธ์

1. สร้างวงจรไมโครคอมพิวเตอร์ชิพเดี่ยว
2. สร้างวงจรเปลี่ยนค่าอุณหภูมิเป็นแรงดัน
3. ออกแบบโปรแกรมการแสดงผลบนจอไมโครคอมพิวเตอร์
4. ศึกษาการใช้โปรแกรม PROTEL
5. สามารถนำโครงงานเครื่องวัดอุณหภูมิ 6 ช่อง ไปใช้งานได้จริง
6. สามารถแสดงผลการวัดอุณหภูมิบนหน้าจอไมโครคอมพิวเตอร์ได้

ในโครงงานเครื่องวัดอุณหภูมิ 6 ช่องนี้ได้เลือกใช้ไมโครคอนโทรลเลอร์ เบอร์ 68HC11A1FN เป็นตัวหลัก เนื่องจากไมโครคอนโทรลเลอร์ตัวนี้ประกอบไปด้วยส่วนต่างๆหลายอย่างที่เอื้ออำนวยต่อการทำโครงงานนี้ เช่น วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล (Analog to digital converter) วงจรสื่อสารอนุกรม (Serial communication) เป็นนอกจากนี้แล้วไมโครคอนโทรลเลอร์ตัวนี้ยังส่วนของพอร์ต (Port) ต่างๆ หน่วยความจำ ซึ่งสามารนำมาใช้งานได้สะดวกและสิ้นเปลืองอุปกรณ์ภายนอกน้อย

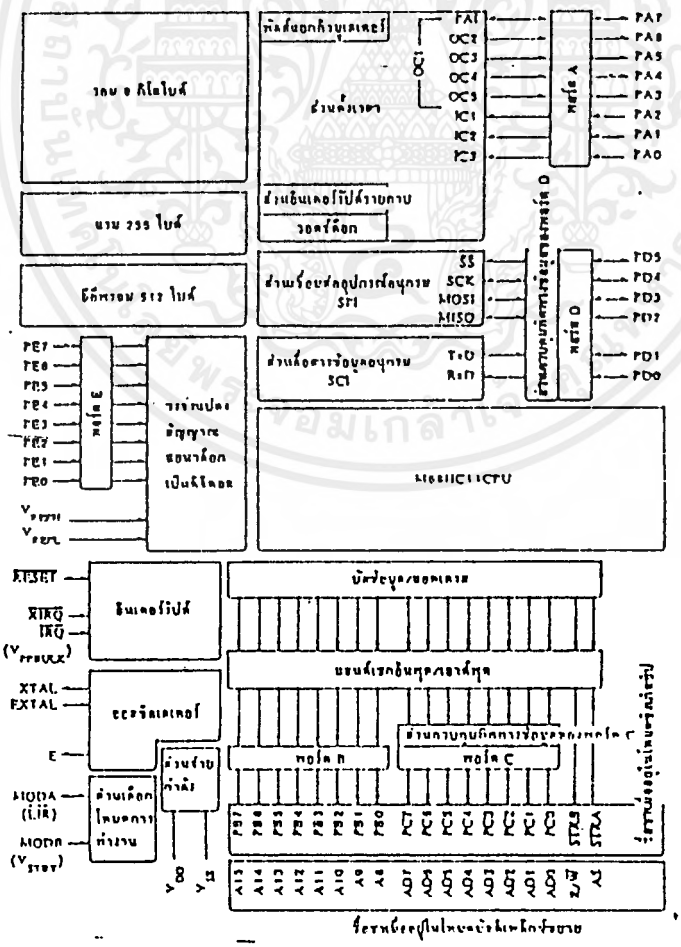
บทที่ 2

ไมโครคอนโทรลเลอร์และจอผลึกเหลว (LCD)

ไมโครคอนโทรลเลอร์

ในการออกแบบไมโครคอมพิวเตอร์เพื่อใช้ในงานควบคุม หรืองานเฉพาะอย่างโดยส่วนใหญ่จะใช้ไอซีประเภทไมโครคอนโทรลเลอร์ เพราะมีการทำงานที่รวดเร็วและเที่ยงตรงแม่นยำ นอกจากนั้นยังสามารถเปลี่ยนแปลงเงื่อนไขของงานโดยเพียงแต่เปลี่ยนซอฟแวร์ (Software) เท่านั้น เหตุผลอีกประการหนึ่งที่นิยมใช้ไมโครคอนโทรลเลอร์คือ มีการรวมอุปกรณ์สนับสนุนทางด้านอินพุทเอาท์พุทและอินเตอร์เฟสต่างๆ ไว้อย่างพร้อมมูล

68HC11A1FN เป็นไมโครคอนโทรลเลอร์อีกตัวหนึ่งของบริษัทโมโตโรล่า ที่ได้ถูกออกแบบมาโดยรวบรวมอุปกรณ์ที่จำเป็นในการใช้งานไว้ภายในตัวของมันเอง ได้แก่ อินพุท เอาท์พุท วงจรตั้งเวลา วงจรนับ วงจรอ่านสัญญาณแบบอะนาลอก ระบบการอินเตอร์รัพต์แบบเวกเตอร์อินเตอร์รัพต์ และระบบป้องกันความผิดพลาดที่เกิดขึ้นจากโปรแกรม รูปที่ 1 เป็นบล็อกไดอะแกรมการภายในของ 68HC11



รูปที่ 1 บล็อกไดอะแกรมภายในของ 68HC11A1FN

2.1 คุณสมบัติของ 68HC11A1FN

- เป็นซีพียูขนาด 8 บิต
- มีหน่วยความจำภายในสำหรับเก็บข้อมูลชนิดอีอีพรอม 512 ไบต์
- มีหน่วยความจำภายในสำหรับเก็บข้อมูลชนิดแรม 256 ไบต์
- มีวงจรตั้งเวลา/วงจรนับขนาด 16 บิต
- มีวงจรนับพัลซ์ขนาด 8 บิต
- มีวงจรรับส่งข้อมูลอนุกรมได้สองทิศทาง(Unversal Synchronous ReceiverTransmitter : USRT)
- ขนาดที่ใช้ในการติดต่อสำหรับวงจรการส่งข้อมูลอนุกรมเป็นแบบมัลติโปรเซสเซอร์
- มีวงจรเปลี่ยนสัญญาณอนาล็อกเป็นดิจิตอล 8 ช่อง
- มีวงจรรีลไทม์อินเตอร์รัพต์
- มีระบบวอทช์ด็อกที่ตั้งเวลาได้
- มีวงจรตรวจสอบสัญญาณนาฬิกาให้กับซีพียู
- มีระบบอินเตอร์รัพต์ 2 ระดับจาก 21 แหล่ง
- สามารถติดต่อกับหน่วยความจำภายนอกได้ 64 กิโลไบต์

2.2 รูปตัวถังของ 68HC11A1FN

68HC11A1FN เป็นไมโครคอนโทรลเลอร์ที่แบ่งออกเป็นหลายรุ่น โดยแบ่งตามลักษณะของหน่วยความจำที่อยู่ภายในซีพียู รูปแบบตัวถังของชิพตระกูลนี้มีด้วยกัน 3 แบบ คือ PLCC (Plastic Leaded Chip Carrier) เป็นตัวถังรูปสี่เหลี่ยม 52 ขา, แบบ DIP(Dual In-Line Package) มีขนาด 48 ขา ในตัวถังแบบนี้จะไม่มีขาสัญญาณของพอร์ต PE4-PE7 และแบบสุดท้ายคือ QFP(Quad Fiat Pack) เป็นตัวถังรูปสี่เหลี่ยมขนาด 64 ขา

2.3 โหมดการทำงาน

68HC11A1FN มีโหมดการทำงาน 4 โหมด สามารถกำหนดที่ขาสัญญาณ MODA และ MODB โดยมีลักษณะการทำงานดังนี้

1. SINGLE SHIP OPERATING MODE โหมดนี้โปรแกรมควบคุมการทำงานจะอยู่ภายในตัวซีพียู และขาสัญญาณต่างๆ สามารถใช้เป็นทั้งอินพุทและเอาท์พุท

2. EXPLANDED MULTIPLEXED OPERATING MODE การทำงานในโหมดนี้ โปรแกรมควบคุมการทำงานจะอยู่ภายนอก โดยใช้พอร์ตอินพุทเอาท์พุทเป็นแอดเดรสและข้อมูลในการติดต่อกับหน่วยความจำภายนอก และสัญญาณควบคุมต่างๆ

3. SPECIAL BOOTSTRAP OPERATING MODE การทำงานในโหมดนี้จะมีลักษณะคล้ายกับ SINGLE SHIP OPERATING MODE คือมีโปรแกรมควบคุมการทำงานจะอยู่ภายในตัวซีพียู แต่

ตำแหน่งในการรันและเวกเตอร์อินเตอร์รัพต์จะต่างกัน และมีลักษณะการทำงานที่พิเศษกว่า SINGLE SHIP MODE โดยสามารถป้องกันการคัดลอกหรือเลียนแบบได้

4. SPECIAL TEST OPERATING MODE โหมดนี้จะใช้ในการทดสอบการทำงานต่างๆตามโรงงานที่ผลิตเป็นจำนวนมาก

2.4 ลักษณะของสัญญาณและการจัดขา

RESET เป็นขาที่ทำงานแบบสองทิศทางคือ เป็นอินพุทสำหรับการทำให้ซีพียูเริ่มต้นการทำงานและเป็นเอาท์พุทแสดงถึงการทำงานภายในผิดพลาด ซึ่งเกิดขึ้นได้ 3 กรณี คือ ความผิดพลาดจากสัญญาณนาฬิกา ความผิดพลาดจากโปรแกรมในการใช้ระบบวอตช์ด็อก และการเอ็กซีคิวต์ออปโคดผิดพลาด

XTAL, EXTAL เป็นขาอินพุทสำหรับต่อคริสตอลเพื่อผลิตสัญญาณนาฬิกาให้กับซีพียูหรือจะใช้สัญญาณนาฬิกาจากภายนอกป้อนเข้าที่ขา EXTAL โดยตรงก็ได้ โดยที่มีตัวต้านทาน 10-100 กิโลโอห์มต่อลงกราวน์เพื่อป้องกันสัญญาณรบกวน

E เป็นเอาท์พุท โดยจะมีความถี่สัญญาณนาฬิกาน้อยกว่าความถี่สัญญาณนาฬิกาที่ขา EXTAL และ XTAL เป็น 4 เท่า และยังเป็นขาที่แสดงการทำงานของซีพียูด้วย คือ ถ้าเป็นสัญญาณลอจิก “0” แสดงว่าซีพียูกำลังประมวลผลภายใน และหากเป็นลอจิก “1” หมายถึงขณะนี้ซีพียูกับข้อมูล เมื่ออยู่ใน STOP โหมดจะไม่มีสัญญาณ E ออกมา

IRQ เป็นขาอินพุทที่ทำงานที่ลอจิก “0” หรือขอบขาลงเป็นการขออินเตอร์รัพต์ซีพียู โดยที่ขานี้ต้องต่อตัวต้านทานภายนอก 4.7 กิโลโอห์ม กับไฟเลี้ยง +5 โวลต์ เพื่อทำให้สภาวะปกติเป็นลอจิก “1”

XIRQเป็นขาอินพุททำงานที่ลอจิก“0”เป็นการขออินเตอร์รัพต์แบบนอนมาสเคเบิลอินเตอร์ รัพต์

MODA,MODB เป็นขาที่ใช้เลือกโหมดการทำงานของ CPU หลังจากที่ซีพียูเลือกโหมดการทำงานแล้ว ขา MODA จะเป็นขาเอาท์พุทลอจิก“0”เพื่อแสดงถึงการเฟตช์ซีพียูที่จะใช้ในการทำให้ซีพียูทำงานทีละคำสั่ง และขา MODB จะเป็นขาอินพุทสำหรับป้อนไฟเพื่อไปเลี้ยงหน่วยความจำภายใน เพื่อไม่ให้ข้อมูลสูญหายหลังจากที่ไม่มีไฟเลี้ยงซีพียู

VRL, VRH เป็นขาอินพุทสำหรับป้อนแรงดันอ้างอิงในวงจรแปลงอะนาล็อกเป็นดิจิตอล

STRB/R/W เป็นขาเอาท์พุทแสดงผลการทำงานในโหมด SINGLE SHIP ขานี้จะเป็นเอาท์พุทแสดงการแฮนด์เชคข้อมูล และเมื่อทำงานในโหมด EXPANDED MULTIPLEXED ขานี้จะเป็นเอาท์พุทแสดงการอ่านเขียนของซีพียู โดยถ้าเป็นลอจิก “0” หมายถึงการเขียนข้อมูล

STRA/AS เป็นทั้งขาอินพุทและเอาท์พุท ถ้าทำงานในโหมด EXPANDED MULTIPLEXED จะเป็นเอาท์พุทในการแยกสัญญาณระหว่างแอดเดรส 8 บิตล่าง กับสัญญาณข้อมูล

PORT A เป็นพอร์ตอินพุทเอาท์พุท โดยบิตที่ 0 ,1 ,2 จะถูกกำหนดเป็นอินพุท และบิตที่ 4 ,5 ,6 เป็นเอาท์พุท ส่วนบิตที่ 7 สามารถกำหนดเป็นอินพุทหรือเอาท์พุทก็ได้

PORT B เป็นพอร์ตเอาท์พุทเมื่อทำงานในโหมด SINGLE SHIP และจะเป็นแอดเดรส 8 บิตบน เมื่อทำงานในโหมด EXPANDED MULTIPLEXED

PORT C เป็นพอร์ตอินพุทเอาท์พุท เมื่อทำงานในโหมด SINGLE SHIP และ เมื่อทำงานในโหมด EXPANDED MULTIPLEXED พอร์ตนี้จะเป็นทั้งสายสัญญาณแอดเดรสและข้อมูล โดยมีสัญญาณ AS เป็นตัวแยกระหว่างแอดเดรสและข้อมูล ซึ่งต้องใช้งานร่วมกับ สัญญาณ R/W

PORT D เป็นพอร์ตอินพุตเอาท์พุทที่สามารถกำหนดได้ว่าจะให้บิตใดเป็นบิตอินพุทเอาท์พุท และยังใช้เป็นพอร์ตในการส่งและรับข้อมูลแบบอนุกรมได้อีกด้วย

PORT E เป็นพอร์ตอินพุทสำหรับใช้งานทั่วไป และยังใช้เป็นพอร์ตอินพุทสำหรับวงจรเปลี่ยนสัญญาณอะนาลอกเป็นดิจิตอล

2.5 การจัดหน่วยความจำ

68HC11A1FN เป็นซีพียูมีการจัดหน่วยความจำอย่างมีประสิทธิภาพ โดยสามารถติดต่อกับหน่วยความจำได้ถึง 64 กิโลไบต์ รวมทั้งหน่วยความจำที่อยู่ในตัวซีพียูด้วย

พื้นที่หน่วยความจำแอดเดรส 0000H-00FFH เป็นพื้นที่หน่วยความจำแรมที่อยู่ภายในชิพของทุกโหมดการทำงาน แอดเดรส 0100H - 0FFFH เป็นพื้นที่หน่วยความจำภายนอกในโหมดการทำงาน EXPANDEDMUX กับ SPECIAL TEST ที่แอดเดรส 1000H - 103FHเป็นที่ของริจิสเตอร์ภายในชิพของทุกโหมดการทำงานแอดเดรส 1040H - B5FFH เป็นพื้นหน่วยความจำภายนอกใน EXPANDED MUX กับ SPECIAL TEST ถัดมาที่แอดเดรส B600H- B7FFH เป็นพื้นที่หน่วยความจำอีพรอมภายในชิพ แอดเดรส BF40H-BFFFH เป็นพื้นที่บูตรอมในโหมด SPECIAL BOOT STARP

ส่วนแอดเดรสที่ B800H - DFFFH เป็นพื้นที่สำหรับหน่วยความจำภายนอกและที่แอดเดรส FFC0H-FFFFH เป็นพื้นที่สำหรับเก็บเวกเตอร์อินเตอร์รัพต์ของอินเตอร์รัพต์ทุกประเภท

2.6 หน่วยความจำแรมภายในซีพียู

หน่วยความจำแรมภายในซีพียูแบ่งออกเป็น 2 ประเภทคือ แรมที่ใช้ในการอ่านเขียนข้อมูลและริจิสเตอร์โดยตั้งแต่แอดเดรส 0000H-00FFH (256 ไบต์) เป็นหน่วยความจำที่ใช้ในการเขียนอ่านข้อมูลทั่วไป ซึ่งหน่วยความจำบริเวณนี้จะสามารถเก็บข้อมูลได้แม้ไฟเลี้ยงของวงจรจะหายไป ข้อมูลแรมส่วนนี้ในสภาวะปกติจะได้ไฟเลี้ยงจาก VDD แต่เมื่อ VDD หายไปจะรับไฟเลี้ยงจากอีกทางหนึ่ง โดยก่อนที่ VDD จะหายไปจะต้องมีสัญญาณรีเซตเพื่อทำให้ซีพียูหยุดการทำงานเพื่อป้องกันไม่ให้มีการเขียนข้อมูล

ใหม่ในหน่วยความจำ และเมื่อ VDD เข้ามาก็จะไปจ่ายให้กับแรมในส่วนนี้แทนไฟเลี้ยงอีกชุดหนึ่งได้ทันทีโดยที่ข้อมูลที่อยู่ในแรมจะไม่เปลี่ยนแปลง กระแสที่ใช้ในการเลี้ยงแรมนี้กระแสไฟน้อยมากจึงสามารถแบตเตอรี่เล็กๆเป็นไฟเลี้ยงให้กับแรมนี้ได้ ส่วนแรมอีกส่วนหนึ่งที่เป็นริจิสเตอร์จะถูกใช้ในการควบคุมการทำงานต่างๆของซีพียู โดยมีหน้าที่แตกต่างกันออกไปดังนี้

DDRD รีจิสเตอร์ควบคุมทิศทางการติดต่อของพอร์ต D

PORTE รีจิสเตอร์พอร์ต E ใช้ใช้สำหรับการอ่านข้อมูลของพอร์ต E

CFORC รีจิสเตอร์บอกผลการเปรียบเทียบของไทเมอร์ต่างๆ

OCIM รีจิสใช้สำหรับกำหนดบิตของข้อมูลในพอร์ต A ที่มีการเปรียบเทียบกับ OCI

OCID รีจิสเตอร์สำหรับเก็บข้อมูลการเปรียบเทียบการเปรียบระหว่างพอร์ต A กับ OCI

OCIM รีจิสเตอร์สำหรับเซตผลการเปรียบเทียบบิตต่อบิตระหว่างพอร์ต A กับรีจิสเตอร์ TOCI

TCNT รีจิสเตอร์ 16 บิต ใช้เป็นตัวฟรีเคาน์เตอร์ของระบบไทเมอร์ โดยจะทำการนับตั้งแต่ 0000-FFFF แล้วกลับมาเริ่มต้นใหม่

TICI รีจิสเตอร์ 16 บิต เป็นอินพุทไทเมอร์ 1

TIC2 รีจิสเตอร์ 16 บิต เป็นอินพุทไมเมอร์ 2

TIC3 รีจิสเตอร์ 16 บิต เป็นอินพุทไมเมอร์ 3

TOCI รีจิสเตอร์ 16 บิต ใช้ในการเปรียบเทียบกับไทเมอร์เคาน์เตอร์

TOC2 รีจิสเตอร์ 16 บิต ใช้ในการเปรียบเทียบกับไทเมอร์เคาน์เตอร์

TOC3 รีจิสเตอร์ 16 บิต ใช้ในการเปรียบเทียบกับไทเมอร์เคาน์เตอร์

TOC4 รีจิสเตอร์ 16 บิต ใช้ในการเปรียบเทียบกับไทเมอร์เคาน์เตอร์

TOC5 รีจิสเตอร์ 16 บิต ใช้ในการเปรียบเทียบกับไทเมอร์เคาน์เตอร์

TCTL1 เป็นรีจิสเตอร์ที่ใช้ในการเลือกลักษณะเอาท์พุทจากการใช้ฟังก์ชันการเปรียบเทียบเอาท์พุท

TCTL2 เป็นรีจิสเตอร์ที่ใช้ในการเลือกลักษณะของอินพุทที่ใช้เพื่อใช้ฟังก์ชันอินพุทแคบเจอร์ (input capture) CAP-TURE

TMSK1 เป็นรีจิสเตอร์ที่ใช้สำหรับอีนาเบิลหรือดีสเอเบิลอินเตอร์รัพต์ของตัวเปรียบเทียบเอาท์พุทกับตัวเปรียบเทียบอินพุท

TFLG1 เป็นรีจิสเตอร์ที่แสดงการเท่ากันของไทเมอร์เคาน์เตอร์กับตัวเปรียบเทียบเอาท์พุทกับตัวเปรียบเทียบอินพุท

TMSK2 เป็นรีจิสเตอร์ที่ใช้ในการอีนาเบิลหรือดิสเอเบิลอินเตอร์รัพต์ของรีลไทม์

TFLG2 เป็นริจิสเตอร์ที่ใช้แสดงการทำงานต่างๆ เป็นรีลไทม์อินเตอร์รัพต์โอเวอร์โฟลว์

PACTL เป็นรีจิสเตอร์ที่ใช้ในการควบคุมพัลส์แอกคิวมูเลเตอร์

PACNT เป็นรีจิสเตอร์เคาร์เตอร์ของพัลส์แอกคิวมูเลเตอร์

APCR เป็นริจิสเตอร์ที่ใช้ควบคุมการทำงานแบบ SPI (Serial Perpheral Interface)

SPSR เป็นรีจิสเตอร์ที่ใช้ในการบอกลักษณะการติดต่อแบบ SPI

SPDR เป็นรีจิสเตอร์สำหรับรับข้อมูลและส่งข้อมูลของ SPI

BACD เป็นรีจิสเตอร์ที่ใช้ในการกำหนดความเร็วในการส่งข้อมูลแลล SCI (Serial Communication Interface)

SCCRI เป็นรีจิสเตอร์ที่ใช้กำหนดรูปแบบในการส่งข้อมูลแบบ SCIS CCR2 เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการอินเตอร์รัพต์

SCCR เป็นรีจิสเตอร์ที่บอกสถานะการรับส่งข้อมูลแบบ SCI

SCDR เป็นรีจิสเตอร์ที่ใช้ในการเก็บข้อมูลการรับหรือส่งข้อมูลแบบ SCI

ADCTL เป็นรีจิสเตอร์ที่ในการเลือกข้อมูลแบบอะนาลอกและแสดงสถานะการอ่านข้อมูล

ADR1 เป็นรีจิสเตอร์ที่ใช้เก็บข้อมูลที่ได้จากวงจร A/D

ADR2 เป็นรีจิสเตอร์ที่ใช้เก็บข้อมูลที่ได้จากวงจร A/D

ADR3 เป็นรีจิสเตอร์ที่ใช้เก็บข้อมูลที่ได้จากวงจร A/D

ADR4 เป็นรีจิสเตอร์ที่ใช้เก็บข้อมูลที่ได้จากวงจร A/D

OPTION เป็นรีจิสเตอร์ที่ใช้กำหนดเวลาของการใช้ระบบ COP(Computer Operation Properly) การเปิดสัญญาณนาฬิกาสำหรับการแซมปลิ้งในวงจร A/D

COPRST การเปิดสัญญาณเลือกรีเซตค่าในระบบ COP

PPROG เป็นรีจิเตอร์ที่ใช้กำหนดรูปแบบการติดต่ออีพรอม

HPIRQ เป็นรีจิสเตอร์ที่ใช้จัดลำดับการอินเตอร์รัพต์

INIT เป็นรีจิสเตอร์ที่ใช้ในการกำหนดตำแหน่งของแรมและรีจิสเตอร์

TEST1 เป็นรีจิสที่ใช้ทดสอบของโรงงานของผู้ผลิต

CONFIG เป็นรีสจิเตอร์ที่ใช้ในการบ่งบอกอุปกรณ์ภายในตัวซีพียู

2.7 รีจิสเตอร์

68HC11A1FN มีรีจิสเตอร์ที่ใช้ในการเขียนโปรแกรมต่างๆ ทั้ง 8 บิตและ 16 บิตอยู่ด้วยกัน 7 ตัวได้แก่ แอกคิวมูเลเตอร์ Aและ B เป็นรีจิสเตอร์ขนาด 8 บิต ที่ใช้การคำนวณทางด้านคณิตศาสตร์ และ ตรรกศาสตร์ นอกนั้นยังรวมกันเป็นรีจิสเตอร์ D ขนาด 16 บิต โดยแอกคิวมูเลเตอร์ A เป็น 8 บิตสูง และแอกคิวมูเลเตอร์ B เป็น 8 บิตต่ำ

อินเด็กซ์รีจิสเตอร์ IX เป็นรีจิสเตอร์ขนาด 16 บิต ใช้เป็นตัวชี้ตำแหน่งของข้อมูลและเป็นตัวตั้งในการบวกแบบ 16 บิตกับแอกคิวมูเลเตอร์ B

อินเด็กซ์รีจิสเตอร์ IY เป็นรีจิสเตอร์ขนาด 16 บิต มีลักษณะคล้ายกับรีจิสเตอร์ IX แต่คำสั่งที่ใช้จะมีออปโค้ดขนาด 2 ไบต์

พอยน์เตอร์ SP เป็นรีจิสเตอร์ขนาด 16 บิต ชี้ตำแหน่งของสแต็คในการเก็บข้อมูลต่างๆจากการเรียกโปรแกรมย่อยหรือการอินเตอร์รัพต์ การเก็บข้อมูลลงในสแต็คมีลักษณะเป็นแบบเข้าก่อนออกทีหลัง (FILO)

โปรแกรมเคาน์เตอร์ PC เป็นรีจิสเตอร์ 16 บิตใช้เป็นตัวชี้ตำแหน่งของคำสั่งต่อไปที่จะประมวลผล

คอนดิชั่นโค้ดรีจิสเตอร์ CRR เป็นรีจิสเตอร์ขนาด 16 บิต เป็นตัวบ่งบอกลักษณะการทำงานตำแหน่งต่างๆของซีพียู และกำหนดการทำงานของซีพียู เช่นการกำหนดให้อีนาเบิลหรือดีสเอเบิลอินเตอร์รัพต์และการทำงานใน STOP โหมด

2.8 วงจรเอทูดี (ANALOG TO DIGITAL CONVERT : ADC)

เป็นสัญญาณแปลงสัญญาณอะนาลอกให้เป็นสัญญาณดิจิตอลขนาด 8 บิตโดยสามารถรับอินพุทได้ถึง 8 ช่องจะทำการรับครั้งละช่องโดยการโปรแกรมเลือกว่าจะทำการแปลงสัญญาณที่ช่องใดผลลัพธ์ที่ได้จากการแปลงจะเป็นอัตราส่วนระหว่างสัญญาณอินพุทกับแหล่งจ่ายที่เป็นไฟอ้างอิงของวงจร A/D โดยแรงดันที่ใช้อยู่ระหว่างในช่อง 2.5-5 โวลต์ เวลาที่ใช้ในการแปลงสัญญาณนั้นจะใช้สัญญาณนาฬิกา 32 ลูกต่อหนึ่งครั้ง หรือที่เวลา 16 ไมโครวินาที ที่ความถี่ 8 MHZ ผลที่อ่านเข้ามาได้แล้วจะถูกเก็บไว้ในรีจิสเตอร์ ADR1-ADR4

2.9 ไทเมอร์ (TIMER)

68HC11 มีไทเมอร์ขนาด 16 บิต ที่ทำงานในลักษณะฟรีรันนิ่งคือจะทำการนับตั้งแต่ 0000H-FFFFH และกลับมาเริ่มต้นใหม่โดยจะใช้สัญญาณนาฬิกาจากภายในตัวซีพียู ไทเมอร์นี้จะนำไปใช้เป็นตัวเปรียบเทียบกับรีจิสเตอร์ไทเมอร์อินพุทและไทเมอร์เอาท์พุท

รีจิสเตอร์ไทเมอร์อินพุท (input capture) เป็นรีจิสเตอร์ที่อ่านได้เพียงอย่างเดียวมีทั้งหมด 3 ตัวมีสัญญาณลอจิก "0" ที่ขาอินพุทของแต่ละตัวหลังจากที่ได้ทำการโหลดค่าฟรีรันไทเมอร์แล้ว สามารถที่จะทำการอินรัพต์ซีพียูได้ และยังนำค่าที่โหลดมา ใช้ในการวิเคราะห์คาบเวลาได้ว่าเป็นความถี่เท่าไร **รีจิสเตอร์ไทเมอร์เอาท์พุท** (out put compare) เป็นรีจิสเตอร์ที่เขียนและอ่านได้ทั้งหมด 5 ตัว อยู่ที่ตำแหน่งตั้งแต่แอดเดรส 1016H-101FH เราสามารถเขียนค่าลงไปยังรีจิสเตอร์ไทเมอร์เอาท์พุท เพื่อจะนำไปเปรียบกับฟรีรันไทเมอร์ เมื่อค่าทั้งสองเท่ากันจะทำการอินเตอร์รัพต์ซีพียูหรือทำการหารความถี่จากสัญญาณนาฬิกาไปยังเอาท์พุททั้ง 5 ตัวซึ่งสามารถควบคุมโดยการกำหนดค่าในรีจิสเตอร์ TMSK1 และ TCTL1

2.10 รีลไทม์ (REAL TIME)

เป็นวงจรส่วนหนึ่งที่ทำหน้าที่เป็นฐานเวลาให้กับโปรแกรม โดยสามารถโปรแกรมเลือกเวลาไปยังรีจิสเตอร์ PACTL ที่บิต RTR0 และ RTR1 บิต RTIF ในรีจิสเตอร์ TFLG2 จะถือเป็นตัวกำหนดว่าจะให้ทำการอินเตอร์รัพต์หรือไม่ ถ้าต้องการควบคุมให้บิต RTIF มีค่าเท่ากับ "1"

2.11 พัลส์แอกคิวมูเลเตอร์ (PULSE ACCUMULATOR)

68HC11 มีรีจิสเตอร์สำหรับนับพัลส์จากภายนอกขนาด 8 บิตโดยความถี่ที่สามารถนับได้สูงสุดจะเท่ากับความถี่สัญญาณภายในซีพียูหารด้วย 64 ลักษณะของการนับจะเป็นการนับขึ้น และถ้านับจนเกินค่าของค่ารีจิสเตอร์แล้วจะทำการอินเตอร์รัพต์ซีพียู โดยการเซ็ตที่บิต PAOVI ของรีจิสเตอร์ TMSK2

2.12 รีเซต (RESET)

การรีเซตของ 68HC11 จะแบ่งออกเป็น 3 ชนิดคือ

(1) การรีเซตจากภายนอกซึ่งเกิดจากการเปิดเครื่องครั้งแรกหรือสัญญาณรีเซตจากภายนอก

(2) การรีเซตจากวงจรวอตช์ด็อก

(3) การรีเซตเนื่องจากการผิดพลาดของสัญญาณนาฬิกาการรีเซตจากการเปิดเครื่องจะใช้จำนวนพัลส์สัญญาณนาฬิกา 4064 ลูก หลังจากที่สัญญาณนาฬิกาลูกแรกทำงาน ซึ่งถ้าใช้ คริส-ตอลภายนอกที่ 8 MHZ จะใช้เวลา 2 มิลลิวินาที ถ้าซีพียูทำงานในโหมด Bootstrap หรือ Special test หลังจากที่ซีพียูรีเซตจะไปเฟตช์ข้อมูลที่เวกเตอร์แอดเดรส FFFCH-FFFFH หรือ BFFEH-BFFFH ในกรณีการรีเซตจากวงจรวอตช์ด็อกจะทำให้สัญญาณที่รีเซตเป็นลอจิก "0" ด้วยและจะไปเฟตช์ข้อมูลที่เวกเตอร์แอดเดรส FFFAH-FFFBF และในกรณีการรีเซตจากสัญญาณนาฬิกาจะไปเฟตช์ข้อมูลที่เวกเตอร์แอดเดรส FFFCH-FFFDH

2.13 อินเตอร์รัพต์ (INTERRUPT)

ลักษณะการอินเตอร์รัพต์มี 2 แบบคือ ฮาร์ดแวร์และซอฟต์แวร์ อินเตอร์รัพต์ ฮาร์ดแวรอินเตอร์รัพต์ยังแบ่งออกได้เป็นการอินเตอร์จากภายในและจากภายนอก สามารถจัดลำดับความสำคัญของการขออินเตอร์รัพต์ได้ด้วยการควบคุมด้วยรีจิสเตอร์ HPRIO การอินเตอร์จะทำงานเป็นแบบเวกเตอร์อินเตอร์รัพต์

2.14 พอร์ตอนุกรม (SERIAL PORT)

พอร์ตอนุกรมของ 68HC11 มีลักษณะเหมือนพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ทั่วไป การรับส่งข้อมูลเป็นแบบฟูลดูเพล็กซ์คือ สามารถรับส่งข้อมูลได้พร้อมกันในเวลาเดียวกันก่อนจะรับส่ง

ข้อมูลมายังรีจิสเตอร์จะต้องอ่านข้อมูลเดิมออกไปก่อน มิฉะนั้นข้อมูลที่ส่งมาใหม่จะมาทับข้อมูลเดิม แอดเดรสของรีจิสเตอร์อยู่ที่ตำแหน่ง 102FH การรับส่งข้อมูลสามารถที่จะกำหนดการรับส่งข้อมูลได้ว่าเป็นแบบ 10 บิตหรือ 11 บิต แต่ลักษณะที่พิเศษของพอร์ตอนุกรมสำหรับ 68HC11 นี้คือ สามารถที่จะทำการตรวจสอบสัญญาณรบกวนได้โดยความกว้างของสัญญาณรบกวนจะต้องไม่เกิน 1/16 ของอัตราความเร็วในการส่ง

2.15 รีจิสเตอร์ของซีพียู

- ซีพียูของไมโครคอนโทรลเลอร์ 68HC11 จะมีรีจิสเตอร์ใช้งานอยู่ 7 ตัวอันได้แก่

1. แอกคิวมูลเลเตอร์ Aและ B
2. แอกคิวมูเลเตอร์ B
3. รีจิสเตอร์อินเด็กซ์ IX
4. รีจิสเตอร์อินเด็กซ์ IY
5. รีจิสเตอร์ตัวชี้สแต็ค (stack pointer: SP)
6. รีจิสเตอร์โปรแกรมเคาน์เตอร์ (program counter : PC)
7. รีจิสเตอร์ระหัสเงื่อนไข (condition code reister:CCR)

แอกคิวมูลเลเตอร์ A และ B

เป็นรีจิสเตอร์ขนาด 8 บิตใช้ในการเก็บค่าของโอเปอร์แรนด์และผลทางการคำนวนทางด้านคณิตศาสตร์ หรือผลจากการจัดการข้อมูลโดยตัวซีพียู ในการประมวลทางคณิตศาสตร์หรือลอจิก จะต้องนำข้อมูลเหล่านั้นมาเก็บไว้ในรีจิสเตอร์ทั้ง 2 ตัวนี้จึงจะสามารถประมวลได้

แอกคิวมูลเลเตอร์ D

เป็นรีจิสเตอร์ขนาด 16 บิต ใช้ในการประมวลผลและเก็บค่าจากการคำนวณทางคณิตศาสตร์และลอจิก แอกคิวมูเลเตอร์ Dก็เกิดมาจากการรวมกันของแอกคิวมูเลเตอร์ A และ B จึงทำให้มีขนาด 16 บิต

รีจิสเตอร์อินเด็กซ์ IX

เป็นรีจิสเตอร์ขนาด 16 บิต ใช้ในการชี้ตำแหน่งแอดเดรสเพื่อเข้าไปจัดการประมวลผลกับข้อมูลในแอดเดรสนั้นๆนอกจากนั้น IX ยังสามารถใช้เป็นตัวนับหรือรีจิสเตอร์เก็บข้อมูลชั่วคราวได้ด้วย

รีจิสเตอร์อินเด็กซ์ IY

เป็นรีจิสเตอร์ขนาด 16 บิต มีหน้าที่เหมือนกับ IX แต่จะแตกต่างกันตรงที่ในทุกคำสั่งที่เกี่ยวข้องกับ IY จะต้องมีข้อมูลไบต์พิเศษของระหัสแมชชีน และไซเกิลพิเศษช่วงเวลาในการเอ็กซีคิวต์คำสั่งด้วย จึงทำให้คำสั่งที่มี IY ไปเกี่ยวข้องต้องมีขนาดอย่างน้อย 2 ไบต์ขึ้นไปหรือที่เรียกว่า ฟรีไบต์ (prebyte)

รีจิสเตอร์ตัวชี้สแต็ค:SP

เป็นรีจิเตอร์ขนาด 16 บิต ใช้เก็บแอดเดรสบนสแต็ค โดยสแต็คใน 68HC11A1FN จะมีลักษณะการเก็บข้อมูลเข้าและนำข้อมูลออกมาเป็นแบบ LIFO (last-in-fist-out) หรือข้อมูลที่เข้าไปในสแต็คหลังสุดเมื่อจะเรียกออกมาจะถูกเรียกออกมาก่อน สแต็คที่ใช้เก็บข้อมูลของรีจิสเตอร์เมื่อต้องการนำรีจิสเตอร์ตัวต่อไปทำงานในโปรแกรมย่อยอื่นหรือต้องการในการตอบสนองการอินเตอร์รพต์ เพื่อเป็นการป้องข้อมูลเดิมสูญหายไปจึงต้องเก็บข้อมูลนั้นไว้ในหน่วยความจำสำรองแห่งหนึ่ง ซึ่งก็คือสแต็คนั้นเอง

รีจิสเตอร์โปรแกรม :PC

เป็นรีจิสเตอร์ 16 บิตใช้เก็บค่าแอดเดรสของคำสั่งถัดไปที่ซีพียูจะไปทำการเอ็กซีคิวซ์

รีจิสเตอร์รหัสเงื่อนไข : CCR

เป็นรีจิสเตอร์ขนาด 8 บิต ในแต่ละบิตจะแสดงความหมายของผลการกระทำคำสั่งที่เพิ่งจะเอ็กซิคิ้วไปของซีพียูดังแสดงในรูปที่ 6 แต่ละบิตของ CCR เป็นอิสระต่อกัน จึงสามารถตรวจสอบสถานะได้โดยการเขียนโปรแกรม และยังนำผลการตรวจสอบนั้นนำไปดำเนินการต่อไปได้

รายละเอียดของแต่บิตใน CCR มีดังนี้

- บิต CARRY/BORROW (C) : บิตนี้จะเซตเมื่อซีพียูทำการประมวลผลทางคณิตศาสตร์แล้วเกิดการทดค่า (carry) หรือยืมค่า (borrow) บิตนี้สามารถใช้งานร่วมกับคำสั่งการหมุน (retate) และเลื่อน (shift) ข้อมูลได้ หรือที่เรียกว่าบิตทด

- บิต OVER FLOW(V) บิตนี้จะเซตเป็น "1"เมื่อซีพียูกระทำตามคำสั่งทางคณิตศาสตร์อแล้วเกิดค่าที่เกินออกมานอกเหนือจากเงื่อนไขดังกล่าว บิตนี้จะเป็น "0"

- บิต ZERO (Z) บิตนี้จะเซตเมื่อผลของการกระทำทางคณิตศาสตร์หรือลอจิก หรือการประมวลผลข้อมูลแล้งทำให้เกิดค่าเป็นศูนย์ขึ้นมา

- บิต NEGATIVE (N) ถ้าหากผลการทำทางคณิตศาสตร์หรือลอจิก หรือการประมวลผลข้อมูลแล้วทำให้เกิดค่าลบ บิตนี้จะเซตผลลัพธ์ที่เป็นลบ จะสามารถที่จะสังเกตได้จากบิตมีนัยสำคัญสูสุด(MSB) มีค่าเป็น "1"

จอแสดงผลผลึกเหลว (Liquid Crystal Display : LCD)

2.16 คำสั่งควบคุมการทำงานของ LCD โมดูล

คำสั่งควบคุมการทำงานของ LCD โมดูลที่ใช้คอนโทรลเลอร์ของฮิตาชิตามคาด้านนั้นมีความยาวประมาณ 30 หน้าสำหรับคำอธิบายครบทุกคุณสมบัติของ LCD โมดูล การนำเสนอในที่นี้คงจะทำได้เฉพาะคำสั่งที่มีความสำคัญ และควรทราบครอบคลุมการใช้งานทั่วไปได้เพียงเท่านั้น ความสามารถหลักของLCDโมดูลซึ่งทำให้นิยมถูกหยิบยกมาใช้งานก็เพราะมันสามารถแสดงผลตัวอักขระแอสกี (ASCII character)ซึ่งมีอยู่ในคอนโทรลเลอร์ของ LCD โมดูลและใน LCD โมดูล แต่ละรุ่นก็มักใช้คอนโทรลเลอร์เบอร์เดียวกันเสมอนั้นคือคำสั่งควบคุมต่างๆจะใช้เหมือนกันใน LCD โมดูลแต่ละรุ่น เช่น ตารางชุดคำสั่งทั่วไปซึ่งการใช้งานกับ LCD โมดูทั่วไป เช่น รุ่น H2570,LM016L,LM16112A เป็นต้น

ก่อนที่จะเริ่มต้นรูปแบบคำสั่งควบคุมต่างๆเราควรทราบถึงศัพท์บางคำเกี่ยวกับองค์ประกอบและการทำงานพื้นฐานของคอนโทรลเลอร์ใน LCD โมดู,กันก่อน การทำงานภายใน LCD โมดูลจะมีบัฟเฟอร์อยู่ภายในซึ่งมีความจุขนาด 80 ตัวอักขระมีชื่อเรียกว่า DDRAM (Display Data RAM) มีตำแหน่งแอสเดรสอยู่ระหว่าง 000H ถึง 04FH ยกตัวอย่างเช่นถ้าเป็น LCD โมดูลขนาด 16 ตัวอักษร 1 บรรทัด DD-RAM ที่เก็บตัวอักขระที่จะแสดงผลทั้ง 16 ตัวอักษรจะถูกเก็บที่ตำแหน่งแอดเดรส 00H ถึง 0FH โดยเริ่มต้นทางด้านซ้ายของจอแสดงผล

การทำให้เกิดช่องว่าง (Windows) ใน DD-RAM สามารถทำโดยการใช้คำสั่งชิฟต์ตัวอักขระในการแสดงผล หรืออีกวิธีหนึ่งที่ง่ายมากขึ้นก็คือในขณะที่มีข้อมูลตัวอักขระเก็บไว้ในแอดเดรสค่าหนึ่ง การแสดงผลตัวอักขระต่อไปหรือทำให้เกิดช่องว่างอาจทำได้โดยการกำหนดจุดเริ่มต้นของแอดเดรสใหม่แตกต่างจากเดิมไป 1 แอดเดรสของ DD-RAM ซึ่งก็ทำให้ได้ผลลัพธ์เช่นเดียวกัน

ในกรณีที่เป็นจอแสดงผล LCD โมดูลแบบ 2 บรรทัดเช่นในรุ่น LM16255 เป็นต้น บรรทัดแรกจะใช้ตำแหน่งแอดเดรสเริ่มต้นที่ 000H สำหรับตัวอักขระเริ่มต้นและในบรรทัดที่ 2 จะเริ่มต้นที่ตำแหน่งแอดเดรส 040H การเขียนโปรแกรมสำหรับ LCD โมดูลแบบ 2 บรรทัดแน่นอนย่อมมีความซับซ้อนกว่า LCD โมดูลแบบ 1 บรรทัด แต่นั้นไม่ใช่ปัญหาใหญ่หากเข้าใจการควบคุมการทำงาน

ผู้เขียนโปรแกรมหรือผู้ใช้งานสามารถทราบถึงการแสดงผลตัวอักขระต่อไปที่จะแสดงผลด้วยเคอร์เซอร์ (cursor) ซึ่งเป็นตัวชี้ตำแหน่งต่อไปของตัวอักขระที่จะแสดงผลใน DD-RAM และตำแหน่งนี้ถูกเรียกว่าแอดเดรสเคาน์เตอร์ (address counter) เคอร์เซอร์สามารถกำหนดให้มีการแสดงผลหรือไม่ก็ได้ขึ้นอยู่กับการเขียนโปรแกรม และกำหนดให้มีการแสดงผลแบบกะพริบก็ได้

โดยทั่วไปการเขียนโปรแกรมเพื่อควบคุมการแสดงผล LCD เมื่อมีการส่งข้อมูลตัวอักขระไปยัง LCD คือการเลื่อนตัวอักขระเดิมไปและเลื่อนเคอร์เซอร์ตามไปด้วยเพื่อชี้ตำแหน่งของตัวอักขระต่อไป การเขียนโปรแกรมให้แสดงผลแบบนี้ทำให้ต่อการอ่านและเข้าใจของผู้ใช้ได้มากที่สุด

ใน LCD โมดูลมีหน่วยความจำอีกส่วนหนึ่งที่สำคัญก็คือ CG-RAM ซึ่งทำหน้าที่เก็บข้อมูลรายละเอียดโครงร่างของตัวอักขระที่สามารถนำมาแสดงผลได้หน่วยความจำในส่วนนี้ผู้ใช้สามารถที่จะสร้าง

ตัวอักขระใดๆได้ตามต้องการแล้วเก็บลงในหน่วยความจำส่วนสำหรับเรียกใช้งานได้ด้วย ถ้าตัวอักขระที่มีอยู่นั้นไม่พอเพียงต่อการใช้งานการใช้งานส่วนของ CG RAM นี้สามารถดูได้จากคาต้าชีตของ LCD โมดูลรุ่นนั้นๆที่ซื้อมาใช้งานได้

การส่งคำสั่งควบคุมมายัง LCD โมดูลสามารถกำหนดเป็นระหัสคำสั่งต่างๆซึ่งจะกล่าวต่อไปนี้ ออกที่ตำแหน่งแอดเดรส 0C001H โดยกำหนดให้ขาสัญญาณ R/W

คำสั่งเคลียร์จอแสดง (Clear Disply)

D7							D0
0	0	0	0	0	0	0	1

เมื่อคอนโทรลเลอร์ประมวลผลใน 20H=space ในรหัสแอสกีทำให้จอแสดงผลไม่ปรากฏตัวอักขระใดๆบนจอภาพ เคอร์เซอร์จะถูกเซ็ตให้อยู่ที่ตำแหน่งเริ่มต้นใหม่อีกครั้งและยกเลิกผลจากการใช้คำสั่งเลื่อนข้อมูล (Display Shift) ที่ผ่านมาแล้ว

คำสั่งเลื่อนเคอร์เซอร์ไปยังตำแหน่งเริ่มต้น (Return H0me)

D7							D0
0	0	0	0	0	0	0	.

คำสั่งนี้มีผลทำให้รีเซ็ตเคอร์เซอร์ให้กลับไปอยู่ที่ตำแหน่งเริ่มต้นใหม่ และรีเซ็ตคำสั่งเลื่อนข้อมูลตัวอักขระใน DD-RAM ไม่เกิดการเปลี่ยนแปลง นั่นคือตัวอักขระบนจอแสดงผลจะยังคงเหมือนเดิมไม่เปลี่ยนแปลง

คำสั่งกำหนดโหมดป้อนข้อมูล (Entry Mode Set)

D7							D0
0	0	0	0	0	1	I/D	S

คำสั่งจะถูกประโยชน์ในการกำหนดการทำงานของ LCDโมดูลหลังจากเกิดการส่งไบต์ข้อมูลไปยังจอแสดงผล บิต I/D (increase/decrease) ทำหน้าที่กำหนดการเพิ่ม (I/D=1) หรือลด (I/D=0) ค่าตำแหน่งแอดเดรสใน DD-RAM แอดเดรสอัตโนมัติเกิดการอ่านหรือเขียนตัวอักขระ ค่าของตำแหน่งแอดเดรสเคาน์เตอร์ (AC)

บิต S (shift bit) เป็นที่ใช้กำหนดลักษณะการใช้ข้อมูลตัวอักขระอย่างอัตโนมัติ โดยถ้าบิต S=1 เมื่อมีการส่งไบต์ข้อมูลใหม่เกิดขึ้นตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักขระข้อมูลเดิมจะถูกดันไปทางซ้าย แต่ถ้าหากบิต S=0 เมื่อเกิดข้อมูลใหม่ตัวเคอร์เซอรจะเลื่อนไปทางขวา

คำสั่งควบคุมการแสดงผล (Display ON/OFF Control)

D7							D0
0	0	0	0	1	D	C	B

คำสั่งนี้ใช้สำหรับควบคุมการแสดงผลหน้าจอและเคอร์เซอร์ โดยปราศจากการเปลี่ยนแปลงข้อมูล DD-RAM ผู้เขียนโปรแกรมสามารถกำหนดหน้าจอแสดงผลให้ปิดหรือเปิดได้ด้วยบิต D โดยกำหนดให้บิต D=1 เป็นการเปิดจอแสดงผล และถ้า D=0 เป็นการปิดจอแสดงผลเช่นเดียวกับบิต C=1 และบิต C=0 เป็นการควบคุมให้คอร์เซอร์เปิดหรือปิดตามลำดับ และบิต B ซึ่งเป็นบิตกำหนดว่าจะให้คอร์เซอร์กะพริบหรือไม่

คำสั่งควบคุมการเลื่อนเคอร์เซอร์และตัวอักขระ (Cursor or Display)

D7							D0
0	0	0	1	S/C	R/L	.	.

คำสั่งนี้ใช้สำหรับควบคุมการเลื่อนของคอร์เซอร์และตัวอักขระที่แสดงผลซึ่งมีความสำคัญและใช้งานบ่อยเนื่องจากผู้เขียนโปรแกรมต้องแสดงผลข้อความบนจอแสดงผลตามแนวนอนผลที่เกิดจากการกำหนดสถานะของบิต S/C และ R/L แสดงในตารางที่ 2

คำสั่งเซ็ตฟังก์ชั่น (Function Set)

D7							D0
0	0	1	DL	N	F	.	.

คำสั่งนี้ใช้สำหรับเซ็ตโหมดการทำงานของ LCD โมดูลหลังจากรีเซ็ตการทำงานหรือเริ่มต้นการทำงานของระบบทุกครั้งในที่นี้เราจะใช้งาน LCD โมดูลในโหมด 8 บิต และการใช้เพียง 1 บรรทัดเท่านั้น ความหมายของบิตต่างๆมีดังนี้

บิต DL=1 หมายถึงการทำงานในโหมดอินเตอร์เฟสแบบ 8 บิต

บิต DL=0 หมายถึงการทำงานในโหมดอินเตอร์เฟสแบบ 4 บิต

บิต N=0 หมายถึงทำงานแบบ 1 บรรทัด

บิต N=1 หมายถึงทำงานแบบ 2 บรรทัดหรือมากกว่า

บิต F=0 หมายถึงทำงานในโหมดความละเอียด 5x7 จุด

บิต F=1 หมายถึงทำงานในโหมดความละเอียด 5x10 จุด

คำสั่งเซ็ตตำแหน่งแอดเดรสใน CG-RAM (Set CG-RAM Address)

D7							D0
0	1	A5	A4	A3	A2	A1	A0

คำสั่งนี้ใช้สำหรับกำหนดค่าตำแหน่งแอดเดรสใน CG-RAM ภายใน LCD โมดูลเพื่อทำการส่งข้อมูลโดยกำหนดตำแหน่งแอดเดรสใน CG-RAM แน่นอนค่าหนึ่งเพื่อถ่ายทอดไบต์ข้อมูลต่อไป บิต a0-a5 เป็นบิตกำหนดตำแหน่งแอดเดรสใน CG-RAM ซึ่งถูกโหลดไปเก็นไว้ในแอดเดรสเคารน์เตอร์ AC

คำสั่งเซ็ตตำแหน่งแอดเดรสใน CG-RAM (Set DD-RAM Address)

D7							D0
1	A6	A5	A4	A3	A2	A1	A0

คำสั่งนี้ใช้สำหรับกำหนดค่าตำแหน่งแอดเดรสใน DD-RAM ภายใน LCD โมดูลเพื่อทำการส่งข้อมูล โดยกำหนดตำแหน่งแอดเดรสใน DD-RAM แน่นอนค่าหนึ่งเพื่อถ่ายทอดไบต์ข้อมูลต่อไป บิต a0-a6 เป็นบิตกำหนดตำแหน่งแอดเดรสใน DD-RAM ซึ่งจะถูกโหลดไปเก็บไว้ในแอดเดรสเคาน์เตอร์ AC

คำสั่งอ่านแฟลกบิวซี (Read Busy Flag)

D7							D0
BF	A6	A5	A4	A3	A2	A1	A0

เมื่อต้องการอ่านสถานะของแฟลกบิวซีต้องกำหนดขาสัญญาณ R/W เป็น "1" เสมอ แฟลกบิวซีเป็นตัวบอกสถานะการทำงานของ LCD โมดูลว่าอยู่ในสถานะพร้อมรับข้อมูลอยู่หรือไม่ โดยถ้า BF=1 หมายถึงขณะนี้ LCD โมดูลยังไม่พร้อมรับข้อมูลต่อไปอันเนื่องจากขบวนการประมวลจากคำสั่งหรือไบต์ข้อมูลที่ผ่านมายังไม่เสร็จสิ้น แต่ถ้า BF=0 หมายถึงขณะนี้ LCD โมดูลพร้อมที่จะรับคำสั่งใหม่หรือข้อมูลใหม่ได้แล้ว นอกจากคำสั่งนี้ทำให้ทราบสถานะของแฟลกบิวซีแล้วในขณะเดียวกันที่บิต a0-a6 จะถูกอ่านด้วยซึ่งเป็นค่าที่เก็บอยู่ในแอดเดรสเคาน์เตอร์ AC

คำสั่งเขียนข้อมูลไปยัง CG หรือ DD-RAM (Write Data to CG or DD-RAM)

D7	D6	D5	D4	D3	D2	D1	D0
D7							D0

เมื่อต้องการใช้คำสั่งนี้ต้องกำหนดให้ขาสัญญาณ R/W=0 และ RS=1 เสมอ คำสั่งนี้ใช้สำหรับเขียนข้อมูลไปยัง CG หรือ DD-RAM ในตำแหน่งแอดเดรสที่ต้องการ ซึ่งขึ้นอยู่กับการกำหนดตำแหน่งแอดเดรส CG-RAM ก่อนหน้าที่จะส่งข้อมูลนี้แล้ว คำสั่งนี้จะมีผลเกี่ยวเนื่องถึงการกำหนดโหมดการทำงานก่อนหน้านี้ด้วยว่าจะทำให้เกิดการเพิ่มหรือลดค่าใน AC หลังจากส่งผ่านไบต์ข้อมูลแล้วซึ่งเป็นผลมาจากคำสั่งกำหนดโหมดป้อนข้อมูล (entry mode command)

คำสั่งอ่านข้อมูลจาก CG หรือ DD-RAM

(Read Data from CG or DD-RAM)

D7							D0
D7	D6	D5	D4	D3	D2	D1	D0

เมื่อต้องการใช้คำสั่งนี้ต้องกำหนดให้ขาสัญญาณ R/W=1 และ RS =1 เสมอ คำสั่งนี้ใช้สำหรับอ่านข้อมูลจาก CG หรือ DD-RAM ในตำแหน่งแอดเดรสที่ต้องการ ขึ้นอยู่กับการกำหนดตำแหน่งแอดเดรสใน CG หรือ DD-RAM ก่อนหน้าที่จะใช้คำสั่งอ่านข้อมูลนี้แล้ว

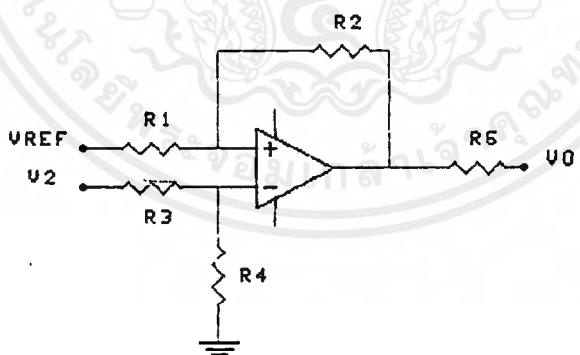
บทที่ 3
โครงสร้างของระบบโครงงาน

บทนี้จะกล่าวถึงรายละเอียดของระบบโครงสร้างทางฮาร์ดแวร์ของ โครงงานเครื่องวัดอุณหภูมิ 6 ช่อง โครงสร้างทางฮาร์ดแวร์ของเครื่องวัดอุณหภูมิ 6 ช่อง แบ่งเป็นส่วนใหญ่ๆ ได้ 2 ส่วน ดังนี้ คือ

- วงจรเปลี่ยนค่าอุณหภูมิเป็นแรงดัน
- วงจรควบคุม

3.1 วงจรเปลี่ยนค่าอุณหภูมิเป็นแรงดัน

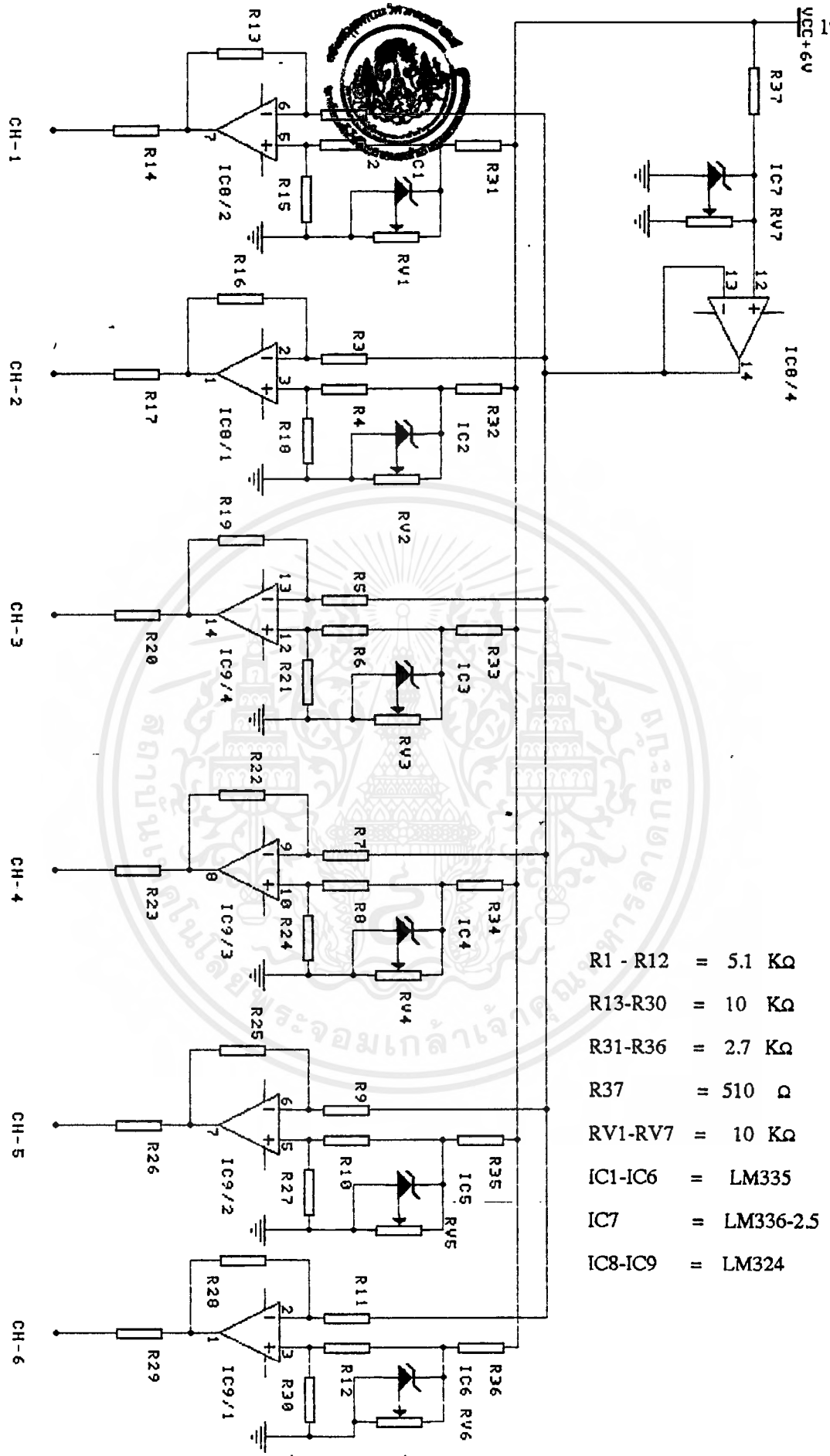
วงจรแสดงให้เห็นดังรูปที่ 3.2 โดยจากวงจรซึ่งประกอบด้วยวงจรดิฟเฟอเรนเชียล(Differential) จำนวน 6 ช่อง เหมือนๆกัน ซึ่งสามารถแยกพิจารณาได้จากวงจรดิฟเฟอเรนเชียลเพียงช่องเดียวเท่านั้น ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 วงจรดิฟเฟอร์เรนเชียล

วงจรตามรูปที่ 3.1 เป็นวงจรดิฟเฟอเรนเชียล โดยกำหนดให้ R1 = R3 = 5.1 kΩ และ R2 = R4 = 10 kΩ ซึ่งทำให้วงจรนี้มีอัตราขยายแรงดัน = R2/R1 และมีแรงดันเอาท์พุท (Vo) คือ

$$Vo = (V2 - Vref)R2 / R1$$



R1 - R12 = 5.1 KΩ
R13-R30 = 10 KΩ
R31-R36 = 2.7 KΩ
R37 = 510 Ω
RV1-RV7 = 10 KΩ
IC1-IC6 = LM335
IC7 = LM336-2.5
IC8-IC9 = LM324

รูปที่ 3.2 วงจรเปลี่ยนค่าอุณหภูมิเป็นแรงดัน

IC1 #LM335 มีคุณสมบัติต่างดังรายละเอียดที่กล่าวมาแล้วในบทที่ 2 โดยที่ LM335 มีค่าในการเปลี่ยนแปลงค่าอุณหภูมิไปเป็นแรงดันคือ 10 mV./ 1 °K นั้นคือ ถ้าอุณหภูมิเปลี่ยนแปลงไป 1° K จะทำให้แรงดันของ LM335 เปลี่ยนแปลงไป 10 mV. สำหรับ RV1 มีค่า 10 KΩ ทำหน้าที่เป็นตัวปรับแต่งให้ค่าแรงดันของ LM335 มีค่ามาตราฐานคือ ปรับให้ LM335 มีแรงดัน 2.985 V. ที่อุณหภูมิ 25 °C

3.2 วงจรส่วนควบคุม

วงจรส่วนควบคุมแสดงดังรูปที่ 3.5 วงจรส่วนนี้ เป็นวงจรส่วนที่ใช้ควบคุมการทำงาน และการประมวลผลข้อมูล วงจรส่วนนี้ทำหน้าที่ต่างๆด้วยกันดังนี้

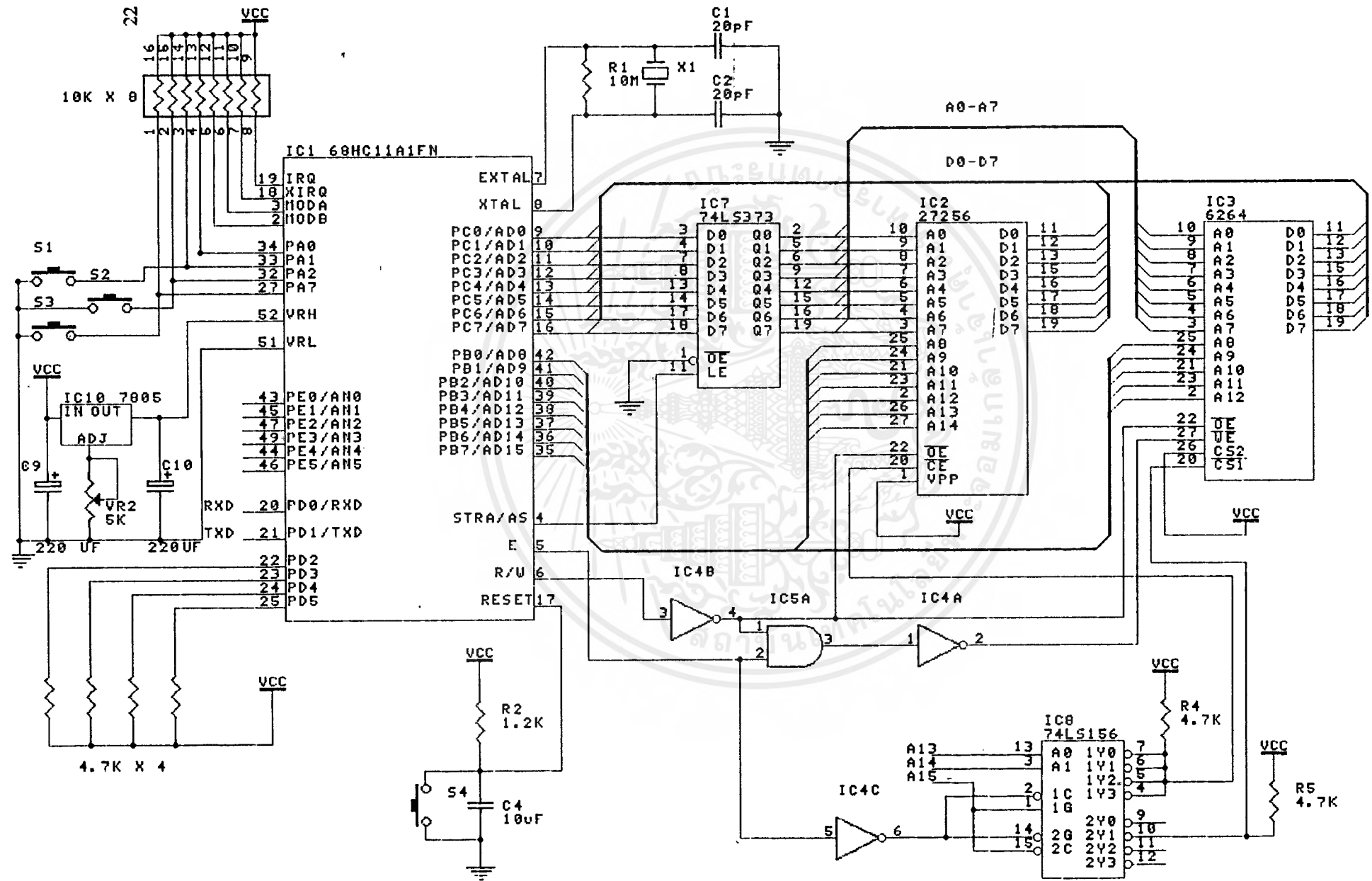
1. ทำหน้าที่เปลี่ยนแปลงสัญญาณอนาล็อก (ANALOG) เป็นสัญญาณดิจิตอล (DIGITAL)
2. ทำหน้าที่ควบคุมการแสดงผลบนจอแสดงผลแบบผลึกเหลว(LCD)
3. ทำหน้าที่ควบคุมการส่งข้อมูลไปให้ไมโครคอมพิวเตอร์ (Microcomputer)

โดยโครงสร้างส่วนนี้ประกอบไปด้วย

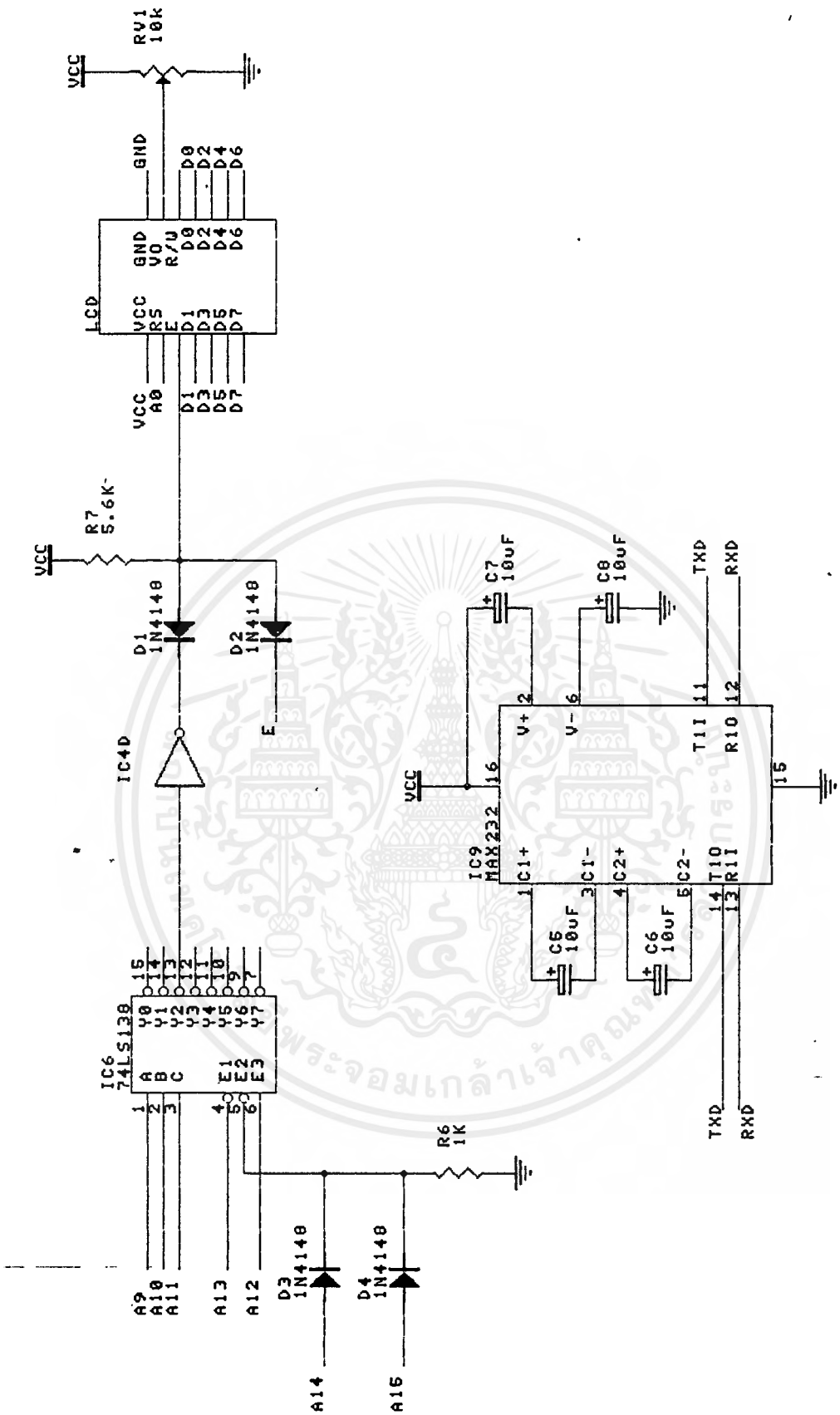
- ไมโครคอนโทรลเลอร์ (Microcontroller) เบอร์ 68HC11A1FN
- หน่วยความจำ (Memory)
  - * หน่วยความจำถาวร (Read Only Memory : ROM)
  - * หน่วยความจำชั่วคราว (Random Access Memory)
- ส่วนสื่อสารอนุกรม
- ส่วนขับจอแสดงผลแบบผลึกเหลว

วงจรส่วนควบคุมนี้แสดงดังรูปที่ 3.5 วงจรส่วนควบคุมนี้จะใช้ไมโครคอนโทรลเลอร์ เบอร์ 68HC11A1FN ซึ่งเป็นไอซี (Integrated Circuit : IC) ที่ผลิตโดยบริษัทโมโตโลรา (MOTOROLA) ภายในไมโครคอนโทรลเลอร์ตัวนี้ ประกอบไปด้วยฮาร์ดแวร์ (Hardware) ดังที่ได้กล่าวไว้ในบทที่ 2 แล้ว ซึ่งเป็ข้อดีโดยทำให้การต่อวงจรใช้งานไม่ยุ่งยากแล้วมีฮาร์ดแวร์เพิ่มเติมเพียงเล็กน้อย

การต่อวงจรส่วนควบคุมนี้จะใช้ไมโครคอนโทรลเลอร์ 68HC11A1FN เป็นหลัก โดยส่วนที่เพิ่มเติมจะประกอบไปด้วย ส่วนของหน่วยความจำ ส่วนสื่อสารอนุกรม ซึ่งการต่อฮาร์ดแวร์นี้จะต้องกำหนดให้ไมโครคอนโทรลเลอร์ทำงานในโหมดมัลติเพล็กซ์ขยาย (Multiplex Expanded) โดยการต่อแรงดัน 5 V. ที่ขา MODA และ MODB



รูปที่ 3.5 วงจรส่วนควบคุมของอุปกรณ์ที่มีการนำไปใช้



รูปที่ 3.5 (ต่อ) วงจรส่วนควบคุม

ไมโครคอนโทรลเลอร์ 68HC11A1FN ประกอบไปด้วยพอร์ต (PORT) 5 พอร์ต โดยพอร์ตทั้ง 5 พอร์ตในวงจรควบคุมมีหน้าที่ต่างๆดังต่อไปนี้

พอร์ต A

พอร์ต A เป็นพอร์ตอินพุท และ พอร์ตเอาท์พุท สำหรับวงจรนี้จะใช้เฉพาะพอร์ตอินพุทคือ PA0 - PA2 เพื่อต่อกับสวิทช์ควบคุม โดยให้ S1 ต่อกับ PA0 , S2 ต่อกับ PA1 และ S2 ต่อกับ PA2 สวิทช์เหล่านี้จะใช้เป็นสวิทช์ควบคุมการทำงานในหน้าที่ต่างๆของโครงงาน

พอร์ต B

พอร์ต B เป็นพอร์ตเอาท์พุท มีหน้าที่การทำงานได้หลายแบบ แต่ในโครงงานนี้ได้ต่อไมโครคอนโทรลเลอร์ในรูปแบบมัลติเพล็กขยาย ดังนั้นขาสัญญาณทั้งหมดของพอร์ต B จะทำหน้าที่เป็นขาแอดเดรส (Address) 8 บิทไบท์สูง

พอร์ต C

ตามโครงสร้างของไมโครคอนโทรลเลอร์ 68HC11A1FN พอร์ต C เป็นพอร์ตอินพุตและเอาท์พุทโดยสามารถส่งข้อมูลได้ 2 ทิศทาง ในวงจรนี้กำหนดให้ไมโครคอนโทรลเลอร์ทำงานในโหมดมัลติเพล็กขยาย ดังนั้นพอร์ต C จึงสามารถเป็นได้ทั้งบัสข้อมูล (Data Bus)และแอดเดรสบัส (Address bus) โดยใช้การมัลติเพล็กในการแยกบัสทั้งสอง ปกติขาสัญญาณของพอร์ต C จะเป็นแอดเดรสบัส A0 ถึง A7 ในแต่ละไซเกิลการทำงานของไมโครคอนโทรลเลอร์ โดยที่ขา E จะมีสถานะเป็น " 0 " ไมโครคอนโทรลเลอร์เปลี่ยนขาสัญญาณนี้เป็นบัสข้อมูล D0 - D7 ที่ขา E จะมีสถานะเป็น "1" โดยทิศทางการเข้าออกของข้อมูลที่พอร์ต C จะแสดงออกมาที่ขาสัญญาณ
R/W

ขาสัญญาณที่พอร์ต C ถูกต่อเข้ากับ IC7 #74L373 ไอซีตัวนี้ถูกควบคุมโดยขา AS ของ IC1 โดยถ้า IC1 ส่งข้อมูลออกมาทางพอร์ต C จะทำให้ AS มีสถานะเป็น " 0 "ทำให้ IC7 อยู่ในสถานะไฮอิมพีแดนซ์ (High impedance) และหาก IC1 ส่งแอดเดรสออกมาจะทำให้ขา AS มีสถานะเป็น " 1 " ทำให้ IC7 เป็นบัฟเฟอร์ (Buffer) และต่อขาแอดเดรสไบท์ต่ำให้แก่ส่วนต่างๆของวงจร

พอร์ต D

พอร์ต D เป็นพอร์ตอินพุทและพอร์ตเอาท์พุท ในโครงงานนี้พอร์ต D จะทำหน้าที่เป็นส่วนสื่อสารอนุกรม โดยใช้ขา PD0 เป็นขารับสัญญาณอนุกรม และขา PD1 เป็นขาส่งสัญญาณอนุกรมขา PD0

และขา PD1 จะต่อกับคอมพิวเตอร์ (Computer) ทางพอร์ตอนุกรม เพื่อใช้ส่งข้อมูลไปแสดงผลทางคอมพิวเตอร์

พอร์ต E

พอร์ต E เป็นพอร์ตอินพุทและเป็นขาอินพุทสำหรับวงจรแปลงสัญญาณอนาล็อก เป็นดิจิตอล (Analog to Digital converter) ใน IC1 มีวงจรนี้ด้วยกัน 8 ช่อง โดยในโครงงานจะใช้เพียง 6 ช่อง

วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล ใน IC1 เป็นวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลแบบซักเซสซีฟแอฟพรีอกซิเมชัน (Successive Approximation) ซึ่งมีความสามารถในการทำงานสูง โดยสามารถสุ่มค่าและเก็บรักษาค่าไว้ได้ ถึงแม้ว่าแรงดันอินพุทจะมีการเปลี่ยนแปลงอย่างรวดเร็วก็ตาม และมีค่าความผิดพลาดต่ำ

วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลนี้จะทำงานร่วมกับขา Vrh และ ขา Vrl โดยขาทั้งสองจะเป็นขาที่ใช้ป้อนแรงดันอ้างอิง เพื่อใช้ในวงจรแปลงสัญญารอนาล็อกเป็นดิจิตอล ค่าความผิดพลาดของวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล มีค่าประมาณ ± 1 LSB ซึ่งประกอบด้วยค่าความผิดพลาดจากการควอนไตซิง (Quantizing) $\pm$ 0.5 LSB และความผิดพลาดจากย่านของแรงดันอ้างอิงคือ ตั้งแต่ระดับ Vrl ถึง Vrh ในโครงงานนี้จะใช้ แรงดันอ้างอิงที่ Vhr = 5 V. และ Vrl = 0 V.ซึ่งเป็นย่านที่มีประสิทธิภาพที่ดีในการทำงาน

ขาสัญญาณที่พอร์ต E ทั้ง 6 ขานี้ต่อเข้ากับส่วนของวงจรเปลี่ยนค่าอุณหภูมิเป็นแรงดัน

3.21 หน่วยความจำ

ในโครงงานนี้จะประกอบไปด้วยหน่วยความจำ 2 แบบด้วยกันคือ

1. หน่วยความจำถาวร (Read only memory : ROM)
2. หน่วยความจำชั่วคราว (Random access memory : RAM)

หน่วยความจำถาวร (ROM)

IC2 คือหน่วยความจำถาวร เบอร์ 27256 มีค่าความจุ 32 กิโลไบท์ (Kilobyte) มีสายสัญญาณข้อมูล 8 เส้น โดย IC2 นี้เป็นหน่วยความจำที่เก็บโปรแกรมหลักหรือโปรแกรมการทำงานของโครงงาน หน่วยความจำนี้อยู่ที่แอดเดรส 8000 - FFFF

หน่วยความจำชั่วคราว (RAM)

IC3 เป็นหน่วยความจำชั่วคราว เบอร์ 6264 มีค่าความจุขนาด 8 กิโลไบท์ มีสายสัญญาณข้อมูล 8 เส้น IC3 ทำหน้าที่เก็บข้อมูลในขณะที่โปรแกรมหลักกำลังทำงาน หน่วยความจำนี้อยู่ที่แอดเดรส 2000 - 3FFF

หน่วยความจำทั้งสองจะถูกติดต่อกับ IC1 ทางพอร์ต B และพอร์ต C ซึ่งเป็นสายสัญญาณแอดเดรส และการกำหนดแอดเดรสคือ IC2 ที่ 8000 - FFFF และ IC3 ที่ 2000 - 3FFF นั้นจะใช้ IC8 เบอร์ 74LS156 เป็นตัวถอดรหัสแอดเดรสตามที่กำหนด

จากวงจรจะเห็นว่าขา A0 และ A1 ของ IC8 ต่อกับสายสัญญาณแอดเดรส A13 และ A14 ตามลำดับ ดังนั้นการถอดรหัสของ IC8 จะเป็นดังตารางที่ 3.2

1C	1G	2C	2G	A1	A0	ผลที่ได้
0	1	1	0	0	0	1Y0
0	1	1	0	0	1	1Y1
0	1	1	0	1	0	1Y2
0	1	1	0	1	1	1Y3
0	0	0	0	0	0	2Y0
0	0	0	0	0	1	2Y1
0	0	0	0	1	0	2Y2
0	0	0	0	1	1	2Y3

ตารางที่ 3.2 การถอดรหัสของ IC8 #74LS156

จากตารางการถอดรหัสแอดเดรสของ IC8 และการต่อกับขา A13 ,A14 , A15 และ E โดยที่ขา 1Y0 - 1Y3 ต่อถึงกันทั้ง 4 ขา แล้วไปต่อกับขา CE (Chip Enable) ของ IC2 ทำให้IC2 มีแอดเดรสอยู่ที่ 8000 - FFFF และที่ขา 2Y1 ของ IC8 ต่อกับขา CS1 (Chip select) ของ IC3 ดังนั้น IC3 จึงอยู่ที่แอดเดรส 2000 - 3FFF

3.22 ตัวแสดงผลผลึกเหลว (LCD)

การควบคุมตัวแสดงผลผลึกเหลว ใช้ IC6 เป็นตัวถอดรหัสแอดเดรส โดยตั้งแอดเดรสไว้ที่

1400 สำหรับการเขียนข้อมูลควบคุมจอแสดงผลผลึกเหลว

1401 สำหรับการเขียนข้อมูลแสดงผลบนจอแสดงผลผลึกเหลว

โดย IC6 ที่ใช้ในการถอดรหัสแอดเดรสของจอแสดงผลผลึกเหลว ใช้เบอร์ 78LS138 การถอดรหัสให้ได้แอดเดรสตามที่ต้องการดังตารางที่ 3.3

E1	E2	E3	C	B	A	ผลที่ได้
0	0	1	0	0	0	Y0
0	0	1	0	0	1	Y1
0	0	1	0	1	0	Y2
0	0	1	0	1	1	Y3
0	0	1	1	0	0	Y4
0	0	1	1	0	1	Y5
0	0	1	1	1	0	Y6
0	0	1	1	1	1	Y7

ตารางที่ 3.3 การถอดรหัสแอดเดรสของ IC6 #74LS138

จากตารางการทำงาน และการต่อขาอินาเบิล (Enable) ของจอแสดงผลผลึกเหลว ที่ขา Y2 ทำให้แอดเดรสของจอแสดงผลผลึกอยู่ที่ 1400 และ 1401

3.23 ส่วนติดต่อสื่อสารกับคอมพิวเตอร์

ในไมโครคอนโทรลเลอร์ เบอร์ 68HC11A1FN มีโครงสร้างที่ใช้ในการติดต่อสื่อสารแบบอนุกรม ซึ่งเป็นข้อดีข้อหนึ่งที่ทำให้ต้องเลือกใช้ไมโครคอนโทรลเลอร์ในโครงงานนี้ ไมโครคอนโทรลเลอร์ตัวนี้ทำการจัดรูปแบบของส่วนสื่อสารอนุกรมในลักษณะฟลูดูเพล็กซ์ (Full duplex) แบบอะซิงโครนัส (Asyncronusce) ตามมาตรฐาน NRZ คือมีบิตเริ่มต้น 1 บิต บิตข้อมูล 8 บิตหรือ 9 บิต (ในโครงงานนี้ใช้บิตข้อมูล 8 บิต) และ บิตหยุด 1 บิต

การควบคุมส่วนสื่อสารอนุกรม

การควบคุมสื่อสารอนุกรมของไมโครคอนโทรลเลอร์ 68HC11A1FN สามารถกระทำได้โดยการเขียนข้อมูลให้กับรีจิสเตอร์สื่อสารอนุกรม ซึ่งมีด้วยกันดังนี้

1. รีจิสเตอร์ควบคุมการสื่อสารอนุกรม (Serial communication control register 1 : SCCR1)

รีจิสเตอร์นี้อยู่ที่แอดเดรส $102C โดยเขียนข้อมูล 00 ซึ่งจะกำหนดให้ ความยาวข้อบิตข้อมูลเป็น 8 บิต 1 บิตเริ่มต้น และ 1 บิตหยุด

2. รีจิสเตอร์ควบคุมการสื่อสารอนุกรม 2 (Serial communication control register 2 : SCCR2) รีจิสเตอร์นี้อยู่ที่แอดเดรส $102D โดยเขียนข้อมูล 0C ซึ่งจะกำหนดให้ อินาเบิลการทำงานของตัวส่งและตัวรับ

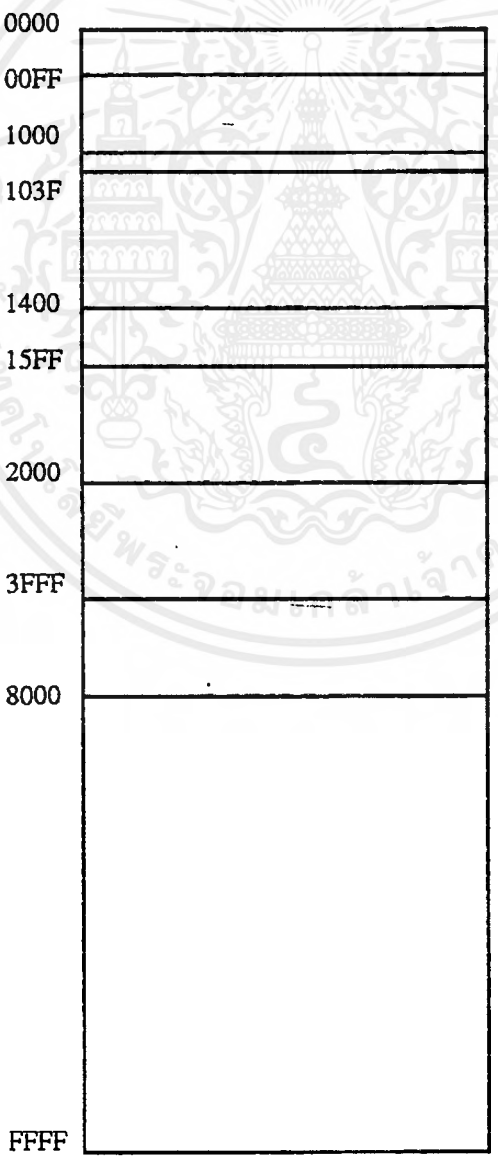
3. รีจิสเตอร์อัตราบอด (Baud rate register : BAUD)

รีจีสเตอร์อยู่ที่แอดเดรส $102B ใช้ในการเลือกอัตราบอด โดยเขียนข้อมูล 30 ซึ่งเป็นการกำหนดอัตราบอดเท่ากับ 9600 บอด

เมื่อทำการตั้งค่าต่างๆให้แก่ส่วนสื่อสารอนุกรมในไมโครคอนโทรลเลอร์แล้ว ทำให้ส่วนสื่อสารพร้อมที่จะทำงานโดยจะส่งข้อมูลออกทางขา PD1 และรับข้อมูลเข้าทางขา PD0 ข้อมูลที่รับเข้าและส่งออกจะให้ผ่านวงจรขยายและไดร์เวอร์ (Driver) อีกที IC9 เบอร์ MAX232 จะทำหน้าที่นี้

การจัดหน่วยความจำ

โครงงานนี้มีการจัดสรรหน่วยความจำดังแสดงในรูปที่ 3.6 โดยจัดแบ่งเป็นส่วนดังนี้



รูปที่ 3.6 แผนที่หน่วยความจำในระบบโครงงาน

0000 - 00FF เป็น STACK MEMORY

1000 - 103F เป็น รีจิสเตอร์ควบคุม 64 ไบท์

1400 - 15FF เป็น แอดเดรสของจอแสดงผลผลึกเหลว

2000 - 3FFF เป็น หน่วยความจำชั่วคราว

A000 - FFFF เป็น หน่วยความจำถาวร

บทที่ 4
การออกแบบซอฟท์แวร์

โครงงานเครื่องวัดอุณหภูมิ 6 ช่อง มีส่วนประกอบที่เป็นซอฟแวร์ด้วยโดยซอฟร์แวร์ของโครงงานนี้ประกอบไปด้วยซอฟร์แวร์ด้วยกัน 2 ส่วนดังนี้

1. ซอฟร์แวร์ที่ติดตั้งในระบบฮาร์ดแวร์
2. ซอฟร์แวร์ที่ติดตั้งบนเครื่องไมโครคอมพิวเตอร์

ซอฟร์แวร์ทั้งสองมีข้อแตกต่างกันตรงที่หน้าที่การใช้งาน และภาษาที่ใช้ออกแบบโดยซอฟร์แวร์ทั้งสองส่วนมีรายละเอียดดังนี้

4.1 ซอฟร์แวร์ที่ติดตั้งในระบบฮาร์ดแวร์

ซอฟร์ส่วนนี้ได้ติดตั้งบนฮาร์ดแวร์ของโครงงาน ซึ่งถูกออกแบบมาจากภาษาแอสเซมบลี้สำหรับหน้าที่ของซอฟร์แวร์ส่วนนี้ มีดังนี้

- เซ็ทค่าต่างๆให้แก่ฮาร์ดแวร์เมื่อเริ่มใช้งานโครงงาน
- รับค่าอุณหภูมิช่องต่างๆ ทั้ง 6 ช่อง
- แสดงผลค่าอุณหภูมิ ณ ช่องต่างๆ ทั้ง 6 ช่อง
- ส่งข้อมูลค่าอุณหภูมิทั้ง 6 ช่องไปยังเครื่องไมโครคอมพิวเตอร์

หน้าที่ของซอฟร์แวร์ส่วนนี้ดังที่กล่าวมาดังนั้นจึงต้องออกแบบให้ซอฟร์แวร์ทำงานหน้าที่ต่างดังที่กล่าวมาแล้ว

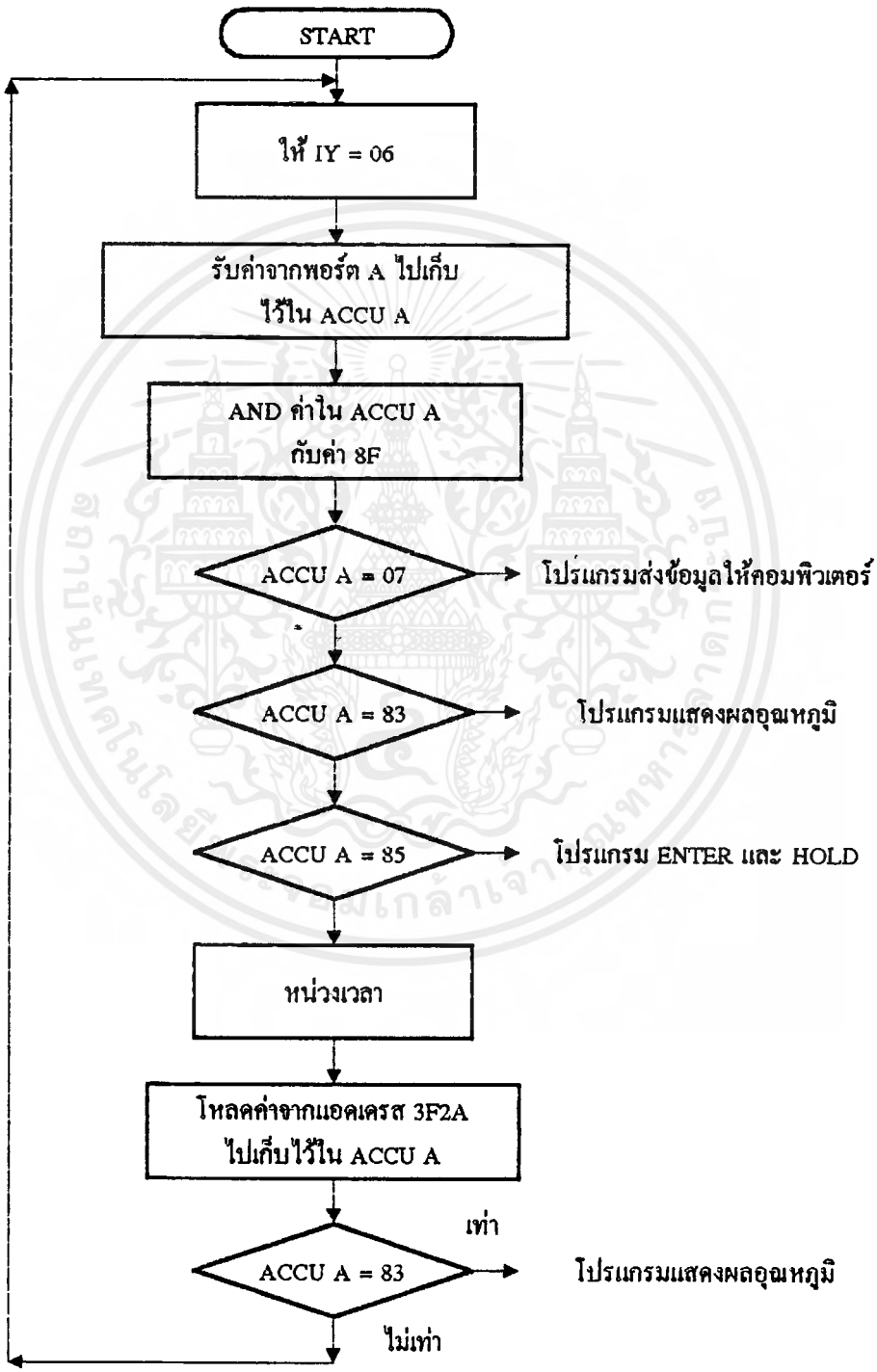
การทำงานของซอฟร์แวร์เริ่มเมื่อเปิดเครื่องใช้งาน ให้กดรีเซ็ทเครื่องแล้วเครื่องจะทำการเซ็ทค่าต่างๆให้แก่รีจิสเตอร์ควบคุมภานใน IC1 ดังนี้

1. เซ็ทอัตราบอด (Baud rate) ด้วยข้อมูล 30
2. เซ็ทออปชั่นรีจิสเตอร์ (Option register) ด้วยข้อมูล 93
3. เซ็ท TMSK2 register ด้วยข้อมูล 00
4. เซ็ท SCCR1 ด้วยข้อมูล 00
5. เซ็ท SCCR2 ด้วยข้อมูล 0C

ต่อจากนั้นเครื่องจะเริ่มทำงานในส่วนของโปรแกรมเครื่องวัดอุณหภูมิ โดยเริ่มพิมพ์ข้อความบนหน้าจอผลึกเหลวดังนี้

6 CHANNELS THERMOMETER

เมื่อทำงานถึงส่วนนี้ก็จะเข้าสู่ โปรแกรมการรับคีย์สวิทช์ชุดที่ 1 ซึ่งคีย์สวิทช์ชุดที่ 1 มีสวิทช์อยู่ด้วยกัน 3 ตัว โปรแกรมส่วนนี้มีโฟลว์ชาร์จ (Flowchart) ดังรูปที่ 4.1



รูปที่ 4.1 แสดงโฟลว์ชาร์จ โปรแกรมรับคีย์ชุดที่ 1

การทำงานของโปรแกรมรับคีย์ชุดที่ 1 ของสวิทช์ต่างๆ มีดังนี้

1.Enter and Hold

สวิทช์ตัวนี้ต่อเข้ากับพอร์ต A ที่ขา PA1 ซึ่งโดยปกติที่พอร์ต A มีค่า 87 ดังนั้นเมื่อกดคีย์สวิทช์ตัวนี้ จะทำให้พอร์ต A มีค่า 85 โดยค่าที่รับได้นี้จะถูกเก็บไว้ในแอคคิวมูเลเตอร์ A แล้วนำไปเปรียบเทียบ โดยถ้าค่าที่รับได้จากพอร์ต A มีค่าไม่เท่ากับ 85 ก็จะผ่านไป แต่ถ้าค่าที่รับได้มีค่าท่ากับ 85 ก็จะไปทำงานที่โปรแกรมแสดงผลข้อความ 6 CHANNEL THERMOMETER บนจอแสดงผลผลึกเหลว

2.DISPLAY

ฟังก์ชันสวิทช์ตัวนี้ ทำหน้าที่แสดงผลค่าอุณหภูมิแต่ละช่อง โดยการแสดงจะแสดงผลทีละช่องเรียงกันไป จากช่องที่ 1,2,3,4,5 และ 6 แล้ววนกลับมาที่ช่อง 1 การเปลี่ยนการแสดงผลจะกดที่สวิทช์ S1 เมื่อกด 1 ครั้งก็จะเปลี่ยนการแสดงเป็นช่องถัดไป

สวิทช์ตัวนี้ต่อเข้ากับพอร์ต A ที่ขา PA2 ดังนั้นเมื่อกดสวิทช์ตัวนี้ จะทำให้พอร์ต A มีค่า 83 ค่า 83 ที่รับได้ที่พอร์ต A จะถูกนำไปเก็บในแอคคิวมูเลเตอร์ A แล้วนำค่าไปเปรียบเทียบเพื่อกระโดดไปทำงานโปรแกรมแสดงผลอุณหภูมิ

3.SAND

หน้าที่ของสวิทช์ตัวนี้คือ ส่งข้อมูลไปให้ยังไมโครคอมพิวเตอร์เพื่อให้ไมโครคอมพิวเตอร์แสดงผล โดยสวิทช์ตัวนี้ต่อที่พอร์ต A ขา PA7 ดังนั้นเมื่อกดสวิทช์ตัวนี้จะทำให้พอร์ต A มีค่า 07

เมื่อกดสวิทช์ตัวนี้โปรแกรมจะกระโดดไปทำงานที่ โปรแกรมย่อยส่งข้อมูลไปให้ยังไมโครคอมพิวเตอร์

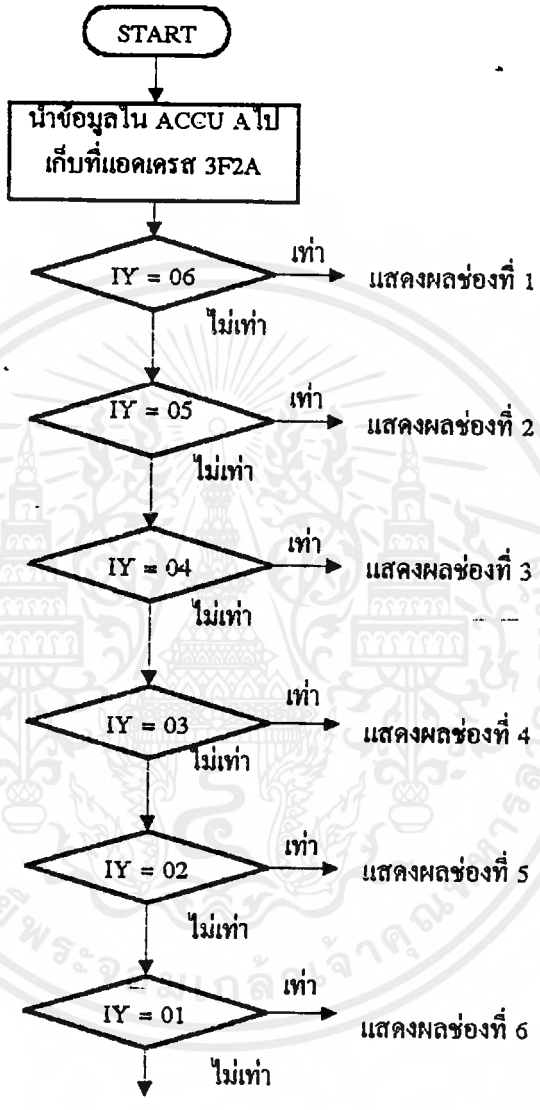
4.1.1 โปรแกรมย่อยแสดงผลอุณหภูมิ

โฟลว์ชาร์จของโปแกรมส่วนนี้แสดงดังรูปที่ 4.2 การทำงานของโปรแกรมเริ่มต้นดังนี้คือ

จากเมนเมนโปรแกรม ค่า 06 ที่เก็บอยู่ใน IY จะถูกนำมาเปรียบเทียบโดยหากเป็นการกด S1 ครั้งแรกค่าใน IY ยังคงเป็น 06 ดังนั้นเมื่อนำมาเปรียบเทียบโปรแกรมก็จะกระโดดไปทำงานที่โปรแกรมแสดงผลช่องที่ 1 ที่โปรแกรมแสดงผลช่องที่ 1 จะมีคำสั่งลดค่า IY ลง 1 ค่าดังนั้นเมื่อทำงานโปรแกรมแสดงผลช่องที่ 1 แล้ว IY จะเท่ากับ 05 เมื่อกลับมาที่โปรแกรมรับคีย์ชุดที่ 1 แล้วมีการกด S1 อีก การแสดงผลก็จะเป็นการแสดงผลช่องที่ 2 โดยการแสดงผลในช่องถัดไปก็เป็นเช่นเดียวกัน

- โปรแกรมแสดงผลช่องที่ 1 ถึง 4

โปรแกรมส่วนนี้จะเป็นการแสดงผลค่าอุณหภูมิในช่องที่ 1 ถึง 4 โดยการแสดงจะแสดงผลทีละช่อง แต่โปรแกรมเหมือนกันจึงสามารถอธิบายรวมกันได้



กลับไปยังจุดเริ่มต้นของโปรแกรมรับคีย์ชุดที่1

รูปที่ 4.2 โฟลว์ชาร์จการแสดงผลอุณหภูมิ

ในโปรแกรมแสดงผลช่องที่ 1 จะมีการเซ็ทค่าวงจรเปลี่ยนสัญญาณอนาล็อกเป็นดิจิตอล ให้แสดงผลแบบ 4 ช่องแรกตามโปรแกรมดังนี้

```
LDAA    #$30
STAA    $1030
```

โดยการเขียนข้อมูล 30 ให้แก่ รีจิสเตอร์ควบคุมและแสดงสถานะของวงจรเปลี่ยนสัญญาณอนาล็อกเป็นดิจิตอล ซึ่งอยู่ที่แอดเดรส 1030 มีความหมายดังนี้ จะเป็นการทำงานแบบต่อเนื่องไม่สิ้นสุด ข้อมูลที่ได้ใหม่ก็จะถูกเขียนทับลงไปตลอดเวลา โดยจะเป็นการแบบหลายช่อง ทำงานเป็นกลุ่มครั้งละ 4 ช่องและรีจิสเตอร์ ADC 1 ตัวจะเก็บข้อมูล 1 ช่อง

หลังจากเซ็ทค่าต่างๆแล้ว โปรแกรมจะรับค่าอุณหภูมิจากพอร์ต E ที่ขา PE1 เป็นค่าอุณหภูมิช่องที่ 1 ซึ่งเป็นสัญญาณอนาล็อก เข้ามาสู่วงจรเปลี่ยนสัญญาณอนาล็อกเป็นดิจิตอล เปลี่ยนแปลงให้เป็นสัญญาณดิจิตอล แล้วเก็บไว้ในแอคคิวมูเลเตอร์ B เมื่อนำค่ามาเก็บไว้ในแอคคิวมูเลเตอร์แล้ว ก็จะนำไปเปรียบเทียบเพื่อแปลงเป็นค่าเลขฐานสิบ แล้วนำไปแสดงผลบนจอผลึกเหลว

- โปรแกรมเปรียบเทียบเลขฐานสิบหกเป็นฐานสิบ

โปรแกรมส่วนนี้มีโฟลว์ชาร์จแสดงดังรูปที่ 4.4 โดยโปรแกรมส่วนนี้จำทำงานเมื่อ โปรแกรมได้รับข้อมูลจากพอร์ต E แล้วนำมาเปรียบเทียบเพื่อเปลี่ยนข้อมูลเลขฐานสิบหกให้เป็นข้อมูลที่สามารถแสดงผลบนจอผลึกเหลวได้

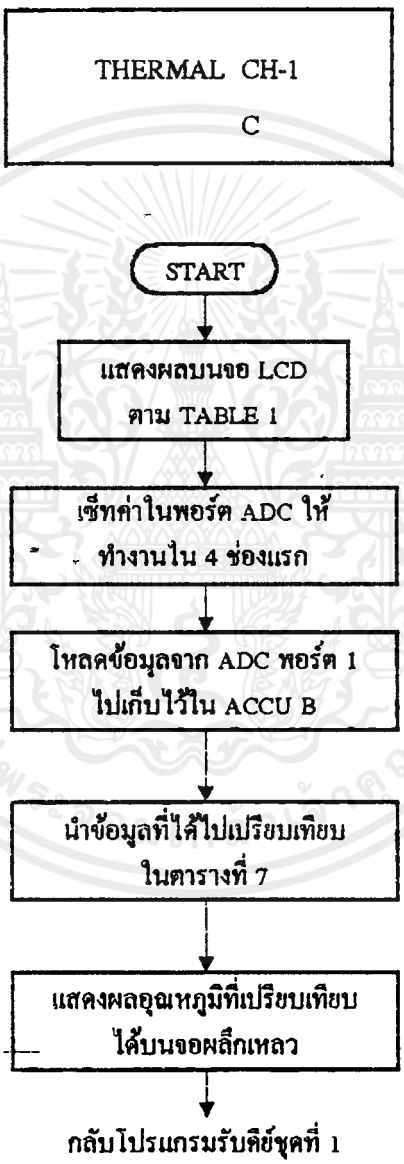
การทำงานเริ่มโดย ข้อมูลค่าอุณหภูมิที่รับได้จากพอร์ต E แล้วถูกนำมาเปลี่ยนให้เป็นเลขฐานสิบหกในวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล แล้วนำมาเก็บไว้ในแอคคิวมูเลเตอร์ B แล้วจึงนำมาเปรียบเทียบที่โปรแกรมส่วนนี้

เริ่มโปรแกรมส่วนนี้ โดยโปรแกรมจะโหลดตำแหน่งเริ่มต้นของตารางการเปรียบเทียบไปไว้ใน IX โดยรูปแบบของตารางเปรียบเทียบมีดังนี้

เลขฐานสิบหก	ข้อมูลแสดงที่ LCD หลักสิบ	ข้อมูลแสดงที่ LCD หลักหน่วย

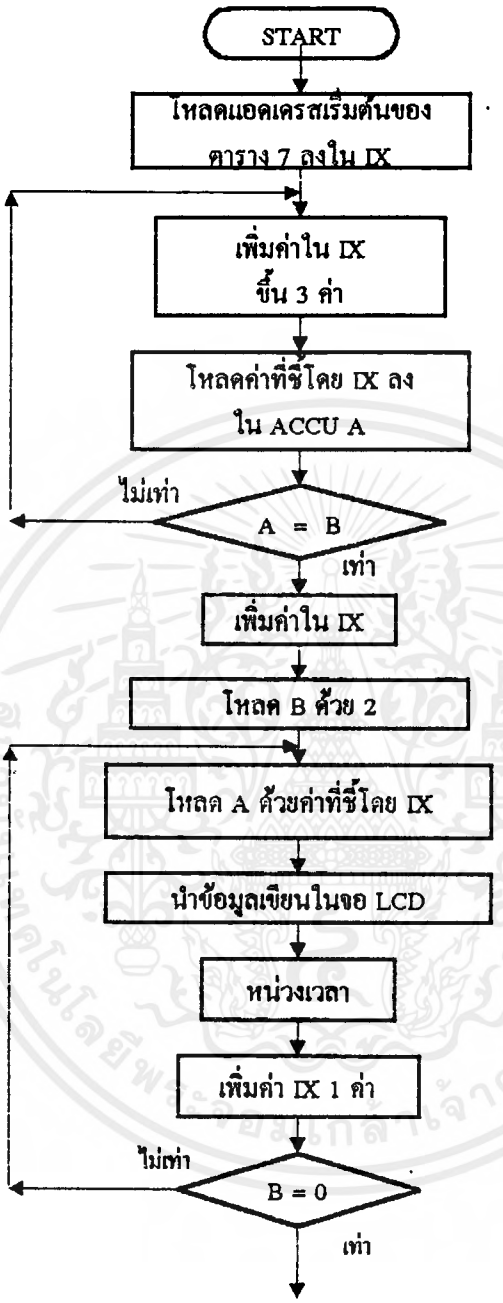
ดังนั้นเมื่อเปรียบเทียบข้อมูลในแอคคิวมูเลเตอร์ B แล้วไม่ตรงกันโปรแกรมก็จะเพิ่มค่าใน IX ขึ้นมาสามค่า แล้วโหลดข้อมูลที่ชี้โดย IX ไปไว้ในแอคคิวมูเลเตอร์ A แล้วจึงนำมาเปรียบเทียบกับค่าในแอคคิวมูเลเตอร์ B ถ้าไม่ใช่ก็จะกลับไปโหลดข้อมูลในตารางมาเปรียบเทียบใหม่ แต่ถ้าใช่ก็จะเพิ่มค่าใน IX หนึ่งค่าเพื่อนำข้อมูลที่ใช้แสดงผลบนจอผลึกเหลวออกมาเพื่อส่งไปแสดงผลบนจอผลึกเหลว ตัวอย่างเช่นถ้าค่าอุณหภูมิที่รับได้ที่พอร์ต E แล้วแปลงค่าเป็นเลขฐานสิบหกได้เป็น 20 การเปรียบเทียบจะเริ่มนำค่า 20 นี้ มาเปรียบเทียบกับข้อมูลในตารางเริ่มต้น ซึ่งก็คือ 00 ซึ่งเมื่อเปรียบเทียบแล้วไม่เท่ากันก็จะเพิ่ม

โดยโฟลว์ชาร์จของโปแกรมส่วนนี้แสดงดังรูปที่ 4.3 การแสดงเริ่มจากการแสดงผลช่องที่ 1 เมื่อมีการกด S1 ครั้งแรก โดย IY ยังมีค่าเท่ากับ 06 และโปรแกรมจะมาทำงานที่ส่วนนี้ เริ่มโดยจะลดค่าใน IY ลง 1 ค่า แล้วเขียนหน้าจอผลึกเหลวดังนี้



ช่องที่ 2,3 และ 4 ทำเช่นเดียวกัน
ช่องที่ 5 และ 6 ให้เซ็ท พอร์ต ADC ให้ทำงาน 4 ช่องหลัง

รูปที่ 4.3 โฟลว์ชาร์จโปรแกรมแสดงผลค่าอุณหภูมิ



รูปที่ 4.4 โฟลว์ชาร์จโปรแกรมเปรียบเทียบค่าอุณหภูมิ

ค่าใน IX 3 ค่า ซึ่งก็จะได้ค่า 01 มาเปรียบเทียบ ซึ่งถ้าไม่ใช่ก็จะเพิ่มค่าใน IX ขึ้นเรื่อยๆ จนกว่าจะได้ข้อมูล 20 มาเปรียบเทียบ โดยเมื่อค่าที่เปรียบเทียบมีค่าเท่ากันก็จะเพิ่มค่าไน IX หนึ่งค่าแล้วนำข้อมูลนี้ไปเขียนในจอผลึกเหลว และเมื่อแสดงผลบนจอผลึกเหลวแล้วก็จะกลับไปสู่โปรแกรมรับคีย์ชุดที่ 1

เมื่อแสดงผลอุณหภูมิเสร็จแล้ว จะกลับไปโ)รแกรมรับคีย์ชุดที่ 1 แล้วจะคอยรับคีย์ โดยถ้าไม่มีการกดคีย์ใดๆ โปรแกรมจะผ่านแล้วไปดึงข้อมูลที่อยู่ในแอดเดรส 3F2A ซึ่งเก็บค่า 83 เอาไว้ ดังนั้นโปรแกรมจะมาทำงานที่โปรแกรมแสดงผลค่าอุณหภูมิแล้วจะทำงานเช่นนี้เรื่อยๆไปจนกว่าจะกดสวิทช์ใดๆ

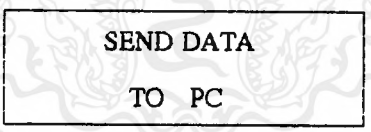
นอกจากนั้นแล้ว ถ้าการแสดงผลค่าอุณหภูมิบนจอเรียบร้อยแล้วกลับที่โปรแกรมรับคีย์ชุดที่ 1 แล้วมีการกด S1 อีกครั้งก็จะเป็นการแสดงผลในช่องถัดไป

4.1.2 โปรแกรมส่งข้อมูลไปยังไมโครคอมพิวเตอร์

โปรแกรมส่วนนี้เริ่มที่ โปรแกรมรับคีย์ชุดที่ 1 แล้วมีการกดสวิทช์ S3 ซึ่งทำให้พอร์ต A มีค่า 07 โดยค่า 07 นี้จะถูกนำมาเก็บในแอดคิวมูเลเตอร์ A โดยค่า 07ในแอดคิวมูเลเตอร์ A จะนำมาเปรียบเทียบในโปรแกรมรับคีย์ชุดที่ 1 ซึ่งถ้าเท่ากับ 07 ก็จะไปทำงานที่โปรแกรมส่งข้อมูลไปยังไมโครคอมพิวเตอร์

โดยโฟลว์ชาร์จของโปรแกรมส่วนนี้แสดงดังรูปที่ 4.5

การเริ่มต้นทำงานเริ่มที่ การแสดงผลบนจอ LCD ดังนี้



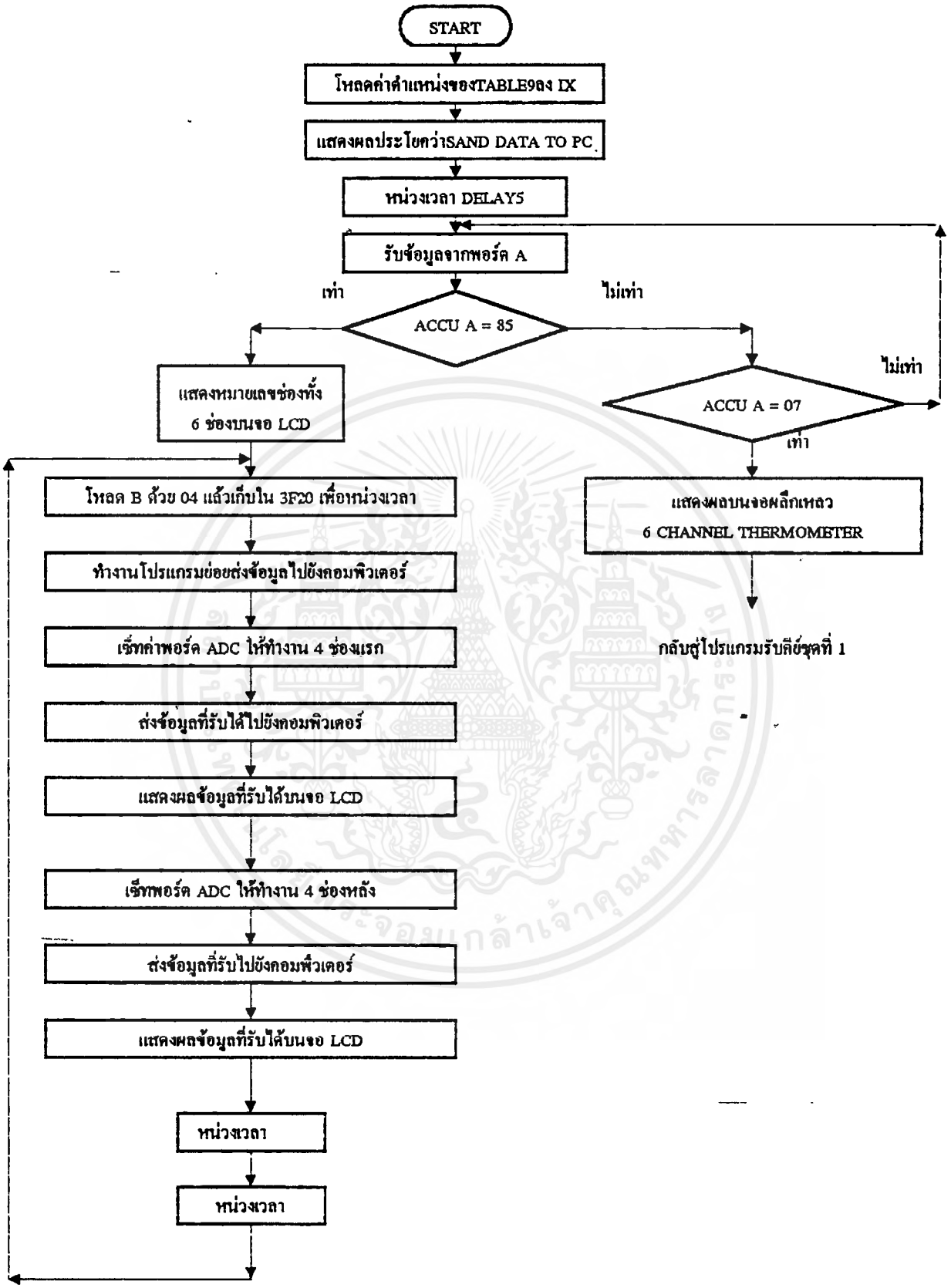
หลังจากนั้นในโปรแกรมส่วนนี้จะรับคีย์สวิทช์ 2 ตัวคือ S1 และ S2 โดย

S1 ทำหน้าที่ เป็นสวิทช์กดเริ่มต้นการทำงานของการส่งข้อมูลไปให้ไมโครคอมพิวเตอร์

S2 ทำหน้าที่ เป็นสวิทช์ยกการส่งข้อมูลและกลับไปสู่โปรแกรมรับคีย์ชุดที่ 1

โปรแกรมที่ใช้ส่งข้อมูลทั้งหกช่อง จะมีโปรแกรมแต่ละช่องเหมือนกัน ดังนั้นจึงสามารถอธิบายได้จากช่องเดียว ดังนี้

```
LDAB    ATC_DAT1
LDAA    $102E
LDAA    $102F
STAB    $102F
```



รูปที่ 4.5 โฟลว์ชาร์จโปรแกรมส่งข้อมูลไปให้ไมโครคอมพิวเตอร์

โดยอธิบายได้ว่า เริ่มต้นให้นำข้อมูลที่ต้องการจะส่งไปเก็บไว้ในแอคคิวมูเลเตอร์ B ต่อมาให้อ่านค่าจากรีจิสเตอร์ SCSR และรีจิสเตอร์ SCDR ซึ่งจะเป็นการทำให้บิต TDRF ในรีจิสเตอร์ SCSR เคลียร์ นั่นคือโปรแกรมพร้อมที่จะส่งข้อมูลไปให้ยังไมโครคอมพิวเตอร์โดยจะส่งเมื่อนำข้อมูลจากแอคคิวมูลเลเตอร์ B โหลดให้แก่รีจิสเตอร์เก็บข้อมูลส่ง เพื่อส่งข้อมูลไปยังไมโครคอมพิวเตอร์แล้วก็จะทำงานแสดงผลบนจอผลึกเหลวด้วยโดยการแสดงจะใช้วิธีการเช่นเดียวกับการแสดงผลค่าอุณหภูมิตามที่กล่าวมาแล้ว

4.1.3 โปรแกรม Enter และ Hold

ในโปรแกรมรับคีย์ชุดที่ 1จะมีสวิทช์ S2 ซึ่งสวิทช์ตัวนี้ได้จัดให้ใช้งานร่วมกับโปรแกรมอื่นๆด้วย สำหรับสวิทช์ตัวนี้จะต่ออยู่ที่พอร์ต A ที่ขา PA2 ดังนั้นเมื่อกดสวิทช์ตัวนี้ก็จะทำให้พอร์ต A มีค่า 85

ในโปรแกรมรับคีย์ชุดที่ 1 ถ้ามีการกด S2 ก็จะทำให้โปรแกรมไปทำงานที่ส่วนแสดงผลข้อความ 6 CHANNEL THERMOMETER บนหน้าจอผลึกเหลว

เมื่ออยู่ในการทำงานของการแสดงผลค่าอุณหภูมิ สวิทช์ตัวนี้จะทำหน้าที่โฮล (Hold)ค่าอุณหภูมิที่ แสดงบนหน้าจอผลึกเหลว

เมื่อทำงานในส่วนส่งข้อมูลให้แก่ไมโครคอมพิวเตอร์ สวิทช์ตัวนี้จะทำหน้า เริ่มส่งข้อมูลให้แก่ไมโครคอมพิวเตอร์

4.1.4 โปรแกรมเซ็ทจอแสดงผลผลึกเหลว

โปรแกรมส่วนนี้จะเป็นการเซ็ทค่าเริ่มต้นการทำงานให้แก่จอผลึกเหลว โดยจะส่งข้อมูลไปให้แก่จอผลึกเหลวตามแแอดเดรส 1400 ซึ่งข้อมูลที่ส่งไปมีดังนี้

0F กำหนดให้ - เคอร์เซอร์อยู่ส่วนล่าง
- ให้เปิดหน้าจอผลึกเหลว
- มีการกระพริบของเคอร์เซอร์

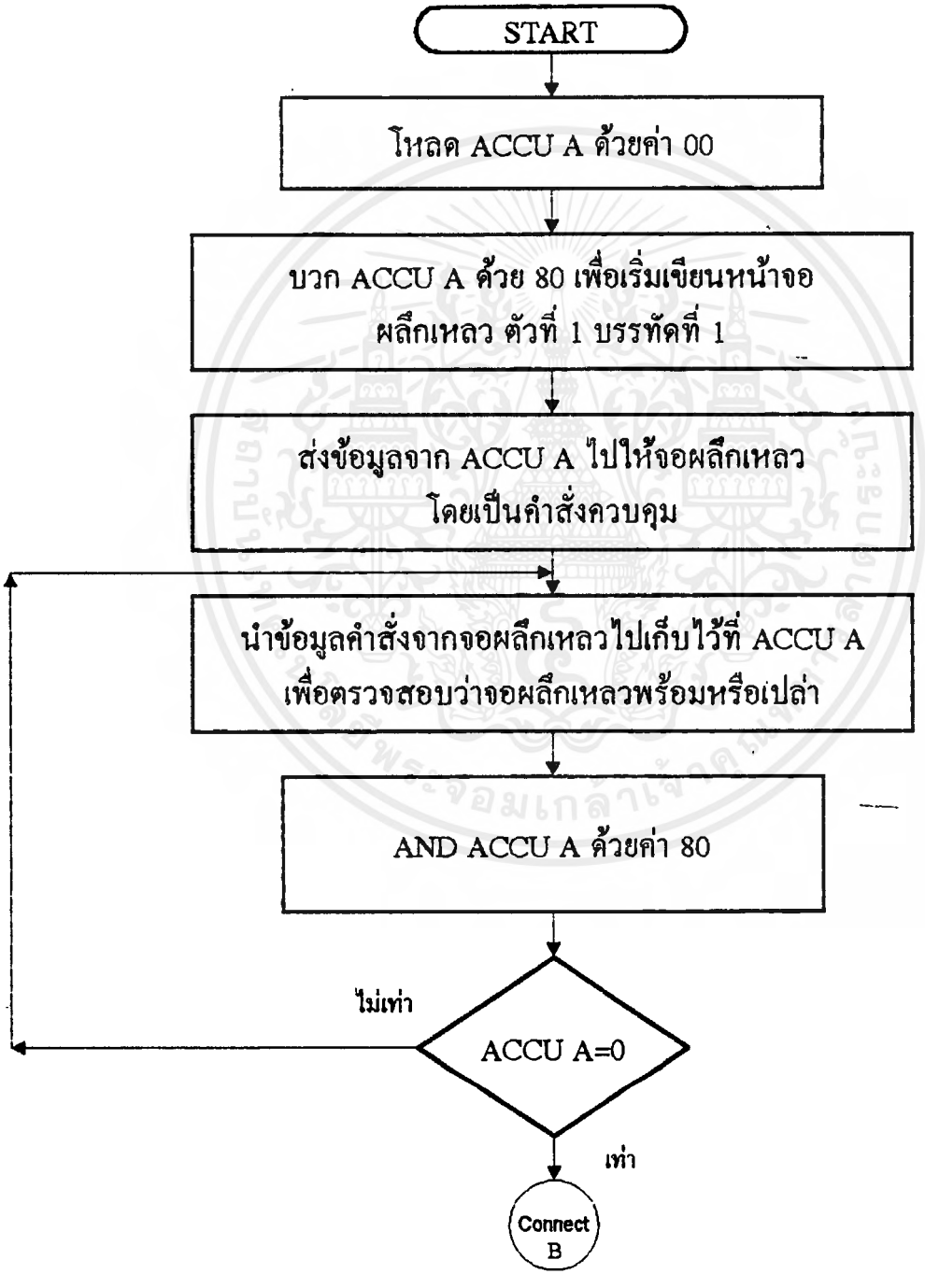
38 กำหนดให้ - จำนวนบิตที่ใช้ในการติดต่อเป็นแบบ 8 บิต
- กำหนดให้แสดงแบบ 2 บรรทัด
- แสดงผลแบบ 5 x 10 จุด

06 กำหนดให้ - แอดเดรสของ DDRAM เพิ่มขึ้นเมื่อมีการเขียนข้อมูลให้ DDRAM
- ถ้ามีการเขียนข้อมูลให้แก่จอผลึกเหลวให้เคอร์เซอร์เลื่อนไปทางขวามือ

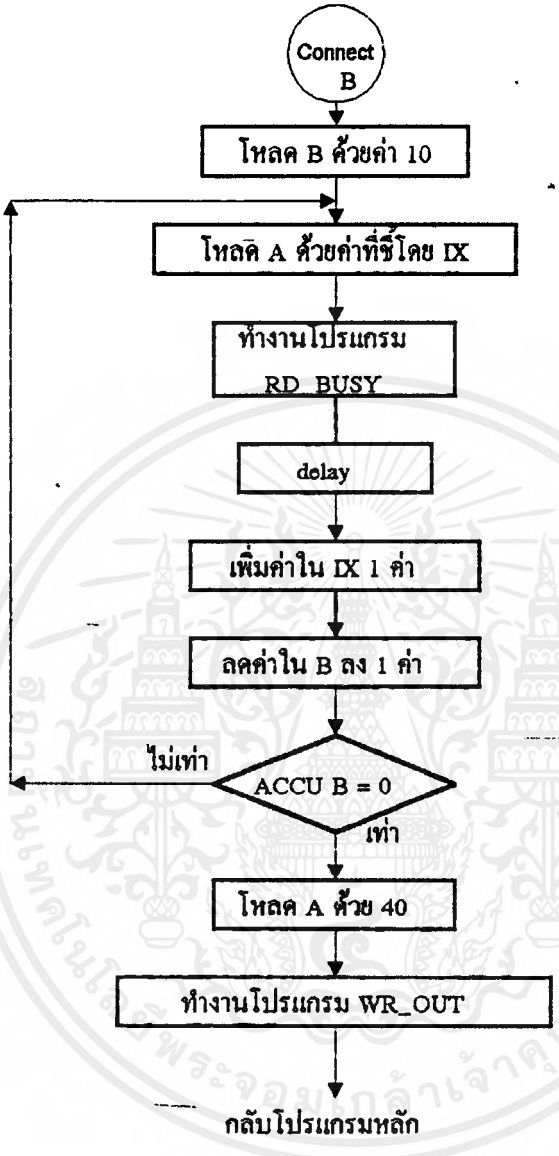
การเขียนข้อมูลส่งให้แก่จอแสดงผลผลึกเหลวแต่ละค่าตามที่กล่าวมาแล้ว จะต้องส่งข้อมูลเหล่านี้ให้แก่จอผลึกเหลวทีละค่า โดยเมื่อส่งข้อมูลไปให้ 1 ค่าแล้วจะต้องมีการหน่วงเวลาเพื่อให้จอผลึกเหลวรับค่าต่างได้ทัน

4.1.5 โปรแกรมแสดงผลบนจอผลึกเหลว

การแสดงผลบนจอผลึกเหลวจะมีการแสดงผลแบบ 2 บรรทัด บรรทัดละ 16 ตัวอักษร ดังนั้นการแสดงจะแสดงผลบรรทัดที่ 1 ก่อนแล้วจึงแสดงผลในบรรทัดที่ 2 ตามมา โดยโฟลว์ชาร์จของโปรแกรมส่วนนี้แสดงดังรูปที่ 4.6



รูปที่ 4.6 โฟลว์ชาร์จการเขียนข้อมูลบนหน้าจอผลึกเหลว



รูปที่ 4.6 (ต่อ) โฟล์ชาร์จการเขียนข้อมูลบนหน้าจอผลึกเหลว

การทำงานเริ่มด้วยการเขียนข้อมูลที่บรรทัดที่ 1 ก่อนโดยการกำหนดแอดเดรสของ DDRAM ให้อยู่ที่คอลัมท์แรกของบรรทัดที่ 1 โดยการส่งข้อมูล 80 ไปยังแอดเดรส 1400 ซึ่งเคอร์เซอร์ของจอผลึกเหลวจะมาอยู่ที่ คอลัมท์ที่ 1 บรรทัดที่ 1

ต่อมาก็จะนำข้อมูลที่จะเขียนที่จะแสดง ส่งไปให้ยังจอผลึกเหลวที่แอดเดรส 1401 จนครบทั้ง 16 ตัวอักษร ก็จะเริ่มต้นเขียนข้อมูลที่บรรทัดที่ 2 ใหม่ โดยการกำหนดแอดเดรสของ DDRAM โดยการส่งข้อมูล C0 ไปยังแอดเดรส 1400 ซึ่งจะทำให้เคอร์เซอร์มาอยู่ที่คอลัมท์ที่ 1 บรรทัดที่ 2 และจะเริ่มต้นเขียนบรรทัดที่ 2 ต่อไป

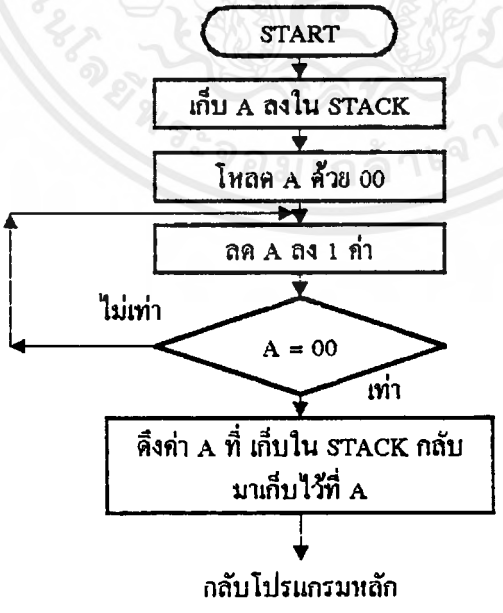
4.1.6 โปรแกรมหน่วงเวลา

โปรแกรมหน่วงเวลา เป็นโปรแกรมที่ใช้สำหรับหน่วงเวลาเพื่อให้การทำงานเป็นไปตามเวลาที่กำหนด และยังทำหน้าที่เพื่อให้การทำงานของส่วนต่างๆ ทำงานได้พร้อมกันและทำงานได้ทันกัน

สำหรับโปรแกรมหน่วงเวลาที่ใช้ในโครงงานนี้มีด้วยกัน 4 โปรแกรม โดยจะอธิบายได้ดังนี้

- DELAY

โปรแกรมหน่วงเวลาโปรแกรมนี้ สามารถหน่วงเวลาได้ในระยะเวลาสั้นๆ โดยมีโฟลว์ชาร์จแสดงดังรูปที่ 4.7



รูปที่ 4.7 โฟลว์ชาร์จโปรแกรมหน่วงเวลา DELAY

การทำงานเริ่มด้วยการโหลดค่า 00 ให้แก่ แอคคิวมูเลเตอร์ A แล้วให้ลดค่าในแอคคิวมูเลเตอร์ A ลง 1 ค่า แล้วให้ทำการเปรียบเทียบ โดยถ้าค่าในแอคคิวมูเลเตอร์ A ยังไม่เป็น 00 ก็ให้ลดค่าในแอคคิวมูเลเตอร์ A ต่อและวนการทำงานไปเรื่อยๆจนค่าในแอคคิวมูเลเตอร์ A จะเป็นค่า 00 จึงจะกลับไปยังโปรแกรมหลัก สำหรับคาบเวลาของโปรแกรมส่วนนี้แสดงดังตารางที่ 4.1

	จำนวนครั้ง	จำนวน Cycle	Cycle ทั้งหมด	เวลา(uS)
PSHA	1	3	3	1.5
LDAA #$00	1	2	2	1
DECA	256	2	512	256
BNE	256	3	768	384
PULA	1	4	4	2
RTS	1	5	5	2.5
				647

ตารางที่ 4.1 ตารางเวลาของโปรแกรม DELAY

ในไมโครคอนโทรลเลอร์ 68HC11A1FN มีใช้สัญญาณนาฬิกา 8 Mhz และสัญญาณนาฬิกาภายในมีค่า 2 Mhz ดังจะได้คาบเวลา

TIME = 1/ FREQUENCY

= 1/2*1000000

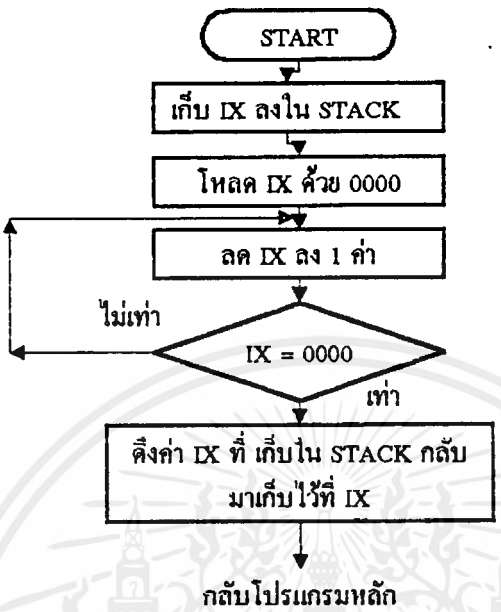
= 0.5 uS

นั่นคือ คาบเวลาที่ใช้ในการคำนวณมีค่า 0.5 uS

- DELAY2

เป็นโปรแกรมหน่วงเวลาที่มีการหน่วงเวลาที่ซ้ำการโปรแกรมหน่วงเวลา DELAY แต่การทำงานของโปรแกรมมีลักษณะเดียวกันกับ DELAY โปรแกรมนี้มีโฟลว์ชาร์จแสดงดังรูปที่ 4.8

การทำงานของโปรแกรมนี้ เริ่มต้นด้วยการโหลดค่า 0000 ให้แก่รีจิสเตอร์อินเด็กซ์ Y แล้วลดค่าในรีจิสเตอร์อินเด็กซ์ Y ลงหนึ่งค่าหลังจากนั้นก็นำค่าในรีจิสเตอร์อินเด็กซ์ Y มาเปรียบเทียบกับค่า 0000 โดยถ้าค่าในรีจิสเตอร์อินเด็กซ์ Y ไม่เท่ากับ 0000 ก็ให้กลับไปลดค่ารีจิสเตอร์อินเด็กซ์ Y ลงอีกหนึ่งค่า แล้วจึงนำมาเปรียบเทียบใหม่ เมื่อนำมาเปรียบเทียบแล้วพบว่าค่าในรีจิสเตอร์อินเด็กซ์ Y มีค่า 0000 ก็ให้กลับไปสู่การทำงานโปรแกรมหลัก



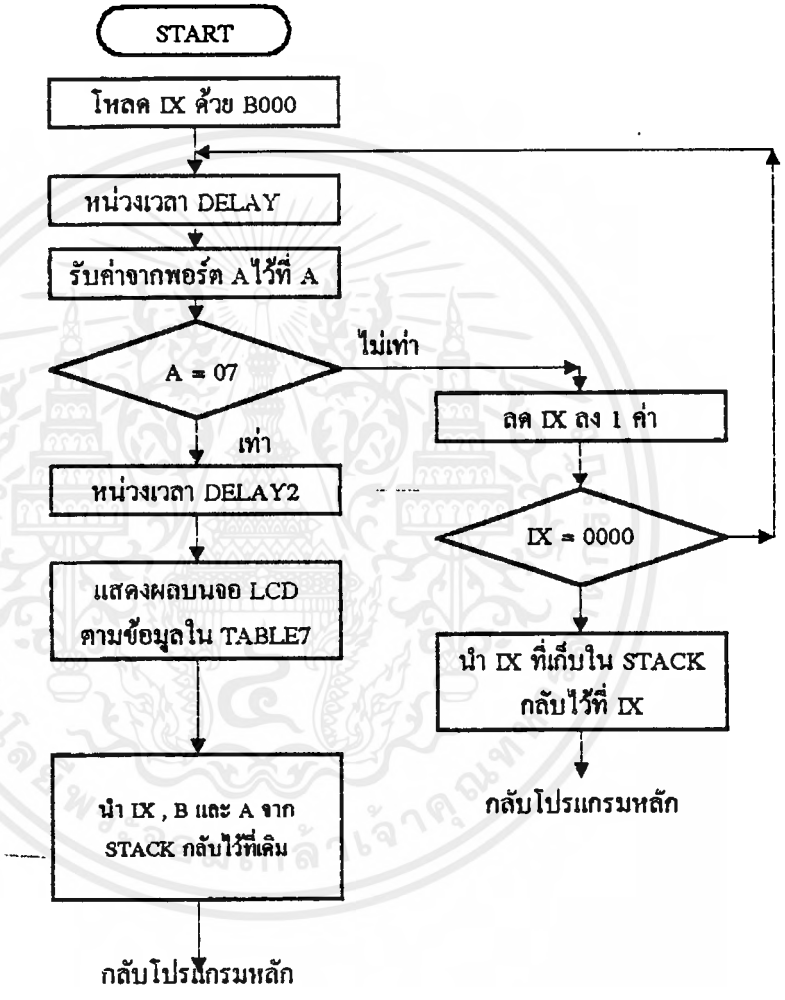
รูปที่ 4.8 โฟลว์ชาร์จโปรแกรมหน่วงเวลา DELAY2

	จำนวนครั้ง	จำนวน Cycle	Cycleทั้งหมด	เวลา (uS)
PSHX	1	4	4	2
LDX #$0000	1	3	3	1.5
DEX	65536	3	196,608	98,304
BNE	65536	3	196,608	98,304
PULX	1	5	5	2.5
RTS	1	5	5	2.5
				196,616

ตารางที่ 4.2 ตารางเวลาของโปรแกรม DELAY2

- DELAY4

โปรแกรมหน่วงเวลาส่วนนี้ มีโฟลว์ชาร์จแสดงดังรูปที่ 4.9



รูปที่ 4.9 โฟลว์ชาร์จโปรแกรมหน่วงเวลา DELAY4

โปรแกรมส่วนนี้นอกจากจะทำหน้าที่หน่วงเวลาแล้ว ยังทำหน้าที่ เป็นโปรแกรมรับคีย์ด้วย โดยการรับคีย์จะรับคีย์สวิทช์ S3 เพื่อออกจากโปรแกรมการส่งข้อมูลไปให้ไมโครคอมพิวเตอร์

การทำงานเริ่มต้นด้วยการโหลดค่า B000 ลงในรีจิสเตอร์อินเด็กซ์ X ต่อจากนั้นก็จะรับค่าจากพอร์ต A เพื่อตรวจจับว่ามีการกดสวิทช์ S3 หรือไม่ ถ้าไม่มีการกดสวิทช์ S3 ก็จะไปลดค่าในรีสเตอร์อินเด็กซ์ X ลง 1 ค่าแล้วเปรียบเทียบโดยถ้าค่าในรีจิสเตอร์อินเด็กซ์ X ไม่เท่ากับศูนย์ ก็จะกลับมารับค่าจากพอร์ต A อีกครั้ง และถ้ามีการกดสวิทช์ S3 โปรแกรมก็กระโดดกลับสู่โปรแกรมรับคีย์ชุดที่ 1

- DELAY5

โปรแกรมนี้เป็นโปรแกรมหน่วงเวลาที่มีลักษณะเช่นเดียวกับโปรแกรมหน่วงเวลา DELAY2แต่มีการหน่วงเวลาที่น้อยกว่า แสดงดังตารางที่ 4.3

	จำนวนครั้ง	จำนวนCycle	Cycle ทั้งหมด	เวลา (uS)
PSHX	1	4	4	2
LDX #$8000	1	3	3	1.5
DEX	32768	3	98,304	49,152
BNE	32768	3	98,304	49,152
PULX	1	5	5	2.5
RTS	1	5	5	2.5
				98,312.5

จากตารางจะเห็นว่าโปรแกรมหน่วงเวลา DELAY5 สามารถหน่วงเวลาได้ 98.3125 มิลลิวินาที

4.2 โปรแกรมแสดงผลบนจอไมโครคอมพิวเตอร์

โปรแกรมส่วนนี้จะเป็นโปรแกรมที่ออกแบบโดยภาษา C และเก็บไว้ในแผ่นดิสค์ และสามารถเรียกใช้งานได้บนไมโครคอมพิวเตอร์ที่สามารถแสดงภาพกราฟฟิกได้ โดยโปรแกรมส่วนนี้มีหน้าที่ดังต่อไปนี้

1. รับข้อมูลที่ส่งมาจากบอร์ดเครื่องวัดอุณหภูมิผ่านทางพอร์ตสื่อสารอนุกรม RS232
2. แสดงผลค่าอุณหภูมิเป็นกราฟ
3. แสดงผลค่าอุณหภูมิเป็นตัวเลข

การพัฒนาโปรรแกรมส่วนนี้ใช้ซอฟร์แวร์ TURBO C ในการพัฒนาโปรแกรมและโปรแกรม

ส่วนนี้อธิบายได้ดังนี้

เริ่มต้นของการทำงานโปรแกรมจะทำการเซ็ทค่าต่างๆ โดยเริ่มต้นของการทำงานจะทำการเซ็ทกราฟฟิกไดรเวอร์เป็นจอ VGA และเซ็ทกราฟฟิกโหมดเป็นจอขนาด 640x350 แบบ 16 สี การเซ็ทกราฟฟิกต่างๆนี้ใช้ฟังก์ชัน initgraph ของ Turbo C ซึ่งมีรูปแบบดังนี้

initgrph(&grph_driver,&grph_mode,&grph_path)

ดังนั้นจะได้ฟังก์ชันที่ใช้ในการเซ็ทกราฟฟิกดังนี้

```
grph_driver = VGA;
grph_mode = VGAMED;
grph_path  = ' ';
initgraph(&grph_driver,&grph_mode,&grph_path);
```

เมื่อมีการเซ็ทจอภาพแล้ว ต่อไปโปรแกรมจะทำการเซ็ทพอร์ตสื่อสารอนุกรม สำหรับการเซ็ทพอร์ตสื่อสารอนุกรม RS232 ของไมโครคอมพิวเตอร์นี้จะใช้อินเตอร์รัฟท์ฟังก์ชันของ BIOS ฟังก์ชัน 14 โดยการกำหนด ah = 0 และกำหนด al = E3 ซึ่งเป็นการกำหนดรูปแบบของพอร์ตสื่อสารอนุกรม RS232 ดังนี้

- กำหนดความยาวของคำเป็นแบบ 8 บิต
- กำหนดบิตหยุด 1 บิต
- กำหนดไม่ให้มีบิตพาริตี้ (parity)
- กำหนดอัตราบอด (baud rate) เป็น 9600

เมื่อมีการเซ็ทจอภาพและเซ็ทพอร์ตสื่อสารแล้วก็จะเริ่มการทำงานของระบบโครงงาน โดยเริ่มต้นของโปรแกรมระบบโครงงาน คือ การแสดงหน้าจอเป็นรูปกราฟที่ใช้แสดงผลค่าอุณหภูมิ

เมื่อแสดงหน้าจอที่จะใช้แสดงผลค่าอุณหภูมิแล้วโปรแกรมก็พร้อมที่จะรับค่าจากพอร์ตสื่อสารอนุกรม RS232 โดยฟังก์ที่ใช้รับค่าจากพอร์ต RS232 จะใช้อินเตอร์รัฟฟังก์ชันของ BIOS ฟังก์ชัน 14 โดยการกำหนด dx = 1 และ ah = 2 และค่าที่รับได้จากพอร์ตสื่อสารอนุกรม RS232 จะถูกเก็บไว้ในรีจิสเตอร์ al ค่าที่อยู่ในรีจิสเตอร์ al นี้จะถูกนำไปแสดงผลยังหน้าจอไมโครคอมพิวเตอร์

บทที่ 5
สรุปผลของโครงงาน

โครงงานเครื่องวัดอุณหภูมิ 6 ช่อง ซึ่งคณะผู้จัดทำได้ทำขึ้นมานี้ โดยเมื่อทำสำเร็จแล้วได้ผลดังนี้

- สามารถวัดอุณหภูมิได้ 6 ช่องพร้อมๆกัน
- สามารถแสดงผลได้ทางจอผลึกเหลว
-สามารถแสดงผลได้ทางจอไมโครคอมพิวเตอร์

สำหรับโครงงานนี้สามารถวัดอุณหภูมิได้ตั้งแต่ 0 - 100 เซลเซียส ซึ่งหากพัฒนาให้สามารถวัดค่ามากกว่านี้ก็สามารถกระทำได้เนื่องจากในไมโครคอนโทรลเลอร์เบอร์ 68HC11A1FN มีวงจรเปลี่ยนสัญญาณอนาล็อกเป็นดิจิตอล ได้ถึง 256 ค่า

ปัญหาที่เกิดขึ้นขณะทำโครงงานนี้

ปัญหาที่เกิดขึ้นในขณะทำโครงงานนี้สามารถแยกเป็นข้อๆได้ดังต่อไปนี้

1. ไมโครคอนโทรลเลอร์เบอร์ 68HC11A1FN ซึ่งเป็นข้อ บริษท โมโตโลรา ทางคณะผู้จัดทำเห็นมีฮาร์ดมากมายและเหมาะสมในการใช้งาน แต่ทางขณะผู้จัดทำยังไม่เคยศึกษาไมโครคอน-โทรลเลอร์เบอร์นี้มาก่อน จึงทำให้ต้องเสียเวลาในการศึกษา

2. ปัญหาก้นเกิดจากซอฟร์แวร์ที่ใช้งานของ ไมโครคอนโทรลเลอร์เบอร์นี้มีไม่มากจึงทำให้ต้องเสียเวลาในการหาซอฟร์แวร์เหล่านี้

3. ปัญหาเนื่องจากการปรับมาตรฐานของตัวตรวจจับค่าอุณหภูมิที่ต้องความละเอียดมาก จึงทำให้การวัดค่าอุณหภูมิมีการผิดพลาดขึ้น

ซึ่งจากการใช้งานในการวัดอุณหภูมินั้นได้ผลการวัดที่ยังแตกต่างกันเล็กน้อยของแต่ละช่อง โดยเมื่อวัดในจุดเดียวกัน สาเหตุเนื่องมาจากค่าผิดของตัวเซนเซอร์ (Sensor) ที่ใช้มีค่าผิดพลาดที่แตกต่างกันและเกิดจากการปรับค่ามาตรฐานที่ต้องใช้ความละเอียดสูงมากๆ

จากผลการใช้งานและการทำโครงงานนี้เมื่อกล่าวโดยรวมแล้วจะทำให้ทราบถึง การใช้งานไอซีไมโครคอนโทรลเลอร์เบอร์ 68HC11A1FN และการสื่อสารข้อมูลข้อมูลระหว่างโครงงานและระบบคอมพิวเตอร์ ได้เป็นอย่างดี ถึงแม้ว่าเราจะไม่ประสบผลสำเร็จดังที่ตั้งใจไว้ แต่เราก็ได้ความรู้ไม่น้อยเลยเกี่ยวกับการทำโครงงานนี้ ซึ่งสามารถนำไปประยุกต์ใช้งานในการทำงานจริงๆได้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของวิชา **PROJECT 2** ปีการศึกษา 2539 และสามารถสำเร็จลุล่วงไปได้ด้วยดี คณะผู้จัดทำจึงขอขอบพระคุณท่านอาจารย์ ไพศาล สิทธิโยภาสกุล และอาจารย์ มนชนก ศรีเสือขาม และอาจารย์คณะวิศวกรรมศาสตร์ทุกท่าน ที่ให้คำปรึกษา และแนะแนวทางแก้ไขปัญหาที่เกิดขึ้น และเพื่อนๆทุกท่าน ตลอดจนถึงเจ้าหน้าที่ห้องสมุดภาคเทคนิคอุตสาหกรรมที่ได้ให้ความช่วยเหลือในทุกๆด้านจึงทำให้ผลงานชิ้นนี้สำเร็จลุล่วงด้วยดี

คณะผู้จัดทำจึงขอขอบพระคุณไว้ ณ ที่นี้

ทรงสมศักดิ์ บุญมิ่ง

วีระ บุญไพโรจน์

หนังสืออ้างอิง

1. วิทยา เรืองวิสุทธ์ , ดร. , " คู่มือโปรแกรมภาษา C สำหรับผู้เริ่มต้น ", บริษัท ซีเอ็ดยูเคชั่น จำกัด , 2537

2. เฮอร์เบิร์ต ชิลด์ , " การประยุกต์ใช้งานภาษา C " , บริษัท ซีเอ็ด-ยูเคชั่น จำกัด , 2535

3. ชัยวัฒน์ พรจิตรวิไล , " เรียนรู้การใช้งาน ไมโครคอนโทรลเลอร์ 68HC11 " บริษัท ซีเอ็ดยูเคชั่น จำกัด , 2538

4. TECHNICAL , MC68HC11A8 , MOTOROLA INC. , 2534

ภาคผนวก

โปรแกรมส่วนที่ติดตั้งในระบบฮาร์ดแวร์

```
PA_CTRL:   .CEQU  $1026
PA_PORT:   .CEQU  $1000
CTRL_DATA: .CEQU  $1400
CHA_DATA:  .CEQU  $1401
ATC_DAT1:  .CEQU  $1031
ATC_DAT2:  .CEQU  $1032
ATC_DAT3:  .CEQU  $1033
ATC_DAT4:  .CEQU  $1034
ATC_CTRL:  .CEQU  $1030
          .ORG   $2200
; ---------------------------------------------------------------
;                 ********* MAIN PROGRAM *********
; ---------------------------------------------------------------
START:     LDD    #$00
START1:    DECA
           CMPA   #$00
           BNE    START1
START2:    DECB
           CMPB   #$00
           BNE    START2
           LDAA   #$93
           STAA   $1039
          LDAA   #$00
          STAA   $1024
          STAA   $1035
          LDAA   #$30
          STAA   $102B
          LDAA   #$00
          STAA   $102C
          LDAA   #$0C
```

```
        STAA   $102D
        LDAA   #$01
        STAA   $103D
        LDS    #$00FF
        LDAA   #$00
        STAA   PA_CTRL
        ldx    #$3200
        stx    $3f00
        ldx    #$3400
        stx    $3f02
        ldx    #$3600
        stx    $3f04
        ldx    #$3800
        stx    $3f06
        ldx    #$3a00
        stx    $3f08
        ldx    #$3c00
        stx    $3f0a
        LDX    #$0000
        STX    $3F0C
        JSR    INIT_LCD
        LDX    #TABLE
        JSR    WR_PORT
        JSR    DELAY
; ---------------------------------------------------------------
;                  SCAN KEY  ... MENU 1 ...
; ---------------------------------------------------------------
L1:     LDY    #$06
LOOP:   JSR    DELAY
        LDAA   PA_PORT        ; /READ DATA FROM PORT A
        ANDA   #$8F
        CMPA   #$07           ; /S4 KEY FOR SAND DATA TO PC.
```

```
        BEQ   SE_PC
        CMPA  #$83          ; /S1 KEY FOR DISPLAY
        BEQ   WR1_PORT
        CMPA  #$85          ; /S2 KEY FOR ENTER OR CLEAR
        BEQ   WR2_DATA
        JSR   DELAY
        LDAA  $3F2A         ; /SAVE PORT A TO $3F2A
        CMPA  #$83
        BEQ   DISPLAY
LOOP_1: JMP   LOOP
WR2_DATA: JMP  WRITE2
SE_PC:  JMP   SEND_PC
DISPLAY JSR   DELAY2
        JMP   DISPLAY_1
; ---------------------------------------------------------------
WR1_PORT: STAA  $3F2A
        CPY   #$06
        BEQ   LP1
        CPY   #$05
        BEQ   LP2
        CPY   #$04
        BEQ   LP3
        CPY   #$03
        BEQ   LP4
        CPY   #$02
        BEQ   LP5
        CPY   #$01
        BEQ   LP6
        JMP   L1
 LP1:    JMP   LOOP1
 LP2:    JMP   LOOP2
 LP3:    JMP   LOOP3
```

```
LP4:    JMP   LOOP4
LP5:    JMP   LOOP5
LP6:    JMP   LOOP6
;---------------------------------------------------------------
;                       CHANNEL 1
;---------------------------------------------------------------
LOOP1:   DEY
         LDX   #TABLE1
         JSR   CLEAR
         JSR   WR_PORT
         JSR   ATOC1
         JSR   DELAY
         LDAB  ATC_DAT1
         JSR   ATOC
         JMP   LOOP
;---------------------------------------------------------------
;                       CHANNEL 2
;---------------------------------------------------------------
LOOP2:   DEY
         LDX   #TABLE2
         JSR   CLEAR
         JSR   WR_PORT
         LDAB  ATC_DAT2
         JSR   ATOC
         JMP   LOOP
;---------------------------------------------------------------
;                       CHANNEL 3
;---------------------------------------------------------------
LOOP3:   DEY
         LDX   #TABLE3
         JSR   CLEAR
         JSR   WR_PORT
```

```
        LDAB   ATC_DAT3
        JSR    ATOC
        JMP    LOOP
;---------------------------------------------------------------
;                       CHANNEL 4
;---------------------------------------------------------------
LOOP4:    DEY
        LDX    #TABLE4
        JSR    CLEAR
        JSR    WR_PORT
        LDAB   ATC_DAT4
        JSR    ATOC
        JMP    LOOP
;---------------------------------------------------------------
;                       CHANNEL 5
;---------------------------------------------------------------
LOOP5:    DEY
        LDX    #TABLE5
        JSR    CLEAR
        JSR    WR_PORT
        JSR    ATOC2
        JSR    DELAY
        LDAB   ATC_DAT1
        JSR    ATOC
        JMP    LOOP
;---------------------------------------------------------------
;                       CHANNEL 6
;---------------------------------------------------------------
LOOP6:    DEY
        LDX    #TABLE6
        JSR    CLEAR
        JSR    WR_PORT
```

```
        LDAB   ATC_DAT2
        JSR    ATOC
        JMP    LOOP
; ---------------------------------------------------------------
;                       INITIAL LCD
; ---------------------------------------------------------------
INIT_LCD:  PSHA
        LDAA   #$0F          ;/Sets entire display,cursor on and blink
        STAA   CTRL_DATA
        JSR    DELAY
        LDAA   #$38          ;/Sets data length is 8 bit,display 2 line
        STAA   CTRL_DATA
        JSR    DELAY
        LDAA   #$06          ;/Sets DDRAM increment
        STAA   CTRL_DATA
        JSR    DELAY
        LDAA   #$01          ;/Clear display
        STAA   CTRL_DATA
        JSR    DELAY
        PULA
        RTS
; ---------------------------------------------------------------
;                    WRITE DATA TO LCD
; ---------------------------------------------------------------
WR_PORT:  LDAA   #$00          ;/Let start point of LCD
        JSR    GT_ADDR
        JSR    WR_OUT
        LDAA   #$40          :/Let start point of 2 line
        JSR    GT_ADDR
        JSR    WR_OUT
        RTS
WR_OUT:   LDAB   #$10          :/Let 16 character
```

```
TEST1:   LDAA  $00,X
         JSR   WR_BYTE
         PSHY
         LDY   #$0B00
L3:      DEY
         BNE   L3
         PULY
         INX
         DECB
         BNE   TEST1
         RTS
WR_BYTE: STAA  CHA_DATA
         JSR   RD_BUSY
         RTS
RD_BUSY: LDAA  CTRL_DATA
         ANDA  #$80
         BNE   RD_BUSY
         RTS
GT_ADDR: ADDA  #$80
         STAA  CTRL_DATA
         JSR   RD_BUSY
         RTS
; ---------------------------------------------------------------
;              SET  PORT C ( ANALOG TO DIGITAL )
; ---------------------------------------------------------------
ATOC1:   LDAA  #$30
         STAA  ATC_CTRL
         RTS
ATOC2:   LDAA  #$34
         STAA  ATC_CTRL
         RTS
ATOC:    LDAA  #$C9
```

```
        STAA  CTRL_DATA
        JSR   DELAY
        LDAA  #$DF
        STAA  CHA_DATA
        JSR   DELAY
        LDX   #TABLE7
LOP:    INX
        INX
        INX
        LDAA  $0,X
        CBA
        BNE   LOP
        INX
        LDAA  #$C5
        STAA  CTRL_DATA
        JSR   DELAY
        LDAB  #$02
LOP1:   LDAA  $0,X
        STAA  CHA_DATA
        JSR   DELAY
        INX
        DECB
        BNE   LOP1
        RTS
CLEAR:  LDAA  #$01
        STAA  CTRL_DATA
        JSR   DELAY
        RTS
; ---------------------------------------------------------------
RETURN: JSR   DELAY
        LDAA  PA_PORT
        CMPA  #$8E
```

```
        BEQ   RETURN_1
        RTS
RETURN_1: JSR   DELAY2
        LDX   #TABLE
        JSR   WR_PORT
        PULX
        PULB
        PULA
        JMP   L1
DS_6_CH1: LDAA  #$83
        STAA  CTRL_DATA
        JSR   DELAY
        JSR   D_6_CH
        RTS
DS_6_CH2: LDAA  #$89
        STAA  CTRL_DATA
        JSR   DELAY
        JSR   D_6_CH
        RTS
DS_6_CH3: LDAA  #$8E
        STAA  CTRL_DATA
        JSR   DELAY
        JSR   D_6_CH
        RTS
DS_6_CH4: LDAA  #$C3
        STAA  CTRL_DATA
        JSR   DELAY
        JSR   D_6_CH
        RTS
DS_6_CH5: LDAA  #$C9
        STAA  CTRL_DATA
        JSR   DELAY
```

```
        JSR    D_6_CH
        RTS
DS_6_CH6: LDAA  #$CE
        STAA   CTRL_DATA
        JSR    DELAY
        JSR    D_6_CH
        RTS
; ---------------------------------------------------------------
;                        DELAY
; ---------------------------------------------------------------
DELAY:    PSHA
        LDAA   #$00
DELAY_1:  DECA
        BNE    DELAY_1
        PULA
        RTS
DELAY2:   PSHX
        LDX    #$0000
DELAY_2:  DEX
        BNE    DELAY_2
        PULX
        RTS
DELAY3:   PSHX
        LDX    #$B000
DELAY_3:  JSR    RETURN
        DEX
        BNE    DELAY_3
        PULX
        RTS
DELAY4:   PSHX
        LDX    #$B000
DELAY_4:  JSR    SEND_RE
```

```
        DEX
        BNE   DELAY_4
        PULX
        RTS
DELAY5:  PSHX
        LDX    #$8000
DELAY_5: DEX
        BNE    DELAY_5
        PULX
        RTS
; ---------------------------------------------------------
:                    DISPLAY
; ---------------------------------------------------------
DISPLAY_1: CPY   #$05
        BEQ   DIS_1
        CPY   #$04
        BEQ   DIS_2
        CPY   #$03
        BEQ   DIS_3
        CPY   #$02
        BEQ   DIS_4
        CPY   #$01
        BEQ   DIS_5
        CPY   #$00
        BEQ   DIS_6
        JMP   LOOP_1
DIS_1:   LDAB  ATC_DAT1
        JSR   ATOC
        JMP   LOOP_1
DIS_2:   LDAB  ATC_DAT2
        JSR   ATOC
        JMP   LOOP_1
```

```
DIS_3:   LDAB  ATC_DAT3
         JSR   ATOC
         JMP   LOOP_1
DIS_4:   LDAB  ATC_DAT4
         JSR   ATOC
         JMP   LOOP_1
DIS_5:   LDAB  ATC_DAT1
         JSR   ATOC
         JMP   LOOP_1
DIS_6:   LDAB  ATC_DAT2
         JSR   ATOC
         JMP   LOOP_1
WRITE2:  JSR   DELAY2
         LDAA  PA_PORT
         CMPA  #$83
         BEQ   WR2_1
         CMPA  #$85
         BEQ   WR2_2
         JMP   WRITE2
WR2_1:   JMP   WR1_PORT
WR2_2:   LDX   #TABLE
         JSR   CLEAR
         JSR   WR_PORT
         JMP   L1
; ---------------------------------------------------------------
;                    ...SEND DATA TO PC...
; ---------------------------------------------------------------
SEND_PC: PSHA
         PSHB
         PSHX
         LDX   #TABLE9
         JSR   WR_PORT
```

```
SEND_1:   JSR   DELAY5
          LDAA  PA_PORT
          CMPA  #$85          ;/Enter DATA to PC.
          BEQ   SEND_2
          CMPA  #$07          ;/Return to MAIN MENU
          BEQ   SEND_ME1
          JMP   SEND_1
SEND_ME1: JMP   RE_ME1
SEND_2:   LDX   #TABLE14
          JSR   WR_PORT
SEND_3:   LDAB  #$04
SEND_4:   STAB  $3F20
          LDAA  #$30          ;/Set A/D PORT
          STAA  ATC_CTRL
;-----------------------------------------------------------------
          LDAB  #$01
          JSR   SE_TO_PC
          LDAB  ATC_DAT1      ;/Receive data to B-ACC.
          JSR   SE_TO_PC
          JSR   DS_6_CH1
;-----------------------------------------------------------------
          LDAB  #$02
          JSR   SE_TO_PC
          LDAB  ATC_DAT2
          JSR   SE_TO_PC
          JSR   DS_6_CH2
;-----------------------------------------------------------------
          LDAB  #$03
          JSR   SE_TO_PC
          LDAB  ATC_DAT3
          JSR   SE_TO_PC
          JSR   DS_6_CH3
```

```
;-----------------------------------------------------------------
        LDAB   #$04
        JSR    SE_TO_PC
        LDAB   ATC_DAT4
        JSR    SE_TO_PC
        JSR    DS_6_CH4
;-----------------------------------------------------------------
        LDAA   #$34
        STAA   ATC_CTRL
;-----------------------------------------------------------------
        LDAB   #$05
        JSR    SE_TO_PC
        LDAB   ATC_DAT1
        JSR    SE_TO_PC
        JSR    DS_6_CH5
;-----------------------------------------------------------------
        LDAB   #$06
        JSR    SE_TO_PC
        LDAB   ATC_DAT2
        JSR    SE_TO_PC
        JSR    DS_6_CH6
;-----------------------------------------------------------------
        JSR    DELAY4
        LDAB   $3F20
        DECB
        BNE    SEND_4
        JMP    SEND_3
SEND_RE:  JSR    DELAY
        LDAA   PA_PORT
        CMPA   #$07
        BEQ    SEND_RE1
        RTS
```

```
SEND_RE1: JSR   DELAY2
          LDX   #TABLE
          JSR   WR_PORT
          PULX
          PULB
          PULA
          JMP   L1
; ---------------------------------------------------------------
SE_TO_PC: PSHA
          JSR   DELAY
          LDAA  $102E          ;/Clear TDRE bit of SCSR
          LDAA  $102F
          STAB  $102F          ;/Send data form A/D Port to PC.
          PULA
          RTS
RE_ME1:   LDX   #TABLE
          JSR   WR_PORT
          JMP   L1
; ---------------------------------------------------------------
;                DISPLAY  6 CHANNEL
; ---------------------------------------------------------------
D_6_CH:   PSHX
          PSHA
          LDX   #TABLE7
L1_6_CH:  INX
          INX
          INX
          LDAA  $00,X
          CBA
          BNE   L1_6_CH
          INX
          LDAB  #$02
```

```
L2_6_CH:  LDAA  $00,X
          STAA  CHA_DATA
          JSR   DELAY
          INX
          DECB
          BNE   L2_6_CH
          PULA
          PULX
          RTS
; ---------------------------------------------------------------
;                          TABLE
; ---------------------------------------------------------------
TABLE1:   .DB   " THERMAL CH-1 "
          .DB   "       C    "
TABLE2:   .DB   " THERMAL CH-2 "
          .DB   "       C    "
TABLE3:   .DB   " THERMAL CH-3 "
          .DB   "       C    "
TABLE4:   .DB   " THERMAL CH-4 "
          .DB   "       C    "
TABLE5:   .DB   " THERMAL CH-5 "
          .DB   "       C    "
TABLE6:   .DB   " THERMAL CH-6 "
          .DB   "       C    "
; ---------------------------------------------------------------
;                  TABLE 7 : COMPARE OF DATA
; ---------------------------------------------------------------
TABLE7:   .DB   $00,$00,$00
          .DB   $0D,$30,$30,$0E,$30,$31,$0F,$30,$32,$10,$30,$33
          .DB   $11,$30,$34,$12,$30,$35,$13,$30,$36,$14,$30,$37
          .DB   $15,$30,$38,$16,$30,$39,$17,$31,$30,$18,$31,$31
          .DB   $19,$31,$32,$1A,$31,$33,$1B,$31,$34,$1C,$31,$35
```

```
        .DB    $1D,$31,$36,$1E,$31,$37,$1F,$31,$38,$20,$31,$39
        .DB    $21,$32,$30,$22,$32,$31,$23,$32,$32,$24,$32,$33
        .DB    $25,$32,$34,$26,$32,$35,$27,$32,$36,$28,$32,$37
        .DB    $29,$32,$38,$2A,$32,$39,$2B,$33,$30,$2C,$33,$31
        .DB    $2D,$33,$32,$2E,$33,$33,$2F,$33,$34,$30,$33,$35
        .DB    $31,$33,$36,$32,$33,$37,$33,$33,$38,$34,$33,$39
        .DB    $35,$34,$30,$36,$34,$31,$37,$34,$32,$38,$34,$33
        .DB    $39,$34,$34,$3A,$34,$35,$3B,$34,$36,$3C,$34,$37
        .DB    $3D,$34,$38,$3E,$34,$39,$3F,$35,$30,$40,$35,$31
        .DB    $41,$35,$32,$42,$35,$33,$43,$35,$34,$44,$35,$35
        .DB    $45,$35,$36,$46,$35,$37,$47,$35,$38,$48,$35,$39
        .DB    $49,$36,$30,$4A,$36,$31,$4B,$36,$32,$4C,$36,$33
        .DB    $4D,$36,$34,$4E,$36,$35,$4F,$36,$36,$50,$36,$37
        .DB    $51,$36,$38,$52,$36,$39,$53,$37,$30,$54,$37,$31
        .DB    $55,$37,$32,$56,$37,$33,$57,$37,$34,$58,$37,$35
        .DB    $59,$37,$36,$5A,$37,$37,$5B,$37,$38,$5C,$37,$39
        .DB    $5D,$38,$30,$5E,$38,$31,$5F,$38,$32,$60,$38,$33
        .DB    $61,$38,$34,$62,$38,$35,$63,$38,$36,$64,$38,$37
        .DB    $65,$38,$38,$66,$38,$39,$67,$39,$30,$68,$39,$31
        .DB    $69,$39,$32,$6A,$39,$33,$6B,$39,$34,$6C,$39,$35
        .DB    $6D,$39,$36,$6E,$39,$37,$6F,$39,$38,$70,$39,$39
        .DB    $71,$30,$30
TABLE8: .DB    "   SAVE DATA   "
        .DB    " 30 SECOND/BYTE "
TABLE9: .DB    "   SEND DATA   "
        .DB    "    TO PC      "
TABLE14: .DB   "1=   2=   3= "
        .DB    "4=   5=   6= "
TABLE:  .DB    "  6 CHANNELS  "
        .DB    "  THERMOMETER  "
        .END
```

โปรแกรมส่วนที่ติดตั้งบนไมโครคอมพิวเตอร์

```
#include <stdio.h>
#include <dos.h>
#include <graphics.h>
#include <stdlib.h>
#include <string.h>

#define ESC 27

#define BLUE        1
#define GREEN       2
#define RED         4
#define INTENSE     8
#define BLUE_BACK   16
#define GREEN_BACK  32
#define RED_BACK    64
#define BLINK       128

FILE *fp;

void goto_xy(),color_puts(),read_cursor_xy(),palette();
void keyi(),clr_scr(),tim(),baud();

main()
{
  int j,grph_driver,grph_mode;
  char grph_path,re,ikey;
  char file_name[20];
  grph_driver=VGA;
  grph_mode=VGAMED;
  grph_path= '';
```

```
initgraph(&grph_driver,&grph_mode,&grph_path);
baud();
project();
{
  union REGS r;
  char key;
  r.h.ah = 6;
  r.h.dl = 0xFF;
  loop:
  int86(0x21,&r,&r);
  key = r.h.al;
  if(key == ESC) {
    clr_scr();
    exit(1);
   }
   goto loop;
 }
}

project()
{
  clrscr();
  setbkcolor(BLUE);
  subl();
  goto_xy(29,0);
  printf("  6 CHANNEL THERMOMETER  ");
  subchan();
  subline();
 }
```

```
subl()
{
  int i=40;
  int j=0;
  int b=0;
  int x,y;
  y=23;
  goto_xy(3,6);
  putchar(248);
  printf("C");
  setcolor(LIGHTRED);
  line(40,100,40,320);
  goto_xy(79,23);
  printf("M");
  line(40,320,640,320);
  for(i=100;i<320;i=i+20) {
  line(38,i,42,i);
  }
  for(i=60;i<640;i=i+20) {
  line(i,318,i,322);
  j=j+1;
    if(j==5) {
      b=10;
      x=17;
      goto_xy(x,y);
      printf("%d",b);
    }
    else if(j==10) {
      b=20;
      x=29;
```

```
      goto_xy(x,y);
      printf("%d",b);
    }
    else if(j==15) {
      b=30;
      x=41;
      goto_xy(x,y);
      printf("%d",b);
    }
  , else if(j==20) {
      b=40;
      x=54;
      goto_xy(x,y);
      printf("%d",b);
    }
    else if(j==25) {
      b=50;
      x=67;
      goto_xy(x,y);
      printf("%d",b);
    }
  }
}

subline()
{
  static int x1,x2,x3,x4,x5,x6;
  static int y1,y2,y3,y4,y5,y6;
  int endx1,endx2,endx3,endx4,endx5,endx6;
  int t1,t2,t3,t4,t5,t6,d,j,a,b,i,k;
```

```
endx1=endx2=endx3=endx4=endx5=endx6=10;
x1=x2=x3=x4=x5=x6=40;
y1=y2=y3=y4=y5=y6=320;
b=0;
d=i=0;
lable1:
b=b+1;
if(b==361) {
  b=361-b;
  clrscr();
  project();
}
{
union REGS r;
while(!(check_stat(1)&256))
  if(kbhit()) {
    getch();
    clr_scr();
    exit(1);
  }
  r.x.dx = 1;
  r.h.ah = 2;
  int86(0x14,&r,&r);
  d = r.h.al;
  if(d==0x01) {
        goto lable2;
  }
  else if(d==0x02) {
        goto lable3;
  }
```

```
else if(d==0x03) {
        goto lable4;
}
else if(d==0x04) {
        goto lable5;
}
else if(d==0x05) {
        goto lable6;
}
else if(d==0x06) {
        goto lable7;
}
else goto lable1;
}
lable2:
  k = 0;
  i = receive(k);
  t1=320-(i*2);
  a=i;
  goto_xy(70,1);
  printf("%3d",a);
  endx1=x1+10;
  subline1(x1,y1,endx1,t1);
  x1=x1+10;
  y1=t1;
  goto lable1;

lable3:
  i = receive(k);
  t2=320-(i*2);
```

```
a=i;
goto_xy(70,2);
printf("%3d",a);
endx2=x2+10;
subline2(x2,y2,endx2,t2);
x2=x2+10;
y2=t2;
goto lable1;

lable4:
i = receive(k);
t3=320-(i*2);
a=i;
goto_xy(70,3);
printf("%3d",a);
endx3=x3+10;
subline3(x3,y3,endx3,t3);
x3=x3+10;
y3=t3;
goto lable1;

lable5:
i = receive(k);
t4=320-(i*2);
a=i;
goto_xy(70,4);
printf("%3d",a);
endx4=x4+10;
subline4(x4,y4,endx4,t4);
x4=x4+10;
```

```
    y4=t4;
    goto lable1;

  lable6:
    i = receive(k);
    t5=320-(i*2);
    a=i;
    goto_xy(70,5);
    printf("%3d",a);
    endx5=x5+10;
    subline5(x5,y5,endx5,t5);
    x5=x5+10;
    y5=t5;
    goto lable1;

  lable7:
    i = receive(k);
    t6=320-(i*2);
    a=i;
    goto_xy(70,6);
    printf("%3d",a);
    endx6=x6+10;
    subline6(x6,y6,endx6,t6);
    x6=x6+10;
    y6=t6;
    goto lable1;
}

subline1(x11,y11,endx11,t11)
int x11,y11,endx11,t11;
```

```
{
  setcolor(BLUE);
  delay(10);
  line(x11,y11,endx11,t11);
}

subline2(x22,y22,endx22,t22)
int x22,y22,endx22,t22;
{
  setcolor(GREEN);
  delay(10);
  line(x22,y22,endx22,t22);
}

subline3(x33,y33,endx33,t33)
int x33,y33,endx33,t33;
{
  setcolor(RED);
  delay(10);
  line(x33,y33,endx33,t33);
}

subline4(x44,y44,endx44,t44)
int x44,y44,endx44,t44;
{
  setcolor(MAGENTA);
  delay(10);
  line(x44,y44,endx44,t44);
}
```

```
subline5(x55,y55,endx55,t55)
int x55,y55,endx55,t55;
{
  setcolor(DARKGRAY);
  delay(10);
  line(x55,y55,endx55,t55);
}

subline6(x66,y66,endx66,t66)
int x66,y66,endx66,t66;
{
  setcolor(YELLOW);
  delay(10);
  line(x66,y66,endx66,t66);
 }

 subchan()
 {
   int y,i;
   y=13;
   i=0;
   setcolor(BLUE);
   for(i=0;i<14;i++) {
   y=y+1;
   line(590,y,600,y);
   }
   goto_xy(75,1);
   printf("CH-1");
   setcolor(GREEN);
   for(i=0;i<14;i++) {
```

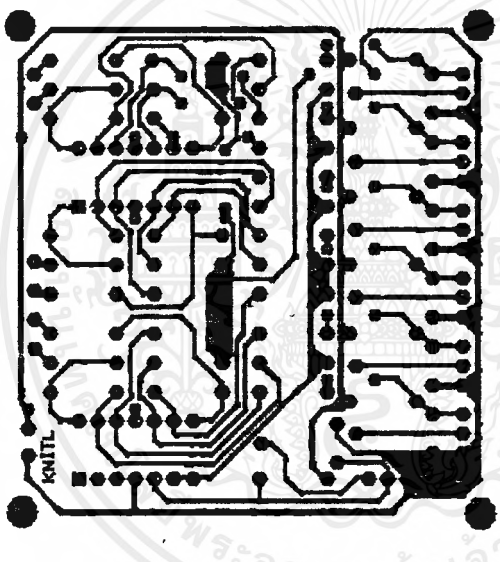
```
y=y+1;
line(590,y,600,y);
}
goto_xy(75,2);
printf("CH-2");
setcolor(RED);
for(i=0;i<14;i++) {
y=y+1;
line(590,y,600,y);
}
goto_xy(75,3);
printf("CH-3");
setcolor(MAGENTA);
for(i=0;i<14;i++) {
y=y+1;
line(590,y,600,y);
}
goto_xy(75,4);
printf("CH-4");
setcolor(DARKGRAY);
for(i=0;i<14;i++) {
y=y+1;
line(590,y,600,y);
}
goto_xy(75,5);
printf("CH-5");
setcolor(YELLOW);
for(i=0;i<14;i++) {
y=y+1;
line(590,y,600,y);
```

```
  }
  goto_xy(75,6);
  printf("CH-6");
}

void color_puts(s,color)
char *s;
char color;
{
  union REGS r;
  char x,y;
  read_cursor_xy(&x,&y);
  while(*s) {
    if(*s=='\n') {
      printf("\n");
      s++;
      x = 0;
      y++;
      continue;
    }
    r.h.ah = 9;
    r.h.al = *s++;
    r.h.bl = color;
    r.h.bh = 0;
    r.x.cx = 1;
   int86(0x10,&r,&r);
  }
}

 void read_cursor_xy(x,y)
```

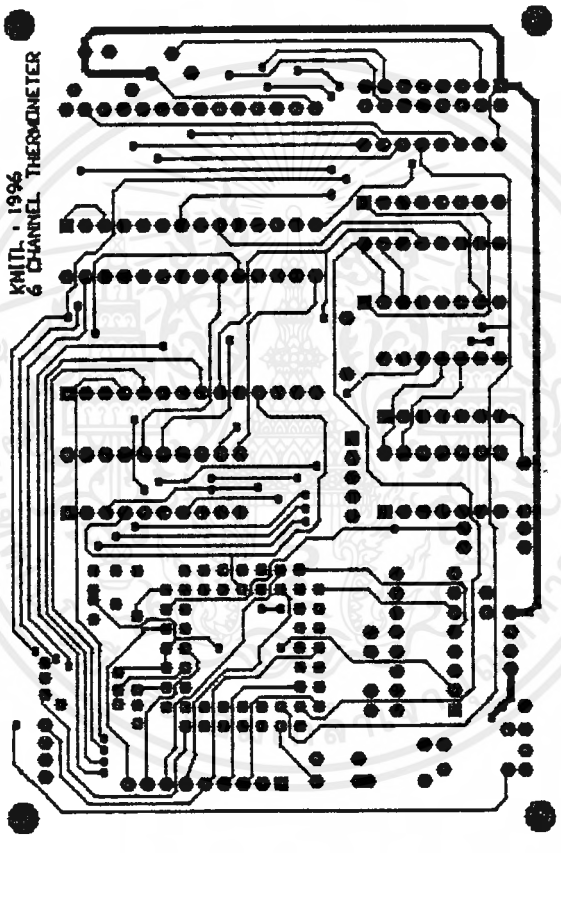
```
char *x,*y;
{
  union REGS r;
  r.h.ah = 3;
  r.h.bh = 0:
  int86(0x10,&r,&r);
  *y = r.h.dl;
  *x = r.h.dh;
}

void palette(pnum)
int pnum;
{
  union REGS r;
  r.h.bh = 1;
  r.h.bl = pnum;
  r.h.ah = 11;
  int86(0x10,&r,&r);
}

void goto_xy(x,y)
int x,y;
{
  union REGS r;
  r.h.ah = 2;
  r.h.dl = x;
  r.h.dh = y;
  r.h.bh = 0;
  int86(0x10,&r,&r);
}
```

```
void clr_scr()
{
  clrscr();
  textmode(3);
}

void tim()
{
  union REGS r;
  int hh,mm,ss;
  r.h.ah = 0x2C;
  int86(0x21,&r,&r);
  hh = r.h.ch;
  mm = r.h.cl;
  ss = r.h.dh;
  goto_xy(70,0);
  printf("%02d",hh);
  printf(":");
  printf("%02d",mm);
  printf(":");
  printf("%02d",ss);
}

void baud()
{
  union REGS r;
  r.h.ah = 0;
  r.h.al = 0xE3;
  int86(0x14,&r,&r);
```

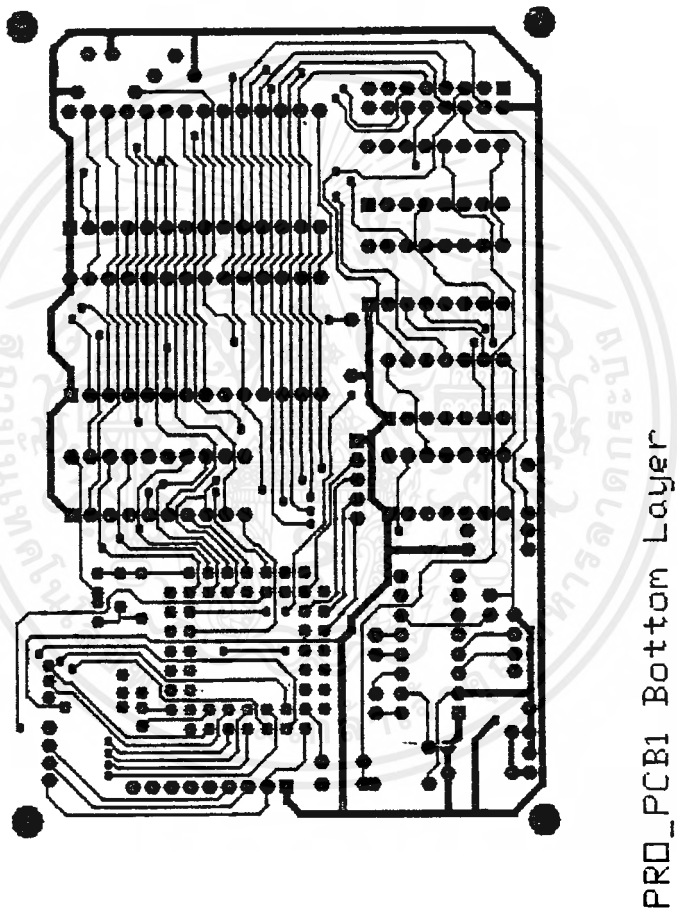
```
}

check_stat()
{
  union REGS r;
  r.x.dx = 1;
  r.h.ah = 3;
  int86(0x14,&r,&r);
}

receive(l)
int l;
{
  int h;
  union REGS r;
  r.x.dx = 1;
  r.h.ah = 2;
  int86(0x14,&r,&r);
  h = r.h.al;
  l = h-13;
  return(l);
}
```



รูปลายวงจรพิมพ์ด้านล่างของวงจรส่วนเปลี่ยนค่าอุณหภูมิเป็นแรงดัน



รูปลายวงจรพิมพ์ด้านล่างของวงจรส่วนควบคุม



รูปลายวงจรพิมพ์ด้านบนของวงจรส่วนควบคุม

APPENDIX A
ELECTRICAL CHARACTERISTICS

Table A-1. Maximum Ratings

Rating	Symbol	Value	Unit
Supply Voltage	V_{DD}	−0.3 to + 7.0	V
Input Voltage	V_{in}	−0.3 to + 7.0	V
Operating Temperature Range MC68HC11A8 MC68HC11A8C MC68HC11A8V MC68HC11A8M	T_A	T_L to T_H 0 to 70 −40 to 85 −40 to 105 −40 to 125	°C
Storage Temperature Range	T_{stg}	−55 to 150	°C
Current Drain per Pin* Excluding V_{DD}, V_{SS}, V_{RH}, and V_{RL}	I_D	25	mA

*One pin at a time, observing maximum power dissipation limits.

Internal circuitry protects the inputs against damage caused by high static voltages or electric fields; however, normal precautions are necessary to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Extended operation at the maximum ratings can adversely affect device reliability. Tying unused inputs to an appropriate logic voltage level (either GND or V_{DD}) enhances reliability of operation.

Table A-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Average Junction Temperature	T_J	$T_A + (P_D \times \Theta_{JA})$	°C
Ambient Temperature	T_A	User-determined	°C
Package Thermal Resistance (Junction-to-Ambient) 52-Pin Plastic Quad Pack (PLCC) 48-Pin Plastic Dual In-Line Package (DIP)	Θ_{JA}	 50 40	°C/W
Total Power Dissipation	P_D	$P_{INT} + P_{I/O}$ $K \div (T_J + 273°C)$ (Note 1)	W
Device Internal Power Dissipation	P_{INT}	$I_{DD} \times V_{DD}$	W
I/O Pin Power Dissipation	$P_{I/O}$ (Note 2)	User-determined	W
A Constant	K	$P_D \times (T_A + 273°C) + \Theta_{JA} \times P_D^2$ (Note 3)	W · °C

NOTES:
1. This is an approximate value, neglecting $P_{I/O}$.
2. For most applications $P_{I/O} \ll P_{INT}$ and can be neglected.
3. K is a constant pertaining to the device. Solve for K with a known T_A and a measured P_D (at equilibrium). Use this value of K to solve for P_D and T_J iteratively for any value of T_A.

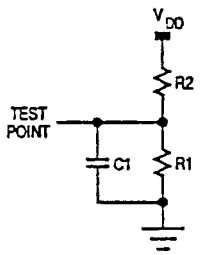
Table A-3. DC Electrical Characteristics

V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Characteristics		Symbol	Min	Max	Unit
Output Voltage (Note 1) I_{Load} = ± 10.0 µA	All Outputs except XTAL All Outputs Except XTAL, RESET, and MODA	V_{OL} V_{OH}	— V_{DD} – 0.1	0.1 —	V
Output High Voltage (Note 1) I_{Load} = – 0.8 mA, V_{DD} = 4.5 V	All Outputs Except XTAL, RESET, and MODA	V_{OH}	V_{DD} – 0.8	—	V
Output Low Voltage I_{Load} = 1.6 mA	All Outputs Except XTAL	V_{OL}	—	0.4	V
Input High Voltage	All Inputs Except RESET RESET	V_{IH}	0.7 x V_{DD} 0.8 x V_{DD}	V_{DD} + 0.3 V_{DD} + 0.3	V
Input Low Voltage	All Inputs	V_{IL}	V_{SS} – 0.3	0.2 x V_{DD}	V
I/O Ports, Three-State Leakage $V_{in} = V_{IH}$ or V_{IL}	PA7, PC0-PC7, PD0-PD5, AS/STRA, MODA/LIR, RESET	I_{OZ}	—	±10	µA
Input Leakage Current (Note 2) $V_{in} = V_{DD}$ or V_{SS} $V_{in} = V_{DD}$ or V_{SS}	PA0-PA2, IRQ, XIRQ MODB/V_{STBY}	I_{in}	— —	±1 ±10	µA
RAM Standby Voltage	Power down	V_{SB}	4.0	V_{DD}	V
RAM Standby Current	Power down	I_{SB}	—	10	µA
Total Supply Current (Note 3) RUN: Single-Chip Mode Expanded Multiplexed Mode WAIT: All Peripheral Functions Shut Down Single-Chip Mode Expanded Multiplexed Mode STOP: No Clocks, Single-Chip Mode		I_{DD} dc – 2 MHz 3 MHz dc – 2 MHz 3 MHz W_{IDD} dc – 2 MHz 3 MHz dc – 2 MHz 3 MHz S_{IDD} dc – 2 MHz 3 MHz	 — — — — — — — — — —	 15 27 27 35 6 15 10 20 50 150	mA mA µA
Input Capacitance PA0-PA2, PE0-PE7, IRQ, XIRQ, EXTAL PA7, PC0-PC7, PD0-PD5, AS/STRA, MODA/LIR, RESET		C_{in}	— —	8 12	pF
Power Dissipation Single-Chip Mode Expanded Multiplexed Mode Single-Chip Mode Expanded Multiplexed Mode		P_D dc – 2 MHz 3 MHz 	 — — — —	 85 150 150 195	mW

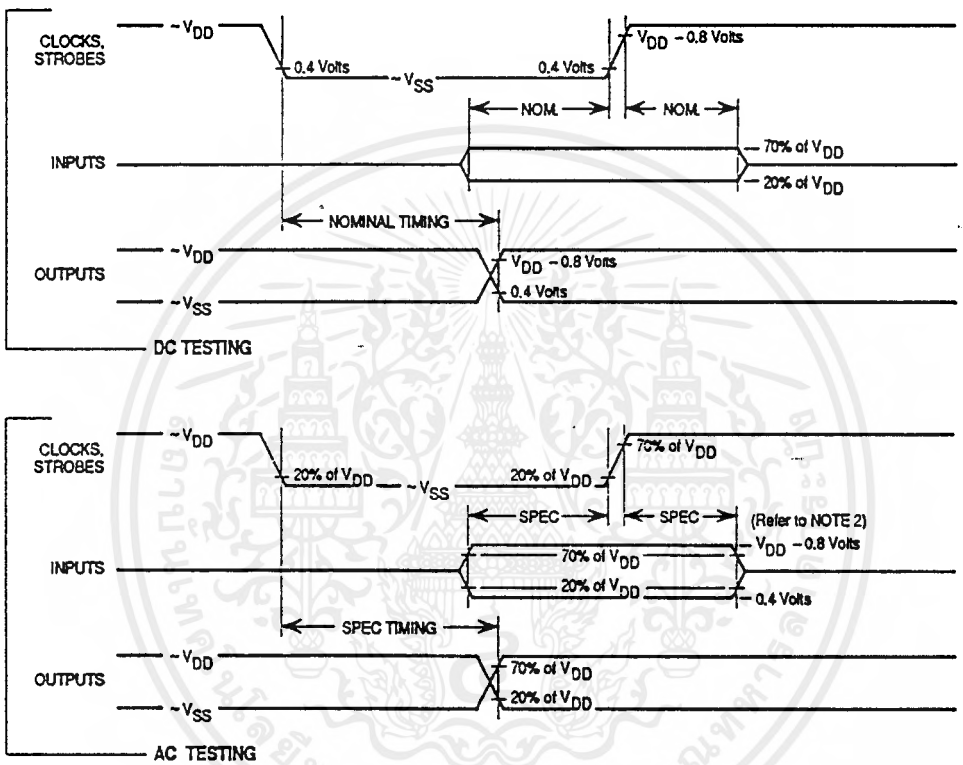
NOTES:

1. V_{OH} specification for RESET and MODA is not applicable because they are open-drain pins. V_{OH} specification not applicable to ports C and D in wired-OR mode.
2. Refer to A/D specification for leakage current for port E.
3. EXTAL is driven with a square wave, and
   t_{cyc} = 500 ns for 2 MHz rating;
   t_{cyc} = 333 ns for 3 MHz rating.
   $V_{IL} \leq 0.2$ V
   $V_{IH} \geq V_{DD} - 0.2$ V
   No dc loads.



Equivalent Test Load [1]

Pins	R1	R2	C1
PA3 – PA7 PB0 – PB7 PC0 – PC7 PD0, PD4 E, AS, R/W	3.26K	2.38K	90pF
PD1 – PD4	3.26K	2.38K	200pF



NOTES:
1. Full test loads are applied during all DC electrical tests and AC timing measurements.
2. During AC timing measurements, inputs are driven to 0.4 volts and V_{DD} – 0.8 volts while timing measurements are taken at the 20% and 70% of V_{DD} points.

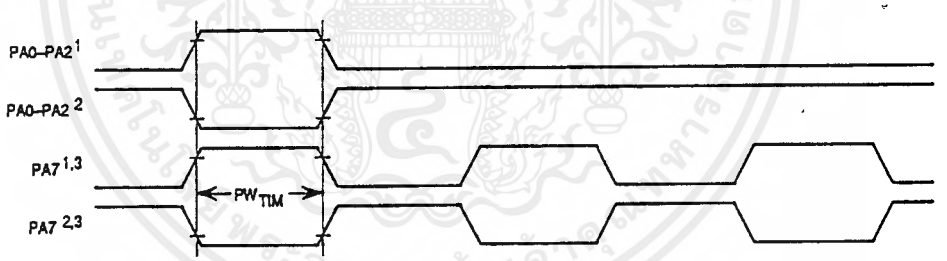
Figure A-1. Test Methods

Table A-4. Control Timing

V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H

Characteristic	Symbol	1.0 MHz		2.0 MHz		3.0 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of Operation	f_o	dc	1.0	dc	2.0	dc	3.0	MHz
E-Clock Period	t_{cyc}	1000	—	500	—	333	—	ns
Crystal Frequency	f_{XTAL}	—	4.0	—	8.0	—	12.0	MHz
External Oscillator Frequency	4 f_o	dc	4.0	dc	8.0	dc	12.0	MHz
Processor Control Setup Time t_{PCSU} = 1/4 t_{cyc} + 50 ns	t_{PCSU}	300	—	175	—	133	—	ns
Reset Input Pulse Width (Note 1) (To Guarantee External Reset Vector) (Minimum Input Time; Can Be Preempted by Internal Reset)	PW_{RSTL}	8 1	— —	8 1	— —	8 1	— —	t_{cyc}
Mode Programming Setup Time	t_{MPS}	2	—	2	—	2	—	t_{cyc}
Mode Programming Hold Time	t_{MPH}	10	—	10	—	10	—	ns
Interrupt Pulse Width, IRQ Edge-Sensitive Mode $PW_{IRQ} = t_{cyc} + 20$ ns	PW_{IRQ}	1020	—	520	—	353	—	ns
Wait Recovery Startup Time	t_{WRS}	—	4	—	4	—	4	t_{cyc}
Timer Pulse Width Input Capture, Pulse Accumulator Input $PW_{TIM} = t_{cyc} + 20$ ns	PW_{TIM}	1020	—	520	—	353	—	ns

NOTES:
1. RESET is recognized during the first clock cycle it is held low. Internal circuitry then drives the pin low for four clock cycles, releases the pin, and samples the pin level two cycles later to determine the source of the interrupt. Refer to SECTION 5 RESETS AND INTERRUPTS for further detail.
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted.



NOTES:
1. Rising edge sensitive input.
2. Falling edge sensitive input.
3. Maximum pulse accumulator clocking rate is E-clock frequency divided by 2.

Figure A-2. Timer Inputs

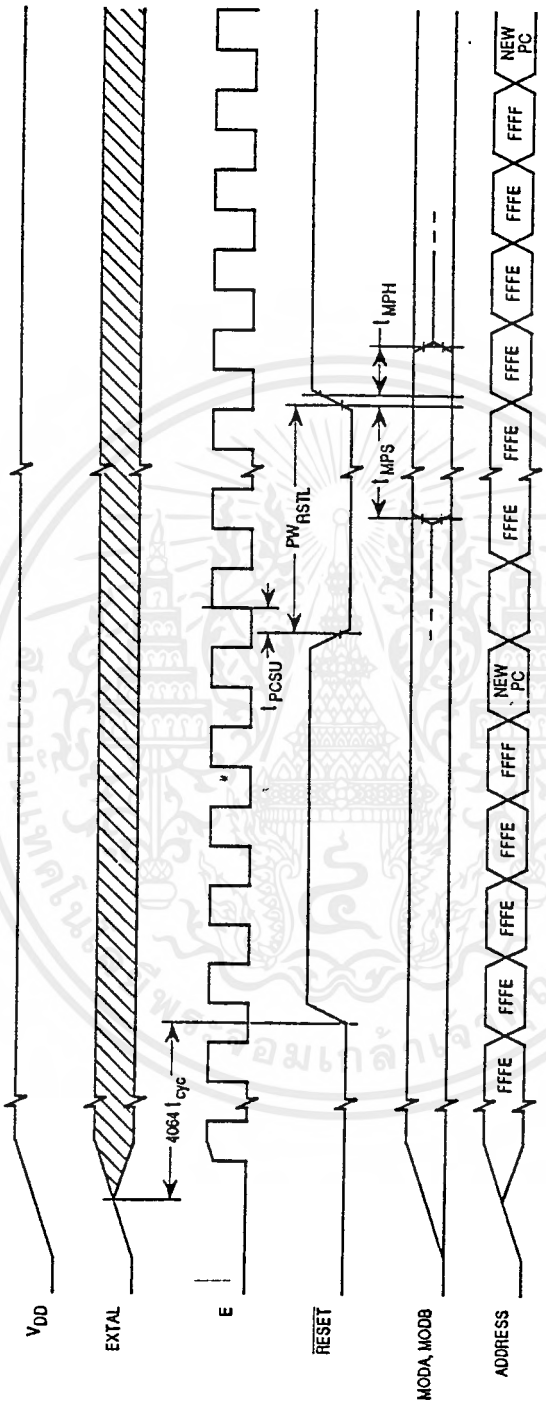
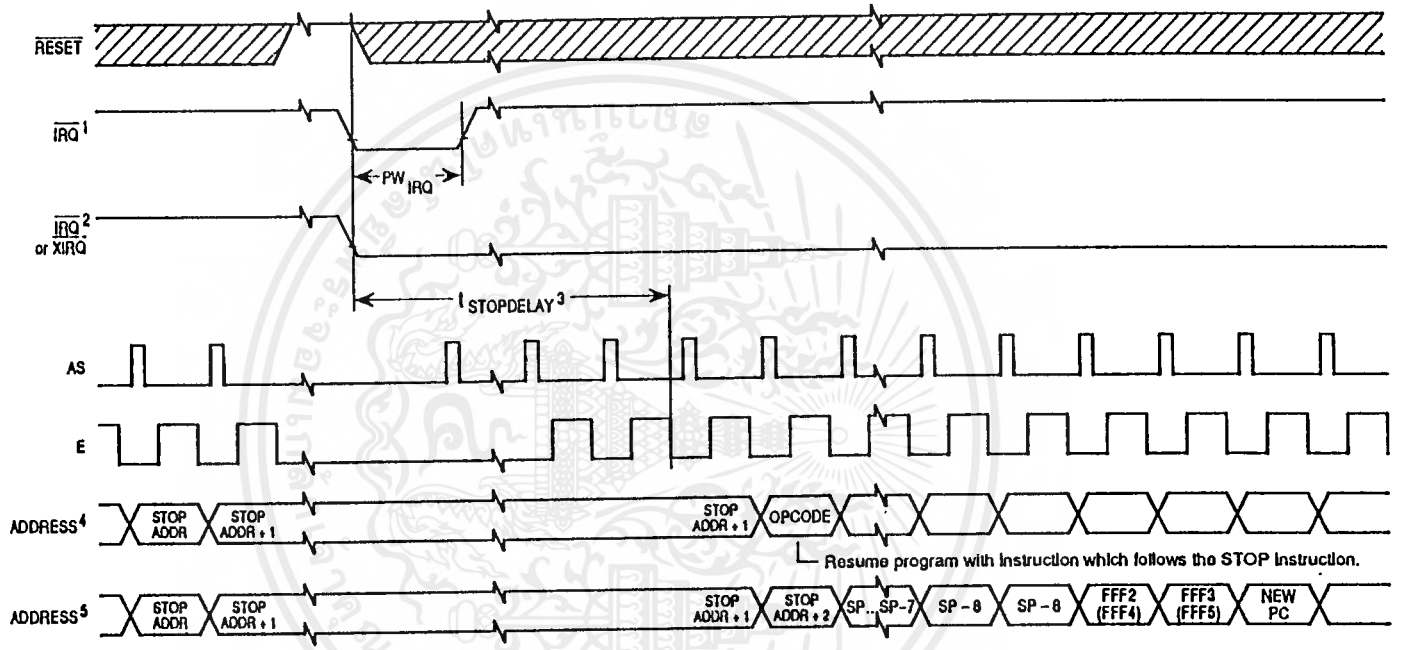


Figure A-3. POR and External Reset Timing Diagram



NOTES:
1. Edge Sensitive $\overline{IRQ}$ pin (IRQE bit = 1)
2. Level Sensitive $\overline{IRQ}$ pin (IRQE bit = 0)
3. $t_{STOPDELAY}$ = 4064 t_{cyc} if DLY bit = 1 or 4 t_{cyc} if DLY = 0.
4. $\overline{XIRQ}$ with X bit in CCR = 1.
5. $\overline{IRQ}$ or ($\overline{XIRQ}$ with X bit in CCR = 0).

Figure A-4. STOP Recovery Timing Diagram

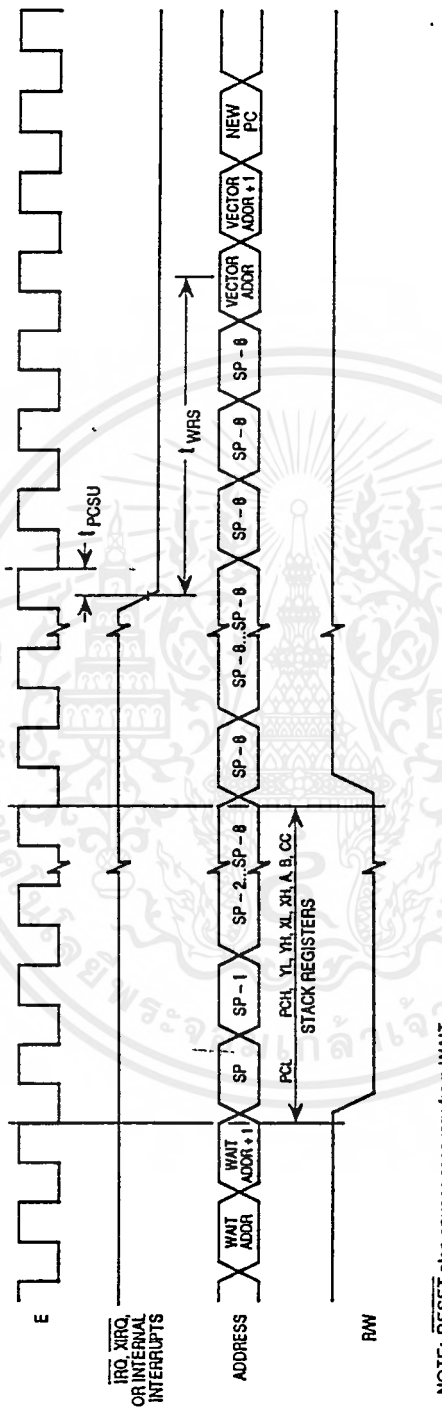


Figure A-5. WAIT Recovery Timing Diagram