Advance Single Board

โดย

นาย ศิริศักดิ์	จังคศิริ	36014439
นาย อัศวิน	ศรีเศรษฐนิล	36014563

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ปีการศึกษา 2539

Advance Single Board

โดย

นาย ศิริศักดิ์ จังคศิริ 36014439

นาย อัศวิน ศรีเศรษฐนิล 36014563

อาจารย์ที่ปรึกษา

อาจารย์ สมศักดิ์ มิตะถา

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง Advance Single Board

ผู้จัดทำ 1. นาย ศิริศักดิ์ จังคศิริ 36014439

2. นาย อัศวิน ศรีเศรษฐนิล 36014563

อาจารย์ที่ปรึกษา .....

(อาจารย์ สมศักดิ์ มิตะถา)

Advance Single Board

นาย ศิริศักดิ์ จังคศิริ
นาย อัศวิน ศรีเศรษฐนิล
อาจารย์ สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
ปีการศึกษา 2539

บทคัดย่อ

ในปัจจุบันได้มีบริษัทต่างๆพัฒนาซิงเกิลบอร์ด (Single Board) รุ่นต่างๆออกมาสู่ท้องตลาดมากมาย ซึ่งแต่ละรุ่น แต่ละบริษัท ต่างก็มีจุดเด่นจุดด้อยต่างกันไป โครงงานนี้จึงมีจุดประสงค์สำคัญคือ เป็นการศึกษาข้อมูลเกี่ยวกับลักษณะที่น่าสนใจของซิงเกิลบอร์ดแต่ละแบบซึ่งได้รับความนิยมในการใช้งานจริงอยู่ในปัจจุบัน และนำข้อมูลดังกล่าวที่ได้มาพัฒนาและออกแบบซิงเกิลบอร์ดรุ่นใหม่ให้มีความสามารถ ,คุณลักษณะที่ผู้ใช้สามารถนำไปใช้ประโยชน์ตามต้องการได้โดยสะดวกมากยิ่งขึ้น

โดยในการออกแบบ ได้มีการออกแบบใหม่หมด ทั้งในส่วนของฮาร์ดแวร์ (Hardware) ที่ใช้ชิป (Chip) ในตระกูล MCS-51 ซึ่งได้รับความนิยมอย่างสูงในปัจจุบันเป็นหน่วยประมวลผลกลาง (Central Processing Unit) หลัก และในด้านซอฟแวร์ (Software) ซึ่งประกอบไปด้วยโปรแกรมย่อย(Subprogram) ที่สำคัญๆ เช่น โปรแกรมควบคุมการทำงานของซิงเกิลบอร์ด (Monitor) , โปรแกรมสร้าง/ แก้ไขข้อความ (Editor), โปรแกรมแปลรหัสคำสั่ง (Assembler), โปรแกรมตรวจสอบการทำงาน (Debugger) ฯ ที่จำเป็นสำหรับผู้ใช้งานเพื่อนำไปใช้ในการศึกษาการทำงาน , พัฒนาต้นแบบอุปกรณ์ต่างๆ ฯ ตามต้องการต่อไป

Advance Single Board

Sirisak Jangkasiri
Asawin Srisetanil
Somsak Mittatha Advisor
1996

Abstract

At the present time,There are a lot of firms which develop many generation of single-board . However each generation of each company has individual strong point and weak point . The important purpose of this project is studying the interesting data of characteristic of them which are popular in market place in order to use it for developing new model of single-board which has wanted capability and a user can use it friendly.

For design process,It is divided into hardware and software. Anyway, the design of them is new design . For hardware, it uses MCS-51 chip ,a popular CPU family at the present, for controling single board . For software, it is included with essential subroutine ,the necessary software tools , such as monitor, editor, assembler, debugger etc. . And a user can use them for studying, developing prototype,and others in the future

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

เนื่องจากความก้าวหน้าทางเทคโนโลยีในปัจจุบัน เป็นผลให้มีการนำไมโครโปรเซสเซอร์ (Microprocessor) มาใช้ในชีวิตประจำวันมากขึ้น ไม่ว่าจะเป็นเครื่องใช้ไฟฟ้าทั่วไป ,ระบบควบคุม ,ระบบสื่อสารข้อมูล ,เครื่องมือวัดทางไฟฟ้า ,ของเด็กเล่น ,ตลอดจนใช้ในการควบคุมแต่ละจุดการทำงาน ในกระบวนการผลิตของโรงงานอุตสาหกรรม ล้วนแล้วแต่มีไมโครโปรเซสเซอร์เป็นตัวควบคุมการทำงานทั้งสิ้น

แต่จากการพัฒนาที่มีมากขึ้นเป็นลำดับ จึงได้มีการนำวงจรพื้นฐานที่จำเป็นรวมเข้าไปในไมโครโปรเซสเซอร์ ซึ่งเรียกรวมกันว่าไมโครคอนโทรลเลอร์ (Microcontroller) ทำให้ไมโครคอนโทรลเลอร์มีความสามารถมากขึ้น ในขณะที่ขนาดของวงจรและจำนวนชิพ (Chip) ที่ใช้ในวงจรมีน้อยลง

MCS-51 เป็นไมโครคอนโทรลเลอร์แบบชิพเดี่ยว ที่ได้รับความสนใจมากตัวหนึ่งในปัจจุบัน ทั้งนี้เป็นเพราะชิปในตระกูล MCS-51 มีข้อดีเมื่อเทียบกับไมโครโปรเซสเซอร์ขนาด 8 บิตในตระกูลอื่นดังต่อไปนี้

1. มีแรมบรรจุไว้ภายใน 128-256 ไบต์ (Bytes)
2. มีวงจรตั้งเวลา(Timer)/วงจรนับ(Counter) ขนาด 16 บิต (Bits) 2 ตัวอยู่ภายใน
3. มีวงจรรับส่งข้อมูลอนุกรมได้ 2 ทิศทาง โดยสามารถกำหนดอัตราเร็วในการรับและส่งข้อมูล (baud rate) ได้ตั้งแต่ 300 ถึง 375 กิโลบิตต่อวินาที
4. มีสัญญาณนาฬิกาภายในตัว
5. มีพอร์ต (Port) ที่สามารถรับหรือส่งข้อมูลได้ทั้ง 2 ทิศทาง จำนวน 4 พอร์ตๆ ละ 8 บิต หรือ สามารถใช้งานเป็นพอร์ตขนาด 1 บิตแยกจากกัน ทำให้เสมือนมีพอร์ตขนาด 1 บิต ใช้งานรวมทั้งสิ้น 32 พอร์ต
6. เป็นชิปที่ได้รับความนิยมสูง จึงมีราคาถูก และหาง่าย

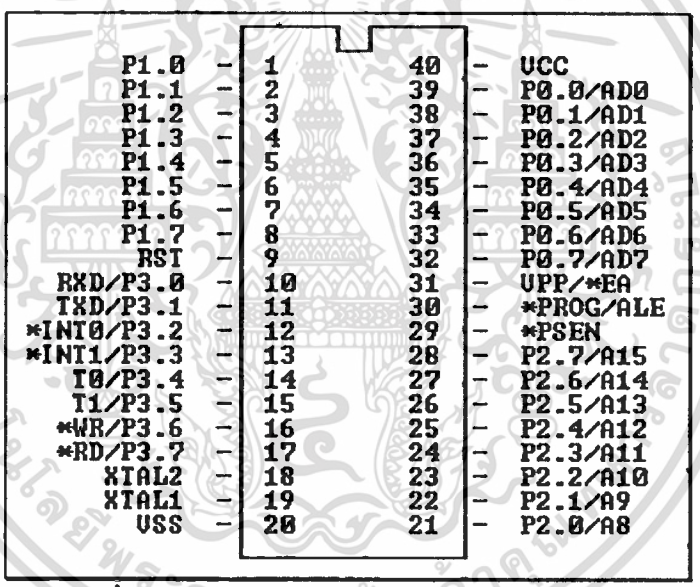
นอกจากนี้ MCS-51 ยังมีคุณสมบัติอื่นๆ ที่น่าสนใจคือ

- ต้องการแหล่งจ่ายไฟ 5 โวลต์ (Volts) เพียงชุดเดียว

- มีหน่วยความจำสำหรับเก็บโปรแกรมควบคุมการทำงานอยู่ภายในชิปอยู่ด้วย ซึ่งอาจเป็นรอม (ROM) หรืออีพรอม (Eprom) ขึ้นอยู่กับรุ่นของชิป เช่น 8XC51 มีขนาด 4 กิโลไบต์ , 80C52 มีขนาด 8 กิโลไบต์ , 8XC51FB มีขนาด 16 กิโลไบต์ , 8XC51FC มีขนาด 32 กิโลไบต์ เป็นต้น (เบอร์ 8031,8032 ไม่มีหน่วยความจำส่วนนี้)

- สามารถใช้หน่วยความจำ สำหรับโปรแกรมและข้อมูลที่อยู่ภายนอกชิปได้อย่างละ 64 กิโลไบต์

- มีคำสั่งคูณและหารเลขขนาด 8 บิตในตัวเอง



ภาพที่ 1 ภาพแสดงการจัดวางขาต่างๆ ของ MCS-51

<u>เป้าหมายของโครงงาน</u>

โครงงานนี้มีจุดประสงค์เพื่อที่จะพัฒนาต้นแบบซิงเกิลบอร์ดขึ้นมาใหม่ โดยนำเอาจุดดีจุดด้อยของซิงเกิลบอร์ดในท้องตลาดที่ยังได้รับความนิยมใช้อยู่ในปัจจุบัน เช่น JAZZ 31, MPF1 และ ET4.0 เป็นต้น มาทำการศึกษาเพื่อใช้ในการปรับปรุงต้นแบบขึ้นมาใหม่ ดังที่แสดงไว้ในตารางที่ 1

	JAZZ-31	ET 4.0	MPF-1	ANA-2.0
หน่วยประมวลผลกลาง	MCS51	Z80	Z80	Z80
การแสดงผล	เซเว่นเซกเมนต์ (7-Segment)	ตัวแสดงผลแบบผลึกเหลว (LCD) แบบ Dot Matrix 16 ตัวอักษร X 1 บรรทัด	ฟลูออเรสเซนต์ (FLuorescent)	เซเว่นเซกเมนต์
การป้อนโปรแกรม	เลขฐาน 16	Mnemonics	Mnemonics	เลขฐาน 16
แอสเซมเบลอร์ในตัว	ไม่มี	มี(ทีละบรรทัด)	มี	ไม่มี
รูปแบบคีย์บอร์ด	คีย์เลขฐาน 16 จำนวน 16 คีย์ + 12 ฟังก์ชันคีย์	32 คีย์ ตัวอักษร + ตัวเลข	คล้ายเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) ตัวอักษร + ตัวเลข	คีย์เลขฐาน 16 จำนวน 16 คีย์ + 20 ฟังก์ชันคีย์
อื่นๆ	นิยมใช้ทั่วไป	นิยมใช้ทั่วไป	หายาก	นิยมใช้ทั่วไป

ตารางที่ 1 ตารางเปรียบเทียบลักษณะของซิงเกิลบอร์ดในท้องตลาด

โดยลักษณะที่สำคัญ (specification) ของซิงเกิลบอร์ดในโครงงานมีดังต่อไปนี้

<u>ฮาร์ดแวร์</u> (HARDWARE)

- ใช้ชิปในตระกูล MCS-51 เป็นหน่วยประมวลผลกลาง
- แสดงผลโดยใช้อุปกรณ์แสดงผลแบบผลึกเหลว LCD (Liquid crystal display) แบบอักขระ (Character LCD Module) ขนาด 4 บรรทัด 20 ตัวอักษร
- มีคีย์บอร์ดภาษาอังกฤษแบบเคอตี้ (Qwerty) ในตัวดังภาพที่ 12
- มีบัสและพอร์ตขยายระบบที่เข้ากันได้กับ JAZZ 31

ซอฟแวร์ (SOFTWARE)

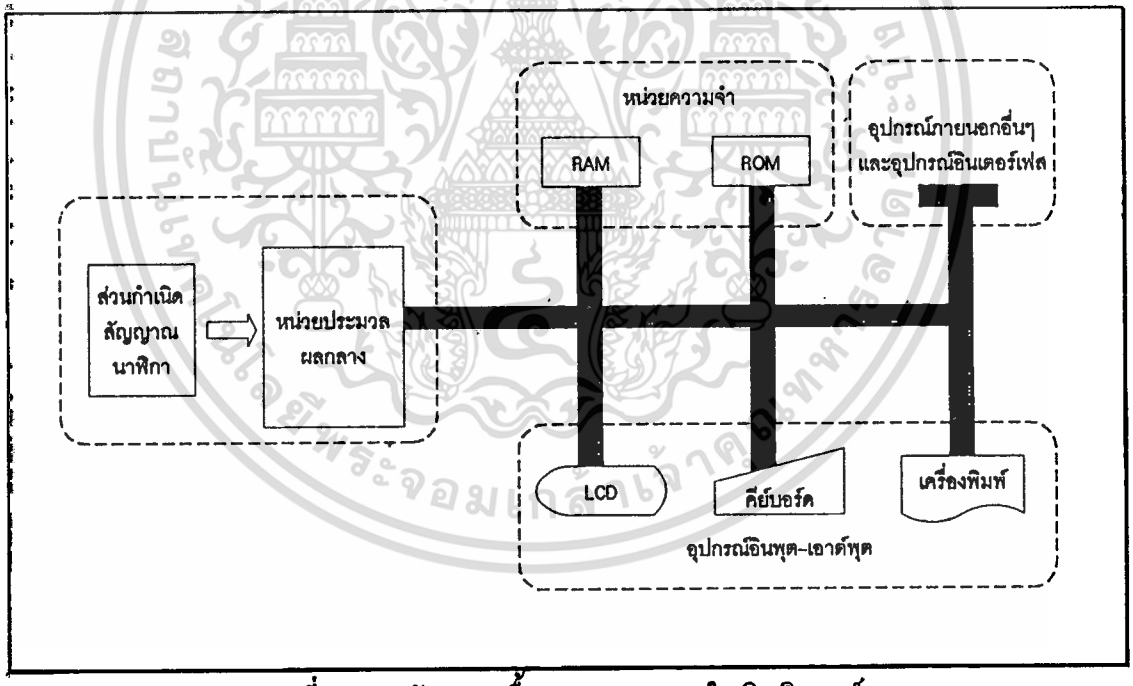
- ป้อนโปรแกรมด้วย Mnemonics ได้
- มีโปรแกรมสร้าง/แก้ไขข้อความ (Editor) และ โปรแกรมแปลรหัสคำสั่ง (Assembler) ในตัว สามารถทำการโปรแกรมและทดสอบโปรแกรมได้ในตัว ทำให้ใช้งานได้โดยลำพัง (Stand Alone) ได้อย่างคล่องตัวขึ้น
- มีโปรแกรมใช้งานทั่วไป (Software Utility) เช่น เครื่องคิดเลข , นาฬิกา เป็นต้น
- เปลี่ยนไปใช้ภาษาเบสิคควบคุมไมโครคอนโทรเลอร์ได้
- สามารถส่งข้อมูล (Upload), รับข้อมูล (Download) และติดต่อกับเครื่องคอมพิวเตอร์ส่วนบุคคลเป็นเทอร์มินอล (Teminal) หรือ Remote Access ผ่านทางพอร์ตอนุกรม RS-232 ของเครื่องคอมพิวเตอร์ส่วนบุคคลได้

ประโยชน์ที่คาดว่าจะได้รับ

ผู้ที่ศึกษาทดลองทางด้านไมโครคอนโทรลเลอร์ มีเครื่องมือที่มีประสิทธิภาพ ทำให้การเรียนรู้และพัฒนาเทคโนโลยีทางด้านไมโครคอนโทรลเลอร์เป็นไปได้อย่างรวดเร็ว

บทที่ 2
ทฤษฎีและหลักการทั่วไป

จากภาพที่ 2 จะแสดงโครงสร้างโดยทั่วไปของซิงเกิลบอร์ด ซึ่งมีความคล้ายคลึงกับระบบไมโครคอมพิวเตอร์ทั่วไป ซึ่งไม่ได้มีแค่เพียงหน่วยประมวลผลกลางเพียงอย่างเดียวเท่านั้น แต่จะประกอบไปด้วย หน่วยความจำทั้งที่เป็น แรม RAM (Random Access Meimmory) ,และรอม ROM (Read Only Memory) และอุปกรณ์อินพุต (Input) เอาต์พุต (Output) ต่างๆ ที่ทำงานร่วมกับหน่วยประมวลผลกลางด้วย



ภาพที่ 2 แผนผังแสดงพื้นฐานของระบบในซิงเกิลบอร์ด

องค์ประกอบพื้นฐานของโครงสร้างของซิงเกิลบอร์ด แบ่งออกเป็นส่วนต่างๆได้ดังนี้

- หน่วยประมวลผลกลาง (Central Processing Unit) เป็นส่วนสำคัญที่ทำหน้าที่ควบคุมการทำงานของส่วนต่างๆของระบบให้เป็นไปตามโปรแกรมที่กำหนดไว้ ซึ่งเก็บไว้ในหน่วยความจำ

- วงจรกำเนิดสัญญาณนาฬิกา (Clock Oscillator) วงจรกำเนิดสัญญาณนาฬิกาเป็นพื้นฐานของระบบ ที่ทำให้แต่ละส่วนของระบบทำงานได้อย่างสัมพันธ์กัน
- หน่วยความจำ (Memory) คือส่วนที่ใช้สำหรับเก็บข้อมูลและโปรแกรม หน่วยความจำแบ่งออกได้เป็น 2 ชนิดคือ รอม และ แรม
    1. รอม (Read Only Memory) คือหน่วยความจำที่ใช้สำหรับเก็บโปรแกรมที่ควบคุมการทำงานของระบบ ถึงแม้จะปิดไฟเลี้ยงแล้วก็ตาม ข้อมูลที่อยู่ในหน่วยความจำรอมนี้ก็จะไม่หายไปด้วย
    2. แรม (Random Access Memory) คือหน่วยความจำในระบบคอมพิวเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือตัวแปรแต่ละชนิดแบบชั่วคราว แต่มีข้อเสียคือเมื่อตัดไฟออกแล้วจะทำให้ข้อมูลหรือสิ่งที่อยู่ในหน่วยความจำนี้หายไป
- อุปกรณ์อินพุต - เอาต์พุต ระบบของคอมพิวเตอร์มีการทำงานตามโปรแกรมที่ได้ตั้งไว้ และจำเป็นต้องมีขบวนการในการส่งผลลัพธ์ออกไปแสดงผลภายนอกโดยผ่านพอร์ตเอาต์พุต เช่น จอแสดงผลแบบผลึกเหลว ,หรือในการปฏิบัติการบางอย่างมีความจำเป็นต้องอ่านข้อมูลจากภายนอกโดยผ่านพอร์ตอินพุต เช่น คีย์บอร์ด เป็นต้น
- อื่นๆ ได้แก่ภาคจ่ายไฟ , ระบบไฟสำรอง , ระบบตรวจสอบแรงดัน , ส่วนถอดรหัสตำแหน่งแอดเดรส (Address Decoder) เป็นต้น

บทที่ 3

การออกแบบและการสร้าง

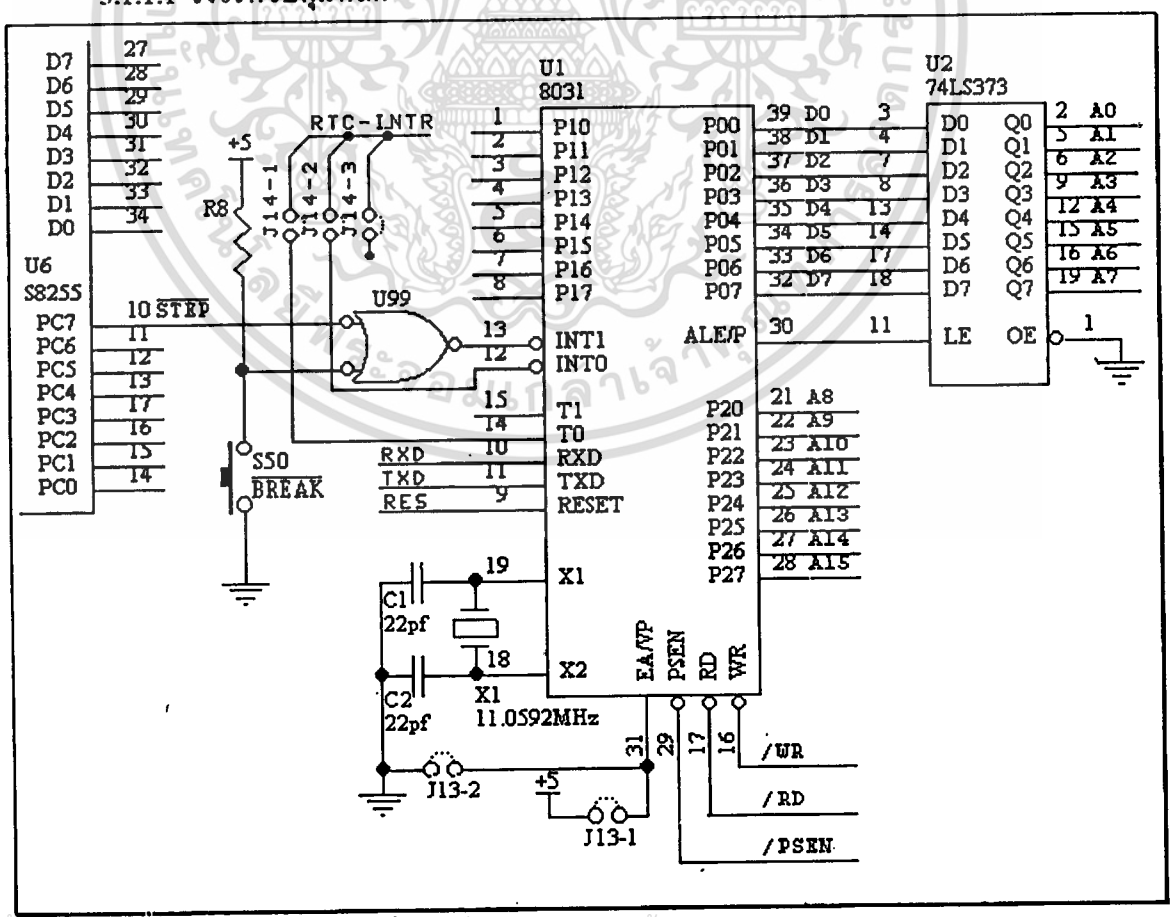
แบ่งออกเป็น 2 ส่วนคือ 1. ฮาร์ดแวร์ (Hardware) 2. ซอฟแวร์ (Software)

3.1) ฮาร์ดแวร์

3.1.1) การออกแบบวงจร

วงจรของชิงเกิลบอร์ดประกอบด้วยส่วนต่างๆที่จำเป็น ได้แก่ ส่วนควบคุมหลัก, ส่วนถอดรหัสตำแหน่งแอดเดรส (Address Decoder) หลัก, ส่วนถอดรหัสแอดเดรสของอุปกรณ์ อินพุต / เอาต์พุต และวงจรย่อยอื่นๆ โดยจะอธิบายรายละเอียดไปในแต่ละส่วนดังนี้

3.1.1.1 วงจรควบคุมหลัก



ภาพที่ 3 ภาพแสดงวงจรส่วนควบคุมหลัก

จากวงจรในภาพที่ 3 เป็นส่วนควบคุมหลัก หัวใจของวงจรอยู่ที่ไมโครคอนโทรลเลอร์ ตระกูล MCS-51 เบอร์ 80C31 ที่พอร์ท P0 หรือบัสข้อมูล (Data Bus) ต่อเข้ากับ 74LS373 ซึ่งเป็นแลตช์ (Latch) ขนาด 8-บิต เพื่อทำหน้าที่แยกและเก็บรักษา (Latch) สัญญาณแอดเดรส (Address) ไบต์ล่าง (A0-A7) ออกจากสัญญาณข้อมูลโดยใช้สัญญาณจากขา ALE (Address Latch Enable) จากหน่วยประมวลผลกลางเป็นตัวควบคุมการแลทช์

ที่ขา INT1 ต่อเข้ากับสัญญาณที่ได้จากการแอนด์ (AND) กันของสัญญาณให้ทำงานทีละขั้น (SINGLE-STEP) ที่สร้างโดย S8255 (U6) พอร์ต C.7 และสัญญาหยุด (Break) ที่ได้จากสวิตช์กด 'BREAK' ทำให้การเกิดอินเตอร์รัป (Interrupt) สามารถเกิดได้สองทางคือจากทางฮาร์ดแวร์คือสั่งทาง สวิตช์ 'BREAK' หรือจาก ซอฟท์แวร์โดยการ เซต (Set) /เคลียร์ (Clear) บิตที่ 7 ของพอร์ต C ของ U6 นั่นเอง ซึ่งสัญญาณที่ได้นี้จะนำไปใช้ในการทำงานโหมดซิงเกิลสเตป (Single.Step Mode) ซึ่งหน่วยประมวลผลกลางจะทำงานทีละคำสั่ง

ที่ขา $EA/\overline{VP}$ สามารถใช้จัมเปอร์ (Jumper) วางบนคอนเนคเตอร์ J13 คู่ที่ 2 เพื่อต่อ $EA/\overline{VP}$ ลงกราวด์ ดังในภาพที่ 3 เพื่อให้หน่วยประมวลผลกลางอ่านโปรแกรมจากรอมภายนอก เนื่องจาก 8031 ไม่มีรอมภายในนั่นเอง หรือหากจะเปลี่ยนไปใช้ชิปที่มีรอมภายในด้วยในตัวด้วย อย่างชิปหมายเลข 8751 หรือ 8951 เพื่อต้องการใช้รอมที่มีอยู่ภายใน ก็สามารถใช้จัมเปอร์เชื่อมคอนเนคเตอร์ J13 คู่ที่ 1 เพื่อให้ $EA/\overline{VP}$ ต่อกับไฟเลี้ยง +5 โวลท์ เพื่อเลือกใช้รอมภายในได้เช่นกัน ดังแสดงในตารางที่ 2

ความหมาย	คู่ที่ 1	คู่ที่ 2
ต่อ $EA/\overline{VP}$ ลงกราวด์	open	Close
ต่อ $EA/\overline{VP}$ กับไฟ +5 V	Close	open

ตารางที่ 2 ตารางแสดงวิธีการวางจัมเปอร์บนคอนเนคเตอร์ J13

หมายเหตุ Close - ใส่จัมเปอร์ลงไป

open - ไม่ใส่จัมเปอร์

ที่ขา X1 และ X2 ต่อเข้ากับคริสตอล 11.0592 MHz โดยสาเหตุที่ใช้คริสตอล (Crystal)ความถี่ 11.0592 MHz เนื่องจากจะทำให้วงจรสร้างฐานเวลาภายในตัว 80C31 เองกำหนดคาบเวลาในการสื่อสารแบบอะซิงโครนัส (Asynchronous) ผ่านทางขา TxD และขา RxD ได้ใกล้เคียงกับค่ามาตรฐานที่สุด ซึ่งจะได้กล่าวถึงในรายละเอียดของหน่วยประมวลผลกลางต่อไป ถึงแม้ว่าการใช้คริสตอลความถี่ 12 MHz จะทำให้คาบของเวลาการทำงาน 1 รอบการทำงาน

(Machine Cycle) ลงตัวที่ 1 ไมโครวินาทีก็ตาม แต่เมื่อเปรียบเทียบถึงข้อดีข้อเสียแล้ว การใช้คริสตอลความถี่ 11.0592 MHz จะทำให้การออกแบบระบบทำได้ง่ายกว่า หากระบบนั้นต้องการใช้วงจรสื่อสารแบบอนุกรม (Serial) ภายในตัวหน่วยประมวลผลกลาง ด้วย

สำหรับขา $\overline{INT0}$ และ $\overline{T0}$ สามารถจะเชื่อมต่อเพื่อรับสัญญาณอินเตอร์รัป (Interrupt) จากขา 1 ของ 6264 ชิปฐานเวลาจริง (Real Time Clock) คือ RTC-INTR หรือจะไม่รับก็ได้ โดยการวางจัมเปอร์ลงบนคอนเนคเตอร์ J14 ตามตำแหน่งดังที่แสดงไว้ในตารางที่ 3 โดยรายละเอียดการทำงานของชิปฐานเวลาจริง นั้นจะกล่าวถึงในเรื่องต่อๆไป

ความหมาย	คู่ที่ 1	คู่ที่ 2	คู่ที่ 3
เชื่อม Interrupt Signal จาก RTC กับ T0 ของ MCS-51	Close	open	open
เชื่อม Interrupt Signal จาก RTC กับ INT0 ของ MCS-51	open	Close	open
ไม่เชื่อม Interrupt Signal จาก RTC กับขาสัญญาณใดๆ	open	open	Close

ตารางที่ 3 ตารางแสดงการวางจัมเปอร์บนคอนเนคเตอร์ J14

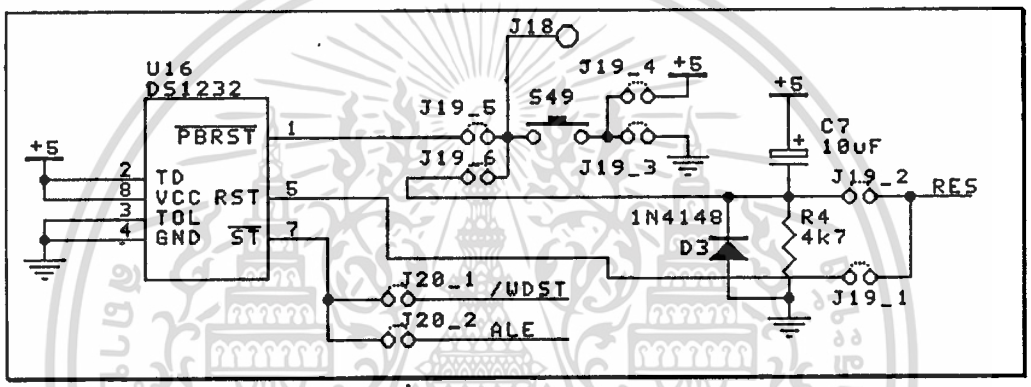
และจากวงจรในภาพที่ 3 และ 4 ที่ขารีเซต (RESET) ของ U1 (ขาที่ 9) สามารถเลือกวิธีการรีเซตตามตารางที่ 4 ได้ 2 วิธีคือ

1 ใช้ชิป DS1232 หลังจากกำหนดจัมเปอร์ตามตารางที่ 4 แล้ว ทุกครั้งที่มีไฟเลี้ยงเข้ามา (Power Up) ชิป DS1232 จะรอเวลาไประยะหนึ่งแล้วจึงส่งสัญญาณรีเซตซึ่งมีค่าทางตรรกเป็น '1' ผ่านขา RST , คอนเนคเตอร์ J19_1 , ไปยังขารีเซตของหน่วยประมวลผลกลาง ดังภาพที่ 4

จากนั้นหากผู้ใช้ต้องการรีเซตอีกครั้ง ก็สามารถกดปุ่มรีเซตเพื่อให้สัญญาณ ที่มีค่าตรรกเป็น '0' ผ่านจากกราวด์ , คอนเนคเตอร์ J19_3 (J19 คู่ที่3) , ปุ่มรีเซต , คอนเนคเตอร์ J19_5 ไปยังขา 1 ของ U16 หรือป้อนสัญญาณจากภายนอกบอร์ดผ่านคอนเนคเตอร์ J18 , คอนเนคเตอร์ J19_5 ผ่านไปยังขา 1 ของ U16 เช่นกัน เพื่อให้ U16 สร้างสัญญาณรีเซตที่มีค่าตรรกเป็น '1' ไปยังขารีเซตของหน่วยประมวลผลกลางผ่านคอนเนคเตอร์ J19_1 ต่อไป

2 ไม่ใช้ชิป DS1232 หลังจากกำหนดจัมเปอร์ตามตารางที่ 4 แล้ว เมื่อไฟเริ่มจ่ายเข้ามาในระบบใหม่ จะมีสัญญาณรีเซตที่มีค่าตรรกเป็น '1' ผ่านมาจากไฟเลี้ยง +5 โวลท์ , C7 , และผ่านคอนเนคเตอร์ J19_2 ทำให้ชิปบนบอร์ดสามารถรีเซตได้ดอนเริ่มจ่ายไฟ

จากนั้นหากผู้ใช้ต้องการรีเซตอีกครั้ง ก็สามารถกดปุ่มรีเซตเพื่อให้สัญญาณ ที่มีค่าตรรกเป็น '1' ผ่านจากไฟเลี้ยง , คอนเนคเตอร์ J19_4 , ปุ่มรีเซต หรือป้อนสัญญาณจากภายนอกบอร์ดผ่านคอนเนคเตอร์ J18 ที่มีค่าตรรกเป็น '1' เพื่อให้สัญญาณรีเซตที่มีค่าตรรกเป็น '1' ถูกส่งผ่านคอนเนคเตอร์ J19_6 , คอนเนคเตอร์ J19_2 ไปยังขารีเซตของหน่วยประมวลผลกลางต่อไป



ภาพที่ 4 ภาพแสดงวงจรรีเซต

การใช้งาน	คู่ที่ 1	คู่ที่ 2	คู่ที่ 3	คู่ที่ 4	คู่ที่ 5	คู่ที่ 6
ใช้ DS1232	Close	open	Close	open	Close	open
ไม่ใช้ DS1232	open	Close	open	Close	open	Close

ตารางที่ 4 ตารางแสดงการวางจัมเปอร์บนคอนเนคเตอร์ J19

อย่างไรก็ตามในกรณีที่กำหนดจัมเปอร์บนคอนเนคเตอร์ J19 เพื่อเลือกใช้ชิป DS1232 หากไม่มีการส่งสัญญาณที่มีค่าตรรกเป็น '0' หรือสัญญาณสโตรบ (Strobe Signal) ไปให้ขา $\overline{ST}$ ภายในกำหนดเวลา ชิปดังกล่าวจะส่งสัญญาณรีเซตออกไปยังชิปอื่นๆ เพื่อสั่งให้เริ่มทำงานใหม่อีกครั้ง

โดยในการใช้งานหากต้องการใช้ DS1232 ตรวจสอบความถูกต้องของโปรแกรม ก็ให้ใส่จัมเปอร์บนคอนเนคเตอร์ J20_1 เพื่อให้โปรแกรม (Program) สามารถส่งสัญญาณ $\overline{WDST}$ เข้ามาที่ขา $\overline{ST}$ และเมื่อใดก็ตามที่โปรแกรมหลุดไปทำงานอื่นที่ไม่ต้องการ จนส่งสัญญาณ $\overline{WDST}$ ไม่ทัน ชิปดังกล่าวก็จะสร้างสัญญาณรีเซตขึ้นมาเพื่อเริ่มทำงานใหม่ [สัญญาณ $\overline{WDST}$ ได้จากการเขียน

โปรแกรมให้ติดต่อกับพอร์ตอินพุต/เอาต์พุต (Input/Output) บางตำแหน่งซึ่งรายละเอียดจะอยู่ในเรื่องส่วนถอดรหัสอินพุต/เอาต์พุต

แต่หากไม่ต้องการเขียนโปรแกรมเพื่อส่งสัญญาณออกมาตลอดเวลา ก็ให้ใส่จัมเปอร์บนคอนเนคเตอร์ J20_2 เพื่อให้สัญญาณจาก ALE ของหน่วยประมวลผลกลางที่มีพัลส์ (Pulse) ตลอดเวลาถูกส่งเข้ามาแทน ชิปดังกล่าวก็จะไม่มีโอกาสสร้างสัญญาณรีเซตเนื่องจากเหตุดังกล่าวได้อีก

การใช้งาน	คู่ที่ 1	คู่ที่ 2
ส่งสัญญาณให้ขา $\overline{ST}$ ด้วยคำสั่งทางซอฟแวร์	Close	open
ส่งสัญญาณให้ขา $\overline{ST}$ ด้วยสัญญาณ ALE	open	Close

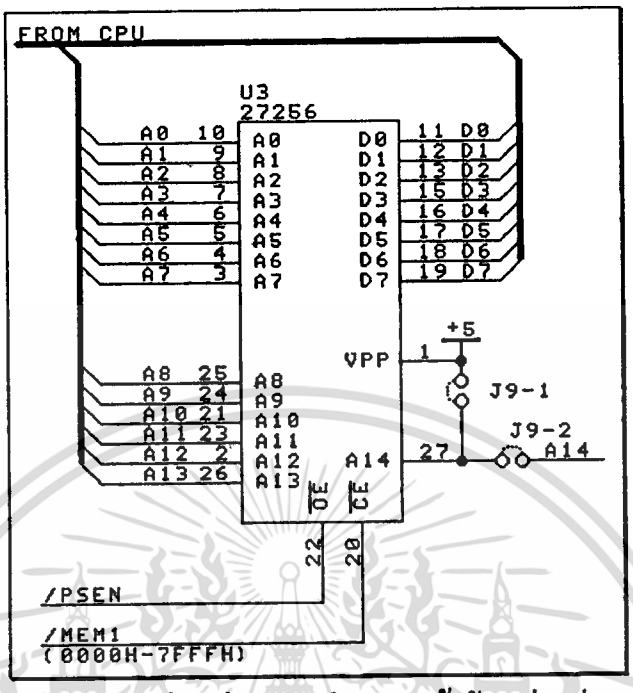
ตารางที่ 5 ตารางแสดงการวางจัมเปอร์บนคอนเนคเตอร์ J20

หมายเหตุ รายละเอียดเพิ่มเติมของชิป DS1232 สามารถดูได้จากภาคผนวก ง

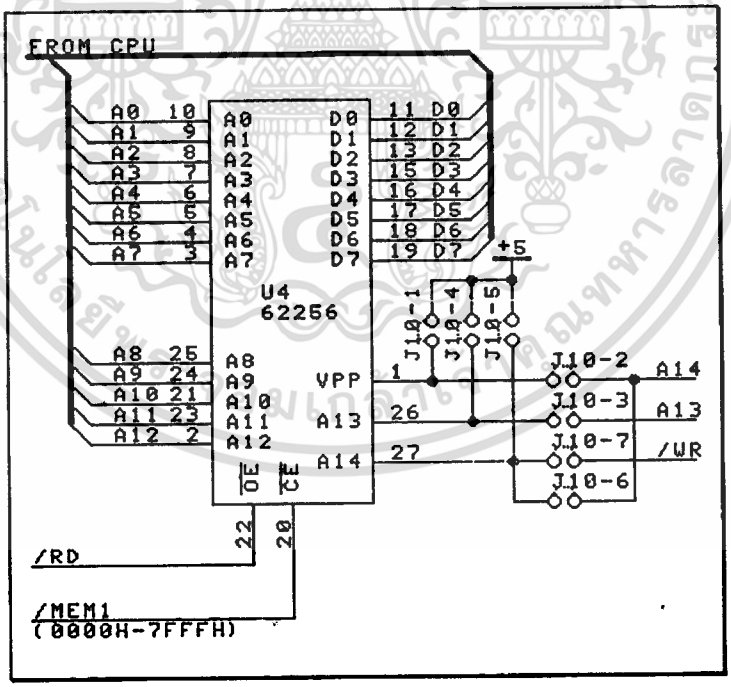
3.1.1.2 การจัดสรรหน่วยความจำ

จากภาพที่ 5, 6, และ 7 เป็นภาพแสดงวงจรอินเตอร์เฟส (Interface) ระหว่างหน่วยประมวลผลกลางกับหน่วยความจำ โดยหน่วยความจำถูกแบ่งออกเป็น 2 ส่วนตามตำแหน่งแอดเดรส (Address) คือ ช่วงตำแหน่ง 0000h - 7FFFh และตำแหน่ง 8000h - FEFFh

- ส่วนที่ 1 ตำแหน่งช่วง 0000h-7FFFh ซึ่งได้แก่ U3 และ U4 ดังในภาพที่ 5 และ 6 ตามลำดับ โดย U3 จะต่อขา $\overline{OE}$ เข้ากับขา $\overline{PSEN}$ ของหน่วยประมวลผลกลาง ซึ่งหมายความว่า U3 เป็นหน่วยความจำโปรแกรม (Code Memory) การเลือกขนาดของ U3 สามารถกำหนดได้ตามต้องการ ดังที่แสดงไว้ในตารางที่ 6



ภาพที่ 5 ภาพแสดงการต่อหน่วยความจำ U3 เข้ากับหน่วยประมวลผลกลาง



ภาพที่ 6 ภาพแสดงการต่อหน่วยความจำ U4 เข้ากับหน่วยประมวลผลกลาง

ชนิดและขนาดของ U3	คู่ที่ 1	คู่ที่ 2
2764 (8K)	Close	open
27128 (16K)	Close	open
27256 (32K)	open	Close

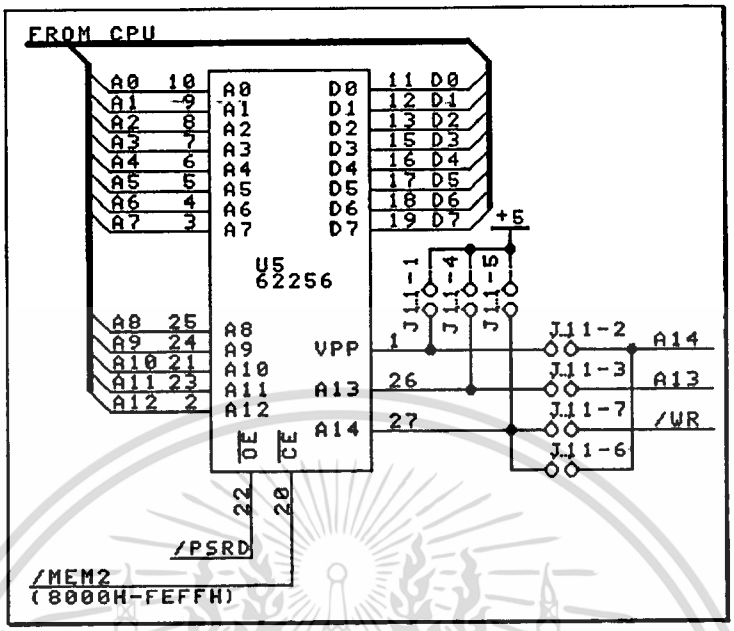
ตารางที่ 6 ตารางแสดงการวางจัมเปอร์บนคอนเนคเตอร์ J9 เพื่อเลือกชนิดและขนาดของ U3

ส่วน U4 ขา $\overline{OE}$ จะเชื่อมเข้ากับขา $\overline{RD}$ ของ หน่วยประมวลผลกลางโดยทำหน้าที่เป็นหน่วยความจำข้อมูล (data memory) โดยในการเลือกชนิดและขนาดของ U4 ว่าเป็น ROM หรือ RAM ขนาดเท่าใดนั้นจะมีการกำหนดผ่านจัมเปอร์บนคอนเนคเตอร์ J10 ตามที่ได้แสดงไว้ในตารางที่ 7

ชนิดและขนาด	วิธีการกำหนดจัมเปอร์						
ของ U4,U5	คู่ที่ 1	คู่ที่ 2	คู่ที่ 3	คู่ที่ 4	คู่ที่ 5	คู่ที่ 6	คู่ที่ 7
SRAM							
6264	Close	open	open	Close	open	open	Close
62256	open	Close	Close	open	open	open	Close
EPROM							
2764	Close	open	open	Close	Close	open	open
27128	Close	open	Close	open	Close	open	open
27256	Close	open	Close	open	open	Close	open

ตารางที่ 7 ตารางแสดงการวางจัมเปอร์บนคอนเนคเตอร์ J10 เพื่อเลือกชนิดของ U4 และคอนเนคเตอร์ J11 เพื่อเลือกชนิดของ U5

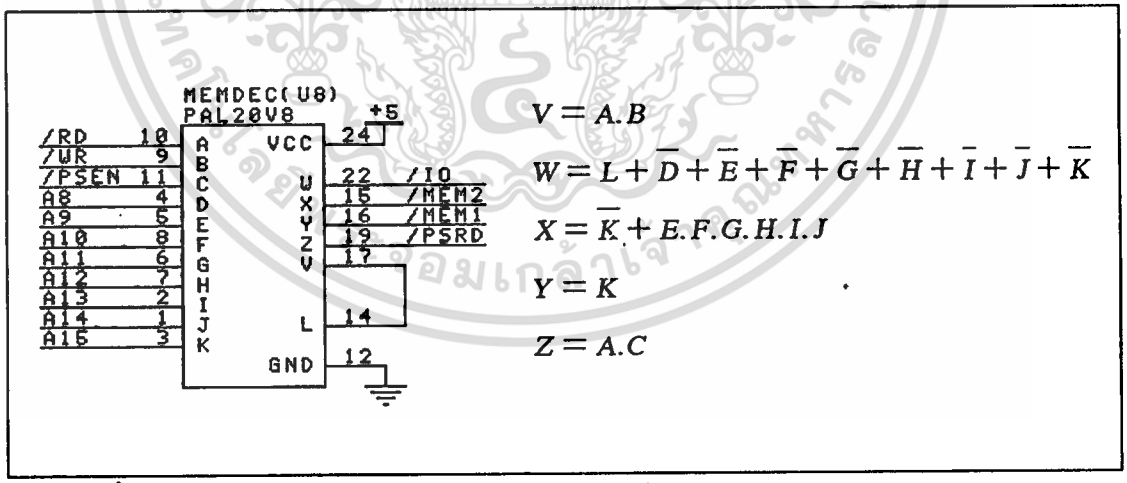
- <u>ส่วนที่ 2</u> ตำแหน่งช่วง 8000h-FFFFh ซึ่งก็คือ U5 ดังในภาพที่ 7 โดยที่ขา $\overline{OE}$ ของ U5 ต่อกับสัญญาณ $\overline{PSRD}$ ที่ได้จากการ AND กันระหว่าง $\overline{PSEN}$ และ $\overline{RD}$ บน PAL20V8 ซึ่งเป็นเกตโปรแกรมได้ดังในภาพที่ 8 ทำให้ U5 เป็นได้ทั้งหน่วยความจำโปรแกรม (code memory) และหน่วยความจำข้อมูล (data memory) และเช่นเดียวกับ U3 และ U4 ผู้ใช้สามารถเลือกชนิดและขนาดของ U5 ด้วยการกำหนดจัมเปอร์บนคอนเนคเตอร์ J11 ตามตารางที่ 7 เช่นเดียวกับคอนเนคเตอร์ J10 ของ U4



ภาพที่ 7 ภาพแสดงการต่อหน่วยความจำ U5 เข้ากับหน่วยประมวลผลกลาง

3.1.1.3 ส่วนถอดรหัสตำแหน่งแอดเดรส (Address)

◊ ส่วนถอดรหัสหลัก



ภาพที่ 8 ภาพแสดงขาสัญญาณ และสมการของ PAL20V8 ส่วนถอดรหัสแอดเดรสหลัก

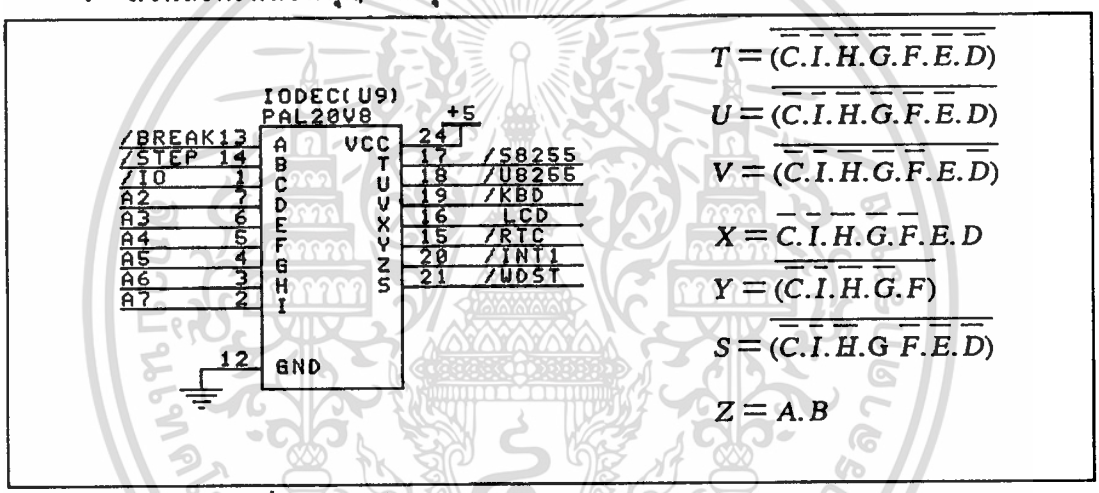
จากภาพที่ 8 จะแสดงส่วนถอดรหัสตำแหน่งหลัก โดยใช้อุปกรณ์ที่เรียกว่า PAL เบอร์ 20V8 ซึ่งเป็นเกตที่สามารถโปรแกรมได้ (Programable Gate) มาใช้ในการถอดรหัส โดยมีสมการและขาสัญญาณที่ใช้งานดังแสดงในภาพที่ 8 เพื่อทำหน้าที่เลือกระหว่าง U3,U4,U5 และอุปกรณ์อินพุต/เอาต์พุต โดย

จะเลือก U3 และ U4 ($\overline{MEM1}$ มีลอจิกเป็น '0') เมื่อมีการใช้งานแอดเดรสในตำแหน่ง 0000h ถึง 7FFFh (32 กิโลไบต์)

หรือเลือก U5 ($\overline{MEM2}$ มีลอจิกเป็น '0') เมื่อมีการติดต่อกับแอดเดรสในตำแหน่ง 8000h ถึง FEFFh (31.75 กิโลไบต์)

หรือ เลือกอุปกรณ์อินพุต/เอาต์พุต ($\overline{IO}$ มีค่าตรรกเป็น '0') เมื่อมีการติดต่อกับแอดเดรสในตำแหน่ง FF00h ถึง FFFFh (256 ไบต์) และมีสัญญาณอ่าน/เขียน ตัวใดตัวหนึ่งทำงาน ($\overline{RD}$ หรือ $\overline{WR}$ มีค่าตรรกเป็น '0')

◊ ส่วนถอดรหัสอินพุต/เอาต์พุต



ภาพที่ 9 ภาพแสดงขาสัญญาณ และสมการของ PAL20V8
ส่วนถอดรหัสแอดเดรสอินพุต/เอาต์พุต

จากภาพที่ 9 หลังจากที่ได้สัญญาณ $\overline{MEM1}$, $\overline{MEM2}$ และ $\overline{IO}$ ที่ถอดรหัสมาจากส่วนถอดรหัสหลักแล้ว หากสัญญาณที่ทำงานในขณะนั้นเป็น $\overline{IO}$ ($\overline{IO}$ มีลอจิกเป็น '0' ส่วน $\overline{MEM1}$ และ $\overline{MEM2}$ มีลอจิกเป็น '1') จะต้องมีการถอดรหัสอีกครั้งหนึ่งในรายละเอียดว่าเป็น IO ตัวใดที่หน่วยประมวลผลกลางต้องการติดต่อด้วย โดยจะใช้ PAL20V8 (IODEC) อีกตัวหนึ่งในการถอดรหัสตำแหน่งดังแสดงในภาพที่ 9 โดยอุปกรณ์อินพุต/เอาต์พุต แต่ละตัวจะมีตำแหน่งการใช้งาน (Address) ดังแสดงในตารางที่ 8

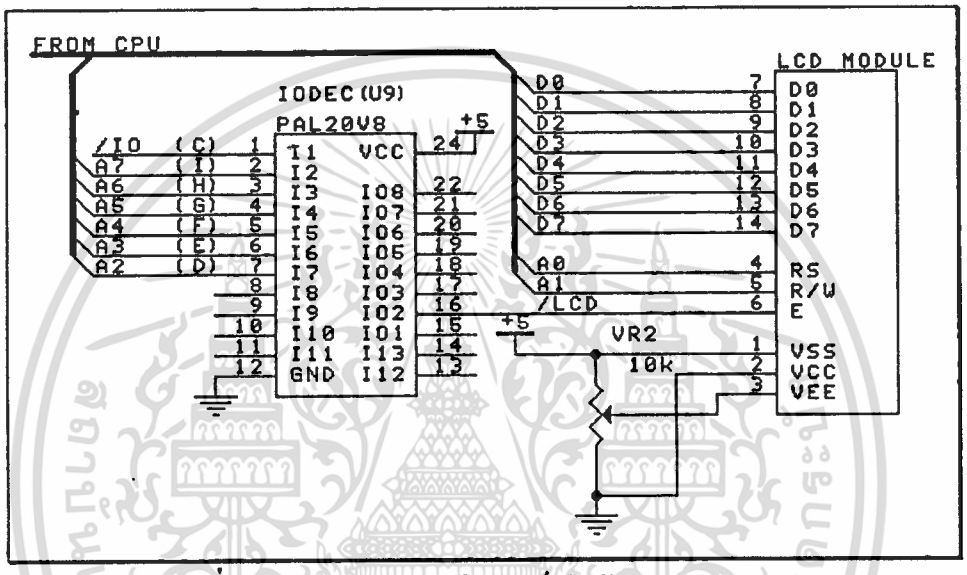
<u>หมายเหตุ</u> สำหรับขาสัญญาณ 20 ($\overline{INT1}$) ในภาพที่ 9 เป็นสัญญาณที่เกิดจากการ AND กันระหว่างสัญญาณ $\overline{BREAK}$ จากสวิทช์ และสัญญาณ $\overline{STEP}$ จากพอร์ต C.7 ของ System 8255 เพื่อสร้างสัญญาณอินเตอร์รัปให้กับขา $\overline{INT1}$ ของหน่วยประมวลผลกลางเพื่อแทนแอนด์เกต (AND Gate) (U99) ในภาพที่ 3 ไม่ใช่สัญญาณเพื่อเลือกอุปกรณ์อินพุต/เอาต์พุต แต่ประการใด

คำอธิบาย	ชื่อพอร์ท	ตำแหน่งเรียกใช้งาน
8255#1 (ระบบ)		
• พอร์ต A	S8255A	FF00h
• พอร์ต B	S8255B	FF01h
• พอร์ต C	S8255C	FF02h
• พอร์ตควบคุม	S8255D	FF03h
8255#2 (ผู้ใช้)		
• พอร์ต A	U8255A	FF04h
• พอร์ต B	U8255B	FF05h
• พอร์ต C	U8255C	FF06h
• พอร์ตควบคุม	U8255D	FF07h
แลทช์(Latch) เลือกกลุ่มคีย์	KBD	FF08h-FF10h
อุปกรณ์แสดงผลแบบผลึก		
เหลว (LCD Module)		
• พอร์ตเขียนคำสั่ง	LCDWI	FF0Ch
• พอร์ตเขียนข้อมูล	LCDWD	FF0Dh
• พอร์ตอ่านสถานะ	LCDRI	FF0Eh
• พอร์ตอ่านข้อมูล	LCDRD	FF0Fh
ชิปฐานเวลาจริง	RTC	FF10h-FF1Fh
DS1232	WDST	FF20h-FF23h

ตารางที่ 8 ตารางแสดงตำแหน่งแอดเดรสของอุปกรณ์ต่างๆ

3.1.1.4 ส่วนแสดงผล

ส่วนแสดงผลเป็นเอาท์พุตที่สำคัญของซิงเกิลบอร์ดในการแสดงผลลัพธ์ต่างๆที่เกิดจากการทำงานให้กับผู้ใช้ได้รับทราบ สำหรับอุปกรณ์ที่ใช้ในการแสดงผลจะใช้ อุปกรณ์แสดงผลแบบผลึกเหลว LCD (Liquid Crystal LCD) ทั้งนี้เนื่องจากมีข้อดีหลายประการเช่น กินไฟน้อย , แสดงตัวอักษร/ตัวเลขได้ , มีแลทช์(Latch) ในตัว เป็นต้น



ภาพที่ 10 ภาพแสดงวงจรอินเตอร์เฟสกับ LCD Module

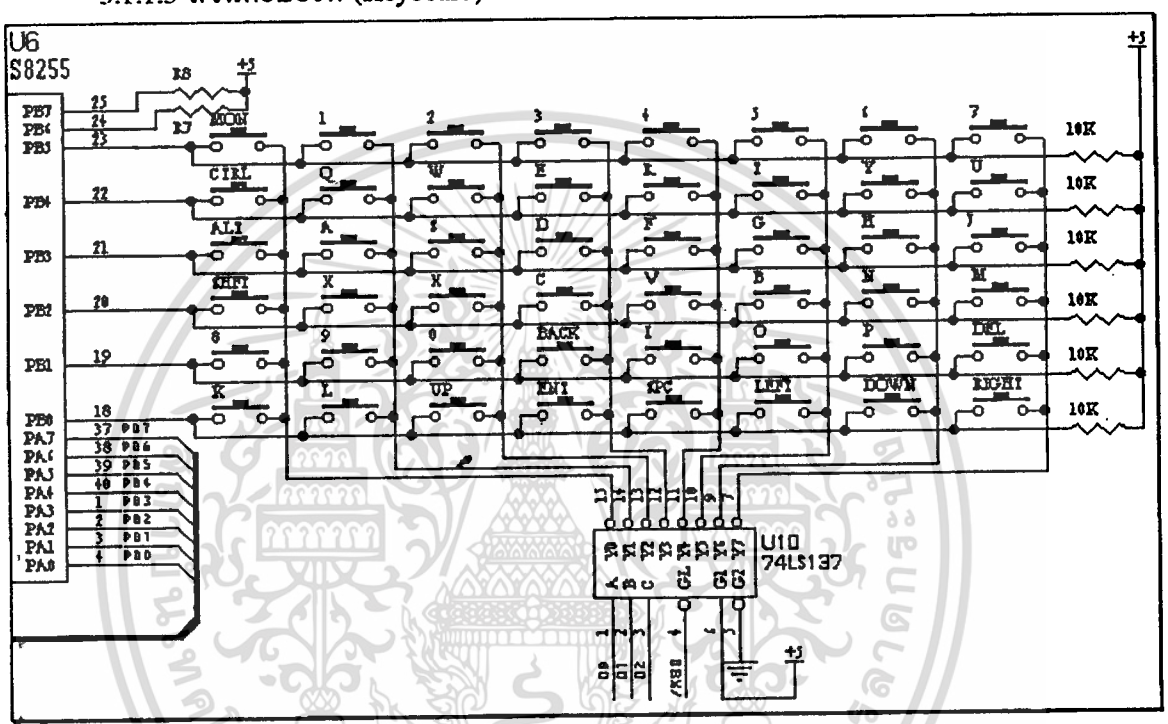
จากภาพที่ 10 เป็นภาพแสดงวงจรอินเตอร์เฟสกับ LCD โดยจะต่อสัญญาณ RS (Register Selection) และขาสัญญาณ $\overline{RD}/\overline{WR}$ ของ LCD เข้ากับขา A0 และ A1 ของหน่วยประมวลผลกลางตามลำดับ ส่วนที่ขา E (Enable) ของ LCD จะต่อกับสัญญาณเลือกพอร์ท FF0Ch - FF0Fh ซึ่งได้มาจากวงจรถอดรหัสอินพุต/เอาต์พุต โดยที่

- พอร์ทสำหรับเขียนคำสั่งบน LCD (Instruction Write) มีแอดเดรสอยู่ที่ FF0Ch
- พอร์ทสำหรับอ่านสถานะบน LCD (Instruction Read) มีแอดเดรสอยู่ที่ FF0Eh
- พอร์ทสำหรับเขียนข้อมูลบน LCD (Data Write) มีแอดเดรสอยู่ที่ FF0Dh
- พอร์ท์สำหรับอ่านข้อมูลบน LCD (Data Read) มีแอดเดรสอยู่ที่ FF0Fh ตามลำดับ ดังแสดงไปแล้วในตารางที่ 8

<u>หมายเหตุ</u> รายละเอียดเพิ่มเติมของการใช้ LCD สามารถดูได้จากภาคผนวก ก

สำหรับความชัดของตัวอักษรบน LCD จะขึ้นกับแรงดันที่ขา VEE ซึ่งสามารถทำได้โดยการปรับที่ตัวต้านทานปรับค่าได้แบบเกือกม้า (VR2) เพื่อกำหนดแรงดันที่ป้อนเข้า VEE ให้เป็นไปตามต้องการ

3.1.1.5 ส่วนคีย์บอร์ด (Keyboard)



ภาพที่ 11 ภาพแสดงวงจรอินเตอร์เฟส กับคีย์บอร์ดแบบ X-Y Matirx

คีย์บอร์ดเป็นอุปกรณ์อินพุตที่สำคัญที่สุดสำหรับซิงเกิลบอร์ดในการติดต่อรับคำสั่งจากผู้ใช้งาน จากภาพที่ 11 เป็นภาพแสดงวงจรอินเตอร์เฟสของหน่วยประมวลผลกลางกับคีย์บอร์ด (Keyboard) แบบเมตริกซ์ (X-Y matrix) กล่าวคือการอ้างตำแหน่งของคีย์ จะใช้ตัวเลขเพื่ออ้างตำแหน่ง 2 ตัว ทั้งนี้เพื่อใช้อ้างว่าคีย์ๆนั้นอยู่หลัก (Column) และแถว (Row) ใดคล้ายกับการอ้างตำแหน่งบนกราฟ X-Y โดยซิงเกิลบอร์ดในโครงงานนี้จะมีจำนวนแถวของคีย์อยู่ 6 แถว แต่ละแถวจะมีคีย์แถวละ 8 คีย์ ทำให้ซิงเกิลบอร์ดสามารถรองรับคีย์ได้ทั้งหมด 48 คีย์

โดยในระหว่างการทำงาน หน่วยประมวลผลกลางจะทำการสแกน (Scan) เพื่อเลือกกลุ่มของคีย์บอร์ดทีละหลัก (Column) โดยการเขียนค่าของหลักที่ต้องการมาที่พอร์ท $\overline{\text{KBD}}$ ผ่านไอซี 74LS137 (U10) ซึ่งเป็นไอซีถอดรหัส 3 เป็น 8 (3-to-8 Decoder) และเป็นแลทช์ (Latch) ในตัวเอง (เก็บค่าที่เขียนเข้ามาไว้ จนกว่าจะมีค่าใหม่เขียนเข้ามาทับของเดิม) ที่ตำแหน่ง FF08h เพื่อเลือก

หลักใดหลักหนึ่งในช่วง [0-8] และอ่านข้อมูลของคีย์ที่ถูกกดผ่านทางพอร์ท B ของ U6 (S8255) ว่าตรงกับแถวใดใน 6 แถว (0-5) โดยที่

- หากไม่มีคีย์ใดถูกกดเลยค่าที่อ่านค่าได้คือ 0FFh
- หากมีคีย์ใดคีย์หนึ่งในหลัก (Column) ที่ถูกเลือกถูกกดจะทำให้บิตใดบิตหนึ่งของพอร์ต B ของ U6 (PB0-PB5) ที่ต่ออยู่กับคีย์นั้นๆ กลายเป็น 0 ตัวอย่างเช่น

- หากหลักที่เลือกคือหลัก 0 และคีย์ที่ถูกกดคือ K ดังนั้น PB0 = 0
- หากหลักที่เลือกคือหลัก 0 และคีย์ที่ถูกกดไม่ใช่ K ดังนั้น PB0 = 1

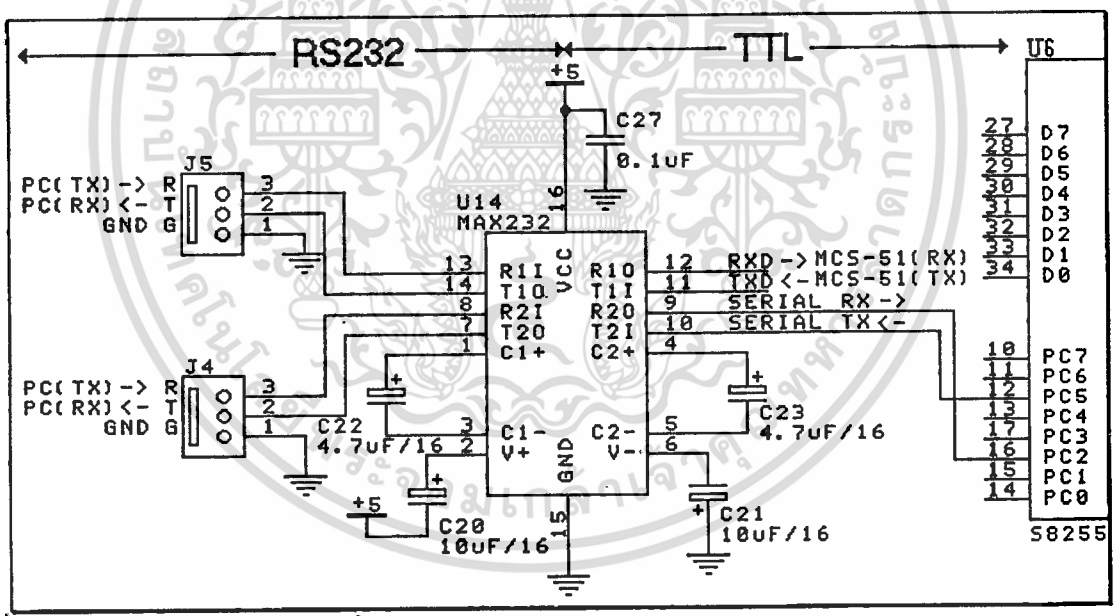
ภาพที่ 12 ภาพแสดงรายละเอียดหน้าปัดคีย์ของซิงเกิลบอร์ด

สำหรับภาพที่ 12 เป็นภาพแสดงหน้าปัดและตำแหน่งของคีย์ต่างๆที่ใช้งานจริง ซึ่งประกอบไปด้วยคีย์ตัวอักษร, ตัวเลข, สัญญลักษณ์, คีย์ลูกศร, คีย์ Ctrl, คีย์ ALT, คีย์ Enter ฯ โดยตำแหน่งการวางจะมีลักษณะคล้ายกับคีย์บอร์ดของคอมพิวเตอร์ส่วนบุคคลทั่วไป

3.1.1.6 ส่วนวงจรอินเตอร์เฟส RS-232

เป็นส่วนที่ใช้ในการสื่อสารข้อมูลกับอุปกรณ์ภายนอก เช่น คอมพิวเตอร์ส่วนบุคคล เพื่อใช้ในการส่ง (Upload) ,รับ (Download) ข้อมูล หรือคำสั่งกับคอมพิวเตอร์ภายนอก โดยที่จะสื่อสารจะมีลักษณะเป็นการสื่อสารแบบอนุกรมแทนที่จะเป็นแบบขนาน ทั้งนี้เป็นเพราะเหตุผลหลายๆ ประการเช่น จำนวนสายที่ใช้น้อยกว่า, สามารถส่งข้อมูลได้ไกลกว่า รวมทั้ง MCS-51 เองก็มีความสามารถสนับสนุนการสื่อสารแบบอนุกรมได้อยู่แล้ว โดยไม่ต้องใช้ชิปอื่นสนับสนุน

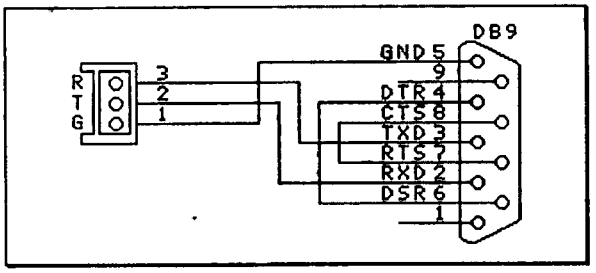
แต่อย่างไรก็ตามในการสื่อสารแบบอนุกรมส่วนมากโดยเฉพาะบนคอมพิวเตอร์ส่วนบุคคลที่จะใช้ในการติดต่อด้วย มักจะมีพอร์ตอนุกรมเป็นไปตามมาตรฐาน RS-232 ในขณะที่ MCS-51 ยังคงมีมาตรฐานเป็นทีทีแอล (TTL) จึงต้องมีอุปกรณ์บางตัวที่เรียกว่าไดรเวอร์ (Driver) เพื่อช่วยในการแปลงสัญญาณไปมาระหว่างทีทีแอล กับ RS-232 โดยอุปกรณ์ที่จะนำมาใช้เป็นไดรเวอร์ กับซิงเกิลบอร์ดนี้คือ MAX232 ซึ่งมีวงจรการใช้งานดังภาพที่ 13



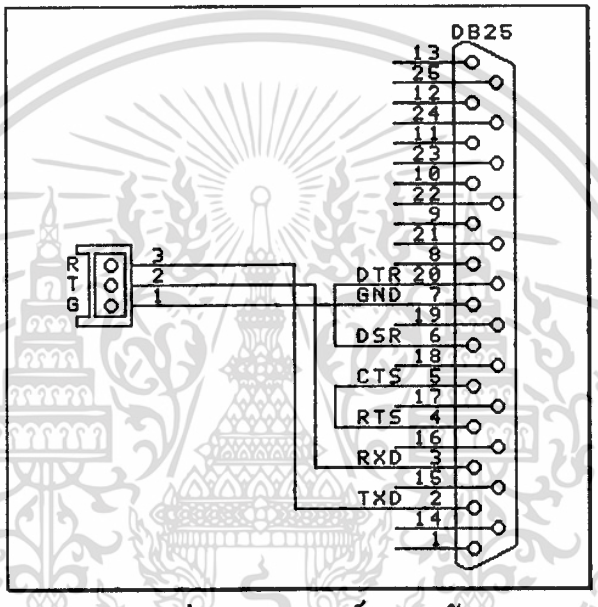
ภาพที่ 13 ภาพแสดงวงจรอินเตอร์เฟส RS-232

จากภาพที่ 13 หน่วยประมวลผลกลางสามารถจะรับส่งข้อมูลผ่านพอร์ตอนุกรมได้ 2 ทาง คือ

1. ผ่านทางขา TXD และ RXD ของ หน่วยประมวลผลกลางโดยตรง โดยเชื่อมผ่านคอนเนคเตอร์ (Connector) J5 ไปยังคอนเนคเตอร์ของอุปกรณ์ภายนอก
2. ผ่านทางพอร์ต C ของ U6 โดยรับผ่านพอร์ต C.2 และส่งผ่านพอร์ต C.5 โดยเชื่อมผ่านคอนเนคเตอร์ J4 ไปยังคอนเนคเตอร์ของอุปกรณ์ภายนอก



ภาพที่ 14 ภาพแสดงการต่อคอนเนคเตอร์ J4/J5 กับคอนเนคเตอร์ DB9



ภาพที่ 15 ภาพแสดงการต่อคอนเนคเตอร์ J4/J5 กับคอนเนคเตอร์ DB25

สำหรับภาพที่ 14 และ 15 เป็นภาพแสดงวงจรอินเตอร์เฟสกับอุปกรณ์ภายนอกที่มีคอนเนคเตอร์ของพอร์ตอนุกรมเป็นแบบ DB9 และแบบ DB25 ตามลำดับ

หมายเหตุ รายละเอียดเพิ่มเติมของ MAX232 สามารถดูได้จากภาคผนวก ค

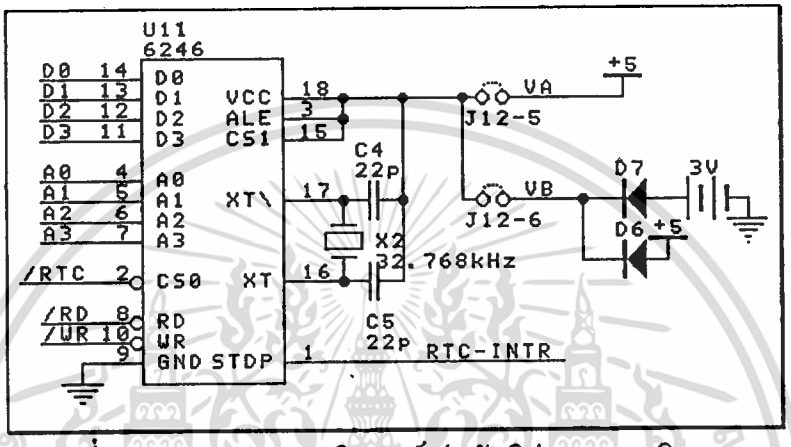
3.1.1.7 ส่วนวงจรสร้างฐานเวลาจริง RTC (Real Time Clock)

จากภาพที่ 16 เป็นภาพแสดงวงจรสร้างฐานเวลาจริงซึ่งเป็นส่วนที่ช่วยเสริมการทำงานของหน่วยประมวลผลกลางในการสร้างแและเก็บฐานของเวลาจริง ทั้งเวลา และ วัน/เดือน/ปี ในปัจจุบัน โดยไม่จำเป็นต้องใช้วงจรนาฬิกา (Timer) ของหน่วยประมวลผลกลาง เพื่อเก็บไปใช้กับงานอื่นๆต่อไป และยังช่วยลดความยุ่งยากในการเขียนโปรแกรม รวมทั้งประหยัดเวลาการทำงานของหน่วยประมวลผลกลาง

นอกจากนี้หากต่อแบตเตอรี่สำรองไว้ ตัวชิปฐานเวลาจริง (RTC) เองยังสามารถทำงานต่อไปได้ แม้หน่วยประมวลผลกลางจะหยุดทำงานลง แต่เมื่อหน่วยประมวลผลกลางกลับมา

ทำงานใหม่อีกครั้งก็ยังสามารถอ่านเวลาจริงกลับมาได้อย่างถูกต้องเช่นเดิม โดยรายละเอียดเกี่ยวกับระบบไปสำรองนั้นจะได้กล่าวในเรื่องภาคจ่ายไฟและระบบไฟสำรองต่อไป

สำหรับชิปฐานเวลาจริงที่ใช้กับซิงเกิลบอร์ดนั้นคือเบอร์ 6246 ซึ่งมีรายละเอียดของขาสัญญาณต่างๆ และการทำงานของวงจรดังนี้



ภาพที่ 16 ภาพแสดงวงจรอินเตอร์เฟสกับชิปฐานเวลาจริง 6246

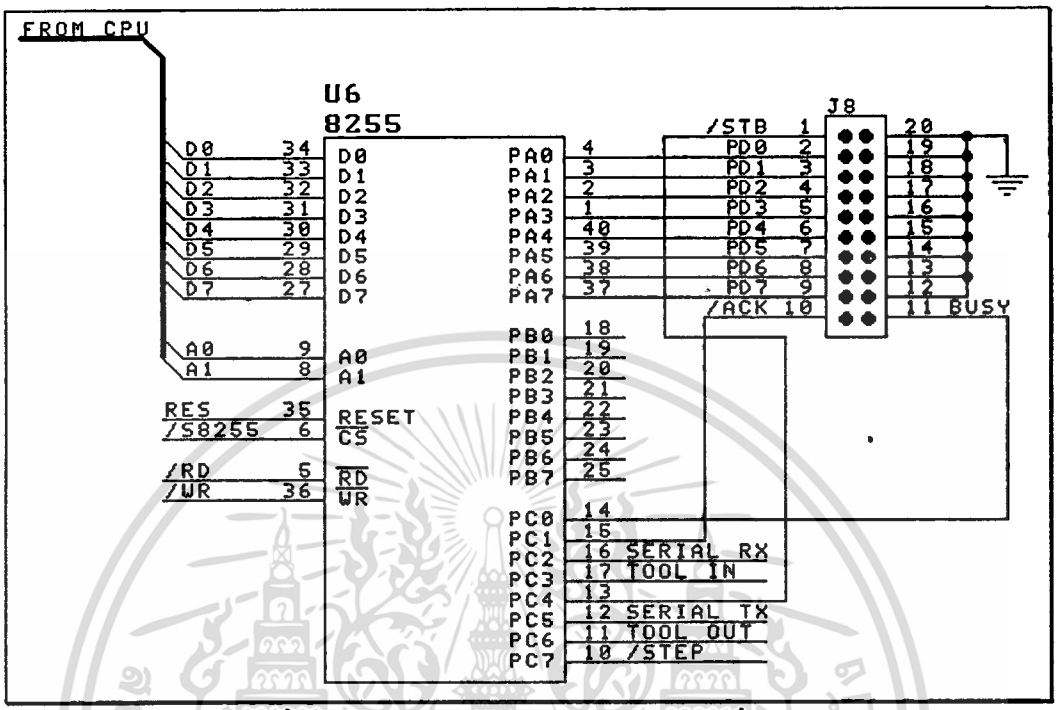
- ขา 1 STDP - เป็นขาสัญญาณเอาท์พุตที่ส่งสัญญาณอินเตอร์รัพท์ออกมาตามเวลาที่ได้กำหนดไว้เช่น ทุก 1 วินาที, ทุก 1 นาที ฯ โดยสามารถจะนำสัญญาณที่ได้นี้ไปเชื่อมกับขา T0 หรือ INT0 ของหน่วยประมวลผลกลาง ได้โดยการกำหนดจัมเปอร์ตามตารางที่ 3 ที่ได้กล่าวไปแล้ว
- ขา 2 CS0 - เป็นขาอินพุตที่ต่อเข้ากับ $\overline{RTC}$ ที่ได้มาจากวงจรถอดรหัสตำแหน่งแอดเดรสของพอร์ตอินพุต/เอาต์พุต ในภาพที่ 9 เพื่อดูว่าหน่วยประมวลผลกลางต้องการอ่าน,เขียน ข้อมูลบนชิปฐานเวลาจริงหรือไม่ เพื่อจะได้ทำการเลือกวิปฐานเวลาจริงได้ถูกต้อง โดยที่

  * หากหน่วยประมวลผลกลาง ต้องการติดต่อกับชิปฐานเวลาจริง สัญญาณ $\overline{RTC}$ จะมีค่าตรรกเป็น '0' (กรณี CS1 ทำงานเป็นปกติคือมีค่าตรรกเป็น '1') ชิปฐานเวลาจริงจะถูกเลือกใช้
  * หากหน่วยประมวลผลกลาง ไม่ต้องการติดต่อกับชิปฐานเวลาจริง สัญญาณ $\overline{RTC}$ จะมีค่าตรรกเป็น '1' ชิปฐานเวลาจริงจะไม่ถูกเลือกใช้งาน
- ขา 15 CS1 - มีหน้าที่คล้ายกับขา CS0 คือจะทำการเลือกใช้งานของชิปฐานเวลาจริงเช่นกัน โดยจะต่อเข้ากับไฟเลี้ยงไว้ตลอดเวลา

- ขา 4-7 A0-A3 - หากชิปฐานเวลาจริงได้รับการเรียกใช้งานจากสัญญาณ $\overline{RTC}$ แล้วจะต้องนำสัญญาณ A0-A3 จากหน่วยประมวลผลกลางมาตรวจสอบอีกครั้งว่า หน่วยประมวลผลกลางต้องการติดต่อกับรีจิสเตอร์ตัวใดของ RTC ใน 16 ตัว
- ขา 8 RD - มีหน้าที่การทำงานเช่นเดียวกับขา $\overline{RD}$ ในไอซีหน่วยความจำ
- ขา 10 WR - มีหน้าที่การทำงานเช่นเดียวกับขา $\overline{WR}$ ในไอซีหน่วยความจำ
- ขา 16,17 XT - ต่อกับคริสตอลภายนอกเพื่อสร้างความถี่
- ขา 11-14 D0-D3 - เป็นบัสข้อมูลของชิปฐานเวลาจริง
- ขา 18 VCC - จะต่อกับไฟเลี้ยงและแบตเตอรีสำรองซึ่งจะได้กล่าวในเรื่องระบบไปสำรองต่อไป
- ขา 9 GND - ต่อกับกราวด์ของระบบ

<u>หมายเหตุ</u> รายละเอียดเพิ่มเติมของชิปฐานเวลาจริง 6246 สามารถดูได้จากภาคผนวก ข

3.1.1.8 ส่วนอินเตอร์เฟสกับเครื่องพิมพ์



ภาพที่ 17 ภาพแสดงวงจรอินเตอร์เฟสกับเครื่องพิมพ์

จากภาพที่ 17 จะแสดงรายละเอียดวงจรอินเตอร์เฟสกับเครื่องพิมพ์ (Printer) โดยจะติดต่อผ่านทาง S8255 (U6) โดยมีรายละเอียดดังนี้

- พอร์ท A ต่อเข้ากับบัสข้อมูลของคอนเนคเตอร์เครื่องพิมพ์

- พอร์ท C ส่วนบนซึ่งจะถูกโปรแกรมกำหนดให้เป็นเอาท์พุตในการทำงานโหมด 0 จะใช้บิตที่ 4 ในการส่งสัญญาณ $\overline{\text{STB}}$ (Strobe) ให้แก่เครื่องพิมพ์

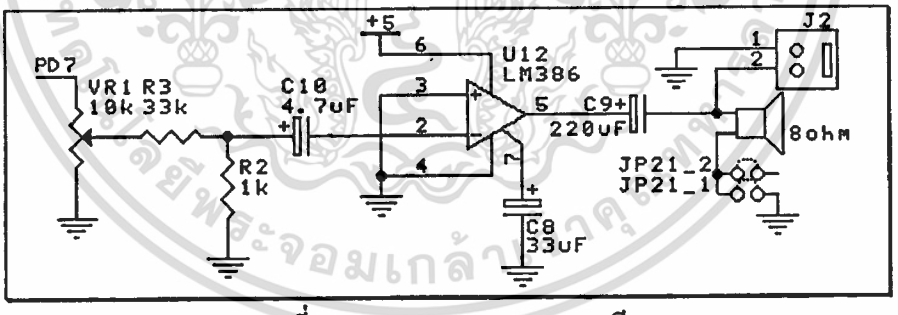
- พอร์ต C ส่วนล่างซึ่งจะถูกโปรแกรมให้เป็นอินพุต จะใช้บิตที่ 0 เพื่ออ่านสถานะความพร้อมของเครื่องพิมพ์ (BUSY) กลับมา และใช้บิตที่ 1 เพื่ออ่านสัญญาณตอบรับ $\overline{\text{ACK}}$ (Acknowledge) กลับมาจากเครื่องพิมพ์

แต่อย่างไรก็ตามคอนเนคเตอร์ที่ใช้ต่อกับเครื่องพิมพ์นั้นจะมีมาตรฐานการเชื่อมต่อโดยเฉพาะนั่นคือโดยทั่วไปมักจะเป็นแบบหัวต่อแบบเซนทรอนิค 36 ขา (Centronic 36 pin) โดยในตารางที่ 9 จะแสดงวิธีการต่อสายจากคอนเนคเตอร์ J8 ออกไปยังคอนเนคเตอร์ของเครื่องพิมพ์ที่เป็นแบบเซนทรอนิค

คอนเนคเตอร์ J8		เส้นทางข้อมูล	เซนทรอนิค 36 ขา
$\overline{STB}$	1	⇒	1
D0	2	⇒	2
D1	3	⇒	3
D2	4	⇒	4
D3	5	⇒	5
D4	6	⇒	6
D5	7	⇒	7
D6	8	⇒	8
D7	9	⇒	9
$\overline{ACK}$	10	⇐	10
BUSY	11	⇐	11
GND	12-20	⇔	27-19

ตารางที่ 9 ตารางแสดงการเชื่อมต่อระหว่างคอนเนคเตอร์ J8 และคอนเนคเตอร์เซนทรอนิค

3.1.1.9 ระบบเสียง



ภาพที่ 18 ภาพแสดงวงจรขยายเสียง

จากภาพที่ 18 ระบบเสียงของซิงเกิลบอร์ดเป็นระบบที่สร้างเสียงด้วยการให้หน่วยประมวลผลกลางสร้างพัลส์เป็นคลื่นรูปสี่เหลี่ยม (Square Wave) ที่มีความถี่สูงผ่านขาสัญญาณ PD7 (พอร์ต A.6) ขนาด 1 บิตของ S8255 หรือ U6 ซึ่งใช้ร่วมกับบัสข้อมูล (Data Bus) ของเครื่องพิมพ์ในภาพที่ 17 และนำไปผ่านชิป LM386 ซึ่งเป็นวงจรปรีแอมป์ (Pre Amp.) เพื่อขยายกำลังของสัญญาณ (Amplitude) อีกครั้งหนึ่ง

สำหรับการปรับระดับความดังของเสียงที่ออกจากลำโพง จะปรับด้วย VR1 (ความต้านทานปรับค่าได้) ซึ่งจะช่วยเปลี่ยนระดับแรงดันของพัลส์ (Pulse) ให้ลดลงมากน้อยตามต้องการก่อนที่จะผ่านเข้าสู่ปรีแอมป์เพื่อนำไปขยายและผ่านเข้าสู่ลำโพงต่อไป

และในกรณีไม่ต้องการใช้ลำโพงในระหว่างใช้เครื่องพิมพ์ ก็สามารถกำหนดจัมเปอร์บนคอนเนคเตอร์ J21 ตามตารางที่ 10 โดยการวางจัมเปอร์บนคู่ที่ 2 ของคอนเนคเตอร์ J21 ทำให้ขาลำโพงไม่ได้ต่อลงกราวด์

การใช้งาน	คู่ที่ 1	คู่ที่ 2
ใช้ลำโพง	Close	open
ไม่ใช้ลำโพง	open	Close

ตารางที่ 10 ตารางแสดงการวางจัมเปอร์บนคอนเนคเตอร์ J21

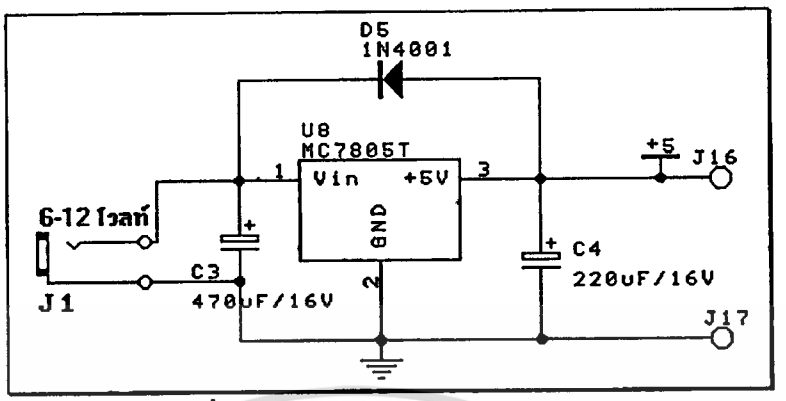
3.1.1.10 พอร์ตใช้งานทั่วไป

จากตารางที่ 13 และภาพที่ 24 สำหรับผู้ใช้งานที่ต้องการนำซิงเกิลบอร์ดไปใช้ควบคุม, ติดต่อกับอุปกรณ์อินพุต/เอาต์พุต อื่นๆ นอกเหนือจากที่มีอยู่ ก็จะมีพอร์ตว่างสำหรับให้ผู้ใช้นำไปใช้ในงานอื่นๆได้อีกดังต่อไปนี้

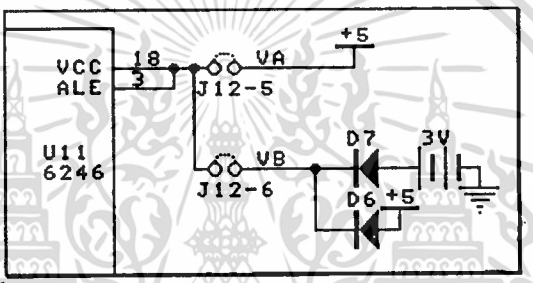
- พอร์ต 1 ของหน่วยประมวลผลกลาง (U1) ผ่านคอนเนคเตอร์ J6
- พอร์ต A-C บน U8255 (U7) ผ่านคอนเนคเตอร์ J7
- พอร์ต C บน S8255 (U6) ในบิต PC.3(Tool In) และ PC.6 (Tool Out) ผ่านคอนเนคเตอร์ J3 (อ้างเป็นบิต)

3.1.1.11 ภาคจ่ายไฟและระบบไฟสำรอง

จากภาพที่ 19 ภาคจ่ายไฟจะใช้ไอซีเรกกูเลเตอร์ (Regulator) เบอร์ 7805 ทำหน้าที่รักษาระดับแรงดันของวงจรให้อยู่ที่ 5 โวลท์ โดยรับไฟกระแสตรง (Direct Current) 6-12 โวลท์ จากเพ้าเวอร์ซัพพลาย (Power Suplay) ภายนอกเช่นอะแดปเตอร์ (Adaptor) ผ่านดีซีแจ็ก (DC Jack) J1 , ไอซีเรกกูเลเตอร์ 7805 ที่ขาอินพุต (ขา 1) สำหรับกรณีผู้ใช้ต้องการป้อนแรงดัน +5 โวลท์ เข้าบอร์ดโดยตรงสามารถต่อแรงดัน +5 โวลท์ ผ่านคอนเนคเตอร์ J16 และกราวน์ผ่านคอนเนคเตอร์ J17 ได้โดยมีไดโอด D5 เป็นตัวช่วยลดแรงดันลงเมื่อเกิดกรณีป้อนแรงดันเกินกว่า +5 โวลท์ ผ่าน J16 โดยไม่ตั้งใจ สำหรับ C3 และ C4 นั้นจะทำหน้าที่เป็นตัวกรองแรงดัน DC ให้เรียบขึ้นอีกครั้ง



ภาพที่ 19 ภาพแสดงวงจรของภาคจ่ายไฟ



ภาพที่ 20 ภาพแสดงระบบไฟสำรองของชิปฐานเวลาจริง

จากภาพที่ 20 ในขณะที่ภาคจ่ายไฟหลักหยุดจ่ายกระแส ข้อมูลต่างๆที่อยู่ในแรม และรีจิสเตอร์ (Register) ในชิปฐานเวลาจริงก็จะสูญหายไปด้วย ดังนั้นระบบไฟสำรองจึงถูกนำมาใช้ในการสำรองไฟเลี้ยงให้กับหน่วยความจำที่เป็นแรมเพื่อทำการรักษา (Backup) ข้อมูล และรักษาฐานเวลาจริงในชิปฐานเวลาจริงให้ดำเนินต่อไป โดยขณะที่ภาคจ่ายไฟหลักหยุดทำงานลง กระแสที่ไหลผ่าน D7 จะหยุดลงเช่นกัน แต่จะมีกระแสจากแบตเตอรีไหลผ่าน D6 มากขึ้นเพื่อชดเชยกระแสไฟจากภาคจ่ายไฟหลัก ซึ่งพอเพียงที่เลี้ยงให้กับแรมและ ชิปฐานเวลาจริง เพราะตามปกติอุปกรณ์เหล่านี้จะกินกระแสน้อยมาก

แต่อย่างไรก็ตามแรม และชิปฐานเวลาจริง แต่ละตัวสามารถกำหนดได้ว่าจะใช้ระบบไฟสำรองหรือไม่ตามต้องการ ด้วยการกำหนดจัมเปอร์บนคอนเนคเตอร์ J12 ดังแสดงในตารางที่ 11

ความหมาย	U4		U5		RTC	
	คู่ที่ 1	คู่ที่ 2	คู่ที่ 3	คู่ที่ 4	คู่ที่ 5	คู่ที่ 6
U4,U5,RTC ไม่ใช้ไฟสำรอง	Close	open	Close	open	Close	open
U4 ใช้ไฟสำรอง	open	Close	Close	open	Close	open
U5 ใช้ไฟสำรอง	Close	open	open	Close	Close	open
RTC ใช้ไฟสำรอง	Close	open	Close	open	open	Close
U4,U5 ใช้ไฟสำรอง	open	Close	open	Close	Close	open
U4,RTC ใช้ไฟสำรอง	open	Close	Close	open	open	Close
U5,RTC ใช้ไฟสำรอง	Close	open	open	Close	open	Close

ตารางที่ 11 ตารางแสดงการการวางจัมเปอร์บนคอนเนคเตอร์ J12 เพื่อเลือก / ไม่เลือกใช้ระบบไฟสำรองของแรม และชิปฐานเวลาจริง

3.1.2) การสร้างวงจร

เมื่อได้รายละเอียดของวงจรทั้งหมดแล้ว ดังในภาพที่ 21 ขั้นต่อไปก็จะเข้าสู่ขั้นตอนการออกแบบลายวงจรพิมพ์และทำแผ่นวงจรพิมพ์ PCB (Print Circuit Board) ของซิงเกิลบอร์ด ดังในภาพที่ 22 และ 23 เป็นภาพแสดงรายละเอียดของลายวงจรพิมพ์ด้านบนและด้านล่างตามลำดับ สำหรับภาพที่ 24 และ 25 จะเป็นภาพแสดงตำแหน่งการวางอุปกรณ์บนแผ่นวงจรพิมพ์และซิลค์สกรีน (Silk Screen) ตามลำดับ โดยรายการอุปกรณ์ต่างๆได้แสดงไว้ดังตารางที่ 12 และ สำหรับตารางที่ 13 จะเป็นสรุปการใช้งานคอนเนคเตอร์ต่างๆที่อยู่บนซิงเกิลบอร์ดซึ่งสัมพันธ์กับภาพที่ 24

อุปกรณ์	รูปแบบ	จำนวน	รหัส	หมายเหตุ
1) ไอซี (IC)				
80C31	DIP40	1	U1	หน่วยประมวลผลกลาง
74LS373	DIP20	1	U2	แอดเดรสแลทช์
27256	DIP28	1	U3	มอนิเตอร์อีพรอม (Monitor Eprom)

ตารางที่ 12 ตารางแสดงรายการอุปกรณ์

อุปกรณ์	รูปแบบ	จำนวน	รหัส	หมายเหตุ
62256	DIP28	2	U4	แรมของระบบ
			U5	แรมของผู้ใช้
8255	DIP40	2	U6	8255 ของระบบ
			U7	8255 ของผู้ใช้
PAL20V8	DIP24N	2	U8	ถอดรหัสตำแหน่งหน่วยความจำ
			U9	ถอดรหัสตำแหน่งอุปกรณ์อินพุต/เอาต์พุต
74LS374	DIP16	1	U10	แลตช์เก็บหลักของคีย์ที่เลือก
6264	DIP18	1	U11	ชิปฐานเวลาจริง
LM386	DIP8	1	U12	ปรีแอมป์ (Pre. Amp) ขยายเสียง
DMC204	LCD20X4	1	U13	อุปกรณ์แสดงผลแบบผลึกเหลว
MAX232	DIP16	1	U14	RS232
7805	TO-220	1	U15	เรกกูเลเตอร์ +5 โวลท์
DS1232	DIP8	1	U16	ไมโครมอนิเตอร์ชิปMicro Monitor Chip
2) ไดโอด (Diode)				
1N4001	DIODE0.4	1	D5	เรกกูเรเตอร์
1N4148	DIODE0.3	3	D3	รีเซต
			D6,D7	แบตเตอรีสำรอง

ตารางที่ 12 (ต่อ)

อุปกรณ์	รูปแบบ	จำนวน	รหัส	หมายเหตุ
LED	LEDUP	1	D11	เรกกูเรเตอร์
3) ตัวต้านทาน (Resister)				
R 330 5% 1/4W	AXIAL0.4	1	R1	เรกกูเรเตอร์
R 1K 5% 1/4W	AXIAL0.4	1	R2	ปรีแอมป์
R 4K7 5% 1/4W	AXIAL0.4	1	R4	รีเซต
R 10K 5% 1/4W	AXIAL0.4	1	R15	พุลอัพ (Pull Up) สัญญาณ $\overline{ACK}$
R 10K 5% 1/4W	AXIAL0.4	1	R16	พุลอัพสัญญาณ Busy ของเครื่องพิมพ์
R 10K 5% 1/4W	RPACK	2	R11	พุลอัพพอร์ต 0
			R12	พุลอัพ S8255
R 33K 5% 1/4W	AXIAL0.4	1	R3	ปรีแอมป์
R เกือกม้านอน 10K	VR6	2	VR1	วอลลุ่ม (Volume)
			VR2	ปรับความชัดอุปกรณ์แสดงผล
4) ตัวเก็บประจุ (Capacitor)				
C 22pF Ceramic	CAP0.	4	C1,C2	คริสตอล1 (Crystal)
			C24,C25	คริสตอล2
C 470uF/16V Electrolyte	RB.2/.4	1	C3	เรกกูเรเตอร์
C 220uF/16V Elecrolyte	RB.15/.3	2	C4	เรกกูเรเตอร์
			C9	ปรีแอมป์

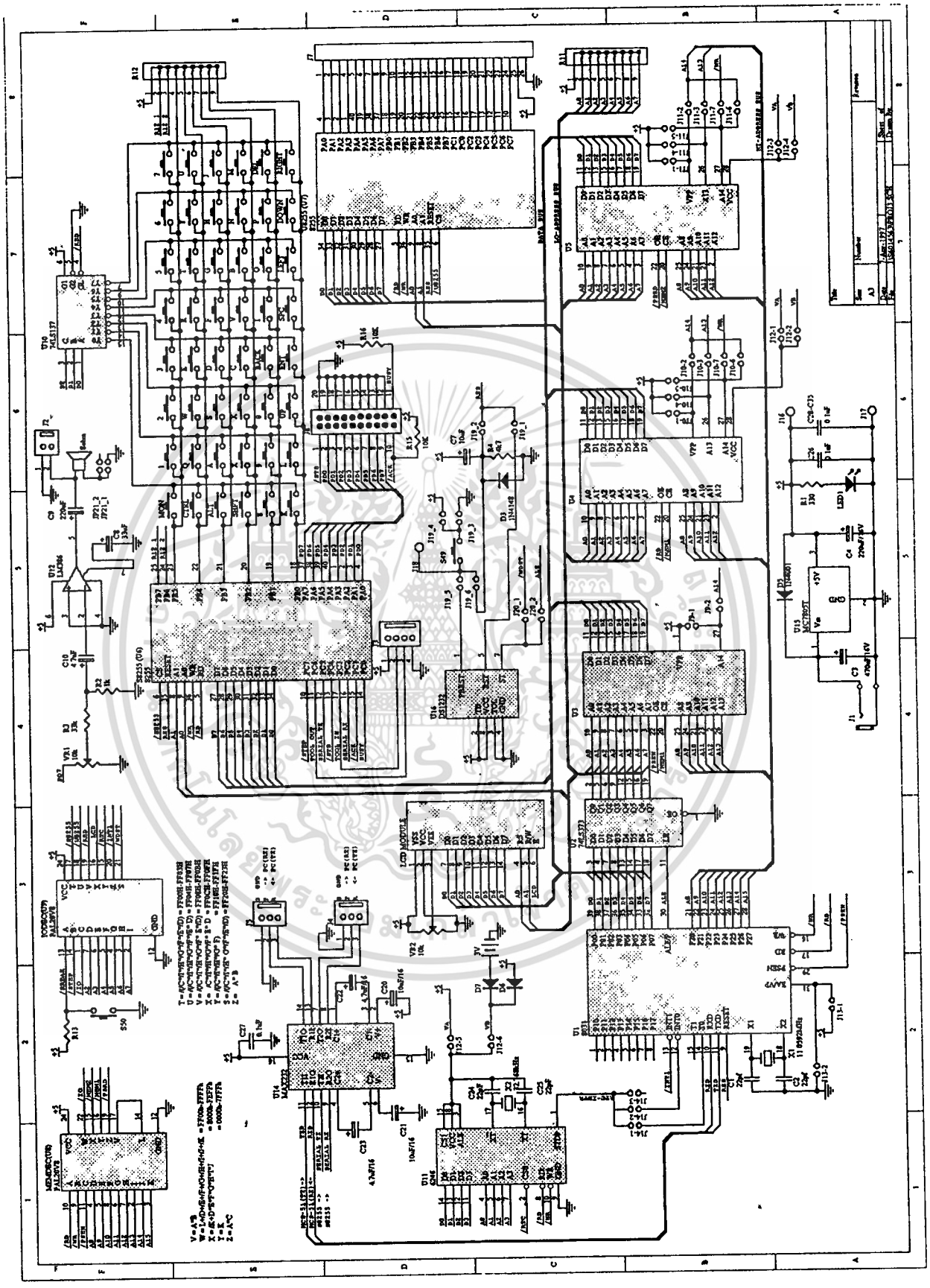
ตารางที่ 12 (ต่อ)

อุปกรณ์	รูปแบบ	จำนวน	รหัส	หมายเหตุ
C 4.7uF/16V Elecrolyte	RB.1/.2	3	C10	ปรีแอมป์
			C22,C23	MAX232
C 33uF/16V Elecrolyte	RB.1/.2	1	C8	ปรีแอมป์
C 10uF/16V Elecrolyte	RB.1/.2	3	C7	รีเซต
			C18	รีเซต
			C19 - C21	MAX232
C 0.1uF Tantalum	TANT0.2	10	C26 - C35	ไอซีบายพาส (IC Bypass)
5) คริสตอล (Crystal)				
11.0592MHz	XTAL	1	X1	คริสตอลหน่วยประมวลผลกลาง
32.798KHz	XTAL2	1	X2	คริสตอลของชิปฐานเวลาจริง
6) คอนเนคเตอร์ (Connector)				
คอนเนคเตอร์ PCB 2 ขา ตัวผู้+ตัวเมีย		1	J2	ลำโพง
คอนเนคเตอร์ PCB 4 ขา ตัวผู้+ตัวเมีย		3	J3 - J5	Max232, สัญญาณ Tool in และ สัญญาณ Tool out
คอนเนคเตอร์ IDC แถวเดี่ยว		1	J13 ,J16 ,J17 ,J18	อุปกรณ์แสดงผล, +5 โวลท์, กราวด์ , รีเซต
คอนเนคเตอร์ IDC แถวคู่ขางอ		1	J15	อุปกรณ์แสดงผล

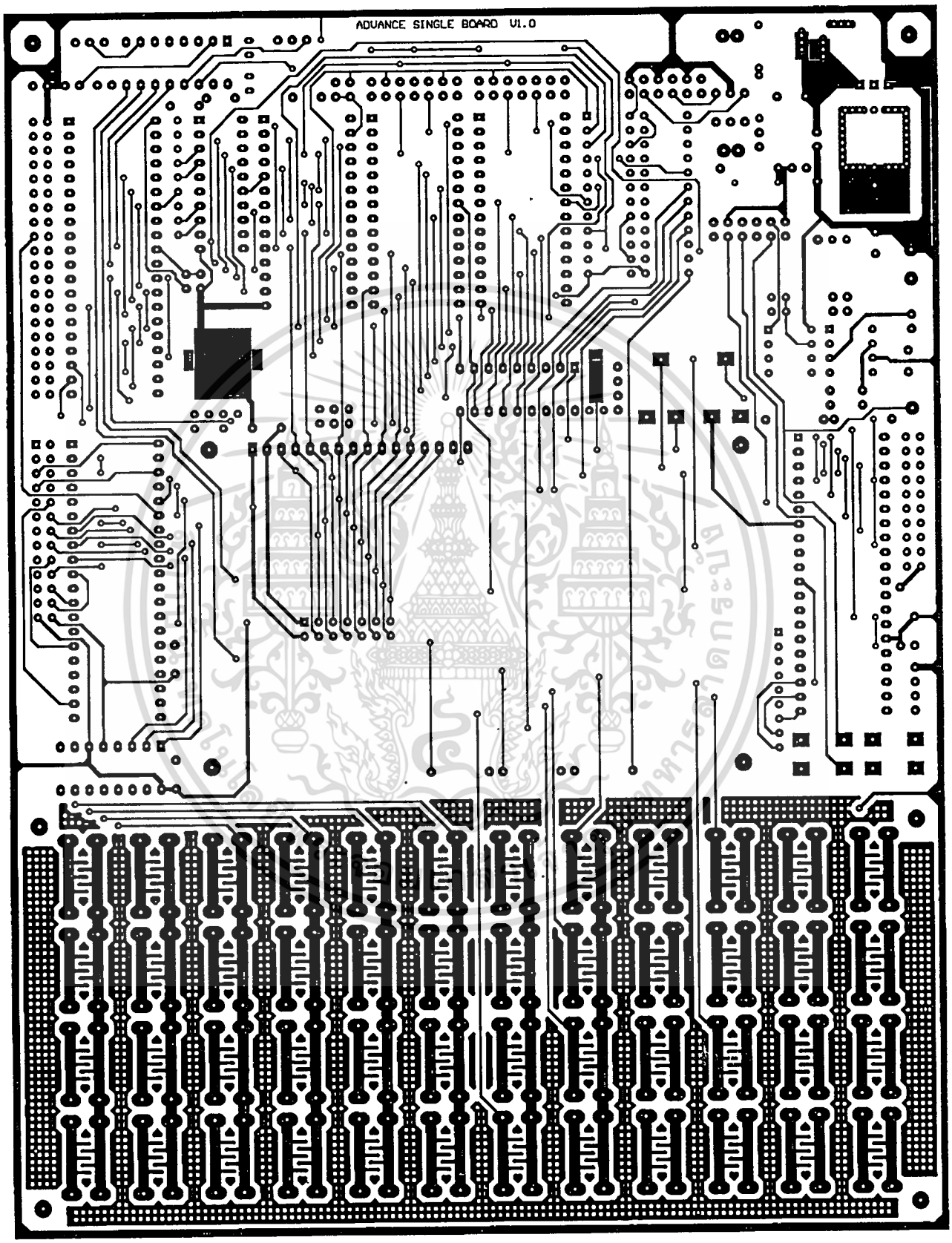
ตารางที่ 12 (ต่อ)

อุปกรณ์	รูปแบบ	จำนวน	รหัส	หมายเหตุ
คอนเนคเตอร์ IDC แถวคู่		1	J6 - J8,J15	คอนเนคเตอร์
			J9 - J14 J19 - J21	จัมเปอร์
จัมเปอร์		1	J9	U3
		6	J10,J11	U4,U5
		3	J12	แบตเตอรีสำรอง
		1	J13	$\overline{EA}/VP$
		1	J14	อินเตอร์รัปชิปฐาน เวลาจริง
		3	J19	DS1232
		1	J20	สัญญาณสโตรบ ของ DS1232
		1	J21	ลำโพง
สายแพ 20 เส้น		1	J15	อุปกรณ์แสดงผล
ดีซีแจ็ค (DC Jack)	DCJACK	1	J1	เรกกูเรเตอร์
7) อื่นๆ				
ถ่านกระดุม 3 V	BATT	1	B1	แบตเตอรี +3 โวลต์
สวิทช์กดติดปล่อยดับ 4 ขา (ใหญ่)	SWPB	48	S1-S48	คีย์บอร์ด
สวิทช์กดติดปล่อยดับ 4 ขา (เล็ก)	SWPBL	2	S49	รีเซต
			S50	สวิทช์เบรค (Break)

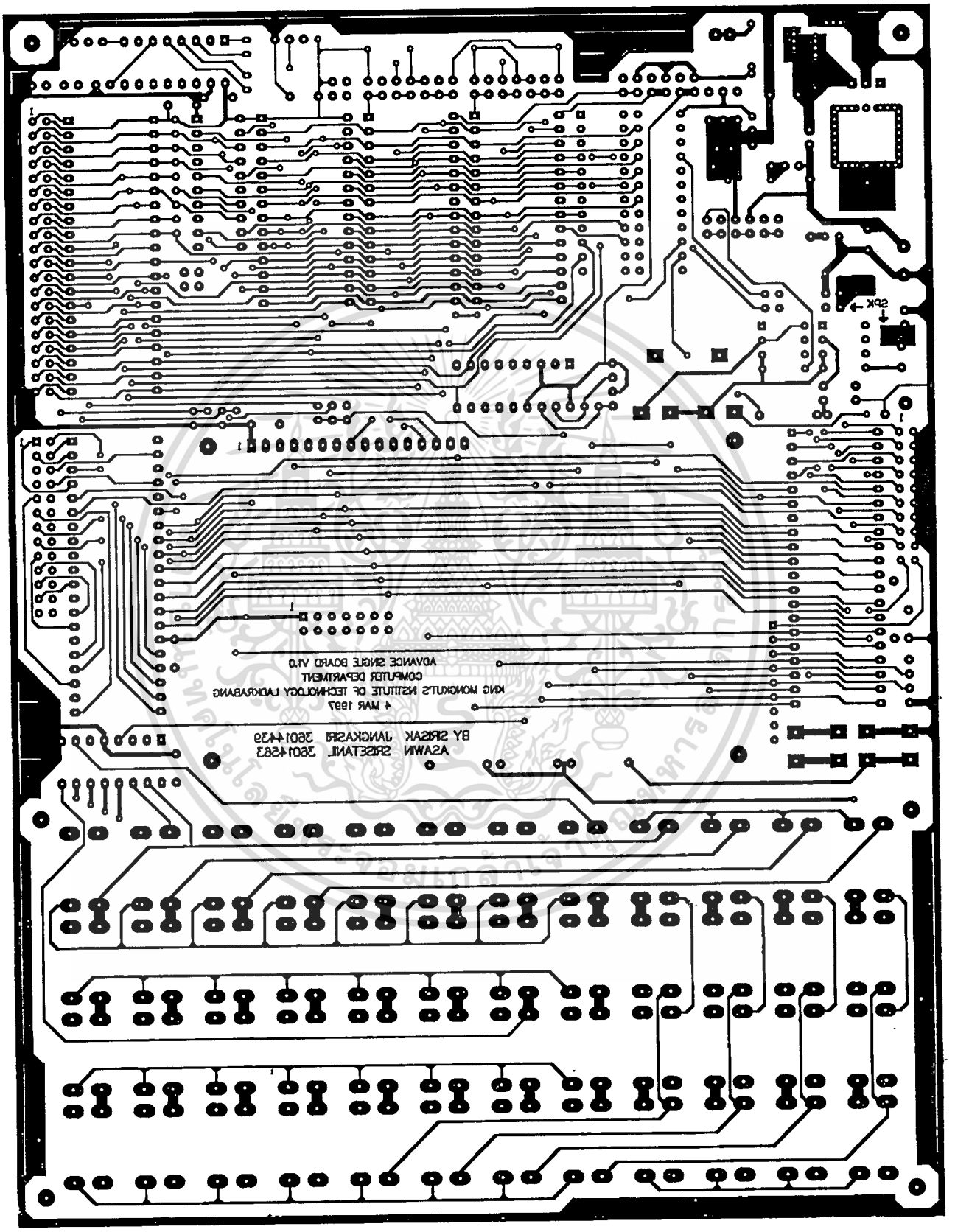
ตารางที่ 12 (ต่อ)



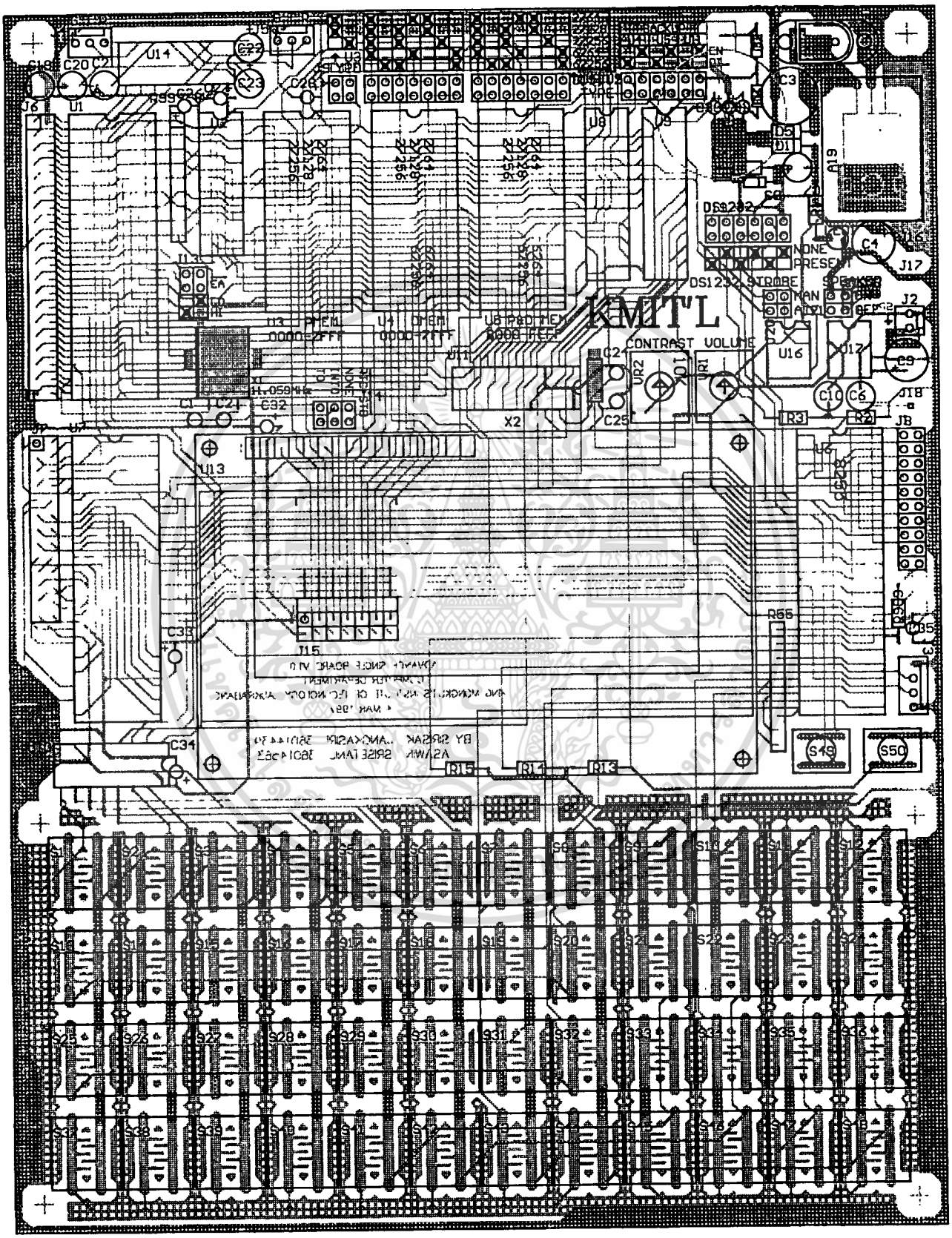
ภาพที่ 21 ภาพแสดงวงจรรวม



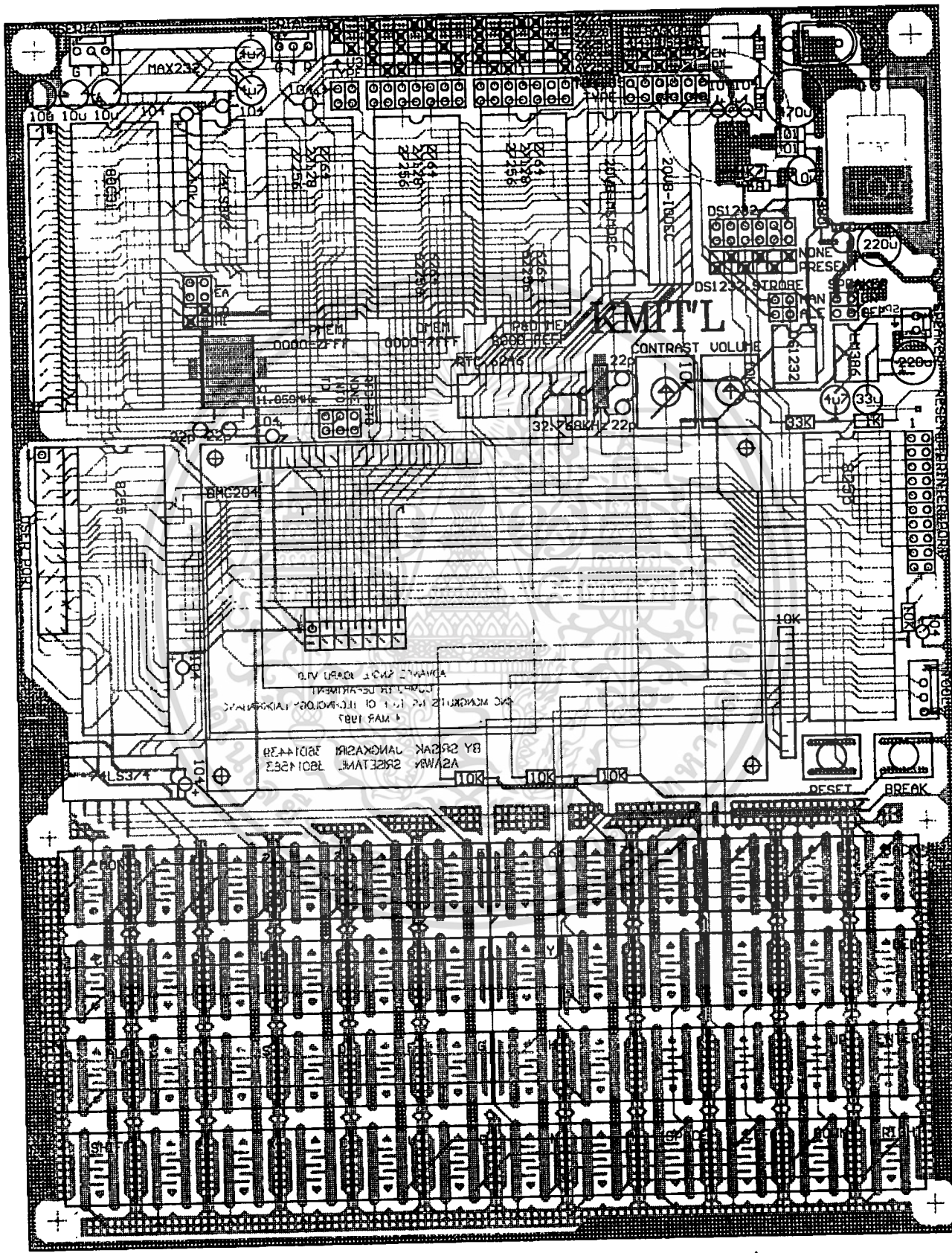
ภาพที่ 22 ภาพแสดงลายวงจรพิมพ์ด้านบน



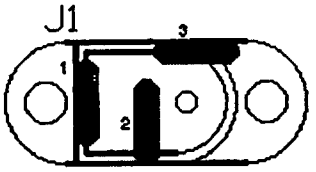
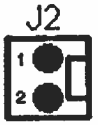
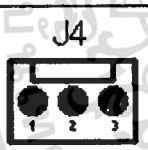
ภาพที่ 23 ภาพแสดงลายวงจรพิมพ์ด้านล่าง



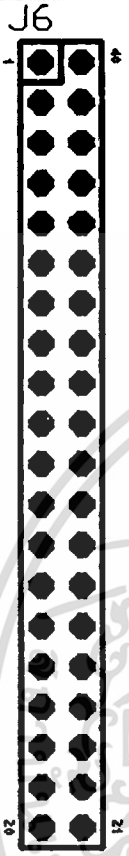
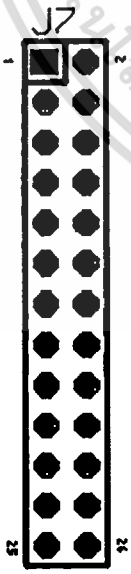
ภาพที่ 24 ภาพแสดงตำแหน่งการวางอุปกรณ์



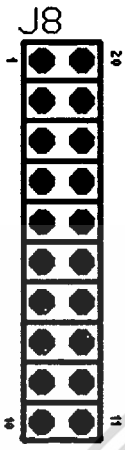
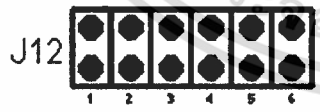
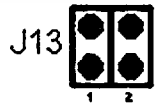
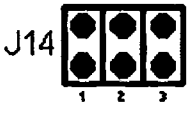
ภาพที่ 25 ภาพแสดงซิลค์สกรีน (Silk Screen) ของแผ่นวงจรพิมพ์

รหัสคอนเนคเตอร์		อุปกรณ์เชื่อมต่อ	หมายเหตุ
J1		ดีซีแจ็ค (DC Jack)	ขา 1,2 = ไฟกระแสตรง +9 โวลท์ ขา 3 = กราวน์ (Ground)
J2		ลำโพง	ขา 1 = กราวน์ ขา 2 = ลำโพงขา +
J3		S8255 (U6)	ขา 1 = +5 โวลท์ ขา 2 = Tool In ขา 3 = Tool out ขา 4 = กราวน์
J4		MAX232 (U14)	ขา 1 = กราวน์ ขา 2 = T2O ขา 3 = R2I
J5		MAX232 (U14)	ขา 1 = กราวน์ ขา 2 = T1O ขา 3 = R1I

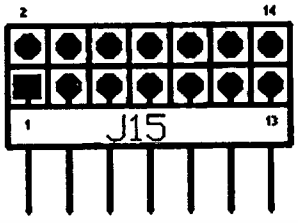
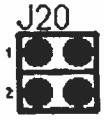
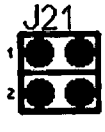
ตารางที่ 13 ตารางสรุปรายละเอียดคอนเนคเตอร์ต่างๆบนซิงเกิลบอร์ด

รหัสคอนเนคเตอร์		อุปกรณ์เชื่อมต่อ	หมายเหตุ
J6		หน่วยประมวลผลกลาง (U1)	ขา 1 - 40 = ต่อเข้ากับขา 1 - 40 ของไมโครคอนโทรลเลอร์
J7		U8255 (U7)	ขา 1 - 8 = PA0 - PA7 ขา 9 - 16 = PB0 - PB7 ขา 17 - 24 = PC0 - PC7 ขา 25 = +5 โวลท์ ขา 26 = กราวน์

ตารางที่ 13 (ต่อ)

รหัสคอนเนคเตอร์		อุปกรณ์เชื่อมต่อ	หมายเหตุ
J8		S8255 (U6) [พอร์ตพริน เตอร์]	ขา 1 = $\overline{STB}$ ขา 2 - 9 = D0 - D9 ขา 10 = $\overline{ACK}$ ขา 11 = BUSY ขา 12 - 20 = GND
J9		จัมเปอร์ 9	ดูตารางที่ 6
J10		จัมเปอร์ 10	ดูตารางที่ 7
J11		จัมเปอร์ 11	ดูตารางที่ 7
J12		จัมเปอร์ 12	ดูตารางที่ 11
J13		จัมเปอร์ 13	ดูตารางที่ 2
J14		จัมเปอร์ 14	ดูตารางที่ 3

ตารางที่ 13 (ต่อ)

รหัสคอนเนคเตอร์		อุปกรณ์เชื่อมต่อ	หมายเหตุ
J15		อุปกรณ์แสดงผลแบบผลึกเหลว (Liquid Crystal Display)	ขา 1 = VSS ขา 2 = VCC ขา 3 = VEE ขา 4 = RS ขา 5 = $R/\overline{W}$ ขา 6 = Enable ขา 7 - 14 = D0 - D7
J16		+5 โวลท์	-
J17		กราวน์	-
J18		รีเซต	-
J19		DS1232	ดูตารางที่ 4
J20		DS1232	ดูตารางที่ 5
J21		ลำโพง	ดูตารางที่ 10

ตารางที่ 13 (ต่อ)

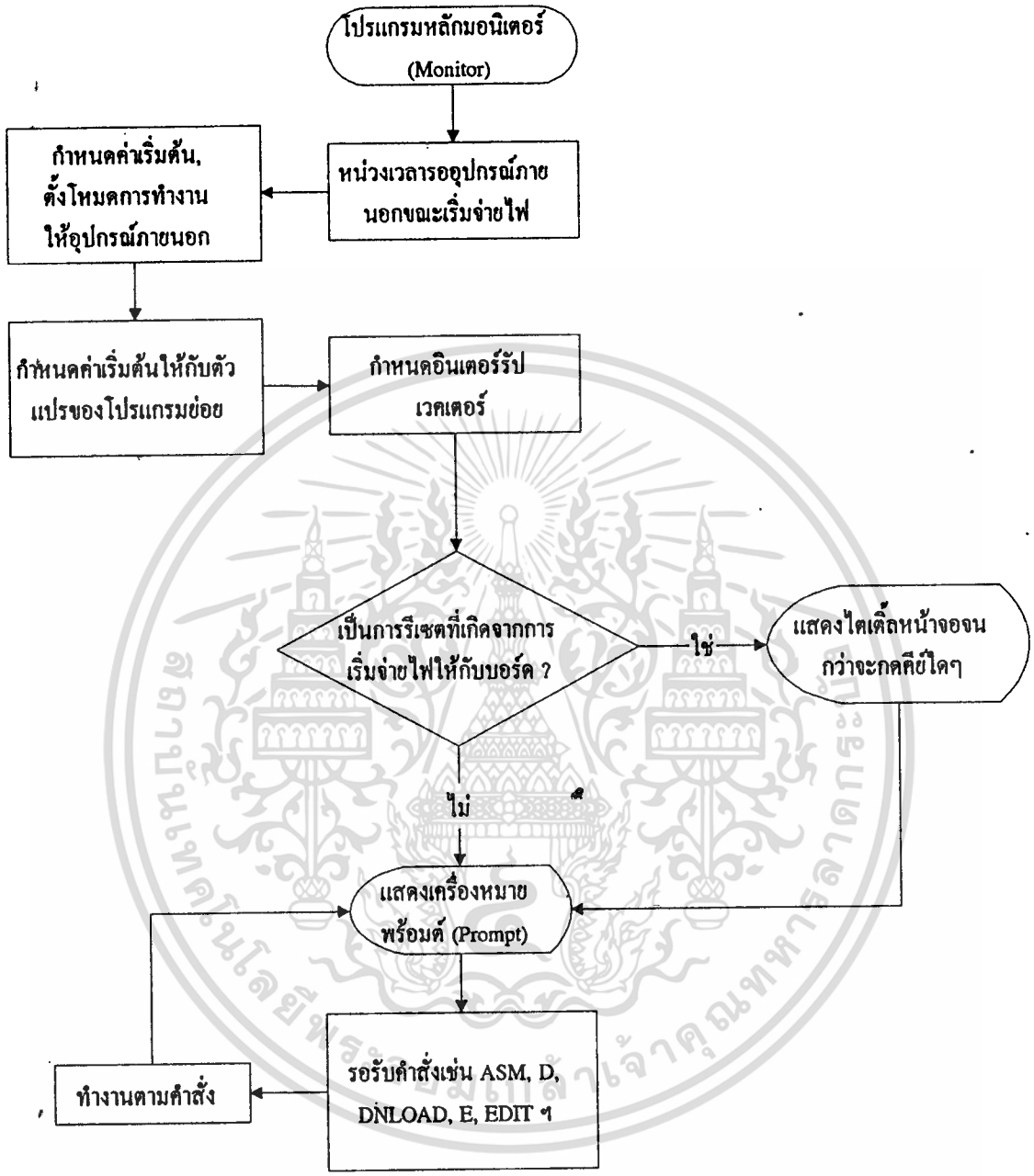
3.2) ซอฟแวร์

ซอฟแวร์เป็นสิ่งสำคัญสำหรับซิงเกิลบอร์ด เนื่องจากเป็นตัวที่กำหนดขั้นตอนการทำงานให้กับซิงเกิลบอร์ด เพื่อให้ได้ผลลัพธ์ตามจุดประสงค์ของผู้ใช้งาน สำหรับซอฟแวร์สามารถแบ่งเป็นส่วนย่อยๆได้ 2 ส่วนคือ

3.2.1) มอนิเตอร์

จากภาพที่ 26 มอนิเตอร์เป็นโปรแกรมส่วนแรกที่เริ่มทำงานในขณะเริ่มจ่ายไฟให้กับบอร์ด เพื่อทำหน้าที่เซตอัพ (Setup) และควบคุม, ดูแลการใช้งานของผู้ใช้ดังนี้

- กำหนดโหมด (Mode) การทำงานเริ่มต้นให้กับอุปกรณ์อินพุตเอาต์พุต (ฮาร์ดแวร์) ดังนี้
  - ◊ S8255 (U6) กำหนดให้พอร์ต A,C (4 บิตบน) เป็นเอาต์พุตพอร์ต และกำหนดให้พอร์ต B,C (4 บิตล่าง) เป็นอินพุตพอร์ต ซึ่งสอดคล้องกับหน้าที่ที่กำหนดไว้แล้วในตอนออกแบบฮาร์ดแวร์
  - ◊ U8255 (U7) กำหนดให้พอร์ต A,C เป็นเอาต์พุตพอร์ต และกำหนดให้พอร์ต B เป็นอินพุตพอร์ต
  - ◊ อุปกรณ์แสดงผลแบบผลึกเหลว กำหนดโหมดการแสดงผล, ลบหน้าจอ ,ย้ายเคอร์เซอร์ไปยังแถวที่ 1 หลักที่ 1
  - ◊ ชิปฐานเวลาจริง กำหนดโหมดของเวลาเป็น 24 ชั่วโมง, หยุดการอินเตอร์รัปของชิป , กำหนดให้เดินเวลา
  - ◊ ความเร็วในติดต่อผ่านพอร์ตอนุกรม โดยกำหนดค่าเริ่มต้นไว้ที่ 9600 บิตต่อวินาที
- กำหนดค่าเริ่มต้นให้กับหน่วยความจำบางส่วนที่สงวนไว้สำหรับระบบ (ซอฟแวร์) เพื่อเก็บสถานะการทำงาน , ตาราง , พอยเตอร์ (Pointer) , ตำแหน่งอินเตอร์รัปเวกเตอร์ (Interrupt Vector) , บัฟเฟอร์ (Buffer) ฯ ที่จำเป็นต่อระบบและโปรแกรมย่อยของระบบ โดยหน่วยความจำที่สงวนไว้มีดังนี้คือ
  - หน่วยความจำชนิดข้อมูลตำแหน่ง 0-3FFFh ใน U4
  - รีจิสเตอร์ภายในของหน่วยประมวลผลกลางตำแหน่ง 10h-21h
- ทำหน้าที่คล้ายเชลล์ (Shell) กล่าวคือรอรับคำสั่งจากพร้อมท์ (Prompt) และกระโดดไปปฏิบัติคำสั่งนั้นๆดังตารางที่ 14



ภาพที่ 26 ภาพแสดงแผนผังการทำงานของมอนิเตอร์

คำสั่ง	พารามิเตอร์ (Parameter)	หมายเหตุ
ASM	-	เรียกโปรแกรมแอสเซมเบลอ (Assembler) เพื่อแปลข้อความของผู้ใช้ในหน่วยความจำที่เป็นภาษาแอสเซมบลี้ (Assembly) เป็นรหัสภาษาเครื่อง (Machine Code) ให้พร้อมที่จะทำงาน
BAUD	[12 | 24 | 48 | 96 | 19 | 1200 | 2400 | 4800 | 9600 | 19200]	แสดง/กำหนดความเร็วในการรับส่งข้อมูลแบบอนุกรม (บิตต่อวินาที)
C	[p | i | d] <address,address> [p | i | d] <address>	เปรียบเทียบค่าในหน่วยความจำ / รีจิสเตอร์ภายใน
CTTY	-	สลับโหมดการแสดงผลระหว่างเทอร์มินอลและอุปกรณ์แสดงผลแบบผลึกเหลว
D	[address ,[address]]	แสดงข้อมูลในหน่วยความจำชนิดข้อมูล
DATE	[dd/mm/yy]	แสดง , ตั้ง วัน/เดือน/ปี ปัจจุบัน
DNLOAD	<address> <-a | -h | -b>	ดาวน์โหลด (Download) ข้อมูลจากคอมพิวเตอร์ภายนอก เป็น แอสกี (Ascii) , ไบนารี (Binary) , หรือ Hex ผ่านพอร์ตอนุกรม
E	<address> [list]	ป้อนเลขฐาน 16 ลงในหน่วยความจำชนิดข้อมูล
EDIT	-	เข้าสู่เอดิเตอร์ (Editor)
EVAL	<expression>	ตีความจากชุดข้อความเป็นตัวเลขมีเครื่องหมายขนาด 2 ไบต์
F	<address,address> <list>	เติมเลขฐาน 16 ลงในหน่วยความจำข้อมูลในช่วงที่กำหนด
G	[address][,address]	กระโดดไปทำงานจากตำแหน่ง หรือในช่วงที่กำหนด
H	[command]	แสดงคำอธิบายถึงคำสั่งต่างๆ
HELP	[command]	แสดงคำอธิบายถึงคำสั่งต่างๆ
I	<port>	แสดงค่าจากพอร์ตหมายเลขนั้นๆ

ตารางที่ 14 ตารางแสดงคำสั่งภายในของมอนิเตอร์

คำสั่ง	พารามิเตอร์ (Parameter)	หมายเหตุ
M	[p | i | d] <address,address> [i | d]<address>	สำเนาข้อมูลในหน่วยความจำ/รีจิสเตอร์ภายในเป็นช่วง ไปยังหน่วยความจำข้อมูล/รีจิสเตอร์ภายใน ตำแหน่งอื่น
O	<port> <byte>	ส่งค่าขนาด 1 ไบต์ออกไปยังพอร์ตนั้นๆ
P	[address]	ไปทำงานในโปรแกรมของผู้ใช้ที่ตำแหน่งที่เก็บในพอยเตอร์ หรือที่ผู้ใช้กำหนดไว้ ครั้งละ 1 คำสั่ง ยกเว้นคำสั่งของผู้ใช้เป็นคำสั่ง ACALL และ LCALL ก็จะทำงานจนกว่าจะจบโปรแกรมย่อยที่เรียกไปด้วย ACALL หรือ LCALL
Q	-	จบการทำงาน (Non Interruptable)
RESET	[*]	รีเซตบางส่วน หรือทั้งหมด
R	[number] [= byte]	แสดง /กำหนดค่าในรีจิสเตอร์
S	<address,address> <list>	ค้นหาข้อความ, ตัวเลขฐานใดๆ ในช่วงหน่วยความจำข้อมูลที่กำหนด
STOP-WATCH	-	นาฬิกาจับเวลา
T	[address]	ทำงานที่ตำแหน่งที่เก็บในพอยเตอร์ หรือที่ผู้ใช้กำหนดไว้ ครั้งละ 1 คำสั่ง
TIME	[hh:mm[:ss]]	แสดง / ตั้งเวลาปัจจุบัน
U	[address][,address]	แปลคำสั่งภาษาเครื่องในหน่วยความจำชนิดโปรแกรมเป็นภาษาแอสเซมบลี (Unassemble)
UPLOAD	<address,address> <-a | -b | -h>	อัพโหลด (Download) ข้อมูลไปคอมพิวเตอร์ภายนอก เป็น แอสกี (Ascii) , ไบนารี , Hex ผ่านพอร์ตอนุกรม
VEC	<number> [= address]	เป็นคำสั่งที่แสดง, หรือกำหนด ตำแหน่งอินเตอร์รัปเวกเตอร์ใหม่
?	[command]	แสดงข้อความอธิบายคำสั่งต่างๆ

ตารางที่ 14 (ต่อ)

หมายเหตุ

[xxx] - จะมีหรือไม่ก็ได้

<xxx> - จำเป็นต้องมี

number,byte - ตัวเลขขนาด 1 ไบต์ เช่น 0, 0ah (ต้องขึ้นด้วยตัวเลขเสมอ)

port - ตัวเลขขนาด 2 ไบต์แทนหมายเลขของพอร์ต เช่น 4000h, 0a000h ฯ (ต้องขึ้นด้วยตัวเลขเสมอ)

address - ตำแหน่งของหน่วยความจำขนาด 2 ไบต์ (ต้องขึ้นด้วยตัวเลขเสมอ)

[p | i | d] <address> - ชนิดของหน่วยความจำ (พารามิเตอร์ปกติคือ d) เมื่อ

- i = รีจิสเตอร์ภายในหน่วยประมวลผลกลาง เช่น 'i 0E0h' หมายถึง ACC
- p = หน่วยความจำที่เก็บคำสั่ง (อ้างด้วย "MOVC") เช่น 'p 8000h'
- d = หน่วยความจำที่เก็บข้อมูล (อ้างด้วย "MOVX") เช่น 'd 4000h'

list - ลำดับรายการของ ไบต์เข้อความ [[,] ไบต์เข้อความ [[,]...]] เช่น 'Hello',20h,0Dh,0Ah,'New Line'

command - คำสั่งในมอนิเตอร์เช่น ASM, I, BAUD เป็นต้น

expression - [จำนวน(ไบต์)เข้อความ [[,]จำนวน(ไบต์)เข้อความ [...]]] เช่น 'String 1',0Dh,0Ah,'String 2',26

3.2.2) คำสั่งในมอนิเตอร์

ดังแสดงไปแล้วในตารางที่ 14 จะเห็นว่าคำสั่งที่เรียกใช้ได้จากมอนิเตอร์พร้อมท์ (Monitor Prompt) มีอยู่มากมาย ซึ่งแต่ละคำสั่งมีรายละเอียดดังนี้

3.2.2.1 ASM

จากภาพที่ 27 คำสั่ง 'ASM' เป็นคำสั่งที่นำโปรแกรมของผู้ใช้ที่ได้รับการสร้าง, แก้ไข (ในหน่วยความจำตำแหน่ง 4000h-7FFFh) ซึ่งเป็นภาษาแอสเซมบลีของ MCS-51 ที่ยังอยู่ในรูปของรหัสแอสกีมา ตรวจสอบไวยากรณ์ , ตีความ, และสร้างรหัสภาษาเครื่องออกมาและเก็บในหน่วยความจำโปรแกรมของผู้ใช้ (8000h-FFEFh) โดยมีความสามารถเช่นเดียวกับ SXA51 ซึ่งเป็นแอสเซมเบลอสำหรับ MCS-51 ซึ่งทำงานบน PC ที่ได้รับความนิยมในกรใช้งานปัจจุบันตัวหนึ่ง โดยมีขั้นตอนการทำงานแบ่งเป็น 2 ช่วงที่สำคัญคือ

- เฟส(Phase) ที่ 1
  - ◊ จะทำการหาอ่านตัวอักษรเข้ามาและตีความเป็นโทเคน (Token)
  - ◊ ดูว่าเป็นโทเคนในแต่ละบรรทัดที่ประกอบกันนั้นเป็นชุดคำสั่งใด ถูกต้องตามไวยากรณ์หรือไม่ มีขนาดเท่าใด หากไม่มีข้อผิดพลาดในโปรแกรมของผู้ใช้ จึงเพิ่มค่าของพอยเตอร์ (Pointer) เท่ากับขนาดคำสั่ง เพื่อจองที่ให้กับคำสั่งดังกล่าวไว้
  - ◊ หากพบคำสั่งเทียม "ORG" ก็จะนำค่าที่ตามหลังมาไปเก็บในพอยน์เตอร์
  - ◊ หากพบคำสั่งเทียม "DB" , "DW" , "DS" ก็จะเพิ่มค่าของพอยน์เตอร์ไปเท่ากับ จำนวนตัวเลข/ตัวอักษร ที่ตามหลัง , จำนวนตัวเลข/ตัวอักษร ที่ตามหลัง * 2 , หรือที่กำหนดไว้ ตามลำดับ เช่นเดียวกับ SXA51
  - ◊ หากโทเคนดังกล่าวเป็นชื่อเลเบล (Label) และไม่มีความผิดพลาดเกิดขึ้นจากการตั้งชื่อ จะนำค่าในพอยน์เตอร์ดังกล่าวซึ่งเป็นตำแหน่งจริงๆของเลเบลนั้นๆพร้อมชื่อไปเก็บไว้ในตารางชั่วคราวตารางตัวแปร (Variable Table)
  - ◊ หากพบคำสั่งเทียม "EQU" ก็จะนำค่าที่ตามหลังมา พร้อมชื่อเลเบลไปเก็บในตารางตัวแปร
- เฟสที่ 2
  - ◊ ทำงานคล้ายกับในเฟสที่ 1 คืออ่านตัวอักษรเข้ามา และแทนด้วยเลขโทเคน ตรวจสอบโทเคนในแต่ละบรรทัดที่ประกอบกันว่าเป็นคำสั่งใด ถูกต้องตามไวยากรณ์หรือไม่ หากไม่มีปัญหาใดๆ ก็จะวางรหัสคำสั่งลงไปและเพิ่มค่าของพอยน์เตอร์ ตามขนาดคำสั่ง

◊ หากโทเคนเป็นชนิดเลเบล ก็จะไปดูตำแหน่งจากตาราง Variable Table หากพบ และ

- ◆ หากคำสั่งดังกล่าวอ้างอิงเลเบลแบบโดยตรง เช่นคำสั่ง "LCALL" ก็จะวางตำแหน่งของเลเบลนั้นลงไปเลย
- ◆ หากมีการอ้างอิงแบบเป็นเพจ (Page) เช่น "ACALL" หรืออ้างอิงแบบสัมพัทธ์ (Relative) ก็จะต้องนำค่าในตารางกับค่าในพอยน์เตอร์ (แอดเดรสปัจจุบัน) ไปคำนวณเสียก่อน

◊ หากพบคำสั่งเทียม "ORG" ก็จะนำค่าที่ตามหลังมาไปเก็บในพอยน์เตอร์

◊ หากพบคำสั่งเทียม "DB" , "DW" , "DS" ก็จะใส่ตัวเลข/รหัสแอสกี ที่ตามหลัง ลงไปในหน่วยความจำโปรแกรม และเพิ่มค่าของพอยน์เตอร์ เหมือนกับโปรแกรม SXA51

◊ หากพบคำสั่งเทียม "EQU" ก็จะข้ามไปบรรทัดต่อไป

ต่อไปนี้เป็นตารางตัวอย่างแสดงโครงสร้างการเก็บไวยากรณ์ของแต่ละคำสั่ง

โทเคนคำสั่ง	โครงสร้างโอเปอแรนด์ (Operand Stucture)	ขนาด (ไบต์)	รหัสคำสั่ง (Opcode)	โหมดการคำนวณของโอเปอแรนด์
T_ACALL	'*',0	2	11h	AREL
T_CLR	T_A,0,	1	E4h	NOP_
T_XCH	T_A,',@!',0	1	C6h	REG

หมายเหตุ

* - ข้อความ (expression) ใดๆ เช่น (500*2)+Label1

? - รีจิสเตอร์ R0..R7

! - รีจิสเตอร์ R0 or R1

T_xxx - หมายเลขประจำโทเคนซึ่งกำหนดไว้ก่อนหน้านี้เช่น T_ACALL EQU 80h

0 - เป็นเครื่องหมายบอกจบของโครงสร้างโอเปอแรนด์

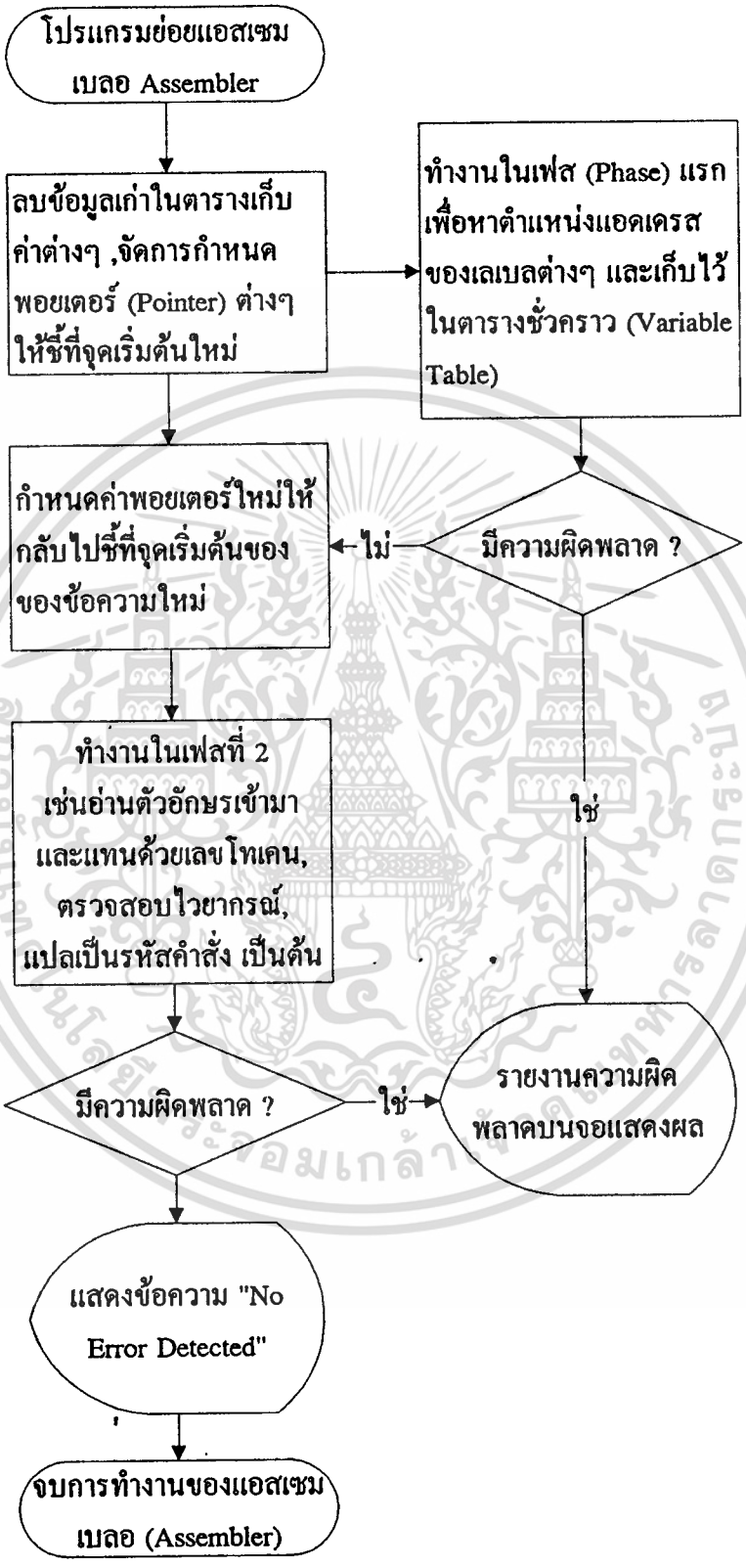
NOP_ - ไม่ต้องทำสิ่งใดหลังจากใส่รหัสภาษาเครื่องลงไป

REG - คำนวนโอเปอแรนด์ตามแบบรีจิสเตอร์เช่น "INC Rx" -> 0000xxxx

"INC R0" -> 00001000

AREL - คำนวณแบบเพจ (1 เพจ ขนาด 2 กิโลไบต์)

ตารางที่ 15 ตารางตัวอย่างแสดงโครงสร้างการเก็บไวยากรณ์ของแต่ละคำสั่ง



ภาพที่ 27 ภาพแสดงแผนผังแสดงขั้นตอนการทำงานของแอสเซมเบลอ

สำหรับส่วนที่ใช้ในการแปลข้อความ (expression) ให้เป็นตัวเลขที่เป็นค่าคงที่นั้น สามารถเขียนเป็นไวยากรณ์แบบ Attributed Translation Grammar อย่างคร่าวๆ ดังแสดงต่อไปนี้

$Expr_p \rightarrow Term_q\ Elist_{q,p}$

$Elist_{p,p} \rightarrow e$

$Elist_{p,q} \rightarrow + \quad Term_r\{Add\}_{p,r,s}\ Elist_{s,q}$

$Elist_{p,q} \rightarrow - \quad Term_r\{Sub\}_{p,r,s}\ Elist_{s,q}$

$Elist_{p,q} \rightarrow and\ Term_r\{Add\}_{p,r,s}\ Elist_{s,q}$

$Elist_{p,q} \rightarrow shr\ Term_r\{Shr\}_{p,r,s}\ Elist_{s,q}$

$Elist_{p,q} \rightarrow shl\ Term_r\{Shl\}_{p,r,s}\ Elist_{s,q}$

$Elist_{p,q} \rightarrow xor\ Term_r\{Xor\}_{p,r,s}\ Elist_{s,q}$

$Elist_{p,q} \rightarrow mod\ Term_r\{Mod\}_{p,r,s}\ Elist_{s,q}$

$Elist_{p,q} \rightarrow or \quad Term_r\{Or\}_{p,r,s}\ Elist_{s,q}$

$Term_p \rightarrow STerm_q\ Tlist_{q,p}$

$Tlist_{p,p} \rightarrow e$

$Tlist_{p,q} \rightarrow * \quad STerm_r\{Mul\}_{p,r,s}\ Tlist_{s,q}$

$Tlist_{p,q} \rightarrow / \quad STerm_r\{Div\}_{p,r,s}\ Tlist_{s,q}$

$STerm_p \rightarrow Factor_q\ Stlist_{q,p}$

$STlist_{p,p} \rightarrow e$

$STlist_{p,q} \rightarrow \wedge \quad Factor_r\{Times\}_{p,r,s}\ Stlist_{s,q}$

$Factor_p \rightarrow SFactor_q\ Flist_{q,p}$

$Flist_{p,p} \rightarrow e$

$Flist_{p,q} \rightarrow . \quad SFactor_r\{OpenTable\}_{p,r,s}\ Flist_{s,q}$

$Sfactor_p \rightarrow (Expr_p)$

$Sfactor_p \rightarrow ident_p$

$Sfactor_p \rightarrow + (Expr_p)$

$Sfactor_p \rightarrow + ident_p$

$Sfactor_p \rightarrow - (Expr_s)\ \{2'S\}_{s,p}$

$Sfactor_p \rightarrow - ident_s\ \{2'S\}_{s,p}$

$Sfactor_p \rightarrow$ high $(Expr_s)$ $\{Hi\}_{s,p}$

$Sfactor_p \rightarrow$ high $ident_s$ $\{Hi\}_{s,p}$

$Sfactor_p \rightarrow$ low $(Expr_s)$ $\{Lo\}_{s,p}$

$Sfactor_p \rightarrow$ low $ident_s$ $\{Lo\}_{s,p}$

โดยไวยากรณ์ดังกล่าวนี้จะทำการแปลเป็นตัวเลขขนาด 2 ไบต์แบบมีเครื่องหมายเช่นเดียวกับโปรแกรม SXA51 เช่น

(200 + 200) / 200 + 1 -1 → 1

ACC.0 + 1 → 0E1h

3.2.2.2 BAUD

```
BAUD [12 | 24 | 48 | 96 | 19 | 1200 |
      2400 | 4800 | 9600 | 19200]
```

เป็นคำสั่งที่แสดง /กำหนดความเร็วในการรับส่งข้อมูลผ่านพอร์ตอนุกรม TXD และ RXD ในหน่วยเป็น บิตต่อวินาที โดยสามารถกำหนดได้ 5 ระดับคือ 1200, 2400, 4800, 9600, 19200 บิตต่อวินาที โดยกำหนดผ่านรีจิสเตอร์ฟังก์ชันพิเศษ (Special Function Register) คือ PCON, SCON, TMOD, TH1, TL1

เช่น BAUD 9600 - 9600 บิตต่อวินาที

BAUD 19 - 19200 บิตต่อวินาที

BAUD - แสดงความเร็วในการรับส่งข้อมูล

3.2.2.3 C

```
C [p | i | d] <address,address>
  [p | i | d] <address>
```

เป็นคำสั่งเปรียบเทียบ (Compare) ค่าในหน่วยความจำ / รีจิสเตอร์ภายใน ที่เป็นช่วง กับหน่วยความจำ / รีจิสเตอร์ อีกช่วงหนึ่ง เช่น

c p 8000h,8010h d 4000h - เปรียบเทียบค่าในหน่วยความจำโปรแกรมจากตำแหน่ง 8000h-8010h กับหน่วยความจำชนิดข้อมูลตำแหน่ง 4000h-4010h

3.2.2.4 CTTY

เป็นคำสั่งสลับโหมดการแสดงผลและควบคุมบอร์ดไปมาระหว่าง จากคอมพิวเตอร์ที่เป็นเทอร์มินอลและจากตัวบอร์ด (Local) เอง (Change Terminal Type) โดยการสลับบิตบางบิตไปมา (Toggle) ในหน่วยความจำของระบบ โดยโปรแกรมย่อยสำหรับการแสดงผลและการรับคำสั่งจะตรวจสอบจากบิตดังกล่าวเอง

3.2.2.5 D

D [address [,address]]

เป็นคำสั่งแสดงข้อมูล (Dump) ในหน่วยความจำชนิดข้อมูล โดยจะแสดงในช่วงที่กำหนด เช่น

d 1000h ,1010h - แสดงข้อมูลในหน่วยความจำชนิดข้อมูลในตำแหน่ง 1000h-1010h (เฉพาะเทอร์มินอลโหมด)

d - แสดงข้อมูลต่อจากตำแหน่งเดิมก่อนหน้า

d 1000h - แสดงข้อมูลในหน่วยความจำชนิดข้อมูลเริ่มจากตำแหน่ง 1000h

หมายเหตุ สำหรับโหมดการแสดงผลที่เป็น LCD หลังจากที่เรียกคำสั่งดังกล่าวแล้ว จะยังไม่จบการทำงานทันทีเหมือนเทอร์มินอลโหมด แต่จะมีคีย์ควบคุมการทำงานดังนี้

SPACE - ดูข้อมูลในไบต์ถัดไป

'-' - ดูข้อมูลในไบต์ก่อนหน้านี้

MON - จบการทำงาน

3.2.2.6 DATE

DATE [dd/mm/yy]

เป็นคำสั่งที่แสดง , เปลี่ยน วัน/เดือน/ปี ปัจจุบัน โดยอ่านเขียนข้อมูล, คำสั่ง ลงบนรีจิสเตอร์ของชิปฐานเวลาจริงดังแสดงในตารางที่ 16 เช่น

DATE - แสดงวันที่ปัจจุบัน

DATE 25/3/97 - ตั้งวันที่ปัจจุบันเป็น 25/3/97

รีจิสเตอร์	ตำแหน่งแอดเดรส	หมายเหตุ
RTC0	FF10h	วินาทีหลักหน่วย
RTC1	FF11h	วินาทีหลักสิบ
RTC2	FF12h	นาทีหลักหน่วย
RTC3	FF13h	นาทีหลักสิบ
RTC4	FF14h	ชั่วโมงหลักหน่วย
RTC5	FF15h	ชั่วโมงหลักสิบ
RTC6	FF16h	วันที่หลักหน่วย
RTC7	FF17h	วันที่หลักสิบ
RTC8	FF18h	เดือนหลักหน่วย
RTC9	FF19h	เดือนหลักสิบ
RTCA	FF1Ah	ปีหลักหน่วย
RTCB	FF1Bh	ปีหลักสิบ
RTCC	FF1Ch	เลขประจำสัปดาห์
RTCD	FF1Dh	30SEC,ADJ,IRQ,BUSY,HOLD
RTCE	FF1Eh	T1,T0,ITRPT/STND,MASK
RTCF	FF1Fh	TEST,24/12,STOP,REST

ตารางที่ 16 ตารางแสดงตำแหน่งรีจิสเตอร์ของชิปฐานเวลาจริง

3.2.2.7 DNLOAD

```
DNLOAD [address] <-a-h-b>
```

เป็นคำสั่งที่อ่านข้อมูลจากคอมพิวเตอร์ภายนอกผ่านพอร์ตอนุกรม ในรูปของรหัส แอสกี ,ไบนารี , หรือ Hex และเปลี่ยนอยู่ในรูปของไบนารีเพื่อนำไปไว้ยังหน่วยความจำชนิดข้อมูลในส่วนของผู้ใช้ตามที่ผู้ใช้ต้องการ

สำหรับการโหลดโปรแกรมจะแสดงข้อความ "Waiting Data" รอจนกว่าจะอ่านพบอักษรตัวแรกจากพอร์ตอนุกรม จากนั้นจะเริ่มอ่านข้อมูลจากพอร์ตอนุกรมเข้ามาเรื่อยๆ หากไม่พบก็จะวนกลับมาอ่านใหม่ และจะหยุดอ่านก็ต่อเมื่อวนกลับมาถึง 100*100 รอบแล้ว หากยังไม่มีข้อมูลเข้ามา (ประมาณไม่ถึง 1 วินาที) ก็จะจบการทำงาน เช่น

DNLOAD 4000h -a - รับข้อมูลจากพอร์ตอนุกรมในรูปของรหัสแอสกีและเริ่มเก็บจากตำแหน่ง 4000h

DNLOAD 8000h -h - รับข้อมูลจากพอร์ตอนุกรมในรูปของ Intel Hex Format และเปลี่ยนเป็นไบนารี เพื่อนำไปเก็บเริ่มจากตำแหน่ง 8000h

หมายเหตุ สำหรับการโหลดแบบแอสกี จะตัดตัวอักษร 0Dh (Line Feed) ออกไปจากสัญญลักษณ์จบบรรทัด (End of Line) เพื่อประหยัดเนื้อที่สำหรับเก็บข้อความ เนื่องจากเอดิเตอร์ใช้เพียงตัวอักษร 0Ah ตัวเดียวแทนสัญญลักษณ์จบบรรทัดก็เพียงพอแล้ว

3.2.2.8 E

E <address> [list]

เป็นคำสั่งที่ใช้ในการป้อนตัวเลขฐาน 16 ด้วยค่าที่ตามหลังตำแหน่งแอดเดรสปลายทางลงไปในหน่วยความจำชนิดข้อมูล ที่ต้องการ เช่น

e 8000h 12,'ab' -> หน่วยความจำ 8000h = 0Ch

8001h = 61h

8002h = 62h

แต่หากไม่ตัวเลข/ตัวอักษรตามหลังตำแหน่งแอดเดรส เช่น 'e 8000h' โปรแกรมก็จะแสดงค่าปัจจุบันก่อนและรอให้ผู้ใช้ป้อนค่าเข้าไปใหม่ หากผู้ใช้กดคีย์ Space ค่าใหม่ก็จะถูกบันทึกและเลื่อนไปตำแหน่งถัดไปจนถึงตำแหน่งสูงสุด จากนั้นก็จะวนไปตำแหน่งต่ำสุดอีกครั้ง จนกว่าจะกดปุ่ม Enter ซึ่งค่าใหม่ก็จะถูกบันทึกเช่นกันแต่จากนั้นก็จะจบการทำงาน

3.2.2.9 EDIT

เป็นคำสั่งที่เรียกโปรแกรมเอดิเตอร์เพื่อเข้าไปทำการแก้ไขข้อความของผู้ใช้ในหน่วยความจำชนิดข้อมูลที่ตำแหน่ง 4000h-7FFFFh

คีย์ที่ใช้งาน	ความหมาย
CTRL-QS	เลื่อนเคอร์เซอร์ไปต้นบรรทัด
CTRL-QD	เลื่อนเคอร์เซอร์ไปท้ายบรรทัด
CTRL-QF	ค้นหาคำที่ต้องการ (เลือก Case Sensitive On/Off ได้) [Find]
CTRL-QX	จบการทำงานของเอดิเตอร์ (Quit)
CTRL-L	ค้นหาคำถัดไป [Find Next]
CTRL-J	เปลี่ยนบรรทัด [Goto Line]
CTRL-C	เลื่อนหน้าจอลง [Page Down]
CTRL-R	เลื่อนหน้าจอขึ้น [Page UP]
CTRL-V	พิมพ์แทรก/พิมพ์ทับ [Insert Mode On/Off]
CTRL-H	ลบตัวอักษร [Back Space]
CTRL-G	ลบตัวอักษร [Delete]
CTRL-Y	ลบทั้งบรรทัด [Delete Line]
CTRL-A	เลื่อนเคอร์เซอร์ไปทางซ้าย 1 คำ [Word Left]
CTRL-F	เลื่อนเคอร์เซอร์ไปทางขวา 1 คำ [Word Right]
CTRL-S	เลื่อนเคอร์เซอร์ไปทางซ้าย 1 ตัวอักษร [Character Left]
CTRL-D	เลื่อนเคอร์เซอร์ไปทางขวา 1 ตัวอักษร [Character Right]
CTRL-E	เลื่อนเคอร์เซอร์ขึ้น 1 บรรทัด [Line Up]
CTRL-X	เลื่อนเคอร์เซอร์ลง 1 บรรทัด [Line Down]
CTRL-KR	อ่านข้อความจากพอร์ตอนุกรม (Down Load) มายังหน่วยความจำข้อมูลที่เป็นบัฟเฟอร์ (Buffer) ของเอดิเตอร์
CTRL-KW	ส่งข้อความที่แก้ไขจากหน่วยความข้อมูลจำผ่านพอร์ตอนุกรม (Upload)
CTRL-N	เปลี่ยนโหมดการแสดงผลเทอร์มินอล <-> อุปกรณ์แสดงผลบนบอร์ด

ตารางที่ 17 ตารางแสดงชุดคำสั่งพิเศษในโปรแกรมเอดิเตอร์

โดยการทำงานหลังจากกำหนดค่าเริ่มต้นต่างๆ (Configuration) แล้วก็จะรอรับอินพุตจากผู้ใช้ (พอร์ตอนุกรม หรือ คีย์บนบอร์ด)

- หากเป็นคีย์พิเศษในตารางที่ 17 ก็จะทำหน้าที่ไปตามจุดประสงค์ของคีย์นั้นๆ

- หากเป็นคีย์ Enter ก็จะตรวจสอบว่าบัฟเฟอร์เต็มหรือยัง ถ้ายังมีที่ว่างก็จะแทรกตัวอักษร 0Dh ลงไปเพียงตัวเดียวเพื่อแทนสัญญลักษณ์จบบรรทัด และแสดงผลบนหน้าจอใหม่เท่าที่จำเป็น
- หากเป็นคีย์ DEL , Backspace ก็จะลบตัวอักษรครั้งละ 1 ตัวเช่นเดียวกับคอมพิวเตอร์ทั่วไป
- หากเป็นคีย์ TAB และยังมีที่เหลือในหน่วยความจำก็จะแทรกตัวอักษร 20h ลงไป 8 ตัว
- คีย์ที่เหลือหากยังมีที่ว่างอยู่ ก็จะแทรกรหัสแอสกีของคีย์นั้นลงไป

3.2.2.10 EVAL

EVAL <expression>

เป็นคำสั่งซึ่งเรียกโปรย่อยของแอสเซมเบลอเพื่อตีความเป็นเลขจำนวนเต็มมีเครื่องหมายขนาด 2 ไบต์และแสดงผลออกมา เช่น

EVAL (2+2-2)*2 -> 4

3.2.2.11 F

F <address,address> <list>

เป็นคำสั่งเติมเลขฐาน 16 ที่ลงในหน่วยความจำชนิดข้อมูลในช่วงตำแหน่งที่กำหนดด้วยตัวเลข/รหัสแอสกี ที่ตามหลังช่วงของตำแหน่ง เช่น

f 8000h 8003h 01,02 -> หน่วยความจำข้อมูล

8000h = 01h

8001h = 02h

8002h = 01h

8002h = 02h

3.2.2.12 G

G [address][,address]

เป็นคำสั่งที่จะให้หน่วยประมวลผลกลางกระโดดทำงานที่ตำแหน่งใดๆ หรือทำงานเป็นช่วงที่กำหนดเช่น

G 8000h - เริ่มทำงานจากตำแหน่ง 8000h

G 8000h,9000h - เริ่มทำงานจากตำแหน่ง 8000h และจะหยุดจนกว่าจะถึงตำแหน่ง9000h

G ,8003h - เริ่มทำงานต่อจากปัจจุบันและหยุดจนกว่าจะถึงตำแหน่ง 8003

G - เริ่มทำงานต่อจากปัจจุบัน

3.2.2.13 H,HELP,?

H	[command]
HELP	[command]
?	[command]

เป็นคำสั่งเพื่อแสดงข้อความอธิบายคำสั่งต่างๆคำสั่งใดคำสั่งหนึ่ง หรือทั้งหมดที่มีอยู่ในมอนิเตอร์โปรแกรม เช่น H EDIT -> อธิบายคำสั่ง EDIT

? -> แสดงคำสั่งทั้งหมด

3.2.2.14 I

I <port>

เป็นคำสั่งแสดงค่าจากพอร์ตหมายเลขนั้นๆ บนจอเช่น

I 0FFFFh - อ่านข้อมูลจากพอร์ตหมายเลข 0FFFFh

<u>หมายเหตุ</u> หากค่าของพอร์ตอยู่ในช่วง 0000h-FEFFh จะเป็นการอ่านข้อมูลจากหน่วยความจำชนิดข้อมูลแทน

3.2.2.15 M

M [p | i | d] <address,address>
[i | d]<address>

เป็นคำสั่งเพื่อทำสำเนาข้อมูลในหน่วยความจำใดๆเป็นช่วง ไปยังหน่วยความจำที่เป็นชนิดข้อมูล หรือรีจิสเตอร์ภายในอีกตำแหน่งหนึ่งเช่น

m p 1000h ,1001h i 00h -> (i00) = (p1000h)
(i01) = (p1001h)

m p 1000h ,1001h d 8000h -> (d8000h) = (p1000h)
(d8001h) = (p1001h)

3.2.2.16 O

O <port> <byte>

เป็นคำสั่งส่งตัวเลขฐาน 16 จำนวน 1 ไบต์ออกไปยังพอร์ตหมายเลขนั้นๆ เช่น

O 0FFFFh 20h -> ส่งค่า 20h ไปยังพอร์ตหมายเลข 0FFFFh

<u>หมายเหตุ</u> หากค่าของพอร์ตอยู่ในช่วง 0000h-FEFFh จะเป็นการส่งข้อมูลไปยังหน่วยความจำชนิดข้อมูลแทน

3.2.2.17 P

P [address]

เป็นคำสั่งให้หน่วยประมวลผลกลางทำงานที่ตำแหน่งปัจจุบัน หรือกำหนดไว้ ทีละคำสั่ง และกลับมาที่มอนิเตอร์และแสดงค่าในรีจิสเตอร์บนจอ ยกเว้นเป็นคำสั่ง ACALL และ LCALL จะทำงานจนกว่าจะจบโปรแกรมย่อยที่ ACALL หรือ LCALL เรียกไป เช่น

P 8000h -> กระโดดไปทำงานที่ตำแหน่ง 8000h

3.2.2.18 Q

เป็นคำสั่งที่สั่งให้หน่วยประมวลผลหยุดทำงานใดๆ (Sleep Mode) และจะไม่รับสัญญาณอินเตอร์รัปใดๆอีกต่อไป

3.2.2.19 R

R [number] [= byte]

เป็นคำสั่งแสดง /กำหนดข้อมูลในรีจิสเตอร์ต่างๆ ตัวอย่างเช่น

R	->	แสดงข้อมูลในรีจิสเตอร์ทั่วไปทั้งหมด (สำหรับโหมด LCD จะใช้คีย์ลูกศร บน, ล่าง ในการเลื่อนจอ และคีย์ MON ในการจบการทำงาน)
R 0	->	แสดงข้อมูลเริ่มจากรีจิสเตอร์ 0 (เฉพาะ LCD)
R ACC	->	แสดงข้อมูลในรีจิสเตอร์ A
R DPH = 0	->	กำหนดค่าในรีจิสเตอร์ DPH ให้เป็น 0

3.2.2.20 RESET

RESET [*]

เป็นคำสั่งรีเซตเพื่อสั่งให้หน่วยประมวลกลางเริ่มทำงานใหม่ , เคลียร์ (Clear) ข้อมูลบางส่วนเช่น ข้อมูลในรีจิสเตอร์ ฯ และยังเก็บข้อมูลในบัฟเฟอร์หรือหน่วยความจำบางส่วน (RESET) หรือรีเซตการทำงานและเคลียร์ข้อมูลใหม่ทั้งหมดและแสดงไตเติ้ลใหม่ (RESET *)

3.2.2.21 S

S <address,address> <list>

เป็นคำสั่งค้นหาข้อความ, ตัวเลขฐานใดๆ ในช่วงหน่วยความจำชนิดข้อมูลที่กำหนด หากพบก็จะแสดงตำแหน่งที่พบออกมาด้วย เช่น

S 0A000h ,0B000h 'abcd' -> ค้นหาชุดของเลขฐาน 16 ซึ่งตรงกับรสัศแอสกีของ 'abcd' ในช่วงหน่วยความจำชนิดข้อมูลตำแหน่ง A000h-B000h

3.2.2.22 STOPWATCH

STOPWATCH

เป็นคำสั่งที่ใช้ซิงเกิลบอร์ดเป็นนาฬิกาจับเวลาโดยมีคีย์ควบคุมการทำงานดังนี้

C - เริ่มจับเวลา

R - รีเซตเวลาใหม่

S - หยุดเวลา

Q - ออกจากโปรแกรม

โดยที่โปรแกรมดังกล่าวจะทำงานได้เฉพาะโหมด LCD เท่านั้น

3.2.2.23 T

T [address] ,

เป็นคำสั่งทำงานเช่นเดียวกับคำสั่ง P คือให้หน่วยประมวลผลกลางกระโดดไปทำงานของผู้ใช้ทีละคำสั่ง แต่ถ้าคำสั่งของผู้ใช้ที่พบเป็น "ACALL" หรือ "LCALL" ก็จะเข้าไปในโปรแกรมย่อยของผู้ใช้เลย จะไม่รอให้กลับออกมาก่อนแล้วค่อยเข้ามอนิเตอร์เหมือนคำสั่ง P

3.2.2.24 TIME

TIME [hh:mm[:ss]]

เป็นคำสั่งที่แสดง , เปลี่ยน เวลาในปัจจุบัน โดยอ่านเขียนข้อมูล, คำสั่ง ลงบนรีจิสเตอร์ของชิปฐานเวลาจริงดังแสดงในตารางที่ 16 เช่น

TIME - แสดงเวลาปัจจุบัน

TIME 12:00 - ตั้งเวลาปัจจุบันเป็น 12:00 น.

3.2.2.25 U

U [address][,address]

เป็นคำสั่งแปลคำสั่งภาษาเครื่องเป็นภาษาแอสเซมบลี (UNASSEMBLE) ในช่วงของหน่วยความจำชนิดโปรแกรมที่กำหนดเช่น

U - แปลคำสั่งต่อจากเดิม

U 8000h,8100h - แปลคำสั่งจากตำแหน่ง 8000h-8100h (เฉพาะเทอร์มินอลโหมด)

U ,8100h - แปลคำสั่งต่อจากเดิมจนถึงตำแหน่ง 8100h (เฉพาะเทอร์มินอลโหมด)

หมายเหตุ สำหรับโหมดการแสดงผลที่เป็น LCD หลังจากที่เรียกคำสั่งดังกล่าวแล้ว จะยังไม่จบการทำงานทันทีเหมือนเทอร์มินอลโหมด แต่จะมีคีย์ควบคุมการทำงานดังนี้

SPACE - แปลคำสั่งถัดไป

MON - จบการทำงาน

3.2.2.26 UPLOAD

UPLOAD <address,address> <-a | -b | -h>

เป็นคำสั่งที่ส่งข้อมูลจากหน่วยความจำชนิดข้อมูลไปยังคอมพิวเตอร์ภายนอก ผ่านพอร์ตอนุกรมในรูปของรหัส แอสกี ,ไบนารี , และ HEX เช่น

upload 4000h,7fffh -a -> ส่งข้อมูลในหน่วยความจำชนิดข้อมูลในตำแหน่ง 4000h-7fffh ออกไปในรูปของรหัสแอสกี

upload 8000h,9000h -h -> ส่งข้อมูลในหน่วยความจำชนิดข้อมูลในตำแหน่ง 8000h-9000h ออกไปในรูปของ Intel Hex Format

upload 8000h,9000h -b -> ส่งข้อมูลในหน่วยความจำชนิดข้อมูลในตำแหน่ง 8000h-9000h ออกไปในรูปของไบนารี

หมายเหตุ สำหรับการอัพโหลดแบบแอสกี หากพบตัวอักษร 0Dh (Line Feed) ซึ่งแทนสัญญลักษณ์จบบรรทัดในเอดิเตอร์และไม่มีตัวอักษร 0Ah ตามมา ก็จะส่งตัวอักษร 0Ah ออกไปด้วย

3.2.2.27 VEC

VEC <number> [= address]

เป็นคำสั่งที่แสดง, หรือกำหนด ตำแหน่งอินเตอร์รัปเวกเตอร์ใหม่โดยความหมายของอินเตอร์รัปเวกเตอร์แต่ละหมายเลขเป็นไปตามตารางที่ 18 เช่น

vec 0 -> แสดงอินเตอร์รัปเวกเตอร์ของการเกิดอินเตอร์รัปภายนอก 0

vec 2 = 8000h -> กำหนดอินเตอร์รัปเวกเตอร์ให้กับการเกิดอินเตอร์รัปภายนอกที่ส่งสัญญาณผ่านขา INT1

หมายเลข	สัญญาณ	ตำแหน่งเริ่มต้น	ความหมาย
0	IE0	8003h	อินเตอร์รัปภายนอก 0
1	TF0	800Bh	วงจรนับ/จับเวลา 0
2	IE1	xxxx	อินเตอร์รัปภายนอก 1 (ปกติระบบสงวนไว้ใช้ในการหยุดการทำงาน Break))
3	TF1	801Bh	วงจรนับ/จับเวลา 1
4	RI หรือ TI	8023h	วงจรรับ/ส่งข้อมูลอนุกรม
5	TF2	802bh	วงจรนับ/จับเวลา 2

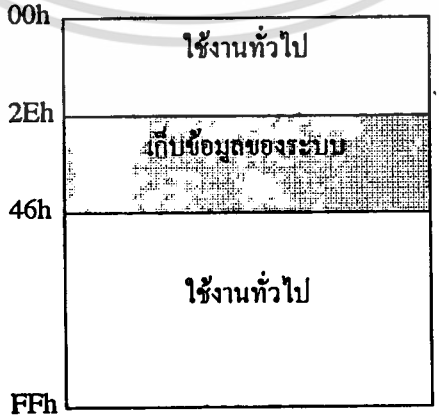
ตารางที่ 18 ตารางแสดงความหมายของหมายเลขอินเตอร์รัปเวกเตอร์

3.2.3) การใช้งานหน่วยความจำ

อย่างไรก็ตามในการใช้งานซอฟแวร์ ผู้ใช้จะต้องระวังการเข้าไปแก้ไข หรือเปลี่ยนแปลงข้อมูลในหน่วยความจำที่ระบบสงวนเอาไว้ มิฉะนั้นการทำงานของโปรแกรมบนบอร์ดอาจไม่ถูกต้องได้ จากภาพที่ 28 ผู้ใช้สามารถจะนำซอร์สโค้ด (Source Code) ของผู้ใช้ที่เป็นแอสกีไปเก็บไว้ในตำแหน่ง 4000h-7FFFh และผู้ใช้สามารถเก็บข้อมูล หรือโปรแกรมของตัวเองที่เป็นไบนารี (Binary) ไว้ในตำแหน่ง 8000h-FEFFh

และจากภาพที่ 29 ในระหว่างที่โปรแกรมของผู้ใช้ทำงานอยู่ ก็ไม่ควรเข้าไปใช้รีจิสเตอร์ภายในของหน่วยประมวลผลกลางในตำแหน่ง 2Fh-46h เช่นกัน

ภาพที่ 28 ภาพแสดงการใช้งานหน่วยความจำชนิดข้อมูล



ภาพที่ 29 ภาพแสดงการใช้งานรีจิสเตอร์ภายในของหน่วยประมวลผลกลาง

บทที่ 4

การทดลองและผลการทดลอง

4.1 ฮาร์ดแวร์

4.1.1 วงจรควบคุมหลัก

อุปกรณ์ 1 ซิงเกิลบอร์ด

2 อีพรอมอีมูเลเตอร์ (Eprom Emulator)

3 ลอจิกโพรบ

ขั้นตอนการทดลอง

- เขียนโปรแกรมส่งค่า #055h ออกที่พอร์ต 1 ของหน่วยประมวลผลกลาง ด้วยคำสั่ง Mov p1,#055h และสังเกตผลโดยใช้ลอจิกโพรบวัดสัญญาณที่ขา 1-7 ของ U1 (8031)
- ทดลองส่งค่าอื่นๆออกมาที่พอร์ต 1

ผลการทดลอง

- สัญญาณที่พอร์ต 1 เป็นไปตามค่าที่ส่งออกมา

สรุปผลการทดลอง

- หน่วยประมวลผลกลางบนซิงเกิลบอร์ดสามารถทำงานตามโปรแกรมของผู้ใช้ได้อย่างถูกต้อง

4.1.2 ส่วนแสดงผล

อุปกรณ์ 1 ซิงเกิลบอร์ด

2 อีพรอมอีมูเลเตอร์ (Eprom Emulator)

ขั้นตอนการทดลอง

- ศึกษาหัวข้อ 3.1.1.4 และวงจรในภาพที่ 10 รวมทั้งตาราง ก.2 ภาคผนวก ก
- ทดลองเขียนโปรแกรมให้หน่วงเวลาประมาณ 1/10 วินาที แล้วจึงส่งคำสั่งต่อไปนี้ไปยังอุปกรณ์แสดงผลแบบผลึกเหลว LCD ที่พอร์ต FF0Ch เพื่อกำหนดโหมดการทำงานให้ LCD

- #00111000b ;function set
- #00000110b ;entry mode set
- #00001111b ;display on,cursor on,blink on
- #00000001b ;clear screen
- #00000010b ;cursor goto home

- ให้โปรแกรมส่งรหัสแอสกีของ 'a' ไปที่พอร์ต FF0Fh เพื่อส่งข้อมูลที่จะแสดงบนจอไปที่ LCD
- ให้โปรแกรมลองส่งตัวอักษรตัวอื่นตามไปบ้าง

ผลการทดลอง

- บนจอ LCD สามารถแสดงตัวอักษรตามรหัสแอสกีที่ส่งไปให้ได้

สรุปผลการทดลอง

- LCD บนบอร์ดสามารถทำงานได้อย่างถูกต้อง

4.1.3 พอร์ตใช้งานทั่วไป 8255

อุปกรณ์ 1 ซิงเกิลบอร์ด

2 อีพรอมอีมูเลเตอร์ (Eprom Emulator)

3 ลอจิกโพรบ

ขั้นตอนการทดลอง

- หมายเลขพอร์ตในตารางที่ 8 ทดลองเขียนโปรแกรมหน่วงเวลา 1/10 วินาที เพื่อรอ 8255 เซตอัพภายใน และส่งค่า #82h ไปยังพอร์ต D ของ U6 ในภาพที่ 21 ที่ตำแหน่ง FF03h เพื่อให้พอร์ต A เป็นเอาต์พุตพอร์ต
- ส่งค่า #055h ไปยังพอร์ต A ของ U6 ที่ตำแหน่ง FF00h
- ใช้ลอจิกโพรบวัดสัญญาณที่พอร์ต A
- ทดลองกับพอร์ต A ของ U7 ในทำนองเดียวกัน

ผลการท่ดลอง

- พอร์ต A สามารถส่งค่าออกมาได้ตามที่โปรแกรมต้องการ

สรุปผลการทดลอง

- 8255 บนบอร์ดสามารถทำงานได้อย่างถูกต้อง

4.1.4 ส่วนคีย์บอร์ด

อุปกรณ์ 1 ซิงเกิลบอร์ด

2 อีพรอมอีมูเลเตอร์ (Eprom Emulator)

ขั้นตอนการทดลอง

- จากวงจรในภาพที่ 11 และหมายเลขพอร์ตในตารางที่ 8 ทดลองเขียนโปรแกรมหน่วงเวลา 1/10 วินาที แล้วส่งคำสั่ง #82h ไปให้ U6 ที่ตำแหน่ง FF03h เพื่อให้พอร์ต B เป็นอินพุต
- ส่งค่า #0 ไปที่พอร์ต FF08h เพื่อเลือกคีย์ในกลุ่มที่ 1
- ให้โปรแกรมอ่านค่าจากพอร์ต B ของ U6 และกำจัด (Mask) 2 บิตบนทิ้งไป จากนั้นนำค่าดังกล่าวไปแสดงที่จอ LCD ตลอดเวลาในรูปของเลขฐาน 2
- ทดลองกดคีย์ในกลุ่มที่ 1 และสังเกตผลบน LCD
- ทดลองกดคีย์ในกลุ่มอื่นๆ และสังเกตผล

ผลการทดลอง

- หากกดคีย์ในกลุ่มที่ 1 ตั้งแต่ 1 คีย์ขึ้นไปจะมีบิตบางบิตที่ตำแหน่งตรงกันกลายเป็น 0 แต่คีย์ในกลุ่มอื่นจะไม่มีผล

สรุปผลการทดลอง

- หน่วยประมวลผลกลางบนซิงเกิลบอร์ดใช้งานวงจรแสกนคีย์บอร์ดได้อย่างถูกต้อง

4.1.5 ส่วนวงจรสร้างฐานเวลาจริง RTC

อุปกรณ์ 1 ซิงเกิลบอร์ด

2 อีพรอมอีมูเลเตอร์ (Eprom Emulator)

3 ลอจิกโพรบ

ขั้นตอนการทดลอง

- จากตารางที่ E.2 และ 16 เขียนโปรแกรมหน่วงเวลา 1/10 วินาที แล้วส่งค่าต่างๆไปยังพอร์ตต่อไปนี้
  - #00000100b -> พอร์ต FF1Fh ;24 ชั่วโมง ;นาฬิกาเดิน
  - #00000100b -> พอร์ต FF1Eh ;อินเตอร์ทุกๆ 1 ;วินาที

- ใช้ลอจิกโพรบวัดที่ขา 1 ของชิปดังกล่าวและสังเกตผล
- อ่านค่าจากพอร์ต FF10h และแสดงผลบนจอ LCD ตลอดเวลา และสังเกตผล

ผลการทดลอง

- มีสัญญาณพัลส์เกิดขึ้นทุกๆ 1 วินาที
- บนจอ LCD จะแสดงค่าตัวเลขเพิ่มขึ้นทุกๆ 1 วินาที

สรุปผลการทดลอง

- หน่วยประมวลผลกลางบนซิงเกิลบอร์ดใช้งานชิปฐานเวลาจริงได้อย่างถูกต้อง

4.1.6 ส่วนวงจรอินเตอร์เฟสอาร์เอส 232 (RS-232)

อุปกรณ์ 1 ซิงเกิลบอร์ด

2 อีพรอมอีมูเลเตอร์ (Eprom Emulator)

3 เครื่องคอมพิวเตอร์ส่วนบุคคลพร้อมโปรแกรม Telix

ขั้นตอนการทดลอง

- ต่อคอนเนคเตอร์ J5 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ภายนอก ดังภาพที่ 13, 14, 15
- รันโปรแกรม Telix บนคอมพิวเตอร์ภายนอก และกำหนดหมายเลขพอร์ตให้ตรงกับที่ต่อ และกำหนดความเร็วในการรับส่งเป็น 9600 บิตต่อวินาที
- เขียนโปรแกรมเพื่อกำหนดความเร็วในการรับส่งข้อมูลของซิงเกิลบอร์ดตามโปรแกรมย่อยต่อไปนี้

```
;initialize for serial port
;*******************************************************
  anl   pcon,#01111111b  ;Serial port operation clock is normal
  mov  scon,#52h         ;Serial in Mode 1 ,and enable REN,and
                          ;Set ti for the first sent character
  mov  tmod,#20h         ;Timer 1 in Mode 2
  mov  th1,#0fdh         ;Set baud rate = 9600 bps
  mov  tl1,#0fdh
  setb  tr1               ;Timer 1 run
```

- ทดลองส่งรหัสแอสกีของ 'B' โดยผ่านรีจิสเตอร์ sbuf ดังโปรแกรมย่อยต่อไปนี้ และสังเกตผลบนจอคอมพิวเตอร์

```
mov  a,#'B'
jnb  ti,$              ;wait sending is success
clr  ti
mov  sbuf,a            ;begin send data
```

- ทดลองให้โปรแกรมบนบอร์ดอ่านค่าจากพอร์ตอนุกรมด้วยโปรแกรมย่อยต่อไปนี้ และแสดงผลบน LCD เมื่อได้รับข้อมูลตลอดเวลา จากนั้นลองกดคีย์บอร์ดของคอมพิวเตอร์ด้วยคีย์ตัวอักษร และสังเกตผลบน LCD

```
ser_get:
;*************************************************************
;This routine will receieve character through serial port
;if no input,then return #0
;output   : acc
;*************************************************************
  clr  a                ;                    (1)
  jnb  ri,$+7           ;if not recieve any character
                        ;then return #0      (3)
  clr  ri               ;clear receiver flag (2)
  mov  a,sbuf           ;read character      (2)
  ret
```

<u>ผลการทดลอง</u>

- ซิงเกิลบอร์ดสามารถส่งข้อมูลให้ปรากฏบนจอคอมพิวเตอร์ได้
- ซิงเกิลบอร์ดสามารถรับข้อมูลที่ส่งมาจากคอมพิวเตอร์ได้

สรุปผลการทดลอง

- หน่วยประมวลผลกลางบนซิงเกิลบอร์ดสามารถติดต่อกับคอมพิวเตอร์ภายนอกได้

4.1.7 ระบบเสียง

อุปกรณ์ 1 ซิงเกิลบอร์ด

2 อีพรอมอีมูเลเตอร์ (Eprom Emulator)

ขั้นตอนการทดลอง

- จากวงจรในภาพที่ 18 และตารางที่ 10 ให้กำหนดจัมเปอร์เพื่อใช้ลำโพง
- เขียนโปรแกรมหน่วงเวลา 1/10 วินาที จากนั้นกำหนดโหมดให้พอร์ต A ของ U6 เป็นเอาต์พุต
- ส่งค่า '0' และ '1' สลับกันไปผ่านพอร์ต A บิตที่ 7 โดยหน่วงเวลาในการสลับค่าเท่ากับ t1 และลองฟังเสียงจากลำโพง
- ทดลองเปลี่ยนเวลาที่ใช้หน่วงและลองฟังเสียงใหม่อีกครั้ง

ผลการทดลอง

- ซิงเกิลบอร์ดสามารถสร้างเสียงที่ความถี่ต่างๆได้

สรุปผลการทดลอง

- หน่วยประมวลผลกลางบนซิงเกิลบอร์ดสามารถสร้างเสียงที่ความถี่ต่างๆได้

4.2 ซอฟแวร์

ต่อไปนี้เป็นตัวอย่างไฟล์ที่ใช้ในการทดสอบเอดิเตอร์และแอสเซมเบลอ

```
;TEST.ASM
;It is downloaded to Single Board
 ORG   0A000H

 port_wi equ 0FF0Ch          ;write instruction
 port_wd equ 0FF0Dh          ;write data
 port_ri equ 0FF0Eh          ;read  instruction    ; PCB Board
 port_rd equ 0FF0Fh          ;read  data
```

```
r0_     EQU     00      ; address of R0 to R7
r1_     EQU     01
r2_     EQU     02
r3_     EQU     03
r4_     EQU     04
r5_     EQU     05
r6_     EQU     06
r7_     EQU     07

lcall delay                 ;delay time
lcall lcdclr                ;clear screen
mov   dptr,#text1           ;get text address for display
lcall lcdwritest            ;display the text on LCD
sjmp  $                     ;stop running

text1: db 'Art & Joe',0

delay:   push r1_
         push r2_
         mov r1,#100
         mov r2,#10
         djnz r2,$
         djnz r1,$-4    ;to "mov r2,#10" command
         pop  r2_
         pop  r1_
         ret

lcdwi:   push dph               ;write instuction in acc to LCD
         push dpl
```

```
        push acc
        mov  dptr,#port_ri
        movx a,@dptr
        jb  acc.7,$-1
        pop acc
        mov dptr,#port_wi
        movx @dptr,a
        pop dpl
        pop dph
        ret
lcdwd:   push dph              ;write data in acc to LCD
        push dpl
        push acc
        mov  dptr,#port_ri
        movx a,@dptr
        jb  acc.7,$-1
        pop acc
        mov dptr,#port_wd
        movx @dptr,a
        pop dpl
        pop dph
        ret

blinkon:                       ;display on ,blink on
  mov a,#00001111b
  acall lcdwi
  ret

lcdclr:  mov a,#01h            ;clear screen (LCD)
        acall lcdwi
```

```
        ret

lcdhome:  mov a,#02h        ;move cursor to home
        acall lcdwi
        ret
lcdwritest:                 ;display string pointed by dptr on LCD
nextch:   clr a
        movc a,@a+dptr
        jz printst1
        push dph
        push dpl
        acall lcdwd
        pop  dpl
        pop  dph
        inc  dptr
        sjmp nextch
printst1: ret

  END                       ;End of User Program
```

4.2.1 เอดิเตอร์

<u>อุปกรณ์</u> 1 ซิงเกิลบอร์ด

2 อีพรอมอีมูเลเตอร์ (Eprom Emulator)

<u>ขั้นตอนการทดลอง</u>

- หลังจากจ่ายไฟให้กับบอร์ด และเข้าสู่มอนิเตอร์พร้อมท์แล้ว (ปรากฏเครื่องหมาย '-') เรียกคำสั่ง "baud 9600"
- เรียกคำสั่ง "edit"
- ลองใช้คำสั่ง CTRL-KR เพื่อรับไฟล์ข้อมูลจากคอมพิวเตอร์ภายนอก จากนั้นบนจอ LCD จะแสดงข้อความ "Ploase Download ..."
- ต่อคอนเนคเตอร์ J5 ตามรูปที่ 13, 14, 15

- กำหนดความเร็วในการรับส่งข้อมูลของคอมพิวเตอร์ผ่านพอร์ตที่ x ด้วยคำสั่งต่อไปนี้ "mode comx:9600,n,8,1"
- ทดลองส่งไฟล์ "test.asm" ไปให้กับบอร์ดด้วยคำสั่ง "copy test.asm comx"
- เมื่อมีข้อความของไฟล์ "test.asm" ปรากฏบนจอ LCD จึงทดลองใช้คำสั่งในตารางที่ 17 ในการแก้ไขข้อความ

ผลการทดลอง

- เอดิเตอร์บนซิงเกิลบอร์ดสามารถทำงานได้อย่างถูกต้อง

สรุปผลการทดลอง

- ผู้ใช้สามารถใช้ซิงเกิลบอร์ดเพื่อแก้ไขโปรแกรมขนาดเล็กได้โดยไม่ต้องอาศัยคอมพิวเตอร์ภายนอก

4.2.2 แอสเซมเบลอ

อุปกรณ์ 1 ซิงเกิลบอร์ด

ขั้นตอนการทดลอง

- ทำการทดลองเช่นเดียวกับข้อ 4.2.1
- ออกจากเอดิเตอร์ด้วยคีย์ CTRL-QX
- เมื่ออยู่ในมอนิเตอร์พร้อมท์เรียกคำสั่ง "asm" จะปรากฏข้อความแสดงขั้นตอนการแอสเซมเบิล (Assemble) โปรแกรม
- หากไม่มีความผิดพลาดเกิดขึ้น ให้ทดลองรันโปรแกรมที่ได้แปลเป็นรหัสภาษาเครื่องแล้วด้วยคำสั่ง "g 8000h" จากมอนิเตอร์พร้อมท์

ผลการทดลอง

- ซิงเกิลบอร์ดสามารถทำงานโปรแกรมที่ได้มาจากแปลด้วยแอสเซมเบลอ บนซิงเกิลบอร์ด ได้อย่างถูกต้อง คือสามารถแสดงข้อความออกจอ LCD คือ "Joe & Art" ได้ถูกต้อง

สรุปผลการทดลอง

- ผู้ใช้สามารถใช้ซิงเกิลบอร์ดเพื่อแก้ไขโปรแกรมขนาดเล็กและแปลเป็นรหัสภาษาเครื่องได้ โดยไม่ต้องอาศัยคอมพิวเตอร์ภายนอกได้ (Stand Alone)

บทที่ 5
ใบงานประกอบการทดลอง

5.1 ใบงานที่ 1
การใช้งานซิงเกิลบอร์ดและเทอร์มินอล

วัตถุประสงค์

1. เพื่อเพิ่มทักษะและสร้างความคุ้นเคยในการใช้งานคำสั่งเบื้องต้นบนซิงเกิลบอร์ด
2. เพื่อให้ผู้ใช้มีพื้นฐานการใช้งานซิงเกิลบอร์ดกับเทอร์มินอล ในการแสดงผลและรับส่งข้อมูล

เครื่องมือและอุปกรณ์

1. ซิงเกิลบอร์ด
2. เครื่องคอมพิวเตอร์ส่วนบุคคล
2. คอนเนคเตอร์ DB9 หรือ DB25 ดังรูปที่ 14 และรูปที่ 15
3. โปรแกรม Telix

การทดลอง

1. ต่อคอนเนคเตอร์ DB9 หรือ DB25 เข้ากับพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ ส่วนปลายอีกด้านหนึ่งต่อเข้ากับคอนเนคเตอร์ J5 (Serial 1) ของซิงเกิลบอร์ดในภาพที่ 24

2. เรียกโปรแกรม Telix จากเครื่องคอมพิวเตอร์ และรอจนคำว่า "Initializing modem" หายไป

3. กำหนดพารามิเตอร์ของโปรแกรม Telix ให้ตรงกับฮาร์ดแวร์ที่ต่อดังนี้

3.1) กด ALT-P จากนั้นจึงเลือกหมายเลขพอร์ตอนุกรมให้ตรงกับที่ต่ออยู่ด้วยคีย์ [1-8], เลือกความเร็วในการติดต่อ (บิตต่อวินาที) ด้วยคีย์ [A-I] ซึ่งหากผู้ใช้ไม่ได้เปลี่ยนความเร็วบนบอร์ดก็ให้เลือกความเร็ว 9600 บิตต่อวินาทีสำหรับการใช้งาน, กดปุ่ม O เลือกไม่มีพาริตี้บิต (None Parity) ,ข้อมูลขนาด 8 บิต ,Stop Bit 1 บิต และกด Enter

3.2) กดคีย์ ALT-O เลื่อนเมนูไปที่ "Ascii Tranfer" กด F เพื่อเปลี่ยนค่าของ "Line Pacing" ให้เปลี่ยนค่าเป็น 0 เพื่อไม่ให้มีการหน่วงเวลาขณะรับ/ส่งข้อมูลกัน และกด Enter เพื่อกลับเข้าสู่เมนู "Configure Telix" และกด W เพื่อบันทึก

4. ป้อนไฟให้กับบอร์ดด้วยอะแดปเตอร์ผ่านคอนเนคเตอร์ J1 ในภาพที่ 24 จนมีตัวอักษรปรากฏบนจอ LCD จากนั้นจึงกดคีย์ใดๆเพื่อเข้าสู่พร้อมท์ (Prompt) ซึ่งมีเครื่องหมาย '-' ปรากฏเพื่อรอรับคำสั่งมอนิเตอร์ในตารางที่ 14 ต่อไป

5. พิมพ์คำสั่ง HELP หรือ H หรือ ? ด้วยคีย์บนบอร์ดและสังเกตผล จากนั้นให้กดคีย์ใดๆ จนกว่าจะกลับมาที่พร้อมท์อีกครั้ง

6. พิมพ์คำสั่ง "CTTY" ด้วยคีย์บนบอร์ดและลองสังเกตผลบนจอ LCD และจอคอมพิวเตอร์

7. เมื่อเครื่องหมายพร้อมท์ปรากฏบนจอคอมพิวเตอร์แล้ว พิมพ์คำสั่ง H อีกครั้งด้วยคีย์บอร์ดบนเครื่องคอมพิวเตอร์ และสังเกตผล

8. เมื่อเครื่องหมายพร้อมท์ปรากฏบนจอคอมพิวเตอร์อีกครั้ง ให้พิมพ์คำสั่ง "H CTTY" ด้วยคีย์บนเครื่องคอมพิวเตอร์ และสังเกตผล

9. พิมพ์คำสั่ง "CTTY" ด้วยคีย์บนเครื่องคอมพิวเตอร์ และลองสังเกตผลบนจอ LCD และจอคอมพิวเตอร์อีกครั้ง

10. ใช้คำสั่ง "CTTY" เพื่อกลับไปที่จอและคีย์ของเครื่องคอมพิวเตอร์อีกครั้ง

11. พิมพ์คำสั่ง "? BAUD" ด้วยคีย์บนเครื่องคอมพิวเตอร์และสังเกตผล

12. พิมพ์คำสั่ง "BAUD" และสั่งเกตผล

13. พิมพ์คำสั่ง "BAUD 48" และลองกดคีย์ใดๆก็ได้ และสังเกตผล

14. กดปุ่ม ALT-P และปุ่ม D เพื่อเปลี่ยนความเร็วเป็น 4800 บิตต่อวินาที และกด Enter จากนั้นจึงกดคีย์ใดๆ และสังเกตผลบนจออีกครั้ง

15. พิมพ์คำสั่ง "BAUD 96" เพื่อกำหนดอัตราการรับส่งข้อมูลให้กลับเป็นปกติ

16. พิมพ์คำสั่ง "EDIT" เพื่อเข้าสู่โปรแกรมเอดิเตอร์ จากนั้นลองพิมพ์ข้อความใดๆ เข้าไปและทดลองใช้คำสั่งในตารางที่ 17 เพื่อช่วยในการแก้ไขข้อความ

17. กดคีย์ CTRL-KR จนปรากฏข้อความ "Waiting Data"

18. กดคีย์ ALT-S เลือกคำสั่ง ASCII และป้อนชื่อไฟล์ที่เป็นรหัสแอสกีความยาวไม่เกิน 16000 ไบต์ และกด Enter เพื่อให้ชื่อปรากฏบนช่อง "Tagged Files" จากนั้นจึงกดคีย์ F10 เพื่อส่งข้อมูล (Upload) จากเครื่องคอมพิวเตอร์ไปให้กับโปรแกรมเอดิเตอร์ จากนั้นจึงสังเกตผลบนจอ

19. ให้ทดลองแก้ไขข้อความที่รับมา แล้วใช้คีย์ CTRL-N เพื่อส่งการควบคุมและแสดงผลกลับไปที่ซิงเกิลบอร์ด และกลับไปกดคีย์ ALT-R บนเครื่องคอมพิวเตอร์เพื่อเข้าสู่เมนูการทำดาวน์โหลด (Download) เลือกคำสั่งไปที่ ASCII และใส่ชื่อไฟล์ 'board.txt'

20. กลับไปที่ซิงเกิลบอร์ดและกดคีย์ CTRL-KW บนบอร์ดจนปรากฏข้อความ "Press key to Upload" จากนั้นจึงกดคีย์ใดๆบนบอร์ด

21. ลบข้อความในเอดิเตอร์ด้วย CTRL-Y และกดคีย์ CRRL-KR บนบอร์ดเพื่อรอรับข้อความใหม่อีกครั้ง

22. กลับไปที่เครื่องคอมพิวเตอร์ และออกจากโปรแกรม Telix จนกลับมาที่ดอสพร้อมท์ (Dos Prompt) จากนั้นให้ใช้คำสั่ง "mode com...:96,n,8,1" และคำสั่ง "copy board.txt com..." และสังเกตุบนจอ LCD อีกครั้ง

<u>คำถามท้ายการทดลอง</u>

1. คำสั่ง H,HELP,? มีจุดประสงค์และวิธีการใช้อย่างไร

2. ซิงเกิลบอร์ดในโครงงานสามารถรองรับอัตราเร็วในการรับส่งข้อมูล (Baud Rate) ได้กี่ระดับ อะไรบ้าง

3. คำสั่ง CTRL-KR, CTRL-KW ในโปรแกรมเอดิเตอร์มีจุดประสงค์ในการใช้อย่างไร

5.2 ใบงานที่ 2
การเขียนโปรแกรม

<u>วัตถุประสงค์</u>

1. เพื่อให้ผู้ใช้มีทักษะพื้นฐานในการเขียนโปรแกรมใช้งานบนซิงเกิลบอร์ด

<u>เครื่องมือและอุปกรณ์</u>

1. ซิงเกิลบอร์ด
2. เครื่องคอมพิวเตอร์ส่วนบุคคล
2. คอนเนคเตอร์ DB9 หรือ DB25 ดังรูปที่ 14 และรูปที่ 15

<u>การทดลอง</u>

1. หลังจากเปิดเครื่อง และเข้าสู่มอนิเตอร์พร้อมท์ในโหมด LCD จึงเรียกคำสั่ง "DNLOAD 4000H -A" จนปรากฏข้อความ "Waiting Data" จากนั้นกลับไปที่เครื่องคอมพิวเตอร์ ซึ่งอยู่ในดอสพร้อมท์และเรียกคำสั่ง "mode com...:96,n,8,1" และ "copy lcdtst.asm com..." เพื่อส่งซอร์สโค้ด "lcdtst.asm" ในหน้าถัดไปให้กับบัฟเฟอร์ของโปรแกรมเอดิเตอร์ที่ตำแหน่ง 4000h
2. เข้าสู่โปรแกรมเอดิเตอร์ด้วยคำสั่ง "EDIT" และสังเกตผล
3. ใช้คำสั่ง CTRL-QX เพื่อออกจากเอดิเตอร์ และเรียกคำสั่ง "ASM" และสังเกตผล
4. เรียกคำสั่ง "G 8000h" และสังเกตผลบนจอ LCD
5. กดปุ่มรีเซตใหม่ และเข้าสู่โปรแกรมเอดิเตอร์
6. เปลี่ยนข้อความในบรรทัด "text1: db 'How do you do ?',0" เป็น
   "text1: db 'I am fine. ',0"
7. เรียกคำสั่ง "ASM" อีกครั้งและเรียกคำสั่ง "G 8000h" และสังเกตผลบน LCD
8. กดปุ่มรีเซตใหม่ เมื่อเข้าสู่มอนิเตอร์พร้อมท์ในโหมด LCD อีกครั้งจึงเรียกคำสั่ง "DNLOAD 8000H -H" จากนั้นกลับไปที่เครื่องคอมพิวเตอร์ซึ่งอยู่ในดอสพร้อมท์และเรียกคำสั่ง "copy lcdtst.hex com..." (เมื่อ lcdtst.hex คือไฟล์ที่เป็นคำสั่งภาษาเครื่องในรูป Intel Hex Format ที่ได้มาจากการแอสเซมเบลอไฟล์ lcdtst.asm)
9. เรียกคำสั่ง "G 8000h" อีกครั้ง และสังเกตผลบนจอ LCD
10. กดปุ่มรีเซตใหม่ เมื่อเข้าสู่มอนิเตอร์พร้อมท์ในโหมด LCD อีกครั้งจึงเรียกคำสั่ง "DNLOAD 8000H -B" จากนั้นกลับไปที่เครื่องคอมพิวเตอร์ซึ่งอยู่ในดอสพร้อมท์และเรียกคำสั่ง

"copy lcdtst.bin com..." เมื่อ lcdtst.bin คือไฟล์ที่เป็นคำสั่งภาษาเครื่องในรูปไบนารีที่ได้มาจากการแอสเซมเบลอไฟล์ lcdtst.asm

11. เรียกคำสั่ง "G 8000h" อีกครั้ง และสังเกตผลบนจอ LCD

```
;LCDTST.ASM
;It is downloaded to Single Board
;Asawin Srisetanil s6014563 STD4D
;Sirisak Jankasiri  s6014439 STD4D
ORG   08000H

port_wi equ 0FF0Ch        ;write instruction
port_wd equ 0FF0Dh        ;write data
port_ri equ 0FF0Eh        ;read  instruction    ; PCB Board
port_rd equ 0FF0Fh        ;read  data

r0_     EQU     00        ; address of R0 to R7
r1_     EQU     01
r2_     EQU     02
r3_     EQU     03
r4_     EQU     04
r5_     EQU     05
r6_     EQU     06
r7_     EQU     07

lcall delay               ;delay time
lcall lcdclr              ;clear screen
mov   dptr,#text1         ;get text address for display
lcall lcdwritest          ;display the text on LCD
sjmp  $                   ;stop running
```

```
text1: db 'How do you do ?',0


delay:    push r1_
          push r2_
          mov r1,#100
          mov r2,#10
          djnz r2,$
          djnz r1,$-4    ;to "mov r2,#10" command
          pop  r2_
          pop  r1_
          ret

lcdwi:    push dph                ;write instuction in acc to LCD
          push dpl
          push acc
          mov  dptr,#port_ri
          movx a,@dptr
          jb  acc.7,$-1
          pop acc
          mov dptr,#port_wi
          movx @dptr,a
          pop dpl
          pop dph
          ret
lcdwd;    push dph                ;write data in acc to LCD
          push dpl
          push acc
          mov  dptr,#port_ri
          movx a,@dptr
          jb  acc.7,$-1
```

```
        pop acc
        mov dptr,#port_wd
        movx @dptr,a
        pop dpl
        pop dph
        ret

blinkon:                    ;display on ,blink on
  mov a,#00001111b
  acall lcdwi
  ret

lcdclr:   mov a,#01h        ;clear screen (LCD)
          acall lcdwi
          ret

lcdhome:  mov a,#02h        ;move cursor to home
          acall lcdwi
          ret
lcdwritest:                 ;display string pointed by dptr on LCD
nextch:   clr a
          movc a,@a+dptr
          jz printst1
          push dph
          push dpl
          acall lcdwd
          pop  dpl
          pop  dph
          inc  dptr
          sjmp nextch
```

```
printst1: ret


  END                    ;End of User Program
```

<u>คำถามท้ายการทดลอง</u>

1. มีวิธีใดบ้างในการนำโปรแกรมของผู้ใช้ไปใช้งานบนซิงเกิลบอร์ด
2. คำสั่ง "ASM" ในมอนิเตอร์มีจุดประสงค์ในการใช้งานอย่างไร

5.3 ใบงานที่ 3
การดีบักโปรแกรม

<u>วัตถุประสงค์</u>

1. เพื่อให้ผู้ใช้ได้เรียนรู้วิธีการและคำสั่งที่จำเป็นในการดีบักโปรแกรม

<u>เครื่องมือและอุปกรณ์</u>

1. ซิงเกิลบอร์ด
2. เครื่องคอมพิวเตอร์ส่วนบุคคล
2. คอนเนคเตอร์ DB9 หรือ DB25 ดังรูปที่ 14 และรูปที่ 15
3. โปรแกรม Telix

<u>การทดลอง</u>

1. เมื่อเข้าสู่มอนิเตอร์พร้อมท์ เรียกคำสั่ง "EDIT"
2. ทดลองป้อนโปรแกรมต่อไปนี้เข้าไป และออกจากเอดิเตอร์กลับมาที่มอนิเตอร์พร้อมท์ เพื่อเรียกคำสั่ง "ASM"

```
    org 8000h
    mov  a,#055h
    mov  dptr,#data1

label1:
    movx @dptr,a
    inc  a
    sjmp label1

data1: ds 1
     end
```

3. เรียกคำสั่ง "RESET" เพื่อเคลียร์ (Clear) ค่าในรีจิสเตอร์ภายใน

4. เรียกคำสั่ง "U 8000h" และสังเกตผลเทียบกับ โปรแกรมที่ป้อนเข้าไป และบันทึกคำสั่งแรกไว้

5. เรียกคำสั่ง "R" และสังเกตผล

6. เรียกคำสั่ง "R ACC" สังเกตผล และบันทึกไว้

7. เรียกคำสั่ง "P 8000h" จากนั้นดูค่าของ ACC อีกครั้งด้วยคำสั่ง "R ACC" และบันทึกไว้เช่นกัน

8. เรียกคำสั่ง "U" เพื่อดูคำสั่งที่จะทำงานถัดไป บันทึกไว้

9 เรียกคำสั่ง "R" และบันทึกค่าของ DPTR

10 เรียกคำสั่ง "P" ใหม่ และเรียกคำสั่ง "R" อีกครั้ง และบันทึกค่าของ DPTR ใหม่เพื่อเทียบกับค่าเดิม

11. เรียกคำสั่ง "U" เพื่อดูคำสั่งที่จะทำงานถัดไป บันทึกไว้

10. เรียกคำสั่ง "D 8009h" บันทึกค่าในไบต์แรกไว้ และเรียกคำสั่ง "R ACC" บันทึกไว้เช่นกัน

11. เรียกคำสั่ง "T" และลองดูค่าในหน่วยความจำชนิดข้อมูลที่ตำแหน่ง 8009h อีกครั้งด้วยคำสั่ง "D 8009h" บันทึกไว้

12. เรียกคำสั่ง "CTTY" เพื่อเข้าสู่เทอร์มินอลโหมด

13. ดาวน์โหลดซอร์สโค้ดในใบงานที่ 2 มาอีกครั้ง และเรียกคำสั่ง "ASM" ใหม่ จากนั้นลองใช้คำสั่ง "P 8000h" ในครั้งแรกและ "P" ในครั้งถัดมาเรื่อยๆ 4-5 ครั้ง และสังเกตผลบน LCD และจอคอมพิวเตอร์

14. กดปุ่มรีเซตใหม่ จากนั้นลองใช้คำสั่ง "P 8000h" ในครั้งแรกและ "T" ในครั้งถัดมาเรื่อยๆ ประมาณ 4-5 ครั้ง และสังเกตผล เพื่อเทียบกับผลที่ได้จากข้อ 13

15. เรียกคำสั่ง "G" และสังเกตผล จากนั้นลองกดปุ่ม Break (S50) และสังเกตผลใหม่

16. เรียกคำสั่ง "RESET" เพื่อเคลียร์จอใหม่

17. เรียกคำสั่ง "G 8000h" อีกครั้งจากนั้นลองกดปุ่ม Break และสังเกตผลใหม่

18. เรียกคำสั่ง "RESET" เพื่อเคลียร์จอใหม่อีกครั้ง

19. เรียกคำสั่ง "G 8000h,800Ch" อีกครั้งสังเกตผลใหม่

<u>คำถามท้ายการทดลอง</u>

1. คำสั่ง "G", "P", "T" ต่างกันอย่างไร

2. สวิทช์ Break มีจุดประสงค์การใช้งานอย่างไร

บทที่ 6
สรุปและวิจารณ์

จากการออกแบบและทดลองทั้งทางฮาร์ดแวร์ ,และซอฟแวร์ ที่ได้กล่าวผ่านมาแล้วนั้น ทำให้ผู้ใช้งานสามารถนำซิงเกิลบอร์ดในโครงงานนี้ไปใช้ประโยชน์ได้โดยไม่ต้องอาศัยเครื่องมืออื่นๆ มากนัก เพราะนอกจากจะมีเครื่องมือหรือทูลที่จำเป็นเหมือนซิงเกิลบอร์ดทั่วไปในท้องตลาดแล้ว ยังมีเครื่องมือช่วยในการเขียนโปรแกรมภาษาแอสเซมบลีด้วยในตัว ทั้งเอดิเตอร์, แอสเซมเบลอ อีกทั้งมีการจัดวางคีย์ให้คล้ายคีย์บอร์ดบนคอมพิวเตอร์ส่วนบุคคลทั่วไปให้มากที่สุด เพื่อเพิ่มความสะดวกในการใช้งานและค้นหาตำแหน่งของคีย์ ทำให้การใช้งานแบบ Stand alone มีประสิทธิภาพมากยิ่งขึ้น

อย่างไรก็ตามเนื่องจากเวลาในการทำโครงงานนี้มีจำกัด ในขณะที่จำนวนเนื้องานที่ต้องทำก็มีมาก ทั้งการเขียนโปรแกรมที่เป็นภาษาแอสเซมบลี, การหาวิธีและทำการทดสอบโปรแกรม, การออกแบบโครงสร้างการเก็บข้อมูล, การออกแบบฮาร์ดแวร์, การทดสอบฮาร์ดแวร์, การหาซื้ออุปกรณ์ ฯลฯ ซึ่งมักจะเกิดปัญหาที่คาดไม่ถึงเสมอ และต้องใช้เวลานานกว่าจะพบและทำการแก้ไขข้อผิดพลาดได้ทั้งที่มีอยู่ในวงจรและตัวโปรแกรม จึงเป็นไปได้ที่อาจจะมีข้อผิดพลาดบางประการที่ทางผู้จัดยังทดสอบไปไม่ถึง ซึ่งทางผู้จัดทำต้องขออภัยล่วงหน้าและจะได้หาโอกาสดำเนินการแก้ไขต่อไป

ภาคผนวก ก

การใช้งานอุปกรณ์แสดงผลแบบผลึกเหลว LCD (Liquid Crystal Display)

ตัวแสดงผล LCD เป็นอุปกรณ์แสดงผลแบบหนึ่งที่นิยมใช้มาก และพบเห็นกันบ่อยๆ LCD สามารถแบ่งประเภทตามลักษณะของการแสดงผลได้ 3 แบบคือ LCD แบบอักขระ (Character LCD Module), แบบกราฟิก (Graphic LCD Module) และLCD แบบเซกเมนต์ (Segment LCD Module)

LCD แบบอักขระ เป็นโมดูล LCD ที่สามารถแสดงตัวอักษร,ตัวเลข,และเครื่องหมายต่างๆได้ โดยสร้างจากจุดเล็ก ๆ หรือเรียกว่า ดอตเมตริกซ์ ซึ่งก็จะมีขนาดความกว้างและสูงของอักขระแต่ละตัว โดยทั่วๆไปมี 2 ขนาดคือ 5x7 จุด 5x10 จุด นอกจากนั้น LCD แบบนี้สามารถแสดงข้อความได้ 1 บรรทัด หรือมากกว่าก็ได้ ขึ้นอยู่กับรุ่นของ LCD นั้นๆ

ขาสัญญาณของโมดูล LCD แบบอักขระ

โมดูล LCD แบบอักขระที่นำมาเป็นตัวอย่างนี้เป็นแบบ 1 บรรทัด 16 ตัวอักษร มีขาต่อใช้งานทั้งสิ้น 14 ขา ได้แก่

Vss (ขา 1): ต่อกราวน์

Vdd (ขา 2): ต่อไฟเลี้ยง +5 โวลต์

Vo (ขา 3): เป็นขาอินพุตสำหรับป้อนแรงดัน เพื่อปรับความเข้มของการแสดงผล

RS (ขา 4): เป็นขาอินพุต ใช้เลือกว่าข้อมูลที่ทำการส่งในขณะนั้นเป็นข้อมูลคำสั่งสำหรับรีจิสเตอร์คำสั่งของ LCD หรือเป็นข้อมูลสำหรับรีจิสเตอร์ข้อมูลแสดงผลของ LCD โดย

- ถ้า RS เป็น "0" ข้อมูลที่ส่งมาจะเป็นข้อมูลคำสั่ง (instruction register)
- ถ้า RS เป็น "1" ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล (data register)

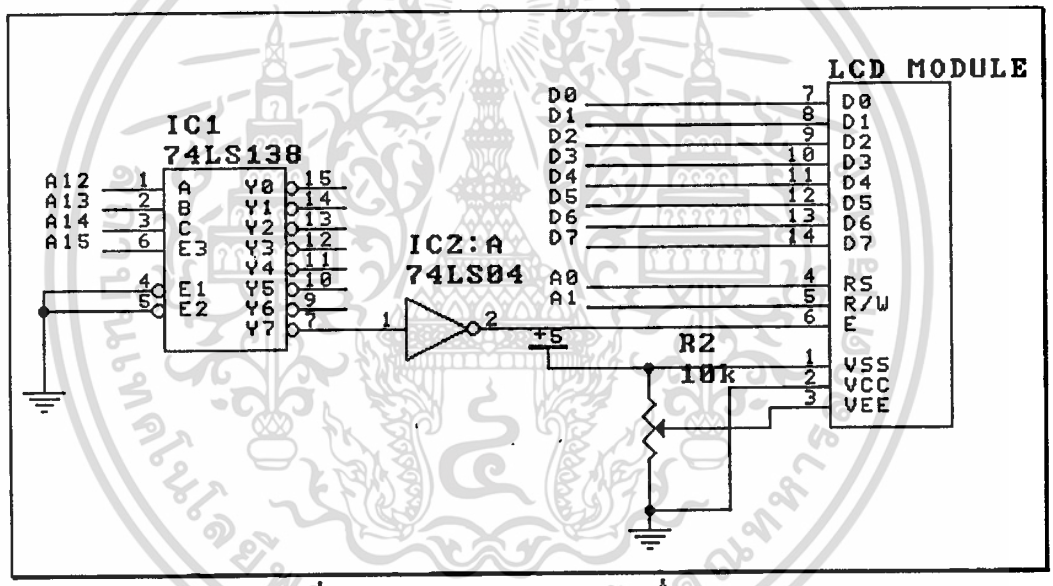
$R/\overline{W}$ (ขา 5): เป็นขาที่ใช้เลือกว่าจะอ่านหรือเขียนข้อมูลกับ LCD ถ้าหากเป็น "0" จะเป็นการเขียนข้อมูล แต่ถ้าเป็น "1" จะเป็นการอ่านข้อมูล

E (ขา 6): เป็นขา Enable ให้ LCD ทำงาน

DB0-DB7 (ขา 7-14): เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอก ขนาด 8 บิต

RS	$R/\overline{W}$	E	การทำงาน
0	0	↘	เขียนคำสั่ง
0	1	⌒	อ่านสถานะของ LCD
0	0	↘	เขียนข้อมูล
0	1	⌒	อ่านข้อมูล

ตารางที่ ก.1 ตารางแสดงความสัมพันธ์ของสัญญาณ E, RS, และ $R/\overline{W}$ ของ LCD



ภาพที่ ก.1 ภาพแสดงตัวอย่างการเชื่อมต่อ LCD

Instruction	RS	RW	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0	Description
Clear display	0	0	0	0	0	0	0	0	0	1	Clear all display and returns the cursor to the home position
Return home	0	0	0	0	0	0	0	0	1	x	Return the cursor to the home position.DD RAM contents remain unchanged
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor ON/OFF (C), blink of cursor position character (B)
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	x	x	Moves the cursor and shifts the display without changing DD RAM contents
Function set	0	0	0	0	1	DL	N	F	x	x	Sets interface data length (DL), number of display lines (L), character font (F)

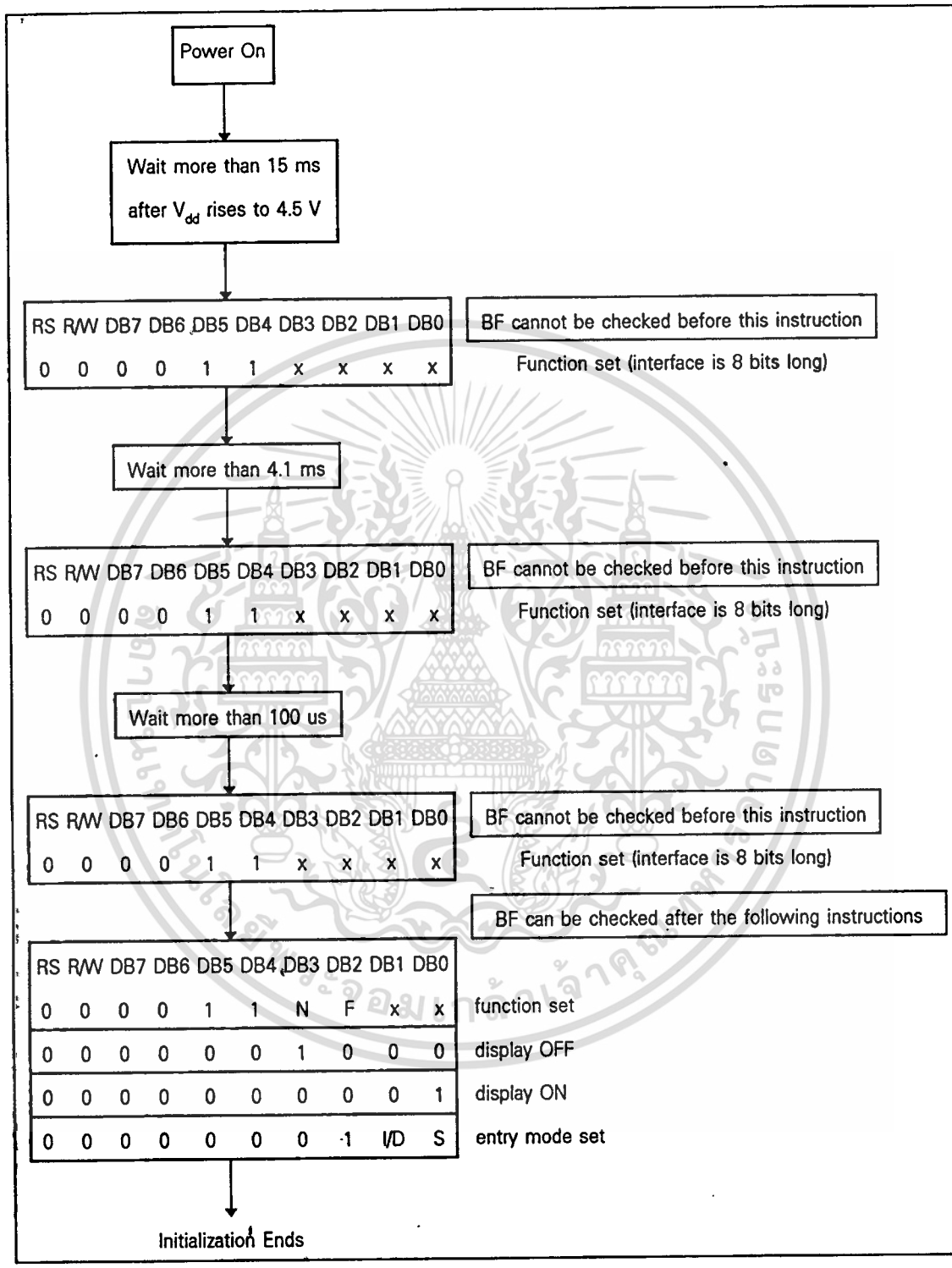
ตารางที่ ก.2 ตารางคำสั่งควบคุม LCD

Instruction	RS	RW	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0	Description
Set CG RAM address	0	0	0	1	A_{CG}						Sets the CG RAM address.
Set DD RAM address	0	0	1	A_{DD}							Sets the DD RAM address
Read busy flag & address	0	1	BF	AC							Read Busy flag (BF) indicating internal operation is being performed and reads address counter contents
Write data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM
Read data to CG or DD RAM	1	1	Read Data								Reads data into DD RAM or CG RAM

I/D = 1 : Increment (+1) I/D = 0 : Decrement (-1)	DD RAM : Display data RAM
S = 1 : Accumulator display shift	CG RAM : Character generator RAM
S/C = 1 : Display shift S/C = 0 : Cursor move	A_{CG} : CG RAM address
R/L = 1 : Shift to the right R/L = 0 : Shift to the left	A_{DD} : DD RAM address
DL = 1 : 8 bits DL = 0 : 4 bits	Corressponds to cursor address
N = 1 : 2 lines N = 0 : 1 line	AC : Address counter used for both of
F = 1 : 5*10 dots F = 0 : 5*7 dots	DD and CG RAM address
BF = 1 : Internally operating	
BF = 0 : Can accept instruction	

ตารางที่ ก.2 (ต่อ)

จากภาพที่ ก.2 ในกวรเริ่มต้นใช้งาน เพื่อให้ LCD สามารถทำงานได้ถูกต้องจำเป็นต้องมีการรีเซตใหม่เสียก่อนทุกครั้งที่เริ่มจ่ายไฟให้เช่นเดียวกับชิปไมโครโปรเซสเซอร์ แต่การรีเซตจะเป็นการรีเซตด้วยคำสั่งทางซอฟแวร์



ภาพที่ ก.2 ภาพแสดงแผนผังวิธีการรีเซต LCD ด้วยคำสั่ง

ภาคผนวก ข

การใช้งานชิปฐานเวลาจริง 6246 (Real Time Clock)

สรุปการใช้งาน

1) ชิป RTC(Real Time Clock) เป็นชิปที่ใช้เพื่อบอกเวลาจริงให้กับระบบ ในการเริ่มต้นใช้งานนั้นรีจิสเตอร์ F จะถูกสั่งงานก่อนโดยการ Set Mode 24/12 เสียก่อน กล่าวคือ

24/12 bit = 1 -> 24 ชม.

24/12 bit = 0 -> 12 ชม.

ส่วน Stop และ Rest ให้ทำงานโดย Set เป็น 1

2) ตั้งโหมดการ Interrupt โดยกำหนดค่าในบิต T_1, T_0 ดังตารางที่ ข.1

T_1	T_0	คาบเวลาการส่งสัญญาณ Interrupt จากขา 1
0	0	7.8125 มิลลิวินาที
0	1	1 วินาที
1	0	1 นาที
1	1	1 ชั่วโมง

ตารางที่ ข.1 ตารางแสดงการตั้งโหมดการส่งสัญญาณอินเตอร์รัป

และให้ Mask bit เป็น 0 คือ enable สัญญาณ O/P ไป Interrupt CPU ผ่านขาสัญญาณ STDP โดยมีคาบการเกิดสัญญาณตาม T_1/T_0

3) เมื่อ set ทุกอย่างถูกต้องและต้องการสั่งให้ RTC เริ่มการทำงาน ให้ Clear Bit Stop และ Rest ในรีจิสเตอร์ F ของ RTC

4) เมื่อเกิด Interrupt ขึ้น ในส่วนโปรแกรม Interrupt ให้ทำการ Clear สัญญาณ O/P ของ RTC โดยส่งค่า 0 ไปที่ Bit IRQ ในรีจิสเตอร์ D

Address Input	Address Input				Registet Name	Data				Count Value	Descripttion
	D_1	D_0	A_1	A_0		D_3	D_2	D_1	D_0		
0	0	0	0	0	S_1	S_8	S_4	S_2	S_1	0~9	1-second digit
1	0	0	0	1	S_{10}	*	S_{40}	S_{20}	S_{10}	0~5	10-second digit
2	0	0	1	0	MI_1	mi_8	mi_4	mi_2	mi_1	0~9	1-minute digit
3	0	0	1	1	MI_{10}	*	mi_{40}	mi_{20}	mi_{10}	0~5	10-minute digit
4	0	1	0	0	H_1	h_8	h_4	h_2	h_1	0~9	1-hour digit
5	0	1	0	1	H_{10}	*	PM/ AM	h_{20}	h_{10}	0~2 or 0~1	10-hour digit
6	0	1	1	0	D_1	d_8	d_4	d_2	d_1	0~9	1-day digit
7	0	1	1	1	D_{10}	*	*	d_{20}	d_{10}	0~3	10-day digit
8	1	0	0	0	MO_1	mo_8	mo_4	mo_2	mo_1	0~9	1-month digit
9	1	0	0	1	MO_{10}	*	*	*	mo_{10}	0~9	10-month digit
A	1	0	1	0	Y_1	y_8	y_4	y_2	y_1	0~9	1-year digit
B	1	0	1	1	Y_{10}	y_{80}	y_{40}	y_{20}	y_{10}	0~9	10-year digit
C	1	1	0	0	W	*	w_4	w_2	w_1	0~6	week digit
D	1	1	0	1	C_D	30 sec. ADJ	IRQ	Busy	Hold	-	Control register D
E	1	1	1	0	C_E	t_1	t_0	Itrpt /Stnd	Mas k	-	Control register E
F	1	1	1	1	C_F	Test	24 / 12	Stop	Rest	-	Control register F

Rest=Reset

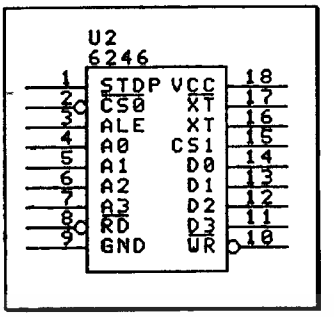
Itrpt/Stnd=Interrupt/Standard

Note 1) Bit * does not exit (unrecognized during a write and held at "0" during a read).

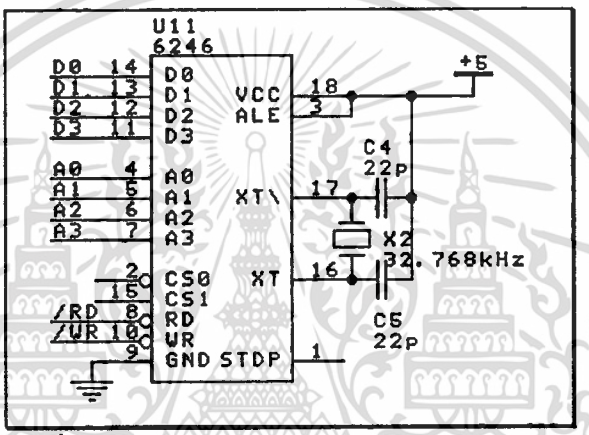
Note 2) Be sure to mask the AM/PM bit when processing 10's of hour's data.

Note 3) Busy bit is read only. The IRQ flag bit can only be set to a "0"

ตารางที่ ข.2 ตารางแสดงตำแหน่งรีจิสเตอร์ภายในชิปฐานเวลาจริง



ภาพที่ ข.1 ภาพแสดงขาสัญญาณของ 6246



ภาพที่ ข.2 ภาพแสดงวงจรตัวอย่างการใช้งาน 6246

ภาคผนวก ค

การใช้งาน MAX232

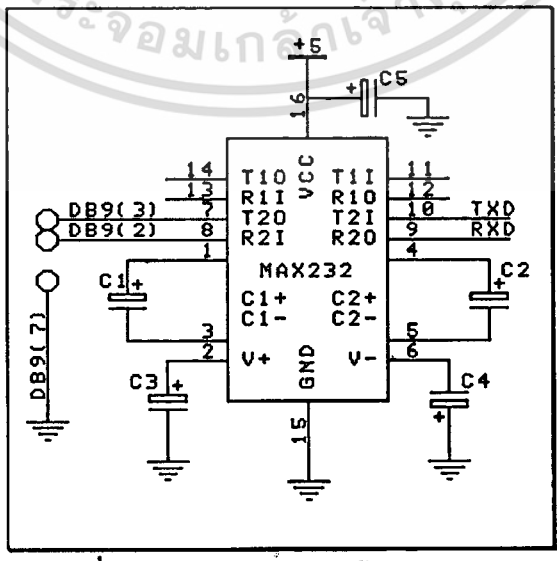
ในการเชื่อมต่อแบบอนุกรมเข้ากับอุปกรณ์คอมพิวเตอร์ต่างๆ มักจะกำหนดใช้การเชื่อมต่อตามมาตรฐาน RS-232C ทั้งนี้เพื่อให้มีการใช้งานเส้นสัญญาณหรือรูปแบบของตัวเชื่อม ที่สอดคล้องกัน

แต่เนื่องจากระดับแรงดันที่ใช้และการแทนความหมายของระดับลอจิกตามมาตรฐานนี้แตกต่างไปจากที่ใช้งานกันในระบบดิจิตอลทั่วไป โดยระดับสัญญาณของ RS-232C เป็นแบบไบโพลาร์ (Bipolar) กล่าวคือ

- ระดับแรงดัน -3 V ถึง -20 V แทนค่าลอจิก 1
- ระดับแรงดัน +3 V ถึง +20 V แทนค่าลอจิก 0

จะเห็นได้ว่ามีความจำเป็นต้องเพิ่มเติมอุปกรณ์หรือวงจรพิเศษเข้าไปเพื่อเปลี่ยนระดับแรงดันจากระบบ 0 ถึง +5 V จากขาสัญญาณของ 8051 ให้เป็นไปตามมาตรฐาน RS-232C

โดยทั่วไปรูปแบบของวงจรสามารถทำได้ในหลายลักษณะ ทั้งการออกแบบสร้างวงจรโดยการใช้ทรานซิสเตอร์ หรือใช้ไอซีวงจรรวมที่เรียกว่า ไดรเวอร์ ได้แก่เบอร์ MAX232 ซึ่งมีตัวอย่างการใช้งานดังภาพที่ ค.1



ภาพที่ ค.1 ภาพแสดงการใช้งาน MAX232

ในการใช้งาน MAX232 ให้เชื่อมต่อตัวเก็บประจุที่ขา C1, C2, ระหว่าง V- กับ GND, และระหว่าง V+ กับ GND ดังรูป ค.1 โดยค่าของตัวเก็บประจุที่ใช้ควรจะมีค่าประมาณ 10 uF สำหรับ MAX232 และ 100 nF สำหรับ MAX232A

pin	name function
1	c1+ \
3	c1- > +5v to +10v voltage doubler
2	V+ (10V) /
4	c2+ \
5	c2- > +10V to -10V voltage inverter
6	V- (-10V) /
15	GND
16	VCC (5V)
11	T1in \
14	T1out \ TTL/CMOS inputs to
10	T2in / RS-232 outputs
7	T2out /
12	R1out \
13	R1in \ RS-232 inputs to
9	R2out / TTL/CMOS outputs
8	R2in /

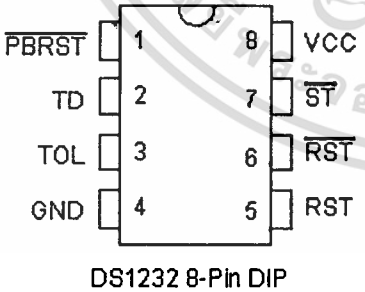
ตารางที่ ค.1 ตารางแสดงหน้าที่ของขาสัญญาณต่างๆใน MAX232

ภาคผนวก ง

DS1232 ไมโครมอนิเตอร์ชิป (Micro Monitor Chip)

FEATURES

- Halts and restarts an out-of-control microprocessor
- Holds microprocessor in check during power transients
- Automatically restarts microprocessor after power failure
- Monitors pushbutton for external override
- Accurate 5% or 10% microprocessor power supply monitoring
- Eliminates the need for discrete components
- Space-saving, 8-pin mini-DIP
- Optional 16-pin SOIC surface mount package
- Industrial temperature -40°C to +85°C available, designated N PIN ASSIGNMENT



DS1232 8-Pin DIP

$\overline{PBRST}$ – Pushbutton Reset Input

TD – Time Delay Set

TOL – Selects 5% or 10% V CC Detect

GND – Ground RST – Reset Output (Active High)

$\overline{RST}$ – Reset Output (Active Low, open drain)

$\overline{ST}$ – Strobe Input V CC – +5 Volt Power

NC – No Connections

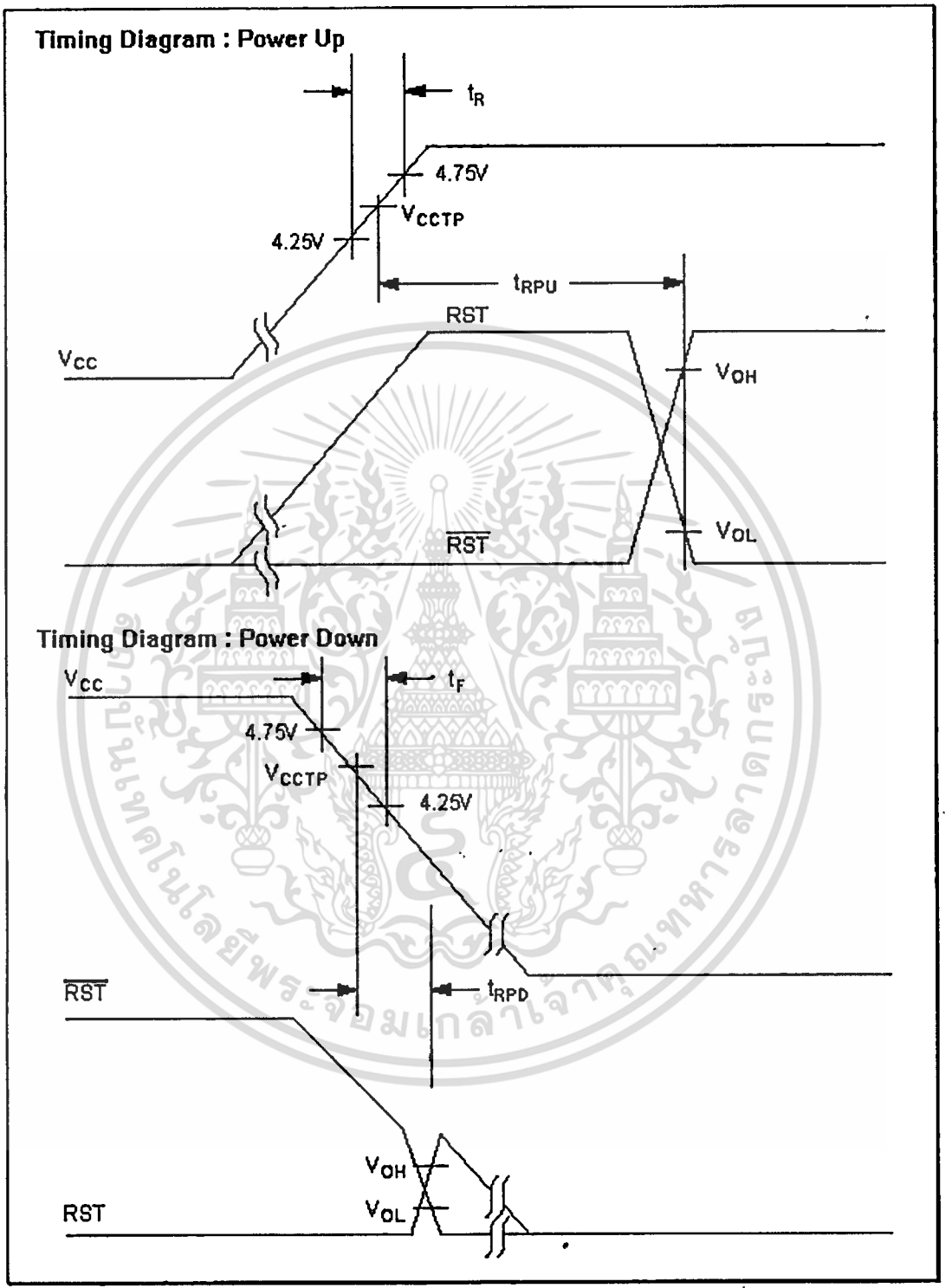
ภาพที่ ง.1 ภาพแสดงขาสัญญาณใช้งาน DS1232

DESCRIPTION

The DS1232 MicroMonitor Chip monitors three vital conditions for a microprocessor: power supply, software execution, and external override. First, a precision temperature-compensated reference and comparator circuit monitors the status of V_{CC} . When an out-of-tolerance condition occurs, an internal power fail signal is generated which forces reset to the active state. When V_{CC} returns to an in-tolerance condition, the reset signals are kept in the active state for a minimum of 250 ms to allow the power supply and processor to stabilize.The second function the DS1232 performs is pushbutton reset control. The DS1232 debounces the pushbutton input and guarantees an active reset pulse width of 250ms minimum. The third function is a watchdog timer. The DS1232 has an internal timer that forces the reset signals to the active state if the strobe input is not driven low prior to time-out. The watchdog timer function can be set to operate on time-out settings of approximately 150 ms, 600 ms, and 1.2 seconds.

OPERATION - POWER MONITOR

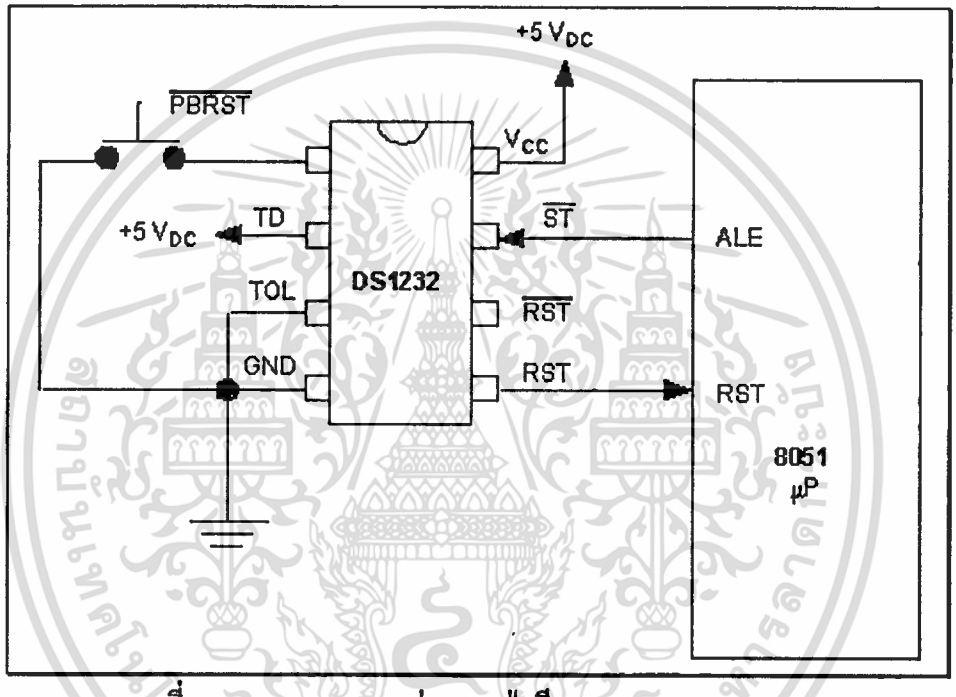
The DS1232 detects out-of-tolerance power supply conditions and warns a processor-based system of impending power failure. When V_{CC} falls below a preset level as defined by TOL (Pin 3), the V_{CC} comparator outputs the signals RST (Pin 5) and $\overline{RST}$ (Pin 6). When TOL is connected to ground, the RST and $\overline{RST}$ signals become active as V_{CC} falls below 4.75 volts. When TOL is connected to V_{CC} , the RST and $\overline{RST}$ signals become active as V_{CC} falls below 4.5 volts. The RST and $\overline{RST}$ are excellent control signals for a microprocessor, as processing is stopped at the last possible moments of valid V_{CC} . On power-up, RST and $\overline{RST}$ are kept active for a minimum of 250 ms to allow the power supply and processor to stabilize.



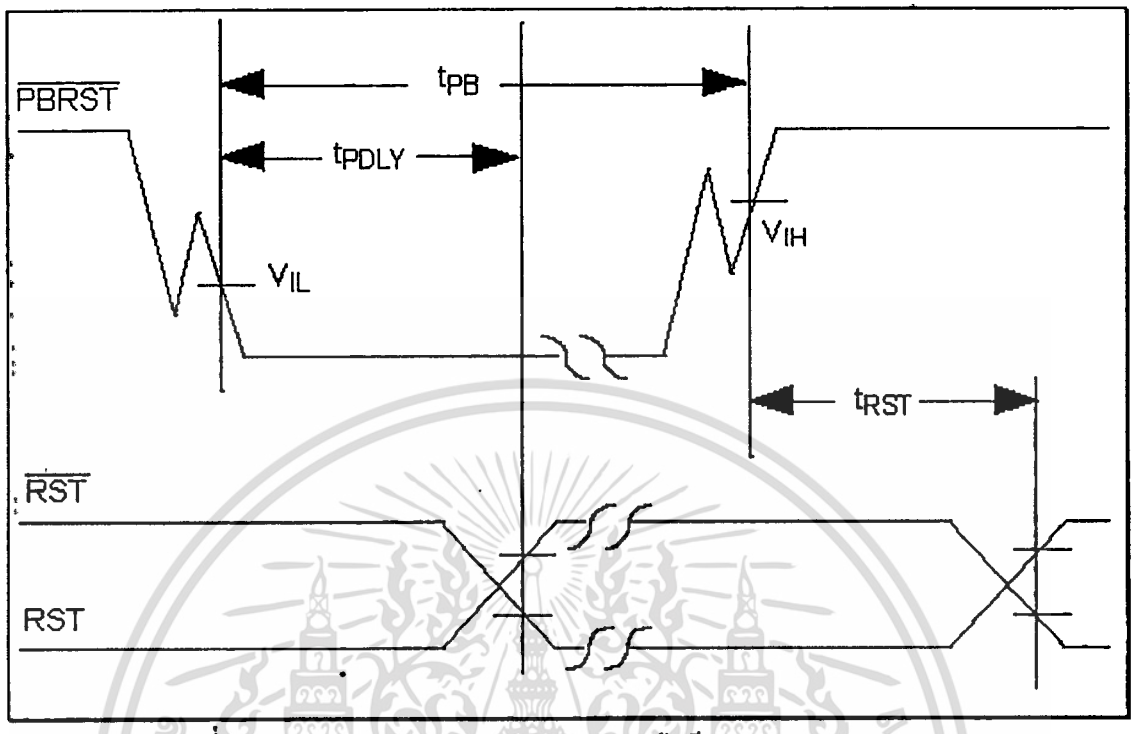
ภาพที่ ง.2 ภาพแสดงผังเวลาการทำงานช่วงปิดเปิดการทำงาน (Power Up/Power Down).

OPERATION - PUSHBUTTON RESET

The DS1232 provides an input pin for direct connection to a pushbutton (Figure 3.3). The pushbutton reset input requires an active low signal. Internally, this input is debounced and timed such that RST and $\overline{RST}$ signals of at least 250 ms minimum are generated. The 250 ms delay starts as the pushbutton reset input is released from low level.



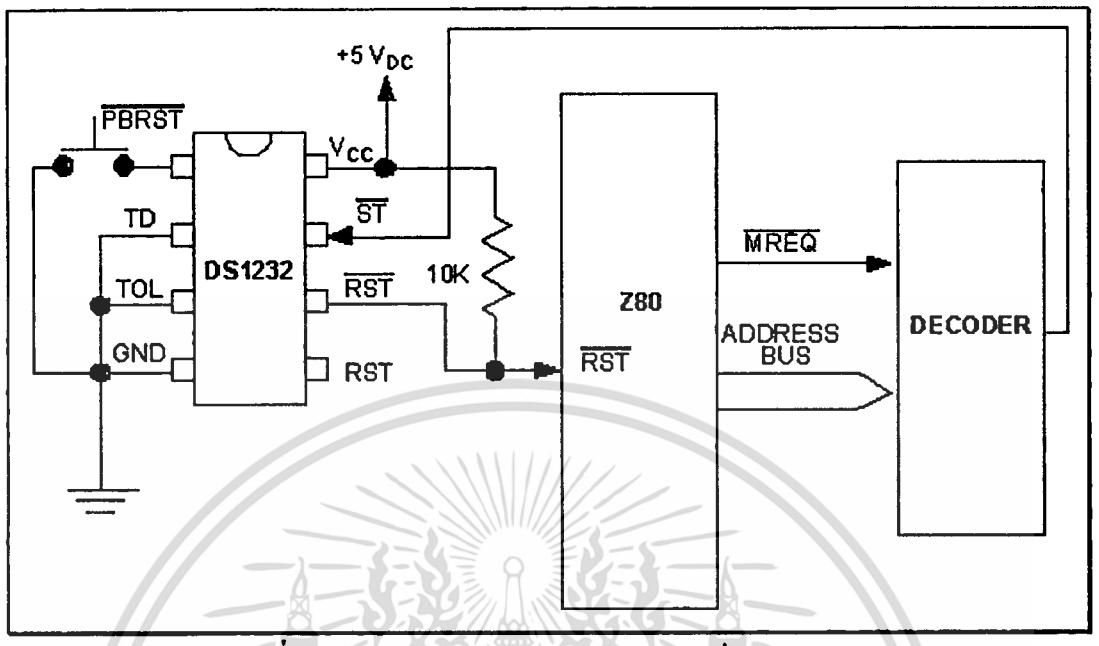
ภาพที่ 3.3 ภาพแสดงการต่อวงจรปุ่มรีเซต (Push Button Reset)



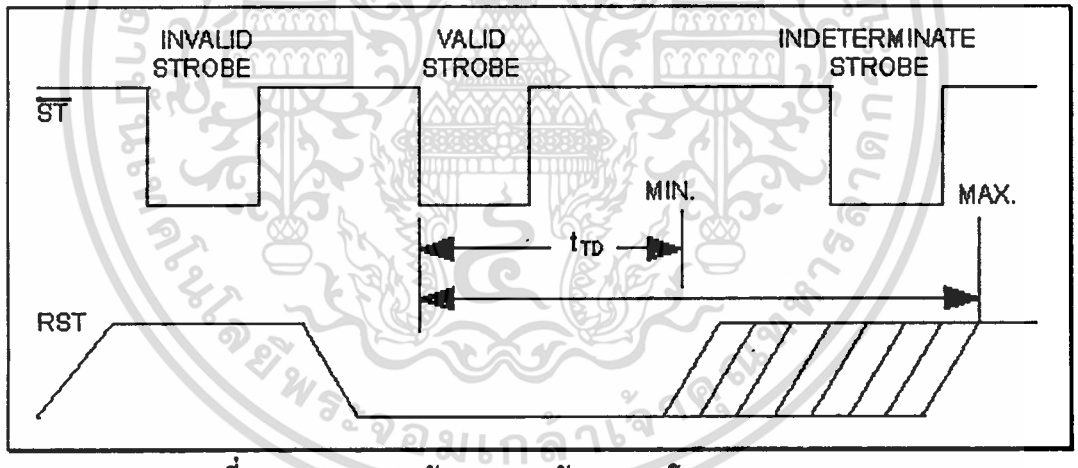
ภาพที่ 3.4 ภาพแสดงผังเวลาการทำงานของปุ่มรีเซต (Push Button Reset)

OPERATION - WATCHDOG TIMER

A watchdog timer function forces RST and $\overline{RST}$ signals to the active state when the $\overline{ST}$ input is not stimulated for a predetermined time period. The time period is set by the TD input to be typically 150 ms with TD connected to ground, 600 ms with TD left unconnected, and 1.2 sec-onds with TD connected to V_{CC} . The watchdog timer starts timing out from the set time period as soon as RST and $\overline{RST}$ are inactive. If a high-to-low transition occurs on the $\overline{ST}$ input pin prior to time-out, the watchdog timer is reset and begins to time-out again. If the watchdog timer is allowed to time-out, then the RST and $\overline{RST}$ signals are driven to the active state for 250 ms minimum. The $\overline{ST}$ input can be derived from microprocessor ad-dress signals, data signals, and/or control signals. When the microprocessor is functioning normally, these signals would, as a matter of routine, cause the watch-dog to be reset prior to time-out. To guarantee that the watchdog timer does not time-out, a high-to-low transi-tion must occur at or less than the minimum shown in Table 3.1. A typical circuit example is shown in Figure 3.5.



ภาพที่ ง.5 ภาพแสดงการต่อวงจรวอทดอกซ์ (WatchDog)



ภาพที่ ง.6 ภาพแสดงผังเวลาของสัญญาณสโตรบ (Strobe Input)

TD PIN	TIME-OUT		
	MIN	TYP	MAX
GND	62.5 ms	150 ms	250 ms
Float	250 ms	600 ms	1000 ms
VCC	500 ms	1200 ms	2000 ms

ตารางที่ ง.1 ตารางแสดง WatchDog Time-Out

Notes

1. PBRST is internally pulled up to V_{CC} with an internal impedance of 10K typical.
2. RST remains within 0.5V of V_{CC} on power-down until V_{CC} drops below 2.0V.
   $\overline{RST}$ remains within 0.5V of GND on power-down until V_{CC} drops below 2.0V.
3. Watchdog can not be disabled. It must be strobed to avoid resets.

กิตติกรรมประกาศ

การที่โครงงานนี้สามารถสำเร็จลงไปได้ด้วยดีนั้น ทั้งนี้ผลสืบเนื่องมาจากความรู้พื้นฐานหลายๆแขนงที่ได้เรียนมา ไม่ว่าจะเป็นวิชา Microprocessor Interfacing, Data Stucture and Algorithms, Compiler Construction ฯ รวมทั้งคำแนะนำต่างๆที่ได้รับจากอาจารย์ในภาควิชาคอมพิวเตอร์ทุกๆท่าน โดยเฉพาะอาจารย์ที่ปรึกษาคือ อาจาย์สมศักดิ์ มิตะถา ที่คอยให้คำชี้แนะที่ประโยชน์ต่อโครงงานนี้อย่างมาก และขอขอบคุณพี่เต๊ียะ พี่รุ่น 28 ที่คอยช่วยเหลือจัดหาอุปกรณ์ต่างๆที่จำเป็น และคอยให้กำลังใจจนงานสำเร็จลุล่วงไปได้

และเพื่อให้โครงงานดังกล่าวสามารถนำไปใช้ให้เกิดประโยชน์ได้จริง ผู้จัดทำจึงขอมอบซิงเกิลบอร์ดพร้อมคู่มือใช้งานและตัวอย่างโปรแกรม ให้เป็นสมบัติของภาควิชาเพื่อใช้ประกอบการเรียน, การค้นคว้าพัฒนา ให้เกิดประโยชน์ยิ่งๆขึ้นไปในอนาคต

เอกสารอ้างอิง

1. แผนกหนังสือพิเศษด้านอิเลคทรอนิคส์ , "เข้าใจ/สร้าง/เล่น ไมโครโปรเซสเซอร์ เล่ม 1" , บ. ซีเอ็ดยูเคชัน จำกัด (มหาชน) , 219 หน้า , 2538

2. สุนทร วิทูสุรพจน์ , "การใช้งานไมโครคอนโทรลเลอร์ ตระกูล 8051" , บ. ซีเอ็ดยูเคชัน จำกัด (มหาชน) , 180 หน้า , 2537

3. ปรเมษฐ์ ประนยานันทน์ , "คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51" บ. ซีเอ็ดยูเคชัน จำกัด (มหาชน) , 380 หน้า , 2536

4. ทีมงาน ETT , "ET-BOARD V4.0 New Generation USER' S MANUAL" , บริษัท อีทีที จำกัด , 282 หน้า , 2535

5. Kenneth J. Ayala , "The 8051 Microcontroller Architecture,Programming and Application" , West Publishing company , 241 p. , 1991