ระบบตรวจจับเวลา 4 ช่องทางโดยไมโครโปรเซสเซอร์

4 CHANNEL-TIMER SYSTEM BY MICROPROCESSOR

โดย

นายนรา ตันตราธิษฐาน รหัส 38014226

นายนิภัทร โรจน์รัตนวาณิชย์ รหัส 38014244

นายบรรพต เปี่ยมถาวรพจน์ รหัส 38014255

อาจารย์ที่ปรึกษา

ผศ. จิรวัฒน์ ปานกลาง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ ปีการศึกษา 2541

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบตรวจจับเวลา 4 ช่องทางโดยไมโครโปรเซสเซอร์

ผู้จัดทำ

1. นายนรา ตันตราธิษฐาน รหัส 38014226
2. นายนิภัทร โรจน์รัตนวาณิชย์ รหัส 38014244
3. นายบรรพต เปี่ยมถาวรพจน์ รหัส 38014255

..... อาจารย์ที่ปรึกษา

(ผศ. จิรวัฒน์ ปานกลาง)

ชื่อโครงงานภาษาไทย ระบบตรวจจับเวลา 4 ช่องทางโดยไมโครโปรเซสเซอร์

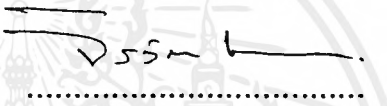
ชื่อโครงงานภาษาอังกฤษ 4 CHANNEL TIMER SYSTEM BY MICROPROCESSOR

นายนรา ตันตราธิษฐาน รหัส 38014226

นายนิภัทร โรจน์รัตนวาณิชย์ รหัส 38014244

นายบรรพต เปี่ยมถาวรพจน์ รหัส 38014255

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้



.....

(ผศ.จิรวัฒน์ ปานกลาง)

อาจารย์ที่ปรึกษา

ระบบตรวจจับเวลา 4 ช่องทางโดยใช้ไมโครโปรเซสเซอร์

นายนิภัทร โรจน์รัตนวาณิชย์
นายนรา ตันตราธิษฐาน
นายบรรพต เปี่ยมถาวรพจน์
ผศ.จิรวัฒน์ ปานกลาง (อาจารย์ที่ปรึกษา)
ภาคการศึกษาที่ 1 ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการประยุกต์ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51ในการใช้งานเป็นเครื่องตรวจจับเวลาโดยใช้ไทม์เมอร์ (timer) ของ MCS-51 เป็นฐานเวลา และทำการบันทึกช่วงเวลาระหว่างตัวตรวจจับแต่ละจุดทั้งหมด 4 จุด โดยใช้ชุดรับส่งอินฟราเรด (infrared sensor) เป็นตัวส่งและเป็นตัวรับในการตรวจจับการตัดผ่าน และส่งสัญญาณที่ได้ในการตัดผ่านส่งให้กับ MCS-51 ซึ่งทำการออกแบบในรูปของการ์ดเชื่อมต่อกับคอมพิวเตอร์ (interface card) ผ่านทางสล็อต IBM PC โดยพีซีสามารถส่งชุดคำสั่งเพื่อควบคุมการทำงานของ MCS-51 และรับข้อมูลที่เป็นผลต่างเวลาของแต่ละจุดการตรวจจับจาก MCS-51 เพื่อนำไปประมวลผล และแสดงผลในการประยุกต์ใช้ในหลาย ๆ รูปแบบ

ตัวอย่าง Application ต่างๆ

4 CHANNEL TIMER SYSTEM BY MICROPROCESSOR

Mr.Nipat Rojrattanavanich

Mr.Nara Tuntratisthan

Mr.Bunphod Phaimthawornphod

Asst.Prof.Jirawat Panklang (Advisor)

1st Semester,Educational Year 1998

Abstract

This thesis is an application of microcontroller by using MCS-51 family to operate as timer. Four set of infrared LEDs and photo transistors are used as object detector. A pulse is generated from each detector. Time between these pulses a is measured by MCS-51 software based timer which is designed on an interface card. Data from MCS-51 be transmitted through IBM PC slot to computer. These data and their their position can be processed and displayed by computer; moreover , computer can also this system by sending command words to MCS-51 such as start-stop timer and reset . This system can be developed into many application and controlled by windows software.

สารบัญ

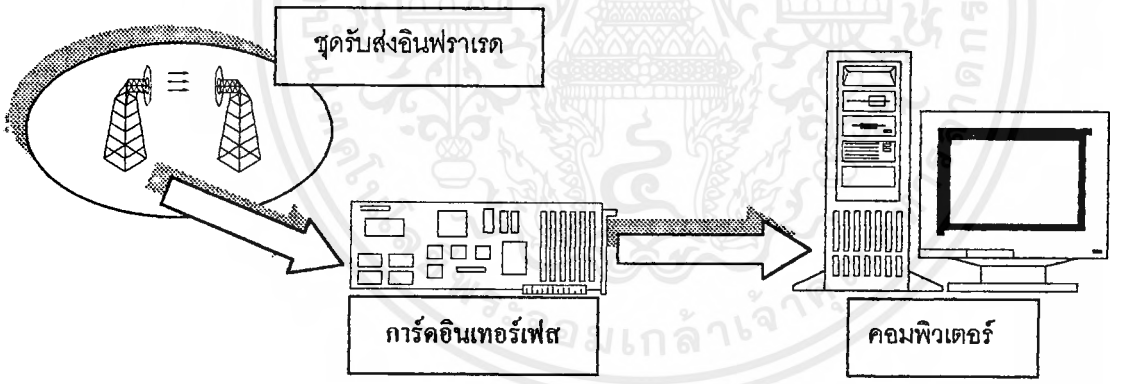
สารบัญรูป

บทที่ 1
บทนำ

1.1 จุดประสงค์และลักษณะของโครงงาน

โครงงานระบบตรวจจับเวลา 4 ช่องทางโดยไมโครโปรเซสเซอร์นี้มีจุดประสงค์หลักเพื่อทำการตรวจจับเวลา โดยจะวัดเวลาผลต่างระหว่างการตัดผ่านชุดรับส่งอินฟราเรดในแต่ละจุด การจับเวลาทำโดยไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 89C51 โดยใช้ไทม์เมอร์ (timer) ของไมโครคอนโทรลเลอร์เป็นฐานเวลา และส่งข้อมูลให้กับคอมพิวเตอร์เพื่อทำการแสดงผล ประมวลผลและประยุกต์ใช้งาน อื่น ๆ ต่อไป

โครงงานนี้เป็นเสมือนการใช้ไมโครคอนโทรลเลอร์ทำหน้าที่แทนนาฬิกา โดยจะรวมอยู่ในลักษณะของการ์ดอินเทอร์เฟส ไมโครคอนโทรลเลอร์เบอร์ 89C51 นี้จะมีทั้งหน่วยความจำสำหรับเก็บโปรแกรม (ROM) และหน่วยความจำสำหรับเก็บข้อมูล (RAM) ภายในชิปอยู่จำนวนหนึ่ง โดยจะ



รูป 1.1 แผนภาพระบบตรวจจับเวลา โดยใช้ไมโครคอนโทรลเลอร์

ทำการเก็บข้อมูลที่เป็นผลต่างระหว่างเวลาการตัดผ่านชุดรับส่งอินฟราเรดไว้ภายในหน่วยความจำภายในนี้และจะทำการส่งข้อมูลที่เก็บไว้ เมื่อคอมพิวเตอร์ต้องการเพื่อการแสดงผลและประยุกต์ใช้งานอื่น การใช้ระบบตรวจจับเวลานี้เพื่อความสะดวกให้ง่ายต่อการใช้งานและควบคุมจะกระทำภายใต้การทำงานของระบบวินโดว์ (window) แผนภาพการทำงานของระบบหลักเป็นดังรูป 1.1

1.2 ส่วนประกอบหลักของระบบ

จากแผนภาพของระบบการทำงานประกอบด้วย 3 ส่วนหลัก ประกอบด้วย ส่วนแรก ได้แก่ชุดรับส่งอินฟราเรดโดยมี หลอดอินฟราเรด (infrared lamp) เป็นตัวส่งในรูปของรังสีอินฟราเรดและใช้โฟโต้ทรานซิสเตอร์ (photo transistor) เป็นตัวรับรังสีอินฟราเรดที่ส่งมาจากหลอดอินฟราเรด ในการเลือกตัวส่งและตัวรับ (หลอดอินฟราเรดและโฟโต้ทรานซิสเตอร์) นั้นต้องเลือกให้มีความเหมาะสมเพื่อให้สามารถทำงานร่วมกันได้ ส่วนที่สองได้แก่ส่วนของไมโครคอนโทรลเลอร์ซึ่งอยู่ในรูปของการ์ดอินเทอร์เฟส ไมโครคอนโทรลเลอร์ในระบบนี้ใช้ตระกูล MCS-51 เบอร์ 89C51 ส่วนที่สามได้แก่ ส่วนแสดงผลและประยุกต์ใช้งาน ในระบบนี้ส่วนแสดงผลใช้คอมพิวเตอร์ในการแสดงผลและประยุกต์ใช้งาน นอกจากนี้หากเราไม่ต้องการใช้คอมพิวเตอร์ในส่วนนี้เราอาจใช้การแสดงผลที่เป็นส่วนเฉพาะ เช่น เซเว่นเซกเมนต์ (7-segment), แอลอีดี (LED) เป็นต้น แต่ส่วนที่เป็นการ์ดอินเทอร์เฟสจะต้องเปลี่ยนให้เป็นแบบการส่งแบบอนุกรมมาตรฐาน RS-232

1.3 หลักการทำงาน

จากแผนภาพของระบบการทำงานจะเริ่มจาก เมื่อมีวัตถุตัดผ่านลำแสงของชุดรับส่งอินฟราเรดนี้จะทำให้เกิดพัลส์ขึ้นที่โฟโต้ทรานซิสเตอร์ที่เป็นตัวรับ สัญญาณพัลส์ของชุดรับรังสีอินฟราเรดแต่ละชุดจะถูกนำมารวมกันบนการ์ดอินเทอร์เฟสและส่งสัญญาณเอาท์พุทที่ได้ให้กับ MCS-51 เพื่อทำการวัดผลต่างเวลาของการตัดแต่ละครั้งโดยใช้ไทม์เมอร์ ของไมโครคอนโทรลเลอร์เป็นฐานเวลา และเก็บผลลัพธ์ที่ได้ซึ่งอยู่ในรูปค่าผลต่างเวลาของแต่ละครั้งไว้ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปไมโครคอนโทรลเลอร์ (ไมโครคอนโทรลเลอร์ เบอร์ 89C51) ซึ่งมีขนาด 128 ไบต์ (bytes)

ระบบจับเวลาในที่นี้ถูกออกแบบเป็นการ์ดเชื่อมต่อกับคอมพิวเตอร์ผ่านทางสล็อตไอซา (ISA : Industrial Standard Architecture) เพื่อรับส่งข้อมูลขนาด 8 บิตกับคอมพิวเตอร์ ผลต่างเวลาที่เก็บไว้ในชิปไมโครคอนโทรลเลอร์จะถูกส่งไปยังหน่วยความจำของคอมพิวเตอร์ เมื่อคอมพิวเตอร์ต้องการข้อมูลในการนำไปแสดงผล ประมวลผลหรือประยุกต์ใช้งานอื่น ๆ เช่นคำนวณหาความเร็ว ความเร่ง เมื่อทราบระยะทางระหว่างแต่ละจุดเพื่อนำไปวิเคราะห์ หรือควบคุมต่อไป ในโครงงานนี้ได้แสดงการประยุกต์ใช้งานด้านต่าง ๆ ไว้เช่น ทางด้านสายการผลิตของผลิตภัณฑ์ และด้านการทดสอบความเร็ว เป็นต้น

บทที่ 2
ไมโครคอนโทรลเลอร์ MCS-51

2.1 คุณสมบัติของและโครงสร้างของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีสมาชิกในตระกูลหลายเบอร์ด้วยกัน แต่ละเบอร์จะมีคุณสมบัติพิเศษบางอย่างแตกต่างกัน เช่น มีหน่วยความจำภายในสำหรับเก็บโปรแกรม และข้อมูลภายในชิปเพิ่มขึ้น มีวงจรเปลี่ยนค่าสัญญาณอนาลอกเป็นดิจิตอลในตัว สามารถรับสัญญาณอินเทอร์รัปต์ได้หลายชนิด เข้าติดต่อกับหน่วยความจำได้โดยตรง (กระบวนการ DMA : Direct Memory Access) ได้ในตัว มีรีจิสเตอร์แต่ละเบอร์ที่ต่างกันออกไป

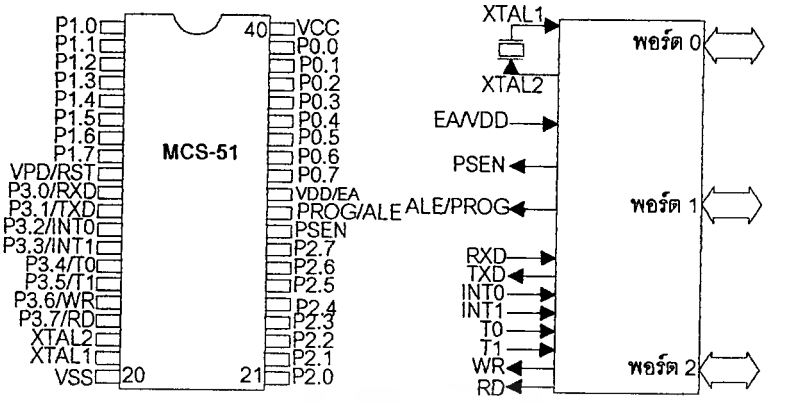
ไมโครคอนโทรลเลอร์เบอร์ตระกูล MCS-51 เบอร์ 89C51 จะมีหน่วยความจำสำหรับเก็บโปรแกรมในชิปไมโครคอนโทรลเลอร์ขนาด 4 กิโลไบต์ และหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปไมโครคอนโทรลเลอร์จำนวน 128 ไบต์ มีพอร์ตขนาด 8 บิต 4 พอร์ต มีรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์ หรือเคาเตอร์ขนาด 16 บิต รวม 2 ตัว รับสัญญาณอินเทอร์รัปต์จากภายนอกได้ 2 ชนิด สามารถรับ และส่งข้อมูลแบบอนุกรมผ่านทางพอร์ตสื่อสารข้อมูลแบบอนุกรม มีวงจรออสซิลเลเตอร์เพื่อสร้างสัญญาณนาฬิกาควบคุมการทำงานในตัวเอง

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 89C51 ใช้แรงดันไฟเพียง 5 โวลต์ในการทำงาน ส่วนกระแสไฟฟ้าที่ใช้จะแตกต่างกันไปตามชนิดของเทคโนโลยีที่ใช้ในการผลิต เบอร์ของไมโครคอนโทรลเลอร์ตระกูลที่มีตัวอักษร C อยู่ตรงกลางเบอร์ เหมือนที่ใช้ในระบบจะเป็นเบอร์ของชิปที่ผลิตโดยอาศัยเทคโนโลยีซีมอส (CMOS) ซึ่งใช้พลังงานในการทำงานน้อยกว่า และสามารถควบคุมการใช้พลังงานของตัวชิปได้จากโปรแกรมเพื่อการประหยัดพลังงานในระบบ

ความเร็วในการประมวลผลของเบอร์ 89C51 สามารถใช้ความถี่ได้ถึง 12 เมกะเฮิรตซ์ ทำให้ช่วงเวลาในการทำงานแต่ละคำสั่งน้อยมาก เมื่อใช้ความถี่ 12 เมกะเฮิรตซ์ คำสั่งที่ใช้เวลาน้อยที่สุดจะใช้เวลาเพียง 1 ไมโครวินาที ส่วนคำสั่งที่ใช้เวลามากที่สุดจะใช้เวลาเพียง 4 ไมโครวินาที

2.2 ตำแหน่งขาของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูปที่ 2.1



รูป 2.1 ตำแหน่งขาของชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51

หน้าที่การใช้งานแต่ละขาของชิปไมโครคอนโทรลเลอร์ในตระกูล MCS-51 มีดังนี้

- ขา Vss (ขา 20) สำหรับต่อลงกราวด์
- ขา Vcc (ขา 40) สำหรับต่อแหล่งจ่ายแรงดันกระแสตรงขนาด 5 โวลต์ (DC 5 Volt)
- ขาพอร์ต 0 (ขา 32-39) ใช้เป็นขาสำหรับพอร์ต 0 ขนาด 8 บิต (P0.0-P0.7) แบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานอินพุต เอาต์พุตพอร์ตทั่วไปได้
- ขาพอร์ต 1 (ขา 1-8) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 1 (P1.0-P1.7) สามารถใช้งานเป็นอินพุตหรือเอาต์พุตพอร์ตทั่วไปได้
- ขาพอร์ต 2 (ขา 21-28) ใช้เป็นขาสำหรับพอร์ต 2 (P2.0-P2.7) ขนาด 8 บิต แบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้
- ขาพอร์ต 3 (ขา 10-17) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 3 (P3.0-P3.7) สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ ในระบบนี้เราใช้ขาพอร์ท 3 นี้ในหน้าที่พิเศษอื่น คือขา P3.2 ใช้เป็นอินพุตเพื่อรับสัญญาณอินเทอร์รัปต์ชนิดที่ 0 การใช้งานพอร์ต 3 ในหน้าที่พิเศษดังกล่าวนี้ จะต้องโหลดค่า 1 ไปยังแต่ละบิตที่ต้องการใช้ก่อนทุกครั้ง
- ขา EA/Vpp (ขา 31) เป็นขาสำหรับใช้เลือกให้ไมโครคอนโทรลเลอร์ ทำงานจากโปรแกรมที่อยู่ภายในหรือภายนอกชิป โดยระบบนี้จะให้ขานี้มีสถานะเป็น 1 หมายถึงบังคับให้ไมโครคอนโทรลเลอร์ ใช้โปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมภายในชิป ด้วยการต่อขา EA กับไฟเลี้ยง
- ขา XTAL 1 (ขา 19) ใช้ต่อคริสตอลภายนอก โดยเป็นอินพุตเข้าสู่วงจรออสซิลเลเตอร์
- ขา XTAL 2 (ขา 18) ใช้ต่อคริสตอลภายนอก โดยเป็นเอาต์พุตออกจากวงจรออสซิลเลเตอร์

2.3 โครงสร้างภายในของ MCS-51

โครงสร้างภายในของชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีดังแสดงในรูปที่ 2.2 โครงสร้างหน่วยความจำภายในชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะแบ่งหน่วยความจำออกเป็นสองส่วนคือ

- หน่วยความจำสำหรับเก็บโปรแกรม (program memory)
- หน่วยความจำสำหรับเก็บข้อมูล (data memory)

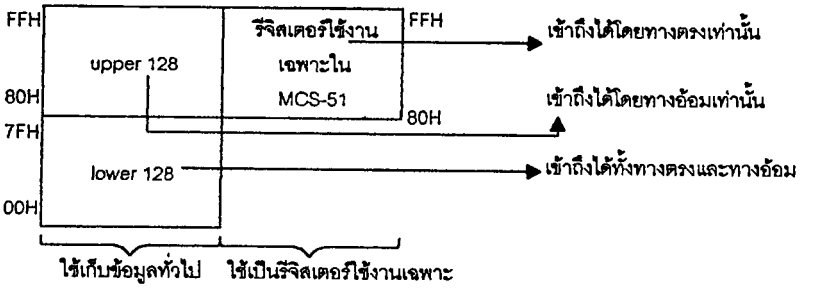
หน่วยความจำสำหรับเก็บโปรแกรมจะใช้เก็บโปรแกรมควบคุมการทำงานของชิปไมโครคอนโทรลเลอร์ ส่วนหน่วยความจำส่วนที่สองคือ หน่วยความจำสำหรับเก็บข้อมูล ซึ่งใช้สำหรับเก็บข้อมูลระหว่างการทำงาน

หน่วยความจำสำหรับเก็บโปรแกรม หน่วยความจำสำหรับเก็บโปรแกรมในไมโครคอนโทรลเลอร์ จะแบ่งออกเป็น 2 ส่วนคือ หน่วยความจำสำหรับเก็บโปรแกรมภายในชิป (internal program memory) และหน่วยความจำสำหรับเก็บโปรแรมภายนอกชิป (enternal program memory) ขนาดของหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปมีได้ตั้งแต่ 0,4,8,16 กิโลไบต์ ขึ้นอยู่กับเบอร์ของชิป

หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำสำหรับเก็บข้อมูลของไมโครคอนโทรลเลอร์จะแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำสำหรับเก็บข้อมูลภายในชิป และหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป หน่วยความจำสำหรับเก็บข้อมูลภายในชิป ก็ยังแบ่งออกเป็น 2 ส่วนย่อยดังนี้

- ส่วนที่ใช้เก็บข้อมูลทั่วไป (internal ram)
- ส่วนที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (special function register)

หน่วยความจำส่วนที่ใช้เก็บข้อมูลทั่วไปภายในชิปเป็นหน่วยความจำสำหรับเก็บข้อมูลที่มีอยู่ภายในไมโครคอนโทรลเลอร์ หน่วยความจำส่วนนี้มีไว้สำหรับเก็บข้อมูลในขณะทำงาน ส่วนหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะเป็นหน่วยความจำสำหรับเก็บข้อมูลภายในไมโครคอนโทรลเลอร์ ซึ่งถูกกำหนดให้เป็นใช้งานเฉพาะเพื่อควบคุมการทำงาน และบอก



รูป 2.2 แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิป

สถานะของซีพียู แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิปทั้งสองบริเวณสามารถแสดงได้ ดังในรูปที่ 2.2

ไมโครคอนโทรลเลอร์จะมีหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปอย่างน้อย 128 ไบต์ หน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิป บริเวณ 128 ไบต์แรกมีชื่อเรียกว่า lower และในบริเวณ 128 ไบต์หลังที่มีเพิ่มในบางเบอร์มีชื่อเรียกว่า upper 128 ดังแสดงในรูปที่ 2.2 หน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์หลัง (ตำแหน่ง 80H ขึ้นไป) จะมีตำแหน่งตรงกับหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (ตำแหน่ง 80H ขึ้นไปเช่นกัน) โดยมีวิธีการเข้าถึงข้อมูลในหน่วยความจำทั้งสองส่วนไม่เหมือนกัน

รีจิสเตอร์ใช้งานเฉพาะ เนื่องจากไมโครคอนโทรลเลอร์ ถูกออกแบบไว้สำหรับควบคุมระบบโดยเฉพาะ จึงทำให้มีความสามารถเฉพาะตัวหลายอย่าง ซึ่งจำเป็นต้องอาศัยวงจรภายในชิปที่มีเพิ่มขึ้นจากไมโครโปรเซสเซอร์ทั่วไป การควบคุมการทำงานของวงจรภายในไมโครคอนโทรลเลอร์จะกระทำผ่านรีจิสเตอร์ที่ถูกกำหนดหน้าที่ไว้แล้ว

รีจิสเตอร์สำหรับใช้งานทั่วไป ในไมโครคอนโทรลเลอร์ที่ใช้จะมีรีจิสเตอร์ใช้งานทั่วไปที่ผู้เขียนโปรแกรมสามารถนำมาใช้งานได้คือ รีจิสเตอร์ A , B (อยู่ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ แต่นับเป็นรีจิสเตอร์ใช้งานทั่วไป เพราะไม่ถูกกำหนดหน้าที่ใช้งานโดยตรง) และรีจิสเตอร์ใช้งานทั่วไป R0-R7 ซึ่งอยู่ในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์แรก

รีจิสเตอร์ใช้งานทั่วไปทั้ง R0-R7 จะมีอยู่ในกลุ่มรีจิสเตอร์ใช้งานทั่วไปทั้ง 4 กลุ่ม ซึ่งจะถูกเลือกใช้งานเพียงกลุ่มเดียวในขณะนั้น ค่าที่เปลี่ยนแปลงไปในรีจิสเตอร์ใช้งานทั่วไปที่ถูกเลือกใช้งานในขณะนั้นจะไม่มีผลต่อรีจิสเตอร์ใช้งานทั่วไปที่มีชื่อเดียวกัน แต่อยู่คนละกลุ่มเลย โครงสร้างเช่นนี้ทำให้มีความสะดวกในการเขียนโปรแกรมเป็นอันมาก โดยเฉพาะกับการเขียนโปรแกรมที่มีการเรียกใช้โปรแกรมย่อย (subroutine)

โครงสร้างพอร์ตของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีพอร์ตขนาด 8 บิต จำนวน 4 พอร์ต (P0 , P1 , P2 , P3) โดยสามารถกำหนดให้ทำงานแบบพอร์ตขนานขนาด 8 บิต 4 พอร์ต หรือจะใช้เป็นพอร์ตขนาด 1 บิต ได้ถึง 32 พอร์ต ทั้งผู้ใช้ยังสามารถกำหนดให้แต่ละพอร์ตใช้งานเป็นอินพุตพอร์ต หรือเอาต์พุตพอร์ตอย่างใดอย่างหนึ่งได้อย่างอิสระ

ไทม์เมอร์/เคาน์เตอร์ ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีรีจิสเตอร์ใช้งานเฉพาะที่สามารถนับจำนวนสัญญาณนาฬิกา หรือแมชชีนไซเกิลของวงจรออกสซิลเลเตอร์ภายใน (ทำงานเห็นไทม์เมอร์) หรือนับจำนวนครั้งของการเปลี่ยนสถานะของสัญญาณภายนอก รีจิสเตอร์ที่ใช้เป็นไทม์เมอร์หรือเคาน์เตอร์มีขนาด 16 บิต จำนวน 2 ตัว คือ รีจิสเตอร์ไทม์เมอร์ 0 และรีจิสเตอร์ไทม์เมอร์ 1 ตามลำดับ การควบคุมการทำงานของไทม์เมอร์ หรือ เคาน์เตอร์ สามารถควบคุมได้จากวงจรภายนอก (ควบคุมด้วยสัญญาณที่ขา INT0 , INT1) หรือควบคุมจากคำสั่งในโปรแกรม ดังนั้นรีจิสเตอร์ที่ใช้เป็นไทม์เมอร์ในไมโครคอนโทรลเลอร์จะสามารถวัดช่วงห่างของเวลา วัดความกว้างของพัลส์ หรือนับจำนวนครั้งของเหตุการณ์ที่เกิดขึ้นภายนอกที่เปลี่ยนให้อยู่ในรูปของสัญญาณไฟฟ้าแล้ว รวมทั้งใช้กำเนิดสัญญาณอินเทอร์รัปต์ที่มีคาบเวลาที่แน่นอนได้

โครงสร้างการอินเทอร์รัปต์ ไมโครคอนโทรลเลอร์ตระกูล MCS-51 สามารถรับสัญญาณอินเทอร์รัปต์ได้ถึง 5 ชนิด โดยจะเป็นสัญญาณอินเทอร์รัปต์ที่เกิดจากภายนอก 2 ชนิด และที่เกิดจากภายในชิปอีก 3 ชนิด เมื่อมีสัญญาณอินเทอร์รัปต์เกิดขึ้นไมโครคอนโทรลเลอร์ จะละการทำงานโปรแกรมที่กำลังทำอยู่ และข้ามไปทำงานโปรแกรมบริการอินเทอร์รัปต์ (interrupt service routine) ที่อยู่ในหน่วยความจำตำแหน่งต่าง ๆ ขึ้นอยู่กับชนิดของสัญญาณอินเทอร์รัปต์

เราสามารถเลือกให้ซีพียูในไมโครคอนโทรลเลอร์ ถูกอินเทอร์รัปต์โดยสัญญาณอินเทอร์รัปต์ที่เกิดขึ้นได้ โดยการกำหนดค่าในรีจิสเตอร์ใช้งานเฉพาะ IE นอกจากนี้ยังสามารถควบคุมความสำคัญในการตอบสนองต่อสัญญาณอินเทอร์รัปต์ของไมโครคอนโทรลเลอร์ได้ด้วยรีจิสเตอร์ใช้งานเฉพาะ IP

2.4 ไทม์เมอร์/เคาน์เตอร์

ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 มีรีจิสเตอร์พิเศษที่สามารถเลือกใช้งานเป็นไทม์เมอร์ หรือเคาน์เตอร์อย่างใดอย่างหนึ่ง (นับจำนวนแมชชีนไซเกิล หรือนับจำนวนพัลส์ที่เกิดขึ้นภายนอกชิป) รีจิสเตอร์ประเภทนี้มีอยู่ด้วยกัน 2 ตัว แต่ละตัวมีขนาด 16 บิต โดยมีชื่อเรียกว่าไทม์เมอร์ 0 และ ไทม์เมอร์ 1 ตามลำดับ

ไทม์เมอร์ : ค่าในรีจิสเตอร์ที่ใช้เป็นไทม์เมอร์ที่ถูกเลือกใช้งานจะถูกเพิ่มค่าทุก ๆ แมชชีนไซเกิล ดังนั้นจึงสามารถคิดว่าในขณะทำงานเป็นไทม์เมอร์ หมายถึงใช้รีจิสเตอร์เป็นตัวนับจำนวนแมชชีนไซเกิลได้ และเนื่องจากใน 1 แมชชีนไซเกิลใด ๆ ของไมโครคอนโทรลเลอร์ประกอบไปด้วย 12 คาบสัญญาณออสซิลเลเตอร์ (oscillator period) ดังนั้นอัตราเร็วในการนับ (count rate) จึงมีค่าเป็น 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้

2.5 โครงสร้างการอินเทอร์รัปต์ MCS-51

MCS-51 สามารถรับสัญญาณอินเทอร์รัปต์ที่เกิดขึ้นได้อย่างน้อย 5 ชนิดด้วยกัน (MCS-51 บางเบอร์ในตระกูลนี้สามารถรับสัญญาณอินเทอร์รัปต์ได้มากกว่า 5 ชนิด แหล่งกำเนิดสัญญาณอินเทอร์รัปต์ทั้ง 5 ชนิดที่ MCS-51 สามารถรับได้

อินเทอร์รัปต์ที่เกิดจากภายนอก (External Interrupts) เป็นอินเทอร์รัปต์ที่เกิดขึ้นจากภายนอกชิปไมโครคอนโทรลเลอร์ มี 2 ชนิดด้วยกันคือ

- อินเทอร์รัปต์ภายนอกชนิด 0 รับได้จากขา INT0
- อินเทอร์รัปต์ภายนอกชนิด 1 รับได้จากขา INT1

ผู้ใช้สามารถกำหนดการตรวจสอบสัญญาณอินเทอร์รัปต์ทั้งสองชนิดที่เกิดขึ้นที่ขา INT0 , INT1 2 แบบ ด้วยกันคือ

- ตรวจสอบจากระดับสัญญาณ (level – activated)
- ตรวจสอบจากการเปลี่ยนสถานะ (transition activated)

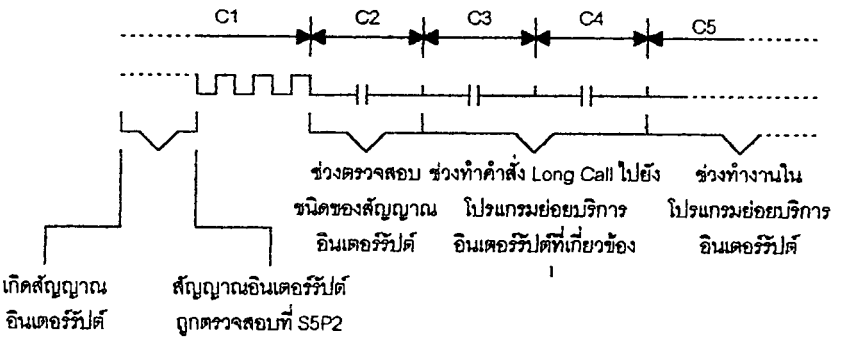
การตรวจสอบสถานะของสัญญาณอินเทอร์รัปต์ภายนอกที่ขาทั้งสอง สามารถเลือกได้เพียงอย่างใดอย่างหนึ่งขึ้นอยู่กับการกำหนดค่าบิต IT0 , IT1 (Interrupt Type control bit) ในรีจิสเตอร์ใช้งานเฉพาะ TCON

2.5.1 การจัดการสัญญาณอินเทอร์รัปต์ของ MCS-51

สัญญาณอินเทอร์รัปต์แต่ละชนิดจะถูกตรวจสอบที่ทุก ๆ สเตท 5 เฟส 2 ของทุก ๆ แมชชีนไซเคิล ค่าที่ตรวจสอบได้จะถูกรับเข้ามาในแมชชีนไซเกิลถัดไป

การตอบสนองต่อสัญญาณอินเทอร์รัปต์จะมีลักษณะในรูปที่ 2.3 จะเห็นว่าประกอบไปด้วยไซเคิลการทำงานต่าง ๆ กัน ได้แก่

-



รูป 2.3 การตอบสนองต่อสัญญาณอินเทอร์รัปต์

- ไซเคิลตรวจสอบสัญญาณอินเทอร์รัปต์ (ตรวจสถานะของบิตที่ทำหน้าที่บอกสถานะสัญญานอินเทอร์รัปต์แต่ละประเภท) จะตรวจสอบทุก ๆ สเตท 5 เฟส 2 ของแต่ละแมชชีนไซเคิล
- ไซเคิลตรวจหาชนิดของสัญญาณอินเทอร์รัปต์ (pooling cycle) ตรวจสอบว่าสัญญาณอินเทอร์รัปต์ที่เกิดขึ้นในไซเกิลตรวจจับสัญญาณอินเทอร์รัปต์เป็นสัญญาณอินเทอร์รัปต์ชนิดใด
- ไซเคิลการสร้างคำสั่ง long call ไปยังโปรแกรมบริการอินเทอร์รัปต์ที่เหมาะสม (ใช้เวลา 2 แมชชีนไซเคิล)
- ไซเคิลการทำคำสั่งที่อยู่ในโปรแกรมบริการอินเทอร์รัปต์

บทที่ 3

การอินเทอร์เฟส (INTERFACE)

3.1 สัญญาณต่างๆบนสล็อตของ IBM/PC

ภายใน IBM/PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรอินเทอร์เฟสเข้าไปในภายหลังได้ โดยผ่านทางสล็อตที่อยู่บนเมนบอร์ด (Main Board) สำหรับสล็อตบนเมนบอร์ดนี้จะมีจำนวน 5 สล็อต (สำหรับใน IBM PC/XT จะมี 8 สล็อต) ซึ่งแต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขา แบ่งออกเป็น 2 ข้างๆละ 31 ขา ส่วนการเรียกตำแหน่งของขาของสล็อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้ายหรือขวา) ของสล็อต โดยขาที่อยู่ทางด้านซ้ายของสล็อตจะเรียกโดยใช้อักษร "B"นำหน้าเลขตำแหน่งของขา เช่น ขา B16 ก็คือขาทางด้านซ้ายของสล็อต ขาที่ 16 (นับจากทางด้านท้ายของเครื่อง)

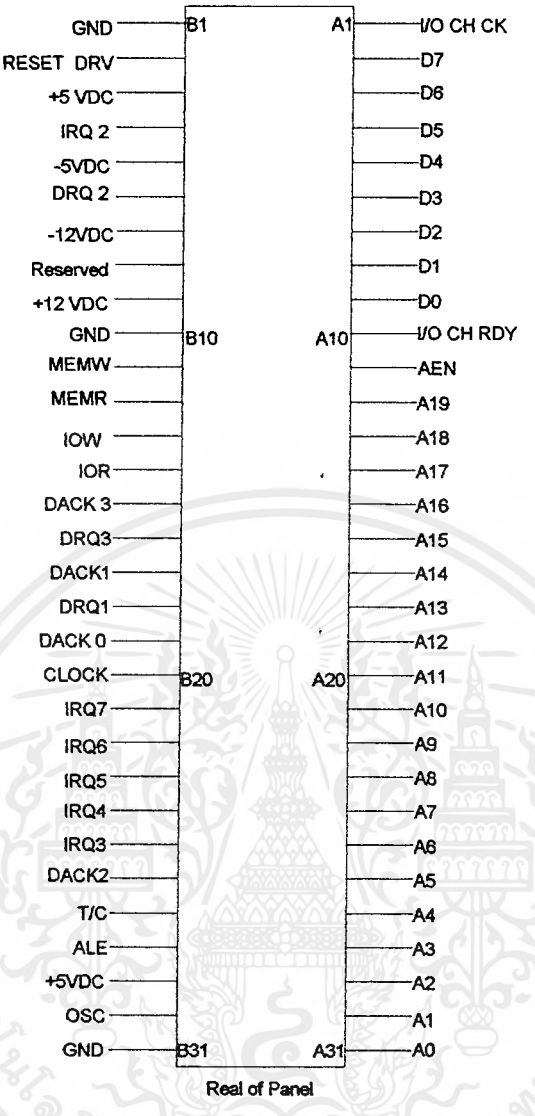
แต่ละขาของสล็อตเหล่านี้จะเชื่อมต่อกับสัญญาณต่างๆบนเมนบอร์ด ทำให้การสร้างวงจรอินเทอร์เฟสกับ IBM/PC สามารถทำได้โดยสะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล็อตเหล่านี้จะประกอบไปด้วย เส้นสัญญาณของบัสแอดเดรส (Address Bus), บัสข้อมูล (Data Bus), บัสควบคุมสำหรับการเขียน/อ่านข้อมูลจากหน่วยความจำ หรือพอร์ต I/O, เส้นสัญญาณสำหรับการขออินเทอร์รัปต์ของวงจรอินเทอร์เฟส, เส้นสัญญาณสำหรับการขอ DMA, สัญญาณฐานเวลา (Timing Signal) ต่างๆที่ใช้ในระบบ, เส้นสัญญาณแสดงการรีเฟรส (refresh) หน่วยความจำ และสัญญาณสำหรับการตรวจสอบความผิดพลาด (I/O CHCK)

นอกจากเส้นสัญญาณเหล่านี้แล้ว สล็อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่างๆที่ใช้ในระบบอีกด้วย คือ +5Vdc, -5Vdc, +12Vdc และ –12Vdc

3.1.1 รายละเอียดเกี่ยวกับสัญญาณต่างๆที่ใช้ในระบบตรวจจับเวลา

- A0-A19 (Address Bus ; ขา A31-A12) :

ขาสัญญาณทั้ง 20 ขานี้เป็นเอาท์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำหรืออุปกรณ์ I/O ที่ 8088 ต้องการติดต่อด้วย โดยที่สัญญาณ A0 จะมีนัยสำคัญต่ำสุด (Least Significant Bit) และ A19 จะมีนัยสำคัญสูงสุด (Most Significant Bit) สำหรับค่าแอดเดรสบนบัสแอดเดรส A0-A19 นี้จะถูกกำหนดโดย 8088 ในระหว่างขบวนการอ่าน/เขียนข้อมูลลงในหน่วยความจำหรืออุปกรณ์



รูป 3.1 ตำแหน่งขาสัญญาณต่าง ๆ บนสล็อต

I/O แต่ในช่วงของขบวนการ DMA นั้น DMA-Controller จะเป็นผู้กำหนดค่าแอดเดรสบนบัสแอดเดรสเอง (ในระหว่างนี้ 8088 จะถูกตัดออกจากระบบ)

จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้จะมีอยู่ 20 เส้น ซึ่งสามารถที่จะอ้างแอดเดรสของหน่วยความจำได้ถึง 1 เมกะไบต์ (Mbyte) แต่อย่างไรก็ตามมีแอดเดรสบางแอดเดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้ว คือแอดเดรสของหน่วยความจำเก็บข้อมูล บนเมนบอร์ดที่ถูกใช้โดยระบบ จำนวน 64 กิโลไบต์ (Kbyte) (สำหรับ IBM PC/XT จะเป็นจำนวน 256 Kbyte) และแอดเดรสสำหรับหน่วยความจำเก็บโปรแกรมอีก 48 กิโลไบต์ ซึ่งถูกจัดในช่วงของแอดเดรสบนสุดใน 1 เมกกะไบต์ คือ 0FC00H จนถึง 0FFFFFH (สำหรับ IBM PC/XT จะเป็น 64 กิโลไบต์)

สำหรับการอ้างแอดเดรสของพอร์ต I/O นั้น จะใช้เส้นแอดเดรสเพียง 16 เส้น คือ A0-A15 ซึ่งจะทำให้อ้างแอดเดรสของพอร์ตได้ 64K พอร์ต โดยผ่านทางชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสที่เหลือคือ A16-A19 นั้นจะไม่ถูกใช้งาน อย่างไรก็ตามภายใน IBM/PC จะใช้เส้นแอดเดรสในการอ้างแอดเดรสของพอร์ตเพียง 10 เส้นคือจาก A0-A9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ในช่วง 0200H จนถึง 03FFH เท่านั้น

- D0-D7 (Data Bus ; ขา A9-A2) :

ขาสัญญาณนี้เป็นสัญญาณเอาท์พุทที่ 8288 Bus Controller สร้างขึ้นเพื่อใช้สำหรับแสดงการเริ่มต้นของบัสไซเคิล และแสดงให้อุปกรณ์ภายนอกทราบว่าแอดเดรสที่ 8088 ต้องการจะติดต่อด้วยนั้นถูกส่งออกมาบนบัสแอดเดรสแล้ว โดยที่สัญญาณ ALE นี้จะเปลี่ยนจากลอจิก "1" เป็น "0" เมื่อค่าแอดเดรสที่ถูกต้องถูกส่งออกมาบนบัสข้อมูลเรียบร้อยแล้ว ดังนั้นขอบขาลงของสัญญาณ ALE นี้จะถูกใช้ในการแลทช์ค่าแอดเดรส/ข้อมูล (Address/Data Bus ; ADo-AD7) ของ 8088 ทำให้สามารถแยกค่าแอดเดรส (Ao-A19) และข้อมูล (Ao-A7) ออกจากกันได้ อย่างไรก็ตามสัญญาณ ALE จะแอคทีฟเฉพาะในบัสไซเคิลที่สร้างขึ้นโดย 8088 เท่านั้นโดยจะไม่แอคทีฟในระหว่างขบวนการ DMA

- I/O CHRDY (I/O Channel Ready ; ขา A 10) :

ขาสัญญาณนี้เป็นอินพุทที่ใช้เพิ่มช่วงเวลาในบัสไซเคิลในกรณีที่อุปกรณ์ I/O หรือหน่วยความจำที่เกี่ยวข้องกับขบวนการในบัสไซเคิลที่เกิดขึ้นนั้น ไม่สามารถทำงานทันในช่วงเวลาปกติของบัสไซเคิลนั้นๆได้ (ช่วงเวลาของบัสไซเคิลที่เกี่ยวกับหน่วยความจำใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 4 ลูก หรือ 840 นาโนวินาที ในขณะที่บัสไซเคิลที่เกี่ยวกับ I/O CHRDY ในช่วงเวลาเท่ากับช่วงเวลาของคล็อก 5 ลูกหรือ 1.05 ไมโครวินาที (usec.)

เมื่ออุปกรณ์ I/O หรือหน่วยความจำต้องการที่จะเพิ่มช่วงเวลาในบัสไซเคิลให้นานขึ้นอีกนั้นจะสามารถทำได้โดยการป้อนลอจิก "0" ให้กับขา I/O CHRDY ในช่วงเวลาที่ I/O หรือหน่วยความจำที่ถูกกำหนดนั้น ได้รับสัญญาณจากการดีโค้ดแอดเดรสและสัญญาณ MEMR, MEMW, IOR หรือ IOW แอคทีฟ สำหรับรายละเอียดเกี่ยวกับการเพิ่มช่วงเวลาในบัสไซเคิลนั้นจะกล่าวถึงอีกครั้งในบท "การสร้าง เวทสเตท (เวทสเตท)"

- IOR (I/O Read ; ขา B14) :

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ที่สร้างขึ้นโดย 8288 Bus Controller เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นนี้ เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ต I/O เพื่อให้พอร์ตI/O ที่มีแอดเดรสบนบัสแอดเดรสนั้นส่งข้อมูล โดยข้อมูลจะต้องถูกส่งออกมาบนบัสข้อมูลก่อนขอบขาขึ้นของสัญญาณ IOR ประมาณ 30 นาโนวินาที เพื่อให้มั่นใจได้ว่า 8088 สามารถรับข้อมูลได้ถูกต้อง สำหรับในขบวนการ DMA 8237A-5DMA Controller จะทำการสร้างสัญญาณ IOR เอง โดยที่ค่าแอด

เดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำ (แทนที่จะเป็นแอดเดรสของพอร์ต I/O) ที่พอร์ต I/O ที่ขอ DMA ต้องการจะนำข้อมูลไปเก็บ การที่พอร์ตใดจะส่งข้อมูลออกมาบนบัสข้อมูลนั้น จะอาศัยสัญญาณ DACK จากDMA Controller เป็นตัวกำหนด เช่นกรณีที่สัญญาณ DACK1 แอคทีฟก็จะแสดงว่าพอร์ต I/O ที่จะต้องส่งข้อมูลออกมาบนบัสข้อมูลก็คือพอร์ต I/O ที่ขอ DMA ผ่านทางแชนแนลที่ 1 (DRQ1) เป็นต้น

- IOW (I/O Write ; ขา B13) :

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่งถูกสร้างขึ้นโดย 8288 Bus Controller เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ต I/O เพื่อให้พอร์ต I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น รับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ อย่างไรก็ตาม เนื่องจากในช่วงเวลาที่สัญญาณ IOW นี้แอคทีฟ (ลอจิก"0") นั้นข้อมูลบนบัสอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ IOW แทนขอบขาลงในการทำให้พอร์ต I/O ที่เกี่ยวข้องรับข้อมูลไปเก็บไว้ เพื่อให้ข้อมูลบนบัสข้อมูลสมบูรณ์เสียก่อน สำหรับในขบวนการ DMA นั้น DMA-Controller จะทำการสร้างสัญญาณ IOW เอง โดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ต I/O ที่ขอ DMA ต้องการจะอ่านข้อมูล

- AEN (Address Enable ; ขา A11)

สัญญาณนี้เป็นเอาท์พุทที่ใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นในช่วงเวลาที่สัญญาณ AEN แอคทีฟ (ลอจิก "1") นั้นเป็นบัสไซเคิลของขบวนการ DMA

สำหรับบนเมนบอร์ดของ IBM/PC นั้น จะใช้สัญญาณนี้ในการดิสเอเบิล (Disable) 8288 Bus Controller และจะใช้ดิสเอเบิลพอร์ต I/O ต่างๆที่ไม่เกี่ยวข้องกับขบวนการ DMA ที่เกิดขึ้นนี้ ที่จำเป็นต้องทำเช่นนี้ก็เพราะในระหว่างขบวนการ DMA นั้น 8237A-5 จะส่งแอดเดรสของหน่วยความจำมาบนบัสแอดเดรส และจะทำให้สัญญาณ IOR หรือ IOW แอคทีฟด้วย ดังนั้นถ้าไม่ทำการดิสเอเบิลพอร์ต I/O ที่ไม่เกี่ยวข้องไว้ ก็อาจจะทำให้พอร์ต I/O ที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรส (ซึ่งเป็นแอดเดรสของหน่วยความจำ) นั้น ทำการอ่านหรือส่งข้อมูลออกมาบนบัสข้อมูลทำให้เกิดความผิดพลาดขึ้นได้

3.1.2 บัสของแหล่งจ่ายไฟของระบบ

- +5Vdc (ขา B3 และ B29) :

ขาทั้งสองนี้ต่อกับแหล่งจ่ายไฟตรง +12 โวลท์ ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$5% คืออยู่ในช่วง +4.75 ถึง +5.25 โวลท์

- +12Vdc (ขา B9) :

ขานี้จะต่อกับแหล่งจ่ายไฟตรง +12 โวลท์ ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) ±5% คืออยู่ในช่วง +11.4 ถึง +12.6 โวลท์

- -5Vdc (ขา B5) :

ขานี้จะต่อกับแหล่งจ่ายไฟตรง -5 โวล์ท ของระบบ โดยจะมีค่าความเที่ยงตรง(Regulated) ±10% คืออยู่ในช่วง -5.5 ถึง –4.5 โวลท์

- -12Vdc (ขา B7) :

ขานี้จะต่อกับแหล่งจ่ายไฟตรง -12 โวลท์ ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) ±10% คืออยู่ในช่วง –13.2 ถึง –10.8 โวลท์

- GND (ขา B1, B10 และ B31) :

ขาทั้งสามนี้ต่อเข้ากับกราวด์ (Ground) ของระบบ

3.1.3 การจัดสัญญาณบนสล็อตของ IBM PC/XT

สำหรับใน IBM PC/XT นั้นจะมีสล็อตสำหรับเชื่อมต่อกับวงจรภายนอกได้มากขึ้นคือใน IBM PC/XT จะทำการเพิ่มจำนานสล็อตบนเมนบอร์ดขึ้นมาเป็น 8 สล็อต จากเดิมที่มีอยู่เพียง 5 สล็อตบน IBM PC โดยการจัดสัญญาณต่างๆในทั้ง 8 สล็อตจะยังคงเหมือนกับใน IBM PC เพียงแต่สัญญาณต่างๆที่จะถูกส่งออกมาที่ขาของสล็อตที่ 8นั้น จะถูกต่อจากวงจรขับกระแส (Buffer) ก่อน และในสล็อตที่ 8 นี้ขา B8 จะถูกใช้งานด้วย โดยจะถูกใช้เป็นขา CARD SLCTD (หรือ Card Selected) ซึ่งขาสัญญาณนี้จะเป็นสัญญาณอินพุทจากวงจรภายนอกที่เสียบอยู่บนสล็อตที่ 8 เพื่อให้วงจรบนเมนบอร์ดทราบว่าคาร์ดที่อยู่บนสล็อตนี้ถูกเลือกใช้งานอยู่ ซึ่งจะทำให้ Driver บนเมนบอร์ดทำการอ่านหรือส่งข้อมูลไปยังสล็อตที่ 8

3.2 การจัดแอดเดรสสำหรับพอร์ต I/O ใน IBM/PC

ในหัวข้อนี้จะกล่าวถึงวิธีการอ้างและใช้งานแอดเดรสต่างๆของพอร์ต I/O ที่ใช้งานอยู่ใน IBM/PC

3.2.1 การอ้างแอดเดรสของพอร์ต I/O

ในการควบคุมและตรวจสอบสภาวะการทำงาน รวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นชิปชัพพอร์ตหรือการ์ดต่างๆ ที่ใช้ในระบบของ IBM/PC นั้น จะกระทำโดยผ่านทางพอร์ต I/O ของระบบ ดังนั้นในการที่จะใช้งานหรือควบคุมการทำงานของอุปกรณ์เหล่านี้ จึงจำเป็นที่จะต้องศึกษาถึงวิธีที่จะควบคุมพอร์ต I/O ต่างๆของระบบด้วย และเนื่องจากการควบคุมหรือติดต่อกับพอร์ตเหล่านี้ ต้องกระทำโดยการอ้างถึงแอดเดรสของพอร์ต I/O เหล่านั้นโดยตรง เราจึงจำเป็นต้องศึกษาถึงหลักการอ้างแอดเดรสของ 8088 ใน IBM/PC ด้วย

สำหรับแอดเดรสของพอร์ต I/O ต่างๆนั้น จะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088 ซึ่งแอดเดรสเหล่านี้เป็นแอดเดรสที่จัดไว้สำหรับพอร์ต I/O โดยเฉพาะ คือแยกจากแอดเดรสของหน่วยความจำโดยเด็ดขาด ส่วนการส่งข้อมูลให้กับพอร์ตเหล่านี้จะทำได้โดยการใช้คำสั่ง OUT ของ 8088 ส่งข้อมูลไปยังแอดเดรสของพอร์ตที่ต้องการ และสำหรับการตรวจสอบหรือการอ่านข้อมูลจากพอร์ต ก็จะทำได้โดยการใช้คำสั่ง IN ของ 8088 อ่านข้อมูลจากแอดเดรสของพอร์ตที่ต้องการเช่นกัน

ภายในไมโครโปรเซสเซอร์เบอร์ 8088 นี้ จะมีแอดเดรสสำหรับใช้กับพอร์ต I/O อยู่ทั้งสิ้น 65,536 หรือ 64K แอดเดรส (ในขณะที่มีแอดเดรสสำหรับหน่วยความจำอยู่ 1 เมกกะไบต์) ซึ่งทำให้การอ้างแอดเดรสของพอร์ต I/O ที่ทำงานร่วมกับ 8088 นั้น ต้องใช้จำนวนเส้นแอดเดรสในบัสแอดเดรสทั้งสิ้น 16 เส้น คือ A0-A15 แต่สำหรับใน IBM/PC นี้ถูกออกแบบมาให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นล่าง คือ A0-A9 เท่านั้น ดังนั้นในการอ้างถึงแอดเดรสของพอร์ตของอุปกรณ์หรือชิพชัพพอร์ตใดๆ ที่ใช้ร่วมกับ IBM/PC จึงใช้จำนวนเส้นแอดเดรสเพียง 10 เส้นด้วย โดยเส้นแอดเดรสที่เหลือคือ A10-A15 นั้น จะไม่ถูกนำไปใช้งาน อย่างไรก็ตามถึงแม้ว่าเส้นแอดเดรส A10-A15 นี้จะไม่ถูกนำไปใช้งาน แต่ค่าแอดเดรสบนเส้นแอดเดรสเหล่านี้ยังคงเปลี่ยนแปลงตามค่าแอดเดรสของพอร์ตที่กำหนดไว้ในคำสั่ง OUT หรือ IN อยู่ด้วย เพียงแต่ไม่ได้ถูกนำมาดีโค้ดร่วมกับแอดเดรส A0-A9 เท่านั้น ตัวอย่างเช่น ในการใช้คำสั่ง OUT ส่งข้อมูลไปยังพอร์ตที่ตรงกับแอดเดรส 0010H นั้น จะให้ผลเหมือนกับการส่งข้อมูลไปยังพอร์ตที่ตรงกับแอดเดรส 0410H, 0810, 0C10H ทั้งนี้เนื่องจากแอดเดรส 6 บิทบนไม่ได้ถูกใช้งานจึงทำให้การเปลี่ยนแปลงค่าแอดเดรสบนเส้นแอดเดรส A10-A15 นั้นไม่ทำให้เกิดความแตกต่างใดๆขึ้น

เนื่องจากใน IBM/PC ได้ใช้งานเส้นแอดเดรสเพียง 10 เส้น (คือ A0-A9) ดังนั้นจึงสามารถที่จะอ้างแอดเดรสของพอร์ตได้สูงสุดเพียง 1024 พอร์ต (จากจำนวน 64 พอร์ต)เท่านั้น นอกจากนี้ในกรณีที่เป็นการอ่านข้อมูลจากพอร์ตของ IBM/PC ข้อมูลในบิท A9 จะถูกจัดให้มีหน้าที่ในการแบ่งพอร์ตทั้ง 1024 พอร์ต ออกเป็น 2 ส่วน (ส่วนละ 512 พอร์ต) อีกด้วย กล่าวคือถ้าข้อมูลในบิท A9 จะถูกจัดให้มีหน้าที่ในการแบ่งพอร์ตออกเป็น 2 ส่วน (ส่วนละ 512 พอร์ต) อีกด้วย กล่าวคือถ้าข้อมูลใน

บิท A9 เป็น "0" แล้ว เราจะทำการอ่านข้อมูลได้เฉพาะจากพอร์ตของอุปกรณ์หรือชิปซัพพอร์ตต่างๆที่อยู่บนเมนบอร์ด (Main Board) ของ IBM/PC เช่น 8253-5, 8237-5 หรือ 8259A เท่านั้น แต่ถ้าข้อมูลในบิท A9 นี้เป็น "1" ก็จะทำการอ่านข้อมูลได้เฉพาะจากพอร์ตที่อยู่บนการ์ดต่างๆเท่านั้น

จากที่ได้กล่าวมานั้นสรุปได้ว่าพอร์ตบน IBM/PC ทั้ง 1024 พอร์ตถูกแบ่งออกเป็น 2 กลุ่ม โดยที่กลุ่มแรกเป็นกลุ่มของพอร์ตที่อยู่บนเมนบอร์ด และกลุ่มที่สองเป็นกลุ่มที่จัดเตรียมไว้สำหรับพอร์ตที่อยู่บนการ์ดต่างๆ

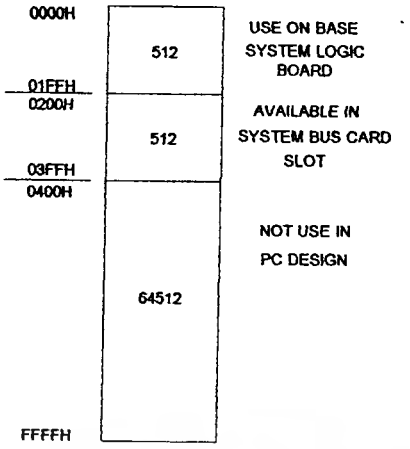
สำหรับในกรณีของการส่งข้อมูลให้กับพอร์ตทั้ง 1024 พอร์ต เราสามารถที่จะเลือกส่งไปยังพอร์ตใดๆ ใน IBM/PC ได้ ดังนั้นการเลือกแอดเดรสสำหรับพอร์ตที่อยู่บนการ์ดจึงสามารถทำได้โดยสะดวก แต่อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงก็คือ ถ้าแอดเดรสที่เราเลือกให้กับพอร์ตนี้ตรงกับค่าแอดเดรสเดิมที่มีอยู่บนเมนบอร์ดแล้ว เมื่อเราทำการส่งข้อมูลให้กับพอร์ตที่อยู่ในตำแหน่งแอดเดรสนี้จะเท่ากับเป็นการส่งข้อมูลให้กับทั้งพอร์ตที่อยู่บนเมนบอร์ดและพอร์ตที่อยู่บนการ์ดด้วย ซึ่งในกรณีเช่นนี้อาจก่อให้เกิดความผิดพลาดขึ้นได้เช่นกัน ดังนั้นในการกำหนดค่าแอดเดรสให้กับพอร์ตที่ถูกสร้างขึ้นบนการ์ดต่างๆ จึงควรใช้ค่าแอดเดรสที่แอดเดรสบิท A9 มีค่าเป็น "1" คือแอดเดรส 0FE00H จนถึง 0FFFFH เท่านั้น (แอดเดรสบิท A10-A15 ไม่ถูกใช้ในการดีโค้ด แต่เพื่อความสะดวกจึงกำหนดให้มีค่าเป็น "1" ในฐานสองทั้งหมด แต่ในการใช้งานจริงอาจเปลี่ยนให้แอดเดรส A10-A15 แต่ละบิทมีค่าเป็น "1" หรือ "0" ก็ได้)

3.2.2 การใช้งานแอดเดรสสำหรับพอร์ต I/O ใน IBM/PC

จากที่ได้กล่าวไว้ในหัวข้อที่ผ่านมานั้น พอร์ต I/O ทั้ง 1024 พอร์ตใน IBM/PC จะถูกแบ่งออกเป็น 2 กลุ่มๆละ 512 พอร์ต สำหรับในหัวข้อนี้จะกล่าวถึงการใช้พอร์ตต่างๆเหล่านี้โดยจะแบ่งออกเป็น 2 กลุ่มดังนี้

1. ในกลุ่มแรกนี้เป็นกลุ่มของพอร์ต I/O ที่อยู่บนเมนบอร์ดของ IBM/PC ซึ่งจะมีแอดเดรสอยู่ในตำแหน่ง 0000H จนถึง 01FFH (ขอให้ระลึกอยู่เสมอว่า A10-A15 นั้นไม่ถูกใช้งาน) หรือแอดเดรสที่มีบิท A9 เป็น "0" นั่นเอง

สำหรับแอดเดรสของพอร์ต I/O ในกลุ่มนี้จะถูกใช้ในการอ้างแอดเดรสของชิพซัพพอร์ตและอุปกรณ์ที่เป็น I/O ต่างๆ บนเมนบอร์ดของ IBM/PC เช่น แอดเดรส 0000H จนถึง 000FH จะถูกใช้เป็นแอดเดรสสำหรับ 8237-5 DMA Controller เป็นต้น ในรูปที่ 3.1 จะแสดงถึงการใช้งานแอดเดรสต่างๆ ตั้งแต่ 0000H จนถึง 01FFH ในการอ้างแอดเดรสของชิพซัพพอร์ตและอุปกรณ์ต่างๆที่ทำหน้าที่เป็น I/O บนเมนบอร์ด IBM/PC และรายละเอียดในรูป 3.2



รูป 3.2 การใช้งานแอดเดรสของพอร์ตบน IBM/PC

จากรูปข้างต้นจะเห็นว่าแอดเดรส 00C0H จนถึงแอดเดรส 01FFH นั้นไม่ได้ถูกใช้งานบนเมนบอร์ดของ IBM/PC ดังนั้นในกรณีนี้เราก็สามารถที่จะใช้งานแอดเดรสต่างๆเหล่านี้ได้ แต่อย่างไรก็ตามแอดเดรสเหล่านี้ก็ยังคงถูกดีโค้ดให้เป็นแอดเดรสที่ใช้ในการอ่านข้อมูลจากพอร์ต I/O บนเมนบอร์ดเท่านั้น ดังนั้นการใช้ค่าแอดเดรส 00C0H-01FFH กับพอร์ต I/O บนการ์ดหรือวงจรอินเทอร์เฟสที่เราสร้างขึ้นนั้น ต้องเป็นพอร์ตเอาท์พุทเพียงชนิดเดียวเท่านั้น กล่าวคือจะทำการอ่านข้อมูลจากพอร์ต I/O (ที่ไม่ได้อยู่ในเมนบอร์ด) ที่มีค่าแอดเดรสอยู่ในช่วง 00C0H-01FFH ไม่ได้

2. ในกลุ่มที่สองนี้ จะเป็นกลุ่มของพอร์ต I/O ที่ถูกใช้งานอยู่บนการ์ดที่ใช้เสียบบนสล็อตต่างๆของ IBM/PC สำหรับแอดเดรสของพอร์ตเหล่านี้จะเริ่มต้นจากแอดเดรส 0200H จนถึง 03FFH ซึ่งก็คือแอดเดรสที่มีบิท A9 เป็น "1" นั่นเอง สำหรับการใช้งานแอดเดรสของพอร์ด I/O ในกลุ่มนี้จะแสดงได้ดังรูปที่ 9-4

อย่างไรก็ตามการใช้งานแอดเดรสในกลุ่มนี้อาจจะเปลี่ยนแปลงไปได้ ทั้งนี้ขึ้นอยู่กับการใช้งานการ์ดต่างๆ ร่วมกับ IBM/PC โดยการ์ดที่ถูกออกแบบผลิตขึ้นใหม่นั้น อาจจะใช้ค่าแอดเดรสต่างๆที่เหลืออยู่นี้ได้ ดังนั้นก่อนที่จะทำการออกแบบวงจรอินเทอร์เฟสที่จำเป็นต้องใช้ค่าแอดเดรสสำหรับพอร์ต I/O จึงควรตรวจสอบดูก่อนว่าการ์ดต่างๆที่ใช้อยู่ในระบบ IBM/PC ที่เราใช้งานอยู่นั้นมีการ์ดใดบ้าง และการ์ดเหล่านั้นใช้งานแอดเดรสใดบ้าง จากนั้นจึงทำการออกแบบวงจรอินเทอร์เฟสโดยเลือกใช้เฉพาะ แอดเดรสที่ยังไม่ถูกใช้งาน

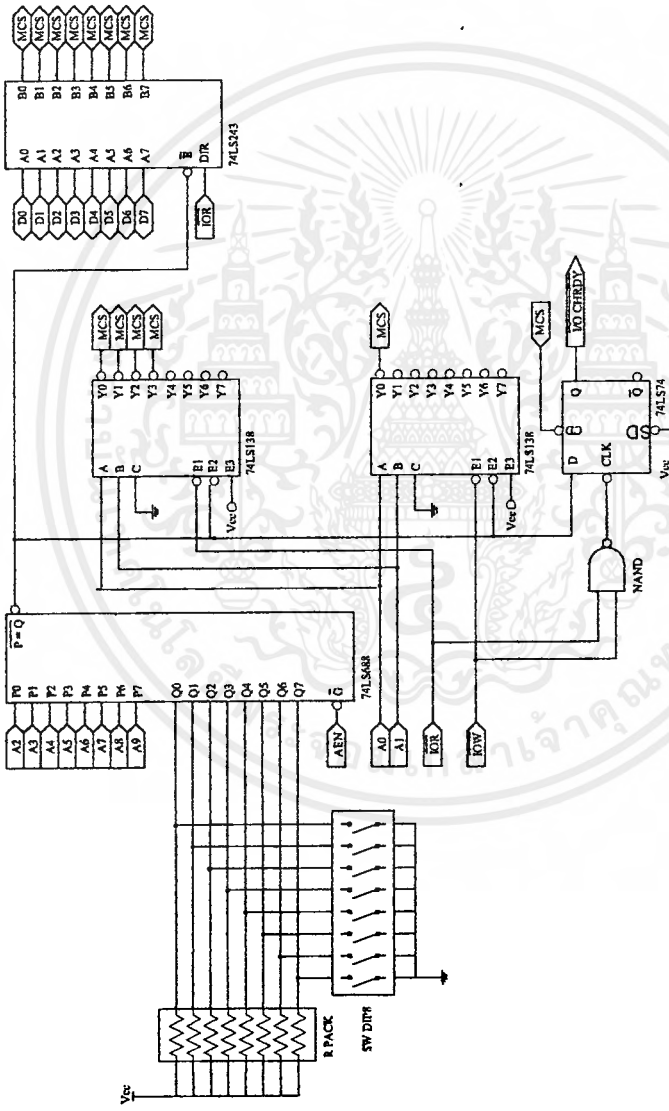
3.2.3 เทคนิคในการดีโค้ดแอดเดรสสำหรับพอร์ต I/O

ในหัวข้อต่างๆที่ผ่านมาข้างต้นนั้น ได้กล่าวถึงการอ้างแอดเดรสและการใช้งานแอดเดรสต่างๆของพอร์ต I/O ใน IBM/PC สำหรับในหัวข้อนี้จะกล่าวถึงวิธีการต่างๆที่ใช้ในการดีโค้ดแอดเดรสต่างๆ ให้เป็นไปตามที่เราต้องการ

Hex range	Usage	
000-00F	DMA chip 8237A-5	Assigned to system board components
020-021	Interrupt 8259A	
040-043	Timer 8253-5	
060-063	PPI 8255A-5	
080-083	DMA page registers	
0Ax	NMI mask register	
0Cx	Reserved	
0Ex	Reserved	
100-1FF	Not usable	
200-20F	Game control	Assigned to feature card port
210-217	Expansion unit	
220-24F	Reserved	
278-27F	Reserved	
2F0-2F7	Reserved	
2F8-2FF	Asynchronous communication (2)	
300-31F	Prototype card	
320-32F	Fixed disk	
378-37F	Printer	
380-38C	SDLC communications	
380-389	Binary synchronous communication (2)	
3A0-3A9	Binary synchronous communication (1)	
3B0-3BF	IBM monochrome display/printer	
3C0-3CF	Reserved	
3D0-3DF	Color/graphics	
3E0-3F7	Reserved	
3F0-3F7	Diskette	
3F8-3FF	Asynchronous communications (1)	

รูป 3.3 การจัดการพอร์ต I/O ของ IBM/PC

การดีโค้ดโดยใช้สวิตช์เลือกการดีโค้ดในแบบคงตัว (Fixed) มีข้อเสียอยู่บางประการคือ แอดเดรสที่เราเลือกใช้งานไว้นั้นอาจจะซ้ำกับแอดเดรสของการ์ดอื่นที่เรานำมาเพิ่มเข้าไปในระบบในภายหลังก็ได้ ซึ่งกรณีเช่นนี้เราต้องแก้ไขวงจรเพื่อหลีกเลี่ยงไปใช้แอดเดรสอื่นที่ยังว่างอยู่ และไม่ถูกใช้งานโดยการ์ดที่เพิ่มเข้าไปใหม่ ซึ่งยุ่งยากและต้องเสียเวลามากขึ้น ปัญหาเช่นนี้เราสามารถแก้ไขได้โดยใช้วงจรดีโค้ดที่สามารถเข้าไปเปลี่ยนแอดเดรสได้ โดยเพียงแต่เปลี่ยนตำแหน่งของสวิตช์ (ในที่นี้คือสวิตช์เลือก) ที่เซ็ทไว้ในวงจรเท่านั้น ดังแสดงตัวอย่างไว้ดังรูปที่ 3.4



รูป 3.4 ตัวอย่างวงจรดีโค้ดโดยใช้สวิตช์เลือก

จากรูปเป็นวงจรที่ทำการดีโค้ดกลุ่มแอดเดรสขนาด 8 แอดเดรส ซึ่งการเลือกกลุ่มแอดเดรสที่จะทำการดีโค้ดนี้จะทำได้โดยการเซ็ท DIP Switch ที่ขา Q0-Q7 ของ 74LS688

สำหรับหน้าที่ของ 74LS688 นี้จะทำการเปรียบเทียบค่าของอินพุท 2 ชุดที่ถูกส่งเข้ามาทางขา P0-P7 และขา Q0-Q7 ถ้าอินพุททั้ง 2 ชุดนี้เท่ากันแล้ว เอาท์พุทที่ขา P=Q จะให้เอาท์พุทเป็นลอจิก "0" จากในวงจรขา P0-P7 ของ 74LS688 ต่อกับแอดเดรสบิท A2-A9 ในขณะที่ขา Q0-Q7 ต่อกับความต้านทานที่ทำหน้าที่เป็น Pull Up (รักษาระดับแรงดันให้เป็นลอจิก "1" ไว้ในกรณีที่ไม่มีอินพุทใดๆเข้ามา) และขา Q0-Q7 นี้จะต่อกับปลายข้างหนึ่งของ DIP Switch ด้วย ส่วนปลายอีกข้างหนึ่งของสวิตช์เลือก (DIP Switch) นั้นจะต่อลงกราวด์ (ลอจิก "0") ไว้ ดังนั้นถ้าเราทำการ "ON" สวิตช์เลือกh ที่ต่อกับขาใดขานั้นก็จะได้รับลอจิก "0" ในขณะที่ถ้าสวิตช์เลือก ที่ต่อกับขาใดถูก OFF" ขานั้นจะได้รับลอจิก "1" และเนื่องจากอินพุทที่ขา P0-P7 (แอดเดรส A2-A9) ต้องเท่ากับอินพุทที่ขา Q0-Q7 ดังนั้นถ้าเราเปลี่ยนแปลงการเซ็ทสวิตช์เลือก เหล่านี้จะทำให้แอดเดรสบิท A2-A9 ซึ่งต่อกับขา P0-P7 นั้นต้องเปลี่ยนแปลงตามไปด้วยจึงจะทำให้เอาท์พุทของ 74LS688 แอดทีฟได้ ทำให้เราสามารถเปลี่ยนค่าแอดเดรสที่ต้องการจะดีโค้ดได้ง่ายกว่าวิธีการดีโค้ดแบบ Fixed สำหรับขา G นั้นจะต่อกับลอจิก "1" (+5 V) และขา P7 ต่อกับแอดเดรสบิท A9 ในกรณีเช่นนี้จึงเท่ากับเป็นการบังคับให้แอดเดรสที่จะทำการดีโค้ดได้นั้น จะต้องมีแอดเดรสบิท A9 เป็น "1" เท่านั้น ส่วนขา G จะต่อกับสัญญาณ AEN การต่อในลักษณะนี้ก็เพื่อป้องกันไม่ให้ 74LS688 ทำการดีโค้ดในระหว่างขบวนการ DMA นั่นเอง เอาท์พุทจากขา P=Q ของ 74LS688 นี้จะถูกนำไปใช้ในการอีนาเบิล 74LS688 ซึ่งทำหน้าที่ในการดีโค้ดแอดเดรส 8 แอดเดรสของกลุ่มแอดเดรสที่เราเลือก

วงจรในลักษณะนี้เราสามารถนำไปใช้ในการดีโค้ดในแบบ Fixed ได้โดยการนำเอาสวิตช์เลือก ออก จากนั้นถ้าอินพุทใดต้องการลอจิก "0" จึงจะใช้ตัวนำเชื่อมต่อระหว่างขั้วทั้งสองแทนการเซ็ทสวิตช์เลือก ให้ "ON" แต่ถ้าอินพุทใดต้องการลอจิก "1" ก็ปล่อยขั้วทั้งสองนั้นไว้

3.2.4 การสร้างเวทสเตท

ในการออกแบบที่เชื่อมต่อกับเครื่องคอมพิวเตอร์นั้น เรามักพบปัญหาเกี่ยวกับความเร็วในการทำงานของวงจรอินเทอร์เฟส ซึ่งโดยทั่วไปแล้วจะช้ากว่าการทำงานของเครื่องคอมพิวเตอร์ สำหรับใน IBM/PC ก็เช่นกัน วงจรอินเทอร์เฟสที่สร้างขึ้นอาจทำงานได้ช้าเกินกว่าที่จะตอบสนองต่อบัสไซเคิลต่างๆของ IBM/PC ได้ ด้วยเหตุนี้ภายใน IBM/PC จึงถูกออกแบบให้มีขาสัญญาณอินพุทเพื่อให้วงจรหรืออุปกรณ์ภายนอกที่ทำงานช้าเกินกว่าจะตอบสนองต่อบัสไซเคิลบน IBM/PC ได้นั้นใช้ในการหน่วงเวลาในบัสไซเคิลให้ช้าลง สำหรับสัญญาณอินพุทดังกล่าวนี้คือ สัญญาณ "READY" (หรือ I/O CHRDY) ซึ่งจะถูกต่อออกมาบนขาของสล็อตของ IBM/PC ด้วย

3.3 การสร้างเวทสเตท (Wait State)

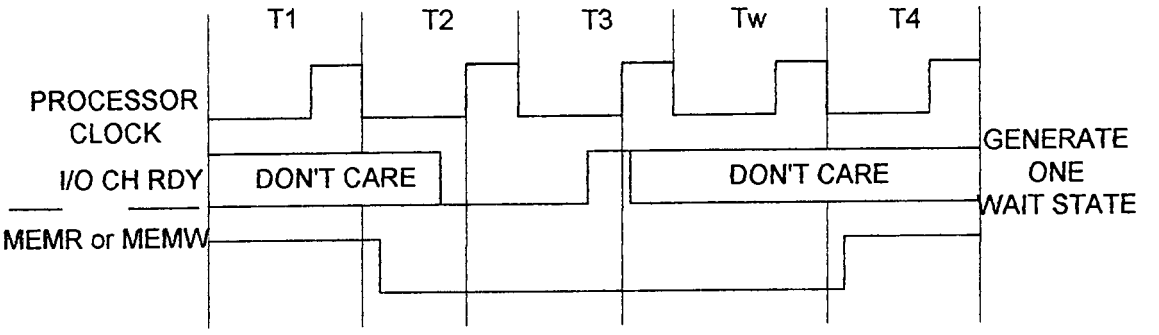
3.3.1 การสร้าง เวทสเตท ในบัสไซเคิลของ 8088

จากที่ได้กล่าวถึงในบทต่างๆที่ผ่านมานั้น จะเห็นได้ว่าโดยทั่วไปในแต่ละบัสไซเคิลของ 8088 จะใช้ช่วงเวลานานเท่ากับช่วงเวลาของคล็อก 4 ลูก คือ T1, T2, T3 และ T4 แต่สำหรับใน IBM/PC จะทำการเพิ่มจำนวนคล็อกในบางบัสไซเคิลขึ้นอีก 1ลูก ซึ่งคล็อกที่เพิ่มเข้าไปนี้มีชื่อเรียกว่า Tw ทำให้ในบางบัสไซเคิลมีช่วงเวลานานเท่ากับช่วงเวลาของคล็อก 5 ลูก ทั้งนี้ก็เพื่อหน่วงเวลาในบัสไซเคิลให้วงจรอินเทอร์เฟสสามารถทำงานตอบสนองต่อบัสไซเคิลเหล่านี้ได้ทัน สำหรับในหัวข้อนี้จะได้กล่าวถึงวิธีการสร้าง เวทสเตท (ในที่นี้คือ Tw) ในบัสไซเคิลที่เกี่ยวกับการอ่าน/เขียนข้อมูลลงในหน่วยความจำ และในบัสไซเคิลที่เกี่ยวกับการอ่าน/เขียนข้อมูลลงบนพอร์ต I/O ซึ่งมีวิธีการควบคุมสัญญาณ READY (I/O CHRDY) ที่แตกต่างกัน

3.3.2 การสร้าง เวทสเตท ในบัสไซเคิลที่เกี่ยวกับหน่วยความจำ

ในกรณีของบัสไซเคิลที่เกี่ยวกับการอ่านหรือเขียนข้อมูลลงในหน่วยความจำนี้ IBM/PC ไม่ได้ทำการเพิ่ม เวทสเตท ใดๆ ลงในบัสไซเคิลเหล่านี้เลย ดังนั้นช่วงเวลาในบัสไซเคิลที่เกี่ยวกับหน่วยความจำจึงมีช่วงเวลานานเท่ากับช่วงเวลาคล็อก 4 ลูก ตามช่วงเวลาปกติของบัสไซเคิลที่ 8088 สร้างขึ้น อย่างไรก็ตามในบางกรณี เช่นกรณีของหน่วยความจำที่เป็น Display Buffer บนการ์ดแสดงผลนั้นจำเป็นต้องใช้ เวทสเตท ในการทำงานด้วย ซึ่งในกรณีเช่นนี้วงจรที่อยู่บนการ์ดแสดงผลจะทำการสร้างเวทสเตท ที่ใช้กับหน่วยความจำที่เป็น Display Buffer นี้ขึ้นเอง

สำหรับในรูปที่ 3.5 จะแสดงถึงไดอะแกรมเวลา (Timing Diagram) ของสัญญาณต่างๆที่เกี่ยวข้องกับการสร้าง เวทสเตท ขึ้นในบัสไซเคิลที่เกี่ยวกับการอ่านหรือเขียนข้อมูลลงบนหน่วยความจำ วงจรสร้าง เวทสเตท บนเมนบอร์ดของ IBM/PC จะทำการตรวจสอบสถานะของสัญญาณที่ขา I/O CHRDY (เพื่อความสะดวกในการกล่าวถึงสัญญาณนี้ ต่อไปจะเรียกเป็นสัญญาณ READ แทน) ในช่วงเวลาขอบขาขึ้นของคล็อก T2 ของแต่ละบัสไซเคิล ซึ่งในช่วงเวลาขอบขาขึ้นของคล็อก T2 นี้ ถ้าวงจรสร้าง เวทสเตท ตรวจพบว่าสัญญาณ READY มีระดับลอจิกเป็น "1" ก็จะไม่มีการสร้าง เวทสเตท หรือ Tw ใดๆขึ้น แต่ถ้าวงจรนี้พบว่าระดับลอจิกของสัญญาณ READY เป็น "0" แล้ววงจรสร้าง เวทสเตท ก็จะทำการสร้าง เวทสเตท ขึ้นในบัสไซเคิลนั้นๆ โดย Tw จะถูกสร้างขึ้นในระหว่างคล็อก T3



รูป 3.5 ไดอะแกรมเวลาของเวทสเตทในบัสไซเคิลของการอ่านหรือเขียนข้อมูลลงบนหน่วยความจำ

ละ T4 ของบัสไซเคิลนั้น อย่างไรก็ตามเนื่องจากส่วนต่างๆของระบบ เช่นวงจรแลทช์ (74LS74) จะมีค่าดีเลย์ (Delay) อยู่ด้วย ดังนั้นเพื่อแก้ปัญหาดังกล่าวนี้และทำให้เราแน่ใจได้ว่าวงจรสร้าง เวทสเตท สามารถทำงานตามที่เราต้องการได้ จึงจำเป็นต้องทำให้สัญญาณ READY มีระดับลอจิกที่เราต้องการ (0 หรือ 1) ก่อนที่วงจรสร้าง เวทสเตท จะทำการตรวจสอบสัญญาณ READY กล่าวคือในกรณีที่เราไม่ต้องการให้มี Tw เกิดขึ้นในบัสไซเคิลแล้ว เราจะต้องทำให้สัญญาณ READY มีระดับลอจิก "1" เป็นเวลาอย่างน้อย 75 นาโนวินาที ก่อนขอบขาขึ้นของคล็อก T2 ในบัสไซเคิลนั้นๆ แต่ถ้าเราต้องการให้เพิ่ม Tw ขึ้นในบัสไซเคิลแล้ว เราก็จะต้องทำให้สัญญาณ READY มีระดับลอจิกเป็น "0" เป็นเวลาอย่างน้อย 60 นาโนวินาที ก่อนขอบขาขึ้นของคล็อก T2 ในบัสไซเคิลนั้นๆ และในกรณีที่เราต้องการเพิ่ม Tw ขึ้นอีกในบัสไซเคิลนั้นก็เพียงแต่รักษาระดับลอจิกของสัญญาณ READY ให้เป็น "0" ไว้จนถึงช่วงเวลาขอบขาขึ้นของคล็อกลูกต่อไป (ในกรณีของ Tw ลูกที่สอง ก็คือคล็อก T3) เท่านั้น แต่ถ้าเราไม่ต้องการเพิ่ม เวทสเตท ขึ้นอีก ก็เพียงแต่ทำให้ระดับลอจิกของสัญญาณ READY กลับเป็น "1" ก่อนที่จะถึงช่วงขอบขาขึ้นของคล็อกลูกต่อไปเป็นเวลาอย่างน้อย 75 นาโนวินาที ดังนั้นจากไดอะแกรมเวลาข้างต้น ซึ่งต้องการเพิ่ม Tw เข้าไปเพียงลูกเดียว จึงทำให้ระดับลอจิกของสัญญาณ READY กลับเป็น "1" ก่อนช่วงเวลาขอบขาขึ้นของคล็อก T3 เป็นเวลานาน 75 นาโนวินาที วงจรที่ใช้สำหรับการส่งสัญญาณให้กับขา I/O CHRDY เพื่อให้วงจรบนเมนบอร์ดของ IBM/PC ทำการสร้าง เวทสเตท หรือ Tw ให้ตั้งแต่ 1 ถึง 6 ลูก ในบัสไซเคิลที่เกี่ยวกับการอ่านหรือเขียนข้อมูลลงในหน่วยความจำ

หลังจากวงจรดีโค้ดทำการดีโค้ดแอดเดรสแล้ว จะส่งสัญญาณดีโค้ดให้กับ 74S74 หลังจากนั้นเมื่อสัญญาณ IOR หรือ IOW แอคทีฟ (ลอจิก "0") ก็จะทำให้ 74S74 ส่งสัญญาณดีโค้ด (ลอจิก "1") ที่ได้รับจากวงจรดีโค้ดแอดเดรสออกไปยังเอาท์พุท Q (ขา 5) ผ่าน inverter ไปให้กับขา I/O CHRDY บนสล็อตของ IBM/PC ซึ่งจะเป็นผลให้วงจรบนเมนบอร์ดทำการสร้าง เวทสเตทให้ ส่วนจำนวนของ

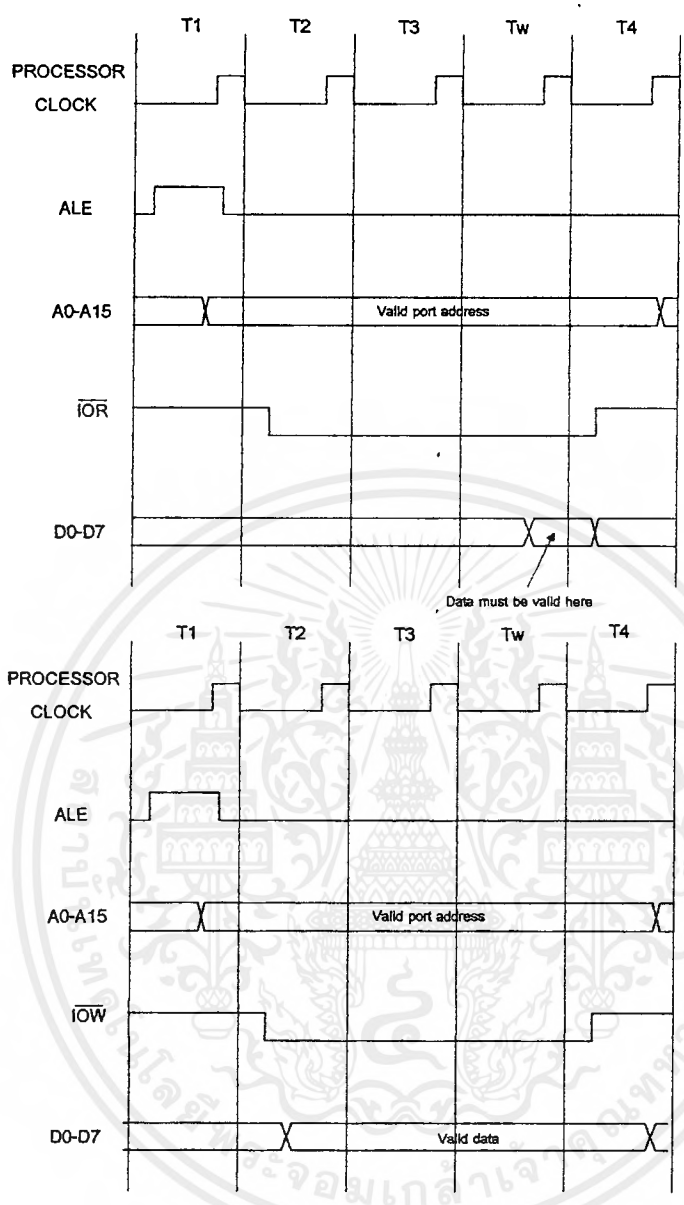
Tw ที่ถูกสร้างขึ้นนั้นจะถูกกำหนดโดยวงจรในส่วนของ 74LS174 ซึ่งถูกออกแบบให้ทำงานเป็น shift register

เราสามารถกำหนดให้การขอ เวทสเตท เกิดขึ้นเฉพาะในบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำเท่านั้นได้ โดยการนำเอา IOW เข้ากับขาอินพุท CLK ของ 74LS74 โดยตรง

3.3.3 การสร้าง เวทสเตท ในบัสไซเคิลที่เกี่ยวกับพอร์ต I/O

สำหรับในบัสไซเคิลที่เกี่ยวกับพอร์ต I/O นี้ โดยทั่วไปก็จะมีช่วงเวลาเท่ากับช่วงเวลาของบัสไซเคิลที่เกี่ยวกับหน่วยความจำคือ ใช้เวลานานเท่ากับช่วงเวลาของคล็อก 4 ลูก แต่วงจรภายใน IBM/PC จะทำการเพิ่มช่วงเวลาในบัสไซเคิลที่เกี่ยวกับพอร์ต I/O นี้ขึ้นอีก ซึ่งจะทำให้ช่วงเวลาในบัสไซเคิลนี้นานเท่ากับช่วงเวลาของคล็อก 5 ลูก สำหรับคล็อกลูกที่เพิ่มเข้าไปในบัสไซเคิลนี้ก็คือ เวทสเตท หรือ Tw นั่นเอง ทั้งนี้ก็เพื่อให้วงจรอินเทอร์เฟส I/O ต่างๆ สามารถตอบสนองต่อการทำงานในบัสไซเคิลได้ทัน อย่างไรก็ตามวงจรอินเทอร์เฟสที่เราสร้างขึ้น ก็อาจยังไม่สามารถทำงานได้เร็วพอที่จะตอบสนองต่อการทำงานในบัสไซเคิลที่ถูกเพิ่ม เวทสเตท เข้าไปนี้ได้ทัน ดังนั้นใน IBM/PC จึงยังคงยอมให้วงจรภายนอกสามารถขอ เวทสเตท เพิ่มขึ้นได้อีก โดยผ่านทางขา I/O CHRDY เช่นเดียวกับในกรณีของบัสไซเคิลที่เกี่ยวกับหน่วยความจำ แต่ยังคงมีรายละเอียดปลีกย่อยบางประการในการขอ เวทสเตท จากวงจรเมนบอร์ดของ IBM/PC กล่าวคือในขณะที่ในบัสไซเคิลที่เกี่ยวกับหน่วยความจำนั้น วงจรที่ทำการสร้าง เวทสเตท จะทำการตรวจสอบสัญญาณ READY นี้ในช่วงเวลาขอบขาขึ้นของคล็อก T2 แต่ในบัสไซเคิลที่เกี่ยวกับพอร์ต I/O วงจรที่ทำการสร้าง เวทสเตท จะตรวจสอบสัญญาณ READY นี้ในช่วงเวลาขอบขาขึ้นของคล็อก T3 ส่วนลักษณะอื่นๆของสัญญาณที่ใช้ในการขอ เวทสเตท นั้นจะเหมือนกับในบัสไซเคิลที่เกี่ยวกับหน่วยความจำ

จากไดอะแกรมเวลารูป 3.6 นั้นถ้าต้องการเพิ่ม Tw เข้าไปในบัสไซเคิลอีก (จากเดิมที่มี Tw อยู่แล้ว 1 ลูก) เราจะต้องทำให้ระดับลอจิกของสัญญาณ READY เป็น "0" เป็นเวลา 60 นาโนวินาที ก่อนช่วงเวลาขอบขาขึ้นของคล็อก T3 ของบัสไซเคิลที่เกี่ยวกับ พอร์ต I/O และถ้าไม่ต้องการให้มี Tw ใดๆเกิดขึ้นอีก ก็จะต้องทำให้ระดับลอจิกของสัญญาณ READY เป็น "1" เป็นเวลา 75 นาโนวินาทีก่อนช่วงเวลาขอบขาขึ้นของคล็อก T3 ของบัสไซเคิลที่เกี่ยวกับพอร์ต I/O สำหรับตัวอย่างวงจรที่ใช้ในการขอ เวทสเตท จากวงจรเมนบอร์ด



รูป 3.6 ไซเคิลแสดงการอ่านและเขียนข้อมูลกับพอร์ต I/O

จากวงจรข้างต้นนั้น จะเป็นวงจรที่สามารถใช้ในการขอให้วงจรบนเมนบอร์ดทำการสร้าง เวทสเตท ในบัสไซเคิลของการอ่าน/เขียนข้อมูลลงบนพอร์ต I/O ได้ตั้งแต่ 1-5 ลูก สำหรับการทำงานของวงจรนี้คล้ายกับในวงจรที่ใช้สำหรับการขอ เวทสเตท ในบัสไซเคิลที่เกี่ยวกับหน่วยความจำกล่าวคือ วงจรในส่วนที่ทำการดีโค้ดแอดเดรสจะทำหน้าที่ในการกำหนดบล็อกแอดเดรสของพอร์ต I/O ที่ต้องการจะให้วงจรบนเมนบอร์ดสร้าง เวทสเตท ขึ้นโดยวงจรในส่วนนี้จะสร้างสัญญาณดีโค้ดส่งให้กับวงจรในส่วนอื่นๆเมื่อเกิดบัสไซเคิลของการอ่าน/เขียนข้อมูลลงบนพอร์ต I/O ที่อ้างถึง แอดเดรสของพอร์ตที่อยู่ในบล็อกแอดเดรสของพอร์ต I/O ที่กำหนด

ถ้าเราต้องการวงจรที่ทำการขอ เวทสเตท เฉพาะในกรณีของบัสไซเคิลที่ทำการอ่านข้อมูลจากพอร์ต I/O เราก็สามารถจะทำได้โดยการนำเอาเกท NAND ที่ขา CLK ของ 74S74 แทน และถ้าต้องการวงจรที่ทำการขอ เวทสเตท เฉพาะในกรณีของบัสไซเคิลที่ทำการเขียนข้อมูลลงบนพอร์ต I/O เราก็สามารถจะทำได้โดยการนำเอาเกท NAND ที่ขา CLK ของ 74S74 ออกและนำเอาเฉพาะสัญญาณ LOW เพียงสัญญาณเดียวต่อผ่าน inverter เข้ากับขา CLK ของ 74L74 แทน

บทที่ 4
ระบบตรวจจับเวลา 4 ช่องทางโดยไมโครโปรเซสเซอร์

จากแผนภาพของระบบตรวจจับเวลาโดยใช้ชิปไมโครคอนโทรลเลอร์ในบทที่ 1 จะเห็นว่าสามารถที่จะแบ่งแยกวงจรได้เป็น 3 ส่วนใหญ่ คือ

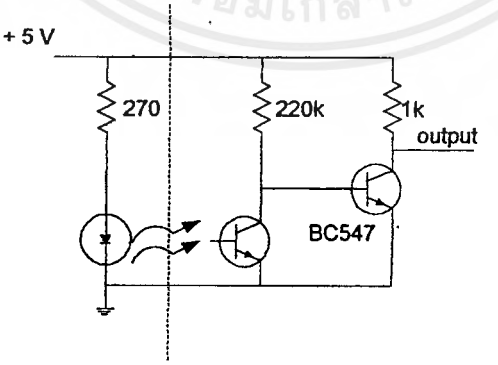
1. ส่วนวงจรชุดรับส่งอินฟราเรด
2. ส่วนของชิปไมโครคอนโทรลเลอร์ในรูปแบบของการ์ดติดต่อกับคอมพิวเตอร์
3. ส่วนของวงจรอินเตอร์เฟสกับสล็อตของคอมพิวเตอร์เพื่อแสดงผลและประมวลผล

4.1 ส่วนวงจรชุดรับส่งอินฟราเรด

วงจรชุดรับส่งอินฟราเรดในระบบตรวจจับเวลานี้ ประกอบด้วย 4 ส่วนหลักสำคัญ ได้แก่

- วงจรชุดส่ง
- วงจรชุดรับ
- ส่วนแยกกราวด์ของแหล่งจ่ายไฟกับกราวด์ของคอมพิวเตอร์
- ส่วนปรับแต่งสัญญาณและตัดสัญญาณรบกวน

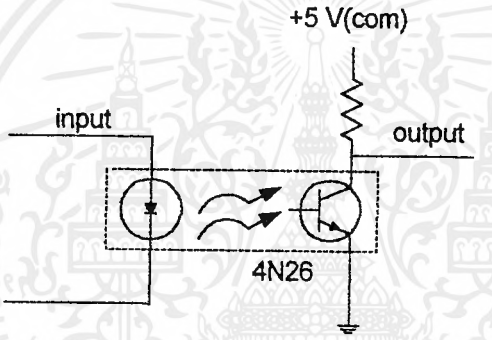
วงจรชุดส่งอินฟราเรดจะใช้หลอดอินฟราเรดทำหน้าราที่เป็นตัวส่งรังสีอินฟราเรด ดังวงจรรูปที่ 4.1 และวงจรชุดรับจะใช้โฟโต้ทรานซิสเตอ ร์ที่ทำหน้าที่เป็นตัวรับแสงอินฟราเรดจากตัวส่ง ดังวง



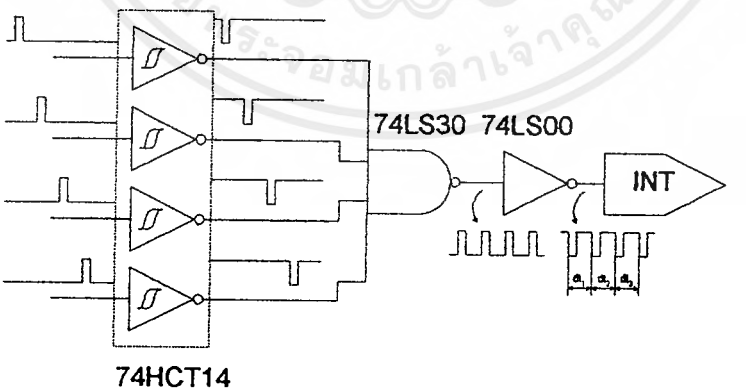
รูป 4.1 วงจรชุดรับ-ส่งอินฟราเรด

จรรูปที่ 4.1 การเลือกตัวรับและตัวส่งนี้ (หลอดอินฟราเรดกับโฟโต้ทรานซิสเตอร์) ต้องเลือกให้ตัวส่งและตัวรับมีความเหมาะสมที่จะสามารถใช้งานร่วมกันได้ ในโครงงานระบบตรวจจับเวลานี้จะประกอบด้วยตัวรับและตัวส่งจำนวน 4 ชุด โดยมีวงจรชุดตัวรับส่งอินฟราเรดดังรูปที่ 4.1 เอาต์พุท (ขาคอลเลคเตอร์ของทรานซิสเตอร์เบอร์ BC547) จะเป็นอินพุทของวงจรส่วนแยกกราวด์แหล่งจ่ายไฟกับกราวด์คอมพิวเตอร์

ส่วนวงจรแยกกราวด์แหล่งจ่ายไฟกับกราวด์คอมพิวเตอร์จะใช้วงจรดังรูปที่ 4.2 โดยใช้ไอซีเบอร์ 4N26 เป็นตัวทำหน้าที่แยกกราวด์โดยอินพุทของวงจรนี้ก็คือเอาต์พุทของวงจรชุดรับ (ขาคอลเลคเตอร์ของทรานซิสเตอร์เบอร์ BC547) โดยที่การต่อวงจรต้องพึงระวังเพราะจะใช้กราวด์ของ 2 ที่ กล่าวคือกราวด์ของแหล่งจ่ายไฟจะใช้คู่กับสัญญาณอินพุท ส่วนกราวด์ของคอมพิวเตอร์จะใช้คู่กับสัญญาณเอาต์พุทและ + 5 โวลท์ ดังรูป 5.2



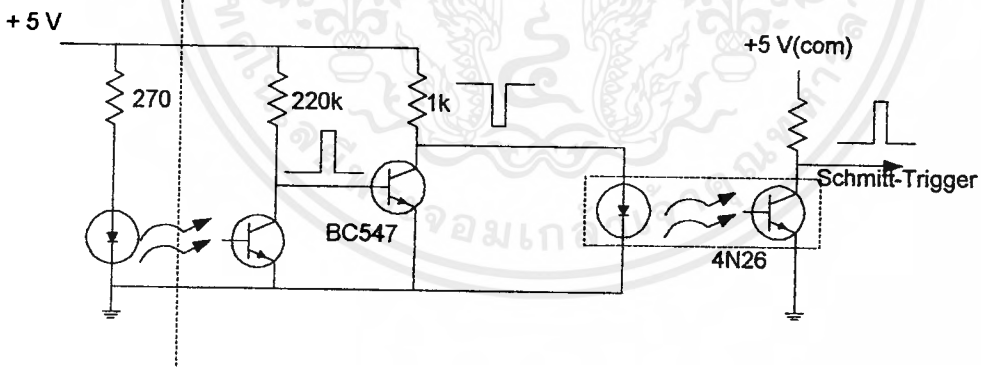
รูป 4.2 วงจรแยกกราวด์แหล่งจ่ายไฟกับกราวด์คอมพิวเตอร์



รูป 4.3 วงจรส่วนปรับแต่งสัญญาณและตัดสัญญาณรบกวน

ส่วนปรับแต่งสัญญาณและตัดสัญญาณรบกวนเป็นวงจรที่ใช้ โดยมีวงจรดังรูปที่ 4.3 โดยใช้ Schmitt-Trigger inverter :74HCT14 โดยจะนำเอาเอาท์พุทที่ได้จากส่วน เอาท์พุทของ 4N26 มาทำการปรับแต่งสัญญาณ สุดท้ายของวงจรนี้หลังจากผ่านแนนเกท (NAND gate) และอินเวร์เตอร์ตามลำดับแล้ว จะนำเข้าสู่การขออินเตอร์รัปต์ในส่วนของไมโครคอนโทรลเลอร์ต่อไป

การทำงานของส่วนชุดรับส่งอินฟราเรดนี้จะเริ่มจากชุดรับส่งอินฟราเรด หลอดอินฟราเรดที่ทำหน้าที่เป็นตัวส่งแสงอินฟราเรด จะถูกไบอัสด้วยกระแสคงที่ตลอดเวลา วัดศักดาตกคร่อมหลอดอินฟราเรดได้ 0.55 โวลท์ ดังนั้นจะมีกระแสไบอัส $\frac{5-.55}{270} = 16$ มิลลิแอมป์ ส่วนโฟโต้ทรานซิสเตอร์ต่อเป็นวงจรคอมมอนอิมิตเตอร์ (common emitter) ถูกไบอัสที่เบส (base) ด้วยแสงอินฟราเรด ขณะเวลาทำงานทั้งหลอดอินฟราเรดและโฟโต้ทรานซิสเตอร์จะถูกไบอัสด้วยไฟเลี้ยง +5 โวลท์ตลอดเวลา ดังนั้นทำให้ โฟโต้ทรานซิสเตอร์ถูกไบอัสที่ขาเบส ทรานซิสเตอร์จะทำงานในสภาวะ on ทำให้ได้ศักดาเอาต์พุทที่ขาคอลเลคเตอร์ (collector) ประมาณ 0 โวลท์ เมื่อมีวัตถุตัวผ่านลำแสงอินฟราเรดจะทำให้โฟโต้ทรานซิสเตอร์จะไม่ถูกไปอัสที่ขาเบสช่วงเวลาหนึ่งซึ่งเป็นช่วงเวลาที่ทรานซิสเตอร์ off ศักดาที่เอาต์พุทจะเพิ่มขึ้น และตกลงมาเป็น 0 อีกเมื่อวัตถุผ่านไปคือเมื่อโฟโต้ทรานซิสเตอร์สามารถรับลำแสงอินฟราเรดได้ ซึ่งเอาต์พุทที่เพิ่มขึ้นขณะวัตถุตัดผ่านนี้จะอยู่ในช่วงเวลาหนึ่งจนกระทั่งวัตถุตัดผ่านไปแล้วจึงเกิดในลักษณะของพัลส์สัญญาณที่จุดต่าง ๆ ดังรูป 4.4



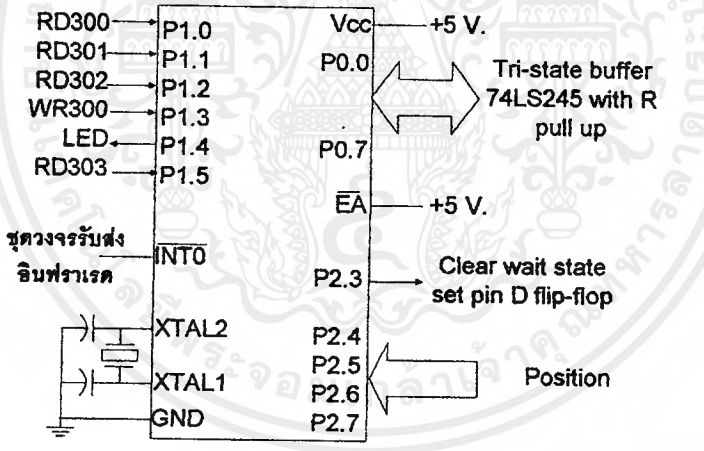
รูป 4.4 สัญญาณที่เกิดขึ้นที่จุดต่างๆ ของส่วนชุดรับส่งอินฟราเรด

หลังจากได้สัญญาณจากชุดรับส่งอินฟราเรดแล้ว สัญญาณจะถูกนำมาทำการแยกกราวด์ของแหล่งจ่ายไฟกับกราวด์คอมพิวเตอร์โดยที่หลังจากผ่านชุดรับส่งอินฟราเรดแล้วสัญญาณที่ได้จะใช้กราวด์ของคอมพิวเตอร์เป็นตัวอ้างอิงแทนกราวด์ของแหล่งจ่ายไฟ หลังจากที่ได้สัญญาณจากการตัดผ่านลำแสงชุดรับส่งอินฟราเรดแต่ละชุดทั้ง 4 ชุดแล้วจะถูกนำมาผ่าน Schmitt-Trigger Inverter

(74LS14) เพื่อปรับแต่งรูปสัญญาณที่ได้มาจากโฟโต้ทรานซิสเตอร์ และจะทำการตัดสัญญาณรบกวนที่อาจเกิดจากแสงอาทิตย์หรือหลอดฟลูออเรสเซนต์และลดระดับสัญญาณลงเป็นทีทีแอล (TTL) จากนั้นสัญญาณทั้ง 4 จะถูกนำไปรวมกันโดยผ่านแนนเกท (NAND gate) (74LS30) และผ่านตัวกลับสัญญาณ (inverter:74LS00) อีกครั้งหนึ่งเพื่อกลับระดับสัญญาณ สัญญาณ ณ.จุดนี้จะนำไปเข้าขา อินเตอร์รัปต์ขอไมโครคอนโทรลเลอร์ เพื่อหาช่วงเวลาระหว่างพัลส์โดยตรวจจับการเปลี่ยนแปลงของระดับสัญญาณจากลอจิก 1 เป็นลอจิก 0 โดยการทำงานที่กล่าวมาข้างต้นนั้นสามารถแสดงได้ดังรูปที่ 4.3 ซึ่งจะแสดงให้เห็นถึงรูปร่างสัญญาณที่จุดต่าง ๆ

4.2 ส่วนของชิปไมโครคอนโทรลเลอร์

ในรูปที่ 4.3 เป็นชิปไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 89C51 มีหน่วยความจำสำหรับเก็บข้อมูลภายในจำนวน 128 ไบท์ และหน่วยความจำสำหรับเก็บโปรแกรม ภายในจำนวน 4 กิโลไบต์ มีการเชื่อมต่อกับส่วนต่าง ๆ ดังรูปที่ 4.5



การเชื่อมต่อส่วนต่าง ๆ กับชิปไมโครคอนโทรลเลอร์

รูป 4.5 การต่อส่วนต่างๆ กับMCS-51

ขาพอร์ต 0 (ขา 32-39) ใช้เป็นขาสำหรับพอร์ต 0 ขนาด 8 บิต (P0.0-P0.7) แบบ Open Drain Bidirectional พอร์ต 0 นี้จะใช้เป็นทั้งอินพุทและเอาต์พุทพอร์ต ซึ่งนำมาเชื่อมต่อกับบัพเฟอร์ 2 ทิศทางเพื่อส่งและรับข้อมูลผ่านทางสล็อต โดยมีการพูลอัพเอาไว้ด้วย

ขาพอร์ต 1 (ขา 1-8) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 1 (P1.0-P1.7) พอร์ต 1 ในระบบนี้ห้าบิทล่างจะใช้เป็นอินพุทพอร์ตในการตรวจสอบสัญญาณที่ได้จากตัวถอดรหัส (decode) ซึ่งมีสัญญาณ RD300 RD301 RD302 RD303 และ WR300 ส่วนพอร์ต 1.4 จะใช้เป็นเอาต์พุทพอร์ตซึ่งจะต่อกับ LED โดยจะให้ติดและดับทุก ๆ 1 วินาที

ขาพอร์ต 2 (ขา 21-28) ใช้เป็นขาสำหรับพอร์ต 2 (P2.0-P2.7) ขนาด 8 บิต แบบ Open Drain Bidirectional พอร์ต 2.3 ใช้เป็นเอาต์พุทพอร์ต นำมาใช้ในการเคลียร์เวทสเตท (wait state) โดยเชื่อมต่อกับขาเซตของดีฟลิปฟลอบ (D-flip flop) พอร์ท 2.4-2.7 ใช้เป็นอินพุทพอร์ทในการตรวจจับตำแหน่งของชุดรับส่งอินฟราเรดเมื่อวัตถุตัดผ่าน

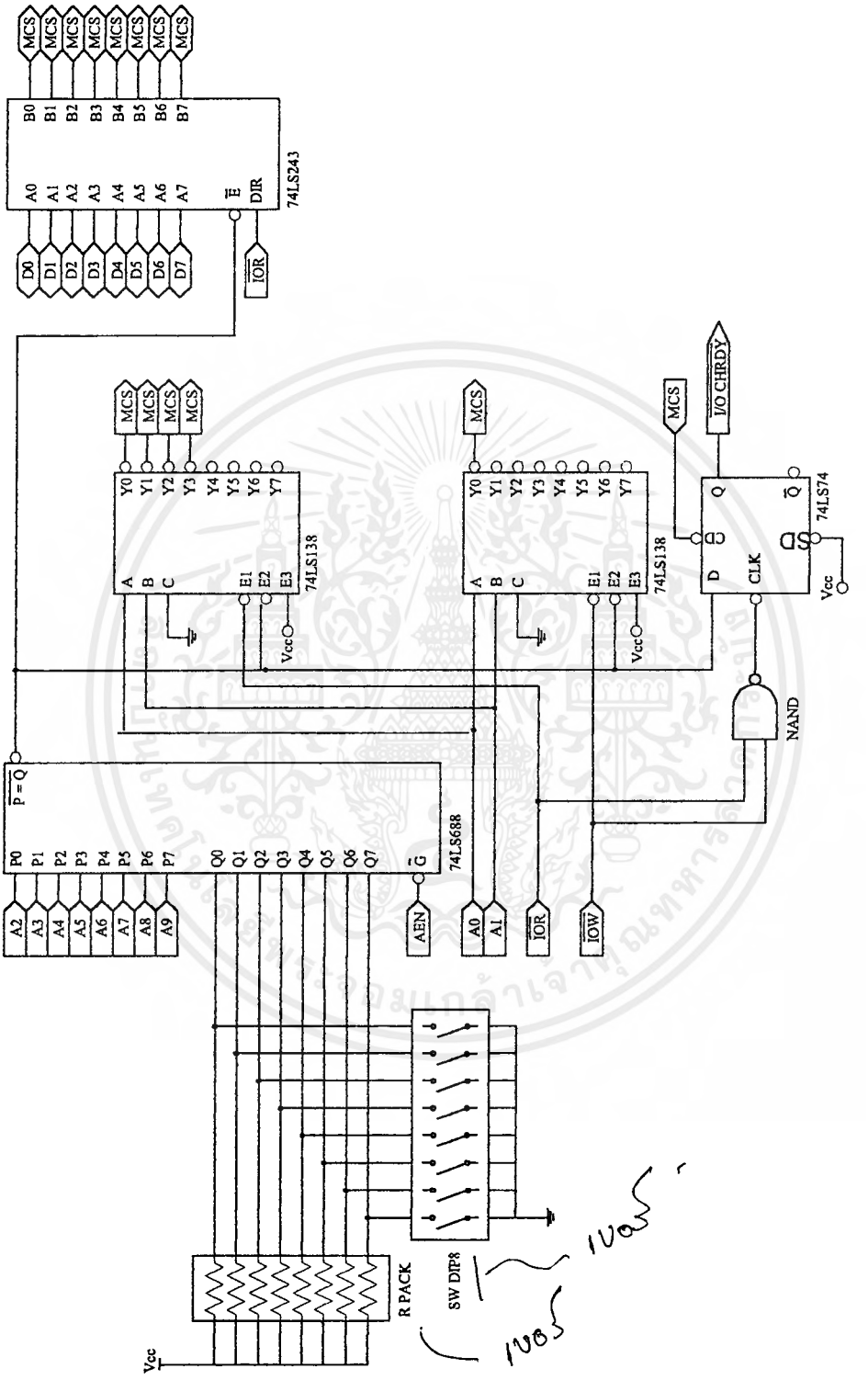
ขา INT0 จะเป็นขาที่ใช้ในการตรวจสอบการอินเตอร์รัปต์จากภายนอก ซึ่งจะเชื่อมต่อกับชุดรับส่งอินฟราเรด

การทำงานของส่วนนี้คือเมื่อมีการอินเตอร์รัปต์จากภายนอกซึ่งเกิดจากวัตถุตัดผ่านลำแสงอินฟราเรดของชุดรับส่งอินฟราเรด ชิปไมโครโปรคอนโทรลเลอร์จะทำการจับเวลาโดยใช้ไทม์เมอร์โหมด 0 ของตัวชิปเป็นฐานเวลา และนำข้อมูลซึ่งเป็นผลต่างของเวลาที่ได้ไปเก็บในหน่วยความจำภายในชิป ต่อมาเมื่อคอมพิวเตอร์ต้องการที่จะแสดงผลก็จะทำการส่งสัญญาณการอ่าน ซึ่งประกอบไปด้วยขาของสัญญาณ RD300 RD301 RD302 ไปยังชิปไมโครคอนโทรลเลอร์ ชิปจะทำการส่งข้อมูลเวลาที่เก็บไว้ในหน่วยความจำผ่านพอร์ต 0 ไปยังคอมพิวเตอร์ คอมพิวเตอร์ก็จะสามารถแสดงผลได้ สำหรับการใช้คำสั่งหยุดเดินไทม์เมอร์และรีเซตชิปไมโครคอนโทรลเลอร์ โดยผ่านทางคอมพิวเตอร์ คอมพิวเตอร์จะส่งสัญญาณการเขียน ซึ่งประกอบด้วยขาสัญญาณ WR300และข้อมูลคำสั่งไปยังพอร์ต 0 ของชิปไมโครคอนโทรลเลอร์ จากนั้นชิปไมโครคอนโทรลเลอร์จะทำการตรวจสอบข้อมูลคำสั่งที่ได้จากคอมพิวเตอร์ว่าเป็นการหยุดเดินไทม์เมอร์หรือรีเซตชิปเอง และชิปจะทำงานตามคำสั่งที่ได้ต่อไป

4.3 การอินเทอร์เฟส กับ IBM PC

ในการติดต่อรับส่งข้อมูลระหว่างชิปไมโครคอนโทรลเลอร์กับคอมพิวเตอร์จำเป็นต้องมีส่วนเชื่อมต่อกับระบบบัสของ IBM PC โดยวงจรอินเทอร์เฟสนี้จะดึงขาสัญญาณต่าง ๆ บนสล็อตมาใช้ดังนี้ คือ ขา A0-A9, D0-D7, I/O CHRDY, IOR, IOR, AEN, +5V, GND

จากหัวข้อที่ผ่านมาในบทที่ 3 แอดเดรสของพอร์ต I/O ที่สามารถใช้กับการ์ดต่าง ๆได้อยู่ในช่วง 0200H-03FFH และช่วงแอดเดรสที่ว่างสามารถนำมาใช้กับการ์ดที่ออกแบบขึ้นมาอยู่ในช่วง 0300H-031FH จากรูป 3.3 และเลือกใช้ 0300H-0302H ซึ่งสามารถเปลี่ยนแปลงได้โดยสวิตช์เลือก ขา



รูป 4.6 วงจรระบบตรวจจับเวลาโดยใช้ไมโครคอนโทรลเลอร์

แอดเดรสที่นำมาดีโค้ดจึงได้แก่ A2-A9 ซึ่งจะถูกนำมาเปรียบเทียบกับลอจิกที่สวิตซ์เลือกโดยชิปเบอร์ 74LS688 (8 bit equality comparator) ถ้าลอจิกที่ได้เท่ากันจะได้เอาต์พุทเป็นสัญญาณเลือกกลุ่ม (group select) ดังรูป 4.6

ส่วนแอดเดรสบัสเส้น A0-A1 จะถูกนำไปดีโค้ดโดยชิปเบอร์ 74LS138 ทั้ง 2 ตัวดังรูป 4.6 โดยใช้สัญญาณ IOR และ IOW จากสล็อตมาเป็นสัญญาณแอนาเบิล (enable) ร่วมกับสัญญาณเลือกกลุ่ม ทำให้ได้สัญญาณ RD300 RD301 RD302 RD303 และ WR300 ซึ่งสัญญาณเหล่านี้จะนำไปเข้ายังพอร์ต 1 ของชิปไมโครคอนโทรลเลอร์ตรวจสอบการติดต่อเพื่ออ่านหรือเขียนข้อมูลกับชิปไมโครคอนโทรลเลอร์

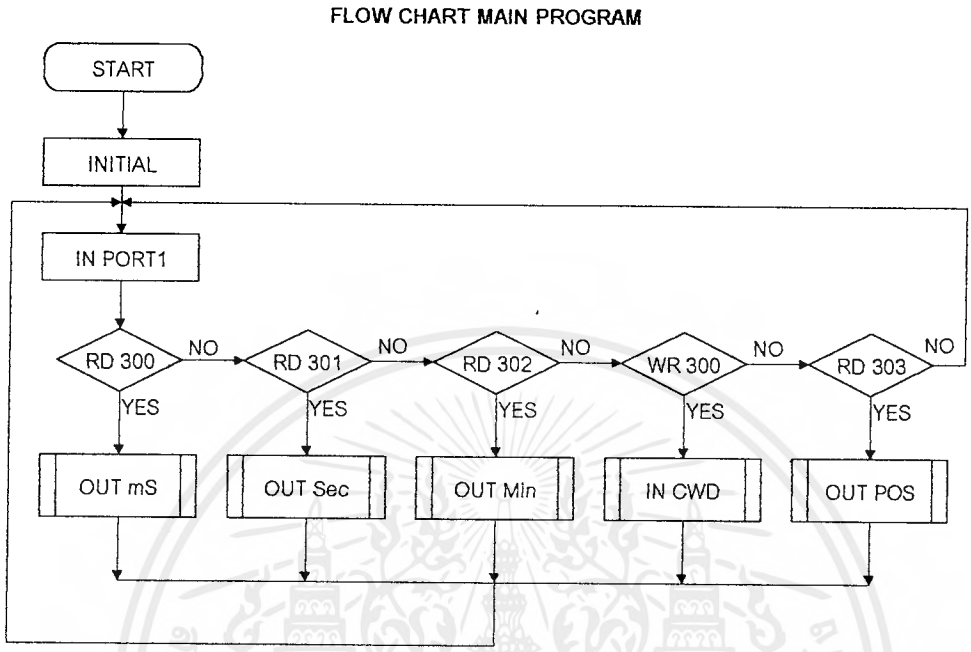
ในการรับส่งข้อมูล 8 บิตระหว่างชิปไมโครคอนโทรลเลอร์กับระบบบัสของคอมพิวเตอร์นั้นใช้ชิบเบอร์ 74LS245 (octal 3 state noninverting bus transceiver) ดังรูป 4.4 โดยสามารถทำการรับส่งข้อมูลแบบซิงโครนัสได้ 2 ทิศทางควบคุมโดยขา DIR (direction) ถ้าเป็นลอจิก 1 ข้อมูลจะถูกส่งจากพอร์ต A ไปยังพอร์ต B และถ้าเป็นลอจิก 0 จะส่งในทิสตรงกันข้ามกับตอนแรก จากวงจรรูป 4.4 ระบบตรวจจับเวลาโดยใช้ไมโครคอนโทรลเลอร์ จะใช้สัญญาณ IOR ควบคุมทิศทางร่วมกับขาสัญญาณ Group Select ทำการเอนาเบิล เมื่อสัญญาณ IOR เ ป็นลอจิก 0 คือแอกทีฟ จะส่งข้อมูลจากชิปไมโครคอนโทรลเลอร์มายังคอมพิวเตอร์และเมื่อมีสัญญาณ IOW จะเกิดสัญญาณ Group Select มาเอนาเบิลเกิดการส่งข้อมูลจากคอมพิวเตอร์ไปยัง MCS-51

เนื่องจากช่วงเวลาการเขียน อ่านข้อมูลจากพอร์ต I/O ของคอมพิวเตอร์เร็วกว่าช่วงเวลาการทำงานของ MCS-51 มาก MCS-51 ไม่สามารถส่งหรือรับข้อมูลได้ทัน จึงต้องมีการของเวทสเตทเพื่อเพิ่มช่วงเวลานี้โดยใช้ชิปเบอร์ 74LS74 (D- flip flop) เมื่อคอมพิวเตอร์ต้องการติดต่อกับไมโครคอนโทรลเลอร์จะมีสัญญาณอ่านหรือเขียนมาเป็นสัญญาณนาฬิกาให้ D flip flop และจะมีสัญญาณ Group Select เป็นอินพุททำให้ได้เอาต์พุทที่ขา Q เป็นสถานะลอจิก 0 ไปเข้าขา I/O CHRDY ทำการขอเวทสเตท และเมื่อชิปไมโครคอนโทรลเลอร์ส่งหรือรับข้อมูลเสร็จแล้วจะส่งสัญญาณมาที่ขา Set ของ D flip flop เพื่อตั้งค่าเอาต์พุทขา Q เป็นลอจิก 1 เป็นการยกเลิกเวทสเตท

4.4 โฟลชาร์ทและโปรแกรมการทำงาน

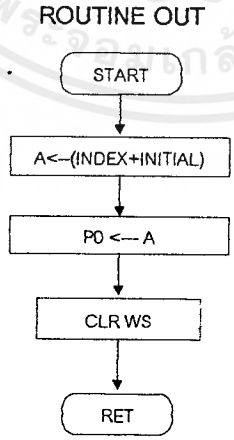
จากหลักการการออกแบบข้างต้นที่กล่าวมาเราสามารถเขียนโฟลชาร์ทในการออกแบบโปรแกรมการทำงานของชิปไมโครคอนโทรลเลอร์ได้ดังต่อไปนี้

โฟลชาร์ทการทำงานของโปรแกรมหลัก การทำงานจะทำการตรวจสอบสัญญาณ RD300 RD301 RD302 RD303 และ WR300 เพื่อทำการรับส่งข้อมูลให้กับ MCS-51



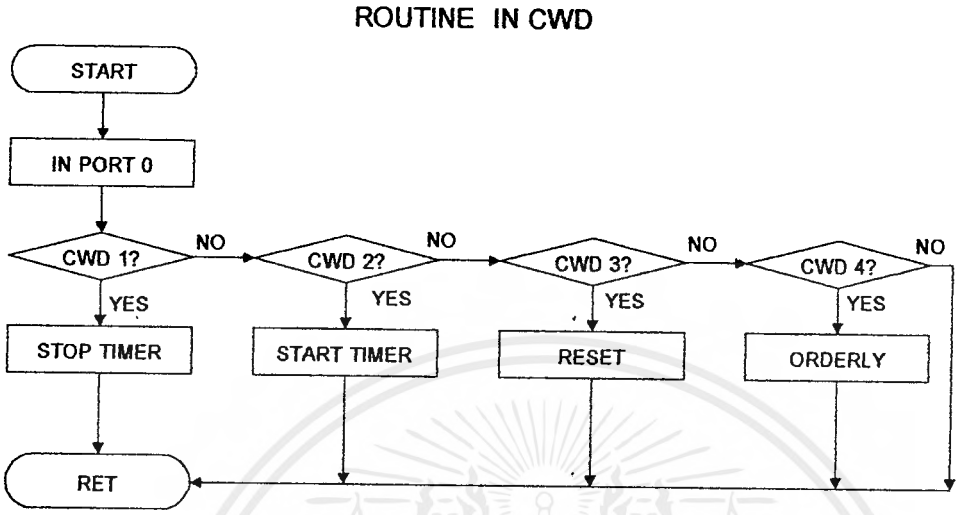
รูป 4.7 โฟลชาร์ทโปรแกรมหลักของชิปไมโครคอนโทรลเลอร์

สำหรับการส่งข้อมูลที่เป็นผลต่างเวลาและตำแหน่งจาก MCS-51 ผ่านทางพอร์ท 0 ให้กับคอมพิวเตอร์ ระบบการทำงานจะเป็นดังโฟลชาร์ท

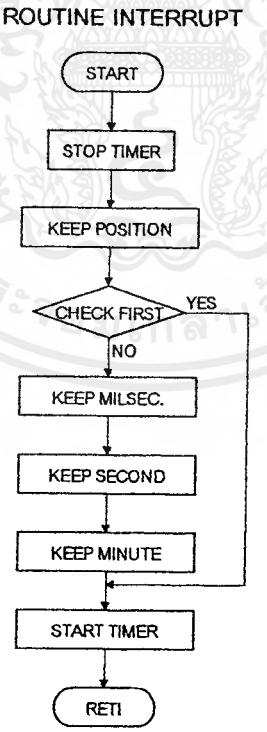


รูป 4.8 โฟลชาร์ทโปรแกรมย่อยส่งผลให้คอมพิวเตอร์

โฟลชาร์ทโปรแกรมย่อยในการรับคำสั่งจากคอมพิวเตอร์ ใช้สำหรับการรับชุดคำสั่งที่มาจากคอมพิวเตอร์เพื่อใช้ควบคุมการทำงานของ MCS-51 โดยส่งผ่านทางพอร์ท 0



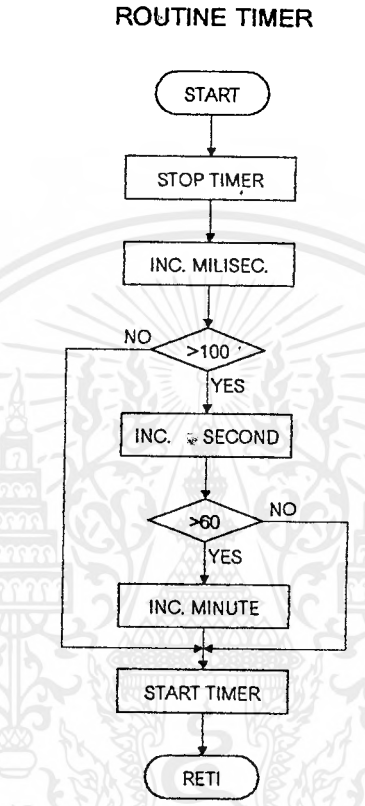
รูป 4.9 โฟลชาร์ทโปรแกรมย่อยรับคำสั่งจากคอมพิวเตอร์



รูป 4.10 โฟลชาร์ทโปรแกรมย่อยอินเตอร์รัปต์

โฟลชาร์ทโปรแกรมย่อยอินเทอร์รัปต์ที่ใช้ในการเก็บข้อมูลที่เป็นผลต่างเวลา และตำแหน่ง เมื่อมีวัตถุตัดผ่านชุดรุปส่งอินฟาเรด ดังรูป 4.10

โฟลชาร์ทโปรแกรมย่อยอินเทอร์รัปต์ไทม์เมอร์ เมื่อ timer register เต็มทุก ๆ 10 มิลลิวินาที จะทำการเพิ่มค่า MILSEC. SECOND และ MINUTE ตามลำดับ



รูป 4.11 โฟลชาร์ทโปรแกรมย่อยอินเตอร์รัปต์ไทม์เมอร์

บทที่ 5

การทดลองและผลการทดลอง

ระบบตรวจจับเวลา 4 ช่องทางนี้ สามารถควบคุมการทำงานของชุดไมโครคอนโทรลเลอร์ผ่านทางพีซี โดยทำการเขียนคำสั่ง ไปที่พอร์ท $300 ในการอ่านค่าเวลาจะอ่านจากพอร์ท $300 $301 $302 และใช้พอร์ท $303 ในการอ่านค่าตำแหน่ง สามารถแบ่งเป็นโปรแกรมย่อยการทำงานออกเป็นดังนี้

- โปรแกรมย่อยอ่านค่าข้อมูลจากพอร์ท

```
function getport(p:word):byte;stdcall;
begin
        asm
                push    edx
                push    eax
                mov     dx,p
                in      al,dx
                mov     @result,al
                pop     eax
                pop     edx
        end;
end;
```

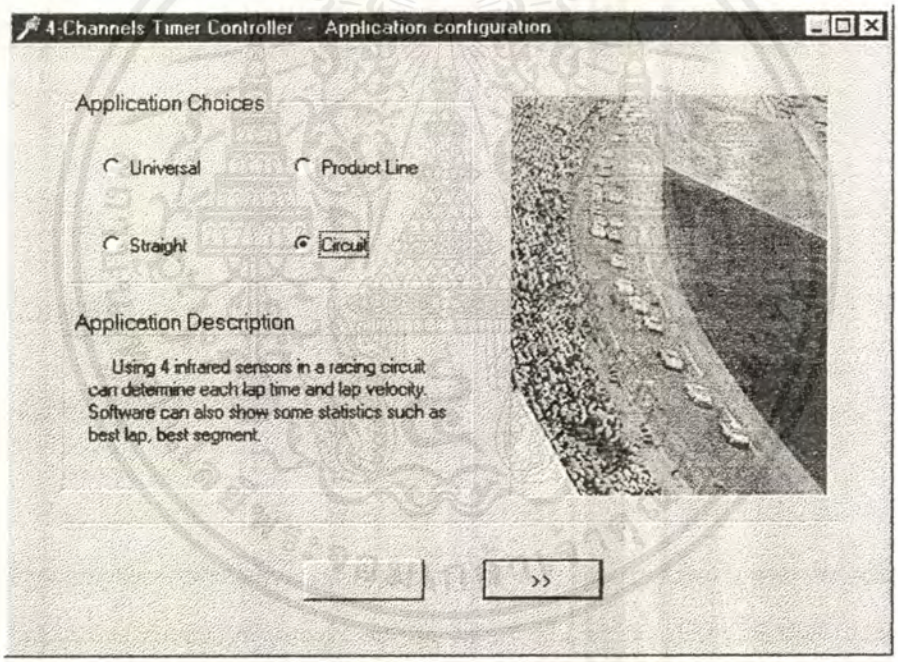
- โปรแกรมย่อยเขียนข้อมูลไปยังพอร์ท

```
Procedure Setport(p:word;b:byte);Stdcall;
begin
        asm
                push    edx
                push    eax
                mov     dx,p
                mov     al,b
```

```
            out     dx,al
            pop     eax
            pop     edx
        end;
    end;
```

- โปรแกรมที่ใช้ในการทดลอง

การเริ่มการทดลองจะเริ่มจากหน้าต่างหลัก เพื่อเลือกโปรแกรมประยุกต์ที่เหมาะสมกับการใช้งาน ซึ่งมีให้เลือก 4 ลักษณะคือ การใช้งานทั่วไป การประยุกต์ใช้ในสายการผลิต การจับเวลาหรือแข่งขันที่มีลักษณะการเคลื่อนที่เป็นเส้นตรง และลักษณะการเคลื่อนที่เป็นรอบ

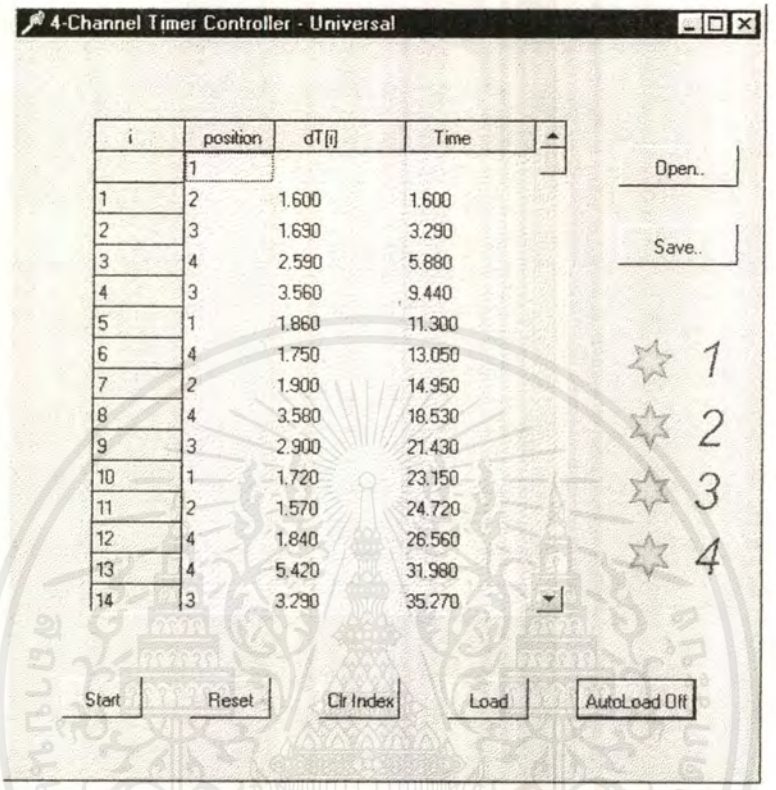


รูป 5.1 หน้าต่างหลักการเลือกใช้โปรแกรมประยุกต์

- โปรแกรมการใช้งานทั่วไป

เป็นโปรแกรมที่สามารถใช้งานในรูปแบบพื้นฐานทั่วไป โดยสามารถอ่านค่าผลต่างของเวลาในการตรวจจับจากชุดรับส่งอินฟราเรด ที่เก็บไว้ในหน่วยความจำสำหรับเก็บข้อมูลภายในไมโครคอนโทรลเลอร์ มาทำการแสดงค่าซึ่งสามารถทำงานได้ทั้งในโหมดของเรียลไทม์ (real time) และนอนเรียลไทม์ (non-real time) โดยในโหมดของเรียลไทม์สามารถแสดงตำแหน่งที่มีการตัดผ่านรังสี

ของชุดรับส่งอินฟราเรดได้ และควบคุมการทำงานพื้นฐานต่าง ๆ ได้คือ เริ่ม-หยุด ไทม์เมอร์ รีเซ็ท ระบบตรวจจับเวลา นอกจากนั้นยังเก็บข้อมูลในรูปแบบของเท็กซ์ไฟล์เพื่อการเรียกดูข้อมูลได้อีก

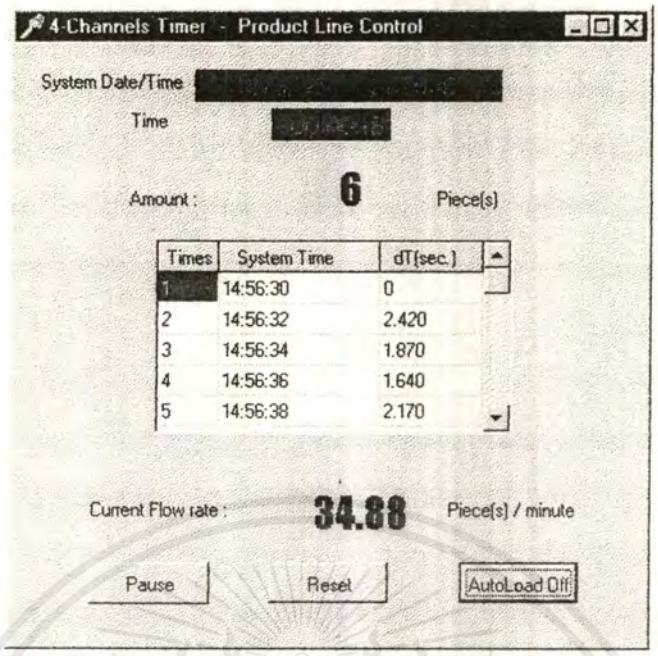


รูป 5.2 หน้าต่างโปรแกรมประยุกต์การใช้งานพื้นฐาน

- โปรแกรมประยุกต์ในสายการผลิต

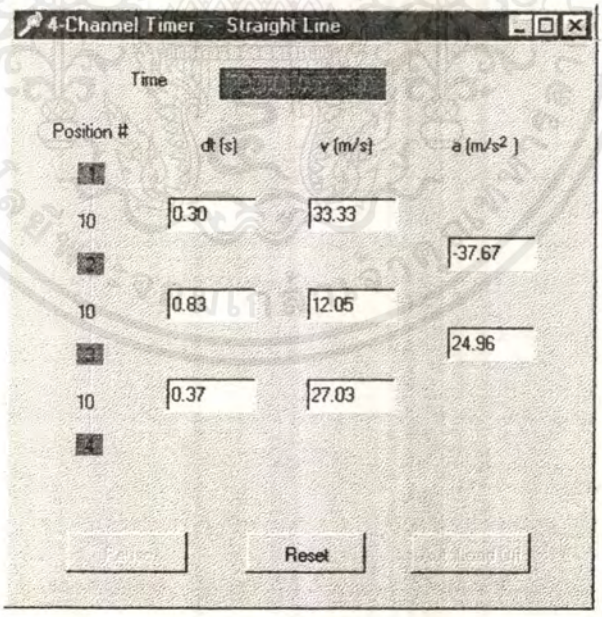
เป็นโปรแกรมที่ใช้นับปริมาณและคำนวณอัตราการไหลของผลิตภัณฑ์ที่ผ่านตัวตรวจจับของระบบโดยสามารถเลือกใช้เฉพาะชุดรับส่งอินฟราเรดที่ต้องการได้ ด้วยการปรับสวิตช์เลือกโปรแกรมอนุญาตให้ตั้งค่าเริ่มต้นของการนับ และค่าสุดท้ายที่ต้องการ โดยจะเตือนเมื่อครบตามค่าที่ตั้งไว้ข้างต้น และสามารถเลือกที่จะรีเซ็ทระบบหรือทำการนับต่อไปได้

ในสายการผลิตจริง อาจพัฒนาโปรแกรมประยุกต์นี้ เพื่อควบคุมฮาร์ดแวร์ภายนอกให้ทำการหยุดสายการผลิตเมื่อครบตามจำนวนที่ต้อง หรือปรับอัตราการไหลให้เร็วสหรือช้าตามต้องการ



รูป 5.3 หน้าต่างโปรแกรมประยุกต์การในสายการผลิต

- โปรแกรมประยุกต์การจับเวลาที่มีลักษณะการเคลื่อนที่เป็นเส้นตรง

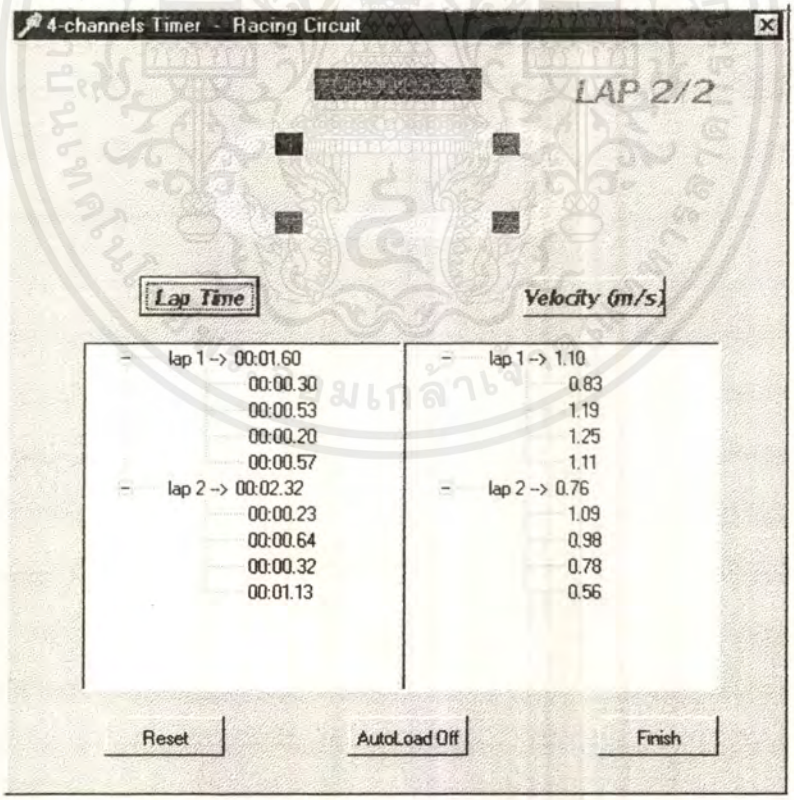


รูป 5.4 หน้าต่างโปรแกรมประยุกต์การจับเวลาที่วัตถุมีลักษณะการเคลื่อนที่เป็นเส้นตรง

โปรแกรมนี้เป็นการจับเวลาที่วัตถุมีลักษณะการเคลื่อนที่เป็นเส้นตรงจากจุดเริ่มต้นไปยังจุดสิ้นสุดโดยผ่านตัวตรวจจับเวลาเป็นช่วง ๆ โปรแกรมอนุญาตให้ตั้งค่าระยะทางที่ชุดตรวจจับเวลาแต่ละชุดอยู่ห่างกัน และจะทำการคำนวณหาความเร็วและความเร่งในแต่ละช่วงออกมาได้ โปรแกรมสามารถควบคุมการทำงานของไมโครคอนโทรลเลอร์โดย มีคำสั่งเริ่ม-หยุดไทม์เมอร์ และรีเซ็ทข้อมูลของไมโครคอนโทรลเลอร์

- โปรแกรมประยุกต์การใช้งานในลักษณะที่เป็นรอบ

  โปรแกรมนี้เป็นการจับเวลาที่วัตถุมีลักษณะการเคลื่อนที่เป็นรอบ ที่จุดเริ่มต้นและจุดสิ้นสุดเป็นจุดเดียวกัน โดยแสดงเวลาในแต่ละช่วงของแต่ละรอบ โปรแกรมอนุญาตให้ตั้งค่าระยะทางที่ชุดตรวจจับเวลาแต่ละชุดอยู่ห่างกัน และจำนวนรอบที่วัตถุจะเคลื่อนที่ จะทำการคำนวณหาความเร็วของในแต่ละช่วงและความเร็วเฉลี่ยในแต่ละรอบ อีกทั้งยังสามารถควบคุมการทำงานของไมโครคอนโทรลเลอร์โดย มีคำสั่งเริ่ม-หยุดไทม์เมอร์ และรีเซ็ทข้อมูลของไมโครคอนโทรลเลอร์



รูป 5.5 หน้าต่างโปรแกรมประยุกต์การใช้งานในลักษณะที่เป็นรอบ

บทที่ 6
สรุปและวิจารณ์

โครงงานระบบตรวจจับเวลา 4 ช่องทางโดยไมโครโปรเซสเซอร์นี้ได้มีการออกแบบและพัฒนาส่วนของวงจรอินเทอร์เฟสกับคอมพิวเตอร์ ให้สามารถส่งผลจากการวัดให้กับคอมพิวเตอร์หรือรับคำสั่งเข้ามาได้

จากผลการทดลอง จะเห็นได้ว่าค่าเวลาที่วัดได้มีค่าถูกต้องตามการทดลองที่วัดจากนาฬิกาจับเวลาและสามารถปฏิบัติตามคำสั่งได้เช่น การหยุดไทม์เมอร์ การเริ่มไทม์เมอร์ จากโปรแกรมประยุกต์รวมไปถึงการเก็บ การเคลียร์ค่าข้อมูลเก่า และเรียกข้อมูลเก่าที่เก็บไว้ ได้ถูกต้อง และจากการประยุกต์การใช้งานในการทดลองตอนสุดท้ายเราสามารถที่จะทดสอบจากอุปกรณ์ที่ใช้ทดสอบในรูปของรางโดยการตัดผ่านที่ความเร็วต่าง ๆ จะได้ความเร็วตามที่เราตัดผ่านอย่างถูกต้อง รวมไปถึงความเร่งที่คำนวณได้ก็ถูกต้องตามหลักการ กล่าวคือถ้าความเร็วที่ใช้เพิ่มขึ้นค่าความเร่งจะเป็นบวกในทางกลับกันถ้าความเร็วที่ใช้ตัดลดลงจะได้ความเร่งที่เป็นลบ นั่นคือสามารถทำระบบตรวจจับเวลา 4 ช่องทางโดยไมโครโปรเซสเซอร์ผ่านทางสล็อตพีซีได้ตามจุดประสงค์

ปัญหาและอุปสรรคที่พบในการทำงานได้แก่

- สามารถหาข้อมูลและหนังสือที่เกี่ยวกับระบบ I/O ของคอมพิวเตอร์ได้น้อย
- การทดลองขณะออกแบบและพัฒนาวงจรทำได้ค่อนข้างยากเนื่องจากการสังเกตผลสัญญาณจากคอมพิวเตอร์ได้ลำบาก
- ทำการตรวจสอบแก้ไขข้อผิดพลาดของโปรแกรมแอสแซมบลี (assembly) ได้ยาก เนื่องจากสามารถตรวจสอบผลได้จากข้อมูลที่คอมพิวเตอร์ได้รับเท่านั้น
- หาเครื่องโปรแกรมหน่วยจำสำหรับเก็บโปรแกรมของชิปไมโครคอนโทรลเลอร์ได้ยาก เนื่องจากทางภาคไม่มีให้นักศึกษาใช้ และไม่มีงบประมาณให้นักศึกษามากพอ
- มีปัญหาเรื่องไซเคิล (cycle) ของการอ่าน/เขียนข้อมูลของพอร์ต I/O ที่ต่างกันในคอมพิวเตอร์แต่ละรุ่น

โครงงานนี้สามารถปฏิบัติงานได้ตามเป้าหมาย คือ สามารถบันทึกเวลา ส่งข้อมูลและติดต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ได้ ในการประยุกต์ใช้โครงงานในรูปแบบต่างๆ เป็นไปตามเป้าหมายคือ สามารถประยุกต์ใช้ได้ในหลายรูปแบบตามลักษณะโปรแกรมที่ใช้ ตั้งแต่การประยุกต์ใช้ในแบบทั่วไป การใช้ในสายการผลิต การใช้ในสนามแข่งขันและทดสอบต่างๆ โดยสามารถเก็บและเรียกข้อมูลเก่าที่เก็บไว้ได้ถูกต้อง ซึ่งการประยุกต์ใช้งานโดยระบบนี้นั้นสามารถเห็น

ถึงข้อดีที่ช่วยลดขีดจำกัดความสามารถของคนที่เห็นได้อย่างชัดเจนคือจากการใช้ตัวตรวจจับอินฟราเรด (infrared sensor) ในการจับเวลาคือสามารถตรวจจับช่วงเวลาน้อย ๆ ได้ทันไม่ขึ้นกับความไวของสายตา หรือความไวของผู้บันทึกจึงสามารถลดความผิดพลาดลงได้เป็นอย่างมาก จึงเป็นระบบที่มีความเที่ยงตรงค่อนข้างสูง

ภาคผนวก ก.
โปรแกรมภาษาแอสเซมบลีที่ใช้กับไมโครคอนโทรลเลอร์

โปรแกรมภาษาแอสเซมบลีที่ใช้กับไมโครคอนโทรลเลอร์

```
;               ###################################
;               # 4 CHANNEL TIMER (TOKE11.ASM)  #
;               ###################################

;BYTE
                INDEX       EQU   01H
                N           EQU   02H
                INDEX1      EQU   03H
                N1          EQU   04H
                MILSEC      EQU   21H
                SECOND      EQU   40H
                MINUTE      EQU   5FH
                POSIT       EQU   10H
                POSIT1      EQU   11H
;BIT
                FIRST       EQU   00H
                SEQ_EN      EQU   02H
                N_IND       EQU   01H
                WS          EQU   P2.3
ORG 0000H
                JMP   OVER
ORG 0003H                                   ;EXT INT 0 ROUTINE
                JMP   KEEP
ORG 000BH                                   ;TIMER INT 0 ROUTINE
                JMP TIMER
OVER:           CALL  CLRWS
                CLR   FIRST
```

```
        CLR   SEQ_EN
        SETB  WS
        CLR   TR0
        SETB  EA                ;ENABLE INTERRUPT
        SETB  ET0               ;ENABLE TIMER INT
        SETB  EX0               ;ENABLE EXT INT
        SETB  IT0               ;ENABLE EDGE INT CHECK
        MOV   N,#00H
        MOV   INDEX,#00H
        MOV   POSIT,#06H
        MOV   POSIT1,#01H
        MOV   R5,#00H           ;MILLISEC:=0
        MOV   R6,#00H           ;SECOND:=0
        MOV   R7,#00H           ;MINUTE:=0
        CLR   P1.4              ;ON LED
COMP:
RD303:  JB    P1.2,WR300        ;CHECK RD303 SIGNAL
        MOV   A,POSIT           ;OUT POSITION
        MOV   P0,A
        CALL  CLRWS
        MOV   POSIT,#06H
WR300:  JB    P1.3,READ
        JMP   INCWD
READ:
        MOV   A,INDEX1
        CJNE  A,N1,RD300        ;IF INDEX<>N
        SETB  N_IND
RD300:  JB    P1.0,RD301        ;CHECK RD300 SIGNAL
        JMP   OUTMS
RD301:  JB    P1.1,RD302        ;CHECK RD301 SIGNAL
        JMP   OUTSEC
```

```
RD302:    JB    P1.2,WR300          ;CHECK RD302 SIGNAL
          JMP   OUTMIN
WR300:    JB    P1.3,COMP           ;CHECK WR300 SIGNAL
          JMP   INCWD
OUTMS:    JB    N_IND,MOVFE
          MOV   A,INDEX             ;DISPLAY MILSEC TO COMPUTER
          ADD   A,#MILSEC
          MOV   R0,A
          MOV   A,@R0
          MOV   P0,A
          JMP   CLRWAIT
OUTSEC:   JB    N_IND,MOVFE
          MOV   A,INDEX             ;DISPLAY SECOND TO COMPUTER
          ADD   A,#SECOND
          MOV   R0,A
          MOV   A,@R0
          MOV   P0,A
          JMP   CLRWAIT
OUTMIN:   JB    M_IND.MOVFE
          MOV   A,INDEX     ;DISPLAY MINUTE TO COMPUTER
          ADD   A,#MINUTE
          MOV   R0,A
          MOV   A,@R0
          MOV   P0,A
          INC   INDEX
          INC   INDEX1
          MOV   A,INDEX
          CJNE  A,#31,CLR
          MOV   INDEX,#00H
          JMP   CLRWAIT
```

```
INCWD:                                  ;WRITE INSTRUCTION TO MCS
STOP:     JNB   P0.0,START              ;CHECK STOP TIMER SIGNAL
          CLR   TR0                     ;STOP TIMER
          JMP   CLRWAIT
START:    JNB   P0.1,RST                ;CHECK START TIMER SIGNAL
          SETB  FIRST
          SETB  TR0                     ;START TIMER
          JMP   CLRWAIT
RST:      JNB   P0.2,CLRWAIT            ;CHECK RESET SIGNAL
          JMP   OVER
CLRIN:    JNB   P0.3,SEQ
          MOV   INDEX,#00H
          MOV   INDEX1,#00H
SEQ:      JNB   P0.4,CLR
          SETB  SEQ_EN
CLRWAIT:  CALL  CLRWS                   ;CLEAR WAIT STATE
          JMP   COMP

;         ###################################
;         #  EXTERNAL INTERUPT  ROUTINE  #
;         ###################################

KEEP:                                   ;KEEP TIME DATA TO RAM
          PUSH  ACC
          CLR   TR0                     ;STOP TIMER
POS1:     JNB   P2.4,POS2               ;FIRST POSITION
          MOV   POSIT,#01H
POS2:     JNB   P2.5,POS3               ;SECOND POSITION
          MOV   POSIT,#02H
POS3:     JNB   P2.6,POS4               ;THIRD POSITION
          MOV   POSIT,#03H
```

```
POS4:       JNB   P2.7,CHECKSEQ          ;FORTH POSITION
            MOV   POSIT,#04H
CHECKSEQ:   JNB   SEQ_EN,CHECKFR         ;SET ORDERLY
            JB    FIRST,SEQUENCE
            MOV   A,POSIT
            CJNE  A,#01H,POPA
            JMP   CHECKFR
SEQUENCE:   INC   POSIT1
            MOV   A,POSIT1
            CJNE  A,#05H,COMPSEQ
            MOV   POSIT1,#01H
COMPSEQ:    MOV   A,POSIT1
            CJNE  A,POSIT,DECPOS1
            JMP   KEEP1
DECPOS1:    DEC   POSIT1
            MOV   A,POSIT1
            CJNE  A,#00H,LOAD
            MOV   POSIT1,#04H
            JMP   LOAD
CHECKFR:    JB    FIRST,KEEP1            ;CHECK FIRST INT
            SETB  FIRST
            JMP   TIME
KEEP1:      MOV   A,N
            CJNE  A,#31,KEEPMS
            MOV   N,#00H
KEEPMS:     MOV   A,N                    ;KEEP MILSEC TO RAM
            ADD   A,#MILSEC
            MOV   R0,A
            MOV   A,R5
            MOV   @R0,A
```

```
KEEPSEC:    MOV   A,N                    ;KEEP SECOND TO RAM
            ADD   A,#SECOND
            MOV   R0,A
            MOV   A,R6
            MOV   @R0,A
KEEPMIN:    MOV   A,N                    ;KEEP MINUTE TO RAM
            ADD   A,#MINUTE
            MOV   R0,A
            MOV   A,R7
            MOV   @R0,A
            INC   N
            INC   N1
            MOV   R5,#00H
            MOV   R6,#00H
            MOV   R7,#00H
LOAD:       CALL  RELOAD                 ;SET INITIAL TO TIMER
            SETB  TR0                    ;START TIMER
POPA:       POP   ACC
            RETI

;           ################################
;           #   TIMER INTERUPT ROUTINE   #
;           ################################

TIMER:      PUSH  ACC                    ;INC VALUE EVERY 10 MS
            CLR   TR0
            INC   R5                     ;1 CYCLE
            MOV   A,R5                   ;1 CYCLE
            CJNE  A,#100,TIME            ;2 CYCLE
            MOV   R5,#00H
```

```
        JB      P1.4,CLRB               ;ON/OFF LED EVERY 1 SEC
        SETB    P1.4
        JMP     CONT
CLRB:   CLR     P1.4
CONT:   INC     R6                      ;INC SECOND VALUE
        MOV     A,R6
        CJNE    A,#60,TIME
        MOV     R6,#00H
        INC     R7                      ;INC MINUTE VALUE
TIME:   CALL    RELOAD                  ;2 CYCLE
        SETB    TR0                     ;2 CYCLE
        POP     ACC                     ;1 CYCLE
        RETI                            ;2 CYCLE

;       ================================
;       #   ROUNTINE RELOAD   #
;       ================================
;       RELOAD TIMER REGISTER

RELOAD: MOV     DPTR,#(-9216+21)        ;2 CYCLE
        MOV     TL0,DPL                 ;2 CYCLE
        MOV     TH0,DPH                 ;2 CYCLE
        MOV     TMOD,#01H               ;2 CYCLE
        RET                             ;2 CYCLE
```

```
;               ==========
;               #   ROUNTINE CLRWS   #
;               ==========
;                 CLEAR WAIT STATE

CLRWS:          CLR    WS                      ;CLEAR WAIT STATE
                SETB   WS
                RET
        END


;                       FROM TOKE11.ASM
```

ภาคผนวก ข
โปรแกรมควบคุมระบบ

โปรแกรมควบคุมระบบ

- โปรแกรมประยุกต์ใช้งานทั่วไป

```
unit simple;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ComCtrls, Buttons, ExtCtrls, Grids;

type
  TForm2 = class(TForm)
    ListView1: TListView;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    Timer1: TTimer;
    Timer2: TTimer;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    BitBtn3: TBitBtn;
    SaveDialog1: TSaveDialog;
    StringGrid1: TStringGrid;
    BitBtn4: TBitBtn;
    OpenDialog1: TOpenDialog;
    Image1: TImage;
    Timer3: TTimer;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    BitBtn5: TBitBtn;
    Image6: TImage;
    Image7: TImage;
    Image8: TImage;
    Image9: TImage;
```

```
    BitBtn6: TBitBtn;
    BitBtn7: TBitBtn;
    BitBtn8: TBitBtn;
    Image5: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Timer1Timer(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure ListView1ColumnClick(Sender: TObject; Column: TListColumn);
    procedure ListView1DblClick(Sender: TObject);
    procedure ListView1Change(Sender: TObject; Item: TListItem;
      Change: TItemChange);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure Timer3Timer(Sender: TObject);
    procedure BitBtn5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    f:Text;
  end;
var
  Form2: TForm2;
  NewColumn:TListColumn;
  ListItem:TListItem;
  ms,s,min:integer;
  i:longint;
  dt:array[0..255] of real;
  time:real;
  started,AutoL:Boolean;
  st:string;
  preposit,posit,tempposit: Byte;
  indexpic:byte;
  tickstar:byte;
```

```
  star :array[1..4] of TImage;

implementation

{$R *.DFM}
uses   startpage,port;

procedure TForm2.FormCreate(Sender: TObject);
begin
with listView1 do
begin
ViewStyle:=vsReport;
parent:=self;
NewColumn:=Columns.Add;
NewColumn.Caption:='Lap';
NewColumn:=Columns.Add;
NewColumn.Caption:='dT';
NewColumn:=Columns.Add;
NewColumn.Caption:='Time';
listview1.Columns[0].Alignment:=taCenter;
listview1.Columns[1].Alignment:=taRightJustify;
listview1.Columns[2].Alignment:=taRightJustify;
listview1.Columns[0].Width:=30;
listview1.Columns[1].Width:=100;
listview1.Columns[2].Width:=100;
end;
dt[0]:=0;
time:=dt[0];
i:=0;
Setport($300,8);
started:=false;
AutoL:=False;
Timer1.Enabled:=False;
Timer2.Enabled:=True;
StringGrid1.Cols[0].text:='     i';
StringGrid1.Cols[1].text:='   position';
StringGrid1.Cols[2].text:='    dT[i]';
StringGrid1.Cols[3].text:='    Time';
indexpic:=1;
star[1]:=image1;
```

```
star[2]:=image2;
star[3]:=image3;
star[4]:=image4;
tickstar:=0;
image5.Align:=alClient;
posit:=0;
preposit:=0;
end;

procedure TForm2.Button3Click(Sender: TObject);
var  temp:integer;
begin
Setport($300,4);
ListView1.Items.Clear;
for temp:=1 to StringGrid1.RowCount do
   StringGrid1.Rows[temp].text:='';
StringGrid1.ColCount:=4;
StringGrid1.RowCount:=2;
i:=0;
dt[0]:=0;
time:=dt[0];
AutoL:=False;
Timer1.Enabled:=False;
BitBtn2.Caption:='AutoLoad On';
Timer2.Enabled:=True;
BitBtn1.Hint:='Start';
started:=False;
posit:=0;
BitBtn2.Enabled:=True;
BitBtn8.Enabled:=True;
BitBtn7.Enabled:=True;
BitBtn1.Enabled:=True;
end;

procedure TForm2.Button4Click(Sender: TObject);
var   st:string;
begin
tempposit:=Getport($303);
if tempposit in[1..4] then
 begin
```

```
      posit :=tempposit;
      timer3.Enabled:=true;  //New-Pass:star start to rotate
   end;
ms := Getport($300) ;
ms := Getport($300) ;
if ms <> 254 then
   begin
   s  := Getport($301);
   min:= Getport($302);
   i:=i+1;
   dt[i]:=min*60 + s + ms/100;
   if dt[i]<>0 then
      begin
      time:=time+dt[i]  ;
         with StringGrid1 do
            begin
            Rows[i].text:=inttostr(i);
            Cells[1,i]:=inttostr(posit);
            str(dt[i]:0:3,st);
            Cells[2,i]:=st;
            str(time:0:3,st);
            Cells[3,i]:=st;
            RowCount:=StringGrid1.RowCount+1;
            end;
      end
      else i:=i-1;
   end;
end;

procedure TForm2.Button5Click(Sender: TObject);
var temp:integer;
begin
Setport($300,8);
i:=0;
time:=0;
for temp:=1 to StringGrid1.RowCount do
    StringGrid1.Rows[temp].text:='';
StringGrid1.ColCount:=4;
StringGrid1.RowCount:=2;
end;
```

```
procedure TForm2.BitBtn1Click(Sender: TObject);
begin
if started then
  begin
  Setport($300,1);{Pause}
  BitBtn1.Caption:='Start';
  BitBtn1.Hint:='Start';
  started:=False;
  end
  else
  begin
  Setport($300,2);{Start}
  BitBtn1.Caption:='Pause';
  BitBtn1.Hint:='Pause';
  started:=true;
  end;
end;

procedure TForm2.BitBtn2Click(Sender: TObject);
begin
if AutoL= False then
  begin
  AutoL:=True;
  BitBtn2.Caption:='AutoLoad Off';
  StringGrid1.SetFocus;
  Timer1.Enabled:=True;
  Button4.Enabled:=False;
  Timer2.Enabled:=False;
  tempposit:=Getport($303);
  ms := Getport($300);
  ms := Getport($300);
  while  (ms<>254)and(i<20) do
    begin
    s  := Getport($301);
    min:= Getport($302);
    i:=i+1;
    dt[i]:=min*60 + s + ms/100;
    if dt[i]<>0 then
      begin
```

```
        time:=time+dt[i] ;
      with StringGrid1 do
          begin
          Rows[i].text:=inttostr(i);
          Cells[1,i]:=inttostr(posit);
          str(dt[i]:0:3,st);
          Cells[2,i]:=st;
          str(time:0:3,st);
          Cells[3,i]:=st;
          RowCount:=StringGrid1.RowCount+1;
          end;
        end
        else i:=i-1;
      ms := Getport($300);
      end; {while loop}
    end {if}
    else
    begin
    AutoL:=False;
    Timer1.Enabled:=False;
    BitBtn2.Caption:='AutoLoad On';
    Timer2.Enabled:=True;
    end;
end;

procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
form1.Close;
end;

procedure TForm2.Timer1Timer(Sender: TObject);
begin
    tempposit:=Getport($303);
    if tempposit in [1..4] then
    begin
        preposit:=posit;
        posit :=tempposit;
        timer3.Enabled:=true;  //New-Pass:star start to rotate
        if preposit=0  then StringGrid1.Cells[1,1]:=IntToStr(posit);
    end;
```

```
    ms := Getport($300);
    if ms<>254 then
    begin
       s  := Getport($301);
       min:= Getport($302);
       i:=i+1;
       dt[i]:=min*60 + s + ms/100;
       if dt[i]<>0 then
        begin
        time:=time+dt[i]  ;
        with StringGrid1 do
          begin
          Rows[i+1].text:=inttostr(i);

          Cells[1,i+1]:=inttostr(posit);

          str(dt[i]:0:3,st);
          Cells[2,i+1]:=st;

          str(time:0:3,st);
          Cells[3,i+1]:=st;
          RowCount:=StringGrid1.RowCount+1;
          end;
        end
       else i:=i-1;
   end; {if}
end;

procedure TForm2.Timer2Timer(Sender: TObject);
begin
if Getport($300)<>254 then   Button4.Enabled:=True
  else Button4.Enabled:=False;
end;

procedure TForm2.ListView1ColumnClick(Sender: TObject;
 Column: TListColumn);
begin
if Column=ListView1.Columns[0] then
  begin
  Label1.Caption:='Col0';
```

```
  with listview1 do
     begin
     SortType:=stData;
     end;
  end;
if Column=ListView1.Columns[1] then
  begin
  Label1.Caption:='Col1';
  with listview1 do
     begin
     SortType:=stText;
     end;
  end;
if Column=ListView1.Columns[2] then
  begin
  with listview1 do
     begin
     Label1.Caption:=listitem.SubItems.Text ;
     label2.caption:=
     Items.Item[listview1.Items.Count-2].SubItems.Strings[0];
     Items.Item[listview1.Items.Count-2].SubItems.SaveToFile('c:\windows\desktop\test.txt');
     SortType:=stNone;
     end;
  end;
end;

procedure TForm2.ListView1DblClick(Sender: TObject);
begin
with listview1 do
   begin
   Items.Item[listview1.Items.Count-2].Caption:='test';
   Items.Item[listview1.Items.Count-1].Focused:=true;
   end;
end;
procedure TForm2.ListView1Change(Sender: TObject; Item: TListItem;
 Change: TItemChange);
begin
if ListView1.VisibleRowCount<ListView1.Items.Count then
  ListView1.Scroll(0,100);
end;
```

```
procedure TForm2.BitBtn3Click(Sender: TObject);
var i:integer;
begin
if SaveDialog1.Execute then
  begin
  assignfile(f,SaveDialog1.Filename);
  Rewrite(f);
  for i:=0 to StringGrid1.RowCount-2 do
    write(f,StringGrid1.rows[i].text);
  closefile(f);
  end;
end;
procedure TForm2.BitBtn4Click(Sender: TObject);
var  v:string;temp:integer;
begin
dt[0]:=0;
time:=dt[0];
i:=0;
if OpenDialog1.Execute then
  begin
  assignfile(f,OpenDialog1.Filename);
  Reset(f);
  for temp:=1 to StringGrid1.RowCount do
    StringGrid1.Rows[temp].text:='';
  StringGrid1.ColCount:=4;
  StringGrid1.RowCount:=2;
  readln(f,v);            //no.
  Stringgrid1.Rows[i].Add(v);
  readln(f,v);            //posit
  Stringgrid1.Rows[i].Add(v);
  readln(f,v);            //dT
  Stringgrid1.Rows[i].Add(v);
  readln(f,v);            //Time
  Stringgrid1.Rows[i].Add(v);
  while not eof(f) do
    begin
    i:=i+1;
    readln(f,v);          //no.
    Stringgrid1.Cells[0,i]:=v;
    readln(f,v);          //posit
```

```
      Stringgrid1.Cells[1,i]:=v;
      readln(f,v);              //dT
      try
       dT[i]:=strtofloat(v);
       Time:=Time+dT[i];
      except  end;
      Stringgrid1.Cells[2,i]:=v;
      readln(f,v);              //Time
      Stringgrid1.Cells[3,i]:=v;
      StringGrid1.RowCount:=StringGrid1.RowCount+1;
     end;
  closefile(f);
  BitBtn2.Enabled:=False;
  BitBtn8.Enabled:=False;
  BitBtn7.Enabled:=False;
  BitBtn1.Enabled:=False;
  end;
end;

procedure TForm2.Timer3Timer(Sender: TObject);
begin
if posit in[1..4]then
begin
case indexpic of
1:star[posit].Picture.LoadFromFile('star1.bmp');
2:star[posit].Picture.LoadFromFile('star2.bmp');
3:star[posit].Picture.LoadFromFile('star3.bmp');
4:star[posit].Picture.LoadFromFile('star4.bmp');
end;
inc(indexpic);
if indexpic>4 then indexpic:=1;
inc (tickstar);
if tickstar=13 then
  begin
  timer3.Enabled:=False;
  tickstar:=0;
  indexpic:=1;
  end;
end; //if posit
end;
```

```
procedure TForm2.BitBtn5Click(Sender: TObject);
begin
timer3.Enabled:=true;
end;
end.
```

- โปรแกรมประยุกต์สายการผลิต

```
unit product;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls, Grids, ComCtrls;

type
  TForm4 = class(TForm)
    Label1: TLabel;
    ListBox1: TListBox;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    Timer1: TTimer;
    Label6: TLabel;
    Label7: TLabel;
    Timer2: TTimer;
    Label9: TLabel;
    StringGrid1: TStringGrid;
    Label10: TLabel;
    Label11: TLabel;
    Timer3: TTimer;
    Label8: TLabel;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
```

```
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure Timer3Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form4: TForm4;
  ms,s,min:integer;
  i:longint;
  dt:array[0..255] of real;
  time:real;
  started,AutoL:Boolean;
  st:string;
  tickcounter:integer;
  totaltime:real;
  hh,mm,ss:integer;
  hhs,mms,sss:string[2];
  FlowRate:real;
  tempposit,preposit,posit:byte;
  addfirst:boolean;
  init: integer;
  chkfull:boolean;
implementation

{$R *.DFM}
uses startpage,port,errordlg,fullform;

procedure TForm4.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form1.Close;
end;

procedure TForm4.Button1Click(Sender: TObject);
```

```
begin
ListBox1.Items.Add('test');
end;

procedure TForm4.BitBtn1Click(Sender: TObject);
begin
if started then
  begin
  Setport($300,1);{Pause}
  BitBtn1.Caption:='Start';
  started:=False;
  Timer2.Enabled:=False;
  Label7.Caption:='Time(pause)';
  end
  else
  begin
  Setport($300,2);{Start}
  BitBtn1.Caption:='Pause';
  started:=true;
  Timer2.Enabled:=True;
  Label7.Caption:='Time(running)'
  end;
end;

procedure TForm4.BitBtn2Click(Sender: TObject);
var temp:integer;
begin
init:=0;
addfirst:=False;
Setport($300,4);
ListBox1.Items.Clear;
for temp:=1 to StringGrid1.RowCount do
   StringGrid1.Rows[temp].text:='';
StringGrid1.RowCount:=2;
StringGrid1.ColCount:=3;
i:=0;
AutoL:=False;
dt[0]:=0;
time:=dt[0];
Timer1.Enabled:=False;
```

```
Timer2.Enabled:=False;
BitBtn3.Caption:='AutoLoad On';
BitBtn1.Caption:='Start';
started:=False;
Label7.Caption:='Time';
Label2.Caption:='0';
Label8.Caption:='  00:00.00';
Label3.Caption:='0';
Tickcounter:=0;
tempposit:=0;
preposit:=0;
posit:=0;
end;

procedure TForm4.BitBtn3Click(Sender: TObject);
var tem:byte;
begin
if AutoL= False then
  begin
  AutoL:=True;
  BitBtn3.Caption:='AutoLoad Off';
  tem:=Getport($303);
  ms := Getport($300);
  ms := Getport($300);
  if (ms=254) and (tem in [1..4]) then
  begin
   addfirst:=true;
   Timer2.Enabled:=True;
   Bitbtn1.Caption:='Pause';
   Label7.Caption:='Time(running)';
   Started:=true;
    StringGrid1.Cells[0,1]:=inttostr(i+1+init);
    StringGrid1.Cells[1,1]:='-:-:-';
    StringGrid1.Cells[2,1]:='0';
  end;
  while  (ms<>254) do
     begin
     s  := Getport($301);
     min:= Getport($302);
     i:=i+1;
```

```
      dt[i]:=min*60 + s + ms/100;
      if dt[i]<>0 then
        begin
        time:=time+dt[i] ;
        str(dt[i]:0:3,st);
        ListBox1.Items.Add(st);
        timer2.Enabled:=True;
        Label7.Caption:='Time(running)';
        BitBtn1.Caption:='Pause';
        started:=true;
        with StringGrid1 do
            begin
            if addfirst=false then
            begin
            addfirst:=true;
            Cells[0,1]:=inttostr(1+init);
            Cells[1,1]:='-:-:-';
            Cells[2,1]:='0';
            end;
            Cells[0,i+1]:=inttostr(i+1+init);
            Cells[1,i+1]:='-:-:-';
            str(dt[i]:0:3,st);
            Cells[2,i+1]:=st;
            RowCount:=StringGrid1.RowCount+1;
            end;
        end
        else i:=i-1;
      ms := Getport($300);
      end; {while loop}
    Timer1.Enabled:=True;
    end {if}
    else
    begin
    AutoL:=False;
    Timer1.Enabled:=False;
    BitBtn3.Caption:='AutoLoad On';
    end;
end;

procedure TForm4.FormCreate(Sender: TObject);
```

```
begin
chkfull:=Form1.CheckBox3.Checked ;
addfirst:=False;
timer3.Enabled:=True;
dt[0]:=0;
time:=dt[0];
if Form1.CheckBox2.Checked then
begin
init:=strtoint(Form1.Edit4.text);
setport($300,4);
end
  else init:=0;
tickcounter:=0;
Setport($300,8);
started:=false;
AutoL:=False;
Timer1.Enabled:=False;
Timer2.Enabled:=False;
Label2.Caption:=inttostr(init);
tempposit:=0;
preposit:=0;
posit:=0;
StringGrid1.Cells[0,0]:=' Times';
StringGrid1.Cells[1,0]:='  System Time';
StringGrid1.Cells[2,0]:='  dT(sec.)';
end;

procedure TForm4.Timer1Timer(Sender: TObject);
var temp:integer;
begin
    if addfirst=false then
    begin
     temp:=Getport($303);
     if temp in [1..4] then
     begin
     addfirst:=true;
     StringGrid1.Cells[0,1]:=inttostr(1+init);
     StringGrid1.Cells[1,1]:=timetostr(now);
     StringGrid1.Cells[2,1]:='0';
     Timer2.Enabled:=True;
```

```
  BitBtn1.Caption:='Pause' ;
  Started:=True;
  end;
 end;
ms := Getport($300);
ms := Getport($300);
 if ms<>254 then
 begin
    s  := Getport($301);
    min:= Getport($302);
    i:=i+1;
    dt[i]:=min*60 + s + ms/100;
    if dt[i]<>0 then
     begin
     time:=time+dt[i] ;
      str(dt[i]:0:3,st);
      ListBox1.Items.Add(st);
      with StringGrid1 do
       begin
       Cells[0,i+1]:=inttostr(i+1+init);//(i+1);
       Cells[1,i+1]:=timetostr(now);
       str(dt[i]:0:3,st);
       Cells[2,i+1]:=st;
       RowCount:=StringGrid1.RowCount+1;
       end;
     end
   else i:=i-1;
end; {if}
if (i=0)and(addfirst=false)then label2.Caption:=inttostr(i+init) else
Label2.Caption:=inttostr(i+init+1);          //display amount
if chkfull  then
  if i+1+init=strtoint(Form1.Edit5.Text) then
  begin
  Timer1.Enabled:=False;
  Timer2.Enabled:=False;
  Timer3.Enabled:=False;
  temp:=Messagebox(0,'Amount is reached to limit. Reset?',' Alert',1);
  case temp of
  1: BitBtn2Click(sender);
  2: begin
```

```
        Timer1.Enabled:=True;
        Timer2.Enabled:=True;
        Timer3.Enabled:=True;
        chkfull:=false;
        end;
      end;{case}
      end;{if i=}
end;

procedure TForm4.Timer2Timer(Sender: TObject);
begin
tickcounter:=tickcounter+1;
totaltime:=tickcounter/8.9;
hh:=round(totaltime) div (60*60);
mm:=(round(totaltime) mod (60*60)) div (60);
ss:=round(totaltime)-(60*60*hh+60*mm) ;
str(hh,hhs);
str(mm,mms);
str(ss,sss);
if length(hhs)<2 then hhs:='0'+hhs;
if length(mms)<2 then mms:='0'+mms;
if length(sss)<2 then sss:='0'+sss;
Label8.Caption:='  '+hhs+':'+mms+'.'+sss;
 try
 FlowRate:=60/dt[i];
 except
 Flowrate:=0;
 end;
Label3.Caption:=Floattostrf(FlowRate,ffFixed,90000,2);
end;

procedure TForm4.Timer3Timer(Sender: TObject);
begin
Label10.Caption:=datetostr(now)+'     '+timetostr(now);
end;

end.
```

- โปรแกรมประยุกต์การผ่านที่เป็นเส้นตรง

```
unit straight;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, Buttons, ExtCtrls;

type
 TForm5 = class(TForm)
  BitBtn1: TBitBtn;
  BitBtn2: TBitBtn;
  BitBtn3: TBitBtn;
  Timer1: TTimer;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
  Label5: TLabel;
  Edit1: TEdit;
  Edit2: TEdit;
  Edit3: TEdit;
  Edit4: TEdit;
  Edit5: TEdit;
  Edit6: TEdit;
  Edit7: TEdit;
  Edit8: TEdit;
  Label6: TLabel;
  Label7: TLabel;
  Label8: TLabel;
  Label9: TLabel;
  Label10: TLabel;
  Label11: TLabel;
  Timer2: TTimer;
  Bevel1: TBevel;
  Label12: TLabel;
  Label13: TLabel;
  Label14: TLabel;
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

```
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form5: TForm5;
  ms,s,min:integer;
  i:longint;
  dt:array[0..255] of real;

  started,AutoL:Boolean;
  st:string;
  totaltime,ss:real;
  tickcounter:integer;
  time:real;
  time_D:real;  //Display Time
  hh,mm:integer;
  hhs,mms,sss,mss:string[5];
  s1,s2,s3,v1,v2,v3 :real;
  a2,a3:real;
  posit,tempposit :byte;

implementation
{$R *.DFM}
uses     startpage,port;

procedure TForm5.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form1.Close;
end;

procedure TForm5.BitBtn1Click(Sender: TObject);
```

```
begin
Label2.Color:=clAqua;
if started then
  begin
  Setport($300,1);{Pause}
  BitBtn1.Caption:='Start';
  started:=False;
  Timer2.Enabled:=False;
  end
  else
  begin
  Setport($300,2);{Start}
  BitBtn1.Caption:='Pause';
  started:=true;
  Timer2.Enabled:=True;
  end;
end;

procedure TForm5.BitBtn2Click(Sender: TObject);
begin
Setport($300,4);    //reset
Setport($300,16);   //set orderly
i:=0;
AutoL:=False;
Timer1.Enabled:=False;
BitBtn3.Caption:='AutoLoad On';
Timer2.Enabled:=False;
BitBtn1.Caption:='Start';
started:=False;
tickcounter:=0;
time:=0;
Totaltime:=0;
hh:=0;
mm:=0;
ss:=0;
Label11.Caption:='  00:00:00.00';
Label2.Color:=clBtnHighLight;
Label3.Color:=clBtnHighLight;
Label4.Color:=clBtnHighLight;
Label5.Color:=clBtnHighLight;
```

```
BitBtn1.Enabled:=True;
BitBtn3.Enabled:=True;
Edit1.Text:='----------';
Edit2.Text:='----------';
Edit3.Text:='----------';
Edit4.Text:='----------';
Edit5.Text:='----------';
Edit6.Text:='----------';
Edit7.Text:='----------';
Edit8.Text:='----------';
Posit:=0;
end;

procedure TForm5.BitBtn3Click(Sender: TObject);
begin
if AutoL= False then
  begin
  AutoL:=True;
  BitBtn3.Caption:='AutoLoad Off';
  Timer1.Enabled:=True;
  ms := Getport($300);
  ms := Getport($300);
  while  (ms<>254)and(i<20) do
    begin
    s  := Getport($301);
    min:= Getport($302);
    i:=i+1;
    dt[i]:=min*60 + s + ms/100;
    if dt[i]<>0 then
      begin
      time:=time+dt[i]  ;
      end
      else i:=i-1;
    ms := Getport($300);
    end; {while loop}

  end {if}
  else
  begin
  AutoL:=False;
```

```
  Timer1.Enabled:=False;
  BitBtn3.Caption:='AutoLoad On';
  end;
end;

procedure TForm5.Timer1Timer(Sender: TObject);
begin
 tempposit:=Getport($303);
   if tempposit in [1..4]  then
     begin
     posit :=tempposit;
     if posit = 1 then  //auto start
        begin
        BitBtn1.Caption:='Pause';
        Label2.Color:=clAqua;
        Timer2.Enabled:=True;
        end;
     end;
 ms := Getport($300);
   if ms<>254 then
   begin
      s  := Getport($301);
      min:= Getport($302);
      i:=i+1;
      dt[i]:=min*60 + s + ms/100;
      if dt[i]<>0 then
       begin
       time:=time+dt[i]  ;
       case i of
       1:begin
        Edit1.Text:=floattostrf(dt[1],ffFixed,10,2);
        v1:= s1/dt[1];
        Edit4.Text:=FloatToStrf(v1,ffFixed,10,2);
        Label3.Color:=clAqua;
        end;
       2:begin
        Edit2.Text:=floattostrf(dt[2],ffFixed,10,2);
        v2:= s2/dt[2];
        Edit5.Text:=FloatToStrf(v2,ffFixed,10,2);
        a2:=(2*(v2-v1))/(dt[1]+dt[2]);
```

```
          Edit7.Text:=FloatToStrf(a2,ffFixed,10,2);
          Label4.Color:=clAqua;
          end;
        3:begin
          Edit3.Text:=floattostrf(dt[3],ffFixed,10,2);
          v3:= s3/dt[3];
          Edit6.Text:=FloatToStrf(v3,ffFixed,10,2);
          a3:=(2*(v3-v2))/(dt[2]+dt[3]);
          Edit8.Text:=FloatToStrf(a3,ffFixed,10,2);
          Label5.Color:=clAqua;
          Timer1.Enabled:=False;
          Timer2.Enabled:=False;
          BitBtn1.Enabled:=False;
          BitBtn3.Enabled:=False;
          hh:=trunc(time/(60*60));
          mm:=trunc((time-hh*60*60)/60);
          ss:=time-(hh*60*60 + mm*60);
          str(hh,hhs);
          str(mm,mms);
          str(ss:0:2,sss);
          if length(hhs)<2 then hhs:='0'+hhs;
          if length(mms)<2 then mms:='0'+mms;
          if length(sss)<5 then sss:='0'+sss;
          Label11.Caption:='  '+hhs+':'+mms+':'+sss;
          end;
        end;{case}
       end
     else i:=i-1;
  end; {if}
end;

procedure TForm5.FormCreate(Sender: TObject);
begin
if Form1.CheckBox4.Checked= False then
  begin
  Label7.Visible:=False;
  Edit4.Visible:=False;
  Edit5.Visible:=False;
  Edit6.Visible:=False;
  Label6.Left:=Label6.Left+70;
```

```
    Edit1.Left:=Edit1.Left+70;
    Edit2.Left:=Edit2.Left+70;
    Edit3.Left:=Edit3.Left+70;
    end
    else
    begin
    s1:=strtofloat(Form1.Edit6.text);
    s2:=strtofloat(Form1.Edit7.text);
    s3:=strtofloat(Form1.Edit8.text);
    Label12.Caption:=Form1.Edit6.text;
    Label13.Caption:=Form1.Edit7.text;
    Label14.Caption:=Form1.Edit8.text;
    end;
  if Form1.CheckBox5.Checked=False then
    begin
    Label8.Visible:=False;
    Label9.Visible:=False;
    Edit7.Visible:=False;
    Edit8.Visible:=False;
    end;
  if Form1.CheckBox4.Checked and (Form1.CheckBox5.Checked=False) then
    begin
    Label6.Left:=Label6.Left+24;
    Edit1.Left:=Edit1.Left+24;
    Edit2.Left:=Edit2.Left+24;
    Edit3.Left:=Edit3.Left+24;
    Label7.Left:=Label7.Left+32;
    Edit4.Left:=Edit4.Left+32;
    Edit5.Left:=Edit5.Left+32;
    Edit6.Left:=Edit6.Left+32;
    end;
  Timer1.Enabled:=False;
  Timer2.Enabled:=False;
  Time:=0;
  Time_D:=0;
  Posit:=0;
  setport($300,16);
  end;

  procedure TForm5.Timer2Timer(Sender: TObject);
```

```
begin
tickcounter:=tickcounter+1;
time_D:=tickcounter/8.9;
hh:=trunc(time_D/(60*60));
mm:=trunc((time_D-hh*60*60)/60);
ss:=time_D-(hh*60*60 + mm*60);
str(hh,hhs);
str(mm,mms);
str(ss:0:2,sss);
if length(hhs)<2 then hhs:='0'+hhs;
if length(mms)<2 then mms:='0'+mms;
if length(sss)<5 then sss:='0'+sss;
Label11.Caption:='  '+hhs+':'+mms+':'+sss;
end;

end.
```

โปรแกรมประยุกต์การเคลื่อนที่เป็นรอบ

```
unit circuit;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 ExtCtrls, StdCtrls, Buttons,comctrls,result;

type
 TForm6 = class(TForm)
  Label1: TLabel;
  Label11: TLabel;
  Timer2: TTimer;
  BitBtn1: TBitBtn;
  Timer1: TTimer;
  BitBtn3: TBitBtn;
  BitBtn2: TBitBtn;
  BitBtn4: TBitBtn;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
  Label5: TLabel;
```

```
    ScrollBox1: TScrollBox;
    Label6: TLabel;
    TreeView1: TTreeView;
    BitBtn5: TBitBtn;
    BitBtn6: TBitBtn;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure BitBtn5Click(Sender: TObject);
    procedure BitBtn6Click(Sender: TObject);
  private
    { Private declarations }

  public
    { Public declarations }

  end;
var
  Form6: TForm6;
  ResultForm:TForm7;
  LapL: array[1..100]of TLabel;
  tL  : array[1..100]of TLabel;
  t   : array[1..100]of  real;
  dt2 : array[1..100,1..4] of real;
  dtL : array[1..100,1..4]of TLabel;
  vL  : array[1..100]of TLabel;
  v   : array[1..100]of real;
  dv  : array[1..100,1..4]of real;
  dvL : array[1..100,1..4]of TLabel;
  dv2 : array[1..100]of real;
  lap,d,n   :byte;  //0-255
 tickcounter:integer;
 time:real;
 j       :integer;
 ms,s,min,i:integer;
```

```
  dt:array[0..400] of real;
  distance:real;
  dis    :array[1..4] of real;
  hh,mm:integer;
  ss:real;
  hhs,mms,sss,mss:string[5];
  started ,AutoL:Boolean;
  posit ,preposit,tempposit :byte;
  treeview2:TTreeview;
  treenodes1:TTreenodes;
  treenode1:TTreenode;
  treenodes2:TTreenodes;
  treenode2:TTreenode;
  ms_s,s_s,min_s:string[3];
  minute,second:string[5];
  expt,expv:Boolean;

implementation

uses startpage,port,result1;

{$R *.DFM}

procedure TForm6.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Form1.Close;
end;

procedure TForm6.FormCreate(Sender: TObject);
var    j,sect : byte;
begin
setport($300,16);
expV:=False;
expT:=False;
if form1.CheckBox16.Checked then
begin
BitBtn6.left:=80;
BitBtn5.visible:=true;
Treeview2:=TTreeview.Create(self);
treeview2.Parent:=self;
```

```
treeview2.Visible:=True;
treeview1.Left:=Treeview1.Left-100;
treeview2.Left:=Treeview1.Left+200;
treeview2.top:=Treeview1.top;
treeview2.width:=Treeview1.width;
treeview2.height:=Treeview1.height;
treeview2.indent:=Treeview1.indent;
end;
dis[1]:=StrtoFloat(Form1.Edit9.Text);
dis[2]:=StrtoFloat(Form1.Edit10.Text);
dis[3]:=StrtoFloat(Form1.Edit11.Text);
dis[4]:=StrtoFloat(Form1.Edit13.Text);
lap:=0;
preposit:=0;
distance:=dis[1]+dis[2]+dis[3]+dis[4];
n:= StrToInt(Form1.Edit12.text)  ;
Label1.Caption:='LAP 0/'+ Form1.Edit12.text;
for j:=1 to n do
begin
   with treeview1 do
   begin
   treenodes1:=treeview1.Items.Create(treeview1);
   treenodes1.Add(treenode1,'lap '+inttostr(j));
   for sect:=1 to 4 do
   Items.AddChild(treeview1.items[5*(j-1)],'section'+inttostr(sect));
   end;
  if form1.CheckBox16.Checked then
  begin
   with treeview2 do
   begin
   treenodes2:=treeview2.Items.Create(treeview2);
   treenodes2.Add(treenode2,'lap '+inttostr(j));
   for sect:=1 to 4 do
   Items.AddChild(treeview2.items[5*(j-1)],'section'+inttostr(sect));
   end;
  end;
LapL[j]:=TLabel.Create(Form6);
LapL[j].parent:=ScrollBox1;
LapL[j].enabled:=True;
LapL[j].Caption:='LAP '+inttostr(j);
```

```
LapL[j].font.Color:=clBlue;
LapL[j].Visible:=True;
LapL[j].Left:=50;
LapL[j].Top:=10+(j-1)*20;
tL[j]:=TLabel.Create(Form6);
tL[j].parent:=ScrollBox1;
tL[j].enabled:=True;
tL[j].font.Color:=clRed;
tL[j].Visible:=False;
tL[j].Left:=100;
tL[j].Top:=10+(j-1)*20;
tL[j].Caption:='Test t';
tL[j].Visible:=True;
if Form1.CheckBox16.Checked then
  begin
  vL[j]:=TLabel.Create(Form6);
  vL[j].parent:=ScrollBox1;
  vL[j].enabled:=True;
  vL[j].font.Color:=clRed;
  vL[j].Visible:=False;
  vL[j].Left:=200;
  vL[j].Top:=10+(j-1)*20;
  vL[j].Caption:='Test v';
  vL[j].Visible:=True;
  end;
end;  //for j
end;

procedure TForm6.Timer2Timer(Sender: TObject);
begin
tickcounter:=tickcounter+1;
time:=tickcounter/8.9;
hh:=trunc(time/(60*60));
mm:=trunc((time-hh*60*60)/60);
ss:=time-(hh*60*60 + mm*60);
str(hh,hhs);
str(mm,mms);
str(ss:0:2,sss);
if length(hhs)<2 then hhs:='0'+hhs;
if length(mms)<2 then mms:='0'+mms;
```

```
if length(sss)<5 then sss:='0'+sss;
Label11.Caption:='  '+hhs+':'+mms+':'+sss;
end;

procedure TForm6.BitBtn1Click(Sender: TObject);
begin
if started then
  begin
  Setport($300,1);{Pause}
  BitBtn1.Caption:='Start';
  started:=False;
  Timer2.Enabled:=False;
  end
  else
  begin
  Setport($300,2);{Start}
  BitBtn1.Caption:='Pause';
  started:=true;
  Timer2.Enabled:=True;
  end;
end;

procedure TForm6.BitBtn3Click(Sender: TObject);
var tem:integer;
begin
if AutoL= False then
  begin
  BitBtn4.SetFocus;
  Tempposit:=Getport($303);
  if tempposit<>6 then
  begin
  posit:=Tempposit;
  preposit:=posit;
  end;
  AutoL:=True;
  BitBtn3.Caption:='AutoLoad Off';
  Timer1.Enabled:=True;
  ms := Getport($300);
  ms := Getport($300);
  while  (ms<254) do
```

```
begin
s  := Getport($301);
min:= Getport($302);
i:=i+1;
dt[i]:=min*60 + s + ms/100;
if i mod 4 <>0 then d:=(i mod 4) else d:=4;
if dt[i]<>0 then
  begin
  if i=1 then Lap:=1;
  ms_s:=inttostr(ms);
  s_s :=inttostr(s);
  min_s:=inttostr(min);
  if length(ms_s)<2 then ms_s:='0'+ms_s;
  if length(s_s)<2 then s_s:='0'+s_s;
  if length(min_s)<2 then min_s:='0'+min_s;
  tem:= 5*(lap-1)+d;
  Treeview1.Items[tem].text:=min_s+':'+s_s+'.'+ms_s;
  if Form1.CheckBox16.Checked then
   begin
   dv2[i]:=dis[d]/dT[i];
   Treeview2.Items[tem].text:= FloatToStrf(dv2[i],ffFixed,15,2) ;
   end;
 time:=time+dt[i]  ;
 dt2[lap,d]:=dt[i];
 if i mod 4 = 0 then
    begin
    t[lap]:=dt2[lap,1]+dt2[lap,2]+dt2[lap,3]+dt2[lap,4];
    tL[lap].Caption:=FloatToStrf(t[lap],ffFixed,15,2);
    minute:=inttostr(trunc(t[lap])div 60);
    if length(minute)<2 then minute:='0'+minute;
    second:=floattostrf(t[lap]- (trunc(t[lap])div 60)*60,ffFixed,15,2);
    if (t[lap]- (trunc(t[lap])div 60)*60) <10 then second:='0'+second;
    if length(second)<5 then second:=second+'0';
    treeview1.Font.Style:=[fsBold];
    Treeview1.Items[5*(lap-1)].text:=Treeview1.Items[5*(lap-1)].text +' --> '+ minute+':'+second ;
    treeview1.Font.Style:=[];
    if Form1.CheckBox16.Checked then
      begin
      v[lap]:=distance/t[lap];
      vL[lap].Caption:=FloatToStrf(v[lap],ffFixed,15,2);
```

```
            vL[lap].Visible:=True;
            treeview2.Font.Style:=[fsBold];
            Treeview2.Items[5*(lap-1)].text:=Treeview2.Items[5*(lap-1)].text+' --> '+FloatToStrf(v[lap],ffFixed,15,2) ;
            treeview2.Font.Style:=[];
            end;
          end;
        Lap:=(i div 4)+1;
        end
        else i:=i-1;
      ms := Getport($300);
      ms := Getport($300);
      end; {while loop}
      if posit<>0 then
      begin
      Timer2.Enabled:=True;
      lap:=(i div 4)+1
      end
      else lap:=0;
      tickcounter:=round(time*8.9);
      hh:=trunc(time/(60*60));
      mm:=trunc((time-hh*60*60)/60);
      ss:=time-(hh*60*60 + mm*60);
      str(hh,hhs);
      str(mm,mms);
      str(ss:0:2,sss);
      if length(hhs)<2 then hhs:='0'+hhs;
      if length(mms)<2 then mms:='0'+mms;
      if length(sss)<5 then sss:='0'+sss;
      Label11.Caption:='   '+hhs+':'+mms+':'+sss;
    end {if}
    else
    begin
    AutoL:=False;
    Timer1.Enabled:=False;
    BitBtn3.Caption:='AutoLoad On';
    end;
  end;

procedure TForm6.BitBtn2Click(Sender: TObject);
var g,sect1:byte;
```

```
begin
treeview1.Items.Clear;
if form1.CheckBox16.Checked then  treeview2.Items.Clear;
expV:=False;
expT:=False;
Setport($300,4);
setport($300,16);
i:=0;
j:=0;
lap:=0;
preposit:=0;
AutoL:=False;
Timer1.Enabled:=False;
BitBtn3.Caption:='AutoLoad On';
Timer2.Enabled:=False;
BitBtn1.Caption:='Start';
started:=False;
tickcounter:=0;
time:=0;
hh:=0;
mm:=0;
ss:=0;
Label11.Caption:='  00:00:00.00';
BitBtn1.Enabled:=True;
BitBtn3.Enabled:=True;
Label2.Color:=clMaroon;
Label3.Color:=clMaroon;
Label4.Color:=clMaroon;
Label5.Color:=clMaroon;
Label1.Caption:='LAP 0/'+ Form1.Edit12.text;
for g:=1 to n do
begin
   with treeview1 do
   begin
   treenodes1:=treeview1.Items.Create(treeview1);
   treenodes1.Add(treenode1,'lap '+inttostr(g));
   for sect1:=1 to 4 do
   Items.AddChild(treeview1.items[5*(g-1)],'section'+inttostr(sect1));
   end;
 if form1.CheckBox16.Checked then
```

```
  begin
   with treeview2 do
   begin
   treenodes2:=treeview2.Items.Create(treeview2);
   treenodes2.Add(treenode2,'lap '+inttostr(g));
   for sect1:=1 to 4 do
   Items.AddChild(treeview2.items[5*(g-1)],'section'+inttostr(sect1));
   end;
  end;
end;
end;

procedure TForm6.Timer1Timer(Sender: TObject);
var tem:integer;
begin
 tempposit:=Getport($303);
 if tempposit<>6 then posit :=tempposit;
 if posit in[1..4] then
 begin
  ms := Getport($300);
  ms := Getport($300);
   if ms<>254 then
   begin
      s  := Getport($301);
      min:= Getport($302);
      i:=i+1;
      dt[i]:=min*60 + s + ms/100;
      if i mod 4 <>0 then d:=(i mod 4) else d:=4;
      if dt[i]<>0 then
       begin
       if i=1 then Lap:=1;
       dt2[lap,d]:=dt[i];
       time:=time+dt[i]  ;
      ms_s:=inttostr(ms);
      s_s :=inttostr(s);
      min_s:=inttostr(min);
      if length(ms_s)<2 then ms_s:='0'+ms_s;
      if length(s_s)<2 then s_s:='0'+s_s;
      if length(min_s)<2 then min_s:='0'+min_s;
      tem:= 5*(lap-1)+d;
```

```
      Treeview1.Items[tem].text:=min_s+':'+s_s+'.'+ms_s;
      if Form1.CheckBox16.Checked then
       begin
       dv2[i]:=dis[d]/dT[i];
       Treeview2.Items[tem].text:= FloatToStrf(dv2[i],ffFixed,15,2) ;
       end;
       if i mod 4 = 0 then
         begin
         t[lap]:=dt2[lap,1]+dt2[lap,2]+dt2[lap,3]+dt2[lap,4];
         tL[lap].Caption:=FloatToStrf(t[lap],ffFixed,15,2);
         minute:=inttostr(trunc(t[lap])div 60);
         if length(minute)<2 then minute:='0'+minute;
         second:=floattostrf(t[lap]- (trunc(t[lap])div 60)*60,ffFixed,15,2);
         if (t[lap]- (trunc(t[lap])div 60)*60) <10 then second:='0'+second;
         if length(second)<5 then second:=second+'0';
         treeview1.Font.Style:=[fsBold];
         Treeview1.Items[5*(lap-1)].text:=Treeview1.Items[5*(lap-1)].text +' --> '+ minute+':'+second ;
         treeview1.Font.Style:=[];
         if Form1.CheckBox16.Checked then
           begin
           v[lap]:=distance/t[lap];
           vL[lap].Caption:=FloatToStrf(v[lap],ffFixed,15,2);
           vL[lap].Visible:=True;
           treeview2.Font.Style:=[fsBold];
           Treeview2.Items[5*(lap-1)].text:=Treeview2.Items[5*(lap-1)].text+' --> '+FloatToStrf(v[lap],ffFixed,15,2) ;
           treeview2.Font.Style:=[];
           end;
         end;
       end
     else i:=i-1;
  end; //if ms
case posit of
1:begin
     if preposit=4 then  lap:=lap+1;
     if preposit=0 then
     begin
      lap:=1;
      Timer2.Enabled:=True;
      bitbtn1.Caption:='Pause';
      started:=true;
```

```
        end;
        Label2.Color:=clBlue;
        Label3.Color:=clMaroon;
        Label4.Color:=clMaroon;
        Label5.Color:=clMaroon;
     end;
   2:begin
        Label2.Color:=clMaroon;
        Label3.Color:=clBlue;
        Label4.Color:=clMaroon;
        Label5.Color:=clMaroon;
     end;

   3:begin
        Label2.Color:=clMaroon;
        Label3.Color:=clMaroon;
        Label4.Color:=clBlue;
        Label5.Color:=clMaroon;
     end;
   4:begin
        Label2.Color:=clMaroon;
        Label3.Color:=clMaroon;
        Label4.Color:=clMaroon;
        Label5.Color:=clBlue;
     end;
  end; //case
 if lap<=n then Label1.Caption:='LAP '+IntToStr(lap)+'/'+ Form1.Edit12.text
   else
   begin
   Timer1.Enabled:=False;
   Timer2.Enabled:=False;
   BitBtn4Click(sender);
   end;
 preposit:=posit;
 end;  //if posit
end;

procedure TForm6.BitBtn4Click(Sender: TObject);
var v:integer;
begin
```

```
Timer1.Enabled:=False;
Timer2.Enabled:=False;
time:=0;
for v:=1 to n do time:=time+t[v];
hh:=trunc(time/(60*60));
mm:=trunc((time-hh*60*60)/60);
ss:=time-(hh*60*60 + mm*60);
str(hh,hhs);
str(mm,mms);
str(ss:0:2,sss);
if length(hhs)<2 then hhs:='0'+hhs;
if length(mms)<2 then mms:='0'+mms;
if length(sss)<5 then sss:='0'+sss;
Label11.Caption:='  '+hhs+':'+mms+':'+sss;
showresult;
end;

procedure TForm6.BitBtn5Click(Sender: TObject);
var recurse:boolean;
i:integer;
begin
recurse:=true;
expV:=not(expV);
if expV=true then
begin
for i:=0 to treeview1.Items.Count -1 do
treeview2.Items.Item[i].expand(recurse);
end
else
begin
for i:=0 to treeview1.Items.Count -1 do
treeview2.Items.Item[i].collapse(recurse);
end;
end;

procedure TForm6.BitBtn6Click(Sender: TObject);
var recurse:boolean;
i:integer;
begin
recurse:=true;
```

```
expT:=not(expT);
if expT=true then
begin
for i:=0 to treeview1.Items.Count -1 do
treeview1.Items.Item[i].expand(recurse);
end
else
begin
for i:=0 to treeview1.Items.Count -1 do
treeview1.Items.Item[i].collapse(recurse);
end;
end;

end.
```

กิตติกรรมประกาศ

สำหรับความก้าวหน้าในโครงงานระบบตรวจจับเวลา 4 ช่องทางโดยไมโครโปรเซสเซอร์นี้ ได้มีความก้าวหน้าและแก้ปัญหาอุปสรรคต่าง ๆ ไปได้ก็ด้วยความกรุณาของ ผศ.จิรวัฒน์ ปานกลาง ที่ได้ให้คำปรึกษา แนะนำ ตลอดจนแนวคิดริเริ่มที่เป็นประโยชน์ต่อโครงงานเป็นอย่างมาก ภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ที่ได้ให้การสนับสนุนในด้านงบประมาณ และที่จะขาดไม่ได้ก็คือคณาจารย์ทุกท่านที่ได้สั่งสอนให้คณะผู้จัดทำมีความรู้ในด้านต่าง ๆ และขอขอบคุณเพื่อน ๆ พี่ ๆ ที่ได้สนับสนุนให้กำลังใจเสมอมา

สุดท้ายที่สำคัญที่สุด ต้องขอกราบขอบพระคุณบิดา มารดา ของคณะผู้จัดทำที่ได้ให้การเลี้ยงดูอบรมสั่งสอน ให้การสนับสนุนในทุก ๆ ด้าน รวมทั้งกำลังใจที่มีให้เสมอมาที่ทำให้โครงงานนี้สำเร็จลุล่วงไปด้วยดี

(นายนิภัทร โรจน์รัตนวาณิชย์)

(นายนรา ตันตราธิษฐาน)

(นายบรรพต เปี่ยมถาวรพจน์)

บรรณานุกรม

1. ธานินทร์ ถาวรศาสนวงศ์ และ ทินกร ดุ๊ก, "การอินเทอร์เฟส IBM PC", สำนักพิมพ์ฟิสิกส์เซ็นเตอร์, 270 หน้า,2536
2. ปรเมษฐ์ ประนยานันทน์ และ ปิยพงศ์ เผ่าวณิช, "คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51" ซีเอ็ดยูเคชั่น, 380 หน้า, 2521
3. Barry B. Brey, " Intel Microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486 , Pentium ,and Pentium Pro Processor Architecture, Programming and Interfacing", Prentice Hall, 907p.,1997
4. Willis J Tompkins and John G webster , "Interfacing Sensors To the IBM PC",Prentice Hall, 447p.,1988