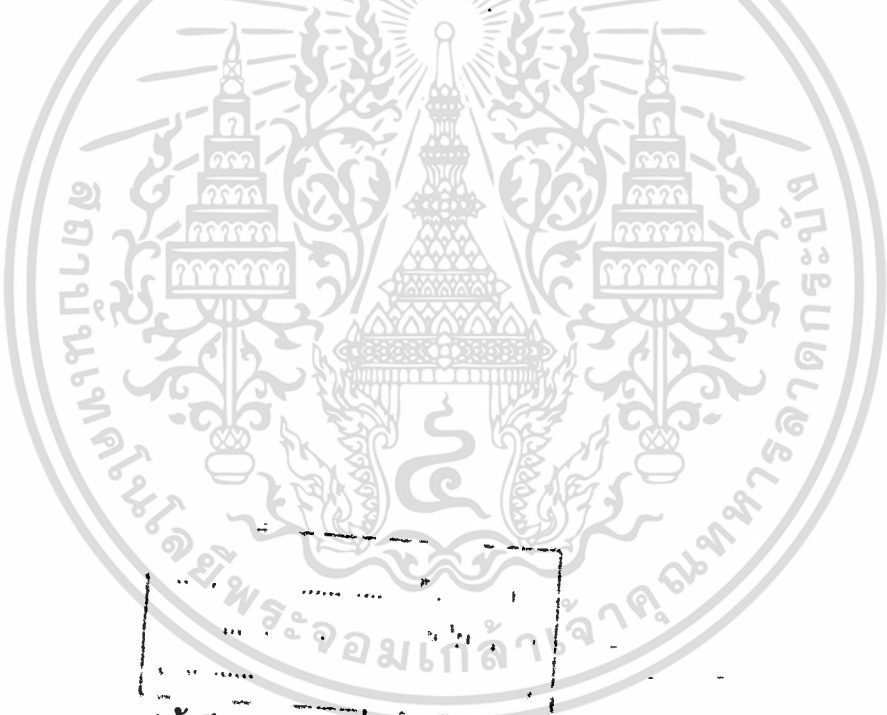
การพัฒนาระบบทัศนศึกษาโดยใช้ไมโครคอมพิวเตอร์

MICROCOMPUTER APPLICATION IN EDUCATION SYSTEM

โดย

นางสาวจุฑามาส เข็มทอง

นางสาวจุฬารัตน์ ดอกกุหลาบ



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบทัศนศึกษาโดยใช้ไมโครคอมพิวเตอร์

(MICROCOMPUTER APPLICATION IN EDUCATION SYSTEM)

ผู้จัดทำ

1. นางสาวจุฑามาส เข็มทอง 34102066
2. นางสาวจุฬารัตน์ ดอกกุหลาบ 34102069

.....

(ดร.กิติพล ชิตสกุล)

อาจารย์ที่ปรึกษา

การพัฒนาระบบทัศนศึกษาโดยใช้ไมโครคอมพิวเตอร์

MICROCOMPUTER APPLICATION IN EDUCATION SYSTEM

โดย นางสาวจุฑามาส เข็มทอง
นางสาวจุฬารัตน์ ตอกกุหลาบ

อาจารย์ที่ปรึกษา ดร.กิติพล ชิตสกุล

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาระบบทัศนศึกษาในห้องสมุดโดยนำไมโครคอมพิวเตอร์มาใช้ร่วมกับวงจรอิเล็กทรอนิกส์ภายนอกและใช้การเก็บข้อมูล (เช่น รูปภาพ) ในรูปแบบฐานข้อมูล ทำให้ข้อมูลถูกเก็บไว้อย่างเป็นระเบียบและง่ายต่อการค้นหา ในการทำงานจะมีส่วนตรวจจับซึ่งใช้หลักการของรีโมทคอนโทรล สะท้อนสัญญาณอินฟราเรดตรวจจับตัวคนที่มานั่งหน้าจอคอมพิวเตอร์ เพื่อไปควบคุมการเปิดเมนูที่หน้าจอคอมพิวเตอร์โดยอัตโนมัติ ในเมนูจะแสดงรายการของข้อมูลที่ถูกจัดเก็บในรูปแบบของไฟล์ข้อมูลในคอมพิวเตอร์และสามารถเลือกข้อมูลที่ต้องการได้โดยเลือกจากเมนูที่หน้าจอ จากนั้นจะแสดงภาพทางหน้าจอคอมพิวเตอร์ เมื่อผู้ต้องการใช้งานเลิกการทำงานโดยลุกออกจากหน้าจอ หน้าจอคอมพิวเตอร์จะปิดโดยอัตโนมัติในเวลาที่กำหนด

ABSTRACT

This project is concerned to development of Visual Education System for using in library. The system consist of 3 parts,micro-computer,user sensing system and software for controlling and searching the knowledge database. During stand by mode,user sensing system continuously detects users by mean of infrared sensing system. If there is one in front of monitor,user interface menu then appears on the monitor and leaves the user to enjoy with his selected topic. Until the user leaves off the monitor, the system return to stand by mode automatically.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลงได้ ด้วยความช่วยเหลืออย่างดียิ่งจากหลาย ๆฝ่าย คณะผู้จัดทำใคร่ขอขอบคุณทุกท่าน ที่มีส่วนสนับสนุน ช่วยเหลือ แนะนำ ในทุก ๆด้าน

คณะผู้จัดทำขอขอบพระคุณ ดร.กิติพล ชิตสกุล อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งช่วยให้คำปรึกษา ทำให้ปริญญานิพนธ์นี้สำเร็จลุล่วงไปด้วยดี

ขอขอบพระคุณ อาจารย์พิชัย ดูศิริวาณิชกร ซึ่งช่วยเหลือคณะผู้จัดทำในการสแกนภาพที่ใช้ในโครงงานนี้

ขอขอบพระคุณ อาจารย์เทอดศักดิ์ ลิ่วหาทอง ผู้ซึ่งให้คำแนะนำในการทำโครงงานนี้

ขอขอบคุณเพื่อน ๆภาควิชาอิเล็กทรอนิกส์ เพื่อน ๆภาควิชาคอมพิวเตอร์ และเพื่อน ๆ ภาควิชาโทรคมนาคม ที่ช่วยเหลือและให้คำแนะนำต่าง ๆ

สุดท้ายนี้ ขอกราบขอบพระคุณ บิดา มารดา ของคณะผู้จัดทำ ที่ให้การสนับสนุนและคอยให้กำลังใจ จนกระทั่งปริญญานิพนธ์นี้สำเร็จลงด้วยดี

สารบาญ

สารบาญตาราง

สารบาญภาพ

บทที่ 1
บทนำ

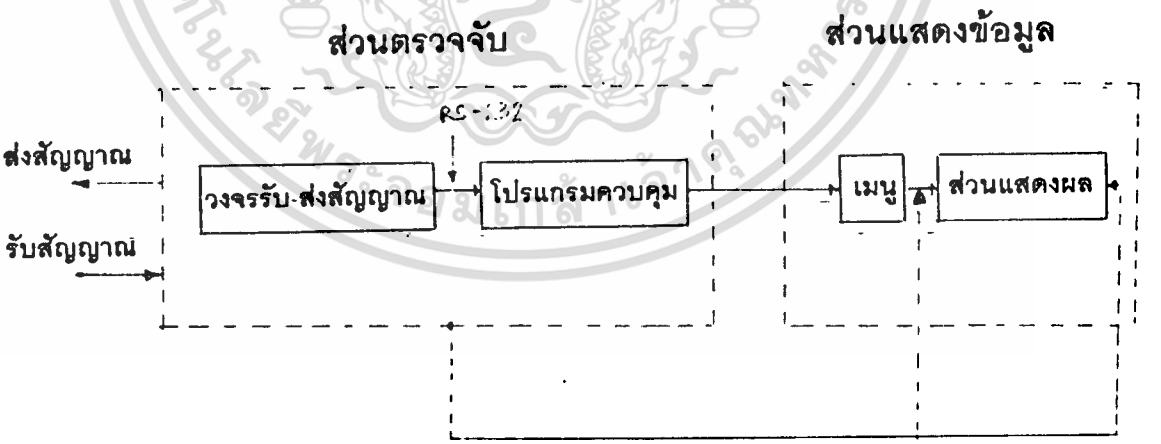
โครงงานนี้เป็นการพัฒนาระบบการนำเสนอข้อมูล โดยใช้ไมโครคอมพิวเตอร์ร่วมกับวงจรอิเล็กทรอนิกส์ภายนอก โดยใช้หลักการของรีโมทคอนโทรลในการสะท้อนสัญญาณอินฟาเรดของส่วนตรวจจับ และส่งสัญญาณนั้นไปควบคุมการทำงานของไมโครคอมพิวเตอร์ ให้แสดงหน้าจอเมนูขึ้นมา จากนั้นจะแสดงข้อมูลที่เลือกทางหน้าจอ

หลักการทำงานของโครงงานนี้คือการใช้คอมพิวเตอร์เป็นตัวควบคุมในการส่งสัญญาณไปยังส่วนตรวจจับเพื่อตรวจจับผู้ที่ต้องการใช้งาน เมื่อมีผู้ต้องการใช้งานมานั่งหน้าจอคอมพิว เตอร์ ส่วนตรวจจับจะส่งสัญญาณไปควบคุมให้แสดงเมนูบนหน้าจอคอมพิวเตอร์ ผู้ใช้งานสามารถเลือกหัวข้อเรื่องตามที่ต้องการ จากนั้น จะแสดงข้อมูลของหัวข้อเรื่องที่เลือกนั้น และเมื่อผู้ใช้เลิกการใช้งานโดยลุกไปจากหน้าจอแล้ว จะทำการปิดหน้าจอคอม พิวเตอร์โดยอัตโนมัติ

จากที่กล่าวมาข้างต้น สามารถแบ่งการทำงานของโครงงานนี้ได้เป็น 2 ส่วน คือ

1. ส่วนตรวจจับ (SENSOR PART)
2. ส่วนแสดงผลข้อมูล (DISPLAY PART)

การทำงานสามารถแสดงได้ดังบล็อกไดอะแกรมดังต่อไปนี้



รูปที่ 1.1 บล็อกไดอะแกรมระบบทัศนศึกษา

บทที่ 2
ลักษณะและความเป็นมาของโครงงาน

การทำโครงงานเริ่มแรก มีหลักการทำงาน คือ ขั้นตอนแรกเริ่มจาก เมื่อมีผู้ต้องการใช้งานมานั่งหน้าจอคอมพิวเตอร์ ส่วนตรวจจับจะตรวจจับได้ว่ามีคนมานั่งหลังจากนั้นเป็นเวลา ประมาณ 10 วินาทีจึงส่งสัญญาณผ่านพอร์ตอนุกรม(RS-232)ไปทริกหน้าจอคอมพิวเตอร์ให้แสดงเมนู ที่ต้องให้ผู้ใช้รอเป็นเวลา 10 วินาทีก็เพื่อให้แน่ใจว่าคนที่มาอยู่หน้าจอคอมพิวเตอร์ต้องการใช้งานจริงๆ มิใช่เพียงแค่เดินผ่านไปมาเท่านั้น สัญญาณที่ส่งมาจากส่วนตรวจจับจะมีส่วนโปรแกรมแบบ polling วนรับสัญญาณอยู่ เมื่อมีสัญญาณเข้ามาจะเปิดหน้าจอแสดงเมนูเพื่อให้ผู้ต้องการใช้เลือกหัวข้อที่ต้องการ ต่อไป และเมื่อเลิกการใช้งานแล้วจะทำการปิดหน้าจอคอมพิวเตอร์เองโดยอัตโนมัติ

จากที่กล่าวมาได้แบ่งการทำงานออกเป็น 2 ส่วน คือ

1. ส่วนตรวจจับ
2. ส่วนโปรแกรมควบคุม

ซึ่งรายละเอียดของการทำงานในแต่ละส่วนมีดังต่อไปนี้

2.1 ส่วนตรวจจับ

ในส่วนของการตรวจจับจะใช้หลักการของรีโมทคอนโทรลแบบอินฟาเรดคือรีโมท คอนโทรลที่ใช้แสงอินฟาเรดเป็นตัวกลางในการรับ-ส่ง ประกอบด้วยวงจร 2 ส่วน คือ วงจรภาครับและวงจรภาคส่ง โดยให้ภาคส่งส่งสัญญาณความถี่สูงออกไป เมื่อสัญญาณที่ส่งออกไปกระทบกับสิ่งกีดขวาง เช่น วัตถุ หรือ ตัวคน สัญญาณจะสะท้อนกลับไปยังที่เครื่องรับ เมื่อวงจรที่เครื่องรับ รับสัญญาณที่สะท้อนกลับมา จะนำสัญญาณไปขยายให้มีความแรงมากขึ้นแล้วทำการ detect ความถี่ที่ตรงกับภาคส่ง แล้วนำสัญญาณไปทริกวงจรในส่วนต่อไปให้ทำงาน แต่สำหรับโครงงานนี้จะนำส่วนตรวจจับมาตรวจจับผู้ที่มานั่งหน้าจอคอมพิวเตอร์ แล้วนำสัญญาณที่รับได้ไปทริกให้วงจรสร้างสัญญาณส่งผ่าน RS-232 เพื่อไปควบคุมการเปิดของหน้าจอคอมพิวเตอร์ต่อไปซึ่งสามารถแบ่งการทำงานของส่วนตรวจจับได้เป็น 3 ส่วนคือ

1. วงจรภาคส่ง
2. วงจรภาครับ และ ขยายสัญญาณ
3. วงจรสร้างสัญญาณควบคุมการเปิดหน้าจอคอมพิวเตอร์

ซึ่งจะขอกล่าวรายละเอียดของแต่ละส่วนดังนี้

2.2 วงจรภาคส่ง

หลักการทำงาน

วงจรทางด้านส่งจะสร้างสัญญาณความถี่สูงโดยใช้วงจร Astable สำหรับโครงงานนี้จะใช้ความถี่ 10kHz โดย 555 (Astable)จะถูกรีเซตตลอดเวลาเพื่อให้สร้างสัญญาณความถี่ 10 kHz เพื่อไปขับอินฟาเรด LEDให้ส่งออกไปตลอดเวลา

การออกแบบ Astable จาก 555 สามารถคำนวณค่าได้จาก

$$f = 1/(t^1+t^2)$$

$$t^1 = 0.693(R^A+2R^B)C$$

$$t^2 = 0.693R^AR^BC$$

สำหรับทรานซิสเตอร์ที่ใช้ต่อกับ LED INFRARED จะต้องทนกระแสได้สูง ดังนั้นต้องเป็น Power transistor ส่วน BC547 ที่ต่ออยู่กับ Power transistor (1061)นั้นมีไว้เพื่อจำกัดกระแสที่ไหลผ่าน LED INFRARED

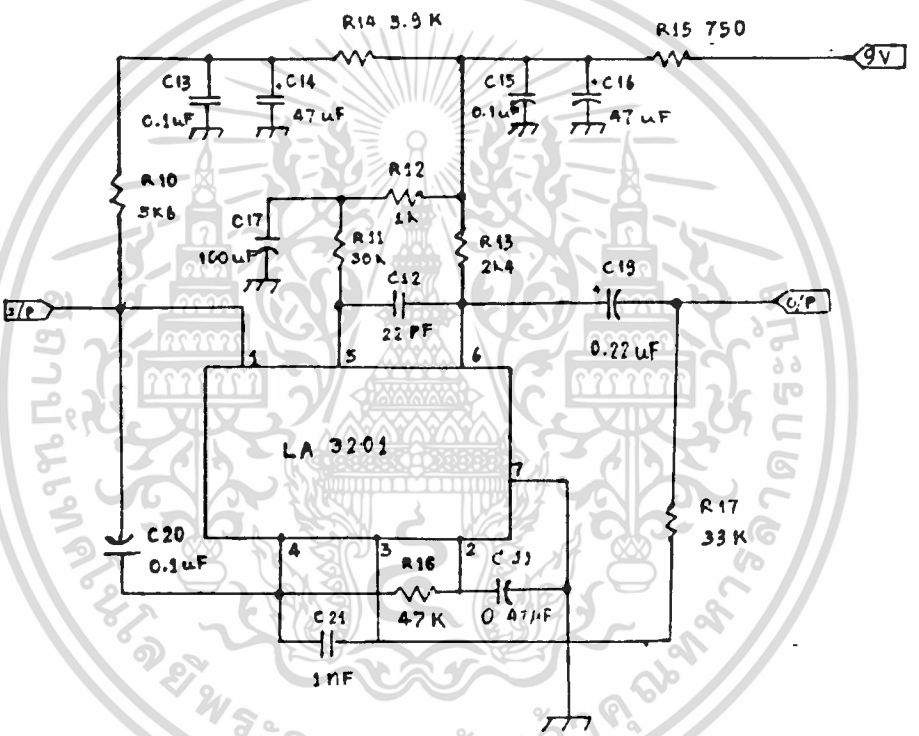
2.3 วงจรภาครับ

หลักการทำงาน

สัญญาณควบคุมที่ส่งมาจากทางด้านส่งสามารถรับได้โดยใช้ MRD 300 แต่สัญญาณที่รับเข้ามานั้นมีขนาด Amplitude ที่เล็กเกินไป นอกจากนี้อาจจะมีสัญญาณรบกวนที่เกิดมาจากแสงภายนอกติดเข้ามาก็ได้ ดังนั้นจึงจำเป็นที่จะต้องกรองเอาเฉพาะสัญญาณความถี่ 10 kHz เท่านั้น แต่เนื่องจากสัญญาณมีขนาดเล็กและไม่สามารถที่จะทำการขยายสัญญาณด้วย Gain สูงๆ พร้อม ๆ กับมี Bandwidth กว้างๆ ได้จึงจำเป็นต้องขยายก่อนที่จะกรองสัญญาณซึ่งในที่นี้จะใช้วงจรขยายที่ออกแบบมาจาก LA3201 ซึ่งเป็นปรีแอมป์มีอัตราขยายประมาณ 45 dB แต่ LA3201 ภายในประกอบไปด้วยวงจรขยายแบบ Common Collector ทั้งหมด 3 State ทำให้สัญญาณที่ได้กลับเฟสกับสัญญาณที่รับมา ดังนั้น จึงต้องใช้วงจรขยาย Bandpass แบบ Inverting เพื่อที่จะกรองสัญญาณความถี่ 10 kHz รวมทั้งทำการขยาย และกลับเฟส สัญญาณอีกครั้งหนึ่ง หลังจากได้สัญญาณที่เอาท์พุทของวงจรขยายแล้ว วงจรตรวจจับสัญญาณจะทำหน้าที่ตรวจจับสัญญาณที่มีความถี่ 10 kHz ในที่นี้จะใช้ IC 567 เป็น Tone Decoder นั่นคือจะนำเอาความถี่ของสัญญาณที่รับเข้ามานั้นทำการเปรียบเทียบกับความถี่ภายในตัวมันเอง ซึ่งค่าความถี่ภายในตัวมันนี้สามารถตั้งค่าได้ ถ้าความถี่ทั้งสองนั้นตรงกันแล้วจะทำให้ Output มีค่าลอจิกเป็นศูนย์

2.4 ภาคขยายสัญญาณ

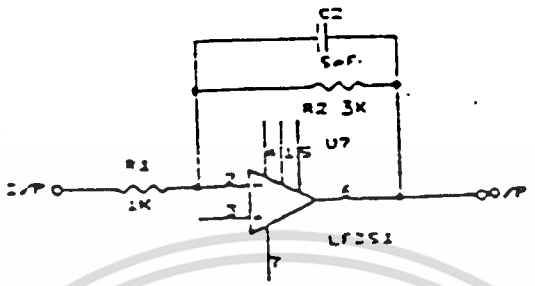
ใช้ IC LA3201 ซึ่งใช้สำหรับขยายสัญญาณจากหัวเล่นเทปแผ่นเสียงซึ่งมีอัตราการขยายสูงนำมาประยุกต์ใช้กับวงจรขยายสัญญาณความถี่ 10 kHz โดยต่อดังรูป



รูปที่ 2 1 Amplifier สำหรับขยายสัญญาณ

วงจรกรองผ่านช่วงความถี่

ใช้ IC LF 351 เป็น OP-AMP แบบ J-FET ในวงจรนี้ใช้สัญญาณทาง AC ดังนั้นขา offset จึงต่อลง GND ได้เลย LF 351 ชุดแรกทำหน้าที่เป็น Buffer โดยต่อวงจรตามรูปข้างล่างซึ่งตั้งค่า Gain เท่ากับ 3 เท่า



รูปที่2.2 Buffer ของวงจรขยาย

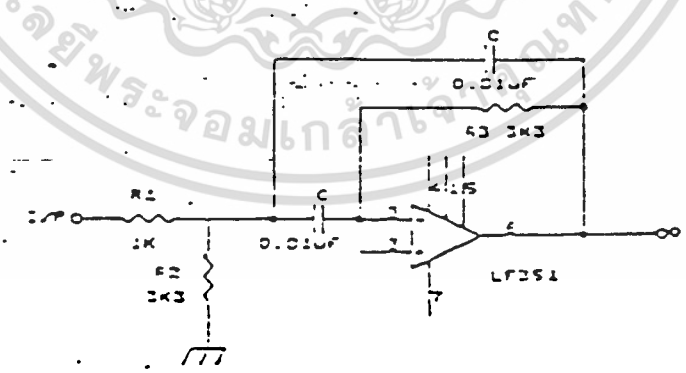
โดยคำนวณได้จากสูตร

$A^f = -R^f/R^{in}$ โดยในวงจรนี้ตั้งค่า R^f = 3 kohm

R^{in} = 1 kohm ดังนั้น A^f = -3

จะเห็นว่าอัตราการขยายติดลบ นั่นคือ output กลับเฟสนั่นเอง

LF 351 ชุดสองทำหน้าที่กรองผ่านช่วงความถี่(Bandpass) ที่ต้องการส่วนความถี่อื่นนอกช่วงนี้จะไม่สามารถผ่านได้ ในวงจรความถี่ที่ต้องการส่งผ่านคือ 10 kHz ซึ่งเป็นความถี่ที่ตัวส่งข้อมูลส่งมา โดยต่อวงจรตามรูป



รูปที่ 2.3 Bandpass Filter

ทำหน้าที่กรองผ่านช่วงความถี่ที่ต้องการ

สำหรับการหาค่า R และ C ต่างๆ เพื่อเลือกช่วงความถี่ที่ส่งผ่านได้จากสูตร

$$f^o = 1/2\ C[(R^1+R^2)/R^1R^2R^3]$$

$$R^1 = Q/2\ f^oG^oC$$

$$R^2 = Q/2\ f^oC(2Q - G^o)$$

$$R^3 = Q/\ f^oC$$

$$G^o = R^o/2R^1$$

$$Q > (G^o/2)$$, Q คือ Quality factor

$$Q = f^o f^{high} - f^{low}$$

จากวงจรกำหนดให้ค่า f^o = 10 kHz C = 0.01 uF

และ Quality factor = 1, $f^{high} - f^{low}$ = 10 kHz นั่นคือช่วงที่ให้ความถี่ผ่านได้ประมาณ 5-15 kHz

R^1 = 1 kohm

R^2 = 3.3 kohm

R^3 = 3.3 kohm

ซึ่งจะเห็นได้ว่า Gain = 3.3 kohm/2 kohm

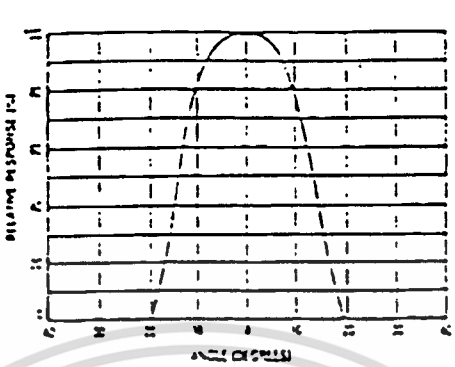
= 1.65 เท่า

การเลือกอุปกรณ์รับแสง

1. ช่วงความยาวคลื่นที่เหมาะกับตัวส่งสัญญาณแสง นั่นคือ พิจารณาว่าใช้อุปกรณ์ที่ส่งเป็นช่วงความยาวคลื่นเท่าใด ในวงจรตัวส่งใช้อินฟาเรดเป็นตัวส่งที่มีช่วงความยาวคลื่นประมาณ 940 mm จึงต้องใช้ตัวรับแสงในช่วงความยาวคลื่น 940 mmได้ดี

2. ตอบสนองความถี่ที่ส่งมาได้ทันเวลา นั่นคือ ต้องพิจารณาความถี่ที่ใช้ส่งว่ามีช่วง Pulse Width เช่นใด ค่าของ Rise time และ Fall time สามารถตอบสนองความถี่ดังกล่าวได้ทัน จากการพิจารณาตัวรับเบอร์ MRD 370 และ MRD 300 จะเห็นว่า MRD 370 มีค่า Rise time 15 uS และ Fall time 25 uS ซึ่งจะทำให้สัญญาณความถี่ 10 kHz ที่รับเข้ามามี Pulse Width ประมาณ 50 uS ที่มี Duty cycle 50% อาจเกิดการผิดเพี้ยนได้ จึงต้องแก้โดยหาตัวส่งที่มีค่า Rise time และ Fall time น้อยกว่านี้ ซึ่งจาก Databook จึงควรใช้ MRD 300

3. มีมุมในการรับแสงกว้าง นั่นคือ พิจารณากราฟ Relative Response กับ Angle ที่ Databook ซึ่งพิจารณาที่ Relative Response ที่ 70 แล้วดูค่า Angle



รูปที่2.4 Angular Response

กราฟสำหรับพิจารณามุมในการรับแสง

4. สามารถจ่ายกระแสได้มากพอ ในขณะที่ได้รับแสงเข้ามา นั่นคือพิจารณาจากกราฟ Light Current กับ Radiantion Flux Density

2.5 วงจรสร้างสัญญาณควบคุมการเปิดหน้าจอคอมพิวเตอร์

เมื่อสัญญาณที่รับเข้ามามีความถี่ตรงกับความถี่ที่เซตไว้ใน Tone decoder แล้วจะได้ output มีค่าลอจิกศูนย์ ทำให้ 567 ทำงานคือให้ลอจิกเป็นศูนย์ แล้วจะทริกให้ 555 ตัวแรกซึ่งต่อเป็น Monostable สร้างสัญญาณพัลส์ ออกทาง output มีค่าลอจิกเป็นหนึ่ง ซึ่งกำหนดให้มี Delaytime 10 s เพื่อทิ้งช่วงเวลาให้แน่ใจว่ามีผู้ต้องการใช้งานจริง ๆ มิใช่เพียงแค่เดินผ่านเท่านั้น สัญญาณพัลส์จาก 555 ชุดแรก จะไปทริก 555 ตัวที่ 2 ซึ่งเป็น Monostable ให้สร้างพัลส์ที่มีความกว้างของพัลส์เท่ากับช่วงของข้อมูล 10 บิทของ baudrate 1200 bit/sec และส่งไปยัง IC MAX232 ซึ่งทำหน้าที่ขยายสัญญาณก่อนที่จะส่งผ่าน RS-232 สัญญาณที่ส่งไปจะไปทริกให้หน้าจอคอมพิวเตอร์เปิดเพื่อแสดงเมนูต่อไป

ช่วงกว้างของพัลส์ (W) สามารถคำนวณได้ดังนี้

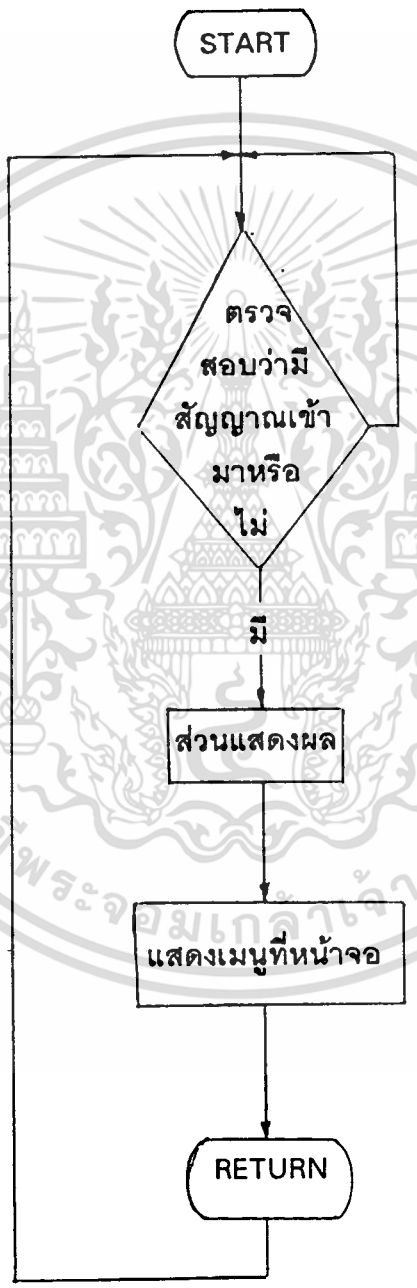
$$W = 1.1RC$$

2.6 ส่วนโปรแกรมคอมพิวเตอร์

ในส่วนของโปรแกรมคอมพิวเตอร์ จะเป็นส่วนจัดการการแสดงผลออกทางหน้าจอคอมพิวเตอร์เมื่อได้รับสัญญาณที่ส่งมาจากส่วนตรวจจับ โดยมีหลักการทำงานคือ ใช้หลักการของการPolling คือการวนลูปตรวจเช็คสัญญาณที่เข้ามาจากพอร์ตอนุกรม(serial port) เมื่อมีสัญญาณเข้ามาก็จะส่งการทำงานไปยังส่วนการแสดงผล ซึ่งจะแสดงเมนูออกทางจอภาพโดยใช้รูปแบบของ Pop up menu เพื่อให้ผู้ต้องการใช้ทำการเลือกหัวข้อที่ต้องการต่อไป

โปรแกรมคอมพิวเตอร์ในส่วนนี้จะควบคุมเฉพาะส่วนที่นำสัญญาณมาทริกเพื่อเปิดหน้าจอคอมพิวเตอร์แสดงรายการเท่านั้น ซึ่งมีโครงสร้างการทำงานดังนี้

FLOW CHART แสดงโครงสร้างการทำงานของส่วนแสดงการเปิดหน้าจอคอมพิวเตอร์



2.7การสร้างเมนูแบบ Pop up

การสร้างเมนูแบบ Pop up นั้นต้องใช้ข้อมูลหลายส่วน ส่วนแรกคือ รายชื่อของตัวเลือกที่แสดงในเมนู ซึ่งเป็นสตริง(string) การส่งข้อมูลนี้ให้ฟังก์ชั่นนำไปใช้งานง่ายที่สุด นำสตริงเหล่านี้ใส่อาร์เรย์(array) 2 มิติ ส่งเฉพาะพอยเตอร์ที่ชี้ไปยังอาร์เรย์นี้ให้กับฟังก์ชั่น การเลือกตัวเลือกของเมนูทำได้ 2 วิธี วิธีแรกคือ เลื่อนแถบสว่างไปยังตัวเลือกที่ต้องการแล้วกด Enter วิธีที่สองคือกดคีย์พิเศษ(hot key) ซึ่งคีย์เหล่านี้ถูกกำหนดไว้ว่าเมื่อถูกกดเท่ากับเลือกตัวเลือกอะไร นอกจากนี้ฟังก์ชั่น pop up ต้องรู้จำนวนของตัวเลือกที่อยู่ในเมนูตำแหน่ง x,y ที่จะแสดงเมนู และต้องรู้ว่าเมนูที่สร้างต้องการตีกรอบด้วยหรือไม่ เป็นต้น

ส่วนหน้าที่ของฟังก์ชั่น pop up() มีดังนี้

1.เก็บหน้าจอ

2.ตีกรอบ

3.แสดงเมนู

4.ตรวจสอบการกดคีย์ และแสดงผล

5.เมื่อเลิกใช้เมนู คืนหน้าจอที่เก็บไว้

2.7.1การเก็บหน้าจอ

การเก็บตัวอักษรที่ปรากฏอยู่บนตำแหน่งใดๆบนจอภาพ ทำได้โดยการอ่านตัวอักษรเข้ามาเก็บไว้ โดยใช้อินเทอร์รัพต์ 16 ฟังก์ชั่นหมายเลข 8 ซึ่งเมื่อเรียกใช้จะส่งรหัสแอสกี และแอตทริบิวต์ของตัวอักษร ณ ตำแหน่งที่เคอร์เซอร์ปรากฏอยู่ และใช้ฟังก์ชั่นที่ทำหน้าที่เปลี่ยนตำแหน่งเคอร์เซอร์ซึ่งคอมไพเลอร์บางตัวมีไว้ให้เรียบร้อยแล้ว เช่น ฟังก์ชั่น gotoxy ( ) แต่ถ้าไม่มี สามารถสร้างฟังก์ชั่นนี้ได้โดยใช้อินเทอร์รัพต์ 16 ฟังก์ชั่นหมายเลข 2 การใช้งานทำได้โดยใส่ค่าคอลัมน์ที่ต้องการให้เคอร์เซอร์ปรากฏในรีจิสเตอร์ DL ค่าของบรรทัดใน DH และหมายเลข page ใน BH (ปกติเท่ากับ 0) ก่อนเรียกใช้

2.7.2การตีกรอบ

ฟังก์ชั่นนี้ทำหน้าที่สร้างกรอบให้กับเมนู โดยส่งตำแหน่งมุมบนซ้ายสุดและมุมล่างขวาสุดไปให้ การวาดกรอบใช้รหัสแอสกี เป็นรูปส่วนต่างๆของกรอบซึ่งอยู่ในชุดอักษรมาตรฐานบนเครื่อง IBM PC หรือ คอมแพติเบิล

2.7.3 การแสดงเมนู

หลักสำคัญของการแสดงเมนูคือ จะส่งพอยน์เตอร์ที่ชี้ไปยังอาร์เรย์ของพอยเตอร์ของสตริงให้กับฟังก์ชั่น pop_up( ) ส่วนวิธีการแสดงสตริงนั้นใช้อาร์เรย์ของพอยเตอร์ ซึ่งพอยเตอร์แต่ละตัวนี้ชี้ไปยังสตริงของตัวเลือก

2.7.4 การตอบสนองอินพุทของผู้ใช้

ผู้ใช้สามารถส่งอินพุทเพื่อเลือกตัวเลือกต่างๆ ได้ 2 วิธีคือ การใช้คีย์ลูกศรเลื่อนแถบสว่างไปยังตัวเลือกที่ต้องการ แล้วกดปุ่ม Enter (โดยทั่วไปแถบสว่างใช้ตัวอักษร reverse) และ Spacebar ก็สามารถใช้เลื่อนแถบสว่างได้ด้วย อีกวิธีหนึ่งคือการกดคีย์พิเศษ

ตำแหน่งของ VIDEO RAM ในหน่วยความจำ สำหรับอะแดปเตอร์แบบโมโนโครม เริ่มที่แอดแดรส B0000000H ส่วนอะแดปเตอร์แบบอื่นเริ่มที่แอดแดรส B8000000H โดยทั่วไปรูทีนที่ทำงานได้กับอะแดปเตอร์ทุกแบบต้องสามารถตรวจสอบได้ว่าขณะนั้นอะแดปเตอร์แสดงผลเป็นแบบใด ซึ่งทำได้โดยใช้อินเทอร์รัพต์ 16 ฟังก์ชั่นหมายเลข 15 ของ BIOS ซึ่งจะส่งค่าของวิดีโอโหมด (video mode)ขณะนั้นกลับมาให้ โดยรูทีนในบทนี้ทำงานได้ในโหมด 2,3 และ 7 โดยโหมด 2,3 เป็นจอสี และ 7 เป็นของจอโมโนโครม ดังนั้นฟังก์ชั่น pop_up() ในหัวข้อนี้จะต้องสามารถตรวจสอบว่า กำลังทำงานกับอะแดปเตอร์ของจอภาพแบบใด และตั้งค่าของตัวแปรพอยต์เตอร์ที่ชี้ไปยัง VIDEO RAM ให้ถูกต้อง

2.8 การทดลองและผลการทดลอง

2.8.1ส่วนตรวจจับ

เมื่อผู้ที่ต้องการใช้งานนั่งหน้าจอคอมพิวเตอร์ วงจรที่ใช้ในการตรวจจับในส่วนของภาคส่งซึ่งวางไว้หน้าจอคอมพิวเตอร์จะส่งสัญญาณความถี่สูง 10 kHz ออกมา เมื่อมีผู้ใช้งานมานั่ง สัญญาณที่ส่งมาจะสะท้อนกลับไปยังวงจรภาครับ สัญญาณที่วงจรภาครับรับได้มีค่าน้อยมาก วงจรส่วนขยายสัญญาณจะนำสัญญาณที่รับได้ไปขยายเฉพาะสัญญาณความถี่ 10 kHz output ที่ได้จะนำไปเปรียบเทียบกับความถี่ภายในของ Tone decoder ซึ่งถูกเซตไว้ให้มีความถี่เท่ากับ 10 kHz เมื่อความถี่ตรงกัน output ของ Tone decoder จะเปลี่ยนจาก High (+5 V) เป็น Low (0 V) ไปทริกวงจร Monostable ชุดแรกให้สร้างสัญญาณ
พัลส์ที่มีความกว้างของพัลส์เท่ากับ 10 วินาทีแล้วนำสัญญาณนี้ไปทริกวงจร Monostable ชุดที่สองให้สร้างสัญญาณพัลส์ที่มีช่วงกว้างของสัญญาณเป็นไปตามโครงสร้างของการส่งสัญญาณผ่านพอร์ตอนุกรม (RS-232) คือ มีขนาด 10 บิท(start bit 1 bit, stop bit 1 bit,data 8 bit)การที่ไม่ใช้ output ของ

Tone decoder ไปทริกวงจร Monostable ชุดที่สองโดยตรง เพราะต้องการหน่วงเวลาไป 10 วินาทีเพื่อให้แน่ใจว่ามีผู้ต้องการใช้งานจริง โดยผู้ใช้ต้องรอหน้าจอคอมพิวเตอร์เป็นเวลาประมาณ 10 วินาที สัญญาณจึงจะส่งไปทริกให้หน้าจอคอมพิวเตอร์เปิดเมนู แต่ถ้ามีผู้เดินผ่านไปมาโดยไม่ได้ใช้งานซึ่งช่วงเวลาในการเดินผ่านไม่ถึง 10 วินาที สัญญาณจะไม่ถูกส่งไปทริกหน้าจอให้เปิดเมนูให้

2.8.2 **ส่วนโปรแกรม**

เมื่อมีสัญญาณจากส่วนตรวจจับสัญญาณส่งสัญญาณผ่าน RS-232 มายังพอร์ทอนุกรม ส่วนของโปรแกรมการPolling ซึ่งทำหน้าที่วนลูปตรวจจับสัญญาณอยู่ตลอดเวลา จะส่งการทำงานให้กับส่วนแสดงผลทางหน้าจอต่อไป เพื่อเปิดหน้าจอแสดงรายการ(menu)หัวข้อต่างๆที่กำหนดไว้ โดยใช้รูปแบบการทำงานของpop up menu

2.9 **สรุปและวิจารณ์ผลการทดลอง**

ระยะการตรวจจับที่สามารถตรวจจับได้มีระยะห่างจากคอมพิวเตอร์ประมาณ 60 เซนติเมตร ซึ่งอยู่ในขั้นที่ใช้งานได้ การที่จะเพิ่มระยะการตรวจจับได้นั้น กระแสที่ขับตัวไดโอดต้องมีค่ามาก จึงจะทำให้มีความเข้มของแสงมากขึ้น ทำให้ส่งได้ไกลขึ้น แต่ต้องจำกัดค่ากระแสที่ใช้งานด้วย สำหรับในส่วนตรวจจับนี้ ไม่สามารถใช้กระแสขับตัวไดโอดที่มีค่ามาก ๆได้ เนื่องจากต้องส่งสัญญาณความถี่ 10 KHz ตลอดเวลาถ้าใช้กระแสค่ามาก จะทำให้ Power transistor และ INFRARED LED ร้อนทำให้เสียหายได้

ในส่วนของโปรแกรมคอมพิวเตอร์ให้หลักการทำงานแบบ Polling ซึ่งทำให้เกิดความล่าช้าในการทำงานและในการทำงานแต่ละครั้งต้องมีการ Run โปรแกรมอยู่ตลอดเวลา ดังนั้น โปรแกรมคอมพิวเตอร์จะยังคงทำงานอยู่ตลอดแม้ว่าจะไม่มีผู้มาใช้งานก็ตามทำให้เกิดการสูญเปล่า จะเห็นว่าระบบการทำงานของโครงงานนี้ ยังไม่สมบูรณ์เท่าที่ควร เนื่องจากเป็นโครงงานต้นแบบ ฉะนั้นจึงต้องมีการปรับปรุงประสิทธิภาพการทำงานให้ดียิ่งขึ้นต่อไป

บทที่ 3
การพัฒนาโครงงาน

เพื่อแก้ไขปัญหาของวงจรภายนอกในส่วนตรวจจับที่เกิดขึ้นจากการทำโครงงานภาคแรก คือ ความร้อนของอุปกรณ์อิเล็กทรอนิกส์ที่เกิดขึ้นเนื่องจากต้องส่งสัญญาณออกมาตรวจจับตลอดเวลา ซึ่ง อาจทำให้อุปกรณ์เสียหายได้ และในส่วนของโปรแกรม คือ ความล่าช้าในการทำงานของส่วนตรวจสอบสัญญาณแบบ polling และพบว่าไม่สามารถนำโปรแกรมนี้มาใช้ในการทำงานขั้นตอนต่อไปได้ จึงได้มีการแก้ไขและพัฒนาหลักการทำงานใหม่ วงจรภายนอกของส่วนตรวจจับและในส่วนของโปรแกรมที่ใช้จึงเปลี่ยนไป

3.1 การพัฒนาโครงงาน

เนื่องจากปัญหาที่เกิดขึ้นกับโครงงานดังที่ได้กล่าวมาแล้ว และได้มีการพัฒนาทั้งส่วนตรวจจับและส่วนแสดงผลข้อมูล ซึ่งจะขอกล่าวรายละเอียดในแต่ละส่วนดังต่อไปนี้

3.2 ส่วนตรวจจับ(SENSOR)

ในส่วนนี้มีการพัฒนาหลักการทำงานซึ่งแตกต่างไปจากเดิมที่วงจรจะสร้างสัญญาณความถี่ค่าหนึ่ง ส่งด้วยแสงอินฟาเรดไปตรวจจับผู้ใช้คอมพิวเตอร์ แล้วไปทริกให้ส่วนแสดงผลข้อมูลทำงาน คือ ส่วนวงจรจะทำหน้าที่เป็นเพียงตัวส่งและรับสัญญาณด้วยแสงอินฟาเรดเท่านั้น โดยสัญญาณที่ส่งไปตรวจจับผู้ใช้คอมพิวเตอร์จะส่งมาจากคอมพิวเตอร์ซึ่งควบคุมโดยโปรแกรม และเว้นช่วงการตรวจสอบเป็นระยะเวลาสั้นๆ เพื่อลดปัญหาเนื่องจากความร้อนที่เกิดขึ้นกับอุปกรณ์อิเลกทรอนิกส์เนื่องจากการส่งสัญญาณตลอดเวลาและในส่วนของวงจรได้ใช้อุปกรณ์ส่ง-รับ อินฟาเรดที่มีประสิทธิภาพดีขึ้น คือ ส่งได้ในระยะไกลขึ้น

3.3 ส่วนแสดงผล(DISPLAY)

ส่วนนี้มีการพัฒนาจากเดิมที่แสดงเมนูแบบง่าย เป็นรูปแบบที่ดีขึ้น

ซึ่งจะกล่าวรายละเอียดการทำงานของแต่ละส่วนในบทต่อไป

บทที่ 4
ส่วนตรวจจับ(SENSOR) ที่พัฒนาขึ้นใหม่

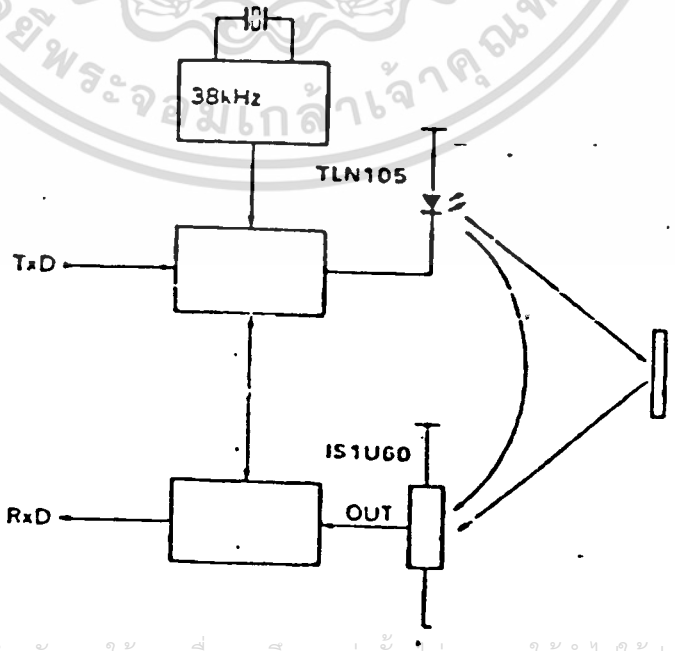
4.1 หลักการทำงาน

ส่วนตรวจจับจะใช้หลักการสะท้อนกลับของแสงอินฟาเรดเมื่อกระทบกับวัตถุ คือ มีวงจรรับสัญญาณจากพอร์ตอนุกรม ซึ่งสัญญาณที่ส่งมาจากคอมพิวเตอร์จะกำหนดโดยส่วนของโปรแกรม โดยโปรแกรมจะควบคุมให้ส่งสัญญาณเป็นรหัส ASCII ใดๆ รหัสหนึ่งเป็นระยะ ๆ โดยมีช่วงเวลาห่างเท่า ๆ กัน ซึ่งโครงงานนี้กำหนดให้ส่งสัญญาณออกมาตรวจสอบทุก 5 วินาทีทำให้ไม่ต้องส่งสัญญาณ ตรวจสอบตลอดเวลา เมื่อสัญญาณรหัส ASCII ส่งเข้ามาในวงจร จะถูกมอดดูเลทด้วยสัญญาณพาหะความถี่สูง และส่งออกไปภายนอกโดยผ่านโฟโต้ไดโอด เมื่อสัญญาณนี้กระทบกับสิ่งกีดขวางจะสะท้อนกลับไปยังแหล่งส่วนวงจรซึ่งรอรับสัญญาณอยู่ แล้วส่งสัญญาณที่รับได้ผ่านพอร์ตอนุกรมเพื่อนำไปเปรียบเทียบกับสัญญาณที่ส่งออกไปว่าเท่ากันหรือไม่ ถ้าเท่ากันก็จะทำส่วนต่อไปคือเปิดหน้าจอเมนู แต่ถ้าไม่เท่ากัน ก็จะทำการตรวจสอบต่อไปจนกว่าสัญญาณที่ส่งออกไปกับที่รับได้เท่ากัน จะเห็นได้ว่าสัญญาณที่ใช้ส่งออกมาตรวจสอบเป็นข้อมูลรหัส ASCII ที่ต้องใช้เช่นนี้ มีเหตุผลคือ จะทำให้ง่ายในการเปรียบเทียบสัญญาณในส่วนของโปรแกรม เพราะสัญญาณที่รับเข้ามาจะอยู่ในรูปของรหัส ASCII ซึ่งคอมพิวเตอร์สามารถรับรู้ได้ทันที

จากที่กล่าวมาสามารถแบ่งการทำงานของส่วนตรวจจับได้เป็น 2 ส่วนคือ

-ส่วนวงจร

-ส่วนโปรแกรมควบคุม



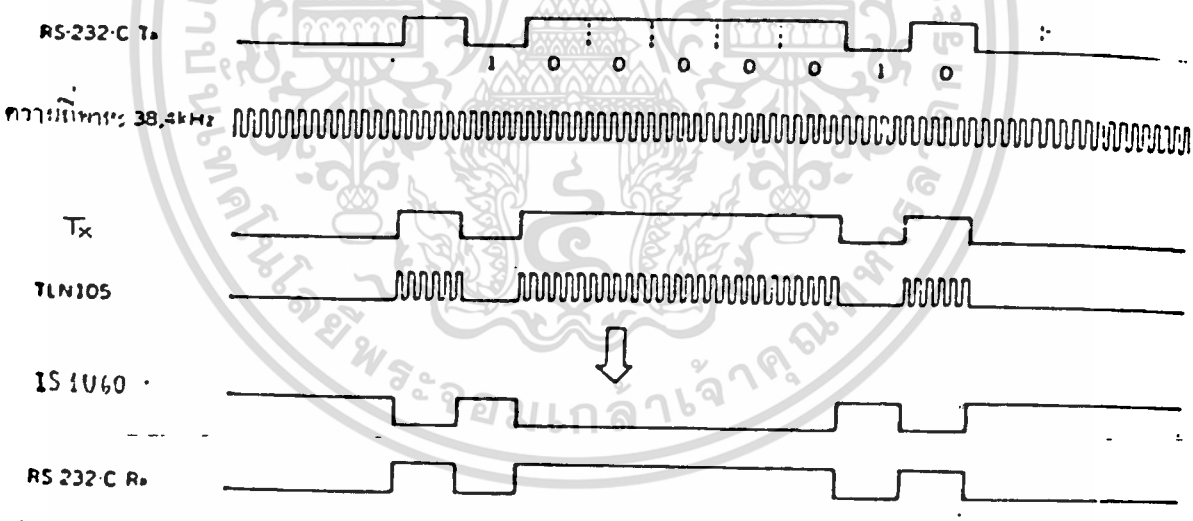
รูปที่ 4.1 แผนภาพแสดงหลักการทำงานของส่วนตรวจจับ

4.2 ส่วนวงจร

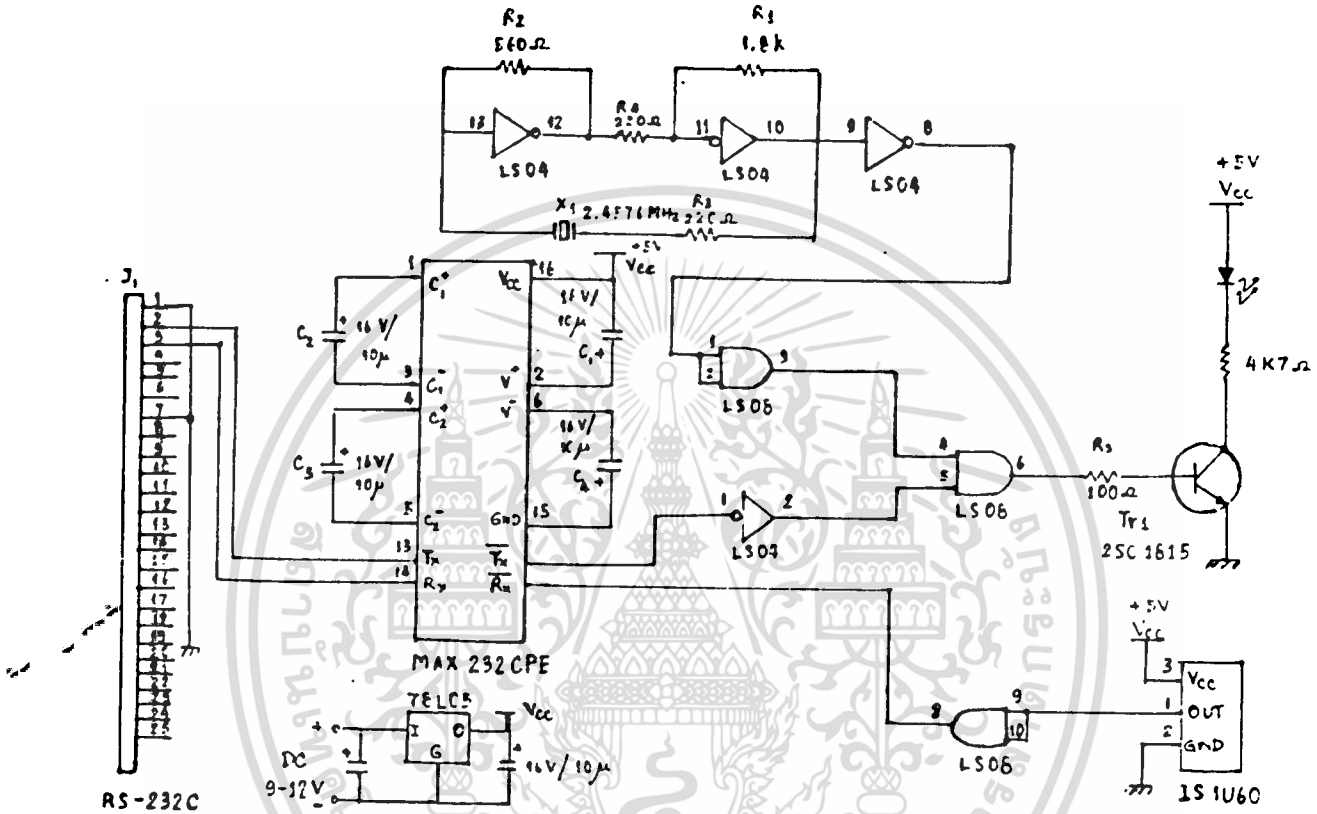
การทำงานของส่วนวงจร คือ รับสัญญาณรหัส ASCII ที่ส่งมาจากคอมพิวเตอร์เป็นช่วง ๆซึ่งกำหนดโดยส่วนโปรแกรม แล้วมอดดูเลทสัญญาณที่รับมากับสัญญาณความถี่สูง สำหรับโครงงานนี้จะใช้สัญญาณความถี่ 38.4 kHz โดยใช้คริสตอลเป็นตัวออสซิลเลทความถี่ สัญญาณมอดดูเลทที่ได้จะส่งผ่านโฟโต้ไดโอดออกไป เมื่อมีคนมาอยู่หน้าจอคอมพิวเตอร์ สัญญาณที่ส่งผ่านแสงอินฟาเรดจะสะท้อนกลับไปยังวงจรซึ่งมีส่วนรับสัญญาณอยู่ สัญญาณที่รับได้จะถูกตัดสัญญาณความถี่สูงทิ้งไป ดังนั้นสัญญาณที่จะส่งต่อไปจึงมีแต่สัญญาณรหัส ASCII เท่านั้น เมื่อสัญญาณรหัส ASCII ที่ได้ส่งผ่านพอร์ต RS-232 ไปยังคอม

พิวเตอร์ซึ่งมีโปรแกรมรอรับสัญญาณ เพื่อทำการเปรียบเทียบต่อไป

วงจรที่ได้เป็นดังรูป



รูปที่4.2 Timing Diagram ของวงจร



รูปที่ 4.3 วงจรที่ใช้งาน

4.3 **ส่วนโปรแกรมควบคุม**

การทำงานของส่วนโปรแกรม คือ ส่งสัญญาณเป็นรหัส ASCII จำนวน 1 ไบท์ผ่านส่วนวงจรเพื่อไปตรวจสอบว่ามีผู้ที่ต้องการใช้งานคอมพิวเตอร์หรือไม่ ซึ่งโครงงานนี้กำหนดให้ส่งเป็นคาร์แรคเตอร์ "C" ซึ่งมีรหัส ASCII เป็น 43h ผ่านพอร์ตอนุกรมของ COM1 ซึ่งมี Address เป็น 03F โดยเว้นช่วงห่างในการส่งแต่ละครั้ง 5 วินาที เพื่อประหยัดพลังงาน และลดอุณหภูมิในอุปกรณ์อิเลคทรอนิกส์ทำให้อุปกรณ์ไม่เสียหายและใช้งานได้นานขึ้น และที่สำคัญคือ โครงงานนี้จะเลิกการทำงานโดยปิดหน้าจอเองโดยอัตโนมัติเมื่อผู้ใช้งานลุกออกไป นั่นหมายถึงว่าสัญญาณรหัส ASCII ที่ส่งออกไปกับที่รับเข้ามาไม่เท่ากัน เพราะไม่มีสิ่งกีดขวางที่ทำให้สัญญาณที่ส่งสะท้อนกลับได้ ซึ่งถ้าจะให้ผลการทำงานของโครงงานเป็นแบบนี้ได้ จะเกิดปัญหาที่ว่าในขณะที่ผู้ใช้กำลังใช้งานคอมพิวเตอร์อยู่ จะตรวจสอบได้อย่างไร ว่าผู้ใช้ยังนั่งอยู่หน้าจอคอมพิวเตอร์ และถ้าผู้ใช้เลิกการใช้งานในช่วงใดช่วงหนึ่งทันที จะรู้ได้อย่างไรว่าผู้ใช้ได้ลุกออกไปแล้ว เพื่อแก้ปัญหาที่กล่าวมาทั้งหมด จำเป็นต้องทำการตรวจสอบตลอดเวลาทั้งที่ไม่มีผู้ใช้คอมพิวเตอร์มานั่งหน้าจอ และ มีผู้ใช้ใช้งานอยู่ การที่จะทำเช่นนี้ได้ โปรแกรมที่ใช้ในส่วนตรวจจับจะต้องเป็นโปรแกรมเรซิเดนต์คือเรียกใช้ได้ทันที โดยไม่ต้องมีการโหลดเข้ามาใหม่หลังจากโหลดครั้งแรก แต่จากการเขียนและทดสอบโปรแกรมเรสซิเดนต์กับโปรแกรมในส่วนอื่นๆ พบว่ามีปัญหาในการทำงาน เนื่องจากโปรแกรมเรสซิเดนต์เป็นโปรแกรมที่ฝังตัวอยู่ในหน่วยความจำ ดังนั้น หากเขียนโปรแกรมผิดพลาดจะมีผลต่อระบบการทำงานของเครื่องคอมพิวเตอร์ อาจทำให้เกิดการ crash ได้ แต่ถ้ายังต้องการใช้โปรแกรมเรสซิเดนต์ในการตรวจสอบสัญญาณแล้ว การเขียนโปรแกรมจะมีความยุ่งยากและซับซ้อนมากยิ่งขึ้น ดังนั้นจึงได้นำวิธีการทดสอบสัญญาณวิธีอื่นมาใช้ ซึ่งง่ายและซับซ้อนน้อยกว่ามาใช้ ดังจะได้กล่าวต่อไป สำหรับการเรียกใช้งานโปรแกรมตรวจจับนี้จะเรียกใช้ทุก 5 วินาที ซึ่งผู้ใช้ที่กำลังใช้งานคอมพิวเตอร์อยู่จะไม่รู้สึกว่าคอมพิวเตอร์ทำงานในส่วนตรวจจับนี้ด้วย

การทำงานของโปรแกรมในส่วนตรวจจับแสดงได้ดัง FLOW CHART ดังนี้

บทที่ 5
การรับและส่งข้อมูลผ่านพอร์ตอนุกรม

ในโครงงานนี้การสื่อสารระหว่างคอมพิวเตอร์และระบบตรวจจับจะผ่านทางพอร์ตอนุกรม จึงนำเสนอลักษณะทั่วๆไป ของพอร์ตอนุกรมต่อไปนี้

5.1 ข้อตกลง(PROTOCAL)ในการสื่อสารข้อมูล

พอร์ตอนุกรมนั้นสามารถปรับเปลี่ยนได้ ไม่ว่าความเร็วของข้อมูลจะส่งผ่านหรือรับเข้าเร็วเท่าใด รวมทั้งวิธีที่ข้อมูลถูกเข้ารหัสในสตรีมของบิต สำหรับพอร์ตอนุกรมสองพอร์ตในการสื่อสารซึ่งกันและกัน การกำหนดความเร็วและเข้ารหัสข้อมูล จะเรียกรวมๆว่า พารามิเตอร์การสื่อสาร (communication parameter) ซึ่งจะต้องเหมือนกันสำหรับพอร์ตทั้งสอง

ความเร็วในการส่งข้อมูลของพอร์ตอนุกรมกำหนดเป็นอัตราบอด(baud rate) อัตราบอดจะมีค่าหยาบๆ เท่ากับจำนวนของบิตที่จะส่งต่อวินาที พอร์ตอนุกรมบนพีซีทั้งหมดสามารถกำหนดได้ที่ 110,300,1200,2400,4800,9600 บอด ในบางระบบสามารถกำหนดได้ถึง 19,200 บอด

พารามิเตอร์ของการสื่อสารอย่างอื่นๆ มีความยาวของเวิร์ด (word lenght)การตรวจสอบพาริตี้ (parity checking)และจำนวนของบิตจบ(stop bits)

ในการส่งข้อมูลผ่านพอร์ตอนุกรมนั้น ข้อมูลแต่ละไบต์จะประกอบด้วย

1 บิตเริ่มต้น(start bit) 1 บิท

2. บิตข้อมูล(data bit)7 หรือ 8 บิท

3. พาริตี้บิต(parity bit) (จะมีหรือไม่มีก็ได้)

4. บิตสิ้นสุด(stop bit) 1 หรือ 2 บิต

สถานะของสายส่งในขณะที่ไม่มีข้อมูลจะมีสถานะเป็นสูง(สถานะทางดิจิตอล มี 2 สถานะ คือ สูง(high) และต่ำ (low)) ข้อมูลบิทใดมีค่าเป็น 0 จะทำให้สายส่งมีสถานะต่ำ ข้อมูลบิทใดมีค่า 1 ก็จะทำให้สายส่งมีสถานะสูงอยู่เช่นเดิม บิตเริ่มต้นใช้สำหรับบอกจุดเริ่มต้นของไบต์ของข้อมูล โดยการทำให้สถานะของสายส่งมีค่าต่ำ เป็นเวลา 1 รอบ จากนั้นจะเป็นบิตของข้อมูล ตามด้วยพาริตี้บิท ซึ่งจะมีหรือไม่ก็ได้ สุดท้ายคือ บิตสิ้นสุด ซึ่งจะมี1 หรือ 2 บิตก็ได้ ขึ้นกับว่าจะใช้เท่าใด

พาริตี้บิต ถ้าหากมีในไบต์ข้อมูล ก็จะทำหน้าที่ตรวจเช็คความผิดพลาดของข้อมูล พาริตี้มีค่า 2 อย่าง คือ เป็นคู่ หรือ คี่ (even or odd) ถ้าเป็นคู่ หมายความว่า เมื่อรวมพาริตี้บิตแล้ว จำนวนของบิตข้อมูลที่มีค่า 1 จะเป็นจำนวนคู่ และถ้าพาริตี้บิตเป็นคี่ หมายความว่า เมื่อรวมพาริตี้บิตแล้วจำนวนของบิตข้อมูลที่มีค่าเป็น 1 จะเป็นจำนวนคี่

อัตราการส่งข้อมูลมีหน่วยเป็นbaud(bit per second)ค่า baud rate ที่ต่ำที่สุดที่มีใช้กันคือ 300 baud ซึ่งจะใช้กับโมเด็มรุ่นเก่า(โมเด็มรุ่นใหม่มักจะใช้ 1200-2400) ส่วนเครื่องคอมพิวเตอร์ระดับ IBM PC สามารถใช้ค่า baud rate ได้สูงถึง 9600 baud

5.2 การใช้งานพอร์ตอนุกรมผ่าน BIOS

การใช้งานพอร์ตอนุกรมสามารถทำได้ 3 วิธี คือ ผ่านทางดอส ผ่านทาง BIOS และสุดท้ายคือการเขียนโปรแกรมควบคุมพอร์ตอนุกรมโดยตรง

การเรียกใช้พอร์ตอนุกรมผ่านทางดอสนั้น ไม่เหมาะสมเท่าใดนัก เพราะว่าดอสไม่มีวิธีที่จะตรวจดูสถานะการรับส่งและ ตัวพอร์ตอนุกรมว่า การรับส่งข้อมูลถูกต้องหรือไม่เพียงใด

การเขียนโปรแกรมควบคุมการทำงานของพอร์ตอนุกรมโดยตรงนั้นคงไม่จำเป็น เพราะมีวิธีที่ดีกว่าและง่ายกว่าคือ การเรียกใช้ผ่าน BIOS อินเตอร์รัพต์หมายเลข 14

5.3 การกำหนดพอร์ตอนุกรม I/O และ IRQ

แต่ละพอร์ตอนุกรมจะติดต่อสื่อสารกับซีพียูโดยทางพอร์ต I/O และฮาร์ดแวร์อินเตอร์รัพต์ แต่ละพอร์ตอนุกรมจะถูกกำหนดไว้แปดพอร์ต I/O (แต่จะใช้จริงเพียงเจ็ดเท่านั้น)และหนึ่งสาย IRQ ซึ่งการกำหนดพอร์ต I/O และ IRQ ของ COM1 ถึง COM4 มีดังต่อไปนี้

ตารางที่5.1 การกำหนดพอร์ต I/O และ IRQ สำหรับ COM1 ถึง COM2

พอร์ตอนุกรม	อินเทอร์รัปต์	แอดเดรสของ I/O(เลขฐาน 16)
COM1	IRQ4	3F8H-3FFH
COM2	IRQ3	3F8H-2FFH
COM3	IRQ4	3E8H-3EFH
COM4	IRQ3	2E8H-2EFH

ระหว่างการเริ่มต้นของระบบ ไบออสจะหาพอร์ตอนุกรมที่ติดตั้งไว้ และวางแอดเดรสฐาน I/O ในตาราง 8 ไบต์ ในแรมระดับต่ำของพื้นที่ข้อมูลไบออส ถ้าไม่มีพอร์ต(ไม่ได้ติดตั้ง)รายการของพอร์ตในตารางจะกำหนดที่ 0 โปรแกรมสามารถตรวจสอบรายการต่างๆ เพื่อดูว่ามีพอร์ตอนุกรมใดอยู่บ้าง

บางโปรแกรมที่ใช้พอร์ตอนุกรม เช่น ไดรเวอร์ของเมาส์แบบอนุกรม จะทำให้รายการตารางของพอร์ตที่ใช้อยู่ให้เป็นศูนย์ ก็เพื่อว่าทำให้โปรแกรมอื่นจะมีโอกาสใช้พอร์ตเดียวกันนี้น้อยลง

ตารางที่ 5.2 ตำแหน่งของแอดเดรสฐาน สำหรับพอร์ตอนุกรมที่ติดตั้งไว้

พอร์ตอนุกรม	ตำแหน่งของแอดเดรสฐานของ I/O
COM1	0040:0000H
COM2	0040:0002H
COM3	0040:0004H
COM4	0040:0006H

บันทึกไว้ว่าไบออสสำรองพื้นที่ว่างสำหรับแอดเดรสของพอร์ตอนุกรมทั้งสี่ แม้ว่าเครื่องพีซีจะสนับสนุนเพียงสอง "อย่างเป็นทางการ" เป็นการแนะนำว่าผู้ออกแบบพีซีแรกเริ่มวางแผนที่จะสนับสนุนสี่พอร์ตแต่พบว่าจำกัดเหลือสอง เพราะว่ามีสาย IRQ ที่ไม่ใช้เพียงสองเท่านั้นบนบัสของพีซีรุ่นแรก จากตารางการกำหนดพอร์ต I/O และ IRQ สำหรับ COM1 และ COM2 จะได้ว่าพอร์ต "ไม่เป็นทางการ" COM3 และ COM4 ใช้สาย IRQ เดียวกันกับ COM1 และ COM2 อันสามารถสร้างปัญหาต่างๆ ได้ เราจะอธิบายปัญหาเหล่านี้ในหัวข้อต่อไป

5.4 อินเตอร์รัปต์ของพอร์ตอนุกรม

เทคนิคการทำโปรแกรมวิธีหนึ่งเพื่ออ่านข้อมูลจากพอร์ตอนุกรม คือ การใช้ polling ด้วยวิธีนี้โปรแกรมจะปฏิบัติการวนตรวจสอบพอร์ตซ้ำแล้วซ้ำเล่า เพื่อดูว่ามีการรับอักขระแล้วหรือไม่ ถ้าได้รับอักขระนั้น จะถูกอ่านในหน่วยความจำ การโปรแกรม polling ทำได้ง่าย และทำงานได้ดีที่ความเร็วในการสื่อสารค่อนข้างไม่มาก อย่างไรก็ตามที่ความเร็วมากกว่า 1200 บอด ข้อมูลอาจจะหายไปเพราะว่าพอร์ตอนุกรมรับอักขระได้เร็วกว่าที่โปรแกรมป้อนให้

เพื่อให้ได้ประสิทธิภาพสูงสุด และป้องกันการสูญหายของข้อมูล เทคนิคที่นิยมใช้กันมากกว่า คือ การใช้การโปรแกรมขับผ่านอินเทอร์รัปต์ของพีซีแต่ละพอร์ตอนุกรมจะต่อเข้ากับสายฮาร์ดแวร์อินเทอร์รัปต์ของพีซี อินเทอร์รัปต์จะให้สัญญาณทุกครั้งที่พอร์ตรับอักขระ ด้วยการโปรแกรมการขับผ่านอินเทอร์รัปต์ โปรแกรมจะรวมรูทีนการบริการอินเทอร์รัปต์ ซึ่งแอดเดรสของการกระทำนี้จะถูกวางไว้ที่อินเตอร์รัปต์เวกเตอร์ สำหรับพอร์ตอนุกรมที่ต้องการ โค้ดบริการในรูทีนจะมีเพียงหน้าที่เดียว คือ การ

พาอักขระที่รออยู่จากพอร์ตอนุกรม และวางตำแหน่งในบัฟเฟอร์หน่วยความจำ ที่โค้ดหลักของโปรแกรมสามารถเข้าถึงได้ โปรแกรมการติดต่อสื่อสารทั้งหมดในเชิงพาณิชย์เป็นแบบการโปรแกรมขับผ่านอินเทอร์รัปต์

จากที่กล่าวมา จะเห็นว่ามีเพียงสองสายฮาร์ดแวร์อินเทอร์รัปต์เท่านั้น คือ IRQ3 และ IRQ4 สำหรับพอร์ตอนุกรม ดังนั้นถ้าติดตั้ง COM3 และ COM4 ก็ต้องใช้สาย IRQ เหล่านี้ร่วมกันกับ COM1 และ COM2

เมื่อมีการติดตั้งพอร์ตอนุกรมมากกว่าสองในเครื่องพีซี(อื่นๆ นอกจากรุ่นพีเอสทู) จะเกิดปัญหาขึ้น เพราะว่าสายอินเทอร์รัปต์บนบัสของพีซีไม่ได้ออกแบบมาให้ใช้ร่วมกัน ถ้าสองการ์ดอนุกรมพยายามที่จะใช้สาย IRQ เดียวกัน เป็นไปได้ที่จะมีการพลาดจากอินเทอร์รัปต์ หรือในบางกรณีก็ทำความเสียหายให้แก่ฮาร์ดแวร์

เหตุผลที่ว่าสามารถสนับสนุน COM3 และ COM4 ได้ทั้งหมด ก็คือว่า การใช้พอร์ตอนุกรมจำนวนมากไม่ได้ใช้งานอินเทอร์รัปต์ ตัวอย่างเช่น การใช้ COM1 เพื่อส่งข้อมูลสู่เครื่องพิมพ์ไม่เกี่ยวข้องกับอินเทอร์รัปต์ ดังนั้นจึงเป็นการยอมตกลงได้สำหรับโปรแกรมสื่อสารแบบขับผ่านอินเทอร์รัปต์ ที่จะใช้ COM3 อย่างไรก็ตาม ถ้าคุณต้องการใช้ COM3 และ COM4 จริงๆ จงแน่ใจว่าไม่มีข้อขัดแย้งของ IRQ จึงจะเป็นการดีที่สุด

5.5 การโปรแกรมพอร์ตอนุกรม

บริการพอร์ตอนุกรมของเอ็มเอสดอส และไบออส บางครั้งก็มีประโยชน์ สำหรับงานที่ค่อนข้างง่าย แต่เพราะว่าบริการเหล่านี้ไม่ใช่แบบขับผ่านอินเทอร์รัปต์ การสนับสนุนจึงมีขีดจำกัดในเรื่องความเร็วในการสื่อสาร

เอ็มเอสดอสมีบริการต่างๆ ให้ผ่านทางอินเทอร์รัปต์ 21H สำหรับอ่านข้อมูลจากพอร์ตอนุกรมและเขียนข้อมูลเข้าไปด้วย อย่างไรก็ดีไม่มีบริการของเอ็มเอสดอสสำหรับจัดตั้งพอร์ตอนุกรม ดังนั้นคุณต้องใช้ภาษาระดับสูง หรือบริการไบออส เพื่อเซตอัตราบอด และพารามิเตอร์ สื่อสารอื่นๆ

การเขียนหรืออ่านบล็อกต่างๆ ของข้อมูล คุณสามารถใช้บริการฐานข้อมูลของแฮนเดินในฟังก์ชัน 3 FH และ 40H ของอินเทอร์รัปต์ 21H เอ็มเอสดอสจะร่วมกับแฮนเดิลสำหรับตัวเอง

บริการพอร์ตอนุกรมของไบออสจะปรับเปลี่ยนได้มากกว่าบริการของเอ็มเอสดอส การใช้บริการเหล่านี้บนระบบที่ค่อนข้างเร็ว จะได้รับการสื่อสารที่ค่อนข้างเชื่อถือได้ ที่ความเร็วสูงได้ถึง 1200 บอด บริการพอร์ตอนุกรมของไบออสสรุปได้ดังนี้.

ตารางที่ 5.3 การบริการพอร์ตอนุกรมของไบออส

บริการไบออส	คำอธิบาย
อินเทอร์รัปต์ 14H ฟังก์ชัน 00H	กำหนดค่าเริ่มต้นและคอนฟิกเกอร์พอร์ตอนุกรม
อินเทอร์รัปต์ 14H ฟังก์ชัน 01H	เขียนหนึ่งอักขระสู่พอร์ตอนุกรม
อินเทอร์รัปต์ 14H ฟังก์ชัน 02H	อ่านหนึ่งอักขระจากพอร์ตอนุกรม
อินเทอร์รัปต์ 14H ฟังก์ชัน 03H	รับรู้สถานภาพของพอร์ตอนุกรม
อินเทอร์รัปต์ 14H ฟังก์ชัน 04H (ในพีเอสทูเท่านั้น)	แสดงการกำหนดค่าเริ่มต้นส่วนขยายของพอร์ตอนุกรม
อินเทอร์รัปต์ 14H ฟังก์ชัน 05H (ในพีเอสทูเท่านั้น)	แสดงการควบคุมส่วนขยายของพอร์ตอนุกรม

บทที่ 6
ส่วนติดต่อกับผู้ใช้ทางหน้าจอคอมพิวเตอร์

6.1 หลักการทำงาน

การทำงานของส่วนแสดงผล คือ เมื่อส่วนตรวจจับสัญญาณตรวจสอบได้ว่า มีผู้ต้องการใช้คอมพิวเตอร์ ส่วนแสดงผลจะแสดงเมนูบนหน้าจอ ซึ่งในเมนูจะแสดงรายการหัวข้อเรื่องให้เลือก โดยผู้ใช้จะเลือกได้โดยการกดคีย์ เมื่อผู้ใช้เลือกหัวข้อเรื่องแล้ว ส่วนแสดงผลข้อมูล จะแสดงข้อมูลของหัวข้อเรื่องนั้นทางจอภาพ การทำงานในส่วนนี้จะควบคุมด้วยโปรแกรมทั้งหมดซึ่งประกอบไปด้วยโปรแกรมสร้างเมนู โปรแกรมจัดการไฟล์ข้อมูล ข้อมูลที่แสดงอาจจะเป็นภาพหรือ ข้อความก็ได้ ซึ่งจะขอกล่าวรายละเอียดในหลักการของการเขียนโปรแกรมของส่วนแสดงผลข้อมูล
โดยแยกออกเป็น 2 ส่วนดังต่อไปนี้

1.โปรแกรมเมนู

2.โปรแกรมแสดงข้อมูล

6.2 โปรแกรมเมนู

เมนูที่ใช้เป็นแบบ pop up คือ เมนูที่แสดงขึ้นมาบนจอโดยทับหน้าจอเดิมเพียงบางส่วน เมื่อเลิกใช้เมนู หน้าจอส่วนที่ถูกเมนูทับจะกลับคืนมาเหมือนเดิม การเลือกตัวเลือกบนเมนูแบบนี้มี 2 วิธี คือ

1. โดยการกดคีย์พิเศษ(hot key) ซึ่งก็คือตัวอักษรหรือตัวเลขที่มีความสัมพันธ์กับตัวเลือกบนเมนู
2. โดยการใช้คีย์ลูกศร(arrow key) เลื่อนแถบสว่างให้ทับตัวเลือกที่ต้องการ แล้วกดปุ่ม Enter

6.3 การสร้างเมนูแบบ pop up

การสร้างเมนูแบบ pop up นั้นต้องใช้ข้อมูลหลายส่วน ส่วนแรกคือ รายชื่อของตัวเลือกที่แสดงในเมนู ซึ่งเป็นสตริง การส่งข้อมูลนี้ให้ฟังก์ชันนำไปใช้งานง่ายที่สุด คือ นำสตริงเหล่านี้ใส่อาร์เรย์ 2 มิติ ส่งเฉพาะพอยต์เตอร์ที่ชี้ไปยังอาร์เรย์นี้ให้กับฟังก์ชัน การเลือกตัวเลือกของเมนูทำได้ 2 วิธี วิธีแรกคือเลื่อนแถบสว่างไปยังตัวเลือกที่ต้องการแล้วกด Enter วิธีที่สองคือกดคีย์พิเศษ(hot key) ซึ่งคีย์เหล่านี้ถูกกำหนดไว้ว่าเมื่อถูกกดเท่ากับเลือกตัวเลือกอะไร การที่ฟังก์ชัน pop up รู้ข้อมูลของคีย์พิเศษได้ ต้องมีการส่งรายชื่อของคีย์พิเศษเหล่านั้นไปให้ วิธีที่ดีที่สุด คือ การส่งชุดของสตริงที่ประกอบไปด้วยตัวอักษรที่เป็นคีย์พิเศษ และเรียงลำดับเช่นเดียวกับตัวเลือกของเมนู

นอกจากนี้ฟังก์ชัน pop up ต้องรู้จำนวนของตัวเลือกที่อยู่ในเมนูตำแหน่ง x,y ที่จะแสดงเมนู และต้องรู้ว่าเมนูที่สร้างต้องการตีกรอบด้วยหรือไม่ จากข้อมูลที่กล่าวมานี้ เขียนเป็นลักษณะของการกำหนดตัวแปร(declare) ในภาษาซีได้ดังนี้

```
/* ฟังก์ชันแสดงเมนูแบบ pop up และส่งค่าตัวเลือกกลับ */
int popup(menu,keys,count,x,y,border)
char *menu[];  /* menu text*/
char *keys;    /*คีย์พิเศษ*/
int count;     /*จำนวนของตัวเลือก*/
int x,y;       /*ตำแหน่ง x,y ที่มุมบนขวาสุด */
int border;    /*=0 ไม่แสดงขอบ*/
```

ส่วนหน้าที่ของฟังก์ชัน pop up() มีดังนี้

1. เก็บหน้าจอ
2. ตีกรอบ
3. แสดงเมนู
4. ตรวจสอบการกดคีย์และแสดงผล
5. เมื่อเลิกใช้เมนู คืนหน้าจอที่เก็บไว้

6.4 การแสดงเมนู

หลักสำคัญของการแสดงเมนูคือ จะส่งพอยน์เตอร์ที่ชี้ไปยังอาร์เรย์ของพอยน์เตอร์ของสตริงให้กับฟังก์ชัน pop up() ส่วนวิธีการแสดงสตริงนั้นใช้อาร์เรย์ของพอยน์เตอร์ ซึ่งพอยน์เตอร์แต่ละตัวนี้ชี้ไปยังสตริงของตัวเลือก ฟังก์ชัน display_menu() ที่แสดงต่อไปนี้ทำหน้าที่แสดงตัวเลือกแต่ละตัวของเมนูลงบนจอภาพ

```
/* ฟังก์ชันแสดงเมนู ณ ตำแหน่งที่ต้องการ*/
void display_menu(menu,x,y,count)
char *menu[]
int x,y,count;

    register int i;
    for(i=0;i<count;i++,x++)

        goto_xy(x,y);
        printf(menu[i]);
```

การสร้างอาร์เรย์ 2 มิติ เพื่อเก็บสตริงของตัวเลือกของเมนูไว้ใช้วิธีการสร้างตัวแปรดังนี้

```
char *<ชื่อของข้อมูล>[]=
"ตัวเลือกที่ 1",
"ตัวเลือกที่ 2",
"ตัวเลือกที่ N",
;
```

การกำหนดตัวแปรแบบนี้คอมไพเลอร์จะเก็บสตริงไว้ใน runtime string table ให้เอง ตัวแปรพอยน์เตอร์แต่ละตัวจะชี้ไปยังอักษรตัวแรกของสตริง เช่น การกำหนดตัวแปรต่อไปนี้ ตัวแปรพอยน์เตอร์ชื่อ fruit ชี้ไปยัง A ใน Apple

```
char *fruit[]=
"Apple",
"Orange",
"Pear",
"Graph",
"Raspberry"
"Straberry",
```

6.5 การตีกรอบ

ฟังก์ชันนี้ทำหน้าที่สร้างกรอบให้กับเมนู โดยส่งตำแหน่งมุมบนด้านซ้ายสุดและมุมล่างขวาสุดไปให้ การวาดกรอบใช้รหัสแอสกี ซึ่งเป็นรูปส่วนต่างๆ ของกรอบซึ่งอยู่ในชุดอักษรมารตรฐานบนเครื่อง IBM PC หรือคอมพาติเบิล

```
void draw_border(startx,starty,endx,endy)
int startx,starty,endx,endy;
  register int i;
   for(i=startx+1; i<endx; i++)
      goto_xy(i,starty);
      putchar(179);
      goto_xy(i,endy);
      putchar(179);
    for(i=starty+1;i<endy;i++)
    goto_xy(startx,i);
      putchar(196);
      goto_xy(endx,i);
      putchar(196);
goto_xy(startx,starty);putchar(218);
goto_xy(startx,endy);putchar(191);
goto_xy(endx,starty);putchar(192);
goto_xy(endx,endy);putchar(217);
```

6.6 การตอบสนองต่ออินพุทของผู้ใช้

จากที่กล่าวมาแล้วว่า ผู้ใช้ส่งอินพุทเพื่อเลือกตัวเลือกต่างๆ ได้ 2 วิธี คือ การใช้คีย์ลูกศรเลื่อนแถบสว่างไปยังตัวเลือกที่ต้องการ แล้วกดปุ่ม Enter (โดยทั่วไปแถบสว่างใช้ตัวอักษร reverse) และ Space bar ก็สามารถใช้เลื่อนแถบสว่างได้ด้วย อีกวิธีคือการกดคีย์พิเศษ ความสามารถเหล่านี้มีในฟังก์ชัน get_resp()

บทที่ 7
การสร้าง การติดตั้ง และการทดลอง

วงจรส่วนตรวจจับ

จากรายละเอียดต่างๆและรูปวงจรที่ใช้งาน ดังที่ได้กล่าวมาแล้วนั้น เมื่อประกอบขึ้นเป็นวงจรเรียบร้อยแล้ว ในการอินเตอร์เฟซเชื่อมต่อกับคอมพิวเตอร์ จะใช้คอนเนคเตอร์ 9 ขาเป็นตัวเชื่อมต่อCOM 1 และใช้คอนเนคเตอร์ 25 ขาเป็นตัวเชื่อมต่อ COM 2 โดยการต่อขา Tx ของ COM 1 และ Rx ของ COM 2 เข้ากับวงจรพร้อมทั้งต่อ GROUND เชื่อมระหว่าง COM 1 , COM 2 และวงจรเข้าด้วยกัน และสามารถเริ่มต้นการทำงานได้ โดยการรันโปรแกรมควบคุมการรับ-ส่งข้อมูล ดังที่ปรากฏในภาคผนวก

เมื่อไม่มีผู้มาใช้งานคอมพิวเตอร์ หน้าจอคอมพิวเตอร์จะยังคงปิดไว้ ไม่แสดงผลข้อมูลใดๆทั้งสิ้น จนกว่าจะมีผู้ต้องการใช้งานมานั่งหน้าจอคอมพิวเตอร์ เมื่อมีผู้มานั่งหน้าจอคอมพิวเตอร์ ส่วนตรวจจับจะSENSOR ตัวคน ในระยะ 50 cm. และจะส่งสัญญาณที่สะท้อนจากตัวคนไปทริกให้เปิดหน้าจอเมนูโดยอัตโนมัติ ซึ่งในการเปิดหน้าจอคอมพิวเตอร์นั้น จะเว้นช่วงระยะเวลาประมาณ 3 วินาที เพื่อเช็คให้แน่นอนว่ามีผู้ต้องการใช้งานมานั่งหน้าจอคอมพิวเตอร์จริงมิใช่แค่เพียงเดินผ่านเท่านั้น

หลังจากนั้น ผู้ใช้งานสามารถเลือกหัวข้อเรื่องที่ต้องการจากเมนู ข้อมูลที่เก็บไว้ในหัวเรื่องนั้นจะถูกแสดงออกมาทางหน้าจอคอมพิวเตอร์ ซึ่งข้อมูลอาจจะเป็นข้อความหรือรูปภาพก็ได้ แต่สำหรับโครงงานนี้จะเก็บข้อมูลในลักษณะของรูปภาพไว้

ขณะที่ผู้ใช้นั่งหน้าจอคอมพิวเตอร์ ส่วนตรวจจับจะยังคงตรวจจับตัวคนว่ามีคนนั่งอยู่หรือไม่ตลอดเวลา หากผู้ใช้เลิกการใช้งานในขณะที่ยังไม่ออกจากการทำงาน(คือ กำลังแสดงข้อมูลอยู่) โดยการลุกออกไปจากหน้าจอ หน้าจอจะปิดโดยอัตโนมัติ หรือถ้าผู้ใช้ต้องการเลิกกการใช้งานโดยการกดคีย์ YES ในเมนู QUIT จากหน้าจอเมนู หน้าจอคอมพิวเตอร์ก็จะปิดลงโดยอัตโนมัติเช่นกัน แต่ถ้าผู้ใช้กดคีย์ YES ในเมนู QUIT และยังคงนั่งอยู่หน้าจอคอมพิวเตอร์ หน้าจอก็จะทำการปิดลงโดยอัตโนมัติ และนับช่วงเวลาประมาณ 3 วินาที ก็จะทำการเปิดหน้าจอเมนูให้อีกครั้งหนึ่ง การทำงานก็จะเป็นเช่นนี้ทุกครั้งที่มีผู้มาใช้งาน

บทที่ 8
บทสรุปและข้อเสนอแนะ

สรุปและวิจารณ์ผลการทดลอง

เนื่องจากโครงงานนี้เป็นโครงงานต้นแบบ ซึ่งผู้จัดทำได้ทำโครงงานนี้ขึ้นมาจาก CONCEPT การทำงานที่ตั้งไว้เท่านั้น คือ ต้องการให้คอมพิวเตอร์เปิด-ปิดหน้าจอโดยอัตโนมัติ เมื่อมีผู้ต้องการใช้งานมานั่งหน้าจอคอมพิวเตอร์ และจะแสดงผลของข้อมูลที่ถูกจัดเก็บไว้อย่างเป็นหมวดหมู่นั้น ให้กับผู้ใช้งาน ตามความต้องการของผู้ใช้งาน

จากที่กล่าวมานั้น ผู้จัดทำได้แบ่งแนวความคิดของการทำงานออกเป็นดังนี้

1. ในส่วนของการตรวจจับ(SENSOR)ตัวคนนั้น จะใช้หลักการสะท้อนของรังสีอินฟาเรดเมื่อกระทบสิ่งกีดขวาง จากโครงงานนี้ได้ใช้วงจรรีโมทคอนโทรลเป็นวงจรที่ใช้รับ-ส่งเพื่อสะท้อนรังสีอินฟาเรดกับตัวคนที่ต้องการมาใช้งาน

โดยวงจรเริ่มแรกที่ใช้ในส่วนตรวจจับนี้ ใช้หลักการของวงจรรีโมททั่วไป คือ สร้างสัญญาณความถี่ค่าหนึ่ง ส่งออกไปสะท้อนกับตัวคน และรับสัญญาณที่สะท้อนนั้นกลับเข้ามา แต่เนื่องจากต้องทำการตรวจจับตัวคนตลอดเวลา วงจรที่ใช้จึงต้องสร้างและส่งสัญญาณตลอดเวลา ทำให้เกิดความร้อนขึ้นในตัวอุปกรณ์อิเล็กทรอนิกส์ของภาคส่ง ซึ่งทำให้อุปกรณ์นั้นเสียหายได้ และระยะทางที่ตรวจจับตัวคนได้นั้นมีระยะห่างใกล้เกินไป ต่อมาได้พัฒนาโดยทดลองเปลี่ยนอุปกรณ์ตัวรับ-ส่งรังสีอินฟาเรดที่มีประสิทธิภาพดีขึ้น แต่ก็ยังมีปัญหาเรื่องความร้อนอยู่ เพราะว่าต้องทำการรับ-ส่งอยู่ตลอดเวลา และในที่สุดได้เปลี่ยนแนวความคิดใหม่ โดยให้วงจรทำงานเป็นช่วงเวลาตามที่กำหนด และใช้อุปกรณ์ตัวรับ-ส่งรังสีอินฟาเรดของโทรทัศน์ ซึ่งมีประสิทธิภาพสูงกว่า ทำให้ระยะตรวจจับได้ไกลขึ้น จนเป็นที่พอใจ ซึ่งเราได้ใช้โปรแกรมคอมพิวเตอร์เป็นตัวกำหนดช่วงเวลาในการทำงานของวงจรแทน

2. ส่วนการติดต่อ(INTERFACE)ระหว่างวงจรอิเล็กทรอนิกส์กับคอมพิวเตอร์ ในส่วนนี้เป็นการรับ-ส่งสัญญาณโดยผ่านพอร์ตอนุกรม RS- 232 ของเครื่องคอมพิวเตอร์ ซึ่งในตอนแรกได้ใช้การรับ-ส่งข้อมูลที่พอร์ตCOM1 พอร์ตเดียวโดยการเซตค่าของ REGISTER ต่างๆของพอร์ตให้เหมาะสม แต่มีปัญหาในการรับ-ส่งข้อมูล เนื่องจากเป็นการรับ-ส่งข้อมูลที่พอร์ตเดียวกันทำให้เกิดความผิดพลาดในการตรวจสอบ เนื่องจากในการรับ-ส่งใช้ความเร็วสูงมากและรับ-ส่งตลอดในช่วงเวลาในช่วงที่กำหนดจนดูเหมือนกับรับและส่งพร้อมกัน ทำให้พอร์ตไม่ทราบว่าจะรับหรือส่งค่าที่ออกมาจึงเกิด error

จึงแก้ไขโดยกำหนดให้ส่งข้อมูลที่พอร์ต COM1 และรับข้อมูลที่พอร์ต COM2 ทำให้ในการรับ-ส่งข้อมูลที่ถูกต้องเพื่อนำไปใช้ในส่วนของการแสดงผล

3. ส่วนแสดงผล เนื่องจากโครงงานนี้เป็นโครงงานต้นแบบ ส่วนแสดงผลข้อมูลจึงยังไม่สมบูรณ์นัก เพราะแสดงตัวอย่างของรูปภาพไว้เพียง สองรูปเท่านั้น ซึ่งสามารถพัฒนาต่อไปโดยการเพิ่มจำนวนของภาพและอาจจะมีข้อความบรรยายรูปภาพด้วย หรือสามารถดัดแปลงไปใช้งานในด้านต่างๆ ได้ เช่น การเก็บข้อมูลจำพวกสาราณุกรม เป็นต้น

ข้อเสนอแนะ

ถ้าต้องการให้โครงงานนี้สมบูรณ์ยิ่งขึ้น ในส่วนของโปรแกรมควบคุมส่วนตรวจจับควรจะเป็นโปรแกรมการฝังตัวหรือโปรแกรมเรสซิเดนต์ เพราะจะทำให้การตรวจจับตัวคนทำได้ดีกว่าเนื่องจากเป็นการฝังตัว ถ้ามีสัญญาณเข้ามาทริก ในส่วนตรวจจับก็จะสามารถทำงานได้ทันที ไม่ต้องมีปัญหาเกี่ยวกับการรันโปรแกรมตลอดเวลา

บรรณานุกรม

1. ปราโมทย์ วาดเขียน และ ดร.วิวัฒน์ กิรานน์;"พื้นฐานการสื่อสารข้อมูล";JICA.

2. มโน มงคลธนานนท์, มงคล มงคลธนานนท์; " คัมภีร์โปรแกรมเมอร์ PETER NORTON"; บริษัท อินฟอร์เมติก บิซิเนส พับลิเคชั่น จำกัด,กรุงเทพฯ.

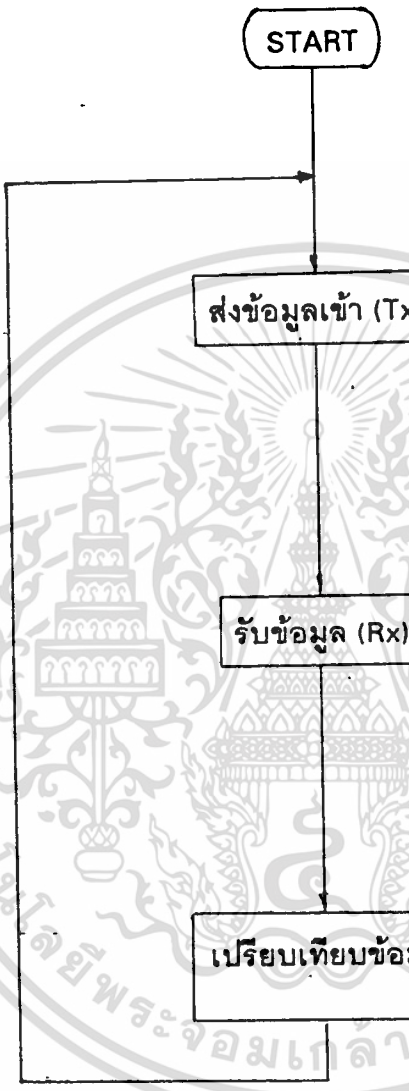
3. ศิวัฒน์ ศิวะบวร, พรชัย จักรธำรงค์, จิรศักดิ์ ชัยวิริยะกุล;" การประยุกต์ใช้งานภาษาซี";บริษัทซีเอ็ดยูเคชั่น จำกัด(มหาชน),กรุงเทพฯ.

4. ธันวา ศรีประโมง;" การเขียนโปรแกรมภาษาซี สำหรับวิศวกรรม"; โครงการตำราวิชาการมหาวิทยาลัยเทคโนโลยีมหานคร,กรุงเทพฯ.

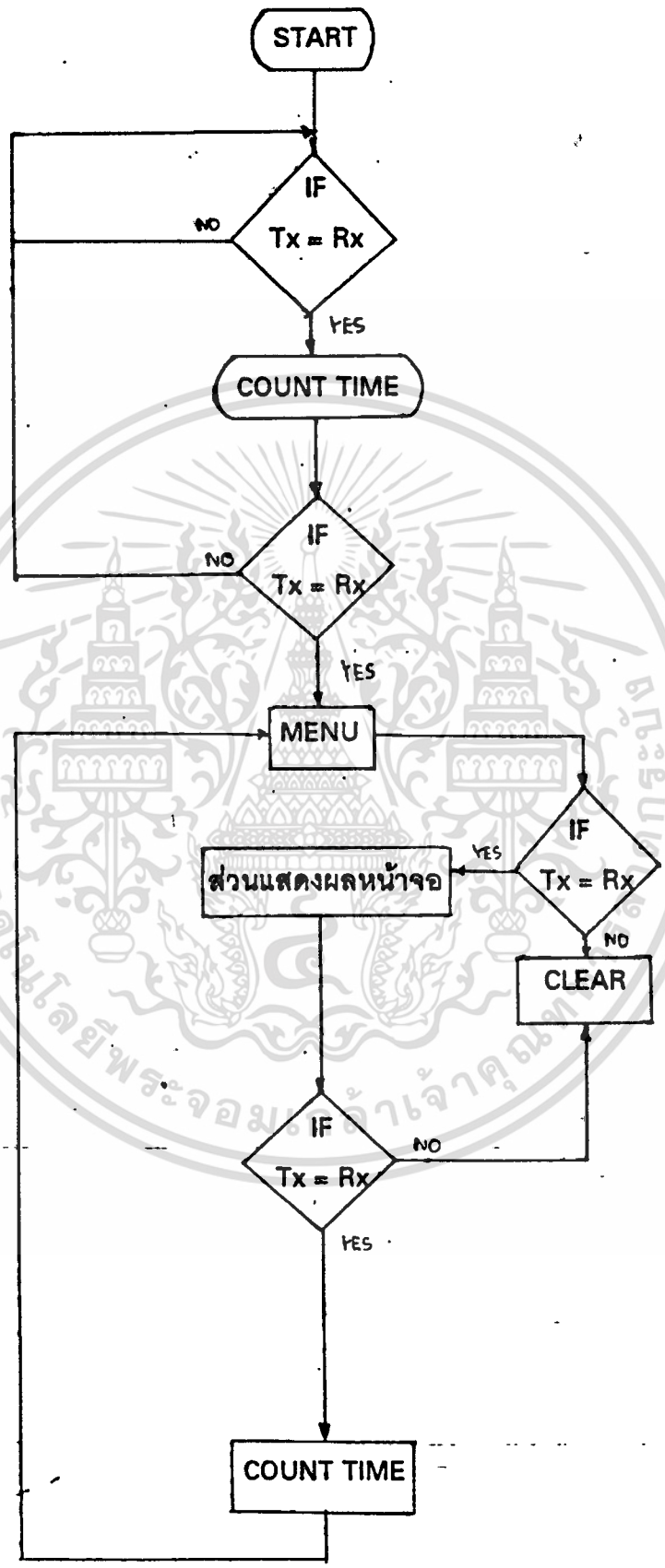
5. WILLIAMS;"DOS 5 : A DEVELOPER 'S GUIDE ADVANCED PROGRAMMING GUIDE TO DOS";PRENTICE HALL.

ภาคผนวก

FLOW CHART แสดงโครงสร้างการทำงาน



FLOW CHART แสดงโครงสร้างของส่วนเปรียบเทียบข้อมูล



```
#include"stdio.h"
#include "dos.h"
#include "bios.h"
#include "conio.h"
#include "stdlib.h"
#include "time.h"

#define TCK 0x1c /*use timer tick interrupt to update screen*/

#define COM1 0x3F8
#define COM2 0x2F8
int First=0,Second=0,Third=0,CheckSent=0;
int chk =1;

int TestCount=0,test=0;
char far *scrptr;
char far *scrptx;
char value[2]={'0','1'},result[2]={'0','0'};
char Oldvalue;
int  Count=0;       /*global variable for display delay*/

void Paracom(int Baudrate,int Parity,int Stop,int Data);
void send_rec(void);
void Close(void);
void Norm_cursor(void);
void interrupt check_data(void);
void interrupt (*oldint)(void);

void Paracom(int Baudrate,int Parity,int Stop,int Data)
{
  unsigned char octpar;
  unsigned int divider;
  unsigned char ops,oms;
            divider = 115200/Baudrate;
            oms = divider >> 8;
            ops = (divider << 8) >> 8;
```

```
        outp ((COM1 + 3),128);
        outp ((COM2 + 3),128);
        outp (COM1,ops);
        outp (COM2,ops);
        outp (COM1+1,oms);
        outp (COM2+1,oms);
        octpar = Data - 5;
        octpar |= (Stop - 1) << 2;
        if (Parity)octpar |= 8;
        if (Parity == 2) octpar |= 16;
        outp(COM1+3,octpar);
        outp(COM2+3,octpar);
}

void interrupt check_data(void)
{
  int i;
  disable();

  Count++;

  /* TestCount = 0 is first Count ; TestCount = 1 is second Count */
  switch(TestCount){
    case 0 :
      if(Count == 36)
          First = 1;
      break;

    case 1 :
      if(Count == 15)
          Second = 1;
      break;

    case 2 :
      if(Count == 36)
```

```
        Third = 1;
    break;
}

if(First == 1)
 {
  switch(chk){
       case 1:
        send_rec();  /* out port */
        ++chk;
       break;
       case 2:
        send_rec();  /*inport */
        if(Oldvalue == result[1])
        {
          ++chk;

        }
        else {
                First = 0;
                Count = 0;
                TestCount = 0;
                chk = 1;
            }
       break;
       case 3:
        send_rec();  /*outport */
        ++chk;
       break;
       case 4:
        send_rec();  /*inport */
        if(Oldvalue == result[1])
        {

          chk=1;
          Count = 0;
```

```
            TestCount = 1;
            First = 0;
        }
         else {
                TestCount = 0;
                First = 0;
                Count = 0;
                chk = 1;
            }
        break;
    }
}

if(Second == 1)
{
   switch(chk){
        case 1:
          send_rec(); /*outport*/
          ++chk;
        break;
        case 2:
          send_rec(); /*inport*/
          if(Oldvalue == result[1])
          {

            TestCount = 2;
            Second = 0;
            Count = 0;
            test=1;
            chk = 1;
          }
          else {
                TestCount = 0;
                Second = 0;
                chk = 1;
                Count = 0;
```

```
            }
        break;
    }


}

if(Third == 1)
{
  switch(chk){
    case 1:
        send_rec(); /*outport*/
        ++chk;
    break;
    case 2:
        send_rec(); /*inport*/
        ++chk;
        if(Oldvalue != result[1])
        {

          scrptx = (char far *)0xb8000000;
          *scrptx = 1;
          TestCount = 1;
          Count = 0;
          Third = 0;
        }
        else {

                scrptx = (char far *)0xb8000000;
                *scrptx = 32;


                Count = 0;
                Third = 0;
                chk = 1;
            }
    break;
  }
```

```
    }
    enable();
    (*oldint)(); /*call old interrupt*/
}

void send_rec(void)
{
    int line_state;
/*   value[1] = 'B'; */
    Paracom(1200,0,1,8);
    if(CheckSent == 1)
    {
      result[1] = inp(COM2);
      CheckSent = 0;
    }

    if(CheckSent == 0)
    {
      Oldvalue = value[1];
      outp(COM1,(++value[1])%126);
      CheckSent = 1;
    }

}

void Close(void)
{
    clrscr();
    _setcursortype(_NOCURSOR);
}

void Norm_cursor(void)
{
    _setcursortype(_NORMALCURSOR);
}
void main(void)
```

```
{

 scrptr = (char far *)0xb8000090;

 Close();

  oldint = getvect(TCK);   /*save old timer interrupt*/
  setvect(TCK,check_data);         /*set new interrrupt*/

   while(!kbhit()){
            scrptx = (char far *)0xb8000000;
            if(test == 1)
                {
                 Norm_cursor();
                 system("tail2.exe");
                 Close();
                 *scrptx=1;
                 test=0;
                 TestCount = 0;
                }
           }
  setvect(TCK,oldint);
}
```

```
#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <mem.h>

#define GOOD_READ                       0
#define  BAD_FILE                       1
#define  BAD_READ                       2
#define MEMORY_ERROR                    3

#define RGB_RED                         0
#define RGB_GREEN                       1
#define RGB_BLUE                        2
#define RGB_SIZE                        3

#define  TIFFlong               4

#define SCREENWIDE                      320
#define SCREENDEEP                      200
#define STEP                            32

#define CURSOR_LEFT                     0x4b00
#define CURSOR_RIGHT                    0x4d00
#define CURSOR_UP                       0x4800
#define CURSOR_DOWN                     0x5000
#define HOME                            0x4700
#define END                             0x4f00

#define  pixels2bytes(n)                ((n+7)/8)
#define greyvalue(r,g,b)        (((r*30)/100)+((g*59)/100)+((b*11)/100))

typedef struct {
        int width,depth,bytes,bits;
        int background;
        char palette[768];
        int (*setup)();
```

```
        int (*closedown)();
        } FILEINFO;

/* local prototype */

typedef struct {
        char id[2];
        long filesize;
        int reserved[2];
        long headersize;
        long infoSize;
        long width;
        long depth;
        int biPlanes;
        int bits;
        long biCompression;
        long biSizeImage;
        long biXpelsPerMeter;
        long biYpelsPerMeter;
        long biClrUsed;
        long biClrImportant;
        } BMPHEAD;

char *farPtr(char *p,long l);
char *getline(unsigned int n);
char *mono2vga(char *p, int width);
char *ega2vga(char *p, int width);
char *rgb2vga(char *p, int width);
int dosetup(FILEINFO *fi);
int doclosedown(FILEINFO *fi);

int putline(char *p, unsigned int n);

char masktable[8]={0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};

static FILEINFO fi;
```

```
static char *buffer=NULL;

main()
{
        read_bmp("a1_1.bmp");
        read_bmp("a3.bmp");
}

read_bmp(char *argv)
{
        FILE *fp;
        char path[81];
        static char results[8][16] = { "Ok",
                                       "Bad file",
                                       "Bad read",
                                       "Memory error",
                                       };
        int r;

        /*if(argc > 1) {
                strmfe(path,argv[1],"BMP");
                strupr(path);*/
                if((fp = fopen(argv,"rb")) != NULL) {
                        fi.setup = dosetup;
                        fi.closedown = doclosedown;
                        unpackbmp(fp,&fi);
                        fclose(fp);
                }else printf("Error opening %s\n",path);
/*      }else puts("Argument:       path to a BMP file");*/
}

int unpackbmp(FILE *fp,FILEINFO *fi)
{
        BMPHEAD bmp;
        char *p,*pr;
        int i,n;
```

```
memcpy(fi->palette,"\000\000\000\377\377\377",6);

if(fread((char *)&bmp,1,sizeof(BMPHEAD),fp)==sizeof(BMPHEAD)){
        if(!memcmp(bmp.id,"BM",2)){
                fi->width=(int)bmp.width;
                fi->depth=(int)bmp.depth;
                fi->bits=bmp.bits;

                if(fi->bits==1) fi->bytes = pixels2bytes(fi->width);
                else if(fi->bits==4) fi->bytes=pixels2bytes(fi->width)>>2;
                else if(fi->bits==8) fi->bytes=fi->width;
                else if(fi->bits==24) fi->bytes=fi->width*3;

                if(fi->bytes & 0x0003) {
                        fi->bytes |= 0x0003;
                        ++fi->bytes;
                }

                if(fi->bits > 1 && fi->bits <= 8) {
                        n = 1 << fi->bits;
                        for(i=0;i<n;++i){
                                fi->palette[(i*RGB_SIZE)+RGB_BLUE] = fgetc(fp);
                                fi->palette[(i*RGB_SIZE)+RGB_GREEN] = fgetc(fp);
                                fi->palette[(i*RGB_SIZE)+RGB_RED] = fgetc(fp);
                                fgetc(fp);
                        }
                }
                else if(fi->bits==24) {
                        for(i=0;i<256;++i)
                                memset(fi->palette+(i*RGB_SIZE),i,RGB_SIZE);
                }
                if((p=malloc(fi->bytes)) != NULL) {
                        if((fi->setup)(fi) != GOOD_READ) {
                                free(p);
                                return(MEMORY_ERROR);
```

```
}
fseek(fp,bmp.headersize,SEEK_SET);

for(i=0;i<fi->depth;++i) {
        if(fread(p,1,fi->bytes,fp) !=fi->bytes) {
                freebuffer();
                free(p);
                return(BAD_READ);
        }

        switch(fi->bits) {
        case 1:
                pr=mono2vga(p,fi->width);
                if(pr != NULL) {
                        putline(pr,fi->depth-1-i);
                        free(pr);
                }
                else {
                        freebuffer();
                        free(p);
                        return(MEMORY_ERROR);
                }
                break;
        case 4:
                pr=ega2vga(p,fi->width);
                if(pr != NULL) {
                        putline(pr,fi->depth-1-i);
                        free(pr);
                }
                else {
                        freebuffer();
                        free(p);
                        return(MEMORY_ERROR);
                }
                break;
        case 8:
```

```
                putline(p,fi->depth-1-i);
                break;
            case 24:
                pr=rgb2vga(p,fi->width);
                if(pr != NULL) {
                    putline(pr,fi->depth-1-i);
                    free(pr);
                }
                else {
                    freebuffer();
                    free(p);
                    return(MEMORY_ERROR);
                }
                break;
            }
        }
        (fi->closedown)(fi);
        free(p);
        return(GOOD_READ);
    }else return(MEMORY_ERROR);
  } else return(BAD_FILE);
 } else return(BAD_READ);
}

char *mono2vga(char *p, int width)
{
    char *pr;
    int i;

    if((pr = malloc(width)) != NULL) {
        for(i=0;i<width;++i) {
            if(p[i>>3] & masktable[i & 0x0007])
                pr[i]=1;
            else
                pr[i]=0;
        } return(pr);
```

```
        } else return(NULL);
}

char *rgb2vga(char *p, int width)
{
        char *pr;
        int i;
        if((pr=malloc(width)) != NULL) {
                for(i=0;i<width;++i){
                        p[i] = greyvalue(p[RGB_RED],p[RGB_GREEN],p[RGB_BLUE]);
                        p+=RGB_SIZE;
                } return(pr);
        } return(NULL);
}

char *ega2vga(char *p, int width)
{
        char *pr;
        int i,j=0;
        if((pr=malloc(width)) != NULL) {
                for(i=0;i<width;) {
                        pr[i++]=(p[j]>>4) & 0x0f;
                        pr[i++]=p[j] & 0x0f;
                        ++j;
                } return(pr);
        } else return(NULL);
}

dosetup(FILEINFO *fi)
{
        union REGS r;

        if(!getbuffer((long)fi->width*(long)fi->depth,fi->width,fi->depth))
                return(MEMORY_ERROR);
```

```
        r.x.ax = 0x0013;
        int86(0x10,&r,&r);

        setvgapalette(fi->palette,256,fi->background);

        return(GOOD_READ);
}

doclosedown(FILEINFO *fi)
{
        union REGS r;
        int c,i,n,x=0,y=0;

        if(fi->width > SCREENWIDE) n = SCREENWIDE;
        else n = fi->width;
        for(i=0;i<SCREENDEEP;++i){
                c = y+i;
                if(c>=fi->depth)break;
                memcpy(MK_FP(0xa000,SCREENWIDE*i),getline(c)+x,n);
        }
        c = GetKey();

        freebuffer();

        r.x.ax=0x0003;
        int86(0x10,&r,&r);
        return(GOOD_READ);
}

strmfe(char *new, char *old, char *ext)
{
        while (*old != 0 && *old != '.') *new++ = *old++;
        *new++ = '.';
        while(*ext) *new++ = *ext++;
        *new = 0;
```

```
}

GetKey()
{
        int c;
        c = getch();
        if(!(c && 0x00ff)) c = getch() << 8;
        return(c);
}

setvgapalette(char *p,int n, int b)
{
        union REGS r;
        int i;
        outp(0x3c6,0xff);
        for(i=0;i<n;++i){
                outp(0x3c8,i);
                outp(0x3c9,(*p++) >> 2);
                outp(0x3c9,(*p++) >> 2);
                outp(0x3c9,(*p++) >> 2);
        }
        r.x.ax = 0x1001;
        r.h.bh = b;
        int86(0x10,&r,&r);
}

/* return a far pointer plus a long integer */
char *farPtr(char *p,long l)
{
        unsigned int seg,off;

        seg = FP_SEG(p);
        off = FP_OFF(p);
        seg += (off / 16);
        off &= 0x000f;
        off += (unsigned int)(l & 0x000fL);
```

```
        seg += (unsigned int)(l / 16L);
        p = MK_FP(seg,off);
        return(p);
}

/* save one line to memory */
int putline(char *p,unsigned int n)
{
        if(n >= 0 && n < fi.depth)
                memcpy(farPtr(buffer,(long)n*(long)fi.width),p,fi.width);
}

/* get one line from memory */
char *getline(unsigned int n)
{
        return(farPtr(buffer,(long)n*(long)fi.width));
}

#pragma warn -par
getbuffer(unsigned long n, int bytes, int lines)
{
        if((buffer = farmalloc(n)) == NULL) return(0);
        else return(1);
}

#pragma warn -par
freebuffer()
{
        if(buffer != NULL) farfree(buffer);
        buffer=NULL;
}
```

```
/*** HEAD.H***/

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <dir.h>
#include <stdlib.h>
#include <alloc.h>
#include <ctype.h>


#define BKSP  8
#define TAB_KEY 9
#define ENTER 13
#define SHF_TAB 15
#define ESC  27
#define HOME 71
#define UP   72
#define LEFT 75
#define RIGHT 77
#define END 79
#define DOWN 80
#define DELETE 83
#define MAGDIX  4
#define TRUE 1
#define FALSE  0
```

```
                    /*** MIDDLE1.CPP ***/

#include <head.h>
extern char far *vid_mem;
void background(ch,startx,endx,starty,endy,attrib)
char ch;
int startx,endx,starty,endy,attrib;
{
        char far *v;
        int i,j;
        for(i=starty;i<=endy;i++)
                for(j=startx;j<=endx;j++)
                {
                        v=vid_mem;
                        v+=(j*2+i*160);
                        *v++=ch;
                        *v=attrib;

                }
}
void border(code,startx,endx,starty,endy,attrib)
int code,startx,endx,starty,endy,attrib;
{
     char far *v;
     int x,y;
     char side_left=179,side_right=179,top=196,bottom=196;
     char top_left=218,bottom_left=192,top_right=191,bottom_right=217;
     switch(code)
```

```
        write_char(startx,starty,top_left,attrib);
        write_char(startx,endy,bottom_left,attrib);
        write_char(endx,starty,top_right,attrib);
        write_char(endx,endy,bottom_right,attrib);
}
void clearline(startx,endx,line,attrib)
int startx,endx,line,attrib;
{
        char far *v;
        int i;
        for (i=startx;i<endx;i++)
        {
                v=vid_mem;
                v+=(i*2+line*160);
                *v++=' ';
                *v=attrib;
        }
}
  int is_in(s,c)
  char *s,c;
  {
     int i;
     for(i=0;*s;i++)
        if(*s++==c) return i+1;
        return 0;
     }

     char *keys;
```

```
int count;
int startx,starty,attrib1,attrib2,shtkey_att1,shtkey_att2;
{
    int len= strlen(menu[0]);
    int endx,endy,choice;
    unsigned char *p;
    if((startx<0)||(startx>79)||(starty<0)||(starty>24))
    {
       printf("Range of popup menu error");
       return ERROR;
    }
    endx=len+startx+1;
    endy=count+starty+1;
    if((endx>79)||(endy>24))
    {
       printf("Menu won't fit");
       return ERROR;
    }
    p=(unsigned char *)farmalloc(2*((endx-startx+2)*(endy-starty+2)));
    if(!p)return ERROR;
    save_video(startx,endx+1,starty,endy+1,p);
    background(' ',startx,endx,starty,endy,attrib1);
    shadow(startx+1,endx+1,starty+1,endy+1);
    border(1,startx,endx,starty,endy,attrib1);
    choice=get_resp(startx+1,starty,menu,keys,count,attrib1,attrib2,
          shtkey_att1,shtkey_att2);
    restore_video(startx,endx+1,starty,endy+1,p);
    farfree(p);
```

```
return choice;
}          */
void sizecurlor(start,end)
int start,end;
{
  union REGS r;
  r.h.ah =1;
  r.h.ch=start;r.h.cl=end;
  int86(0x10,&r,&r);
  }

void restore_video(startx,endx,starty,endy,buf_ptr)
int startx,endx,starty,endy;
unsigned char *buf_ptr;
{
    int i,j;
    char far *v;
    for(i=starty;i<=endy;i++)
      for(j=startx;j<=endx;j++)
       {
          v=vid_mem;
          v+=(j*2+i*160);
          *v++=*buf_ptr++;
          *v =*buf_ptr++;
      }
  }
  void save_video(startx,endx,starty,endy,buf_ptr)
  int startx,endx,starty,endy;
```

```
unsigned char *buf_ptr;
{
  int i,j;
  char far *v;
  for(i=starty;i<=endy;i++)
    for(j=startx;j<=endx;j++)
    {
      v=vid_mem;
      v+=(j*2+i*160);
      *buf_ptr++=*v++;
      *buf_ptr++=*v;
    }
}

void shadow(startx,endx,starty,endy)
int startx,endx,starty,endy;
{
    int x,y;
    char far *v;
    for(x=startx;x<=endx;x++)
    {
      v=vid_mem;
      v+=x*2+y*160;
      v++;
      *v++=0x08;
    }
  endx--;
  for(y=starty;y<=endy;y++)
```

```
    {
      v=vid_mem;
      v+=endx*2+y*160;
      v++;
      *v++=0x08;
      v++;
      *v=0x08;
     }
  }

int video_mode()
{
    union REGS r;
    r.h.ah=15;
    return int86(0x10,&r,&r)&255;
}
void write_char(x,y,ch,attrib)
int x,y,attrib;
char ch;
{
    char far *v;
    v=vid_mem;
    v+=(x*2+y*160);
    *v++=ch;
    *v=attrib;
}
void write_menu(x,y,str,attrib1,attrib2)
int x,y;
```

```
char *str;
int attrib1,attrib2;
{
    int i;
    char far *v;
    v=vid_mem;
    v+=x*2+y*160;
    for(i=x;*str;i++)
    {
      if(*str=='%')
      {
          str++;
          *v++=*str++;
          *v++=attrib2;
       }
       else
       {
          *v++=*str++;
          *v++=attrib1;
       }
    }
}
```

```
#include "head.h"
#include <process.h>


#define COM1  0
#define F10   68
#define Alt_F 33
#define Alt_T 20
#define Alt_S 31
#define Alt_Q  16
#define MAX_FRAME 3
#define BACKG 0x70
#define FRAME 0x17
#define NORM_VID 0x30
#define HOT_KEY 0x74
#define HOT_REV 0x24
#define MENU_BAR 0x70
#define DES_ATT 0x71
#define BORDER 0x70
#define REV_VID  0x20


void make_main_menu(int ,char **,char *,int,int ,int);
void display_pulldown(int);
void user_service(void);
void clear_dialog(void);


char far *vid_mem;
struct menu_info {
```

```
        int startx,starty,endx,endy;
        unsigned char *p;
        char **menu;
        char *keys;
        int count;
        }menu_num[MAX_FRAME];
char  *main_menu[]={
        "%Video  %Show",

        "%Slide  %Show",

        "%Quit",
        };
char  *mainkey = "vsq";
char  *Video[]={
        "%Thailand",
        "%British",
        "%France",
        "%Denmark",
        "%Australia",
        "%Swiszerland",
        };
 char *Slide[] = {
        "%Engineer %Fac.",
        "%Magic Land",
        "%Nobel Prize",
        "%My Country",
        "%Wild Animals",
```

```
    "%Sun flower",
    };
char *Quit[]={
    "%Yes      ",
    "%No       ",
    };

char *describ[]={
  "Select story from vediotape",
  "Select story from image file",
  "Select [Yes] if you want exit from Program",
  };

int main_menu_startx[MAX_FRAME]= {8,30,58};

FILE *fp;

void main(void)
{
  int i,vmode;
  vmode=video_mode();
  if((vmode!=2)&&(vmode!=3)&&(vmode!=7))
   {
    printf("Video is not in 80 text mode ");
    exit(1);
   }
   if(vmode==7)vid_mem=(char far *) 0xB0000000;
   else vid_mem =(char far *) 0xB8000000;
```

```
_setcursortype(_NOCURSOR);

make_main_menu(0,Video,"nlscdr",6,8,1);

make_main_menu(1,Slide,"avdmws",6,30,1);

make_main_menu(2,Quit,"yn",2,58,1);


background(' ',0,79,0,24,BACKG);
background(' ',0,79,1,23,FRAME);


write_menu(2,24,"%F10-Menu | Arrow keys Select",MENU_BAR,HOT_KEY);
user_service();
for(i=0;i<MAX_FRAME;i++)
  farfree(menu_num[i].p);

  background(' ',0,79,0,24,0x07);
  _setcursortype(_NORMALCURSOR);
  gotoxy(0,0);
}

void make_main_menu(int num,char *menu[],char *keys,int count,
                    int startx,int starty)
{
  int i,len;
  int endx,endy,vmode;
```

```
    unsigned char *p;
    len=0;
    for(i=0;i<count;i++)
        if(len<strlen(menu[i]))len=strlen(menu[i]);
    endx=startx+len+4;
    endy=count+1+starty;
    if((endy>25)||(endx>80))
    {
        printf("Size of menu error\n");
        exit(1);
    }
    p=(unsigned char *)farmalloc(2*((endx-startx+3)*(endy-starty+3)));
    if(!p)
        exit(1);
    menu_num[num].startx=startx;
    menu_num[num].endx=endx-5;
    menu_num[num].starty=starty;
    menu_num[num].endy=endy;
    menu_num[num].p=p;
    menu_num[num].menu=(char **)menu;
    menu_num[num].keys=keys;
    menu_num[num].count =count;
}
void display_pulldown(int num)
{
    char **m;
    int i;
    m=menu_num[num].menu;
```

```
  save_video(menu_num[num].startx-1,menu_num[num].endx+2,
                menu_num[num].starty,menu_num[num].endy+2,
                menu_num[num].p);
  background(' ',menu_num[num].startx-1,menu_num[num].endx,
                menu_num[num].starty,menu_num[num].endy,BACKG);
  shadow(menu_num[num].startx,menu_num[num].endx+1,
                menu_num[num].starty+1,menu_num[num].endy+1);
  border(1,menu_num[num].startx-1,menu_num[num].endx,
                menu_num[num].starty,menu_num[num].endy,BORDER);
  for(i=0;i<menu_num[num].count;i++)
  write_menu(menu_num[num].startx,i+2,m[i],MENU_BAR,HOT_KEY);

}
void user_service()
{
  int mode=0, choice=0,num=0;
  int leftx=0;
  int sel_choice=0;
  int user_select=FALSE;
  char **m;
  inkey_t c;
  for(num=0;num<MAX_FRAME;num++)
    write_menu(main_menu_startx[num],0,main_menu[num],
                MENU_BAR,HOT_KEY);
    clearline(leftx,79,24,MENU_BAR);
    write_menu(leftx,24,
    "%F10-MenuArrow keys-Select.Enter Information."
          ,MENU_BAR,HOT_KEY);
```

```
num=0;
for(;;)
{
    while(!bioskey(1));
    c.i = bioskey(0);
    if(mode==0)
     {
        if(c.ch[0]=='\r')
         {
            mode=1;
            write_menu(main_menu_startx[num],0,main_menu[num],
            REV_VID,HOT_REV);
            clearline(leftx,79,24,MENU_BAR);
            write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
         }
        else
         {
        switch(c.ch[1])
         {
            case F10:
            {
              mode=1;
              write_menu(main_menu_startx[num],0,main_menu[num],
                        REV_VID,HOT_REV);
              clearline(leftx,79,24,MENU_BAR);
              write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
             }break;
```

```
            case UP:
          break;
            case LEFT:
          break;
            case RIGHT:
          break;
            case DOWN:
          break;
        }
    }
}
    else if (mode==1)
    {
      if(c.ch[0])
      {
        switch(c.ch[0])
        {
          case '\r':
          {
            if(num==MAX_FRAME)
              return;
              display_pulldown(num);
              m=menu_num[num].menu;
              write_menu(menu_num[num].startx,choice+2,m[choice],
                        REV_VID,HOT_REV);
              mode=2;
          }break;
```

```
    case  ESC :
    {
       mode=0;
       write_menu(main_menu_startx[num],0,main_menu[num],
                  MENU_BAR,HOT_KEY);
       clearline(leftx,79,24,MENU_BAR);
       write_menu(2,24,
           "%F%1%0-Menu | Arrow keys-Select.Enter-Information",
           MENU_BAR,HOT_KEY);
       }break;
    }
}
else
{
   switch(c.ch[1])
   {
      case F10:
       {
        mode=0;
        write_menu(main_menu_startx[num],0,main_menu[num],
                   MENU_BAR,HOT_KEY);
        clearline(leftx,79,24,MENU_BAR);
        write_menu(2,24,
             "%F%1%0-MenulArrow keys-Select.Enter-Information",
             MENU_BAR,HOT_KEY);
       }break;

      case Alt_F:
```

```
{
 write_menu(main_menu_startx[num],0,main_menu[num],
            MENU_BAR,HOT_KEY);
 num=0;
 write_menu(main_menu_startx[num],0,main_menu[num],
            REV_VID,HOT_REV);
 clearline(leftx,79,24,MENU_BAR);
 write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
 display_pulldown(num);
 m=menu_num[num].menu;
 write_menu(menu_num[num].startx,choice+2,m[choice],
            REV_VID,HOT_REV);
 mode=2;
}break;

case Alt_T:
{
  write_menu(main_menu_startx[num],0,main_menu[num],
             MENU_BAR,HOT_KEY);
  num=1;
  write_menu(main_menu_startx[num],0,main_menu[num],
             REV_VID,HOT_REV);
  clearline(leftx,79,24,MENU_BAR);
  write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
  display_pulldown(num);
  m=menu_num[num].menu;
  write_menu(menu_num[num].startx,choice+2,m[choice],
             REV_VID,HOT_REV);
```

```
    mode=2;
  }break;

case Alt_S:
  {
    write_menu(main_menu_startx[num],0,main_menu[num],
               MENU_BAR,HOT_KEY);
    num=2;
    write_menu(main_menu_startx[num],0,main_menu[num],
               REV_VID,HOT_REV);
    clearline(leftx,79,24,MENU_BAR);
    write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
    display_pulldown(num);
    m=menu_num[num].menu;
    write_menu(menu_num[num].startx,choice+2,m[choice],
               REV_VID,HOT_REV);
    mode=2;
  }break;

case Alt_Q:
    exit(1);

case LEFT:
  {
    write_menu(main_menu_startx[num],0,main_menu[num],
               MENU_BAR,HOT_KEY);
    num--;
    if(num<0) num=MAX_FRAME-1;
```

```
            write_menu(main_menu_startx[num],0,main_menu[num],
                    REV_VID,HOT_REV);
            clearline(leftx,79,24,MENU_BAR);
            write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
          }break;

          case RIGHT:
          {
            write_menu(main_menu_startx[num],0,main_menu[num],
                    MENU_BAR,HOT_KEY);
            num++;
            if(num>MAX_FRAME-1) num=0;
              write_menu(main_menu_startx[num],0,main_menu[num],
                    REV_VID,HOT_REV);
             clearline(leftx,79,24,MENU_BAR);
              write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
          }break;

          case DOWN:
          {
            display_pulldown(num);
            m=menu_num[num].menu;
            write_menu(menu_num[num].startx,choice+2,m[choice],
                    REV_VID,HOT_REV);
            mode=2;
          }break;
        }
      }
```

```
}
else if(mode==2)
{
    if(c.ch[0])
    {
      sel_choice=is_in(menu_num[num].keys,tolower(c.ch[0]));
      if(sel_choice)
       {
         write_menu(menu_num[num].startx,choice+2,m[choice],
                    MENU_BAR,HOT_KEY);
         choice=sel_choice-1;
         write_menu(menu_num[num].startx,choice+2,m[choice],
                    REV_VID,HOT_REV);
         clearline(leftx,79,24,MENU_BAR);
         write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
        restore_video(menu_num[num].startx-1,
                      menu_num[num].endx+2,menu_num[num].starty,
                      menu_num[num].endy+2,menu_num[num].p);
         user_select=TRUE;
       }
      if(!user_select)
       {
         switch(c.ch[0])
         {
           case '\r':
            {
              restore_video(menu_num[num].startx-1,
```

```
                menu_num[num].endx+2,menu_num[num].starty,
                menu_num[num].endy+2,menu_num[num].p);
            user_select=TRUE;
           }break;
          case ESC:
           {
            restore_video(menu_num[num].startx-1,
                menu_num[num].endx+2,
                menu_num[num].starty,menu_num[num].endy+2,
                menu_num[num].p);
            choice=0;
            mode=1;
           }break;
        }
      }
    }
  else
   {
     switch(c.ch[1])
     {
       case UP:
       {
        write_menu(menu_num[num].startx,choice+2,m[choice],
                MENU_BAR,HOT_KEY);
        choice--;
        if(choice<0)choice=menu_num[num].count-1;
        write_menu(menu_num[num].startx,choice+2,m[choice],
                REV_VID,HOT_REV);
```

```
}break;
case DOWN:
{
   write_menu(menu_num[num].startx,choice+2,m[choice],
              MENU_BAR,HOT_KEY);
  choice++;
  if(choice>(menu_num[num].count-1))
  choice=0;
  write_menu(menu_num[num].startx,choice+2,m[choice],
              REV_VID,HOT_REV);
}break;
case LEFT:
{
  restore_video(menu_num[num].startx-1,
              menu_num[num].endx+2,
              menu_num[num].starty,menu_num[num].endy+2,
              menu_num[num].p);
  write_menu(main_menu_startx[num],0,main_menu[num],
              MENU_BAR,HOT_KEY);
  num--;
    if(num<0)num=MAX_FRAME-1;
  write_menu(main_menu_startx[num],0,main_menu[num],
              REV_VID,HOT_REV);
  clearline(leftx,79,24,MENU_BAR);
  write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
  m=menu_num[num].menu;
  display_pulldown(num);
  choice=0;
```

```
            write_menu(menu_num[num].startx,choice+2,m[choice],
                    REV_VID,HOT_REV);
          }break;
        case RIGHT:
          {
            restore_video(menu_num[num].startx-1,
                        menu_num[num].endx+2,
            menu_num[num].starty,menu_num[num].endy+2,
                        menu_num[num].p);
            write_menu(main_menu_startx[num],0,main_menu[num],
                    MENU_BAR,HOT_KEY);
            num++;
              if(num>MAX_FRAME-1) num=0;
            write_menu(main_menu_startx[num],0,main_menu[num],
                    REV_VID,HOT_REV);
            clearline(leftx,79,24,MENU_BAR);
            write_menu(leftx,24,describ[num],DES_ATT,HOT_REV);
            m=menu_num[num].menu;
            display_pulldown(num);
            choice=0;
            write_menu(menu_num[num].startx,choice+2,m[choice],
                    REV_VID,HOT_REV);
          }break;
        }
      }
    }
    if(user_select)
    {
```

```
user_select=FALSE;
mode=1;
switch(num)
{
  case 0:
  {
    switch(choice)
    {
      case 0:{exit(1);}
       break;
      case 1:{exit(1);}
       break;
      case 2:{exit(1);}
       break;
      case 3:{exit(1);}
        break;
      case 4: {exit(1);}
        break;
      case 5: {exit(1);}
        break;
    }
    choice=0;
  }break;
  case 1:
  {
    switch(choice)
    {
       case 0:
```

```
            break;
          case 1:
            break;
          case 2:
             break;
          case 3:
             break;
          case 4:
             break;
          case 5:
             break;
        }
       choice=0;
    }break;
case 2:
    {
       switch(choice)
        {
          case 0:
            {
              exit(1);
            }break;
          case 1:
             break;
        choice=0;
       }break;

}
```