ระบบรักษาความปลอดภัยแบบติดตามวัตถุอัตโนมัติ

AUTOMATIC OBJECT TRACKING SECURITY SYSTEM

โดย

นายประมุข จิตจาตุรันต์

นายพงศ์ชัย เอื้อพงศธร

นางสาวมณียา กันกา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ระบบรักษาความปลอดภัยแบบติดตามวัตถุอัตโนมัติ

AUTOMATIC OBJECT TRACKING SECURITY SYSTEM

โดย

นายประมุข จิตจาตุรันต์ 44010286

นายพงศ์ชัย เอื้อพงศธร 44010308

นางสาวมณียา กันกา 44010369

อาจารย์ที่ปรึกษา

ผศ. นภัทร สระเอี่ยม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบรักษาความปลอดภัยแบบติดตามวัตถุอัตโนมัติ**

AUTOMATIC OBJECT TRACKING SECURITY SYSTEM

ผู้จัดทำ

1. **นายประมุข จิตจาตุรันต์ 44010286**
2. นายพงศ์ชัย เอื้อพงศธร 44010308
3. นางสาวมณียา กันกา **44010369**

..... อาจารย์ที่ปรึกษา

(ผศ. นภัทร สระเอี่ยม)

ระบบรักษาความปลอดภัยแบบติดตามวัตถุอัตโนมัติ

AUTOMATIC OBJECT TRACKING SECURITY SYSTEM

โดย นายประมุข จิตจาตุรันต์ 44010286

นายพงศ์ชัย เอื้อพงศธร 44010308

นางสาวมณียา กันกา 44010369

อาจารย์ที่ปรึกษา ผศ. นภัทร สระเอี่ยม

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอ ระบบรักษาความปลอดภัยโดยใช้กล้องติดตามวัตถุอัตโนมัติ ซึ่งเป็นการนำเอาไมโครคอนโทรลเลอร์มาประยุกต์ใช้งานร่วมกับการเขียนโปรแกรม โดยนำภาพที่ได้จากกล้องส่งไปยังเครื่องคอมพิวเตอร์เพื่อทำการประมวลผล โดยการเปรียบเทียบความแตกต่างของภาพเพื่อหาตำแหน่งของวัตถุที่เคลื่อนไหวผ่านกล้อง และส่งคำสั่งไปยังส่วนควบคุมการหมุนของกล้องให้สามารถติดตามตำแหน่งของวัตถุได้โดยอัตโนมัติ

ABTRACT

This project presents automatic security system that use camera for tracking object. The microcontroller and programming is applied for this system. The image from the camera is transmitted to the computer for calculation and using comparison algorithm to indicate a position of object that moves pass the camera and send command to controlling camera's position to be able to track the object automatically.

สารบัญ

หน้า

สารบัญรูปภาพ

สารบัญตาราง

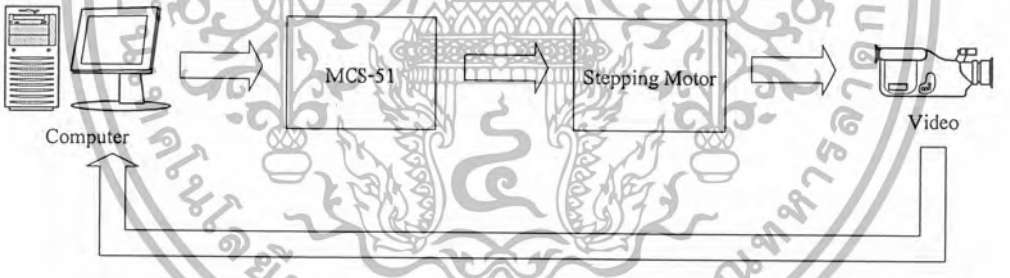
บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันงานทางด้านรักษาความปลอดภัยถือว่ามีความสำคัญมากสำหรับหน่วยงานต่างๆ จึงได้มีการคิดค้นระบบกล้องวีดีโอวงจรปิดขึ้นมาใช้ในระบบรักษาความปลอดภัย ซึ่งที่ใช้กันเป็นส่วนใหญ่นั้นยังเป็นแบบกล้องวีดีโอเพียงตัวเดียวที่ไม่สามารถเคลื่อนที่ได้ ภาพที่ได้ก็อยู่ในบริเวณที่จำกัดไม่สามารถครอบคลุมพื้นที่ได้เท่าที่ควร หรือไม่ก็มีการใช้กล้องวีดีโอหลายตัวเพื่อให้ครอบคลุมพื้นที่ได้ในบริเวณดังกล่าวได้ซึ่งเป็นการสิ้นเปลือง ดังนั้นในปริญญานิพนธ์ฉบับนี้จึงได้นำระบบดังกล่าวมาพัฒนาโดยนำกล้องวีดีโอขนาดเล็กราคาถูกมาประยุกต์ใช้กับคอมพิวเตอร์ส่วนบุคคล (Personal computer) ที่ในปัจจุบันก็มีราคาถูกลงมามาก โดยเราจะทำการเขียนโปรแกรมขึ้นมาเพื่อควบคุมกล้องวีดีโอให้สามารถบังคับทิศทางการจับภาพของกล้อง ซึ่งทำให้เราสามารถจับภาพได้ในบริเวณกว้างจากกล้องเพียงตัวเดียว รวมถึงการใช้ฮาร์ดดิส (Hard disk) ที่มีความคงทนในการเก็บบันทึกภาพ

1.2 ขอบเขตของปริญญานิพนธ์

ระบบรักษาความปลอดภัยแบบติดตามวัตถุอัตโนมัติ มีบล็อคไดอะแกรม (Block diagram) ดังแสดงในรูปที่ 1.1



รูปที่ 1.1 บล็อคไดอะแกรมการทำงาน

จากรูปที่ 1.1 ระบบรักษาความปลอดภัยแบบติดตามวัตถุอัตโนมัติประกอบด้วย 2 ส่วนหลัก คือ ส่วนฮาร์ดแวร์ (Hardware) และส่วนซอฟต์แวร์ (Software)

1. ส่วนฮาร์ดแวร์ ประกอบด้วยกล้องถ่ายภาพวีดิโอ 1 ตัวที่ทำหน้าที่รับภาพเข้ามา และระบบควบคุมการหมุนของกล้องให้สามารถในทิศทางหมุนขึ้น-ลง, ซ้าย-ขวา ตามการเคลื่อนที่ของวัตถุ

2. ส่วนซอฟต์แวร์ เป็นส่วนของโปรแกรมที่ติดตั้งอยู่บนเครื่องคอมพิวเตอร์ โดยจะรับสัญญาณภาพมาทำการประมวลผลหาความแตกต่างของภาพ แล้วก็ส่งออกไปทางพอร์ต RS-232 เพื่อส่งคำสั่งไปยังส่วนของฮาร์ดแวร์เพื่อไปควบคุมการเคลื่อนที่ของกล้องให้หมุนขึ้น-ลง, ซ้าย-ขวา และก็สามารถนำภาพจากกล้องมาแสดงผลและบันทึกได้

1.3 วิธีการดำเนินงาน

เริ่มจากการศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้อง ได้แก่ ศึกษาและหาข้อมูลในการดึงภาพจากกล้อง, ศึกษาการทำงานของไมโครคอนโทรลเลอร์ MCS-51, ศึกษาการเขียนโปรแกรมคอนโทรล (Control) ตัว MCS-51 เพื่อควบคุมสเต็ปปิ้งมอเตอร์ (Stepping Motor) และเพื่อติดต่อรับส่งข้อมูลกับคอมพิวเตอร์ผ่านพอร์ต RS-232 หลังจากที่ได้ศึกษาทฤษฎีต่างๆ แล้วก็นำทฤษฎีเหล่านั้นมาใช้งาน

ในส่วนของซอฟต์แวร์ก็ทำการเขียนโปรแกรมดึงภาพจากกล้องมาทำการเปรียบเทียบภาพเพื่อหาความแตกต่างของภาพระหว่างภาพอ้างอิงและภาพในเวลาต่อมาเพื่อหาตำแหน่งและทิศทางการเคลื่อนที่ของวัตถุ, เขียนโปรแกรมคอนโทรลตัว MCS-51 เพื่อควบคุมสเต็ปปิ้งมอเตอร์ และเพื่อติดต่อรับส่งข้อมูลกับคอมพิวเตอร์ผ่านพอร์ต RS-232

ในส่วนของฮาร์ดแวร์ทำการออกแบบระบบควบคุมบังคับการเคลื่อนที่ของกล้องในทิศทางการหมุนขึ้น-ลง, ซ้าย-ขวา ออกแบบวงจรส่วนประมวลผล MCS-51, ออกแบบวงจรขับสเต็ปปิ้งมอเตอร์

จากนั้นก็ลงมือทำระบบทั้งส่วนฮาร์ดแวร์ และซอฟต์แวร์ เสร็จแล้วก็ทำการทดสอบระบบ แก้ไขจุดบกพร่องที่พบจากการทดสอบ ท้ายสุดก็นำผลการทดลองมาทำการวิจารณ์และสรุป

บทที่ 2
ทฤษฎีและหลักการ

จากที่ได้กล่าวไปแล้วว่าระบบรักษาความปลอดภัยแบบติดตามวัตถุอัตโนมัตินั้นประกอบด้วย 2 ส่วนหลัก คือ ส่วนฮาร์ดแวร์ และส่วนซอฟต์แวร์ โดยทั้งสองส่วนนี้ต้องอาศัยทฤษฎีและหลักการที่เกี่ยวข้องดังนี้

1. ไมโครคอนโทรลเลอร์ MCS-51 ใช้ในส่วนวงจรรับสัญญาณอินพุตจากพอร์ต RS-232 และวงจรขับสัญญาณทางเอาต์พุตไปยังส่วนของสเต็ปปิ้งมอเตอร์

2. มาตรฐานพอร์ตอนุกรมแบบ RS-232 เป็นส่วนที่จะต้องนำมาใช้ในการอ้างอิงสัญญาณที่ต้องส่งข้อมูลผ่านทางพอร์ต RS-232 ของเครื่องคอมพิวเตอร์ไปยังวงจรควบคุมการเคลื่อนที่ของกล้อง

3. สเต็ปปิ้งมอเตอร์ เป็นส่วนที่ต้องใช้ในการขับเคลื่อนกล้อง โดยต้องทำการศึกษาในเรื่องของโครงสร้างและรูปแบบการทำงานของสเต็ปปิ้งมอเตอร์ รวมถึงการกระตุ้นและควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ ให้สามารถนำไปใช้ได้อย่างเหมาะสม

4. โมเดลสี เป็นส่วนที่ต้องนำมาใช้ในการอ้างอิงการแปลงภาพจากโมเดลสี RGB ให้อยู่ในรูปแบบภาพระดับเทาที่จะต้องนำไปใช้ในการเปรียบเทียบหาความแตกต่างของภาพ

5. ฟอร์แมตของภาพ เป็นส่วนที่ใช้ในการจัดเก็บรูปภาพ

2.1 ไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่บรรจุความสามารถมากมายไม่ว่าจะเป็นหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณทางเอาต์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกา ทำให้ไมโครคอนโทรลเลอร์สามารถนำไปประยุกต์ใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี โดยช่วยลดจำนวนของอุปกรณ์และขนาดของระบบลง ในขณะที่มีขีดความสามารถสูงขึ้นภายใต้งบประมาณที่เหมาะสม

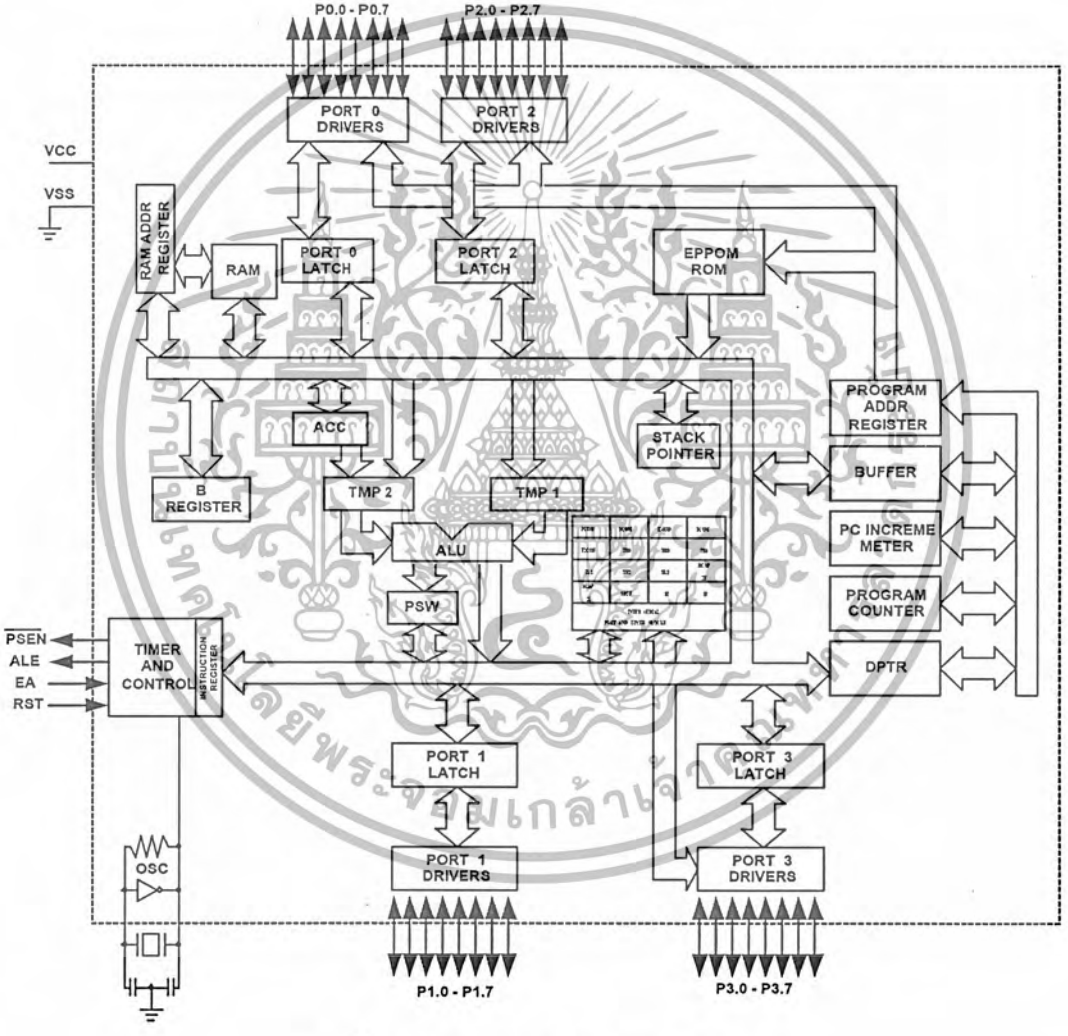
2.1.1 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ ตระกูล MCS-51

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบRAM ในบางเบอร์จะมีหน่วยความจำแบบ EEPROM เพิ่มเติม
- ขาพอร์ตเป็นแบบ 2 ทิศทาง สามารถใช้งานได้เป็นทั้งอินพุตและเอาต์พุต
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ (Full-duplex)
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิต อย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท

- มีวงจรกำเนิดสัญญาณนาฬิกาภายในชิป
- มีวงจรสื่อสารอนุกรมแบบ SPI
- มีวอตช์ด็อกไทเมอร์ในตัว

2.1.2 โครงสร้างของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51มีด้วยกันหลายเบอร์ขึ้นกับโครงสร้างภายใน บางเบอร์จะมีหน่วยความจำภายในเป็นแบบ ROM บางเบอร์เป็นแบบ EPROM บางเบอร์มี RAM ภายใน 128 ไบต์ บางเบอร์มี 256 ไบต์ เป็นต้น รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แสดงดังรูปที่ 2.1



รูปที่ 2.1 โครงสร้างภายในของ MCS-51

2.1.3 การจัดขาต่างๆ ของ MCS-51

ไอซี ไมโครคอนโทรลเลอร์ MCS-51 โครงสร้าง ไอซีเป็นแบบ DIP มีขาทั้งหมด 40 ขาโดยขาต่างๆจะใช้เป็นขาพอร์ตอินพุต,เอาต์พุต,ขาสัญญาณควบคุม,ขาตำแหน่งหน่วยความจำ และขาข้อมูลดังรูปที่ 2.2

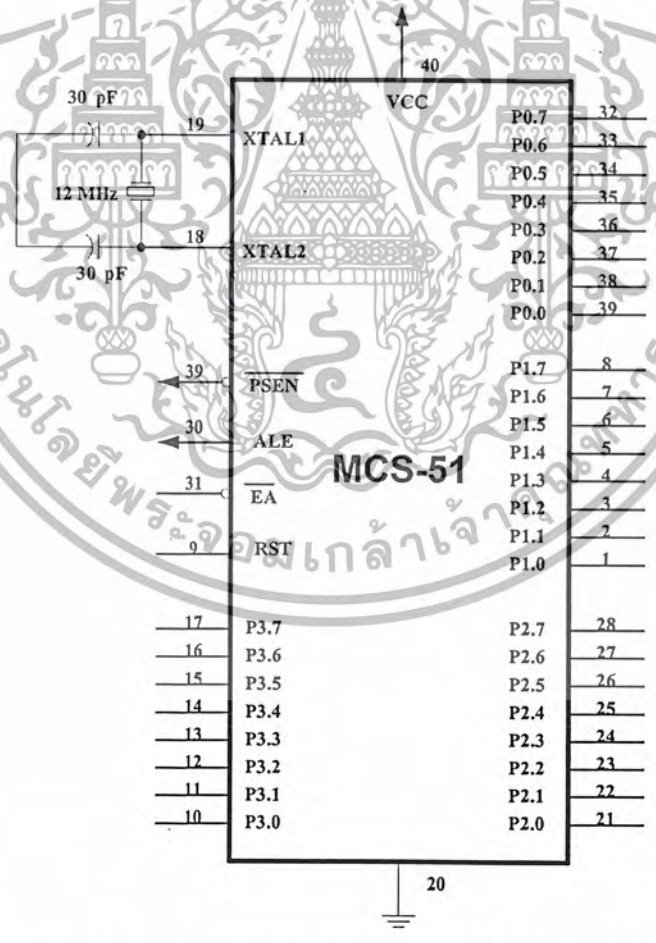
ความหมายของขาต่างๆมีดังนั้น

2.1.3.1 พอร์ต 0 (Port 0) ได้แก่ขาที่ 32-39 ของ MCS-51 สามารถใช้เป็นอินพุตเอาต์พุตได้ นอกจากนี้ในการติดต่อกับหน่วยความจำภายนอกยังใช้เป็นขา Address Bus และ Data Bus อีกด้วย

2.1.3.2 พอร์ต 1 (Port 1) ได้แก่ขาที่ 1-8 เป็นพอร์ต 8 บิต สามารถอ้างทีละบิตได้คือ P1.0,P1.1,...,P1.7

2.1.3.3 พอร์ต 2 (Port 2) ได้แก่ขาที่ 21-28 จะใช้งาน 2 หน้าที่คือใช้เป็นพอร์ต 8 บิตกับใช้เป็นขาแอดเดรส 8 บิตในการอ้างหน่วยความจำภายนอก

2.1.3.4 พอร์ต 3 (Port 3) ได้แก่ขาที่ 10-17 จะใช้งานสองหน้าที่คือ เป็นพอร์ตอินพุตและเอาต์พุต และใช้เป็นขาควบคุมต่างๆดังตารางที่ 2.1



รูปที่ 2.2 ขาต่างๆ ของไมโครคอนโทรลเลอร์ MCS-51

ตารางที่ 2.1 หน้าที่ของขาต่างๆ ในพอร์ต 3

บิต	ชื่อ	หน้าที่พิเศษ
P3.0	RXD	ใช้รับข้อมูลทางพอร์ตอนุกรม
P3.1	TXD	ใช้ส่งข้อมูลทางพอร์ตอนุกรม
P3.2	INT0	อินเทอร์รัปต์ภายนอกหมายเลข 0
P3.3	INT1	อินเทอร์รัปต์ภายนอกหมายเลข 1
P3.4	T0	ตัวจับเวลา/ตัวนับ ตัวที่ 0
P3.5	T1	ตัวจับเวลา/ตัวนับ ตัวที่ 1
P3.6	WR	สัญญาณเขียนข้อมูลหน่วยความจำภายนอก
P3.7	RD	สัญญาณอ่านข้อมูลหน่วยความจำภายนอก

2.1.3.5 PSEN (Program Store Enable) ขา PSEN เป็นขาที่ส่งสัญญาณออก ได้แก่ขา 29 ขานี้จะแอคทีฟเมื่อ MCS-51 ต้องการอ่าน Code โปรแกรมภายนอก โดยปกติถ้าหน่วยความจำภายนอกเป็น EPROM ขา PSEN จะต่อกับขา Output Enable (OE) ของ EPROM

2.1.3.6 ALE (Address Latch Enable) เนื่องจากพอร์ต 0 สามารถใช้เป็นขาอ้างตำแหน่งและขาข้อมูล MCS-51 จึงมีขา ALE ได้แก่ขา 30 Multiplex สัญญาณ Address Bus ของพอร์ต 0 ในการใช้งานระบบ MCS-51 นั้นจำเป็นต้องมีอุปกรณ์มาต่อกับพอร์ต 0 ที่ทำหน้าที่ Latch สัญญาณ Address Bus เมื่อ MCS-51 ต้องการติดต่อกับหน่วยความจำภายนอก MCS-51 จะส่งสัญญาณ Address Bus ออกมาก่อนทางพอร์ต 0 จากนั้นจะส่งสัญญาณ ALE มา Latch อุปกรณ์ภายนอก ให้เก็บค่า Address Bus ของพอร์ต 0 ไว้เพื่อใช้พอร์ตเป็น Data Bus ต่อไป

2.1.3.7 EA (External Access) ได้แก่ขาที่ 31 ถ้าขานี้เป็นลอจิก "1" จะใช้เบอร์ 8051/8052 เพื่อบอกว่าให้อ่านโปรแกรมจากหน่วยความจำโปรแกรมภายใน แต่ถ้าเป็นลอจิก "0" จะบอกว่าให้ MCS-51 ทำโปรแกรมโดย อ่านจากหน่วยความจำโปรแกรมภายนอก (ถ้าขา EA เป็น "0" ขา PSEN จะแอคทีฟ) ถ้าหากเป็นเบอร์ 8031 หรือ 8032 ขา EA จะเป็น "0"เสมอ เพราะว่าไม่มีโปรแกรมหน่วยความจำภายใน แต่ถ้าใช้เบอร์ 8051/8052ซึ่งมีหน่วยความจำโปรแกรมภายในและให้ขา EA เป็น "0" ซึ่งจะ Disabled ROM ภายในและจะอ่านโปรแกรมจาก EPROM ภายนอกแทน

2.1.3.8 RST (Reset) ได้แก่ขา 9 จะใช้ในการรีเซต MCS-51 โดยจะให้ขานี้เป็นลอจิก "1" อย่างน้อย 2 Machine Cycles จึงจะรีเซตระบบได้

2.1.4 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรเลอร์ตระกูล MCS-51 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด (วงจรสื่อสารแบบฟูลดูเพล็กซ์ หมายถึง วงจรสื่อสารที่สามารถทำการรับและส่งข้อมูลในลักษณะ 2 ทิศทาง ได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสาร ข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51

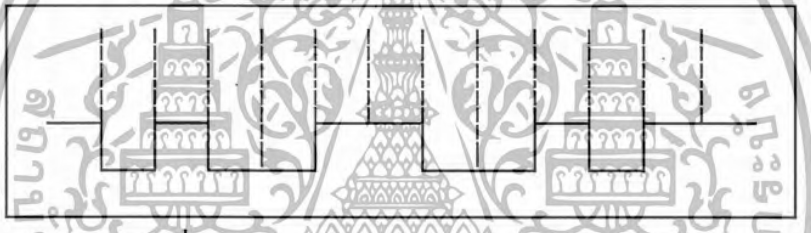
แบบแฟลชเป็นแบบอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 แต่ในปัจจุบันสามารถติดต่อกันในมาตรฐาน RS-422 หรือ RS-485 ได้แล้ว โดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

2.1.4.1 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน เรียกอัตราเร็วนี้ว่า อัตราบอด หรือ บอดเรด (Buad Rate) มีหน่วยเป็น บิตต่อวินาที (Bit Per Second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (Start Bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรม มีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity Bit) มีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด (Stop Bit) มีขนาด 1 บิต



รูปที่ 2.3 รูปแบบข้อมูลอนุกรมแบบอะซิงโครนัส

รูปที่ 2.3 แสดงรูปแบบของข้อมูล อนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูล ขา Data จะมีสถานะลอจิก “1” เรียกสถานะนี้ว่า สถานะหยุดรอ (Waiting Stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา Data มีลอจิก “0” ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ข้อมูลที่ต้องการส่งมีจำนวน 8 บิต ตามด้วยบิตพาริตี้ ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือ บิตปิดท้ายหรือ บิตหยุด โดยจะเป็นการทำให้ขา Data มีสถานะลอจิก “1”อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต ,1.5บิต หรือ 2บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรดที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่าได้แก่ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ เนื่องจากอัตราบอดคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมติว่า ข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูล 1 ไบต์จะเท่ากับความยาว 10 บิต ถ้าใช้บอดเรดในการส่งข้อมูลเท่ากับ 9,600 บิตต่อวินาที ก็สามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd) แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบแบบพาริตี้คี่(หรือคู่)คือ การนับจำนวนลอจิก "1" ของข้อมูลขนาด 1 ไบต์และของบิตพาริตี้ว่ามีจำนวนรวมเป็นเลขคี่(หรือคู่)หรือไม่ ถ้าใช่แสดงว่าข้อมูลที่ส่งมาถูกต้อง ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99H หรือ 1001 1001B จะเห็นได้ว่าข้อมูลในไบต์นี้มีจำนวนลอกจิกเป็น "1" จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดการตรวจสอบค่าพาริตี้เป็นแบบคู่ ค่าของบิตพาริตี้จะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดการตรวจสอบพาริตี้เป็นแบบคี่ ค่าของบิตพาริตี้จะต้องเป็น "1" ข้อมูล 1 ไบต์รวมทั้งในบิตพาริตี้มีจำนวนรวมเป็นคี่

บิตพาริตี้จะถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART (Universal Asynchronous Receiver Transmitter : เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก 1 ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตที่มีการผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตให้เป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

2.1.4.2 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน MCS-51

ในการทำงานของวงจรพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 มีรีจิสเตอร์ที่ต้องเกี่ยวข้องอยู่ 2 ตัว มีรายละเอียดดังต่อไปนี้

1. รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial Data Buffer Register)

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต แบ่งเป็น 2ส่วนคือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (receiver buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือ P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

2. รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial Port Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิตมีแอดเดรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SM0	SM1	SM2	SM3	SM4	SM5	SM6	SM7

รูปที่ 2.4 รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม

SMO-SMI (Serial Port Mode Bit 0-1): ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

SM2 : ใช้ในการอีนาเบิลการสื่อสารในแบบ มัลติโปรเซสเซอร์ (Multiprocessor) ในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 ในโหมด2และ3 ถ้าบิตนี้เป็น "1" บิต RI จะไม่แอคทีฟถ้าบิตที่ 9 ที่รับเข้ามาเป็น "0" (ข้อมูลบิตที่ 9 เก็บไว้ที่บิต RB8) ในการทำงานโหมด 1 ถ้าบิตนี้ถูกเซตบิต RI จะไม่แอคทีฟถ้ายังไม่รับบิตหยุด ส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน

REN (Enable Serial Reception): ใช้ในการอีนาเบิลการรับข้อมูลพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการรับข้อมูลต้องเซตบิตนี้ให้เป็น "1"

TB8 : ใช้สำหรับเก็บข้อมูลที่บิต 9 ที่ต้องการส่งออกไปขณะทำงานในโหมด 2 และ3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 เซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

RB8 : ใช้สำหรับข้อมูลบิตที่ 9 ที่เข้ามาขณะทำงานในโหมด 2และ3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แต่ถ้าหากพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น "0" ข้อมูลที่บิต RB8 คือข้อมูลของบิตหยุด สำหรับการทำงานในโหมด 0 บิตนี้จะไม่ใช้งาน บิต RB8 นี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

TI (Transmit Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์เมื่อทำการส่งข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 ส่วนการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

RI (Receive Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการรับบิตข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 ส่วนการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการรับบิตหยุดข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นกรณีที่บิต SM2 มีการเซต บิตนี้จะเซตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

2.1.4.3 โหมดการทำงานของพอร์ตอนุกรมใน MCS-51

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเลือกโหมดการทำงานได้ถึง 4 โหมดคือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์
2. โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตรบอดได้

3.โหมด 2 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราบอดคงที่

4.โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดการทำงานของวงจรพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 กระทำได้โดยการกำหนดข้อมูลให้แก้บิต SM0 และ SM1 ในรีจิสเตอร์ SCON

2.1.4.4 อัตราบอดของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51

1.โหมด 0 อัตราบอดของโหมดนี้มีค่าคงที่ โดยสามารถคำนวณได้จากสูตร

อัตราบอดในโหมด 0 = ความถี่ของสัญญาณนาฬิกา/12 มีหน่วยเป็นบิตต่อวินาที

2.โหมด 1 และ 3 ทั้งสองโหมดนี้สามารถเลือกแหล่งกำเนิดอัตราบอดได้ 2แหล่ง คือ จากอัตราโอเวอร์โฟลวของไทเมอร์ 1 และ 2 สำหรับอัตราบอดเมื่อใช้การโอเวอร์โฟลวของไทเมอร์ 1 จะต้องใช้ค่าของบิต SMOD ใน รีจิสเตอร์ PCON มาพิจารณาประกอบด้วย โดยสามารถหาค่าอัตราบอดได้จาก

$$\text{อัตราบอด} = (2^{\text{ค่าของบิต SMOD}}/32) \times \text{อัตราโอเวอร์โฟลวของไทเมอร์ 1}$$

ถ้าหากในไทเมอร์ 1 ไม่ได้อีนาเบิลการอินเตอร์รัปต์ไว้ สามารถคำนวณหาค่าอัตราบอดได้จาก

$$\text{อัตราบอด} = (2^{\text{ค่าในรีจิสเตอร์ SMOD}}/32) \times (\text{ความถี่สัญญาณนาฬิกา}/\{12x[256-(TH1)]\})$$

ตารางที่ 2.2 แสดงการกำหนดอัตราบอดโดยใช้ไทเมอร์ 1

ตารางที่ 2.2 การเลือกอัตราบอดของวงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

อัตราบอด	ความถี่สัญญาณนาฬิกา	SMOD	ไมเมอร์ 1		
			C/T	โหมด	ค่ารีโหลด
โหมด 0:สูงสุด1MHz	12 MHz	x	x	x	x
โหมด 2 : สูงสุด 375K	12 MHz	1	x	x	x
โหมด 1,3 : 62.5K	12 MHz	1	0	2	FFH
19.2K(19200)	11.0592 MHz	1	0	2	FDH
9.6K(9600)	11.0592 MHz	0	0	2	FDH
4.8K(4800)	11.0592 MHz	0	0	2	FAH
2.4K(2400)	11.0592 MHz	0	0	2	F4H
1.2K(1200)	11.0592 MHz	0	0	2	E8H
137.5	11.0592 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FBBEH

กรณีที่ใช้ไทเมอร์ 2 ในการกำหนดอัตราบอด โดยกำหนดให้ไทเมอร์ 2 ทำงานในโหมดกำเนิดอัตราบอด (Baud rate generator) สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = อัตราโอเวอร์โฟลวของไทเมอร์ 2/16 มีหน่วยเป็นบิตต่อวินาที

ถ้าหากกำหนดให้ไทเมอร์ 2 ทำงานในโหมดไทเมอร์หรือเคาน์เตอร์ตามปกติ สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = ความถี่สัญญาณนาฬิกา/(32x(65536-(RCAP2H,RCAP2L)))

โดยที่ (RCAP2H, RCAP2L) เป็นค่าของรีจิสเตอร์ RCAP2H และ RCAP2L มีขนาด 16 บิตไม่คิดเครื่องหมาย

3. โหมด 2

ในโหมดนี้การกำหนดอัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON ถ้า SMOD เป็น "0" อัตราบอดจะเท่ากับ 1/64 ของความถี่สัญญาณนาฬิกา ในกรณีที่ SMOD เป็น "1" อัตราบอดจะเท่ากับ 1/32 ของความถี่สัญญาณนาฬิกา สามารถแสดงเป็นสูตรคำนวณทางคณิตศาสตร์ได้ดังนี้

อัตราบอด = $(2^{\text{ค่าของบิต SMOD}}/64)$ x ความถี่สัญญาณนาฬิกา

2.1.4.5 การกำหนดค่าของไทเมอร์เพื่อเลือกอัตราบอด

ในการใช้งานพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สิ่งที่ต้องให้ความสนใจมากที่สุดประการหนึ่งคือ อัตราการถ่ายทอดข้อมูล หรือ อัตราบอด ซึ่งการกำหนดอัตราบอดนั้นจะขึ้นอยู่กับค่าความถี่สัญญาณนาฬิกาเป็นหลัก สำหรับโหมดการทำงานของพอร์ตอนุกรมที่สามารถเลือกอัตราบอดได้อย่างอิสระคือโหมด 1 และ 3 โดยกำหนดได้จากอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 หากไทเมอร์ 1 มีการเกิดโอเวอร์โฟลวในอัตราที่สูงมากอัตราบอดก็จะมีค่าสูงมากขึ้นตาม นั่นหมายความว่า อัตราในการถ่ายทอดข้อมูลจะสูงมาก สามารถถ่ายทอดข้อมูลได้อย่างรวดเร็ว ดังนั้นการกำหนดค่าและโหมดการทำงานของ ไทเมอร์ 1 จึงเป็นสิ่งที่มีผลต่อการเปลี่ยนแปลงของอัตราบอดด้วย

ในการใช้ไทเมอร์ 1 เพื่อกำหนดอัตราบอดในโหมด 1 และ3 ของพอร์ตอนุกรมจะต้องกำหนดให้ไทเมอร์ทำงานในโหมด 2 หรือโหมด8 บิตแบบตั้งค่าการนับอัตโนมัติ ในกรณีการกำหนดค่ารีโหลดให้แก่รีจิสเตอร์ TH1 จะเป็นตัวแปรหลักที่ใช้ในการกำหนดอัตราบอดให้แก่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

เริ่มต้นด้วยการเคลียร์บิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON ให้เป็น "0" ค่าของการรีโหลดให้แก่ TH1 สามารถคำนวณได้จาก

TH1 = 256-((ค่าความถี่ของคริสตอล/384)/อัตราบอด)

แต่ถ้าบิต SMOD เกิดการเซต จะเป็นการอีนาเบิลการทวีคูณของอัตราบอด ดังนั้นการกำหนดค่าให้แก่ TH1 จึงต้องคำนวณจาก

TH1 = 256-((ค่าความถี่ของคริสตอล/192)/อัตราบอด)

ยกตัวอย่าง ถ้าหากในไมโครคอนโทรลเลอร์ AT89C51 ใช้คริสตอล 11.0592 MHz ต้องการกำหนดอัตราบอดของพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ไว้ที่ 19200 บิตต่อวินาที ในกรณีที่ไม่อีนาเบิลการทวีคูณของอัตราบอด ค่ารีโหลดของไมโครคอนโทรลเลอร์จะเท่ากับ

TH1 = 256-((ค่าความถี่ของคริสตอล-384)/อัตราบอด)

= 256-((11059200/384)/19200)

= 256-(28800/19200)

= 256-1.5=254.5

เนื่องจากผลลัพธ์ที่ได้เป็นค่าที่ไม่ใช่จำนวนเต็ม ถ้าหากกำหนดค่าของ TH1 เป็น254 เมื่อทำการแทนค่าเพื่อคำนวณหาค่าอัตราค่าบอด จะได้อัตราบอดเท่ากับ 14400 บิตต่อวินาที และถ้าหากกำหนดค่าของ TH1 เป็น 255 อัตราบอดจะมีค่าเท่ากับ 28800 บิตต่อวินาที ดังนั้นจะเห็นได้ว่าค่าของ TH1 ที่ไม่เป็นจำนวนเต็มจะไม่สามารถทำให้เกิดอัตราบอดตามต้องการได้

ทางแก้ไขคือ ให้ทำการอีนาเบิลการทวีคูณอัตราบอด โดยการเซตบิต SMOD ในรีจิสเตอร์ PCON ให้เป็น "1"จากนั้น ทำการแทนค่าลงในสมการหาค่า TH1 เมื่อมีการเซตบิต SMOD ได้ผลลัพธ์ดังนี้

TH1 = 256-((ค่าความถี่ของคริสตอล/192)/อัตราบอด)

= 256-((11059200/192)/19200)

= 256-(57600/19200)

= 256-3 = 253

ค่าของ TH1 ที่ได้ทำการแทนค่าคำนวณหาค่าอัตราบอดจะได้เท่ากับ 19200บิตต่อวินาที

สามารถสรุปขั้นตอนในการเลือกอัตราบอดโดยการกำหนดค่าของไทเมอร์ 1 ได้ดังนี้

1. กำหนดให้พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 ทำงานในโหมด1หรือ3

2. กำหนดให้ไทเมอร์ 1 ทำงานในโหมด 2 หรือโหมด 8บิตตั้งค่าอัตโนมัติ

3. กำหนดข้อมูลให้แก่ TH1 เท่ากับ 253 เพื่อให้สามารถกำเนิดอัตราบอดได้ 19200 บิตต่อวินาที ตามที่ต้องการ

4. ทำการเซตบิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON เพื่ออีนาเบิลการทวีคูณอัตราบอด

2.1.4.6 การเขียนหรือส่งข้อมูลออกจากพอร์ตอนุกรม

ข้อมูลที่ต้องการส่งออกทุกค่าต้องนำไปเก็บไว้ในรีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรม ซึ่งก็คือ รีจิสเตอร์ SBUF ดังตัวอย่าง

```
MOV SBUF ,# ' A '
```

จากคำสั่งข้างต้นเป็นการส่งข้อมูลของตัวอักษร A ออกไปยังพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ อย่างไรก็ตามก่อนที่จะส่งข้อมูลทุกครั้ง ต้องแน่ใจว่าบิต TI เคลียร์ หรือมีค่าเป็น "0" และเมื่อทำการส่งข้อมูลเรียบร้อยแล้ว ก็จะเกิดการเซตบิต TI เพื่อแจ้งให้ทราบ ดังตัวอย่างโปรแกรมต่อไปนี้

```
CLR TI              ;เคลียร์บิต TI เพื่อเตรียมการส่งข้อมูลออก
MOV SBUF,#'A'       ;ส่งข้อมูลของตัวอักษร A ไปยังพอร์ตอนุกรม
JNB TI,S            ;รอการเซตของบิต TI เพื่อแจ้งการส่งข้อมูลที่เสร็จสมบูรณ์
```

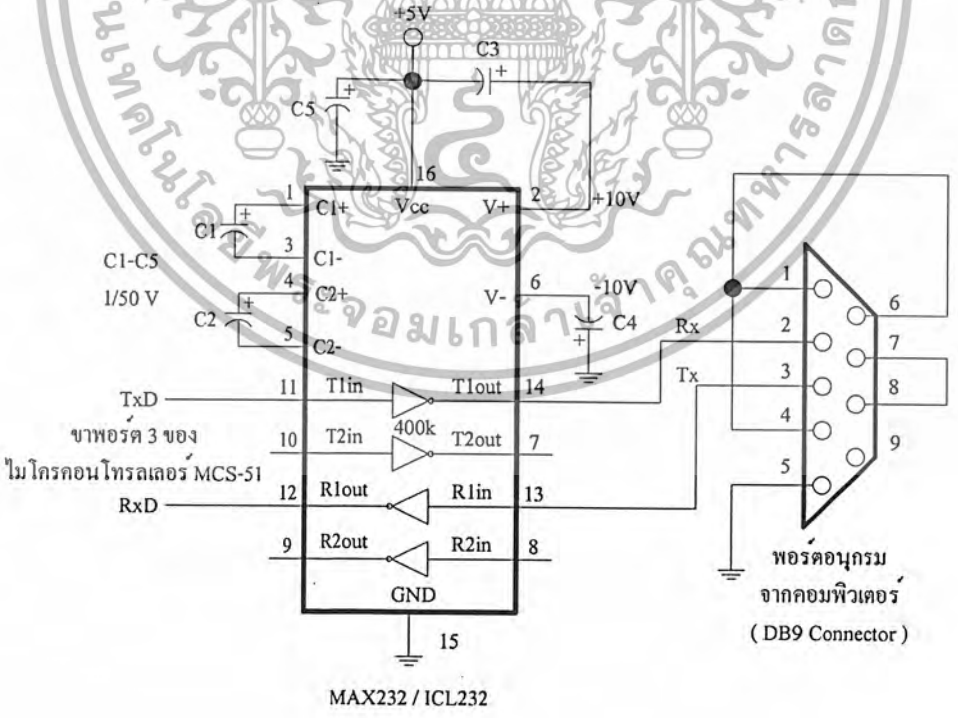
2.1.4.7 การอ่านหรือรับข้อมูลจากพอร์ตอนุกรม

การรับข้อมูลจากพอร์ตอนุกรมสามารถกระทำได้ง่ายมาก เพียงแค่ทำการตรวจสอบว่าบิต RI เกิดการเซตขึ้นหรือไม่ ถ้าพบว่ามีการเซตเกิดขึ้นแล้ว ให้ทำการอ่านค่าจากรีจิสเตอร์ SBUF โดยต้องทำการโอนย้ายข้อมูลผ่านทางแอกคิวมูเลตอร์หรือรีจิสเตอร์ A ดังตัวอย่าง

```
CLR  RI          ;เคลียร์บิต RI เพื่อเตรียมการรับข้อมูล
JNB RI,$         ;รอคอยการเซตของบิต RIอันเป็นการแจ้งให้ทราบว่าการรับ
                  ข้อมูลเสร็จสมบูรณ์และมีข้อมูลที่เกิดขึ้นที่รีจิสเตอร์SBUF
MOV  A,SBUF      ;อ่านค่าจากรีจิสเตอร์โดยการโอนย้ายข้อมูลผ่านทาง
                  รีจิสเตอร์ A
CLR  RI          ;หลังจากทำการอ่านข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์
                  บิตRIเสมอ
```

2.1.4.8 การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ 3 ถึง 12 โวลต์ ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่แปลงระดับสัญญาณ



รูปที่ 2.5 วงจรเชื่อมต่อ MAX232 หรือ ICL232 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์

ไอซีที่ทำหน้าที่ในการแปลงระดับสัญญาณนี้จะต้องทำการแปลงข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลที่รับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่าวมีด้วยกันหลายเบอร์จากผู้ผลิต อาทิ MAX232 จาก MAXIM หรือ ICL232 จาก HARRIS เป็นต้น ในรูปที่ 2.5 แสดงส่วนวงจรของการต่อกับพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์

2.2 มาตรฐานพอร์ตอนุกรมแบบ RS-232

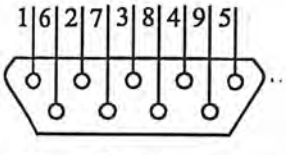
มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (EIA) ได้วางมาตรฐานที่มีชื่อเรียกว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3 ถึง -12 โวลต์แสดงว่ามีข้อมูล และ +3 ถึง +12 โวลต์แสดงว่าเป็นช่องว่าง มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับ วงจรข้อมูลปลายทาง (Data circuit Terminal : DCE) ได้ว่าอุปกรณ์ดีทีอี (DTE) จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์ หรือ ไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ดีซีอี (DCE) จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจากทางดีทีอี เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองผ่านทางมาตรฐาน RS-232

ข้อแตกต่างของอุปกรณ์ ดีทีอี และอุปกรณ์ ดีซีอี อย่างหนึ่งที่เห็นได้ชัดเจนคือ คอนเน็กเตอร์ของดีทีอี จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ ดีซีอี จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ ใช้กันอยู่ทั่วไปจะเป็นแบบ ดีทีอี ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ ดีซีอี

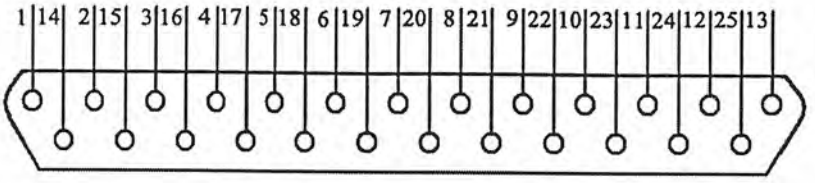
สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็ม หรือเมาส์ โดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุดถึง 20 เมตร

2.2.1 คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้ หรือ DB-9 ตัวผู้ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่น ๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 2.6



รูปที่ 2.6 การจัดขาของคอนเน็กเตอร์พอร์ตอนุกรมแบบ DB-9 ตามมาตรฐาน RS-232



รูปที่ 2.7 การจัดขาของคอนเน็กเตอร์พอร์ตอนุกรมแบบ DB-25 ตามมาตรฐาน RS-232

สำหรับรายละเอียดของขาคอนเน็กเตอร์นั้นมีดังนี้

1. ดาต้า แคเรีย ดีเทค (Data Carrier Detect : DCD) หรืออาจเรียกว่า แคเรียดีเทค (Carrier Detect : CD) ขานี้จะแอคทีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็มสำหรับการใช้งานปกติ ขานี้จะไม่ถูกใช้งานมากนัก

2. ขารับข้อมูล (Receive Data) หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์บัฟเฟอร์

3. ขาส่งข้อมูล (Transmitted Data) หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

4. ดาต้า เทอร์มินอล เรดดี้ (Data Terminai Ready : DTR) เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทางและ ขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อเป็นแบบ นูลโมเด็ม ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์

5. ซิกแนล กราวด์ (Signal Ground : GND) ขากราวด์ของระบบ

6. ดาต้า เซต เรดดี้ (Data Set Ready : DSR) ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

7. รีเควส ทู เซนด์ (Request To Rend : RTS) เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือ ขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ นูลโมเด็ม 3 เส้น จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกันเพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

8. เคลียร์ ทู เซนด์ (Clear To Send : CTS) ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่

9. ริง อินดิเคเตอร์ (Ring Indicator : RI) ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

ตารางที่ 2.3 สถานะของสัญญาณกับแรงดันมาตรฐานของ RS-232

VOTAGE RANGE	LOGIC	MEANING	TTY	SYNOYMS
+3V TO +25V	0	ON	SPACE	ACTIVE
-25 TO -3V	1	OFF	MARK	INACTIVE

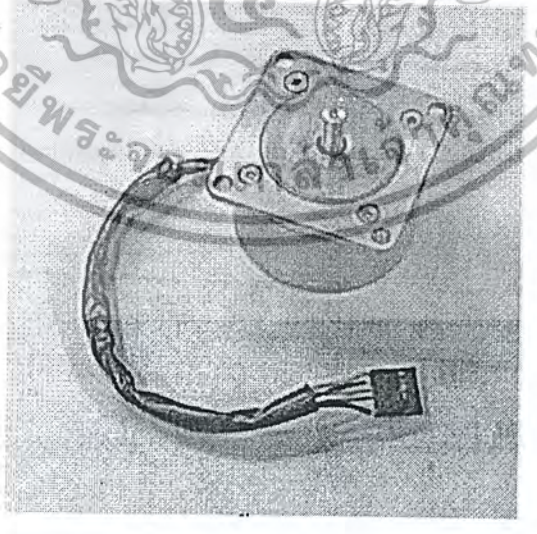
จากตารางที่ 2.3 สัญญาณที่ขาทุกขาของคอนเน็คเตอร์ RS-232 จะเป็นสภาวะใดสภาวะหนึ่ง

2.3 สเต็ปปิ้งมอเตอร์

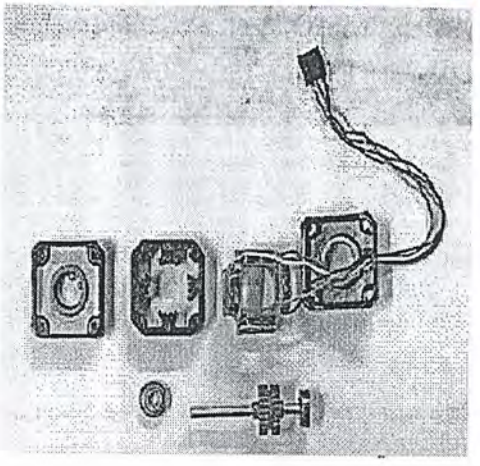
สเต็ปปิ้งมอเตอร์ เป็นมอเตอร์ที่ขับเคลื่อนด้วยพัลส์ ลักษณะการขับเคลื่อน จะหมุนรอบแกนได้ 360 องศา มีลักษณะไม่ต่อเนื่อง แต่มีลักษณะเป็นสเต็ป โดยแต่ละสเต็ปจะขับเคลื่อนได้ 1,1.5,1.8 หรือ 2 องศา แล้วแต่โครงสร้างของมอเตอร์และลักษณะที่นำมอเตอร์ไปใช้ ตัวอย่างงานที่ต้องการตำแหน่งแม่นยำ เช่น ระบบขับเคลื่อนหัวแม่พิมพ์ในเครื่องพิมพ์ (Printer) ระบบขับเคลื่อนหัวอ่านในเครื่องอ่านบันทึกเหล็ก ระบบขับเคลื่อนตำแหน่งของปากกาใน X-Y PLOTER เป็นต้น

2.3.1 ลักษณะและโครงสร้างของสเต็ปปิ้งมอเตอร์

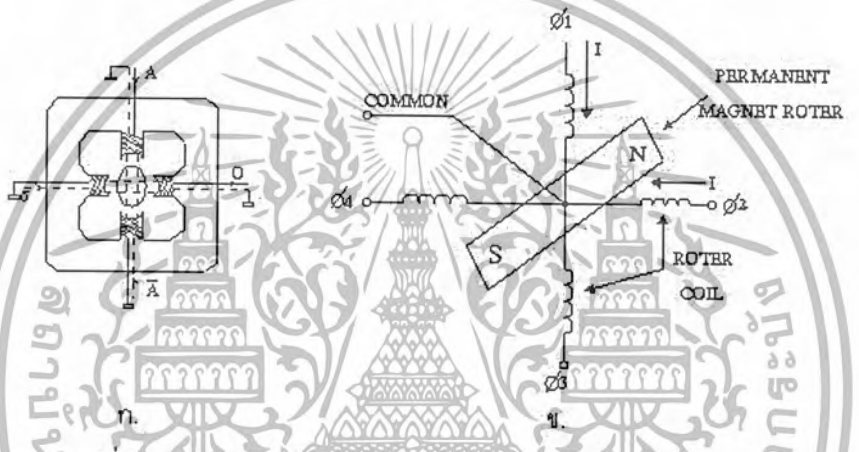
สเต็ปปิ้งมอเตอร์มีลักษณะดังรูป



รูปที่ 2.8 สเต็ปปิ้งมอเตอร์แบบมีสาย 5 เส้น



รูปที่ 2.9 โครงสร้างของสเต็ปปิ้งมอเตอร์



รูปที่ 2.10 (ก) โครงสร้าง (ข) วงจรเทียบเท่าของมอเตอร์ ชนิด 4 ขด

2.3.2 ชนิดของสเต็ปปิ้งมอเตอร์

2.3.2.1 แบบแม่เหล็กถาวร (Permanent Magnet-PM) สเต็ปปิ้งมอเตอร์แบบ PM จะมีสเตเตอร์ (Stator) ที่พันขดลวดไว้หลายๆโพล โดยมีโรเตอร์ (Rotor) เป็นรูปทรงกระบอกฟันเลื่อย และโรเตอร์ทำด้วยแม่เหล็กถาวร เพื่อป้อนไฟกระแสตรงให้กับขดสเตเตอร์ทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์และ ทำให้มอเตอร์หมุน มอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้โรเตอร์หยุดอยู่กับที่ แม้จะไม่ได้ป้อนไฟเข้าขดลวด

2.3.2.2 แบบแปรค่ารีลักแตนซ์ (Variable Reluctance-VR) สเต็ปปิ้งมอเตอร์แบบ VR จะมีการหมุนโรเตอร์ได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟให้โรเตอร์ ทำจากสารเฟอร์โรแมกเนติกกำลังอ่อน มีลักษณะเป็นฟันเลื่อย รูปทรงกระบอก โดยจะมีความสัมพันธ์โดยตรงกับจำนวนโพลในสเตเตอร์ แรงบิดที่เกิดขึ้นจะหมุนโรเตอร์ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนซ์ต่ำที่สุด ตำแหน่งที่จะเกิดแน่นอนและมีเสถียรภาพแต่จะเกิดขึ้นได้หลายๆ จุดดังนั้นเมื่อป้อนไฟเข้าขดลวดต่างๆ ในมอเตอร์แตกต่างขดกันไป ก็ทำให้มอเตอร์หมุนไปตำแหน่งต่างๆกัน โรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM

2.3.2.3 แบบผสม (Hybrid-H) สเต็ปปิ้งมอเตอร์แบบ H จะเป็นลูกผสมของ VR กับ PM โดยจะมีสเตเตอร์คล้ายกับที่ใช้ใน VR โรเตอร์มีหมวกหุ้มปลาย ซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูง โดยการควบคุมขนาดรูปร่างของหมวกแม่เหล็กอย่างดีจะทำให้ได้มุม การหมุนและจำนวนครั้งในการหมุนน้อยและแม่นยำข้อดีก็คือให้แรงบิดสูง มีขนาดกระทัดรัดและให้แรงฉุดยึดโรเตอร์นิ่งกับที่ตอนไม่จ่ายไฟ

2.3.3 การกระตุ้นและควบคุมการหมุนของสเต็ปปิ้งมอเตอร์

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบซีเควนเชียล (Sequential) ในรูปแบบที่ถูกต้องด้วย สามารถแบ่งได้เป็น 3 รูปแบบคือ

2.3.3.1 แบบหนึ่งเฟสหรือแบบฟูลสเต็ป (Full Step) เป็นการกระตุ้นโดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งไล่เรียงถัดกันไป ทำให้ในการกระตุ้นรูปแบบนี้จึงมีขดลวดเพียงขดลวดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น วงจรแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ 2.4

ตารางที่ 2.4 การควบคุมสเต็ปปิ้งมอเตอร์แบบหนึ่งเฟสหรือแบบฟูลสเต็ป (Full Step)

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

2.3.3.2 แบบ 2 เฟส เป็นการกระตุ้นคล้ายแบบหนึ่งเฟส แต่จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดไปเช่นเดียวกับแบบเวฟ การกระตุ้นสเต็ปปิ้งมอเตอร์แบบนี้สามารถเพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกัน และต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือต้องใช้กำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ 2.5

ตารางที่ 2.5 การควบคุมสเต็ปปิ้งมอเตอร์แบบ 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

2.3.3.3 แบบครึ่งสเต็ป เป็นรูปแบบการผสมผสานระหว่างการกระตุ้นแบบเวฟ และแบบ 2 เฟส เพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นเรียงกันไปเป็นลำดับดังนี้ เริ่มจากขดลวดที่ 1, 1และ2 ,2, 2และ3 ,3, 3และ4 ,4, 4และ1 แล้ววนกลับมายังขดลวดที่ 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเต็ปมีระยะสั้นลง แต่ละสเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องพึงระวังเรื่อง เมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องหมุนถึง 2 สเต็ป จึงจะได้ระยะเท่ากับ 1 สเต็ปเต็มของการควบคุมใน 2 แบบแรก สำหรับแหล่งจ่ายไฟฟ้าต้องใช้ขนาดเท่ากับแบบ 2 เฟสเป็นอย่างน้อยจึงจะพอ ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ 2.6

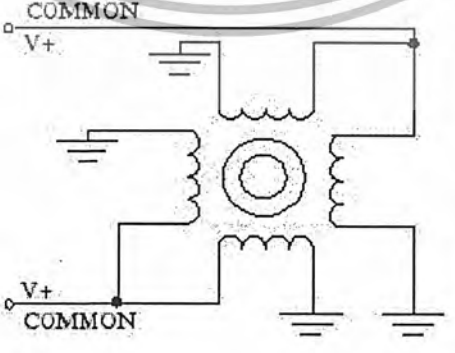
ตารางที่ 2.6 การควบคุมสเต็ปปิ้งมอเตอร์แบบครึ่งสเต็ป

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน

2.3.4 การตรวจสอบหาสาย Common และสาย Ground ของสเต็ปปิ้ง แบบ PM (แบบแกนโรเตอร์เป็นแม่เหล็กถาวร)

โดยทั่วไป SP Motor แบบ PM จะมีอยู่ 2 ชนิด

1. ชนิดที่เป็น Common ภายนอก SP Motor แบบนี้มีสายอยู่ 6 เส้น ดังรูปที่ 2.11

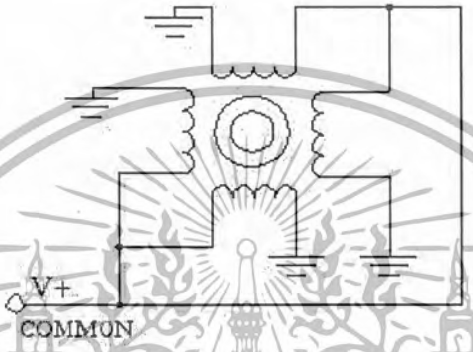


รูปที่ 2.11 สเต็ปปิ้งมอเตอร์ ชนิดมีสาย 6 เส้น

จากรูปที่ 2.11 สายที่เป็น Common มี 2 สาย และสายที่เป็น Ground มี 4 สาย

สาย Common 1 เส้น จะ Drive Ground 2 เส้น ในการเช็คให้ใช้มิเตอร์วัดหาสายที่เป็น Common ก่อนโดยการตั้ง Range ของมิเตอร์ที่ R*1 จับที่สายทีละคู่ ถ้าหากวัดสาย Common เทียบกับ สาย Ground ได้ถูกต้อง ค่าความต้านทานที่อ่านได้จะน้อย แต่ถ้าวัดผิดสาย คือวัดสาย Ground เทียบกับ Ground ค่าความต้านทานที่อ่านได้จะสูงกว่า แต่ถ้าวัดสาย Common เทียบกับสาย Ground ที่ไม่ใช่คู่กันแล้ว เข็มมิเตอร์ก็จะไม่กระดิก ให้ทดลองวัดเปรียบเทียบกันทีละคู่ ก็จะทราบว่าสายใดเป็นสาย Common สายใดเป็นสาย Ground

2. ชนิดที่เป็น Common ภายใน SP Motor แบบนี้มีสายอยู่ 5 เส้นดังรูปที่ 2.12



รูปที่ 2.12 สเต็ปปิ้งมอเตอร์ ชนิดมีสาย 5 เส้น

จากรูปที่ 2.12 สายที่เป็น Common มี 1 สาย และสายที่เป็น Ground มี 4 สาย

ในการวัดให้ทำแบบเดียวกับการวัด SP Motor ชนิด Common ภายนอก แตกต่างกันเพียงแบบ Common ภายใน สาย Common 1 เส้น Drive สาย Ground 4 เส้น ดังนั้นหากสายเส้นใดเมื่อวัดเทียบกับสายเส้นอื่น แล้วมีค่าความต้านทานน้อยที่สุดสายเส้นนั้นเป็นสาย Common และที่เหลืออีก 4 เส้นจะเป็นสาย Ground

2.3.5 การเรียงเฟสของ Stepping Motor แบบ PM

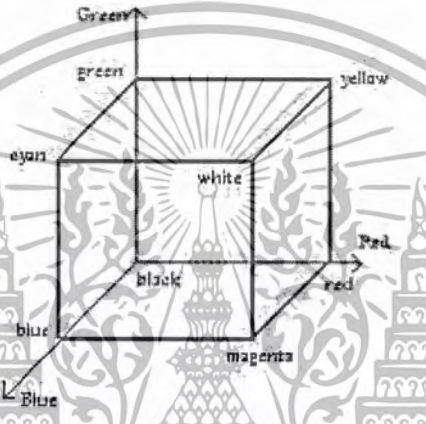
เมื่อเราทราบว่าสายเส้นใดเป็นสาย Common แล้ว แต่เรายังไม่ทราบว่าสาย Ground เส้นใดเป็นเฟสที่ 1 เฟสที่ 2 เฟสที่ 3 และเฟสที่ 4 ในการเรียงเฟสนั้นให้ใช้มิเตอร์วัดโดยนำ V+ เข้าที่สาย Common วัดเทียบกับสาย Ground เส้นใดก็ได้ 1 เส้น จะทำให้แกนโรเตอร์เคลื่อนไปข้างหน้า 1Step เมื่อเปลี่ยนสาย Ground เส้นแรกเป็นเส้นที่ 2 หาก Motor ไม่เคลื่อนที่ไปข้างหน้าแสดงว่าการเรียงเฟสไม่ถูกต้องก็ให้วัดเทียบกับสาย Ground เส้นใหม่ต่อไป หาก Motor เคลื่อนที่ไปข้างหน้าตามกัน ก็ให้วัดที่สาย Ground เส้นต่อไปเรื่อยๆ จะทำให้ทราบว่าสายเส้นใดเป็นเฟสแรก สายเส้นใดเป็นเฟสที่ 2 เฟสที่ 3 และเฟสที่ 4 การเรียงเฟสของ SP Motor แบบ PM ทั้งชนิดที่เป็น Common ภายนอกและชนิดที่เป็น Common ภายใน ใช้หลักการเดียวกัน

2.4 โมเดลสี

2.4.1 แบบ RGB

โมเดลสีชนิดนี้ประกอบด้วยการรวมกันของแม่สีหลักซึ่งได้แก่ แดง(R), เขียว (G) และน้ำเงิน (B) ซึ่งค่าสีต่างๆ ในแถบสเปคตรัมของสีจะได้มาจากการผสมกันในอัตราส่วนที่แตกต่างกันของแม่สีทั้งสาม

โมเดลสี RGB นี้จะแสดงด้วยแกนของลูกบาศก์สามแกนในระนาบ 3 มิติ ซึ่งค่าสีแดง เขียว และน้ำเงิน จะอยู่ที่มุมทั้งสามของแต่ละแกนดังแสดงด้วยรูป ซึ่งจะเห็นว่าค่าสีดำจะอยู่ที่จุดกำเนิด (origin) สีขาวจะอยู่ที่มุมตรงข้ามกับสีดำ ค่าของสีในช่วงระดับเทาจะอยู่ตามเส้นที่เชื่อมระหว่างค่าสีดำและค่าสีขาว จากรูปถ้าเป็นในระบบการแสดงสีแบบ 24 บิต (แบ่งออกเป็น 8 บิตต่อแม่สีหนึ่งสี) ค่าสีแดงจะถูกแทนด้วยค่า (255,0,0) เป็นต้น



รูปที่ 2.13 ลูกบาศก์สีของโมเดลแบบ RGB

โมเดลสี RGB ง่ายต่อการออกแบบและใช้งานในระบบคอมพิวเตอร์กราฟฟิก แต่ไม่เหมาะสำหรับการประยุกต์ใช้งานอื่นๆ ค่าของสีแดง สีเขียว และสีน้ำเงิน จะมีความสัมพันธ์กันอย่างมากซึ่งจะเป็นการยากที่จะนำไปประมวลผลเกี่ยวกับภาพ ดังนั้นหลายครั้งที่มีความจำเป็นที่จะต้องทำการแปลงภาพจากโมเดลสี RGB ให้อยู่ในรูปแบบของภาพแบบระดับเทา (Gray scale image) เพื่อความสะดวกดังกล่าว

ในการแปลงภาพจากโมเดลสี RGB ให้อยู่ในรูปแบบภาพระดับเทานั้น สามารถทำได้โดยการใช้สมการแปลงดังนี้

$$\text{Gray scale intensity} = 0.299R+0.587G+0.114B$$

ซึ่งเป็นสมการที่ใช้สำหรับการแปลงภาพจากมาตรฐาน NTSC

ในระบบดิจิตอลวีดีโอนั้น เรากล่าวถึงจำนวนบิตที่ใช้ในการสุ่มตัวอย่างสัญญาณแต่ละครั้งเป็นตัววัดความละเอียดของภาพที่ได้เป็นจำนวนบิตต่อพิกเซล (bit per pixel :bpp)

เนื่องจากในการสุ่มตัวอย่างสัญญาณแต่ละครั้งนั้นจะได้เป็นหนึ่งพิกเซลโดยระบบmonochrome ที่มีคุณภาพสูงนั้นจะใช้ 8 bpp ซึ่งหมายถึงจำนวนระดับเทา (gray level) ของภาพเท่ากับ 256 ระดับ แต่ในระบบการแสดงผลแบบสีนั้นเราต้องการหนึ่งช่องสัญญาณแบบ monochrome ต่อแม่สีแต่ละสี (แดง,เขียว

และน้ำเงิน) ซึ่งจะต้องใช้จำนวนบิตทั้งหมดเป็น 24 บิต ซึ่งจะได้ค่าระดับสีที่เป็นไปได้คือ 16,777,216 สี ในระบบดิจิตอลวีดีโอบางระบบซึ่งจะใช้ 24 บิตต่อพิกเซล

2.5 ฟอร์แมตของภาพ BMP

ย่อมาจากคำว่า Windows Device Independent bitmap เป็นฟอร์แมตพื้นฐานที่ใช้งานกันเฉพาะบน Windows เท่านั้น ใช้งานได้ดีกับโปรแกรมที่ทำงานภายใต้ Window 3.x, Windows95 หรือ Windows 98 ไฟล์ฟอร์แมตนี้ที่พบเห็นอยู่บ่อยๆ คือ ภาพวอลล์เปเปอร์ที่แสดงบนจอภาพของ Windows นั่นเอง มีขนาดทั้ง 4 และ 8 บิต ที่แสดงสีได้ตั้งแต่ 2, 16, 256 และ 16.7 ล้านสี

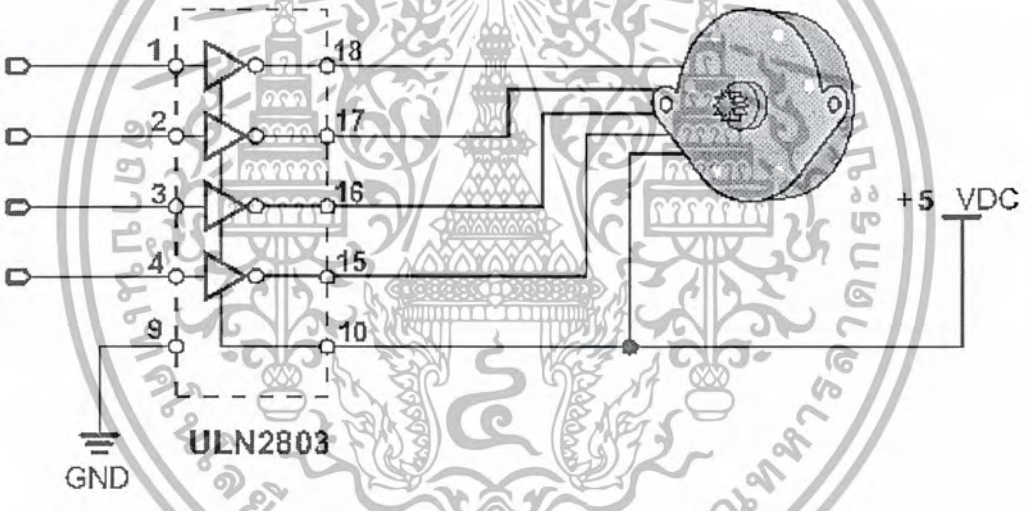
บทที่ 3
การออกแบบและการสร้าง

3.1 ส่วนของฮาร์ดแวร์

การออกแบบวงจรควบคุมการทำงานนั้น ประกอบด้วยวงจรของไมโครคอนโทรลเลอร์ MCS-51 ที่มีโปรแกรมควบคุมสั่งงานสเต็ปปิ้งมอเตอร์ให้หมุนซ้ายขวา วงจรไมโครคอนโทรลเลอร์ MCS-51 นี้จะเชื่อมต่อกับวงจรไดรเวอร์ ซึ่งทำหน้าที่ขยายกระแสในการควบคุมการทำงานของมอเตอร์ให้มีประสิทธิภาพสูงสุด การควบคุมมอเตอร์นั้นสิ่งสำคัญคือกระแสที่จ่ายให้กับขดลวดของมอเตอร์ต้องคงที่เสมอเพราะการหมุนแต่ละครั้งนั้นต้องมีความถูกต้องแม่นยำ

3.1.1 วงจรไดร์เวอร์ควบคุมการหมุนของสเต็ปปิ้งมอเตอร์

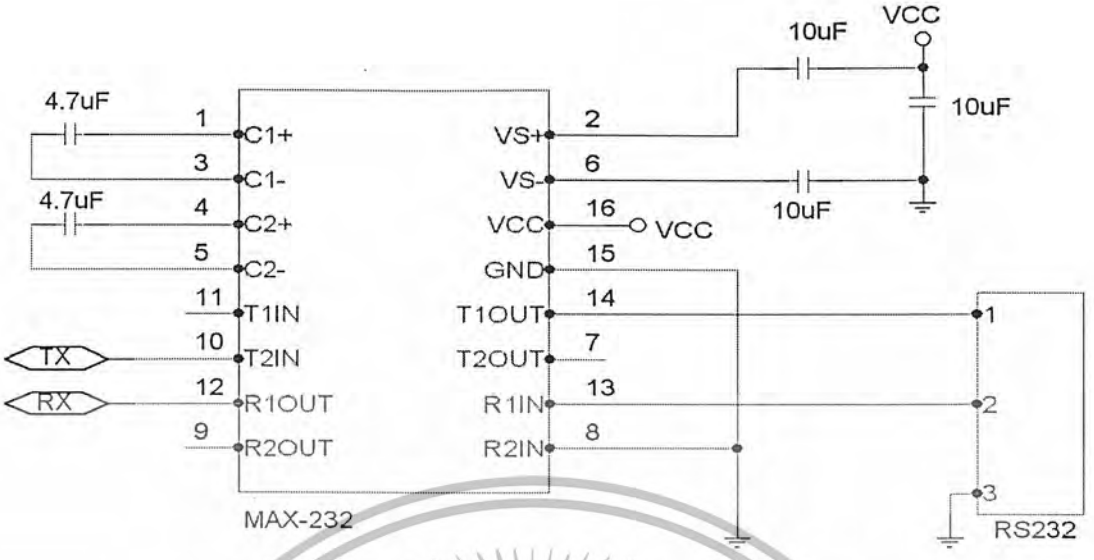
ในโครงงานนี้เราเลือกใช้สเต็ปปิ้งมอเตอร์แบบมีสาย 5 เส้นที่มีสเต็ปการหมุนหนึ่งครั้งเท่ากับ 7.5 องศาโดยกระตุ้นและควบคุมการหมุนของสเต็ปปิ้งมอเตอร์แบบ 2 เฟส ซึ่งในวงจรไดรเวอร์นี้เราเลือกใช้เป็นไอซีเบอร์ ULN2803 ซึ่งมีการต่อใช้งานดังรูปที่ 3.1



รูปที่ 3.1 วงจรขับสเต็ปปิ้งมอเตอร์โดยใช้ไอซีไดรเวอร์เบอร์ ULN2803

3.1.2 ส่วนการเชื่อมต่อและควบคุม

ส่วนของการติดต่อส่งผ่านข้อมูลจะใช้การสื่อสารแบบอนุกรม โดยจะใช้ไอซี MAX232 เพื่อเปลี่ยนระดับสัญญาณข้อมูลดิจิตอล (Digital data) ไปเป็นระดับสัญญาณดิจิตอล (Digital Signal) ตามมาตรฐาน RS-232 ซึ่งมีการเชื่อมต่อใช้งานดังแสดงในรูปที่ 3.2



รูปที่ 3.2 การต่อไอซี MAX232

ตัวไมโครคอนโทรลเลอร์ MCS-51 จะเป็นตัวควบคุมและสั่งงานให้สเต็ปปิ้งมอเตอร์ทำงานหมุนซ้ายขวา โดยจะรับข้อมูลการสั่งงานจากวงจรเชื่อมต่อพอร์ตอนุกรมของคอมพิวเตอร์ซึ่งมีไอซี MAX232 เป็นตัวทำหน้าที่ในการแปลงข้อมูลระดับ RS-232 ที่รับจากคอมพิวเตอร์ไปเป็นระดับทีทีแอล ก่อนที่จะถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 เพื่อทำการประมวลผลต่อไปได้ (เนื่องจากตัว MCS-51 ไม่เข้าใจสัญญาณ RS-232) หน้าที่อีกอย่างหนึ่งของ MAX232 คือแปลงข้อมูลที่จะส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 เพื่อใช้ตรวจสอบการทำงานของไมโครคอนโทรลเลอร์บนเครื่องคอมพิวเตอร์

3.2 ส่วนของซอฟต์แวร์

3.2.1 การออกแบบโปรแกรมควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ของไมโครคอนโทรลเลอร์

ในส่วนควบคุมการหมุนของมอเตอร์จะอาศัยการควบคุมจากไมโครคอนโทรลเลอร์ MCS-51 ให้ทำการสร้างสัญญาณพัลส์ส่งออกทางพอร์ต 1 ที่ตำแหน่ง 4 บิตบนและล่าง ซึ่งทำการต่อกับเฟสทั้ง 4 เฟสของมอเตอร์ทั้ง 2 ตัว ผ่านทางวงจรขับกระแสโดยมีการออกแบบขั้นตอนการทำงานของโปรแกรมเป็นลักษณะดังต่อ ไปนี้

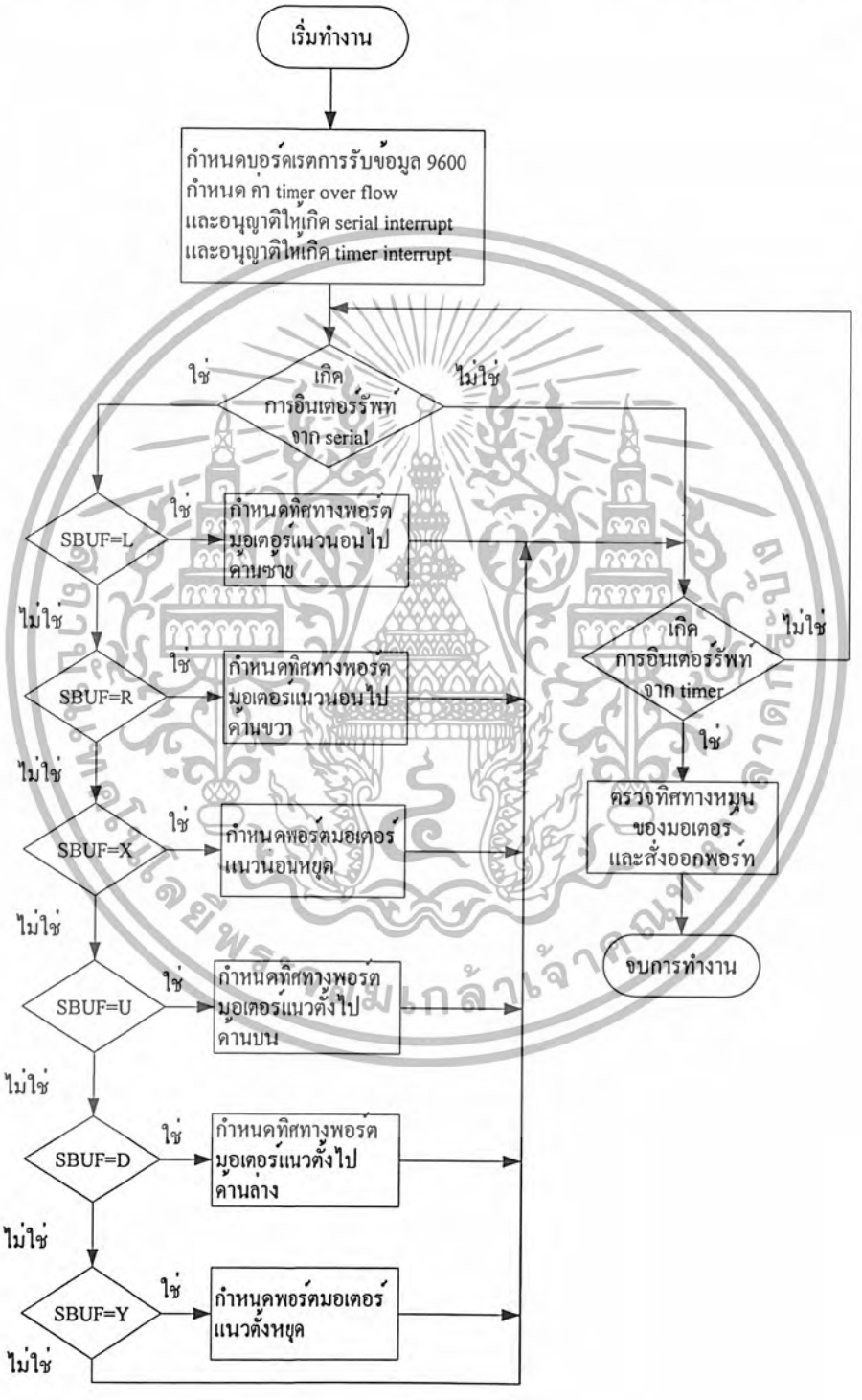
1.ทำการกำหนดค่าเริ่มต้นให้แก่รีจิสเตอร์ภายในไมโครคอนโทรลเลอร์ รวมถึงกำหนดสถานะของพอร์ตเพื่อเป็นการจัดเตรียมช่องทางสำหรับรับส่งข้อมูลจากคอมพิวเตอร์ผ่านทาง RS-232 รวมถึงกำหนดการทำงานภายในไมโครคอนโทรลเลอร์นั่นเอง เช่น โหมดการรับส่งข้อมูล, บอร์ดเรต, กำหนดค่าไทเมอร์โอเวอร์โฟลว, และกำหนดอินเตอร์รัพท์

2.ตรวจสอบการอินเตอร์รัพท์จากพอร์ตอนุกรม ถ้ามีการอินเตอร์รัพท์ก็ทำการตรวจสอบและกำหนดทิศทางการหมุนของมอเตอร์ว่าจะให้ไปทางซ้ายหรือขวา,บนหรือล่าง และสั่งหยุด

3.นำข้อมูลที่ได้รับจากข้อที่ 2 มาทำการสร้างพัลส์แล้วส่งออกทางพอร์ต 1 ของไมโครคอนโทรลเลอร์ โดยที่ 4 บิตบนจะควบคุมการหมุนซ้ายขวาของกล้อง ส่วน 4 บิตล่างจะควบคุมการหมุนบนล่างของกล้อง

4.กลับไปรอรับข้อมูลแล้วดำเนินตามขั้นตอนที่ 2 เป็นเช่นนี้เรื่อยไป

สำหรับโฟล์วชาร์ท (Flow Chart) ในส่วนของการควบคุมมอเตอร์ สามารถแสดงได้ดังรูปที่ 3.3



รูปที่ 3.3 โฟล์วชาร์ทการทำงานของกระบวนการควบคุมการหมุนของสเต็ปปิ้งมอเตอร์

3.2.2 การออกแบบหน้าต่างของโปรมแกรม

ส่วนนี้จะใช้ในการควบคุมและรับข้อมูลจากส่วนของฮาร์ดแวร์มาแสดงผลที่คอมพิวเตอร์ ซึ่งมีขั้นตอนการทำงานดังนี้

1.เริ่มทำการบันทึกภาพในเฟรมแรกที่ใช้ในการอ้างอิง โดยดึงค่าสี RGB ของภาพในเฟรมแรกและนำมาแปลงเป็นภาพสี Grayscale โดยใช้สมการ

$$\text{Gray scale intensity} = 0.299R+0.587G+0.114B$$

สมการนี้จะนำองค์ประกอบของสี R, G, B มาคูณกับค่าคงที่ ผลลัพธ์ที่ได้จะเป็นค่าความสว่างของแต่ละพิกเซลที่ทำการคำนวณและเก็บไว้ในค่าคงที่ค่าหนึ่ง

บันทึกภาพเฟรมที่สองแล้วทำการคำนวณค่าความสว่างของแต่ละพิกเซล โดยใช้วิธีเดียวกับภาพแรก แล้วเก็บค่าความสว่างไว้ในค่าคงที่อีกค่าหนึ่ง

2.ทำการเปรียบเทียบภาพในเฟรมแรกและเฟรมที่สอง โดยนำค่าความสว่างที่ตำแหน่งพิกเซลเดียวกันของแต่ละเฟรมมาเปรียบเทียบกัน ถ้าแตกต่างกันเกินค่า Luminance Threshold จะถือว่าพิกเซลนั้นมีการเปลี่ยนแปลง และจะทำการนับจำนวนพิกเซลที่มีความเปลี่ยนแปลงนั้นไว้ โดยค่า Luminance Threshold นั้นจะเป็นค่าคงที่ ที่เรากำหนดไว้

3.นำจำนวนจุดที่มีค่าความสว่างแตกต่างกันมาเปรียบเทียบกับค่า Threshold Value ถ้าจำนวนจุดที่มีค่าความสว่างแตกต่างกันมีมากกว่าค่า Threshold Value ซึ่งเป็นค่าคงที่ ที่เรากำหนดไว้ ก็จะเริ่มทำการหาค่ากึ่งกลางของกลุ่มพิกเซลที่มีค่าความสว่างแตกต่างกัน

4.การหากึ่งกลางของกลุ่มพิกเซลที่มีค่าความสว่างแตกต่างกัน จะแยกหาเป็นแนวแกนนอนกับแนวแกนตั้ง ทำได้โดยการนำอาตำแหน่งของแต่ละพิกเซลมาบวกกันแล้วหารด้วยจำนวนพิกเซลทั้งหมดที่มีการเปลี่ยนแปลงจะทำให้เราทราบค่าตำแหน่งกึ่งกลาง โดยแยกเป็นตำแหน่งในแนวแกนตั้ง (OBJY) และตำแหน่งในแนวแกนนอน (OBJX)

5.โดยนำค่ากึ่งกลางของวัตถุในแนวแกนนอน (OBJX) มาลบด้วยครึ่งหนึ่งของความยาวของภาพ (CurX/2) จากการคำนวณนี้ จะเป็นได้ทั้งค่าบวก, ลบ และ ศูนย์ ทำให้สะดวกในการพิจารณาว่าตำแหน่งกึ่งกลางของวัตถุในแนวนอนนี้อยู่ทางด้านซ้ายหรือด้านขวาจากจุดกึ่งกลางภาพ และทำเช่นเดียวกันกับค่ากึ่งกลางของวัตถุในแนวแกนตั้ง

6.โปรแกรมนี้จะมีการกำหนดกรอบสี่เหลี่ยมจัตุรัสโดยมีกึ่งกลางอยู่ที่จุดกึ่งกลางภาพดังรูปที่ 3.4

รูปที่ 3.4 ภาพแสดงตำแหน่งของกรอบสี่เหลี่ยมที่เรากำหนดขึ้นไว้เปรียบเทียบ

โดยโปรแกรมจะนำค่า OBJX-(CurX/2) มาเปรียบเทียบกับตำแหน่งของกรอบ ถ้าหากค่าดังกล่าวเลยตำแหน่งของกรอบออกไป ก็จะมีการสั่งให้มอเตอร์มีการเคลื่อนที่ ซึ่งการกำหนดทิศทางจะขึ้นกับเครื่องหมายของค่า OBJX-(CurX/2) แต่ถ้าค่า OBJX-(CurX/2) นั้นยังอยู่ภายในกรอบสี่เหลี่ยมจะไม่มีการสั่งให้มอเตอร์เคลื่อนที่ และทำเช่นเดียวกันในแนวแกนตั้ง

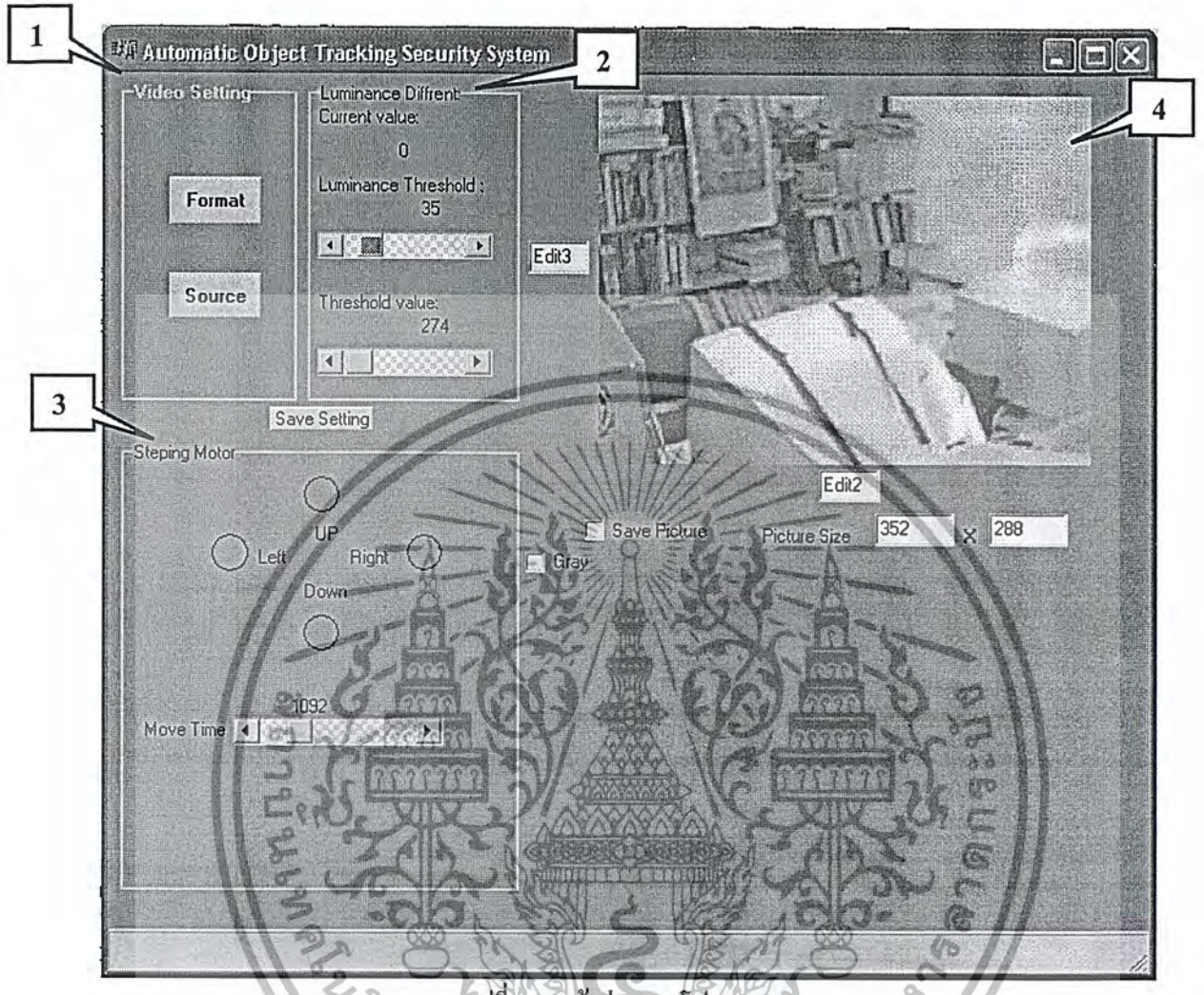
7.นำค่า OBJX-(CurX/2) มาพิจารณาถ้ามีค่าเป็นลบจะควบคุมมอเตอร์ให้หมุนซ้ายโดยส่ง "L" และถ้าค่า OBJX-(CurX/2) มีค่าเป็นบวกจะควบคุมมอเตอร์ให้หมุนขวาโดยส่ง "R" ออกไปยังไมโครคอนโทรลเลอร์ นำค่า OBJY-(CurY/2) มาพิจารณาถ้ามีค่าเป็นลบจะควบคุมให้มอเตอร์หมุนลงล่างโดยส่ง "D" และถ้าค่า OBJY-(CurY/2) มีค่าเป็นบวกจะควบคุมให้มอเตอร์หมุนขึ้นบนโดยส่ง "U" ออกไปยังไมโครคอนโทรลเลอร์

8.ทำการหน่วงเวลารอมอเตอร์หมุนเท่ากับค่า Move Time และสั่งให้มอเตอร์หยุดหมุนเมื่อถึงตำแหน่งที่ต้องการ จากนั้นกลับไปทำตามขั้นตอนที่ 1 เป็นเช่นนี้เรื่อยไป

โดยโฟล์วชาร์ทการทำงานในส่วนนี้แสดงได้ดังรูปที่ 3.4

รูปที่ 3.4 โฟล์วชาร์ทการทำงานของโปรแกรม

จากโฟล์วชาร์ทการทำงานของโปรแกรม ก็สามารถออกแบบหน้าต่างของโปรมได้ดังรูปที่ 3.5 ซึ่งมีรายละเอียดในการใช้งานของส่วนต่างๆ ดังนี้



รูปที่ 3.5 หน้าต่างของโปรแกรม

หมายเลข 1 คือ ส่วนที่ใช้ในการกำหนดค่าพารามิเตอร์ต่างๆ ของกล้อง

หมายเลข 2 คือ ส่วนของค่าพารามิเตอร์ต่างๆ ที่ใช้ในการเปรียบเทียบความแตกต่างของภาพ

หมายเลข 3 คือ ส่วนของระบบควบคุมการเคลื่อนที่ของกล้อง

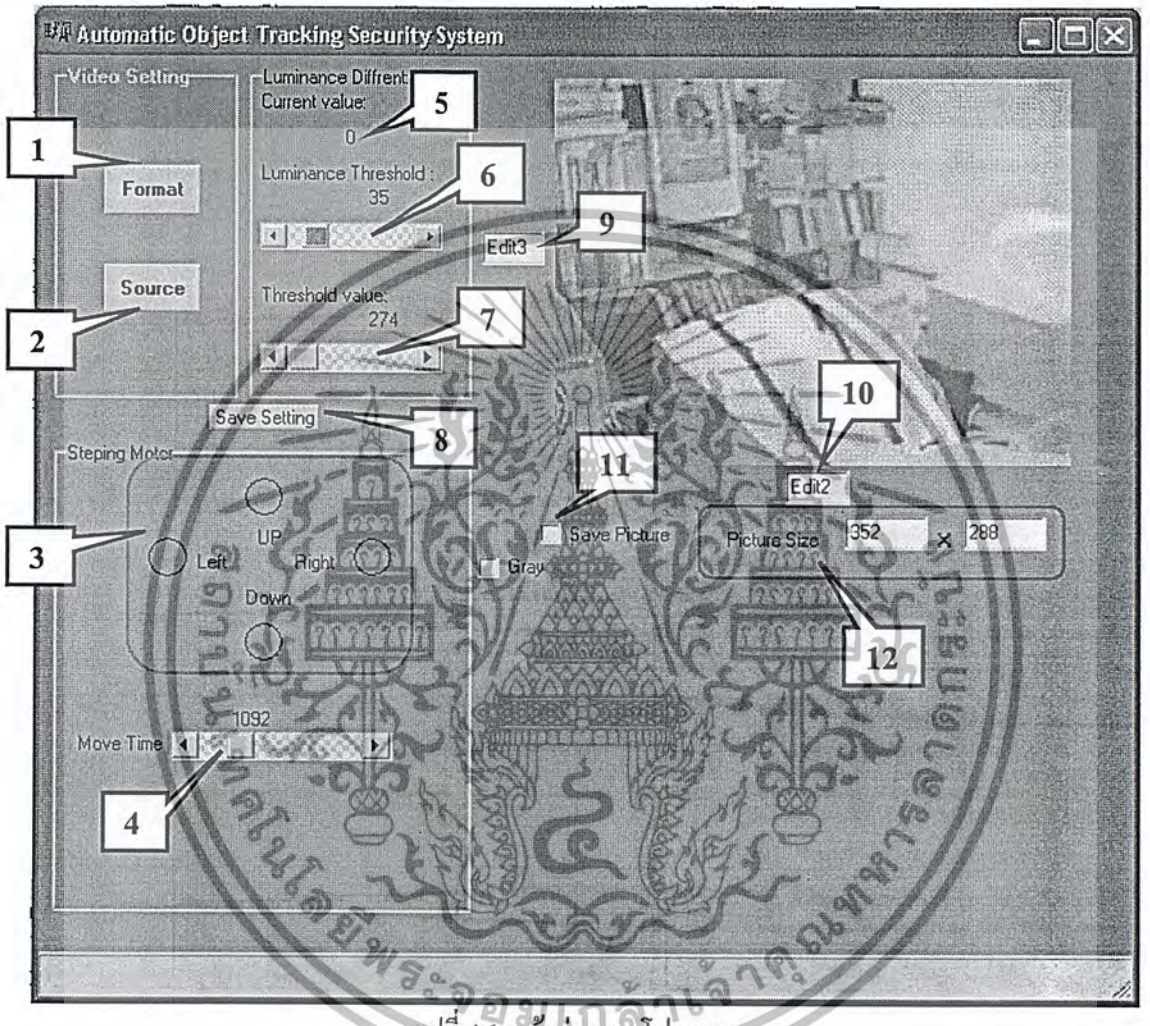
หมายเลข 4 คือ ส่วนของการแสดงภาพ

บทที่ 4

การทดลองและผลการทดลอง

4.1 แสดงหน้าต่างโปรแกรมและรายละเอียดการใช้งาน

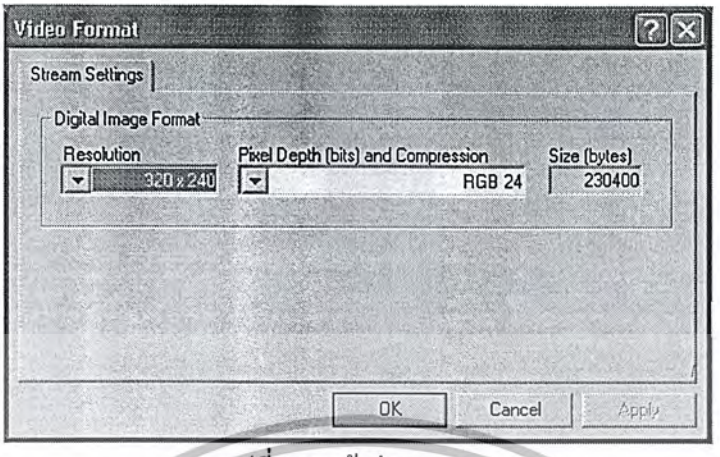
หน้าต่างโปรแกรมแสดงดังรูปที่ 4.1 มีรายละเอียดการใช้งานของส่วนต่างๆ ดังต่อไปนี้



รูปที่ 4.1 หน้าต่างของโปรแกรม

หมายเลข 1 ปุ่ม Format

- เมื่อทำการคลิกจะปรากฏหน้าต่างดังนี้

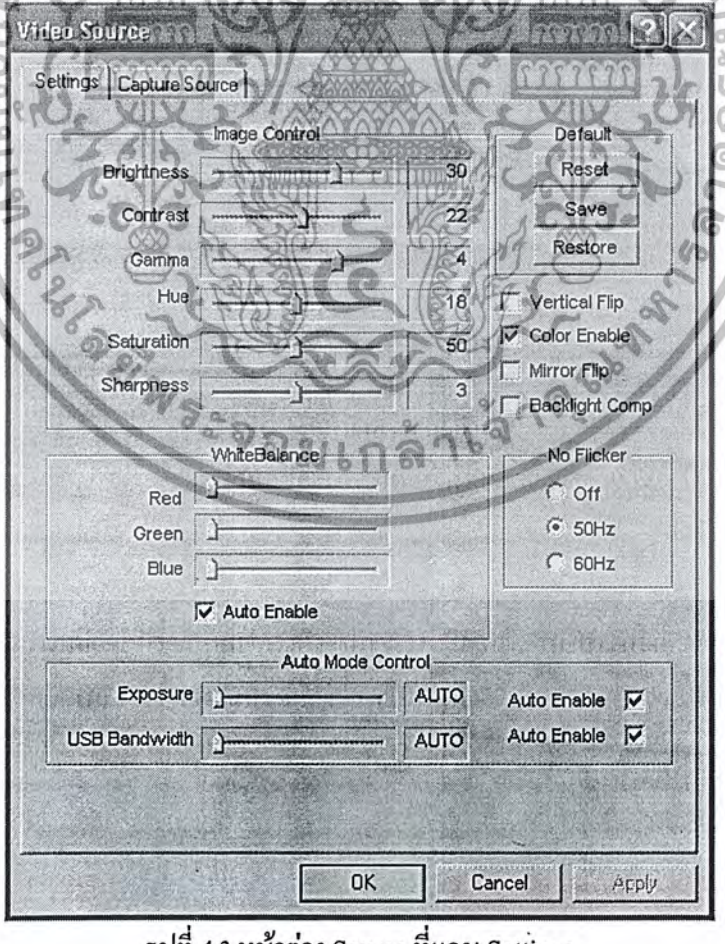


รูปที่ 4.2 หน้าต่างFormat

- สามารถทำการกำหนดฟอร์แมตของภาพ ได้แก่ ค่า Resolution, Pixel Depth(bits) and Compression, Size(bytes)

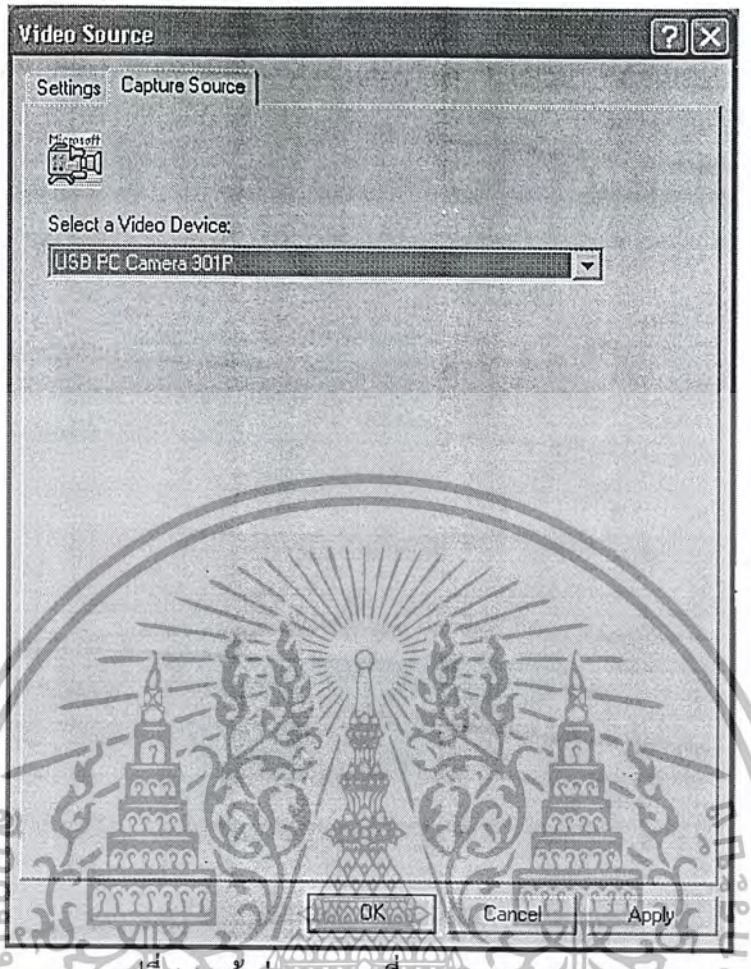
หมายเลข 2 ปุ่ม Source

- เมื่อทำการคลิกจะปรากฏหน้าต่างดังรูปที่ 4.3 โดยสามารถทำการปรับคุณสมบัติต่างๆของภาพ เช่น ความสว่าง,ความคมชัดของภาพ เป็นต้น



รูปที่ 4.3 หน้าต่าง Source ที่แถบ Settings

- ใช้เลือกกล้องตัวที่จะใช้งาน ดังรูปที่ 4.4



รูปที่ 4.4 หน้าต่าง Source ที่แถบ Capture Source

หมายเลข 3 เป็นบริเวณที่แสดงทิศทางการเคลื่อนที่ของกล้องโดยจะเปลี่ยนเป็นสีขาวในทิศทางที่กล้องจะเคลื่อนที่ไป

หมายเลข 4 ใช้กำหนดคาบเวลาในการหมุนของกล้อง เมื่อตรวจพบความเคลื่อนไหว

หมายเลข 5 ค่าจำนวนพิกเซลที่แตกต่างกันของภาพ เฟรมที่อยู่ติดกัน

หมายเลข 6 กำหนดค่าคงที่ความต่างของสี ที่ใช้ในการตัดสินใจของโปรแกรมว่าภาพในเฟรมที่อยู่ติดกันนั้นแตกต่างกันหรือไม่ ซึ่งถ้าเกินค่าดังกล่าวแล้วพิกเซลนั้นจะถือว่ามีความแตกต่างกัน

หมายเลข 7 ค่าคงที่ของจำนวนพิกเซลที่ใช้ในการตัดสินใจของโปรแกรมว่าจะให้มีการส่งข้อมูลที่ประมวลได้ไปควบคุมมอเตอร์

หมายเลข 8 ปุ่ม Save Setting ใช้ในการบันทึกค่าคงที่ความต่างของสี และค่าคงที่ของจำนวนพิกเซลที่จะเริ่มให้มอเตอร์ทำงาน เพื่อให้สะดวกในการใช้งานครั้งต่อไป

หมายเลข 9 เป็นช่องที่แสดงตำแหน่งกึ่งกลางของวัตถุที่มีการเคลื่อนไหวในแนวตั้ง

หมายเลข 10 เป็นช่องที่แสดงตำแหน่งกึ่งกลางของวัตถุที่มีการเคลื่อนไหวในแนวนอน

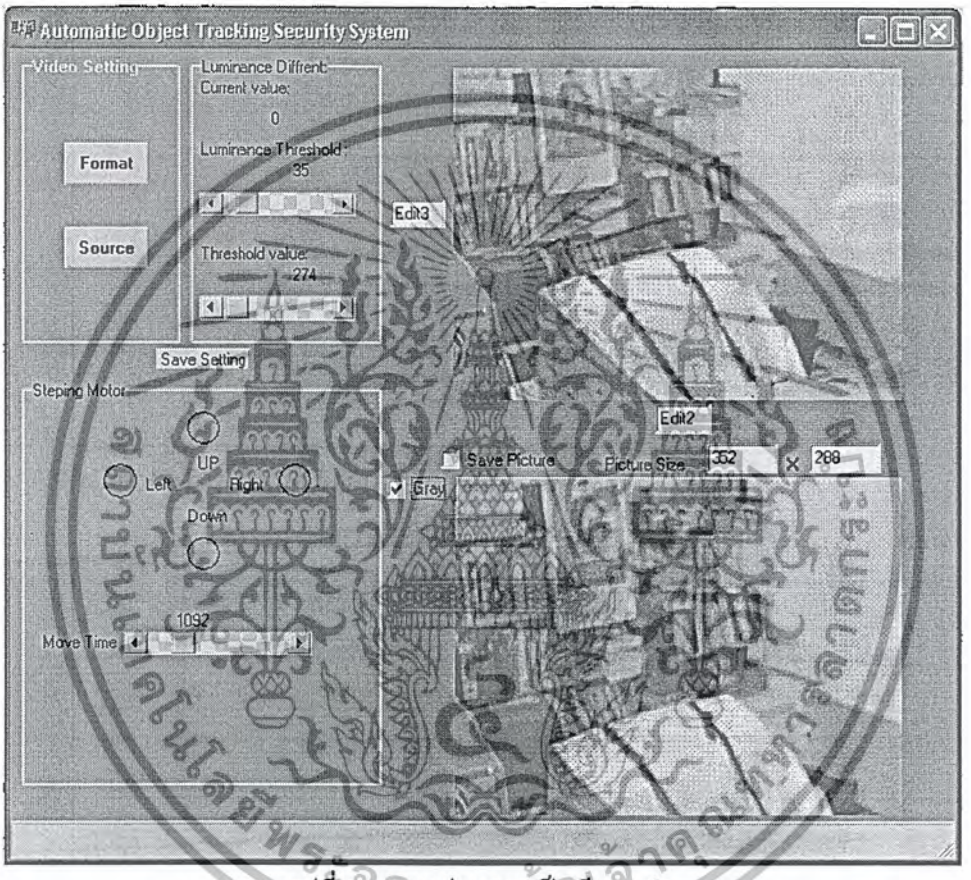
หมายเลข 11 เป็นช่องที่สั่งให้โปรแกรมทำการบันทึกภาพเมื่อมีการเคลื่อนไหวของวัตถุ

หมายเลข 12 แสดงขนาดของภาพ

4.2 ผลการทดลอง

4.2.1 การแปลงภาพสีเป็นภาพสีขาว-ดำ

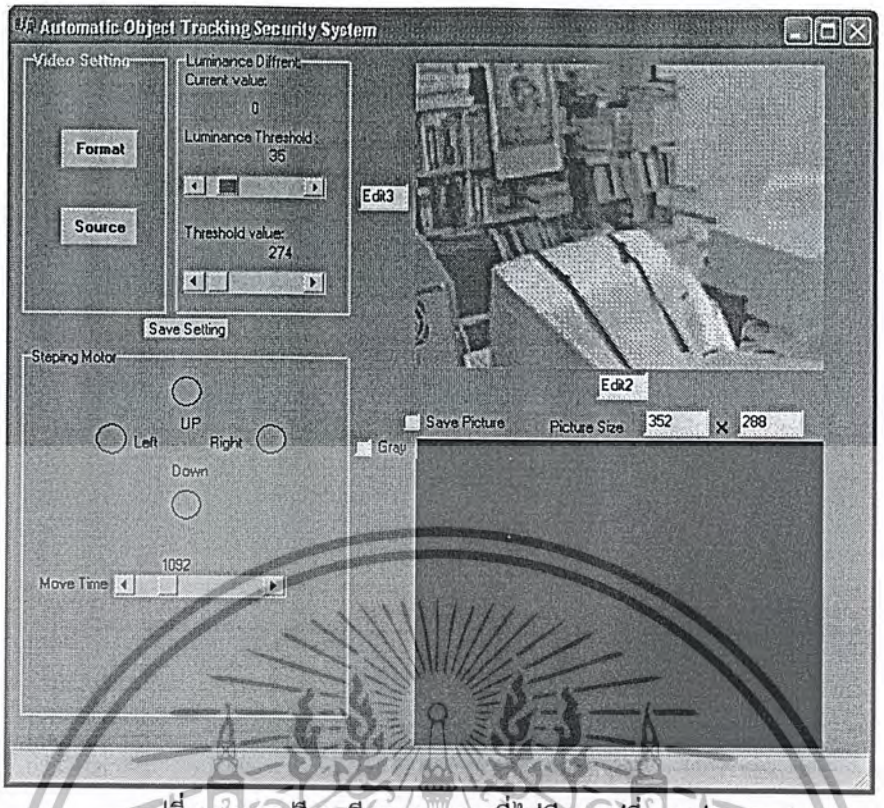
ก่อนที่จะนำภาพไปประมวลผล เราจะต้องทำการแปลงเป็นภาพขาว-ดำ เพื่อสะดวกในการเปรียบเทียบความแตกต่าง และขจัดปัญหาเรื่องความเปลี่ยนแปลงของแสง ซึ่งผลของการแปลงแสดงได้ดังรูปที่ 4.5



รูปที่ 4.5 การแปลงภาพเป็นสีขาว-ดำ

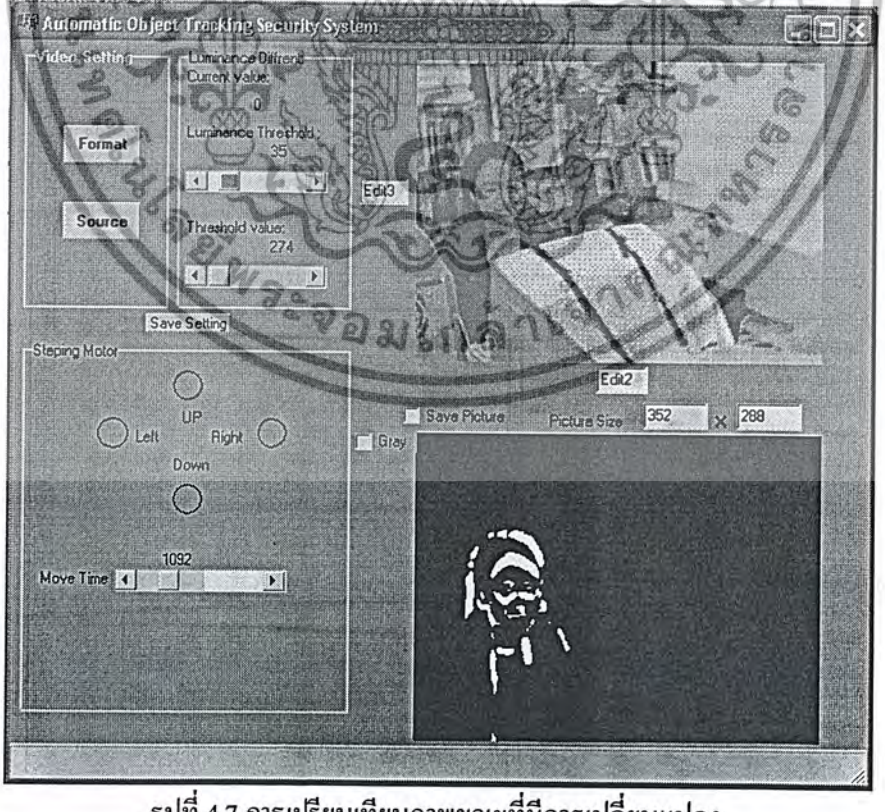
4.2.2 การเปรียบเทียบภาพ

ถ้าหากภาพไม่มีการเปลี่ยนแปลง การเปรียบเทียบภาพระหว่างสองเฟรมที่อยู่ติดกันจะไม่มีความแตกต่าง และผลที่ได้จะแสดงเป็นสีดำ ดังรูปที่ 4.6



รูปที่ 4.6 การเปรียบเทียบภาพขณะที่ไม่มีการเปลี่ยนแปลง

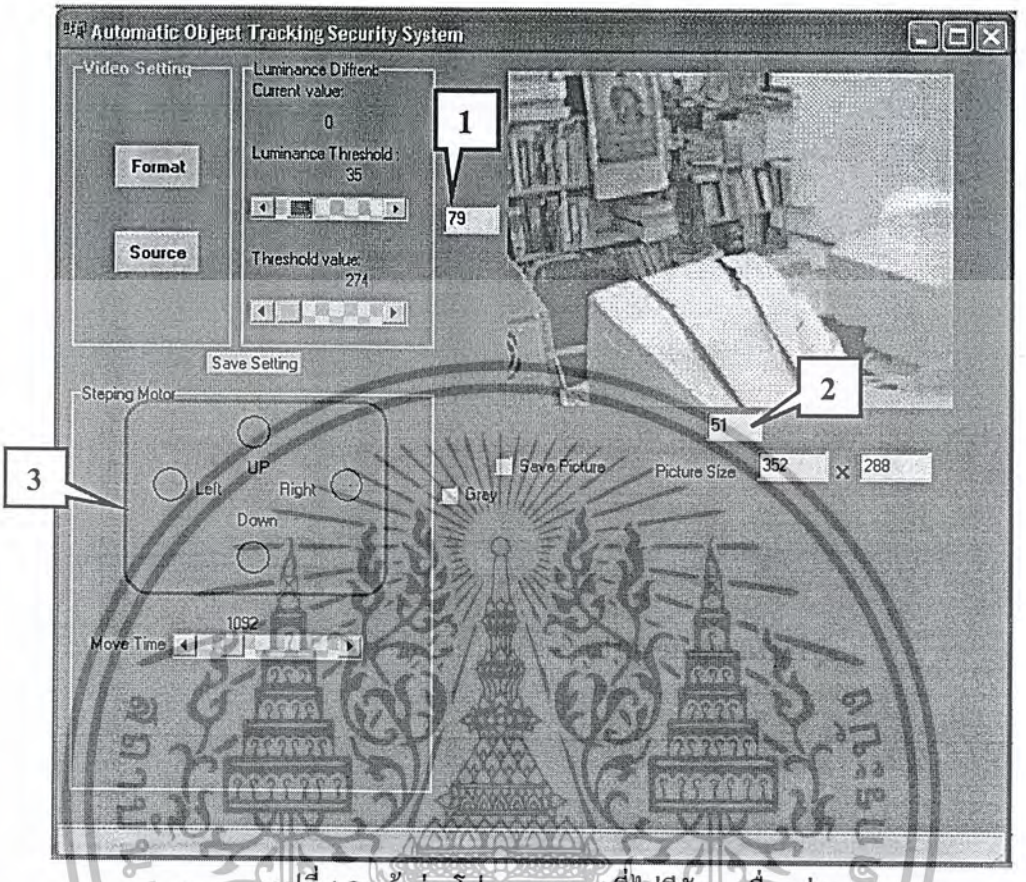
ถ้าหากภาพมีการเปลี่ยนแปลง เช่น มีวัตถุเคลื่อนที่ผ่านกล้อง จะทำให้การเปรียบเทียบภาพระหว่างสองเฟรมที่อยู่ติดกันเกิดความแตกต่างกัน โดยบริเวณที่ต่างกันจะแสดงเป็นสีขาว ดังรูปที่ 4.7



รูปที่ 4.7 การเปรียบเทียบภาพขณะที่มีการเปลี่ยนแปลง

4.2.3 การทำงานของโปรแกรม

ขณะที่ไม่มีวัตถุเคลื่อนผ่านหน้ากล้อง หน้าต่างของโปรแกรมจะแสดงดังรูปที่ 4.8



รูปที่ 4.8 หน้าต่างโปรแกรมขณะที่ไม่มีวัตถุเคลื่อนผ่าน

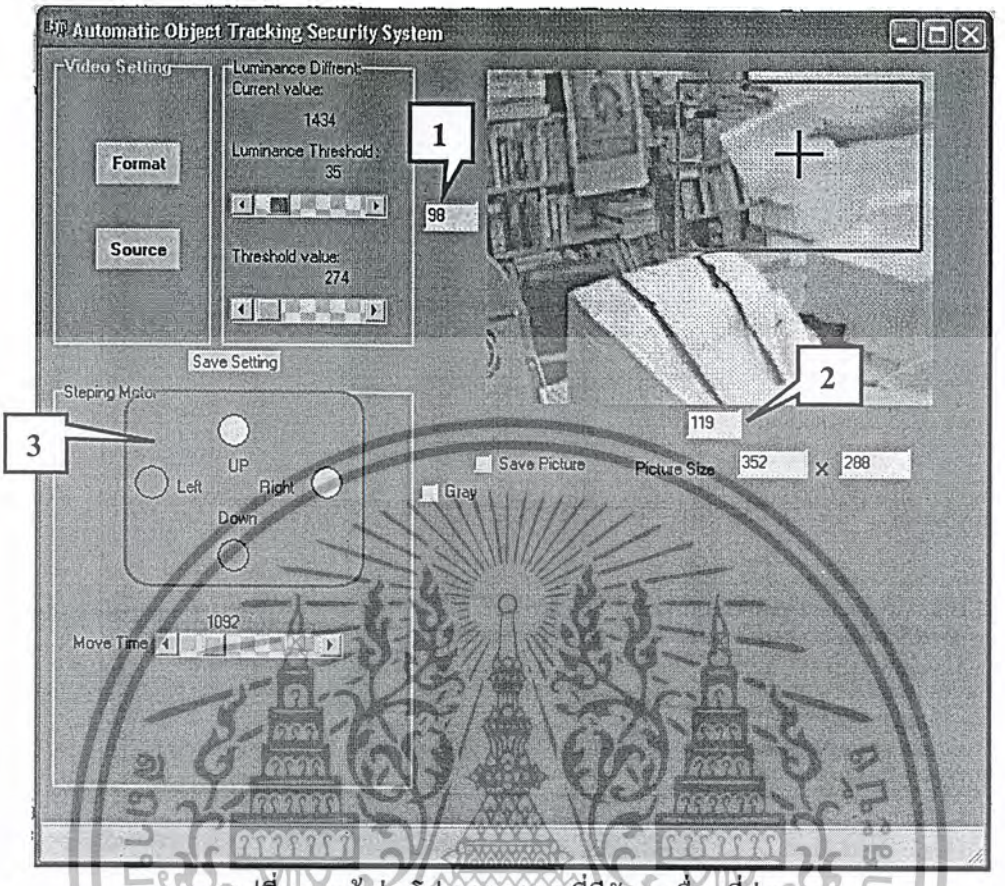
จากรูปที่ 4.8

หมายเลข 1 จะแสดงตำแหน่งกึ่งกลางในแนวนอนของวัตถุสุดท้ายที่เคลื่อนที่ผ่านกล้อง

หมายเลข 2 จะแสดงตำแหน่งกึ่งกลางในแนวตั้งของวัตถุสุดท้ายที่เคลื่อนที่ผ่านกล้อง

หมายเลข 3 เมื่อไม่มีวัตถุเคลื่อนผ่าน จะไม่มีการสั่งงานให้มอเตอร์หมุนกล้อง โดยสังเกตได้จาก จะไม่มีไฟสีขาวเกิดขึ้น

เมื่อมีวัตถุเคลื่อนที่ผ่านกล้องจะแสดงดังรูปที่ 4.9



รูปที่ 4.9 หน้าต่างโปรแกรมขณะที่มีวัตถุเคลื่อนที่ผ่าน

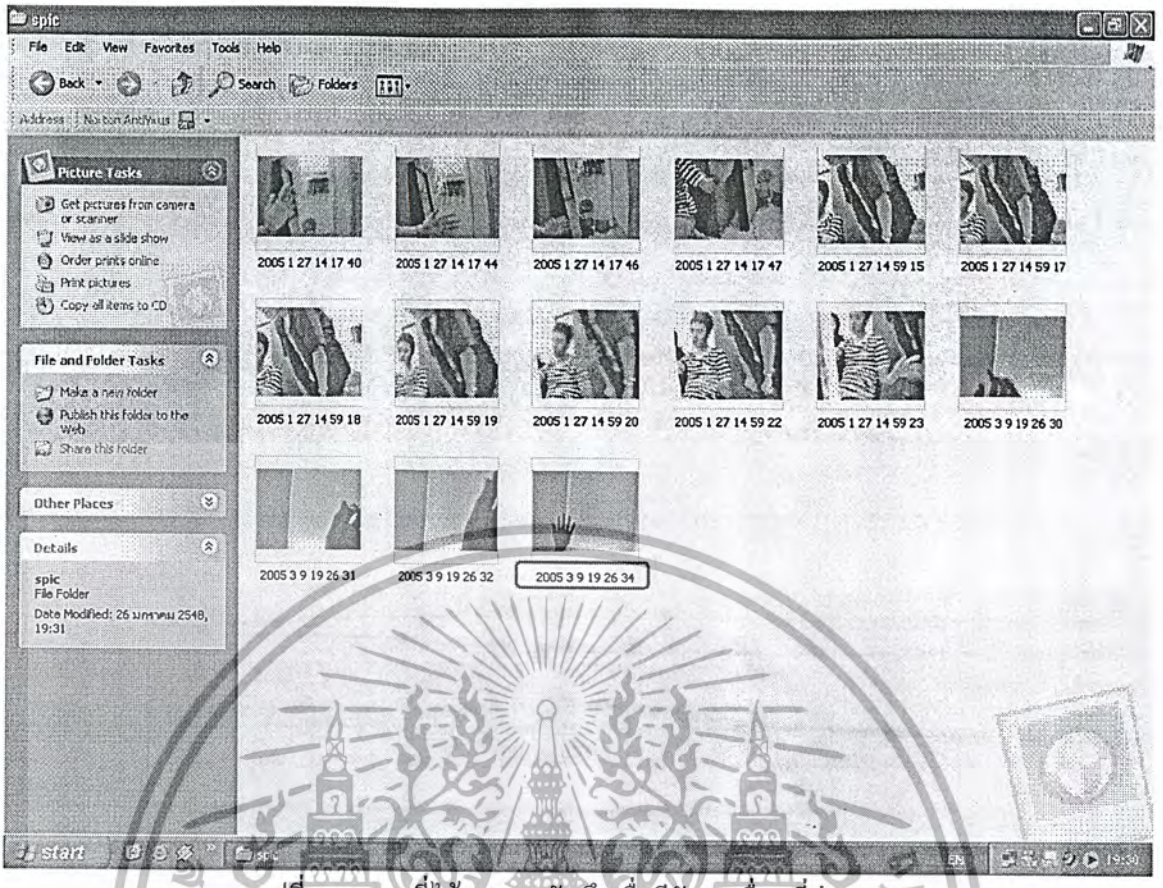
หมายเลข 1 จะแสดงตำแหน่งกึ่งกลางในแนวนอนของวัตถุที่เคลื่อนที่ผ่านกล้อง

หมายเลข 2 จะแสดงตำแหน่งกึ่งกลางในแนวตั้งของวัตถุที่เคลื่อนที่ผ่านกล้อง

หมายเลข 3 เมื่อมีวัตถุเคลื่อนผ่าน จะมีการสั่งงานให้มอเตอร์หมุนกล้องไปยังตำแหน่งกึ่งกลางของวัตถุที่เคลื่อนที่ผ่าน โดยสังเกตได้จากจะมีไฟสีขาวเกิดขึ้น ในทิศทางนั้น

4.2.4 การบันทึกภาพ

หากต้องการบันทึกภาพวัตถุที่เคลื่อนที่ผ่าน เก็บไว้ภายในเครื่องคอมพิวเตอร์ ทำได้โดยการคลิกเครื่องหมายถูก บริเวณปุ่ม Save Picture จะทำให้ภาพของวัตถุที่เคลื่อนไหวถูกจัดเก็บไว้ในโฟลเดอร์ที่เรากำหนดไว้ โดยในที่นี้เรากำหนดให้เก็บไว้ในโฟลเดอร์ C:\spic โดยผลการบันทึกภาพแสดงดังรูปที่ 4.10



รูปที่ 4.10 ภาพที่ได้จากการบันทึกเมื่อมีวัตถุเคลื่อนที่ผ่าน

จากรูปที่ 4.10 เป็นภาพที่ได้จากการบันทึกเมื่อมีวัตถุเคลื่อนที่ผ่าน โดยชื่อภาพจะแสดงเป็น ปี เดือน วัน และเวลา ทำให้เราสามารถตรวจสอบได้ง่าย

บทที่ 5
บทสรุปและวิจารณ์

5.1 บทสรุป

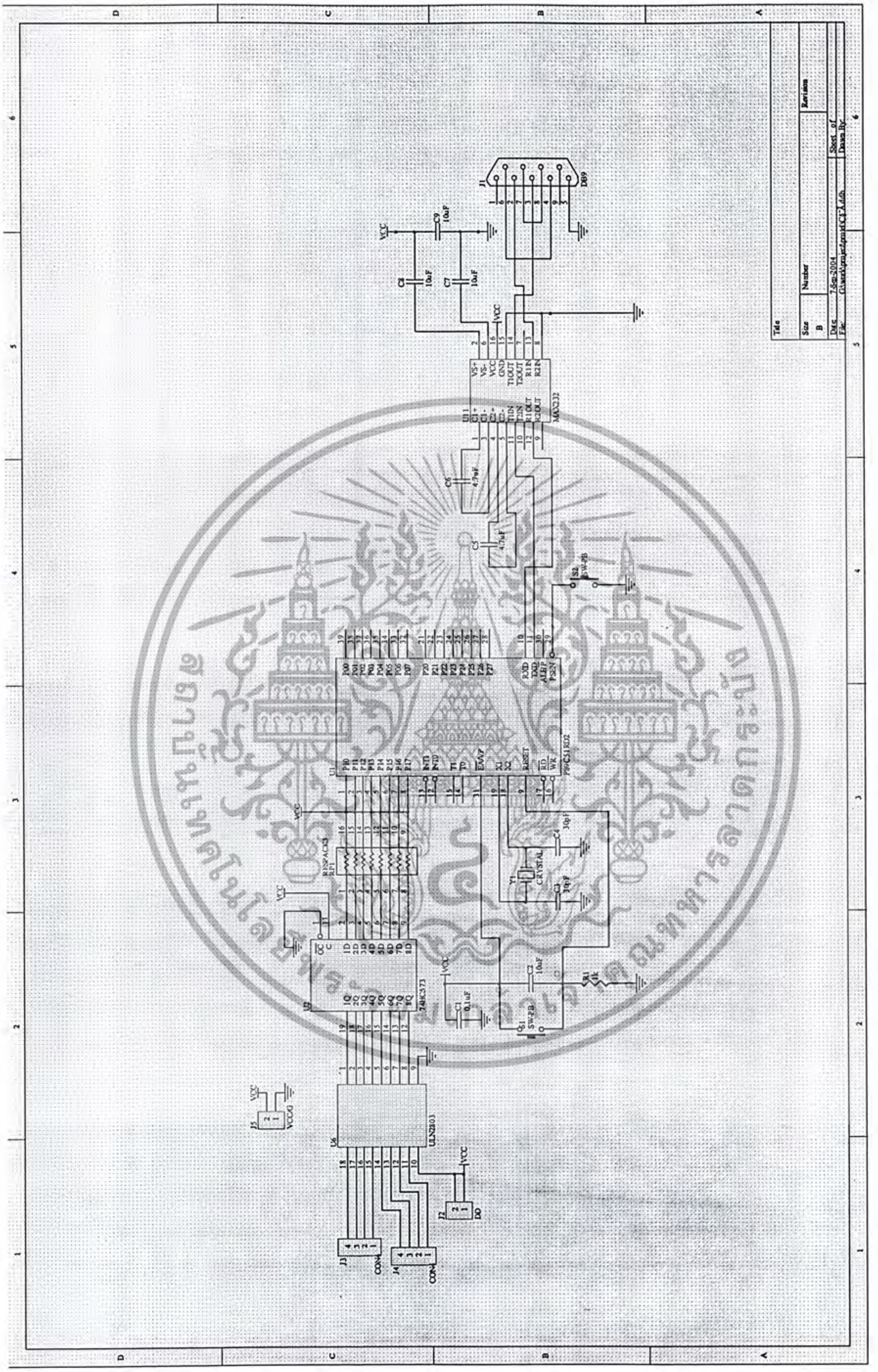
ระบบรักษาความปลอดภัยแบบติดตามวัตถุอัตโนมัติ มีความสามารถในการใช้งานได้ดังนี้

- สามารถแสดงภาพที่ได้รับจากกล้องผ่านทางหน้าจอคอมพิวเตอร์
- สามารถติดตามวัตถุที่มีการเคลื่อนที่ได้โดยอัตโนมัติ ซึ่งอาศัยการเปรียบเทียบความแตกต่างของภาพเฟรมอ้างอิงกับเฟรมที่นำมาเปรียบเทียบ
- สามารถแสดงทิศทางในการเคลื่อนที่ของกล้องผ่านทางหน้าจอคอมพิวเตอร์ได้
- สามารถทำการบันทึกภาพขณะที่มีการเคลื่อนไหวของวัตถุผ่านหน้ากล้อง โดยภาพที่บันทึกนั้นจะมีรายละเอียดของวันและเวลา

5.2 แนวทางพัฒนาต่อ

นำระบบนี้ไปพัฒนาเพิ่มเติมให้เป็นระบบไร้สาย เพื่อเพิ่มขีดความสามารถในการนำระบบไปใช้งานในที่ต่างๆ และอาจจะใช้เทคนิควิธีการอื่นในการหาทิศทางการเคลื่อนไหวของวัตถุที่มีประสิทธิภาพมากกว่านี้

ภาคผนวก



```
#include <vcl.h>

#pragma hdrstop
#include <stdio.h>
#include "capture.h"
#include "Unit1.h"
#include <wingdi.h>

//---------------------------------------------------------------------------

#pragma package(smart_init)
#pragma link "MSCommLib_OCX"
#pragma resource "*.dfm"

#define MaxWidth 800
#define MaxHight 600
#define MaxPixel (MaxWidth *MaxHight)

TForm1 *Form1;
FILE *Capfile;
FILE *Runfile;

char header[60];

int DeviceID=0;
unsigned int LThreshold=0;
unsigned int MThreshold=0;
int frame2=0;
int Speed=0;
int Step=0;
OleVariant vIn;
AnsiString strIn,strOut;
 int OBJX;
 int OBJY;

DWORD ThreadId;
HANDLE hThread;
extern int curW,curH;
 Word Year, Month, Day, Hour, Min, Sec, MSec;
struct RGBS
{
unsigned char BLUE;
unsigned char GREEN;
unsigned char RED;
} ;
struct RGBS RGB[MaxPixel];
unsigned char OLDLU [MaxPixel];
struct BMPFiles
{
BITMAPFILEHEADER FileHeader;
BITMAPINFOHEADER FileInfoHeader;
} BMPFile;

 void SaveSetting( void )
{
 FILE *inifile;
inifile=fopen("setting.ini","w");
fprintf(inifile,"%d %d %d %d %d",
DeviceID,LThreshold,MThreshold,Speed,Step);
fclose( inifile);
```

```
}
int GetSetting(void )
{
int ret;
FILE *inifile;
inifile=fopen("setting.ini","r");
ret= fscanf(inifile,"%d%d%d%d%d",
&DeviceID,&LThreshold,&MThreshold,&Speed,&Step);
fclose( inifile);
return ret;
}
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------

void __fastcall TForm1::FormCreate(TObject *Sender)
{
MSComm1->CommPort=1;
MSComm1->PortOpen=true;
Timer1->Enabled=false;
Timer2->Enabled=true;
Capfile=fopen("temp.bmp" , "w");    //  open file
fclose(Capfile);
Capfile=fopen("temp.bmp" , "r");    //  open file
 if (GetSetting()==0)
 {
        DeviceID=0;
        MThreshold=0;
        LThreshold=0;
        Speed=0;
        Step=0;
 }
 ScrollBar1->Position=LThreshold;
 ScrollBar2->Position=MThreshold;
 ScrollBarSpeed->Position=Speed;
 //CreateDeviceList();
 Video_Startup(Panel1->Handle );

}
//---------------------------------------------------------------------

void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{  HWND area;
  area=this->Handle;

  ConnectCapDriver(  hCapWnd,DeviceID);
}
//---------------------------------------------------------------------

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction
&Action)
{
fclose(Capfile);
}
void SendSerial( OleVariant chr)
{
 Form1->MSComm1->Output=chr;
}
```

```
void Running(void)
{
unsigned int SumX=0;
unsigned int SumY=0;
unsigned int centerX;
unsigned int centerY;

unsigned int PixCounter;
 int x,y;
unsigned char Lu;
int pos;
Captofile("temp.bmp");
fseek(Capfile, 18, 0);
fread(&curW,4,1,Capfile);
fread(&curH,4,1,Capfile);

fseek(Capfile, 54, 0);
fread(&RGB,1,curW*curH*3,Capfile);
Form1->Edit1->Text=curW;
Form1->Edit4->Text=curH;
PixCounter=0;

 if (frame2==0)goto kk;

for (x=0;x<curW;x++)
{
        for (y=0;y<curH;y++)
        {
                pos=(y*curW)+ (x);
                Lu=(RGB[pos].RED*0.3)+(RGB[pos].GREEN *0.6)+
(RGB[pos].BLUE*0.1);

                if (abs(Lu-OLDLU[pos]) >LThreshold)
                {
                        PixCounter++;
                        SumX+= x;
                        SumY+= y;
                }
                OLDLU[pos]=Lu ;
        }
}
if    ((frame2==2) &&( PixCounter>(MThreshold+1)) )
{
        OBJX=SumX/PixCounter;
        Form1->Edit2->Text=  OBJX-(curW/2);

        OBJY =SumY/PixCounter;
        Form1->Edit3->Text=  OBJY-(curH/2);
        if  ((OBJX-(curW/2))>50)
        {
                Form1->ShapeR->Brush->Style=bsSolid;
                Form1->ShapeL->Brush->Style=bsClear;
                strOut='L';

        }
        else if  ((OBJX-(curW/2))< -50)
        {
                Form1->ShapeL->Brush->Style=bsSolid;
                Form1->ShapeR->Brush->Style=bsClear;
                strOut='R';
         }
```

```
        else
        {
                Form1->ShapeR->Brush->Style=bsClear;
                Form1->ShapeL->Brush->Style=bsClear;
        }
        Form1->MSComm1->Output= strOut;
        if  ((OBJY-(curH/2))>50)
        {
                Form1->ShapeU->Brush->Style=bsSolid;
                Form1->ShapeD->Brush->Style=bsClear;
                strOut='U';
        }
        else if  ((OBJY-(curH/2))<-50)
        {
                Form1->ShapeD->Brush->Style=bsSolid;
                Form1->ShapeU->Brush->Style=bsClear;
                strOut='D';
        }
        else
        {
                Form1->ShapeU->Brush->Style=bsClear;
                Form1->ShapeD->Brush->Style=bsClear;
        }

        Form1->MSComm1->Output= strOut;
        Form1->Timer1->Enabled=false;
        if (Form1->CheckBox1->Checked ==true)
        {       TDateTime dtPresent = Now();
                DecodeDate(dtPresent, Year, Month, Day);

                DecodeTime(dtPresent, Hour, Min, Sec, MSec);
                Form1->Memo1->Text=  AnsiString("c:\\spic\\") +
IntToStr(Year) + " "+ IntToStr(Month)+ " " +IntToStr(Day)+" "+
IntToStr(Hour) +" "+ IntToStr(Min)+" "
+IntToStr(Sec)+AnsiString(".bmp")  ;

                CopyFile("temp.bmp", Form1->Memo1->Lines-
>GetText()   ,false);
        }
}
else
{
                Form1->ShapeR->Brush->Style=bsClear;
                Form1->ShapeL->Brush->Style=bsClear;

                Form1->ShapeU->Brush->Style=bsClear;
                Form1->ShapeD->Brush->Style=bsClear;
}

kk:
if (frame2<2)
        {frame2++;}
else
{ frame2=0;}
Form1->LabelPXC->Caption= PixCounter;
}

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
Running();
}
```

```
void __fastcall TForm1::ButtonFormatClick(TObject *Sender)
{
ShowFormat();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ButtonSourceClick(TObject *Sender)
{
ShowSource();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::SavesetClick(TObject *Sender)
{
SaveSetting();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ScrollBar1Change(TObject *Sender)
{
 LThreshold=ScrollBar1->Position;
 Label4->Caption= LThreshold;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ScrollBar2Change(TObject *Sender)
{
MThreshold=ScrollBar2->Position;
 LabelMove->Caption= MThreshold;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ScrollBarSpeedChange(TObject *Sender)
{
Speed=ScrollBarSpeed->Position;
LabelSpeed->Caption= Speed;
Timer2->Interval=Speed ;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
 strOut="XY";
 Form1->MSComm1->Output= strOut;
 Timer1->Enabled=true;
}
//---------------------------------------------------------------------------
```

```
#pragma hdrstop
#include <stdio.h>
#include "capture.h"
#include "Unit1.h"

#include <windows.h>
#include <avicap.h>
//---------------------------------------------------------------------------

#pragma package(smart_init)
extern int DeviceID;
   int curW,curH;
 HWND hCapWnd;
 CAPSTATUS capStat;
bool ConnectCapDriver(HWND hCapWnd , int nDriverIndex )
{
   bool retVal ;
   CAPDRIVERCAPS Caps;
   int I ;

    if(!( capDriverConnect(hCapWnd,  nDriverIndex))) return false;

    if  (!(capDriverGetCaps(hCapWnd,
&Caps,sizeof(CAPDRIVERCAPS))))return false;
    capPreviewRate(hCapWnd, 1);
    capPreview(hCapWnd, True) ;
    return  true ;
}

void Video_Startup(HWND hCapWndTaget )
{   bool  retVal;
    int CurrenDevice=1;

    hCapWnd = capCreateCaptureWindow("BCC CAP WINDOW", WS_CHILD |
WS_VISIBLE, 0, 0, 320, 240, hCapWndTaget, 0);

    if    (hCapWnd==0) return;

    retVal = ConnectCapDriver(hCapWnd, DeviceID);
    capPreviewRate(hCapWnd, 100);

     capPreviewScale(hCapWnd, True) ;
    capPreview(hCapWnd, True) ;
}
bool Captofile(char fn[])
{
return capFileSaveDIB(hCapWnd,fn);
}
void CaptoClipBoard (void)
{
     capPalettePaste(hCapWnd);
}
bool GrabFrame(void)
{
   return  capGrabFrame(hCapWnd);
}
void GrabFrameNoStop(void)
{
     capGrabFrameNoStop(hCapWnd);
}
void ShowFormat(void)
```

```
{
 capDlgVideoFormat(hCapWnd);
}
void ShowSource(void)
{
 capDlgVideoSource(hCapWnd);
}
void DriverDisconnect(void)
{
 capDriverDisconnect( hCapWnd);
}
```

```
STEP_MOTOR  EQU   P1

MOTOR_X           EQU   30H
MOTOR_Y           EQU   31H
SPEED       EQU   32H
DEGREEX           EQU   33H
DEGREEY           EQU   34H
LEFT_ENABLE BIT   0
RIGHT_ENABLE      BIT   1
UP_ENABLE   BIT   2
DOWN_ENABLE BIT   3
;--------------------------------

      ;CODE START
      ORG   0
      LJMP  MAIN

      ;INTERRUPT VECTOR OF TIMER/COUNTER OVERFLOW
      ORG   0BH
      LJMP  ON_TIMER0_OVER_FLOW


      ;INTERRUPT VECTOR OF RECIVE OR TRANSMIT SERIAL COMPLEATED
      ORG   23H
      LJMP  ON_SERIAL
;--------------------------------

ORG   100H

ON_TIMER0_OVER_FLOW:
      MOV   TH0,#060H
      PUSH  ACC
      PUSH  PSW

_NXL: JNB   LEFT_ENABLE,_NXR
      MOV   A,DEGREEX
      CLR   C
      SUBB  A,#228
      JNC   _NXR
      INC   DEGREEX
      CALL  STEP_LEFT

_NXR: JNB   RIGHT_ENABLE,_NYU

      MOV   A,DEGREEX
      CLR   C
      SUBB  A,#28
      JC    _NYU          ;1.8 *100 =180 DEGRREE
      DEC   DEGREEX
      CALL  STEP_RIGHT

_NYU:

      JNB   UP_ENABLE,_NYD

      MOV   A,DEGREEY
      CLR   C
      SUBB  A,#228
      JNC   _NYD          ;1.8 *100 =180 DEGRREE
      INC   DEGREEY
```

```
        CALL   STEP_UP

_NYD:   JNB    DOWN_ENABLE,_NEX

        MOV    A,DEGREEY
        CLR    C
        SUBB   A,#28
        JC     _NEX          ;1.8 *100 =180 DEGRREE
        DEC    DEGREEY

        CALL   STEP_DOWN

_NEX:   POP    PSW
        POP    ACC
        RETI
;--------------------------------
ON_SERIAL:
        PUSH   ACC
        PUSH   PSW

        JB     TI,_EXIT_SERIAL
        MOV    A,SBUF

        CJNE   A,#"L",_SX1
        SETB   LEFT_ENABLE
        CLR    RIGHT_ENABLE

_SX1:   CJNE   A,#"R",_SX2
        SETB   RIGHT_ENABLE
        CLR    LEFT_ENABLE

_SX2:   CJNE   A,#"U",_SX3
        SETB   UP_ENABLE
        CLR    DOWN_ENABLE

_SX3:   CJNE   A,#"D",_SX4
        SETB   DOWN_ENABLE
        CLR    UP_ENABLE

_SX4:   CJNE   A,#"X",_SX5
        CLR    LEFT_ENABLE
        CLR    RIGHT_ENABLE

_SX5:   CJNE   A,#"Y",_SX6
        CLR    DOWN_ENABLE
        CLR    UP_ENABLE

_SX6:

_EXIT_SERIAL:
        POP    PSW
        POP    ACC
        CLR    RI
        CLR    TI
        RETI
;--------------------------------

MAIN:   MOV    SP,#50H
        MOV    TH1,#0FDH     ;9600 BIT PER SEC
        MOV    SCON,#50H     ;
        MOV    TMOD,#21H     ;
```

```
        SETB   TR1          ;
        SETB   TR0          ;
        SETB   EA           ;
        SETB   ET0          ;
        SETB   ES
        MOV    MOTOR_X,#1H
        MOV    MOTOR_Y,#10H
        CLR    DOWN_ENABLE
        CLR    UP_ENABLE
        CLR    LEFT_ENABLE
        CLR    RIGHT_ENABLE
        MOV    DEGREEY,#128
        MOV    DEGREEX,#128
LOOP:
        LJMP   LOOP

;--------------------------------
STEP_LEFT:
              MOV    A,MOTOR_X
              RL     A
              JNB    ACC.4,_NO_X
              MOV    A,#1
_NO_X:              MOV    MOTOR_X,A
              LCALL  UPDATE_PORT
              RET
;--------------------------------
STEP_RIGHT:
              MOV    A,MOTOR_X
              RR     A
              JNB    ACC.7,_NO_X2
              MOV    A,#8
_NO_X2:             MOV    MOTOR_X,A
              LCALL  UPDATE_PORT
              RET
;--------------------------------
STEP_UP:
              MOV    A,MOTOR_Y
              RL     A
              JNB    ACC.0,_NO_Y
              MOV    A,#10H
_NO_Y:              MOV    MOTOR_Y,A
              LCALL  UPDATE_PORT
              RET
;--------------------------------
STEP_DOWN:
              MOV    A,MOTOR_Y
              RR     A
              JNB    ACC.3,_NO_Y2
              MOV    A,#80H
_NO_Y2:             MOV    MOTOR_Y,A
              LCALL  UPDATE_PORT
              RET
;--------------------------------
UPDATE_PORT:        MOV    A,MOTOR_Y
              ORL    A,MOTOR_X
              MOV    STEP_MOTOR,A
              RET
;--------------------------------
              END
```

หนังสืออ้างอิง

ยุทธนา ลีลาศวัฒนกุล, "เริ่มต้นการเขียนโปรแกรมด้วยภาษา C++", หจก.ไทยเจริญการพิมพ์, กรุงเทพ : พิมพ์ครั้งที่ 2 2547

ยุทธนา ลีลาศวัฒนกุล, "คู่มือการเขียนโปรแกรมและใช้งาน Visual C++ 6.0 ฉบับโปรแกรมเมอร์", สำนักพิมพ์ อินโฟเพรส, นนทบุรี : พิมพ์ครั้งที่ 1 2544

รศ.สมยศ จุณณะปิยะ, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์", สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, กรุงเทพ : พิมพ์ครั้งที่ 5 2546