ชุดทดสอบ ลูกเห็บกับแผงโซลาร์เซลล์

(HAIL TEST EQUIPMENT FOR SOLAR CELL'S PANEL)

โดย

นาย วัชระ การะพันธ์ รหัส 42015570

นาย สุกิจ เจริญจันทร์ รหัส 42015580

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

สาขาเทคโนโลยีอิเล็กทรอนิกส์ ภาควิชา เทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

(HAIL TEST EQUIPMENT FOR SOLAR CELL'S PANEL)

By

Mr. Watchara Karaphan ID.42015570

Mr. Sugit Charoenchan ID.42015580

Project Report Submitted in Partial Fulfillment of the Requirement

For the Bachelor's Degree

Department of Industrial Technology

Faculty of Engineer

King Mongkut's Institute of Technology Ladkrabang

2001

หัวข้อปริญญานิพนธ์	ชุดทดสอบลูกเห็บ กับแผงโซลาร์เซลล์
	(HAIL TEST EQUIPMENT FOR SOLAR CELL'S PANEL)
ชื่อนักศึกษา	นาย วัชระ การะพันธ์ รหัส 42015570
	นาย สุกิจ เจริญจันทร์ รหัส 42015580
อาจารย์ที่ปรึกษา	รศ.ประกิจ ดังติสานนท์
	ผศ.ดร.ปีติเขต สู้รักษา
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2544

ภาควิชาเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

..... หัวหน้าภาควิชา
( )

คณะกรรมการสอบปริญญานิพนธ์

..... อาจารย์ที่ปรึกษา
( )

..... อาจารย์ที่ปรึกษา
( )

..... กรรมการ
( )

..... กรรมการ
( )

..... กรรมการ
( )

Project Report	**(HAIL TEST EQUIPMENT FOR SOLAR CELL'S PANEL)**
By	Mr. Watchara Karaphan ID.42015570
	Mr. Sugit Charoenchan ID.42015580
Department	Industrial Technology
Porject Report Advisor	Assoc.Prof.Prakit Tangtosanon
	Asst.Prof.Dr.Pitikhate Sooraksa

Accepted by the Faculty of Engineer,King Mongkut's Institute of Technology Ladkrabang in Partial Fulfillment of the Requirement for the Bachelor's Degree

Project Report Committee

.....Chairman
( )

.....Chairman
( )

.....Member
( )

.....Member
( )

.....Member
( )

หัวข้อปริญญานิพนธ์	ชุดทดสอบลูกเห็บ กับแผงโซลาร์เซลล์		
	(HAIL TEST EQUIPMENT FOR SOLAR CELL'S PANEL)		
ชื่อนักศึกษา	นาย วัชระ	การะพันธ์	รหัส 42015570
	นาย สุกิจ	เจริญจันทร์	รหัส 42015580
อาจารย์ที่ปรึกษา	รศ.ประกิจ	ตังติสานนท์	
	ผศ.ดร.ปีติเขต	สู้รักษา	
ภาควิชา	เทคนิคอุตสาหกรรม		
ปีการศึกษา	2544		

บทคัดย่อ

จุดมุ่งหมายของปริญญานิพนธ์นี้เพื่อออกแบบและสร้างอุปกรณ์ที่ใช้ทดสอบมาตราฐานของแผงโซลาร์เซลล์ ด้านความสามารถในการรับแรงตกกระทบจากลูกเห็บโดยธรรมชาติ โดยอาศัยการทำงานของไมโครคอนโทรลเลอร์ MCS-51 และโปรแกรม VISUAL BASIC 6.0 มาใช้ควบคุมการทำงานและสร้างส่วนติดต่อสื่อสารระหว่างผู้ใช้กับคอมพิวเตอร์

ผลที่ได้จากปริญญานิพนธ์นี้สามารถนำไปพัฒนาเพิ่มเติมเพื่อเป็นชุดทดสอบที่สมบูรณ์ยิ่งขึ้น เพื่อใช้งานจริง

Project Report **(HAIL TEST EQUIPMENT FOR SOLAR CELL'S PANEL)**

By	Mr. Watchara Karaphan ID.42015570
	Mr. Sugit Charoenchan ID.42015580
Department	Industrial Technology
Porject Report Advisor	Assoc.Prof.Prakit Tangtisanon
	Asst.Prof.Dr.Pitikhate Sooraksa
Academic Year	2001

Abstract

This study aims to design and build the equipment for testing a solar cell panel for Tolerance from hails.The machanical equipment can interface with computer by a program developed by using Visual basic 6.0.

This protype can be extended to building in the real application.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้มิอาจลุล่วงไปได้หากขาดความช่วยเหลือจากบุคคลหลายๆท่าน ดังนั้นทางคณะผู้จัดทำจึงใคร่ขอขอบคุณบุคคลต่างๆดังต่อไปนี้

ขอขอบคุณ ผศ.ดร.ปิติเขต สู้รักษา ที่กรุณาให้คำแนะนำและเป็นที่ปรึกษาโครงงานนี้ด้วยดีตลอดมา พร้อมกันนี้ขอขอบคุณ รศ.ประกิต ตังติสานนท์ อาจารย์ บุญชนะ ภู่ระหงษ์ และคณาจารย์ภาควิชาเทคนิคอุตสาหกรรมทุกท่านที่ให้คำปรึกษาต่างๆ

สุดท้ายขอขอบคุณ เจตน์ ออสวัสดิ์ , สุมิตร จิรวัฒโฑ , เกรียงศักดิ์ ชัยสวัสดี ที่ให้คำปรึกษาด้านการเขียนโปรแกรมและวงจรควบคุมการทำงาน รวมถึงเพื่อนๆทุกคนที่คอยช่วยเหลือและเป็นกำลังใจให้เสมอมา และที่ขาดไม่ได้ พ่อ แม่ ที่เคารพรักของเรา...

คณะผู้จัดทำ

สารบัญ

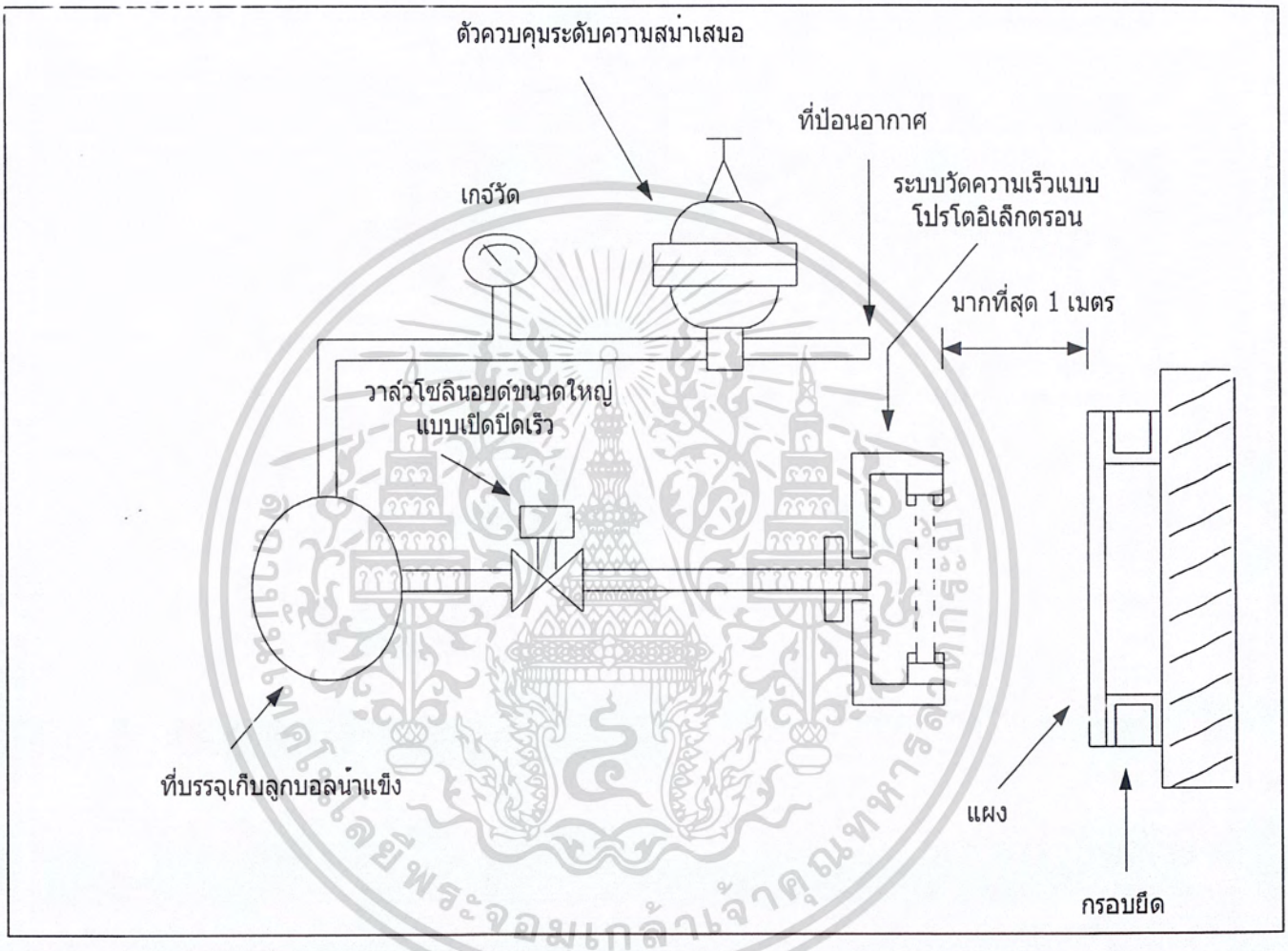
บทที่ 1
บทนำ

ความเป็นมาและสาเหตุจูงใจของโครงงาน

ในอนาคต การนำพลังงานจากแหล่งพลังงานทดแทนมาใช้งานมีแนวโน้มสูงขึ้น อาจจะด้วยสาเหตุหลายประการไม่ว่าจะเป็น แหล่งพลังงานหลักๆที่ใช้อยู่ในปัจจุบันเริ่มขาดแคลนและมีแนวโน้มว่าจะหมดไปหากมีการใช้อย่างสิ้นเปลือง หรือค่อนข้างจะมีราคาสูงขึ้น ฯลฯ

พลังงานแสงอาทิตย์ก็เป็นอีกแหล่งหนึ่งที่เริ่มมีการนำมาใช้กันอย่างกว้างขวาง อาจเป็นเพราะว่าเป็นแหล่งพลังงานที่ได้มาโดยไม่ต้องไปแสวงหา และเป็นแหล่งพลังงานที่ไม่มีวันหมดสิ้น ถึงแม้ว่าจะต้องลงทุนค่อนข้างสูงในตอนแรก แต่ก็นับว่าคุ้มค่ากับการลงทุนเมื่อเทียบกับผลตอบแทนที่จะได้รับ หนึ่งในส่วนของการลงทุนนั้นก็คือ “แผงโซล่าร์เซลล์” (Solar Cell) ซึ่งเป็นอุปกรณ์หลักที่ใช้ผลิตกระแสไฟฟ้าและเป็นอุปกรณ์ที่ค่อนข้างจะมีราคาสูง ดังนั้นแผงโซล่าร์เซลล์ที่เราจะนำมาใช้จึงต้องมีคุณภาพที่ดี มีอายุการใช้งานที่ยาวนานคุ้มค่ากับการลงทุน

จากสาเหตุนี้จึงทำให้เกิดแนวคิดและเหตุจูงใจของโครงงานนี้ขึ้นมา อันเนื่องมาจากการที่ทาง สำนักงานวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (สวทช.) และ สพช.(สำนักงานนโยบายพลังงานแห่งชาติ) ได้ร่างกำหนดมาตราฐานของแผง โซล่าร์เซลล์ที่ใช้ในประเทศไทยขึ้นมา เพื่อให้มีความเหมาะสมกับสภาพภูมิอากาศและสภาพแวดล้อมของประเทศไทย และหนึ่งหัวข้อของร่างมาตราฐานนี้ก็คือ “ความทนทานต่อแรงกระทบจากลูกเห็บ” ซึ่งวัตถุประสงค์หลักของโครงงานนี้ก็คือ สร้างเครื่องมือหรือชุดทดสอบลูกเห็บ (Hail -Test)ขึ้นมารองรับมาตราฐานที่ได้กำหนดขึ้นเพื่อนำไปใช้ทดสอบการใช้งานซึ่งมีที่มาจากแนวคิดขั้นต้นดังรูป



วัตถุประสงค์ของการทำโครงงานมีดังนี้

-เพื่อสร้าง ฮาร์ดแวร์(Hard Ware) มารองรับมาตราฐานที่กำหนดขึ้น

-เพื่อศึกษาถึงการใช้งานไมโครคอนโทรลเลอร์(Microcontroller) MCS-51 ที่จะใช้ควบคุมการทำงานของชุดทดสอบฯ

-เพื่อศึกษาถึงการสร้างซอร์ฟแวร์ (Soft Ware) ควบคุมการทำงานโดยโปรแกรม วิชวล เบสิก (Visual Basic) และภาษา แอสแซมบลี (Assembly)

-เพื่อศึกษาการใช้งานคอมพิวเตอร์(Computer) ควบคุมอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม

ขอบเขตของโครงการมีดังนี้คือ

-สามารถยิงลูกบอลน้ำแข็งได้ตรงตามตำแหน่งที่ถูกต้องด้วยความเร็วที่กำหนด

-สามารถควบคุมการทำงานได้ 2 Mode คือ

1. Manual Mode คือการ ควบคุมตำแหน่งการยิงโดย User ผ่าน Computer ตามตำแหน่งที่ต้องการ
2. Automatic Mode คือการควบคุมตำแหน่งการยิงอัตโนมัติ ตามตำแหน่งที่กำหนดไว้โดย Microcontroller

ขั้นตอนการดำเนินงาน

1. ศึกษาการใช้ Program Visualbasic6.0 และการใช้ Computer ควบคุมอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม
2. ศึกษาการเขียนโปแกรมภาษา Assembly เพื่อใช้ควบคุม Microcontroller
3. ศึกษาการใช้งาน Microcontroller MCS-51 เพื่อควบคุมอุปกรณ์ต่างๆ
4. ออกแบบโครงสร้างทาง Hard Ware
5. ออกแบบโปรแกรมที่ใช้ควบคุมการทำงานผ่าน Computer
6. ทดสอบการทำงานของ Hard Ware ร่วมกับโปรแกรมที่เขียนขึ้น
7. แก้ไขข้อผิดพลาดต่างๆที่เกิดขึ้น
8. ทดสอบการทำงานของระบบทั้งหมด
9. วิเคราะห์ถึงข้อบกพร่องต่างๆของโครงงานและตรวจสอบความถูกต้องของการใช้งาน
10. สรุปผลและจัดทำรายงานพร้อมทั้งนำเสนอผลงาน

บทที่ 2 ทฤษฎีและหลักการของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ที่จะกล่าวถึงนี้เป็นไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งมีหน่วยความจำภายในเป็นแบบแฟลซ (flash memory) ของบริษัท Atmel Corperration มีเบอร์ขึ้นต้นด้วย AT89xx ซึ่งมีข้อดีอยู่หลายประการ

คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51 Series AT89xx

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมแบบแฟลซซึ่งสามารถลบและเขียนใหม่ได้พันครั้ง
- หน่วยความจำพื้นฐานเป็นหน่วยความจำแบบแรม(RAM) ในบางเบอร์จะมีหน่วยความจำแบบอีพรอม(EPROM)เพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทางสามารถใช้งานได้ทั้งอินพุตและเอาต์พุท
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเฟล็กซ์
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์(Interrupt)ได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาภายในชิป
- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
- มีวอตช์ด็อกไทเมอร์ในตัว สำหรับในอนุกรม AT89Sxx

โครงสร้างของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลซ แสดงดังรูปที่ 1

การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

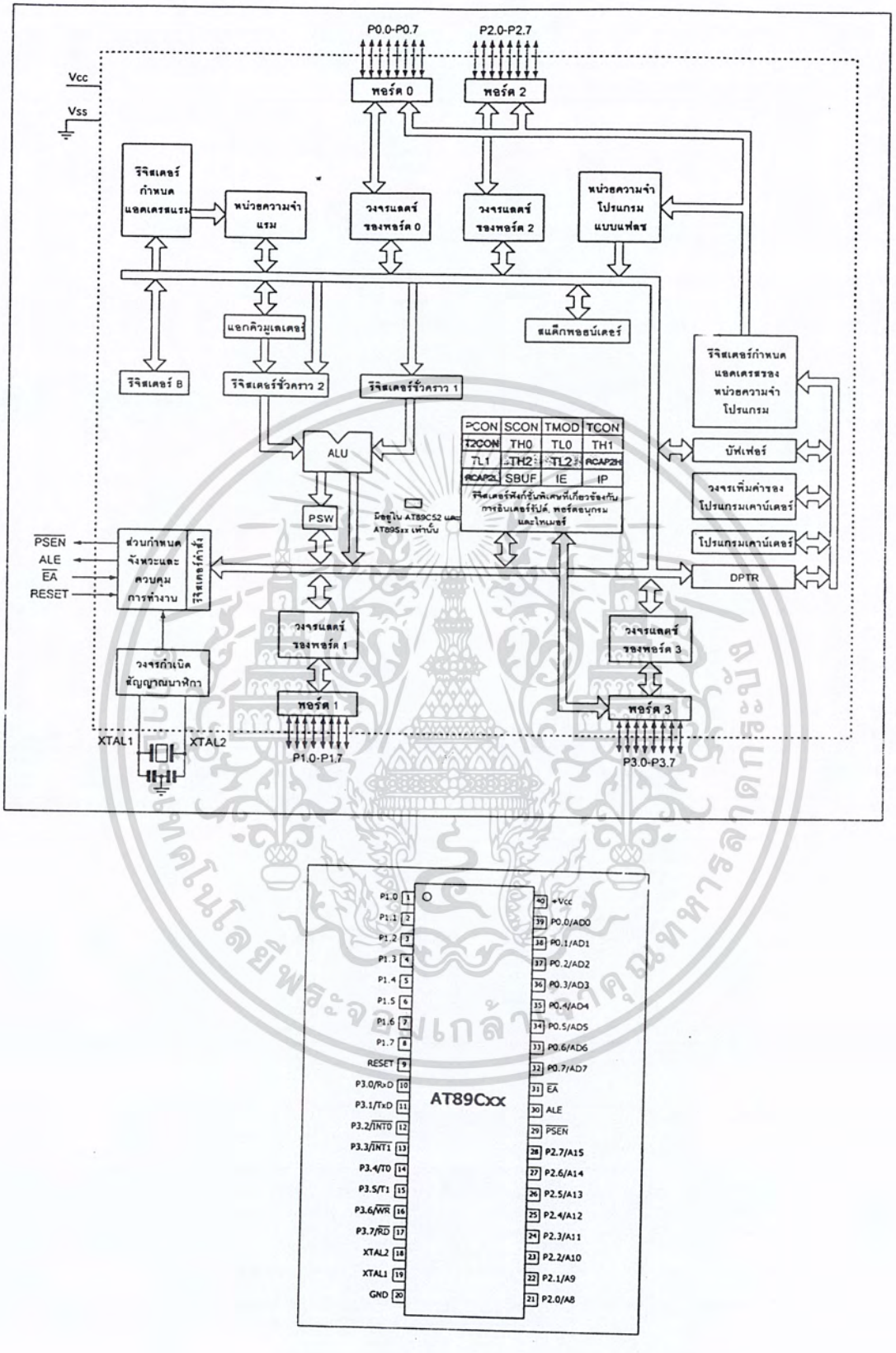
ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานเหมือนกัน โดยมีรายละเอียดดังต่อไปนี้

VCC : สำหรับแหล่งจ่ายไฟ (+5V)

GND : สำหรับต่อกราวด์

PO (P0.0-P0.7) : มี 8 ขา แต่สามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตที่ต้องการติดต่อด้วย ส่งผลให้ขานั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง(High Input Impedance)สามารถใช้งานเป็นขาอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7)

โดยใช้กระบวนการมัลติเพล็กซ์ (Multiplex) เข้าช่วย เพื่อสลับการทำงานให้เป็นได้ทั้งขาแอดเดรส และขาข้อมูล



รูปที่1. โครงสร้างหลักของ MCS-51/**รูปที่2.** การจัดขามาตราฐานของMCS-51

P1 (P1.0-P1.7) : มี 8 ขา แต่สามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป หากต้องการให้เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังบิตนั้น นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และ P1.1 เป็นขาอินพุตทริกเกอร์ของไทเมอร์ 2 ในขณะที่ P1.4- P1.7 ใช้เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

P2 (P2.0-P2.7) มี 8 ขาสำหรับใช้งานเป็นทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไปนอกจากนั้นยังถูกใช้งานเป็นแอดเดรสไบต์สูงของหน่วยความจำ (A8-A15)

P3 (P3.0- P3.7) : มี 8 ขา สามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไปเช่นกัน นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีการใช้งานพิเศษ ดังมีรายละเอียดขั้นต้นดังต่อไปนี้

ขาพอร์ต	หน้าที่พิเศษ
P3.0	RxD (สำหรับรับข้อมูลแบบอนุกรม)
P3.1	RxD (สำหรับส่งข้อมูลแบบอนุกรม)
P3.2	INT0 (ขาอินเตอร์รัพท์ภายนอก 0)
P3.3	INT1 (ขาอินเตอร์รัพท์ภายนอก 1)
P3.4	T0 (ขาอินพุตของ Timer 0)
P3.5	T1 (ขาอินพุตของ Timer 1)
P3.6	WR (สำหรับเขียนหน่วยความจำข้อมูลภายนอก)
P3.7	RD (สำหรับอ่านหน่วยความจำข้อมูลภายนอก)

RST : สำหรับรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยให้ลอจิก(Logic)หนึ่งเป็นเวลาอย่างน้อย 2 ช่วงแมชชีน ไซเคิล (Machine Cycle)

ALE/PROG : เป็นขาที่ใช้ในการควบคุมการแลตซ์ (Latch) ของพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังเป็นขาสำหรับพัลซ์ (Pulse) ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบ อีอีพรอม(EEPROM)

PSEN : ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหนน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์

จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละ Machine Cycle แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่งสัญญาณใดๆออกมา

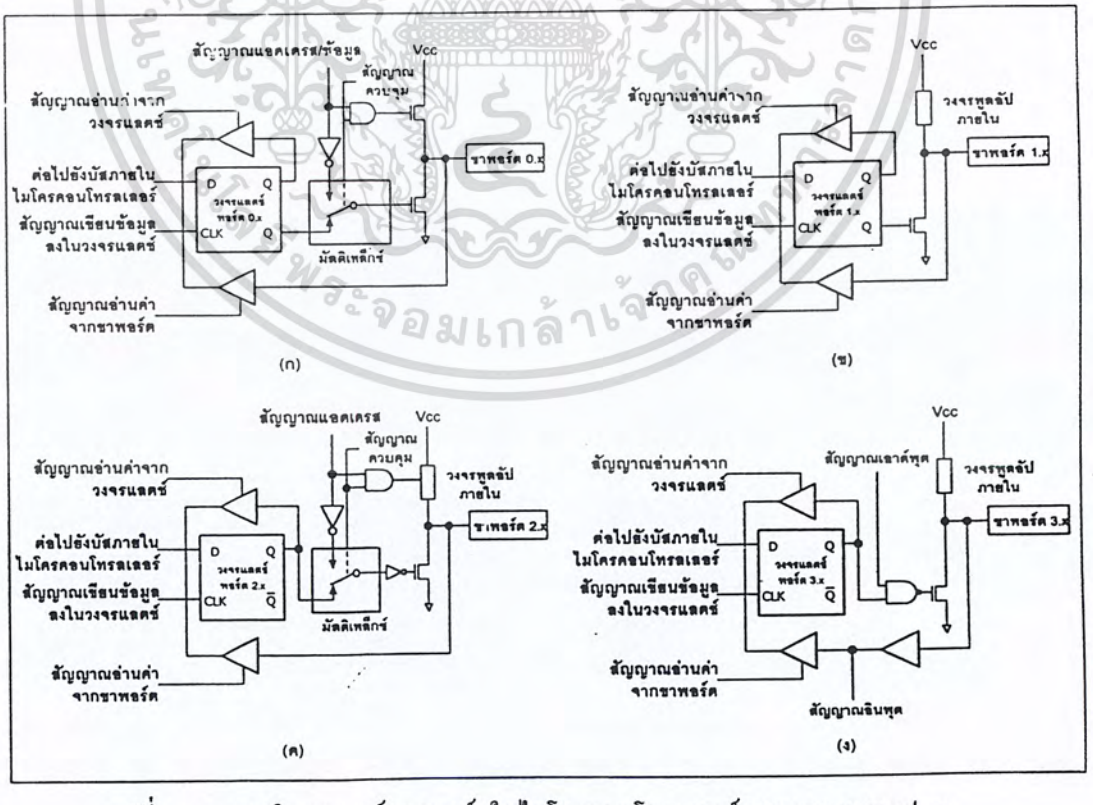
EA/Vpp : ใช้สำหรับเลือกการติดต่อกับหน่วยความจำโปรแกรมภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น 0 เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขานี้เป็น 1 เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ขานี้ยังใช้เป็นขาอินพุตสำหรับแรงดันไฟสูงสำหรับการโปรแกรมหน่วยความจำภายในตัวไมโครคอนโทรลเลอร สำหรับไมโครอนโทรลเลอร์แบบแฟลซต้องการแรงดันสำหรับการโปรแกรม +12 V

XTAL1: ขาเข้าของวงจรกำเนิดความถี่อ้างอิงภายในของ MCS-51

XTAL2: ขาออกของวงจรกำเนิดความถี่อ้างอิงภายในของ MCS-51

โครงสร้างและการทำงานของพอร์ต

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช มีพอร์ตใช้งานทั้งสิ้น 4 พอร์ตคือ พอร์ต 0 ถึง พอร์ต 4 ทุกพอร์ตของไมโครคอนโทรลเลอร์แบบแฟลชมีวงจรแลตช์และวงจรขับตลอดจนบัฟเฟอร์อินพุตดังแสดงให้เห็นในสถาปัตยกรรมดังรูปที่ 3.



รูปที่ 3. วงจรภายในของพอร์ตทุกพอร์ตในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ที่พอร์ต 0 และพอร์ต 2 จะใช้งานเป็นพอร์ตอินพุตและเอาต์พุตสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ตและพอร์ต 1 บางขานอกจากจะใช้เป็นอินพุตและเอาต์พุตตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็นไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด

จากรูปแสดงวงจรภายใน รูป (ก) เป็นวงจรของพอร์ต 0 วงจรแลตซ์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิบฟลอบนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตซ์สามารถกระทำได้อย่างอิสระต่อกัน ด้วยสัญญาณที่แยกจากกันนั่นคือสัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องทำการส่งสัญญาณมายังขา CLK ของดีฟลิปฟลอปในขณะที่ข้อมูลจะถูกส่งมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิปฟลอบ

ที่พอร์ตนี้วงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ตว่า ต้องการใช้งานเป็นขาอินพุตเอาต์พุตปกติหรือใช้ในการติดต่อหน่วยความจำภายนอกไมโครคอนโทรลเลอร์

เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรพูลอัป(Pull Up) ภายใน หากมีการนำพอร์ต 0 ไปใช้งานเป็นพอร์ตอินพุตจะต้องต่อตัวต้านทานพูลอัปภายนอกที่ขาพอร์ต 0 ทุกขาด้วย

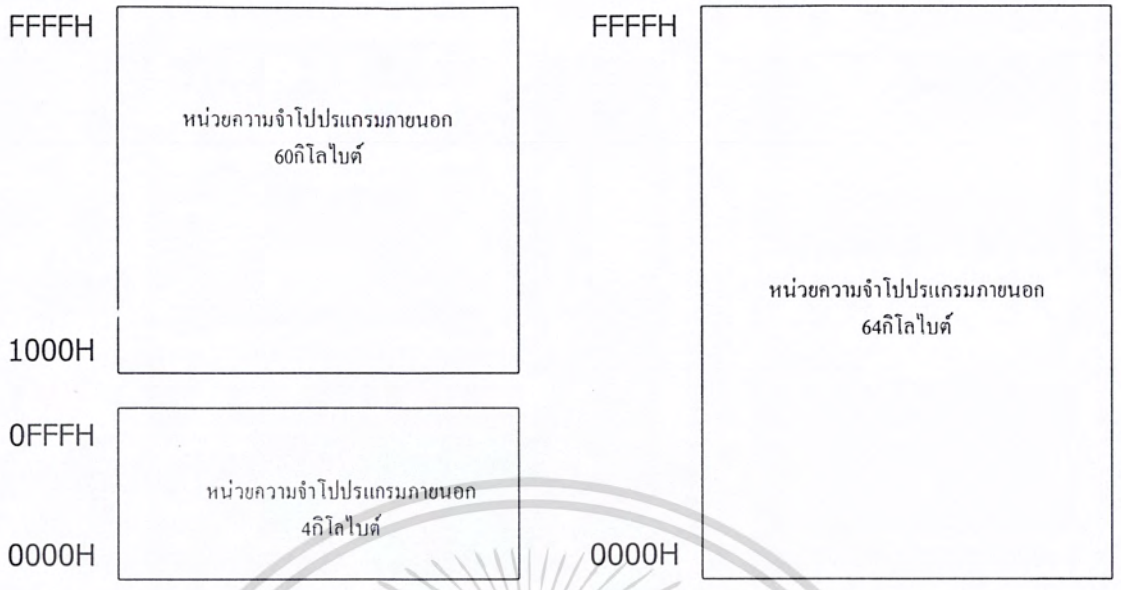
โครงสร้างหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีหน่วยความจำภายในหลักๆ อยู่ 2 ส่วนคือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ซึ่งมีขนาดและการจัดสรรต่างกันไปในแต่ละเบอร์ซึ่งจะได้กล่าวถึงต่อไป

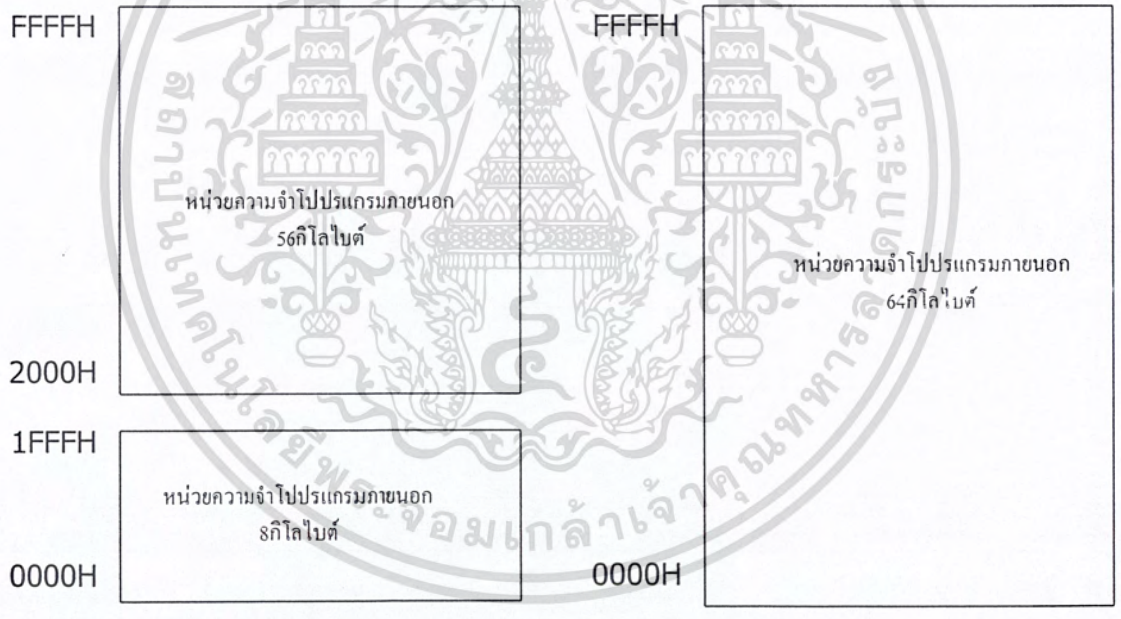
หน่วยความจำโปรแกรม (Program Memory)

ในรูปที่4. แสดงการจัดหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ต่างๆที่นิยมใช้งาน อันประกอบด้วยเบอร์ AT89C51 และ AT89C52 จะเห็นได้ว่าทั้งสองเบอร์สามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ โดยสามารถเลือกใช้หน่วยความจำโปรแกรมภายในอย่างเดียวหรือรวมกับภายนอกหรือใช้ภายนอกอย่างเดียวก็ได้ ซึ่งภายใน AT89C51 จะมีหน่วยความจำโปรแกรมภายใน 4 กิโลไบต์ ในขณะที่เบอร์ AT89C52 จะมีขนาด 8 กิโลไบต์

ในกรณีที่ใช้หน่วยความจำภายในและภายนอกรวมกัน(รูปซ้าย) หากใช้ AT89C51 ก็จะสามารถติดต่อหน่วยความจำภายนอกได้ 60 กิโลไบต์ และถ้าใช้เบอร์ AT89C52 จะสามารถติดต่อหน่วยความจำโปรแกรมภายนอกได้ 56 กิโลไบต์



(ก) การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์เบอร์ AT89c51



(ข) การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์เบอร์ AT89c52

รูปที่ 4. แสงการจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

หน่วยความจำข้อมูล (Data Memory)

มีด้วยกัน 2 แบบคือ หน่วยความจำข้อมูลภายในและภายนอกโดยไมโครคอนโทรลเลอร์แบบแฟลชอนุกรม AT89xx สามารถติดต่อกับหน่วยความจำข้อมูลภายนอกได้สูงสุด 64 กิโลไบต์ โดยการใช้คำสั่ง MOVX ในการติดต่อกับหน่วยความจำข้อมูลภายนอก สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชทุกเบอร์ จะมีหน่วยความจำภายในเป็นแบบแรม (Random Access Memory) โดยแต่ละเบอร์จะมีขนาดแตกต่างกันไป ในเบอร์ AT89C51 มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์ ในขณะที่เบอร์ AT89C52 มีขนาด 256 ไบต์

สำหรับการจัดสรรหน่วยความจำข้อมูลภายในแบ่งเป็น 3 ส่วนโดยมีรายละเอียดดังนี้

-ส่วนที่ 1 เรียกว่า Register Banks 0-3 ซึ่งอยู่ที่ตำแหน่งหน่วยความจำภายในตั้งแต่ 00H ถึง 1FH จำนวน 32 ไบต์โดยจะแบ่งออกเป็นชุด ชุดละ 8 ไบต์จำนวน 4 ชุด ซึ่งแต่ละชุดจะมีชื่อเรียกเป็น R0 ถึง R7 จะเป็น Register ที่ใช้งานเมื่อ MCS-51 ถูกรีเซ็ต Register Bank 0 จะถูกเลือกใช้

-ส่วนที่ 2 เรียกว่า Bit Addressable Area ซึ่งมีขนาด 16 ไบต์ที่ตำแหน่งหน่วยความจำข้อมูล 20H ถึง 2FH ในส่วนนี้สามารถที่จะอ้างอิงข้อมูลได้ในระดับบิตถึง 128 บิต โดยการอ้างอิงตำแหน่งโดยตรงในลักษณะบิต ตั้งแต่ตำแหน่ง 00H ถึง 7FH

-ส่วนที่ 3 เรียกว่า Scratch Pad Area จะอยู่ที่ตำแหน่ง 30H ถึง 7FH ซึ่งเป็นบริเวณหน่วยความจำข้อมูลภายในเอนกประสงค์ที่ผู้ใช้สามารถใช้ได้โดยตรง นอกจากนี้สามารถใช้หน่วยความจำข้อมูลบริเวณนี้สำหรับการเก็บข้อมูลแบบสแตค (Stack) ได้ด้วย

ในส่วนของหน่วยความจำข้อมูลภายในที่อ้างอิงแบบไดเร็ค (Direct) เพียงอย่างเดียวหรือที่เรียกว่า SFR ซึ่งเป็นส่วนสำหรับเก็บหรือกำหนดการทำงานภายในของ MCS-51 ดังแสดงในรูปนั้น บริเวณนี้จะมีขนาด 128 ไบต์แต่ในการใช้งานนั้นใช้ได้เฉพาะตำแหน่งที่แสดงในรูป เท่านั้น หากผู้ใช้อ้างตำแหน่งที่นอกเหนือจากนี้จะได้ข้อมูลที่คาดเดาไม่ได้ โดยแต่ละตำแหน่งจะมีหน้าที่ดังนี้

ACC : เป็น Accumulator ซึ่งเป็นรีจีสเตอร์สำหรับการประมวลผลทางคณิตศาสตร์และลอจิก โดยผู้ใช้สามารถอ้างอิงในรูปแบบของไบต์หรือระดับบิตได้

B : เป็นรีจีสเตอร์พิเศษสำหรับใช้กับคำสั่งคูณหรือคำสั่งหาร นอกจากนี้ยังใช้เป็นรีจีสเตอร์สำหรับเก็บพักข้อมูลได้

PSW : เป็นรีจีสเตอร์ Program Status Word หรือแฟลกจะแสดงสถานะการทำงานของ MCS-51 สำรับการตรวจสอบ

SP : เป็นรีจีสเตอร์สำหรับชี้หน่วยควมจำข้อมูลภายในสำหรับการเก็บแบบ Stack

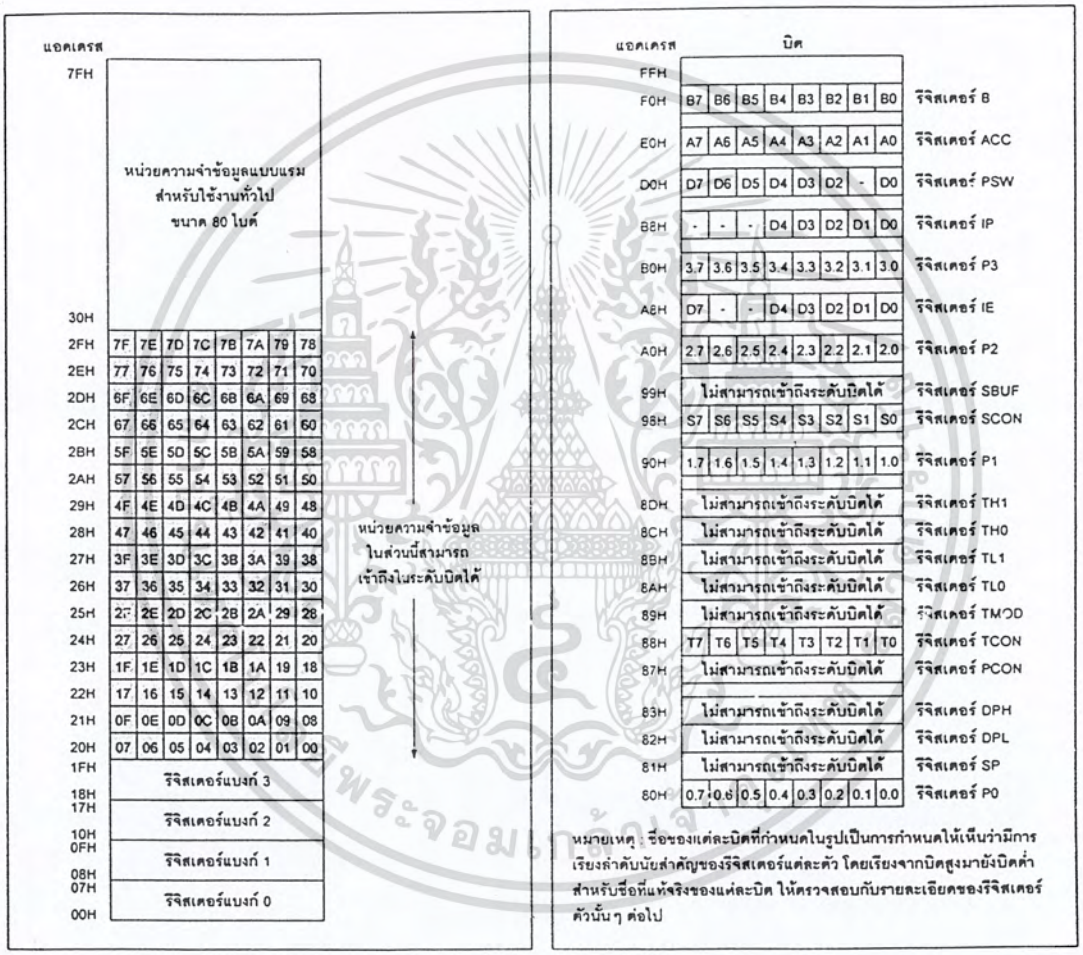
DPTR : เป็นรีจีสเตอร์ขนาด 16 บิต โดยแบ่งเป็น 8 บิตบนและ 8 บิตล่างใช้สำหรับชี้ตำแหน่งของหน่วยความจำข้อมูลภายนอกหรือสำหรับการอ่านตารางข้อมูลของหน่วยความจำโปรแกรม

P0 : เป็นรีจีสเตอร์สำหรับพอร์ต 0 ของ MCS-51

P1 : เป็นรีจีสเตอร์สำหรับพอร์ต 1 ของ MCS-51

P2 : เป็นรีจีสเตอร์สำหรับพอร์ต 2 ของ MCS-51

P3 : เป็นรีจีสเตอร์สำหรับพอร์ต 3 ของ MCS-51



รูปที่ 5. โครงสร้างของหน่วยความจำข้อมูล/การจัดสรรพื้นที่ของ SFR

IP : เป็นรีจีสเตอร์สำหรับกำหนดลำดับความสำคัญของการอินเตอร์รัพท์ ของ MCS-51

IE : เป็นรีจีสเตอร์สำหรับกำหนดการรับหรือไม่รับการอินเตอร์รัพท์

TMOD: เป็นรีจีสเตอร์สำหรับควบคุมหน้าที่ของ Timer/Counter ของ MCS-51

TCON: เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ Timer/Counter ของ MCS-51

T2CON: เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ Timer/Counter2 ของ 8052

TH0 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 0 8บิตบน

TL0 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 0 8บิตล่าง

TH1 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 1 8บิตบน

TL1 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 1 8บิตล่าง

TH2 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timre/Counter 2 8บิตบน

TL2 : เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 2 8บิตล่าง

RCAP2H : เป็น Capture Register ของ Timer/Counter2 บิตบนของ 8052

RCAP2L : เป็น Capture Register ของ Timer/Counter2 บิตล่างของ 8052

SCON : เป็นรีจีสเตอร์สำหรับควบคุมการรับส่งข้อมูลแบบอนุกรมของ MCS-51

SBUF : เป็นรีจีสเตอร์สำหรับเก็บพักข้อมูลที่ได้จากการรับส่งข้อมูลแบบอนุกรมของ MCS-51

PCON : เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ MCS-51 ด้านเกี่ยวกับการใช้กำลังไฟฟ้า

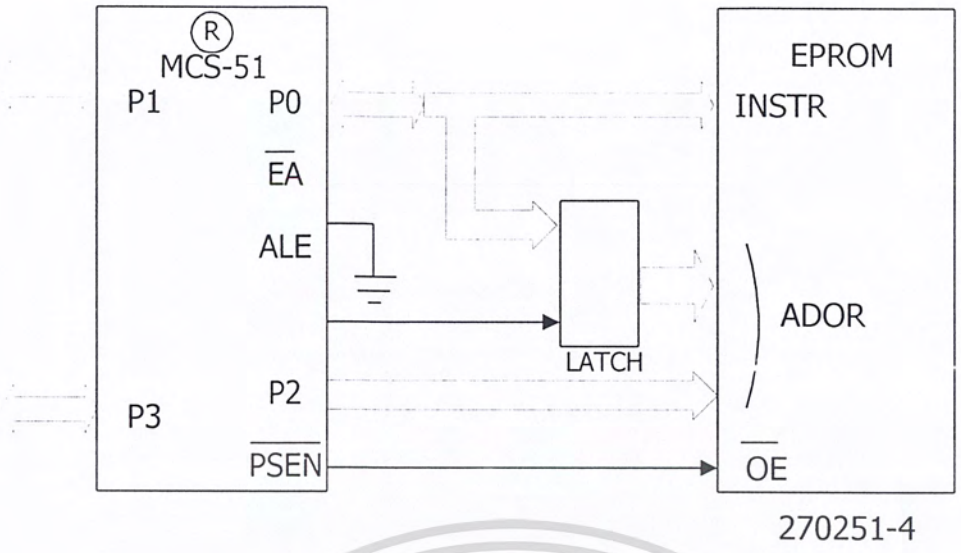
การต่อหน่วยความจำภายนอก

สำหรับการต่อหน่วยความจำภายนอกสามารถแบ่งได้เป็น 2 ส่วน ได้แก่หน่วยความจำโปรแกรม และหน่วยความจำข้อมูล เนื่องจาก MCS-51 นั้นส่วนดาตา (Data) จะใช้ขาร่วมกับ Address 8บิตล่างดังนั้นในการใช้งานหน่วยความจำภายนอกจะต้องมีอุปกรณ์ Latch สำหรับสัญญาณ Address โดยอุปกรณ์ Latch ดังกล่าวได้แก่ 74LS373 โดยสามารถแยกการใช้งานดังนี้

หน่วยความจำโปรแกรม

เนื่องจาก MCS-51 มีหน่วยความจำโปรแกรมขนาด 4 กิโลไบต์ ดังนั้นจึงขึ้นอยู่กับการใช้งานหากต้องการใช้หน่วยความจำโปรแกรมภายในร่วมกับหน่วยความจำโปรแกรมภายนอกจะต้องต่อขา EA ของ MCS-51 ไวที่ Vcc โดยจะมีตำแหน่งของหน่วยความจำโปรแกรมดังนี้

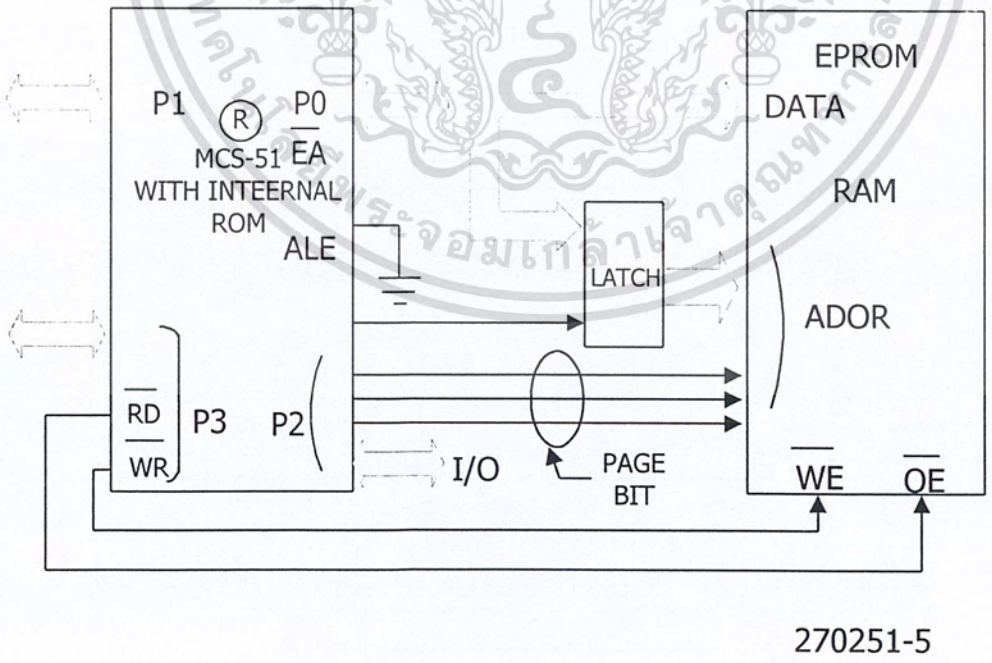
ตำแหน่งของหน่วยความจำภายในเริ่มตั้งแต่ 0000H ถึง 0FFFH หน่วยความจำโปรแกรมภายนอกเริ่มจาก 1000H ถึง FFFFH หากใช้หน่วยความจำภายนอกทั้งหมดขา EA ของMCS-51 จะต่อกับ GND จะได้หน่วยความจำโปรแกรมภายนอกจาก 0000H ถึง FFFFH จากรูปแสดงการต่อหน่วยความจำภายนอกโดยใช้สัญญาณ ALE เป็นสัญญาณให้อุปกรณ์ Latch รับข้อมูลและสัญญาณ PSEN สำหรับให้อุปกรณ์หน่วยความจำโปรแกรมทำงาน ในกรณีที่หน่วยความจำโปรแกรมหลายตัวก็สามารถใช้อุปกรณ์ดีโค้ดเดอร์ (Decoder) ช่วยในการเลือกอุปกรณ์ได้



รูปที่6.แสดงการต่อหน่วยความจำโปรแกรมภายนอก

หน่วยความจำข้อมูลภายนอก

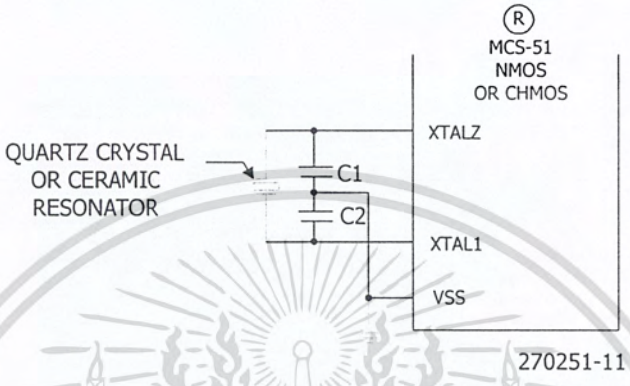
ในการต่อหน่วยความจำข้อมูลภายนอก สามารถทำได้เช่นเดียวกับหน่วยความจำโปแกรม เพียงแต่จะใช้สัญญาณ RD,WR สำหรับในการอ่านและเขียนหน่วยความจำภายนอก ดังแสดงในรูป



รูปที่7 แสดงการต่อหน่วยความจำข้อมูลภายนอก

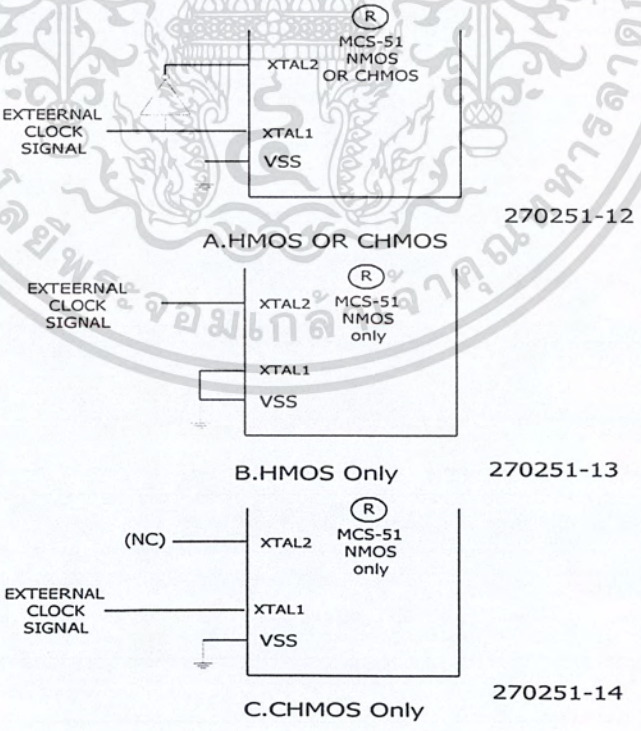
การต่อวงจรฐานเวลาของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 จะมีส่วนสร้างความถี่ภายในเพียงแต่ต่อคริสตัล Crystal หรือ Ceramic Resonator ระหว่างขา XTAL1 กับ XTAL2 และตัวเก็บประจุขนาด 30 pf ลงกราวด์ดังแสดงในรูป



รูปที่8. แสดงการต่อ Crystal หรือ Ceramic Resonator

นอกจากนี้แล้วยังสามารถใช้แหล่งสร้างความถี่ภายนอกได้ ขึ้นอยู่กับโครงสร้างภายในของ MCS-51 ดังแสดงในรูป ซึ่งผู้ใช้จะต้องระมัดระวังการใช้งาน



รูปที่ 9.การใช้ความถี่จากภายนอก

การคำนวณความเร็วการรับส่งข้อมูลอนุกรม (Generating Baud Rate)

การกำหนดความเร็วสำหรับการรับส่งข้อมูลแบบอนุกรมสามารถแบ่งออกได้ตาม Mode การทำงานดังนี้

Mode 0

ความเร็วการรับส่งข้อมูลแบบอนุกรมใน Mode นี้จะกำหนดอัตราการรับส่งตายตัวเท่ากับ 1/12 ของความถี่ของชุดกำเนิดความถี่อ้างอิงของ MCS-51 และจะไม่ใช้ Timer/Counter ดังนั้นเพียงกำหนดที่รีจีสเตอร์ SCON ก็เพียงพอ จะได้

Baud Rate = Osc.Freq/12

Mode 1

ในการกำหนดความเร็วการรับส่งข้อมูลแบบอนุกรมใน Mode 1 นี้จะใช้ Timer1 เป็นฐานเวลาของการทำงาน โดยจะใช้การทำงานของ Timer1 ใน Mode2 (Auto-Reload) โดยสามารถคำนวณได้ดังนี้

Baud Rate = K*Osc.Freq/32*12*[256-(TH1)]

K = 1 เมื่อ SMOD ในรีจีสเตอร์ PCON = 0

K = 2 เมื่อ SMOD ในรีจีสเตอร์ PCON = 1

ส่วนมากแล้วผู้ใช้จะทราบค่าของ Baud Rate ที่จะส่งนั้น จะได้ค่าของ Timer 1 สำหรับ Reload ได้เป็น

TH1 = 256-[K*Osc.Freq/384*Baud Rate]

จากตารางต่อไปนี้แสดงค่า Baud Rate ต่างๆและค่า Reload ของ Timer1

Baud Rate	fosc	SMOD	Timer1		
			C/$\overline{T}$	Mode	Reload Value
Mode 0 Macx: 1MHz	12MHz	x	x	x	x
Mode 2 Macx: 375k	12MHz	1	x	x	x
Modes 1.3: 62.5k	12MHz	1	0	2	FFH
19.2k	11.059MHz	1	0	2	FDH
19.2k	11.059MHz	1	0	2	FDH
4.8k	11.059MHz	0	0	2	FAH
2.4k	11.059MHz	0	0	2	F4H
1.2k	11.059MHz	0	0	2	E8H
137.5	11.059MHz	0	0	2	1DH
110	6MHz	0	0	2	72H
110	12MHz	0	0	1	FEEBH

Mode 2

ความเร็วการรับส่งในMode นี้จะเป็นค่าคงที่ซึ่งมี 2 ค่า ขึ้นอยู่กับค่า SMOD ในรีจีสเตอร์ PCON ดังนี้

เมื่อ SMOD = 1 Baud Rate = 1/32 Osc.Freq

เมื่อ SMOD = 0 Baud Rate = 1/64 Osc.Freq

Mode 3

การกำหนดความเร็วการรับส่งใน Mode 3 จะคิดเช่นเดียวกับ Mode 1

บทที่ 3 การเชื่อมต่อ Computer กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม

ในการที่จะเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพ่วงอื่นๆหรือคอมพิวเตอร์ด้วยกันนั้นมีทางเลือกอยู่ 2 ทาง คือ การรับส่งข้อมูลแบบขนานและแบบอนุกรม การรับส่งข้อมูลแบบขนานจะเป็นการรับส่งข้อมูลคราวละ 4 บิตหรือ 8 บิตในเวลาเดียวกัน ซึ่งจะทำให้การรับส่งข้อมูลทำได้ที่ความเร็วสูง ซึ่งก็หมายความว่าจำนวนของสายที่ใช้ในการส่งจะต้องมีเท่ากับจำนวนบิตของข้อมูลด้วย ในขณะที่การรับส่งข้อมูลแบบอนุกรมเป็นการรับส่งข้อมูลครั้งละ 1 บิต แต่ก็สามารถรับส่งข้อมูลคราวละหลายๆบิตได้ หากแต่จะต้องมีการตกลงกัน ระหว่างตัวส่งและตัวรับว่าจะมีการรับส่งข้อมูลคราวละกี่บิต ตัวรับจะต้องรอข้อมูลให้ครบทุกบิตเสียก่อนจึงจะทำการประมวลผลส่ง ผลให้การสื่อสารข้อมูลอนุกรมมีความเร็วต่ำกว่าแบบขนาน ในด้านจำนวนสายการรับส่งข้อมูลแบบอนุกรมใช้จำนวนสายที่น้อยกว่ามาก อย่างน้อยที่สุดใช้เพียง 2-3 เส้นเท่านั้น แต่อัตราเร็วในการรับส่งข้อมูลอาจต่ำกว่าแบบขนาน อย่างไรก็ตามการรับส่งข้อมูลแบบอนุกรมสามารถใช้สายสัญญาณที่มีความยาวมากกว่าแบบขนาน ทำให้ระยะทางในการสื่อสารแบบอนุกรมสามารถทำได้มากกว่า

การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมแบ่งเป็น 2 แบบคือการสื่อสารแบบอนุกรมซิงโครนัส (Synchronous) และการสื่อสารอนุกรมแบบอะซิงโครนัส (Asynchronous) การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับส่งสัญญาณด้วย ตัวอย่างเช่น คีย์บอร์ดของคอมพิวเตอร์ซึ่งสายสัญญาณเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกาส่วนอีกสายจะเป็นสายของสัญญาณข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อย 3 เส้น คือ สัญญาณนาฬิกา, ข้อมูลและกราวด์

ส่วนการสื่อสารแบบอะซิงโครนัสนั้น คือการรับส่งข้อมูลไปโดยไม่มีสัญญาณนาฬิการ่วมด้วยแต่จะใช้การการกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากันซึ่งเรียกว่า อัตราการส่งถ่ายข้อมูล หรือ บอดเรต (Baudrate) มีหน่วยเป็น บิตต่อวินาที (bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (Start Bit) ซึ่งจะมีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมจะมีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity Bit) จะมีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้าย (Stop Bit) จะมีขนาด 1,1.5 หรือ 2 บิต

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับส่งข้อมูลแบบอะซิงโครนัสเรียกว่า Universal Asynchronous Receiver/Transmitter หรือ UART อัตราความเร็วในการรับส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสคือ ค่าบอดเรต ซึ่งก็คือค่าจำนวนบิตต่อวินาทีที่ใช้ในการรับส่งข้อมูล บอดเรตมาตรฐานที่ใช้สำหรับพอร์ตอนุกรมRS-232ได้แก่110,150,300,600,1200,2400,4800,9600 และ 19200 บิตต่อวินาที และมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ซึ่งการรับส่งข้อมูลแบบอนุกรมโดยไม่ผ่านโมเด็มอาจกำหนดบอดเรตได้สูงถึง 115200 บิตต่อวินาที

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหดจะใช้ UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ UART เบอร์ 8250 UART ชิพเหล่านี้มีระดับแรงดันเป็นแบบ ทีทีแอล (0 และ +5V) แต่เพื่อให้มีระดับแรงดันเป็นไปตามมาตรฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลมากขึ้นระดับแรงดัน ทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้น โดยลอจิก “0” มีระดับแรงดัน +3V ถึง +12V ในขณะที่โลจิก “1” มีระดับแรงดัน –3V ถึง –12V

มาตรฐานพอร์ตอนุกรม RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งอยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตรฐานนี้ตอนแรกจะใช้คอนเน็กเตอร์ (Connecter) เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับแรงดันตั้งแต่ –3 ถึง –12V แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +12V แสดงว่าเป็นช่องว่าง (Space)

มาตรฐาน RS-232 ได้กำหนดรูปแบบของการเชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) ไว้ว่า อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตรฐาน RS-232

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ ที่อยู่ที่โมเด็มจะเป็นแบบ DCE

สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยการรับส่งข้อมูลได้ที่ความยาวสายสัญญาณสูงสุดถึง 20 เมตร

การสร้างโปรแกรมติดต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ภายนอกเพื่อใช้ควบคุมการทำงาน

โครงงานนี้ได้ใช้โปรแกรม Visual basic6.0 มาสร้างส่วนที่ใช้ควบคุมอุปกรณ์ภายนอกโดยสั่งงานผ่านหน้าจอคอมพิวเตอร์ ซึ่ง Visual Basic สามารถใช้สร้างโปรแกรมบนวินโดวส์โดยอาศัยการออกแบบโปรแกรมในลักษณะ Visualize ซึ่งใช้การกำหนดตำแหน่งของ Object ลงบนจอภาพเพื่อติดต่อกับผู้ใช้โดยตรง Object เหล่านี้จะเปลี่ยนไปตามเหตุการณ์ (Even) ต่างๆที่เกิดขึ้นเช่น การเคลื่อนเมาส์หรือการรับข้อมูลจากคีย์บอร์ด ในการกำหนดการใช้งานผ่าน Evenใดๆจะใช้ภาษา Basic เข้ามาช่วยในการเขียนโปรแกรม ดังนั้นอาจกล่าวได้ว่า การพัฒนาโปรแกรมบนวินโดวส์โดยใช้ Visual Basic มีความง่ายและสะดวกในการใช้งาน รวมทั้งมีขั้นตอนน้อย เพียงแต่เลือก Form และ Control ที่เหมาะสม แล้ววาดลงบนจอภาพเพื่อใช้ติดต่อกับผู้ใช้ จากนั้นจึงทำการเขียนภาษา Basic เพื่อสร้างโปรแกรมด้วยตนเอง ด้วยวิธีที่ง่ายและรวดเร็วกว่าที่คิด จึงทำให้ผู้ใช้เรียนรู้ได้ภายใน 2-3 ชั่วโมงและสามารถสร้างโปรแกรมวินโดวส์เป็นโปรแกรมแรกได้

นอกจากนี้ Visual Basic ยังใช้ได้ตั้งแต่ User ระดับต้น เพื่อสร้างโปรแกรมง่ายๆบนวินโดวส์หรือ Programmerระดับกลางที่จะเรียกใช้ฟังก์ชั่นการทำงานต่างๆของ Visual Basic ได้อย่างมีประสิทธิภาพ ตลอดจน Programmer ระดับอาชีพที่จะพัฒนาโปรแกรมในระดับสูงโดยใช้ Object Linking and Embedding(OLE) และ Windows Applications Programmimg Interface (API) มาประกอบในการเขียนโปรแกรม

ขั้นตอนในการเขียนโปรแกรมของ Visual Basic

ขั้นตอนในการเขียนโปรแกรม ประกอบด้วยขั้นตอนหลัก 2 ขั้นตอนดังนี้

ขั้นตอนที่ 1. สร้างจอภาพของโปรแกรม

ในขั้นตอนนี้จะทำการออกแบบฟอร์ม (Form) เพื่อใช้ในการติดต่อกับผู้ใช้ หรือเรียกว่า "User Interface" ในการพัฒนาโปรแกรมแบบเดิม ขั้นตอนนี้จะใช้เวลาและค่าใช้จ่ายค่อนข้างสูง เนื่องจากจะต้องเขียนโปรแกรมเพื่อสร้างจอภาพต่างๆ จากนั้นต้องคอมไพล์ (Compile) โปรแกรมนั้นแล้วรัน (Run) จึงจะเห็นจอภาพที่จัดทำขึ้น แต่สำหรับ Visual Basic ปัญหาในลักษณะนี้ได้ถูกแก้ไขโดยใช้เทคนิคของVisualize ซึ่งเป็นความสามารถส่วนหนึ่งของ Visual Basic ขั้นตอนนี้สามารถทำได้ง่ายดายเพียงแต่นำเอาคอนโทรล (Control) ต่างๆในทูลบ็อค (Toolbox) ที่ต้องการใช้งานมาวางไว้บน Form ซึ่งทำให้ประหยัดเวลาและทำให้เห็นลักษณะจอภาพที่ออกแบบไว้ขณะนั้นเลย

ขั้นตอนที่ 2. การเขียนโปรแกรม

เมื่อทำการวาง Control ต่างๆลงบน Form เรียบร้อยแล้ว (Control ต่างๆเมื่อถูกนำมาวางไว้บน Form จะเรียกว่า "Object") ขั้นตอนต่อมาก็คือ การเขียนโปรแกรมเพื่อกำหนดการทำงานให้กับแต่ละ Object ภายใต้เหตุการณ์(Even)ต่างๆที่จะเกิดขึ้นกับหน้าจอนั้นๆ

การส่งค่าผ่านพอร์ต RS-232 โดยใช้ MSComm Control

ในการบังคับอุปกรณ์ที่ใช้ควบคุมส่วนต่างๆของ Hard Ware ให้ทำงานนั้นต้องใช้ Function คำสั่ง MSComm ที่มีอยู่ใน Visual Basic มาช่วย

MSComm เป็น Object Controller ที่ใช้ทางด้านการสื่อสารผ่าน Port Series ของคอมพิวเตอร์แอปพลิเคชั่น MSComm ไปใช้นั้นมีอยู่หลายประเภทด้วยกัน เช่น การเขียนโปรแกรมสื่อสารกับคอมพิวเตอร์อื่นๆ หรือศูนย์บริการ BBS หรือแม้กระทั่งโฮสต์ที่ให้บริการทางอินเตอร์เน็ตโดยผ่านโมเด็ม และนอกจากนี้ผู้อ่านยังสามารถใช้คอนโทรลMSComm ในการติดต่อหรือควบคุมบอร์ดต่างๆหรือแม้กระทั่งบาร์โค้ด (Barcode Reader) ที่ต่อผ่านพอร์ตอนุกรมก็ได้เช่นกัน

ด้วยเหตุนี้คอนโทรล MSComm จึงมีประโยชน์เป็นอย่างมากในการสร้างแอพพลิเคชั่นด้านการสื่อสารหรือการควบคุมบอร์ดต่างๆที่มีใช้ตามโรงงานอุตสาหกรรมทั่วไปซึ่งในปัจจุบันมีการนำ Visual Basic มาใช้กันอย่างแพร่หลาย เช่นการใช้ Visual Basic ร่วมกับคอนโทรล MSComm ในการติดต่อกับเครื่องวัดการใช้กำลังไฟฟ้าของหม้อแปลงไฟฟ้าในโรงงานอุตสาหกรรม เป็นต้น

จะเห็นว่าเราสามารถที่จะนำ Visual Basic มาประยุกต์ใช้กับอุตสาหกรรมที่ต้องการควบคุมบอร์ดหรืออุปกรณ์ด้านอิเล็กทรอนิกส์ผ่านทางพอร์ตอนุกรมแบบอัตโนมัติ (Realtime-automatic control) ได้โดยอาศัยเพียงคอนโทรล MSComm เท่านั้น

คอนโทรล MSComm

คอนโทรล MSComm (Communications)เป็นคอนโทรลตัวหนึ่งที่ช่วยในการติดต่อกับพอร์ตอนุกรม(Serial Port) เช่น การติดต่อผ่านโมเด็ม (Modem) หรือการติดต่อโดยตรงกับบอร์ดอิเล็กทรอนิกส์ เป็นต้น ซึ่งคอนโทรล MSComm ที่มากับ Visual Basic จะเป็นคอนโทรลที่ตอบสนองต่อเหตุการแบบ Even-Driven นั่นคือคอนโทรลจะทำหน้าที่ตรวจสอบการเกิดขึ้นหรือการร้องขอให้เกิดเหตุการณ์ต่างๆ กับพอร์ตอนุกรมโดยอัตโนมัติ และจะมีการแจ้งเตือนให้ผู้อ่านได้ทราบโดยผ่านทางโพรซีเยอร์เหตุการณ์เช่นเดียวกับคอนโทรลทั่วๆไปของ Visual Basic นั่นเอง ดังนั้นในการเขียนโค้ดผู้อ่านจึงไม่จำเป็นต้องสร้างโพรซีเยอร์ที่คอยทำหน้าที่ตรวจสอบเหตุการณ์ต่างๆ ของพอร์ตอนุกรมซึ่งจะทำให้ง่ายต่อการใช้งานเป็นอย่างมาก

คอนโทรล MSComm จะมีหน้าที่มาตรฐานหลักๆสำหรับการสื่อสารผ่านพอร์ตอนุกรม 3 ประการดังต่อไปนี้

-หมุนหมายเลขติดต่อกับโทรศัพท์ปลายทางที่กำหนด

-ตรวจสอบการเข้ามาของข้อมูลยังพอร์ตอนุกรมโดยอัตโนมัติ

-ส่งข้อมูลตามที่กำหนดจากโปรแกรมไปยังพอร์ตอนุกรม

ในความเป็นจริง คอนโทรล MSComm ไม่ได้ทำหน้าที่ติดต่อกับพอร์ตอนุกรมโดยตรง แต่มันจะทำหน้าที่เรียกฟังก์ชั่นวินโดวส์ API ซึ่งวินโดวส์จะทำการส่งหรือรับข้อมูลผ่านทางพอร์ตอนุกรมโดยอาศัยไดรเวอร์ Comm.drv อีกทอดหนึ่ง ดังนั้นจึงสามารถสรุปสั้นๆได้ว่าทุกครั้งที่ผู้อ่านมีการเรียกใช้ MSComm ก็หมายถึงการเรียกใช้วินโดวส์ API ซึ่งจะถูกตีความหมายอีกทอดหนึ่งโดยไดรเวอร์ Comm.drv จากนั้นก็จะส่งผ่านข้อมูลที่ถูกจัดรูปแบบตามมาตรฐานการสื่อสาร(ทั้งนี้ขึ้นอยู่กับอุปกรณ์ที่ต่อเข้ากับพอร์ตอนุกรม) ให้กับดีไวซ์ไดรเวอร์อีกทอดหนึ่งนั่นเอง

การกำหนดคุณสมบัติของคอนโทรล MSComm ในขณะออกแบบ ผู้อ่าสามารถกระทำได้อย่างสะดวกโดยการคลิกที่ปุ่มของรายการ (Custom) ในหน้าต่างคุณสมบัติ ซึ่งก็จะปรากฏไดอะล็อกบ็อกซ์ Property Page เพื่อให้ผู้อ่านปรับแต่งค่าคุณสมบัติของคอนโทรล MSComm สนับสนุนดังรูป ปุ่มคำสั่งของไดอะล็อกบ็อกซ์ Poperty Page มีความหมายดังนี้

-ปุ่มคำสั่ง OK ยอมรับการแก้ไขคุณสมบัติของคอนโทรล MSComm

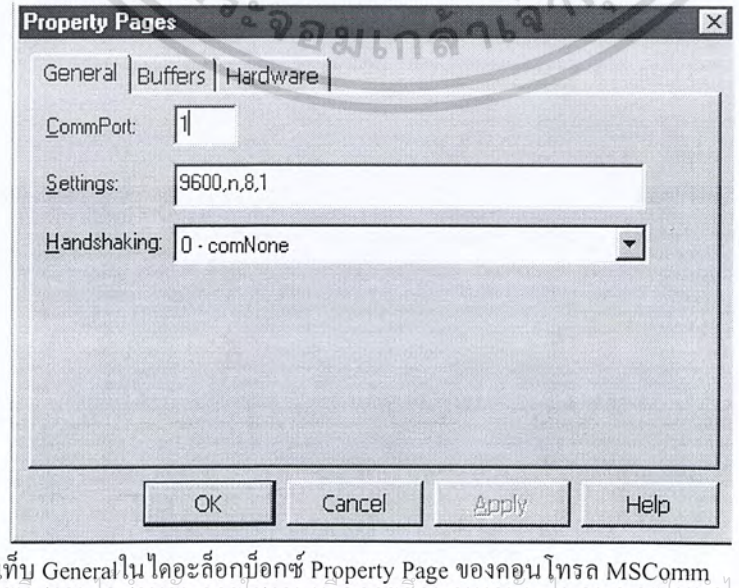
-ปุ่มคำสั่ง Cancle ยกเลิกการแก้ไขคุณสมบัติของคอนโทรล MSCOmm

-ปุ่มคำสั่ง Apply อัปเดตคุณสมบัติที่ถูกแก้ไขของคอนโทรล MSComm

-ปุ่มคำสั่ง Help แสดงผล Help ของคอนโทรล MSComm

-CommPort หมายเลขของพอร์ตอนุกรม

-Setting พารามิเตอร์สำหรับการสื่อสาร เช่น Baud Rate หรือ Parity bit เป็นต้น



รูปที่10. แสดงแท็บ Generalในไดอะล็อกบ็อกซ์ Property Page ของคอนโทรล MSComm

-Handshaking ตรวจสอบการตอบรับการสื่อสาร(Handshaking)แท็บบัฟเฟอร์ (Buffer) การกำหนดคุณสมบัติที่เกี่ยวกับบัฟเฟอร์ข้อมูล

-InBufferSize ขนาดของบัฟเฟอร์สำหรับด้านรับข้อมูลเข้า

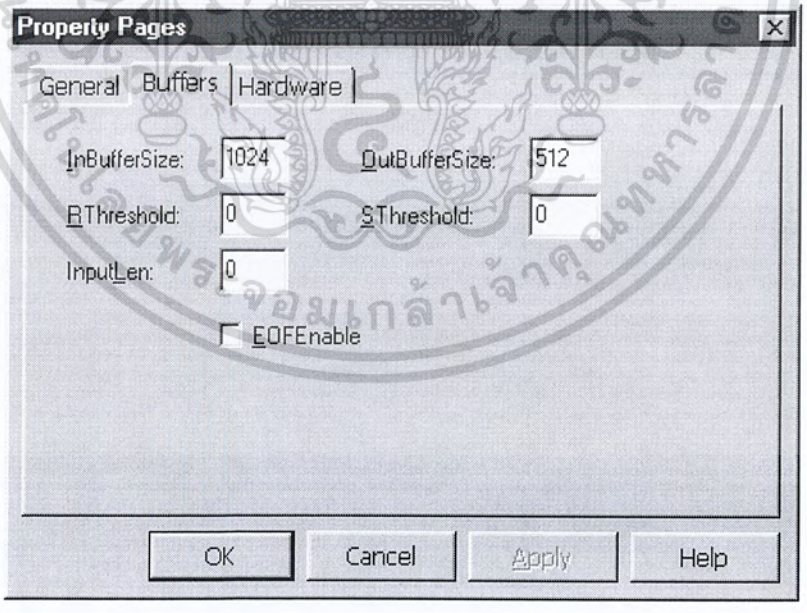
-Rthreshold จำนวนตัวอักษรที่จะรับเข้า ก่อนที่คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEven มีค่าเท่ากับ ComEvReceive และมีการเรียกเหตุการณ์ OnComm

-InputLen จำนวนตัวอักษรที่คุณสมบัติ Input จะอ่านข้อมูลจากบัฟเฟอร์ด้านรับเข้า

-OutBufferSize ขนาดของบัฟเฟอร์ด้านส่งออกข้อมูล

-Sthreshold จำนวนตัวอักษรที่น้อยที่สุดที่ถูกจัดเก็บในบัฟเฟอร์ด้านส่งออกก่อนที่คอนโทรล MSCOmm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเท่ากับ ComEvSend และมีการเรียกเหตุการณ์ OnComm

-EOFEnable กำหนดให้คอนโทรล MSComm มีการตรวจสอบตัวอักษรรหัสจุดสิ้นสุดของไฟล์ (EOF)ในระหว่างการรับเข้าข้อมูล



รูปที่11.แสดงแท็บ Buffer ในไดอะล็อกบ็อกซ์ Property Page ของคอนโทรล MSComm

-PairtyReplace กำหนดตัวอักษรสำหรับแทนที่ตัวอักษรที่ไม่เป็นจริง ในขณะที่เกิดข้อผิดพลาดของParity Errer

-NullDiscard ตรวจสอบการส่งตัวอักษรนูล (Null) จากพอร์ตอนุกรมไปยังบัฟเฟอร์ด้านรับเข้า

-RTSEnable มีการใช้งานสาย Request To Send (RTS)

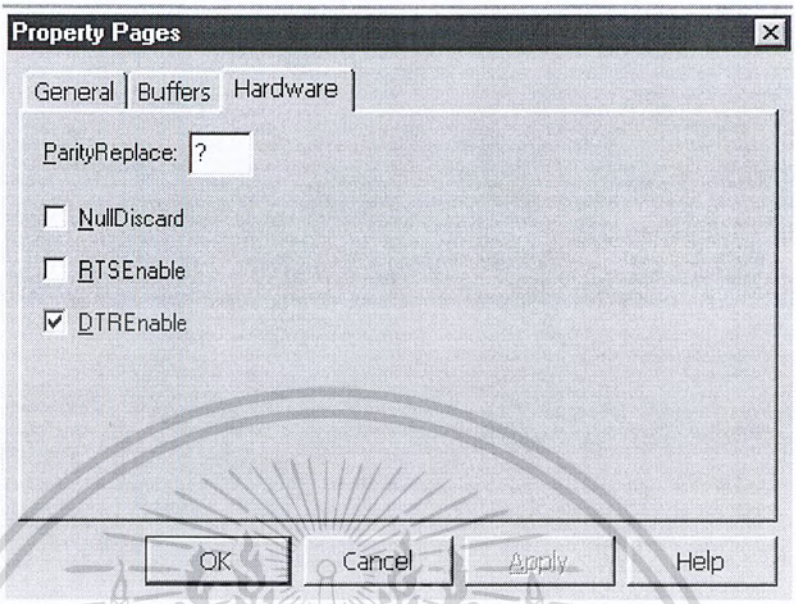
-DTREnable มีการใช้งานสาย Data Terminal Ready (DTR)

สำหรับรายละเอียดทั้งหมดของคุณสมบัติที่สามารถกำหนดในไดอะล็อกบ็อกซ์ Property Page ของคอนโทรล MSComm ตามที่กล่าวมาแล้วในข้างต้น ผู้อ่านสามารถที่จะดูรายละเอียดได้จากแต่ละคุณสมบัติดังที่จะกล่าวต่อไป

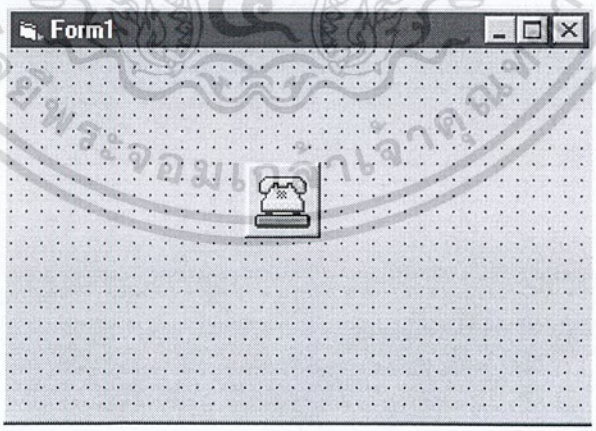
สำหรับฟอร์มหนึ่งๆผู้อ่านสามารถเพิ่มได้หลายๆ คอนโทรล MSComm ทั้งนี้ขึ้นอยู่กับความต้องการของผู้ใช้ว่าต้องการติดต่อกับพอร์ตอนุกรมใดบ้าง สำหรับวินโดวส์ 95 และ NT 4.0 ผู้ใช้สามารถติดตั้งพอร์ตอนุกรมได้มากกว่า 4 พอร์ต โดยเมื่อผู้ใช้เพิ่มคอนโทรล MSComm ลงในฟอร์มก็จะปรากฏดังรูปที่ 4 ซึ่งจะสนับสนุนเหตุการ์ณและโพรซีเยอร์เหตุการณ์ดังต่อไปนี้

คุณสมบัติ

Break	CDholding	CommEven	CommID
CommPort	CTSHolding	DSRHolding	DTREnable
EOFEnable	Handshaking	InBufferCount	InBufferSize
Index	Input	InputLen	InputMode
Name	NollDiscard	Object	OutBufferCount
OutBuffersize	Output	Parent	ParityReplace
PortOpen	TRHeshold	RTSEnable	Setting
Stheshold	Tag		



รูปที่12. แสดงแท็บ Hard Ware ในไดอะล็อกบ็อกซ์ Property Page ของคอนโทรล MSComm



รูปที่13.แสดงคอนโทรล MSComm ที่ใช้ในการออกแบบ

Break

กำหนดหรือยกเลิกสัญญาณการหยุด (Break Signal) ซึ่งผู้อ่านสามารถกำหนดได้เฉพาะในขณะทำงานเท่านั้น

รูปแบบการใช้งาน

object.Break [=boolean]

boolean หมายถึง ข้อมูลชนิดบูลีน (Booolean) ที่กำหนดสถานะของสัญญาณการหยุด ดังต่อไปนี้

True หมายถึง กำหนดสถานะของสัญญาณการหยุด

False หมายถึง ยกเลิกสถานะของสัญญาณการหยุด

การกำหนดสถานะของสัญญาณการหยุด หมายถึง การกำหนดให้สสัญญาณ Transmission อยู่ในสถานะของการหยุด ซึ่งสัญญาณการหยุดจะทำหน้าที่หยุดการรับสัญญาณชั่วคราว จนกว่าสถานะดังกล่าวจะถูกยกเลิก โดยการกำหนดให้คุณสมบัติ Break มีค่าเท่ากับ False อีกครั้ง ซึ่งโดยปกติเราจะกำหนดสถานะของสัญญาณการหยุด ก็ต่อเมื่ออุปกรณ์ที่เราติดต่อด้วยนั้นต้องการให้มีการกำหนดสถานะของสัญญาณหยุดเท่านั้น

CDHolding

ตรวจสอบสัญญาณพาหะ (Carrier) โดยการค้นหาจากสถานะของสายสัญญาณ Carrier Detect(CD) ซึ่งผู้อ่านสามารถกำหนดได้เฉพาะในขณะทำงานเท่านั้น

รูปแบบการใช้งาน

object.CDHolding

สัญญาณพาหะจะถูกส่งจากโมเด็มมายังพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ เพื่อแสดงสถานะความพร้อมที่จะติดต่อสื่อสาร (Online) ของโมเด็ม ซึ่งค่าที่ได้จากคุณสมบัติ CDHolding จะเป็นค่าบูลีนดังต่อไปนี้

True หมายถึง สายสัญญาณ Carrier Detect อยู่ในสถานะ High (ไม่พร้อมที่จะติดต่อสื่อสารในขณะนั้น)

False หมายถึง สายสัญญาณ Carrier Detect อยู่ในสถานะ Low (พร้อมที่จะติดต่อสื่อสารในขณะนั้น)

ถ้าหากสายสัญญาณ Carrier Detect (ในบางครั้งคราววงการสื่อสารด้วยโมเด็มมักจะเรียว่า ReceiveLine Signal Detect : RLSD ก็ได้) อยู่ในสถานะ High ก็แสดงว่าเกิดสถานะ Time-Out ในขณะนั้น ซึ่งในกรณีนี้คอนโทรล MSComm ก็จะกำหนดคุณสมบัติ CommEvent มี

ค่าเท่ากับ CommEventCDTO และเรียกโพรซีเยอร์เหตุการณ์ OnComm ทันทีเพื่อแสดงให้เห็นถึงการเกิดข้อผิดพลาด

หากผู้อ่านเขียนโปรแกรมที่ติดต่อกับโฮสต์ต่างๆ เช่น BBS หรืออินเตอร์เน็ต เป็นต้น ผู้อ่านควรที่จะเขียนโค้ดเพื่อตรวจสอบสถานะของสัญญาณเป็นช่วงๆ ทั้งนี้เพราะโฮสต์ปลายทางอาจมีการยกเลิกการสื่อสารโดยการวางสายของโมเด็มได้ตลอดเวลา เพราะโดยปกติโฮสต์ปลายทางเหล่านี้จะมีการติดต่อสื่อสารกับผู้ใช้จำนวนมากในคราวเดียวกัน

CommEvent

รายงานเหตุการทุกครั้งที่เกิดข้อผิดพลาดหรือมีการสื่อสาร ซึ่งผู้อ่านสามารถอ่านค่าได้เฉพาะขณะทำงานเท่านั้น

รูปแบบการใช้งาน

object.CommEvent

คอนโทรล MSComm จะมีการเรียกโพรซีเยอร์เหตุการณ์ OnComm ทุกครั้งที่มีข้อผิดพลาดหรือมีการสื่อสารเกิดขึ้น ซึ่งค่าตัวเลขที่เป็นจำนวนเต็ม แสดงถึงข้อผิดพลาดหรือเหตุการณ์ที่มีการติดต่อสื่อสารดังกล่าว ก็จะถูกจัดเก็บไว้ในคุณสมบัติCommEvent เสมอ ดังนั้นหากผู้อ่านต้องการตรวจสอบข้อผิดพลาดหรือเหตุการณ์ที่มีการติดต่อสื่อสารภายในโพรซีเยอร์ OnComm ก็ควรจะใช้ค่าตัวเลขจากคุณสมบัติ CommEvent ในการตรวจสอบเสมอ ซึ่งค่าตัวเลขที่รายงานในคุณสมบัติ CommEvent มีดังต่อไปนี้

ค่าคงที่	ค่าตัวเลข	รายละเอียด
CommEventBreak	1001	ได้รับสัญญาณการหยุด (BreakSignal)
CommEventCTSTO	1002	สายสัญญาณ ClearToSend อยู่ในสถานะ Low (timeout) ในขณะที่พยายามจะส่งออกตัวอักษร
CommEventDSRTO	1003	สายสัญญาณ DataSetReady อยู่ในสถานะLow(timeout) ในขณะที่พยายามจะส่งออกตัวอักษร
CommEventFrame	1004	เฟรมของข้อมูลไม่ถูกต้องซึ่งถูกตรวจพบโดยฮาร์ดแวร์
CommEventOverun	1006	เกิด Port Overun หมายถึงมีการรับตัวอักษรใหม่เข้ามาในขณะที่ตัวอักษรก่อนหน้านี้ยังไม่ถูกอ่านจากฮาร์ดแวร์
CommEvenCDTO	1007	สายสัญญาณ Carrier Detect อยู่ในสถานะ Low(timeout) ในขณะที่พยายามที่จะส่งออกตัวอักษร

CommEventRxOver 1008 Reciever Buffer Overflow หมายถึงขนาดของบัฟเฟอร์ด้านรับเข้าข้อมูล (reciever buffer) ไม่เพียงพอกับขนาดของพอร์ตที่รับเข้ามา

CommEventRxParity 1009 Parity Errer ซึ่งถูกตรวจพบโดยฮาร์ดแวร์

CommEventTxFull 1010 Transmit Buffer Full หมายถึงบัฟเฟอร์ด้านส่งออกข้อมูล (Transmit Buffer) เต็ม ในขณะที่พยายามเก็บข้อมูลใหม่ลงในบัฟเฟอร์

CommEventDCB 1011 Unexpected errer หมายถึง เกิดข้อผิดพลาดที่ไม่ได้ถูกนิยามเอาไว้ขณะอ่าน Device Control Block (DCB) จากพอร์ตอนุกรม

เหตุการณ์ (Event)

ค่าคงที่	ตัวเลข	รายละเอียด
CmEvCTS	1	มีจำนวนตัวอักษรด้านส่งข้อมูลออก น้อยกว่าตัวอักษรที่กำหนดในคุณสมบัติ Sthreshold
ComEvRecieve	2	การรับเข้าจำนวนตัวอักษรที่กำหนดในคุณสมบัติ Rthreshold ซึ่งจะเกิดขึ้นอย่างต่อเนื่องจนกว่าผู้อ่านจะใช้คุณสมบัติ Input ในการอ่านข้อมูลจากบัฟเฟอร์สำหรับรับเข้าข้อมูล
ComEvCTS	3	มีการเปลี่ยนแปลงสถานะของสายสัญญาณClearToSend
ComEvDSR	4	มีการเปลี่ยนแปลงสถานะของสายสัญญาณDataSetReady ซึ่งเหตุการณ์นี้จะเกิดขึ้นเมื่อมีการเปลี่ยนแปลงสถานะของสายสัญญาณ DataSetReady จาก 1 เป็น 0 เท่านั้น
ComEvCD	5	มีการเปลี่ยนสถานะของสายสัญญาณ Carrier Detect
ComEvRing	6	มีการตรวจพบการเรียกหมายเลข (สัญญาณเสียงกริ่ง) ซึ่ง UART บางตัวจะไม่สนับสนุนคุณสมบัตินี้
ComEvEOF	7	มีการรับตัวอักษรรหัสสิ้นสุดของไฟล์

CommID

รายงานหมายเลข handle ของดีไวซ์ (Device) ด้านการสื่อสารที่ได้จากคุณสมบัติ CommID จะเป็นค่าจำนวนเต็ม long สำหรับนำไปใช้งานร่วมกับวินโดวส์ API อื่นๆ

CommPort

รายงานหรือกำหนดหมายเลขของพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ที่ต้องการติดต่อ

รูปแบบการใช้งาน

object.CommPort[=value] value หมายถึง ข้อมูลชนิดจำนวนเต็ม ที่กำหนดหมายเลขของพอร์ตอนุกรม

สำหรับหมายเลขของพอร์ตอนุกรมสามารถมีค่าได้ตั้งแต่ 1 ถึง 16 (ค่าปกติจะเท่ากับ 1) ซึ่งก่อนที่ผู้อ่านจะเปิดพอร์ตด้วยคุณสมบัติ PortOpen ผู้อ่านต้องกำหนดหมายเลขของพอร์ตอนุกรมให้กับคุณสมบัติ CmmPort เสียก่อน โดยถ้าหากหมายเลขของพอร์ตอนุกรมที่กำหนดให้คุณสมบัติ CommPort ไม่เป็นความจริง ก็จะเกิดข้อผิดพลาด 68 (Device Unavailable) ทันที ซึ่งในกรณีนี้ผู้อ่านสามารถแก้ไขได้โดยการกำหนดหมายเลขของพอร์ตอนุกรมที่ถูกต้องเสียใหม่แล้วจึงทำการเปิดพอร์ตอนุกรมอีกครั้งด้วยคุณสมบัติ PortOpen

CTSHolding

ตรวจสอบสถานะของสายสัญญาณ ClearToSend(CTS) ก่อนที่จะส่งข้อมูลไปยังบัฟเฟอร์ของโมเด็ม ซึ่งผู้อ่านสามารถอ่านค่าได้เฉพาะในขณะทำงานเท่านั้น

รูปแบบการใช้งาน

object.CTSHolding

ซึ่งค่าที่ได้จากคุณสมบัติ CTSHolding จะเป็นค่าบูลีนดังนี้

True หมายถึง สายสัญญาณ ClearToSend อยู่ในสถานะ High

False หมายถึง สายสัญญาณ ClearToSend อยู่ในสถานะ Low

โดยปกติClearToSend จะถูกส่งจากโมเด็มมายังพอร์ตอนุกรมของคอมพิวเตอร์ เพื่อแสดงให้เห็นถึงสถานะความพร้อมในการส่งข้อมูลมายังโมเด็ม ซึ่งถ้าหากสายสัญญาณ ClearToSend อยู่ในสถานะLow ก็แสดงว่าเกิดสภาวะ Time Out ในขณะนั้น ซึ่งในกรณีนี้คอนโทรลMSComm ก็จะกำหนดให้คุณสมบัติ CommEvent มีค่าเท่ากับ comEventCTSTO และเรียกโพรซีเยอร์เหตุการณ์ OnComm ทันทีเพื่อแสดงให้เห็นการเกิดข้อผิดพลาด

สายสัญญาณ ClearToSend จะถูกใช้ในการติดต่อสื่อสารของฮาร์ดแวร์ (Hardware handshaking) โดยการใช้วิธี RTS/CTS (Request To Send/Clear To Send)

DSRHolding

ตรวจสอบสถานะของสายสัญญาณ Data Set Ready (DSR) ซึ่งผู้อ่านสามารถอ่านค่าได้เฉพาะในขณะทำงานเท่านั้น

รูปแบบการใช้งาน

object.DSRHolding

ซึ่งค่าที่ได้จากคุณสมบัติ DSRHolding จะเป็นค่าบูลีนดังต่อไปนี้

True หมายถึง สายสัญญาณ DataSetReady อยู่ในสถานะ high

False หมายถึง สายสัญญาณ DataSetReady อยู่ในสถานะ low

โดยปกติสัญญาณ DataSetReady จะถูกส่งจากโมเด็มมายังพอร์ตอนุกรมของคอมพิวเตอร์ เพื่อแสดงให้เห็นถึงสถานะความพร้อมในการทำงานของโมเด็ม ซึ่งถ้าหากสายสัญญาณ DataSetReady อยู่ในสถานะ high ก็แสดงว่าเกิดสถานะ time out ในขณะนั้น ซึ่งในกรณีนี้คอนโทรล MSComm ก็จะกำหนดให้คุณสมบัติ CommEven มีค่าเท่ากับ CommEvenDSRTO และเรียกโพรซีเยอร์เหตุการณ์ OnComm ทันทีเพื่อแสดงให้เห็นถึงการเกิดข้อผิดพลาด

โดยปกติคุณสมบัติ DSRHolding จะถูกใช้ในการเขียนโพรซีเยอร์สำหรับติดต่อสื่อสาร (handshaking) ด้วยวิธี Data Set Ready/Data Terminal Ready สำหรับติดต่อกับเครื่องจักรหรืออุปกรณ์ประเภท Data Terminal Equipment (DTF)

DTREnable

ตรวจสอบหรือกำหนดสถานะของสายสัญญาณ Data Terminal Ready (DTR) ในระหว่างที่มีการสื่อสารข้อมูล ซึ่งผู้อ่านสามารถอ่านค่าได้ขณะทำงานเท่านั้น

รูปแบบการใช้งาน

object.DTREnable[=boolean]

boolean หมายถึง ข้อมูลชนิดบูลีนที่บอกถึงสถานะสายสัญญาณ DataTerminalReady ดังนี้

True หมายถึง ยอมให้มีการใช้สายสัญญาณ Data Terminal Ready

False หมายถึง ยกเลิกการใช้สายสัญญาณ Data Terminal Ready (default)

โดยปกติสัญญาณ Data Terminal Ready จะถูกส่งจากคอมพิวเตอร์มายังโมเด็ม เพื่อแสดงให้เห็นถึงสถานะความพร้อมในการรับข้อมูลจากโมเด็ม โดยถ้าหาก DTR ถูกกำหนดให้เท่ากับ True สายสัญญาณ Data Terminal Ready จะอยู่ในสถานะ high(on) เมื่อพอร์ตอนุกรมเปิด และ (off) เมื่อพอร์ตอนุกรมถูกปิด แต่ถ้าหาก DTR ถูกกำหนดให้เท่ากับ False สายสัญญาณ DTR จะอยู่ในสถานะ low (off) ตลอดเวลา ในการสื่อสารผ่านโมเด็มโดยทั่วไปสัญญาณ DTR จะอยู่ในสถานะ low จะหมายถึงการวางหูโทรศัพท์หรือสิ้นสุดการสื่อสารนั่นเอง

EOFEnable

กำหนดให้คอนโทรล MSComm มีการตรวจสอบตัวอักษรรหัสจุดสิ้นสุดของไฟล์ (EOF)ในระหว่างการรับเข้าของข้อมูล โดยที่การรับเข้าข้อมูลจะสิ้นสุดที่การพบตัวอักษร EOF

รูปแบบการใช้งาน

object.EOFEnable[=boolean]

boolean หมายถึง ข้อมูลชนิดบูลีนที่บอกถึงสถานะการตรวจสอบตัวอักษร EOF ดังต่อไปนี้

True หมายถึง เหตุการณ์ OnComm จะถูกเรียกทันทีที่พบตัวอักษร EOF

False หมายถึง เหตุการณ์ OnCOmm จะไม่ถูกเรียกใช้เมื่อพบตัวอักษร EOF (default)

โดยถ้าผู้อ่านกำหนดให้คุณสมบัติ EOFEnable มีค่าเท่ากับ True เมื่อเหตุการณ์ OnComm จะถูกเรียกทันทีที่พบตัวอักษร EOF คุณสมบัติ CommEven ก็จะถูกกไหนดให้มีค่าเท่ากับ CommEvenEOFทันที

Handshaking

รายงานหรือกำหนดการใช้โปรโตคอลการตอบรับการติดต่อสื่อสารฮาร์ดแวร์ (hardware handshaking protocol)

รูปแบบการใช้งาน

object.Handshaking[=value]

value หมายถึง ข้อมูลชนิดเลขจำนวนเต็มที่กำหนดโปรโตคอลตอบรับการติดต่อสื่อสาร ดังต่อไปนี้

ค่าคงที่	ค่าตัวเลข	รายละเอียด
comNone	0	ไม่มีการตอบรับการติดต่อสื่อสาร (default)
comXOnXOff	1	การตอบรับการติดต่อสื่อสารแบบ XON/XOFF
comRTS	2	การตอบรับการติดต่อสื่อสารแบบ RTS/CTS (Request To Sens/Clear To Send)
comTRSXOnXOff	3	การตอบรับการติดต่อสื่อสารทั้งแบบ Request To Send และ XonXOff

ในด้านการสื่อสารด้วยโมเด็ม handshaking หมายถึง โปรโตคอลการสื่อสารภายในที่ซอฟต์แวร์ใช้ในส่งข้อมูลจากพอร์ตอนุกรมของคอมพิวเตอร์ไปยังบัฟเฟอร์ด้านรับเข้าข้อมูลโดยทุกครั้งที่มีการรับเข้าข้อมูลมายังพอร์ตอนุกรมของคอมพิวเตอร์ มันก็จะถูกจัดส่งมายังบัฟเฟอร์ด้านรับข้อมูลเข้าทันที ทั้งนี้เพื่อให้ซอฟต์แวร์สามารถอ่านข้อมูลดังกล่าวได้ต่อไป ซึ่งโปรโตคอลการสื่อสารภายในดังกล่าวจะเป็นการช่วยในการตรวจสอบเพื่อป้องกันการเสียหายของข้อมูล เมื่อเกิดปัญหาขนาดของบัฟเฟอร์ไม่เพียงพอกับจำนวนข้อมูลที่ส่งเข้ามา (buffer overun)

InBufferCount

รายงานจำนวนของตัวอักษรที่รออยู่ในบัฟเฟอร์ด้านรับเข้า ซึ่งผู้อ่านสามารถอ่านค่าได้ในขณะรันแอปพลิเคชั่นเท่านั้น

รูปแบบการใช้งาน

object.InBufferCount[=value]

value หมายถึง ข้อมูลชนิดเลขจำนวนเต็ม ที่กำหนดจำนวนตัวอักษรที่รออยู่ในบัฟเฟอร์ด้านรับข้อมูลเข้าซึ่งจะเป็นจำนวนตัวอักษรที่ซอฟต์แวร์จะสามารถอ่านค่าได้

ถ้าหากมีการลบตัวอักษรทั้งหมดที่รออยู่ในบัฟเฟอร์ด้านรับเข้าทิ้ง ค่าคุณสมบัติของ InBufferCount ก็จะเท่ากับ 0

InBufferSize

รายงานหรือกำหนดขนาดของบัฟเฟอร์ด้านรับเข้า ซึ่งมีหน่วยเป็นไบต์ (โดยปกติ 1 ไบต์จะเท่ากับ 1 ตัวอักษร)

รูปแบบการใช้งาน

object.InBufferSize[=value]

value หมายถึง ข้อมูลชนิดจำนวนเต็มที่กำหนดขนาดของบัฟเฟอร์ด้านรับเข้า ซึ่งปกติซอฟต์แวร์จะกำหนดให้มีค่าเท่ากับ 1,024 ไบต์ (1 KB)

ในการเลือกขนาดของบัฟเฟอร์ด้านรับเข้าให้เหมาะสมนั้น ในทางปฏิบัติเป็นสิ่งที่ยากมาก ทั้งนี้ก็ขึ้นอยู่กับความหนาแน่นของข้อมูลที่มีการส่งไป-มา ในแต่ละครั้งและความเร็วของการติดต่อสื่อสาร(transmision rate) ของโมเด็ม ซึ่งโดยปกติโปรแกรมเมอร์โดยทั่วจะกำหนดให้มีค่าเท่ากับ 1,024 ไบต์ (1 KB) โดยหากเกิดข้อผิดพลาด overflow ในขณะรันแอปพลิเคชั่น ก็ให้ทำการเพิ่มขนาดของบัฟเฟอร์ด้านรับเข้าต่อไป

Input

รายงานพร้อมทั้งทำการลบข้อมูลในบัฟเฟอร์ด้านรับเข้าทิ้ง ซึ่งผู้อ่านสามารถอ่านค่าได้ขณะทำงานเท่านั้น

รูปแบบการใช้งาน

object.Input

ทุกครั้งที่มีการใช้คุณสมบัติ Input ในการอ่านข้อมูลจากบัฟเฟอร์ด้านรับเข้า จำนวนของตัวอักษรที่อ่านได้จะถูกกำหนดลงในคุณสมบัติ Input Len ทันที ซึ่งถ้าหากผู้อ่านกำหนดให้คุณสมบัติ Input Len เท่ากับ 0 ก็จะหมายถึงการกำหนดคุณสมบัติให้ Input อ่านข้อมูลทั้งหมดจากข้อมูลด้านรับเข้านั่นเอง ซึ่งข้อมูลที่อ่านจากคุณสมบัติ Input จะเป็นข้อมูลแบบข้อความหรือไบนารีนั้นขึ้นอยู่กับการกำหนดค่าคุณสมบัติของ Input Mode ดังจะได้กล่าวรายละเอียดต่อไป

InputLen

รายงานหรือกำหนดจำนวนตัวอักษรที่อ่านโดยคุณสมบัติ Input จากบัฟเฟอร์ด้านรับเข้าโดยมีหน่วยเป็น ไบต์

รูปแบบการใช้งาน

object.InputLen[=value]

value หมายถึง ข้อมูลชนิดเลขจำนวนเต็ม ที่กำหนดจำนวนของตัวอักษรที่กำหนดโดยคุณสมบัติ Input

โดยปกติผู้เขียนมักจะกำหนดให้คุณสมบัติ Input มีค่าเท่ากับ 0 เพื่อให้คุณสมบัติ Input มีการอ่านตัวอักษรทั้งหมดจากบัฟเฟอร์ด้านรับเข้า แต่หากผู้เขียนติดต่อกับบอร์ดควบคุมต่างๆ (single control board) ก็จะกำหนดค่าคุณสมบัติ InputLen ให้มีค่าเท่ากับขนาดของเฟรมข้อมูลที่มีการรับส่งในแต่ละครั้ง ซึ่งจะมีค่าคงที่

InputMode

รายงานหรือกำหนดชนิดของข้อมูลที่จะถูกอ่านโดยคุณสมบัติ Input จากบัฟเฟอร์ด้านรับเข้าข้อมูล

รูปแบบการใช้งาน

object.InputMode[=value]

value หมายถึง ข้อมูลชนิดตัวเลขจำนวนเต็ม ที่กำหนดจำนวนชนิดของข้อมูลที่อ่านโดยคุณสมบัติ input ดังนี้

ค่าคงที่	ค่าตัวเลข	รายละเอียด
comInputModeText	0	ข้อมูลชนิดข้อความทั่วไป (default)
comInputModeBinary	1	ข้อมูลชนิดไบนารี

การกำหนดชนิดของข้อมูลที่จะถูกอ่านโดยคุณสมบัติ Input ก็ขึ้นอยู่กับลักษณะงานที่ผู้อ่านกำลังควบคุม โดยปกติถ้าหากข้อมูลชนิดตัวอักษร ANSI ทั่วไป ก็จะกำหนดให้เป็นข้อมูลชนิดข้อความ (comInputModeText) แต่ถ้าหากข้อมูลประกอบด้วยตัวอักษรควบคุม (แอสกีตั้งแต่ 0 ถึง 31) ก็จะกำหนดให้เป็นข้อมูลชนิดไบนารี (comInputModeBinary)

NullDiscard

รายงานหรือกำหนดตรวจสอบการส่งตัวอักษร Null (แอสกี 0) จากพอร์ตอนุกรมไปยังบัฟเฟอร์ด้านรับเข้า

รูปแบบการใช้งาน

object.NullDiscard[=boolean]

True หมายถึง ไม่มีการส่งตัวอักษร Null จากพอร์ตอนุกรมไปยังบัฟเฟอร์ด้านรับเข้า

False หมายถึง มีการส่งตัวอักษร Null จากพอร์ตอนุกรมไปยังบัฟเฟอร์ด้านรับเข้า (default)

OutBufferCount

รายงานจำนวนตัวอักษรที่รออยู่ในบัฟเฟอร์ด้านส่งออก ซึ่งผู้อ่านสามารถอ่านค่าได้เฉพาะขณะรันแอปพลิเคชั่นเท่านั้น

รูปแบบการใช้งาน

object.OutBufferCount[=value]

value หมายถึง ข้อมูลชนิดเลขจำนวนเต็ม ที่กำหนดจำนวนตัวอักษรที่รออยู่ในบัฟเฟอร์ด้านส่งออก

ถ้าหากมีการลบตัวอักษรทั้งหมดที่รออยู่ในบัฟเฟอร์ด้านส่งออกทิ้ง ค่าคุณสมบัติของ OutBufferCount จะมีค่าเท่ากับ 0

OutBufferSize

รายงานหรือกำหนดขนาดของบัฟเฟอร์ด้านส่งออก ซึ่งมีหน่วยเป็นไบต์(โดยปกติ 1 ไบต์จะมีค่าเท่ากับ 1 ตัวอักษร)

รูปแบบการใช้งาน

object.OutBufferSize[=value]

value หมายถึง ข้อมูลชนิดเลขจำนวนเต็ม ที่กำหนดขนาดของบัฟเฟอร์ด้านส่งออก ซึ่งปกติซอฟต์แวร์จะกำหนดให้มีค่าเท่ากับ 512 ไบต์

ในการเลือกขนาดของบัฟเฟอร์ด้านส่งออกที่เหมาะสมนั้น ในทางปฏิบัติเป็นสิ่งที่ยากมากเช่นกัน ทั้งนี้ก็ขึ้นอยู่กับความหนาแน่นของข้อมูลที่มีการส่งไป-มา ในแต่ละครั้ง และความเร็วของการสื่อสารของโมเด็ม ซึ่งปกติโปรแกรมเมอร์โดยทั่วไปจะกำหนดให้มีค่าเท่ากับ 512 ไบต์ โดยหากเกิดข้อผิดพลาด overflow ในขณะรันแอปพลิเคชั่น ก็ให้เพิ่มขนาดของบัฟเฟอร์ด้านรับเข้าต่อไป

Output

ทำการส่งข้อมูลไปยังบัฟเฟอร์ด้านส่งออก ซึ่งผู้อ่านสามารถกำหนดค่าได้ขณะทำงานเท่านั้น

รูปแบบการใช้งาน

object.Output[=value]

value หมายถึง ข้อมูลชนิดสตริงหรือ variant ที่ต้องการส่งไปยังบัฟเฟอร์ด้านส่งออก

สำหรับชนิดของข้อมูลที่ถูกส่งโดยคุณสมบัติ Output จะเป็นข้อมูลแบบข้อความหรือไบนารีก็ได้ขึ้นอยู่กับการกำหนดค่าคุณสมบัติ InputMode ดังที่กล่าวมาแล้วข้างต้น

ParityReplace

กำหนดตัวอักษรสำหรับแทนที่ตัวอักษรที่ไม่เป็นจริงในขณะที่เกิดข้อผิดพลาด parity error

รูปแบบการใช้งาน

object.ParityReplace[=value]

value หมายถึง ข้อมูลชนิดสตริงที่กำหนดตัวอักษรใดๆ

โดยปกติผู้เขียนมักจะกำหนดให้คุณสมบัติ ParityReplace มีค่าเท่ากับตัวอักษร หรือในบางกรณีจะให้เท่ากับสตริงว่าง("") เพื่อไม่ให้มีการแทนที่ตัวอักษรที่ไม่เป็นจริง ในขณะที่เกิดข้อผิดพลาด parity error ซึ่งในกรณีนี้คุณสมบัติ CommEven ก็จะถูกกำหนดให้มีค่าเท่ากับ commEvenRXParity ทันที

ในการสื่อสารแบบใช้ parity bit นั้นเป็นการกำหนดให้มีการเพิ่มบิต (0 หรือ 1) ลงในข้อมูลที่มีการส่งไป-มาในแต่ละครั้ง ทั้งนี้เพื่อใช้บิตดังกล่าวในการตรวจสอบความถูกต้องของข้อมูลโดยการตรวจสอบผลบวกของบิตทั้งหมดว่ามีค่าเท่ากับเลขคี่ (1) หรือคู่(0)

PortOpen

กำหนดสถานะการเปิด (open) หรือ ปิด(close) ของพอร์ตอนุกรมของเครื่องคอมพิวเตอร์

รูปแบบการใช้งาน

object.PortOpen[=boolean]

boolean หมายถึง ข้อมูลชนิดบูลีนที่กำหนดสถานะของพอร์ตอนุกรม ดังต่อไปนี้

True หมายถึง พอร์ตอนุกรมถูกเปิด

False หมายถึง พอร์ตอนุกรมถูกปิด (default)

พอร์ตอนุกรมถูกปิดโดยอัตโนมัติโดยคอนโทรล MSComm เมื่อแอปพลิเคชั่นสิ้นสุดการทำงาน โดยถ้าหากหมายเลขพอร์ตอนุกรมที่กำหนดให้เปิดไม่มีการติดตั้งอยู่จริง ก็จะเกิดข้อผิดพลาดหมายเลข 68 (Device unsvailable) ทันที

Rthreshold

รายงานหรือกำหนดตัวอักษรที่จะรับเข้าก่อนที่คอนโทรล MSComm จะกำหนดให้คุณสมบัติ ComEven มีค่าเท่ากับ comEvRecieve และมีการเรียกโพรซีเยอร์เหตุการณ์ OnComm

รูปแบบการใช้งาน

object.Rthreshold[=value]

value หมายถึง ข้อมูลชนิดเลขจำนวนเต็ม ที่กำหนดตัวอักษรที่จะรับเข้าก่อนที่คอนโทรล MSComm ตามรายละเอียดข้างต้น

ถ้าหากคุณสมบัติ Rthreshold มีค่าเท่ากับ 0 (default) ก็จะเป็นการยกเลิกการเรียกโพรซีเยอร์เหตุการณ์ OnComm เมื่อมีการรับเข้าตัวอักษรมายังบัฟเฟอร์ด้านรับเข้า และในทางกลับกันถ้าหากคุณสมบัติ Rthreshold มีค่าเท่ากับ 1 ก็จะมีการเรียกโพรซีเยอร์เหตุการณ์ OnComm ทุกครั้งที่มีการรับตัวอักษรมายังบัฟเฟอร์ด้านรับเข้า

RTSEnable

กำหนดให้มีการใช้สาย Request To Send (RTS) ซึ่งปกติสัญญาณ Request To Send จะถูกส่งจากเครื่องคอมพิวเตอร์ไปยังโมเด็ม เพื่อเป็นการแจ้งการขอส่งสัญญาณจากคอมพิวเตอร์

รูปแบบการใช้งาน

object.RTSEnable[=boolean]

boolean หมายถึง ข้อมูลชนิดบูลีนที่กำหนดให้มีการใช้งานสาย Request To Send ดังต่อไปนี้

True หมายถึง มีการใช้สายสัญญาณ RTS (Request To Send)

False หมายถึง ไม่มีการใช้สายสัญญาณ RTS (default)

ถ้าหากคุณสมบัติ RTSEnable มีค่าเท่ากับ True สายสัญญาณ Request To Send จะอยู่ในสถานะ high เมื่อพอร์ตอนุกรมถูกเปิด และอยู่ในสถานะ low เมื่อพอร์ตอนุกรมถูกปิด โดยปกติสายสัญญาณ Request To Send จะถูกใช้ในการติดต่อสื่อสารแบบฮาร์ดแวร์ RTS/CTS เท่านั้น

Setting

รายงานหรือกำหนดพารามิเตอร์ในการสื่อสารผ่านพอร์ตอนุกรม

รูปแบบการใช้งาน

object.Setting[=value]

value หมายถึง ข้อมูลชนิดสตริงที่กำหนดค่าพารามิเตอร์ในการสื่อสารผ่านพอร์ตอนุกรมดังนี้

รูปแบบของการกำหนดลำดับของค่าพารามิเตอร์ในการสื่อสารผ่านพอร์ตอนุกรม สำหรับคุณสมบัติ Setting จะต้องเรียงลำดับดังนี้ “BBBB,P,D,S”(โดยปกติทั่วๆไป จะมีค่าเท่ากับ “9,600 N,8,1) เพราะถ้าลำดับไม่ถูกต้องก็จะเกิดข้อผิดพลาดหมายเลข 380 (Invalid Property Value) ทันทีซึ่งสัญลักษณ์ แต่ละตัวมีความหมายดังต่อไปนี้

BBB หมายถึง ความเร็วของการส่งถ่ายข้อมูลในหน่วยของ baud rate ซึ่งในทางปฏิบัติ 1 baud rate จะมีค่าเท่ากับ 1 bps (bits per second) หรือมากกว่าก็ได้ สำหรับค่า baud rate ที่คอนโทรล MSComm สามารถรับได้มีค่าดังต่อไปนี้

110,300,600,1200,2400,9600(default),14400,19200,28800,38400,(reserved),56000 (reserved),128000(reserve)หรือ 256000(reserved)

P หมายถึง บิตพาริตี้ (parity bit) สำหรับใช้ในการตรวจสอบความถูกต้องของข้อมูล ซึ่งสามารถมีค่าได้ดังต่อไปนี้

พารามิเตอร์	ความหมาย
E	Even
M	Mark
N	None(default)
O	Odd
S	Space
D	ขนาดของบิตข้อมูล ซึ่งสามารถมีค่าได้ดังต่อไปนี้ 1(default),1.5 หรือ 2
S	ขนาดของบิตหยุด (stop bit) ซึ่งสามารถมีค่าดังต่อไป 1 (default)

Sthreshold

รายงานหรือกำหนดตัวอักษรที่น้อยที่สุดที่เก็บในบัฟเฟอร์ด้านส่งออก ก่อนคอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEven มีค่าเท่ากับ comEvSend และมีการเรียกเหตุการณ์ OnComm

รูปแบบการใช้งาน

object.Sthreshold [=value]

value หมายถึง ข้อมูลชนิดตัวเลขจำนวนเต็ม ที่กำหนดจำนวนตัวอักษรที่น้อยที่สุดที่ถูกจัดเก็บในบัฟเฟอร์ด้านส่งออก ตามรายละเอียดข้างต้น

ถ้าหากคุณสมบัติ Sthreshold มีค่าเท่ากับ 0 (default) ก็จะเป็นการยกเลิกการเรียกโพรซีเยอร์เหตุการณ์ OnComm เมื่อมีการส่งออกตัวอักษรไปยังบัฟเฟอร์ด้านส่งออก และในทางกลับกันถ้าคุณสมบัติ Sthreshold มีค่าเท่ากับ 1 ก็จะมีการเรียกโพรซีเยอร์เหตุการณ์ OnComm เมื่อบัฟเฟอร์ด้านส่งออกว่าง

โดยถ้าหากจำนวนตัวอักษรด้านส่งออกมีค่าน้อยกว่าค่าตัวเลขที่กำหนด คุณสมบัติ CommEven ก็จะมีค่าเท่ากับ comEvSend และพร้อมทั้งเกิดการเรียกโพรซีเยอร์เหตุการณ์ OnComm ทันที เช่น กำหนดให้คุณสมบัติ Sthreshold มีค่าเท่ากับ 10 และถ้าหากจำนวนตัวอักษรในบัฟเฟอร์ข้อมูลด้านส่งออกลดลงจาก 10 เป็น 9 ก็จะเกิดเหตุการณ์ OnComm ทันที

OnComm

เกิดขึ้นเมื่อเกิดการเปลี่ยนแปลงค่าของคุณสมบัติ CommEvent ซึ่งเป็นการบอกถึงการเกิดข้อผิดพลาดหรือมีการสื่อสารเกิดขึ้นก็ได้

รูปแบบโพรซีเยอร์เหตุการณ์

Private Sub object_OnComm()

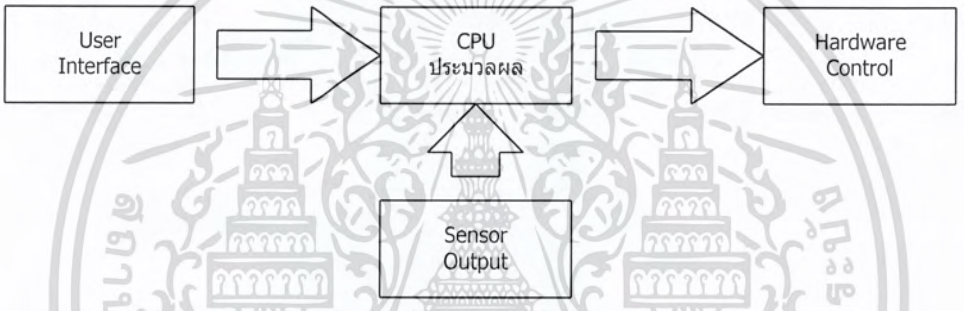
โดยปกติเมื่อคอนโทรล MSComm มีการเรียกโพรซีเยอร์เหตุการณ์ OnComm เรามักจะมีการเขียนโค้ดภายในโพรซีเยอร์เหตุการณ์นี้ เพื่อทำการตรวจสอบค่าของคุณสมบัติ CommEvent ทั้งนี้เพื่อตรวจสอบสถานะของการสื่อสารหรือข้อผิดพลาดที่เกิดขึ้นนั่นเอง

บทที่ 4 โครงสร้างและการออกแบบโครงงาน

โครงงานนี้ประกอบด้วยส่วนสำคัญๆ 3 ส่วนคือ

1. ส่วนของการ Interface กับComputer

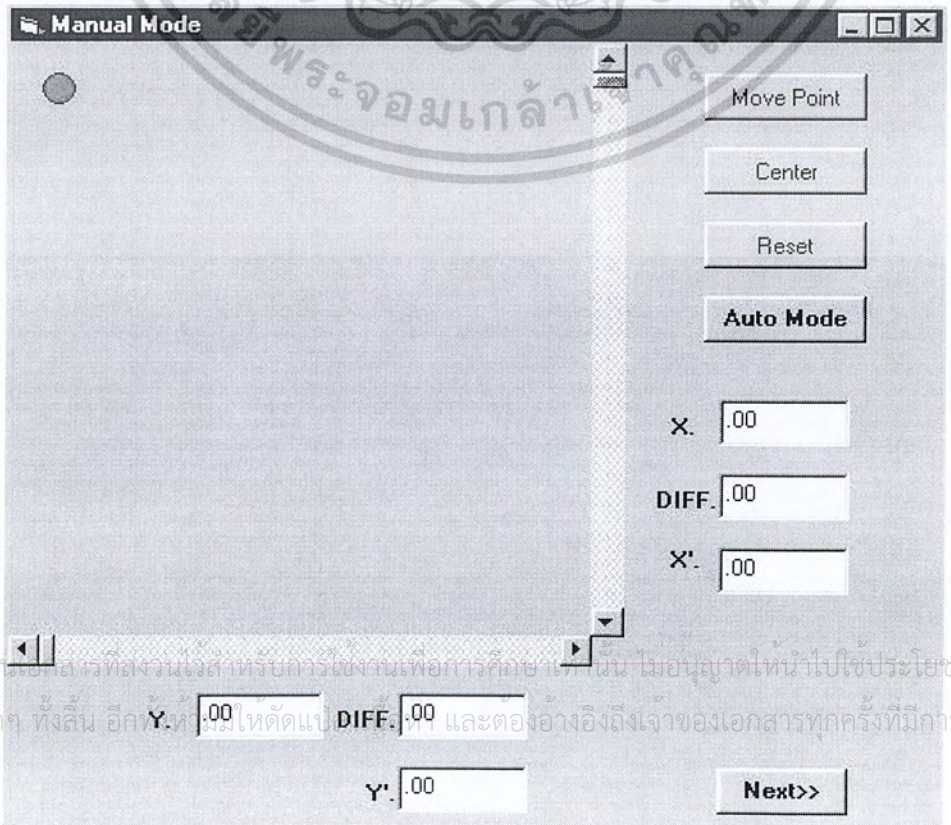
- ใช้ MSComm Control ใน Visual Basic มาใช้ควบคุมการทำงานซึ่ง สามารถเขียนเป็น Block Diagram ได้ดังนี้



-Soft Ware ที่สร้างขึ้น และ วิธีการใช้งาน

Manual Mode

เมื่อ User ทำการเปิด Soft Ware ที่สร้างขึ้นก็จะได้ภาพดังรูป



การเปิดโปรแกรมขึ้นมาตอนแรกจะอยู่ใน Manual Mode การทำงานของโปรแกรม จะใช้ Scroll Bar ในการเลื่อนตำแหน่งตามพิกัดที่เราต้องการ โดยระยะในการเคลื่อนที่ตามแนวแกน X และแกน Y คือ 60 CM. ซึ่งค่าที่แสดงใน Text Box หมายถึง

- X,Y คือ ค่าของ Scroll Bar หรือ ระยะที่เราต้องการให้เคลื่อนที่
- DIFF คือ ระยะที่เปลี่ยนแปลงจากเดิม
- X',Y' คือ ค่าใหม่ของ Scroll Bar หลังจากเคลื่อนที่แล้ว(ใช้ Check การทำงานของโปรแกรม)

เมื่อเลือกตำแหน่งที่ต้องการยิงได้แล้วจากนั้นจึงทำการคลิ๊กปุ่ม Move Point เพื่อสั่งงานให้หัวยิงกระสุนเลื่อนไปตามตำแหน่งนั้น

สำหรับปุ่มอื่นๆมีหน้าที่ดังต่อไปนี้

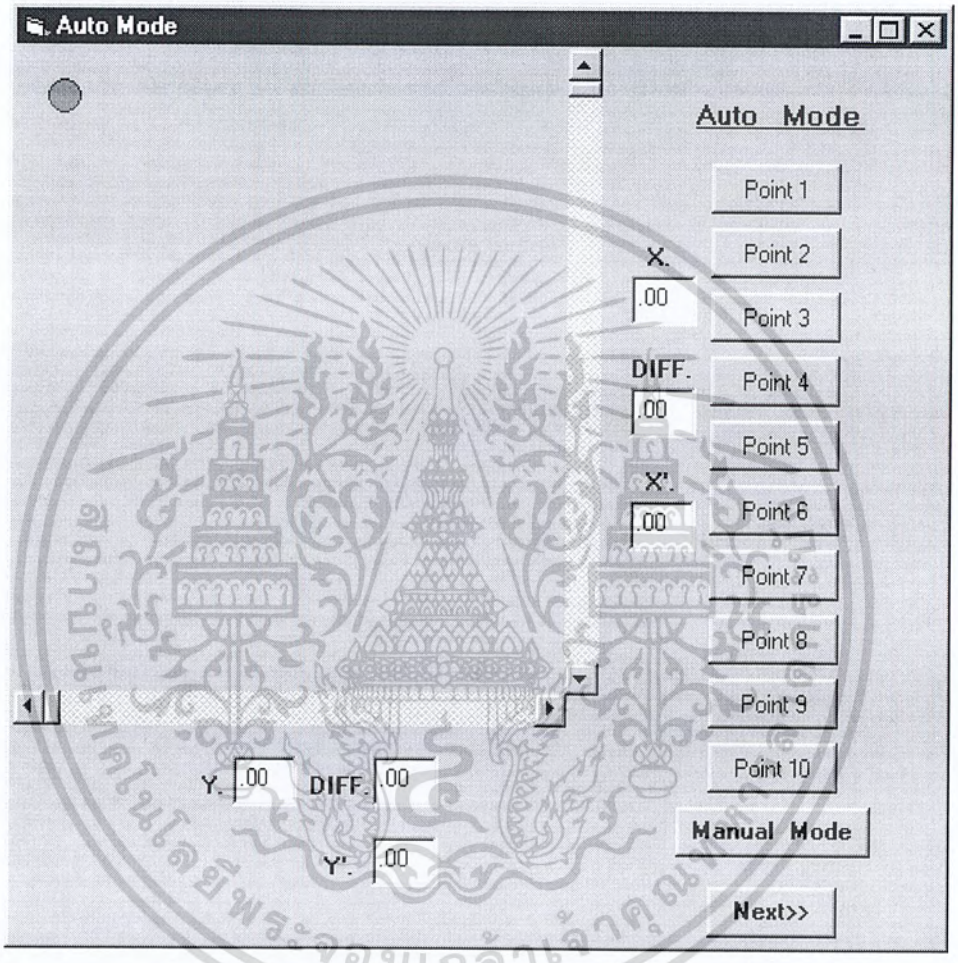
-Center ตั้งค่าตำแหน่งไว้ที่ศูนย์กลาง(30,30)

-Reset ตั้งค่าไว้ที่จุด 0,0

-Auto Mode เข้าสู่การทำงานใน Auto Mode

Auto Mode

การทำงานใน Auto mode จะเป็นการเคลื่อนที่ตามพิกัดที่ได้กำหนดไว้ โดยกำหนดมีอยู่ทั้งหมด 10 จุด ซึ่งกำหนดมาจากพิกัดตามร่างมาตราฐาน สวทช.กำหนดไว้ดังรูป

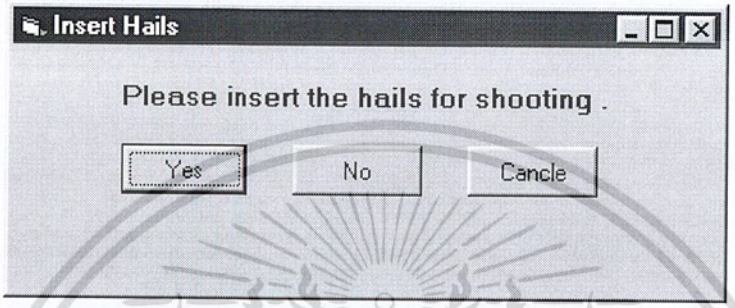


หากพิกัดที่ตั้งไว้ไม่ตรงตามต้องการก็สามารถกลับเข้าสู่การทำงานใน Manual Mode ได้โดยคลิ๊กปุ่ม Manual Mode

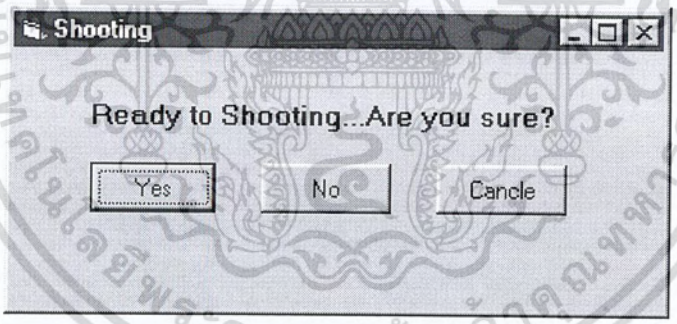
หากได้จุดตามต้องการแล้วให้คลิ๊กปุ่ม Next เพื่อเข้าสู่การทำงานขั้นต่อไป

Insert Hail

ในขั้นตอนนี้จะเป็นการบรรจุกระสุนเข้าสู่หัวยิงโดยโปรแกรมจะบอกให้เราทำการบรรจุกระสุนโดยกดปุ่ม Yes หากเราไม่ต้องการที่จะยิงกด No โปรแกรมจะจบการทำงาน และหากเราต้องการจะเปลี่ยนพิกัดใหม่กด Cancle โปรแกรมจะกลับไปสู่ Manual Mode ซึ่งแสดงดังรูป

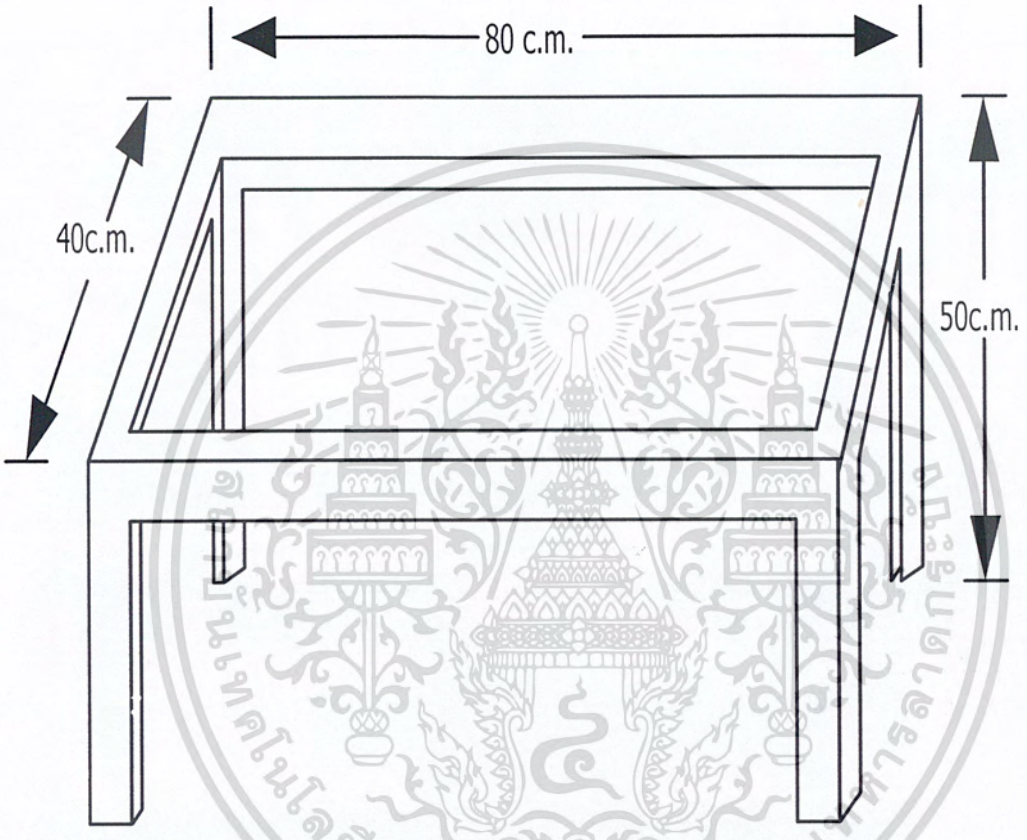


เมื่อกด Yes และทำการบรรจุกระสุนพร้อมยิงแล้วจะเข้าสู่ขั้นตอนการยิงซึ่งโปรแกรมจะให้เราตัดสินใจว่าจะยิงหรือไม่ดังรูป

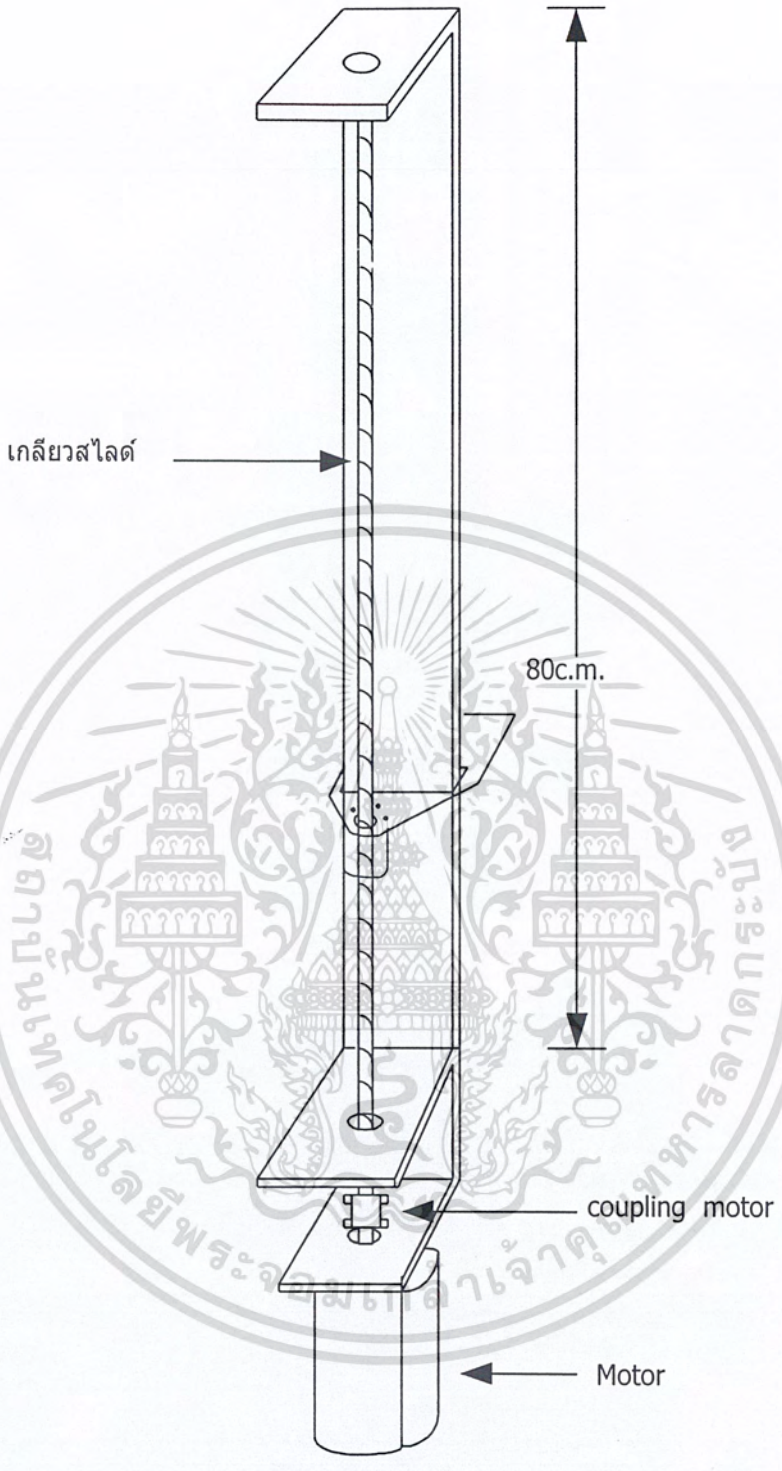


- Yes ยิงกระสุน
- No ยกเลิก จบการทำงาน
- Cancle ตั้งค่าพิกัดใหม่ กลับสู่ Manual Mode

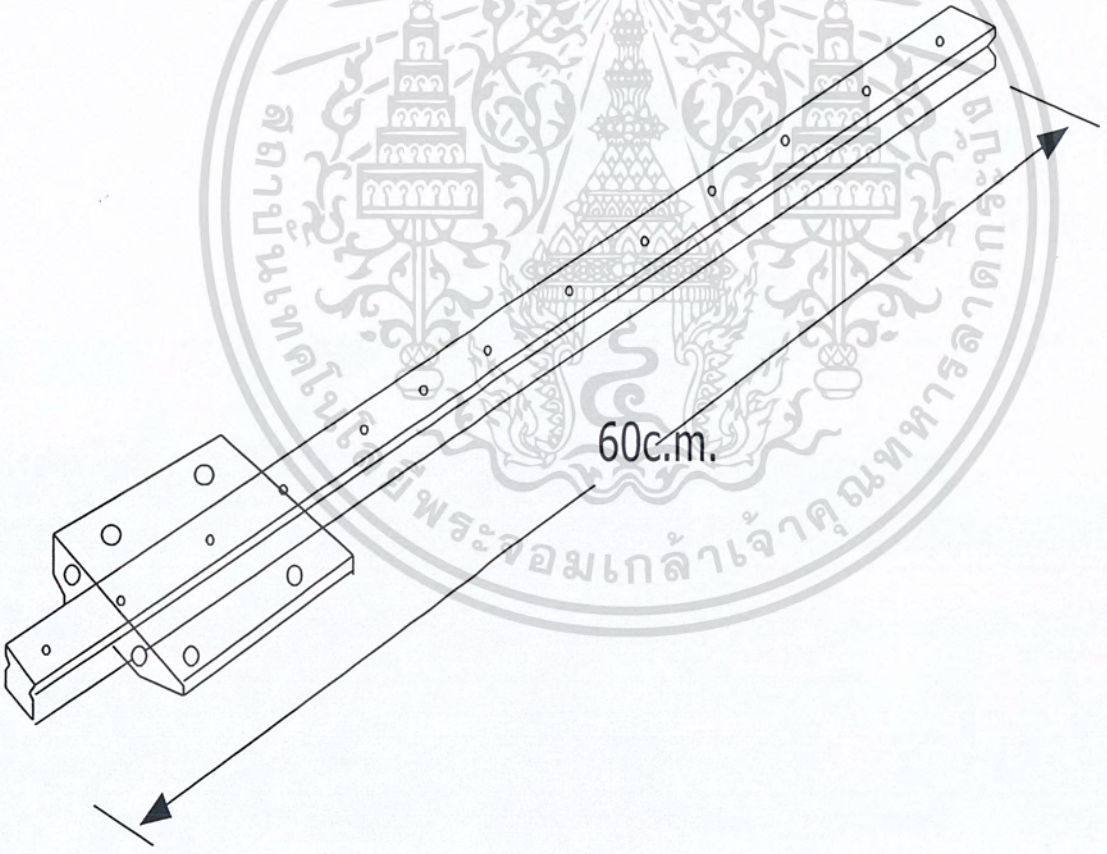
2. ส่วน Hard ware ที่ใช้ทดสอบ



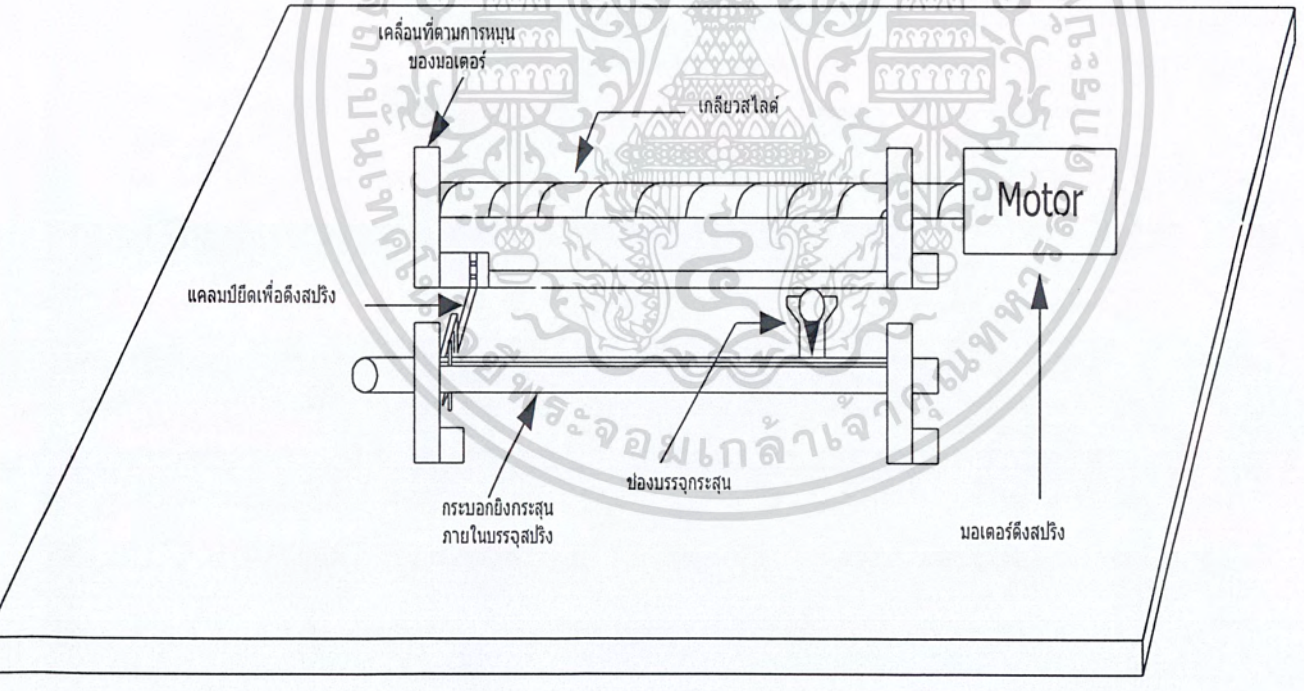
รูปที่1 โครงสร้างส่วนฐานของชุดยิง



รูปที่2 ชุดเคลื่อนที่แนวแกน Y



รูปที่3 Linear Slide สำหรับเลื่อนชุดยิง



รูปที่4 โครงสร้างชุดหัวยิงกระสุน

การทดลองที่ 1

ทดสอบระยะเวลาของ Delay Time กับระยะการหมุนของมอเตอร์

จุดประสงค์

เพื่อทำการทดสอบหาค่า Delay Time ที่เหมาะสมในการควบคุม Hard Ware

ลำดับขั้นการทดลอง

1. เปิดโปรแกรมควบคุมการทำงานขึ้นมา
2. ทดลองกำหนดค่าใน Sub Programe Delay เท่ากับ 152000
3. ต่อ Serial Port เข้ากับ Microcontroller Broad
4. ต่อ Microcontroller Broad กับ Motor
5. ทดลอง Run Programe โดยให้ Motor หมุนไปตามทิศทางต่างๆ
6. บันทึกค่าที่วัดได้ในตาราง

ผลการทดลอง

ทิศทาง	ค่า Delay Time	ระยะเคลื่อนที่
Right	152000 μsec	1 c.m.
Left	152000 μsec	1 c.m.
Front	152000 μsec	1 c.m.
Back	152000 μsec	1 c.m.
Right	340000 μsec	2 c.m.
Left	340000 μsec	2 c.m.
Front	340000 μsec	2 c.m.
Back	340000 μsec	2 c.m.

การทดลองที่ 2

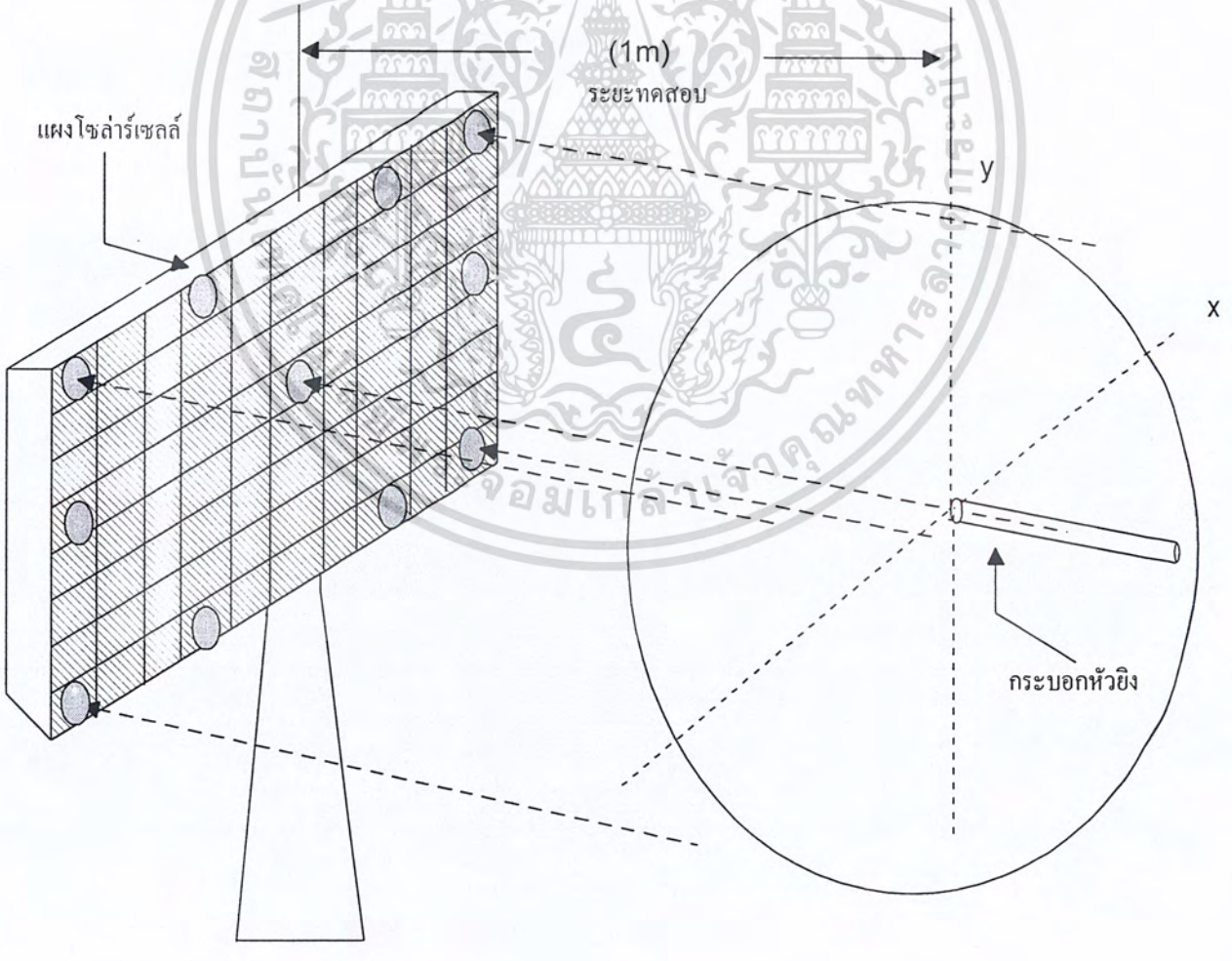
การทำงานร่วมกันของ Soft Ware และ Hard Ware

จุดประสงค์

ทดลองการทำงานและ ทำการปรับค่าให้แน่นอน

ลำดับการทดลอง

1. เปิดโปรแกรมควบคุมการทำงานขึ้นมา
2. ทดลอง Run Programe และ กดปุ่ม Reset
3. ตรวจสอบจุดพิกัดของหัวยิงกระสุน
4. ทดลองเลื่อนจุดตาม Manual Mode
5. ตรวจสอบจุดพิกัดของหัวยิงกระสุน



ผลการทดลอง

ปุ่มที่ใช้	พิกัดหัวยิง (mm)	ระยะผิดพลาด (mm)
Reset	0,0	0.5,0.5
1	550,550	1,1
2	150,580	0.5,1
3	120,300	0.5,0.75
4	480,300	0.8,0.75
5	100,100	0.5,0.5
6	500,100	1,0.5
7	20,570	0.5,1
8	580,20	1,0.5
9	300,30	0.75,0.5
10	300,570	0.5,1

บทที่5

บทสรุปและวิจารณ์ แนวทางในการพัฒนา

5.1 สรุปโครงงาน

ชุดทดสอบลูกเห็บกับแผงโซลาเซลล์ที่เราสร้างขึ้นมาจะเน้นทั้งส่วนฮาร์ดแวร์และซอฟแวร์ ซึ่งฮาร์ดแวร์จะใช้ DC มอเตอร์ 3 ตัว ใช้ขับแกน X ,Y และชุดยิง ตามลำดับ อีกส่วนก็คือ Linear Slide และแกนเกลียว Slide ซึ่งเป็นส่วนประกอบที่สำคัญ

สุดท้ายก็คือส่วนฐานของชุดยิงซึ่งมีขนาด 40 cm * 80 cm *50 cm ตามลำดับ

ส่วนซอร์ฟแวร์ จะใช้ VB 6.0 เป็นตัวสั่งควบคุมการทำงานผ่านบอร์ด MCS-51 ควบคุมผ่านพอร์ตอนุกรม โดยคอมพิวเตอร์

จากการทดลองพอสรุปได้คือ ระยะการยิงจะมีผลต่อความแรงและความเร็วของกระสุน,พิกัด,ความแม่นยำจะขึ้นอยู่กับซอร์ฟแวร์ที่เราความควบคุม จากการทดลองสามารถยิงเป้าหมายหรือแผงโซล่าร์เซลล์ในระยะหวังผลได้

ประมาณ 0.5-1.2 เมตรผิดพลาด

ประมาณ 1- 2.5 %

จุดที่ต้องปรับปรุงแก้ไขคือชุดหัวยิง เมื่อใช้งานไปนานๆอาจจะเกิดค่าผิดพลาดได้เนื่องจากการเสื่อมสภาพของชุดแมคคานิคส์นั่นเอง

5.2 ปัญหาในการสร้างโครงงาน

- ผู้สร้างขาดอุปกรณ์จำเป็นบางอย่างในการทำงานเช่น เครื่องกลึง, เครื่องเชื่อมเหล็ก ซึ่งทางคณะไม่มี
- อะไหล่อุปกรณ์บางอย่างค่อนข้างหายากไม่ได้ สเป็คตามต้องการต้องทำการดัดแปลงแก้ไขก่อนใช้งานและราคาค่อนข้างแพง

5.3 แนวทางในการพัฒนา

แนวทางในการพัฒนาเป็นสิ่งจำเป็นที่จะช่วยให้โครงงานมีประสิทธิภาพและมาตราฐานเป็นที่ยอมรับในระดับสากล เช่น ติดตั้ง SENSER ให้กับระบบและปรับปรุงให้มีค่าผิดพลาดต่ำลง

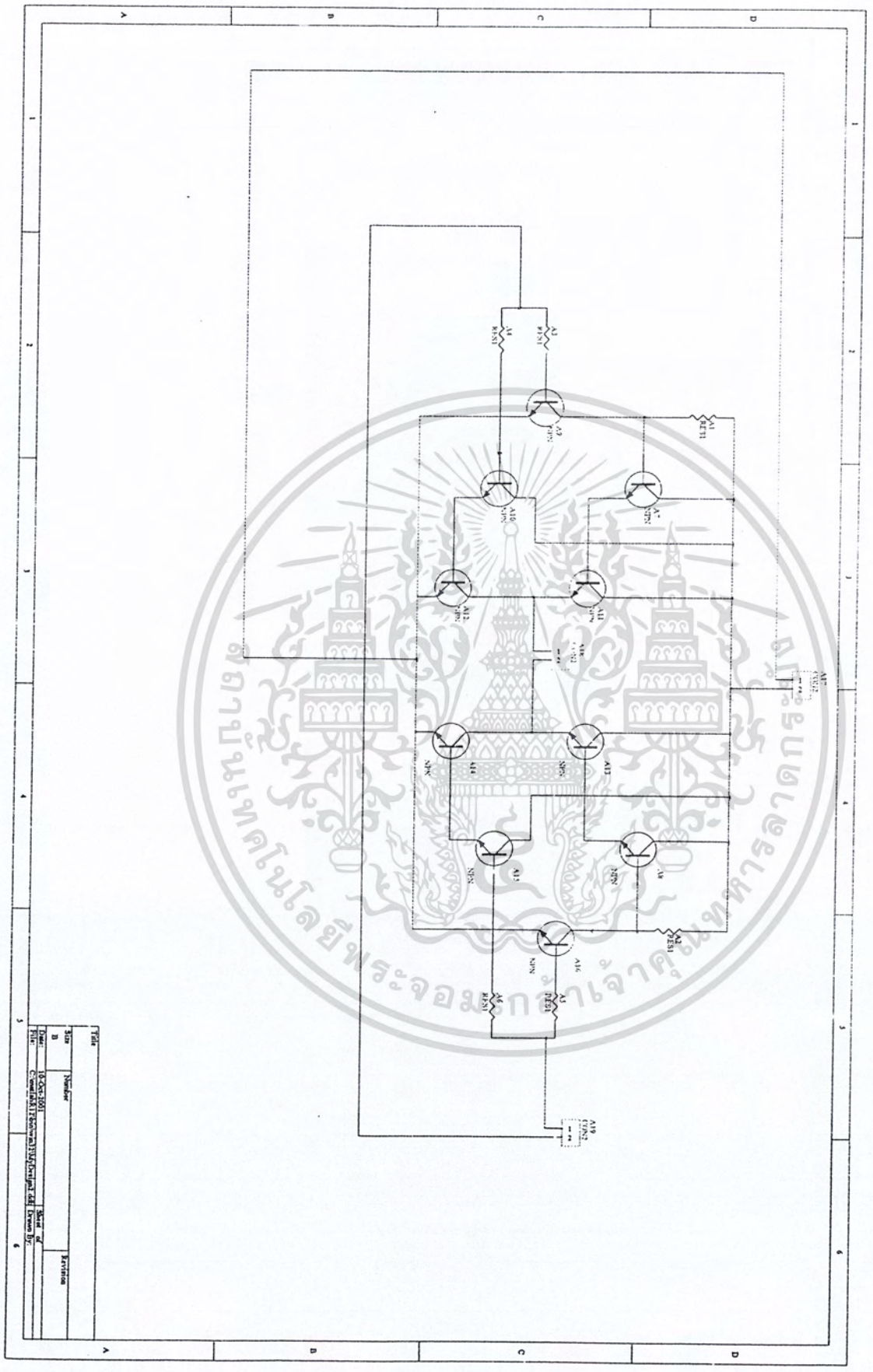
- ติดตั้งกล้องเพื่อให้ SCAN หาพิกัดได้ง่ายและแม่นยำ
- ออกแบบชุดยิงให้เป็นระบบนิวเมติกส์เพื่อง่ายต่อการควบคุม

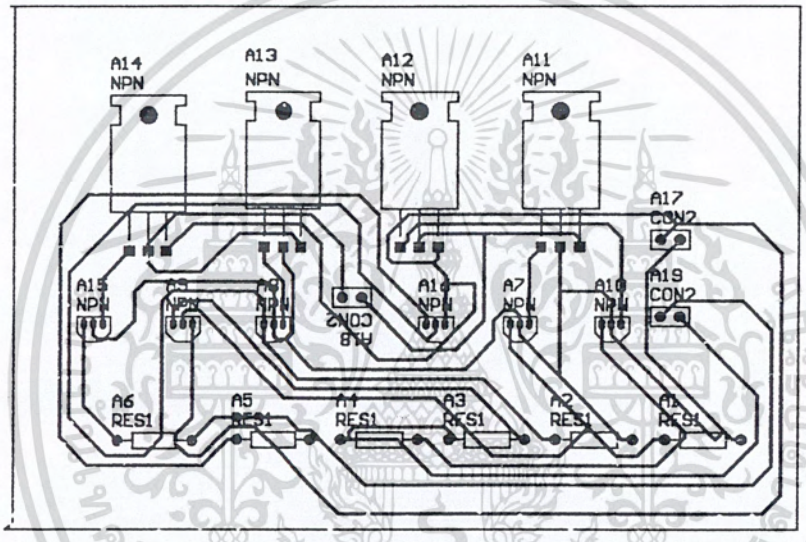
จากบทสรุปและแนวทางที่กล่าวมาทั้งหมดผู้สร้างหวังว่าโครงงานชิ้นนี้จะให้ประโยชน์แด่ผู้ที่สนใจและทำการศึกษารวมทั้งเป็นแนวทางในการพัฒนาไม่มากก็น้อย หากบกพร่องประการใดผู้สร้างยินดีรับคำติชมทุกประการ

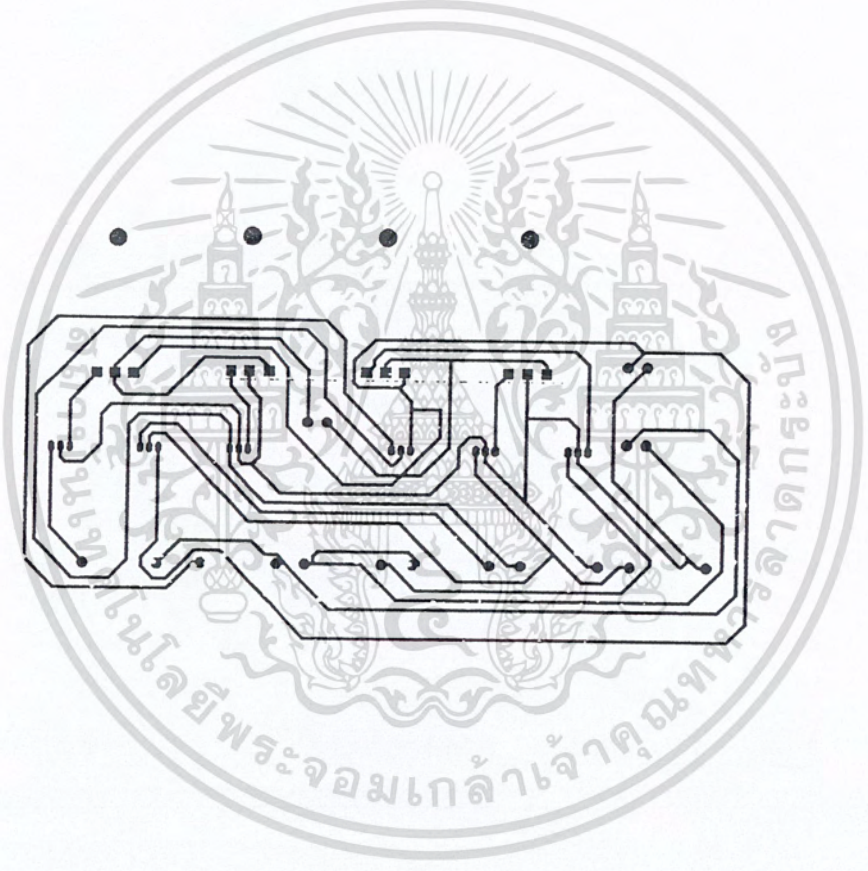
ภาคผนวก ก.

วงจร Controll Motor และ PCB

1.วงจร Controll Motor แกน X และแกน Y







ภาคผนวก ข.

ส่วนขอของ โปรแกรมที่ใช้ควบคุมการทำงาน

โปรมแกรม Assembly สำหรับบอร์ดควบคุม Hard Ware

```
;*************************************************
;*                                               *
;*  FILENAME      PROJECT2.ASM                   *
;*  DESCRIPTION   PROGRAM                        *
;*  HARDWARE      NONE                           *
;*  ASSEMBLER     SXA51                          *
;*  START-DATE    28/06/2544                     *
;*  UPDATE        05/10/2544                     *
;*  ENGINEER      O.JEDT                         *
;*  COMPANY       INDUSTRIAL TECHNOLOGY (KMIT'L) *
;*                                               *
;*************************************************

              ORG     0000H
              AJMP    MAIN
MAIN:         MOV     DPTR,#07FFH
              ACALL   POR
              MOV     SP,#100
              MOV     TMOD,#021H
              MOV     SCON,#052H
              MOV     PCON,#70H
              MOV     TH1,#0FBH
              MOV     TL1,#0FBH
              SETB    TR1

;************* GET DATA AND RETURN VALUE ******************
```

```
M:      JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI

;*************************************************************

        CJNE    A,#'1',C1
        MOV     P1,#01H
        SJMP    M
C1:     CJNE    A,#'2',C2
        MOV     P1,#02H
        SJMP    M
C2:     CJNE    A,#'3',C3
        MOV     P1,#04H
        SJMP    M
C3:     CJNE    A,#'4',C4
        MOV     P1,#08H
        SJMP    M
C4:     CJNE    A,#'5',C5
        MOV     P1,#10H
        SJMP    M
C5:     CJNE    A,#'6',C6
        MOV     P1,#20H
        SJMP    M
C6:     CJNE    A,#'7',C7
        MOV     P1,#40H
```

```
                SJMP    M
C7:             CJNE    A,#'8',C8
                MOV     P1,#80H
                SJMP    M
C8:             CJNE    A,#'9',C9
                MOV     P2,#01H
                SJMP    M
C9:             CJNE    A,#'A',CA
                MOV     P2,#02H
                SJMP    M
CA:             CJNE    A,#'B',CB
                MOV     P2,#04H
                SJMP    M
CB:             CJNE    A,#'C',CC
                MOV     P2,#08H
                SJMP    M
CC:             CJNE    A,#'D',CD
                MOV     P2,#10H
                SJMP    M
CD:             CJNE    A,#'E',CE
                MOV     P2,#020H
                AJMP    M
CE:             CJNE    A,#'F',CF
                MOV     P2,#40H
                AJMP    M
CF:             CJNE    A,#'0',C0
                MOV     P2,#80H
                AJMP    M
;               CLEAR PORT
C0:               MOV     P1,#0
```

```
                MOV     P2,#0
                AJMP    M

;*****************************************

POR:            INC     DPTR
POR1:           MOV     A,DPH
                CJNE    A,#0,POR
                RET

;*****************************************
 END
```

-โปรแกรม Visual Basic ใช้สั่งงานผ่าน พอร์ตอนุกรม และติดต่อกับผู้ใช้

1. Manual Mode

```
Public i As Integer, j As Integer, m As Integer, n As Integer, x As Integer, y As Integer
Private Sub LCDdelay1()
Dim a As Long
  For a = 1 To 500000 * x
  Next a
End Sub
Private Sub LCDdelay2()
Dim b As Long
  For b = 1 To 500000 * y
  Next b
End Sub

Private Sub Command1_Click()
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
```

```
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
```

```
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command2_Click()
VScroll1.Value = 30
HScroll1.Value = 30
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
```

```
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub


Private Sub Command3_Click()
VScroll1.Value = 0
HScroll1.Value = 0
```

```
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
```

```
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command4_Click()
HScroll1.Value = 0
VScroll1.Value = 0
i = VScroll1.Value
j = HScroll1.Value
Text3.Text = Format(HScroll1.Value, "###.00")
Text4.Text = Format(VScroll1.Value, "###.00")
If i > m Then
x = i - m
MSComm1.Output = "1"
```

```
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Cail LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
```

```
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
End If
Unload Form1
Form2.Show
End Sub
Private Sub Command5_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
End If
Unload Form1
Form3.Show
End Sub

Private Sub Form_Load()
```

```
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
HScroll1.Value = 0
VScroll1.Value = 0
i = VScroll1.Value
j = HScroll1.Value
Text1.Text = Format(HScroll1.Value, "###.00")
Text2.Text = Format(VScroll1.Value, "###.00")
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
```

```
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub HScroll1_Change()
Text1.Text = Format(HScroll1.Value, "###.00")
j = HScroll1.Value
```

```
If HScroll1.Value = 0 Then
Shape1.Left = HScroll1.Value + 240
Else
Shape1.Left = HScroll1.Value * 68
End If
End Sub

Private Sub HScroll1_Scroll()
HScroll1_Change
End Sub

Private Sub MSComm1_OnComm()
MSComm1.InBufferCount = 0
End Sub

Private Sub VScroll1_Change()
Text2.Text = Format(VScroll1.Value, "###.00")
i = VScroll1.Value
If VScroll1.Value = 0 Then
Shape1.Top = VScroll1.Value + 240
Else
Shape1.Top = VScroll1.Value * 74
End If
End Sub

Private Sub VScroll1_Scroll()
VScroll1_Change
End Sub
```

2. Auto Mode

```
Public i As Integer, j As Integer, m As Integer, n As Integer, x As Integer, y As Integer
Private Sub LCDdelay1()
Dim a As Long
  For a = 1 To 50000 * x
  Next a
End Sub
Private Sub LCDdelay2()
Dim b As Long
  For b = 1 To 50000 * y
  Next b
End Sub
Private Sub Command1_Click()
VScroll1.Value = 55
HScroll1.Value = 55
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
```

```
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
```

```
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command10_Click()
VScroll1.Value = 57
HScroll1.Value = 30
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
```

```
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command11_Click()
HScroll1.Value = 0
```

```
VScroll1.Value = 0
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
```

```
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
End If
Unload Form2
Form1.Show
End Sub
Private Sub Command12_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
```

```
End If
Unload Form2
Form3.Show
End Sub

Private Sub Command2_Click()
VScroll1.Value = 58
HScroll1.Value = 25
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
```

```
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub
Private Sub Command3_Click()
VScroll1.Value = 30
HScroll1.Value = 20
```

```
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
```

```
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command4_Click()
VScroll1.Value = 30
HScroll1.Value = 40
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
```

```
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
```

```
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command5_Click()
VScroll1.Value = 10
HScroll1.Value = 10
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
```

```
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
```

```
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command6_Click()
VScroll1.Value = 10
HScroll1.Value = 50
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
```

```
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command7_Click()
```

```
VScroll1.Value = 56
HScroll1.Value = 2
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
```

```
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Command8_Click()
VScroll1.Value = 4
HScroll1.Value = 58
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
```

```
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
```

```
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub
Private Sub Command9_Click()
VScroll1.Value = 3
HScroll1.Value = 30
i = VScroll1.Value
j = HScroll1.Value
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
```

```
MSComm1.Output = "2"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
```

```
m = i
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub

Private Sub Form_Load()
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
HScroll1.Value = 0
VScroll1.Value = 0
i = VScroll1.Value
j = HScroll1.Value
Text1.Text = Format(HScroll1.Value, "###.00")
Text2.Text = Format(VScroll1.Value, "###.00")
If i > m Then
x = i - m
MSComm1.Output = "1"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
ElseIf i < m Then
x = m - i
MSComm1.Output = "2"
Call LCDdelay1
```

```
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Else
End If
If j > n Then
y = j - n
Text1.Text = Chr(&H32)
MSComm1.Output = "3"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
ElseIf j < n Then
y = n - j
MSComm1.Output = "4"
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
Call LCDdelay2
MSComm1.Output = Chr(&H0)
Else
End If
Text4.Text = Format(x, "###.00")
m = i
```

```
Text6.Text = Format(m, "###.00")
Text3.Text = Format(y, "###.00")
n = j
Text5.Text = Format(n, "###.00")
End Sub
Private Sub HScroll1_Change()
Text1.Text = Format(HScroll1.Value, "###.00")
j = HScroll1.Value
If HScroll1.Value = 0 Then
Shape1.Left = HScroll1.Value + 240
Else
Shape1.Left = HScroll1.Value * 68
End If
End Sub

Private Sub HScroll1_Scroll()
HScroll1_Change
End Sub

Private Sub MSComm1_OnComm()
MSComm1.InBufferCount = 0
End Sub

Private Sub VScroll1_Change()
Text2.Text = Format(VScroll1.Value, "###.00")
i = VScroll1.Value
If VScroll1.Value = 0 Then
Shape1.Top = VScroll1.Value + 240
Else
Shape1.Top = VScroll1.Value * 74
```

```
End If
End Sub

Private Sub VScroll1_Scroll()
VScroll1_Change
End Sub
```

3. Insert Hail

```
Private Sub LCDdelay1()
Dim a As Long
  For a = 1 To 500000
  Next a
End Sub
Private Sub Command1_Click()
MSComm1.Output = "5"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = "7"
MSComm1.Output = "6"
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
Call LCDdelay1
MSComm1.Output = Chr(&H0)
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
```

```
End If
Unload Form3
Form4.Show
End Sub

Private Sub Command2_Click()
End
End Sub

Private Sub Command3_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
End If
Unload Form3
Form1.Show
End Sub

Private Sub Form_Load()
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
End Sub

Private Sub MSComm1_OnComm()
MSComm1.InBufferCount = 0
End Sub
```

4. Shooting

```
Private Sub LCDdelay1()
Dim a As Long
  For a = 1 To 500000
  Next a
End Sub
Private Sub Command1_Click()
MSComm1.Output = Chr(&H0)
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
End If
Unload Form4
Form5.Show
End Sub

Private Sub Command2_Click()
End
End Sub

Private Sub Command3_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
End If
Unload Form4
Form1.Show
End Sub

Private Sub Form_Load()
```

```
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
End Sub

Private Sub MSComm1_OnComm()
MSComm1.InBufferCount = 0
End Sub
```

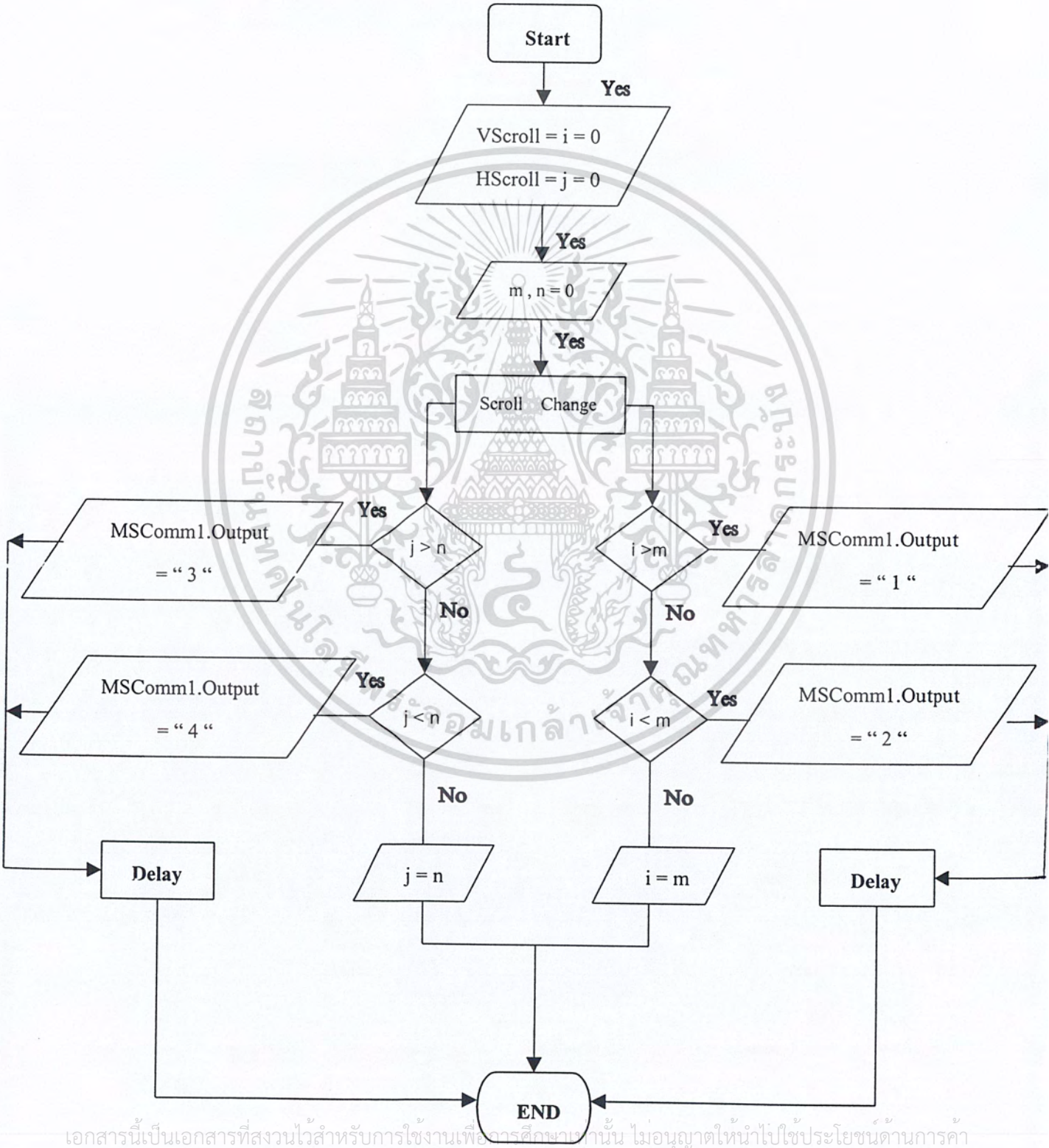
5. Question

```
Private Sub Command1_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Else
End If
Unload Form5
Form1.Show
End Sub

Private Sub Command2_Click()
End
End Sub

Private Sub Form_Load()
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
End Sub

Private Sub MSComm1_OnComm()
MSComm1.InBufferCount = 0
End Sub
```

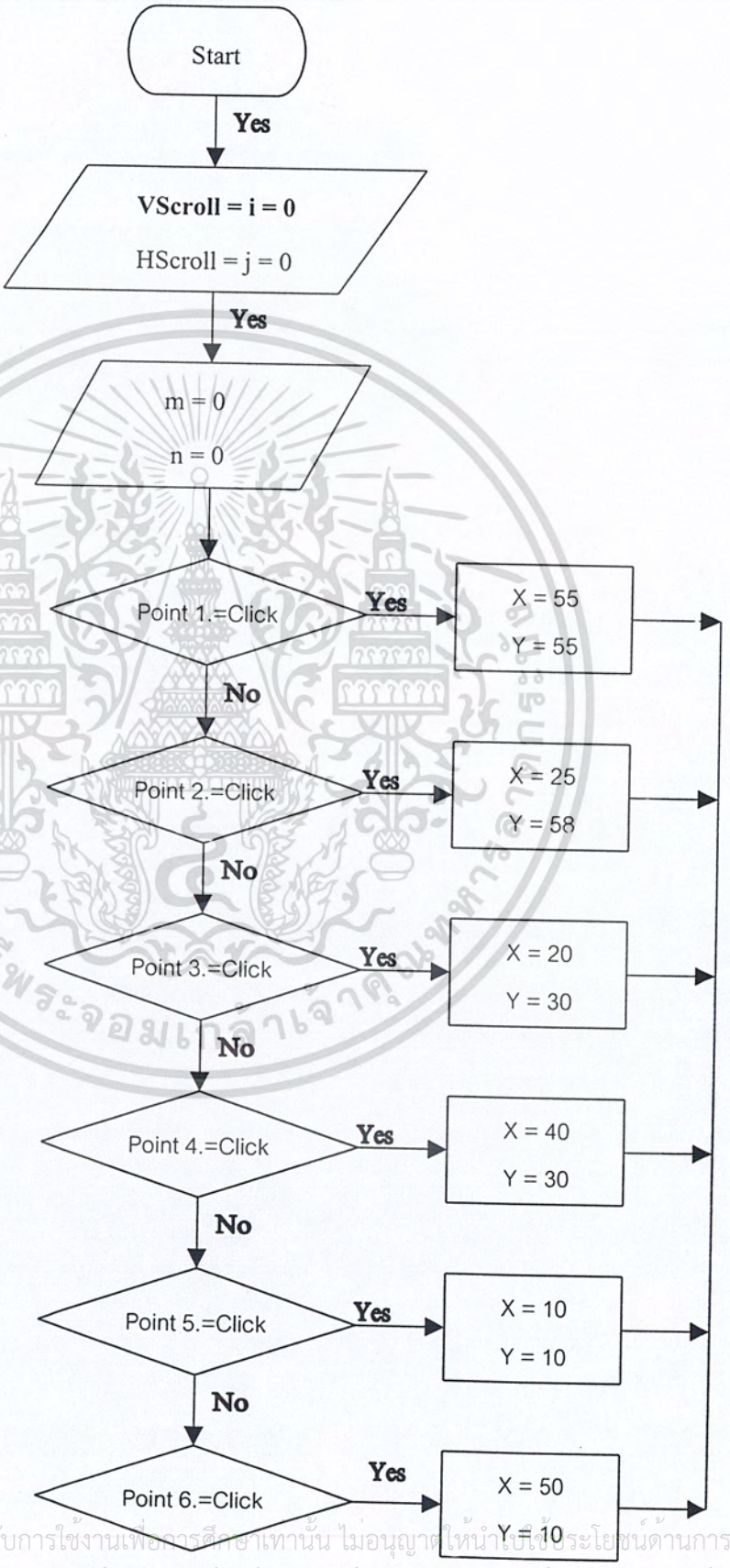
ภาคผนวก ค.

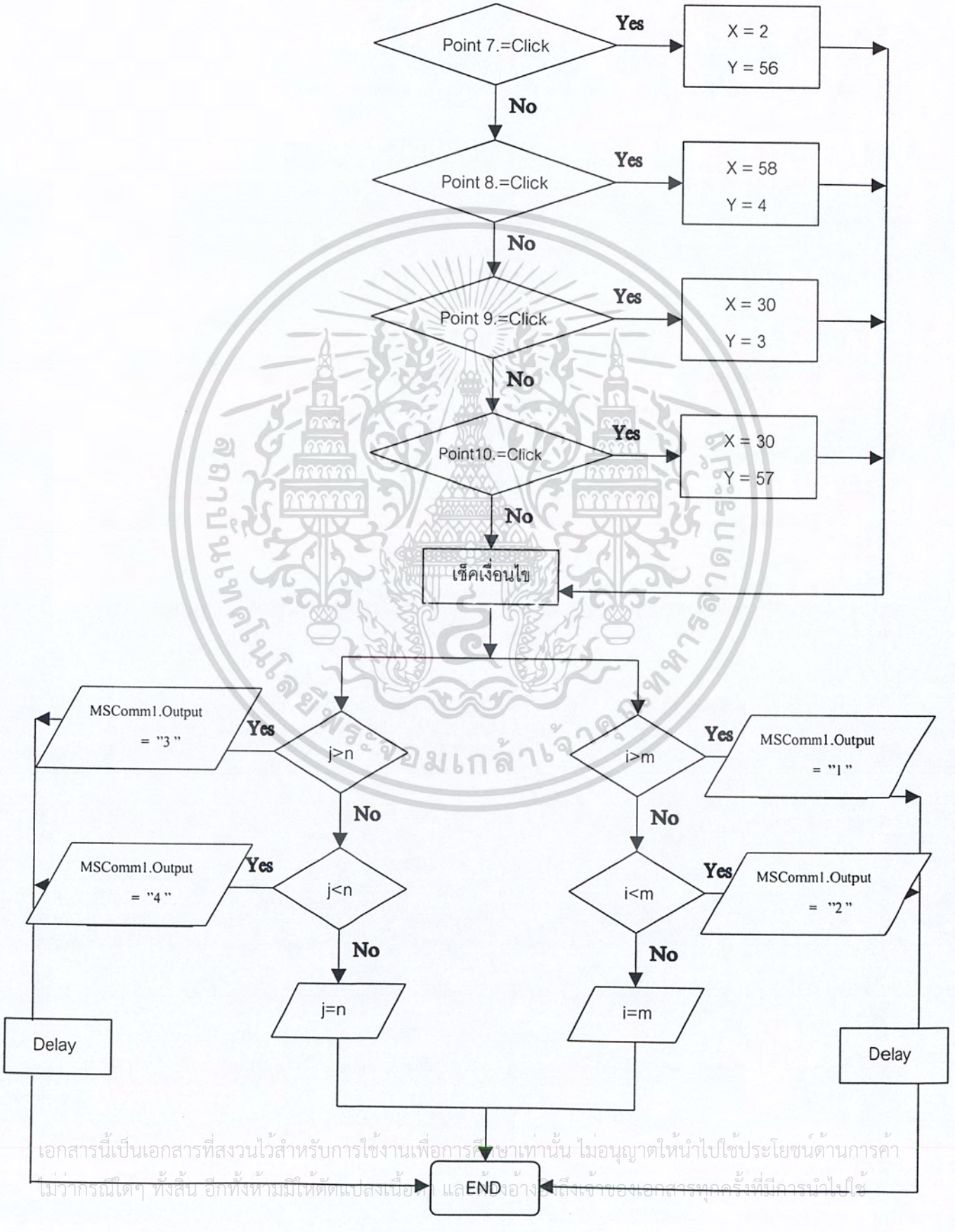
โฟลวชาร์ตแสดงการทำงานของโปรแกรม Visual Basic 6.0

1. Flow Chart ของ Manual Mode



2.Auto Mode

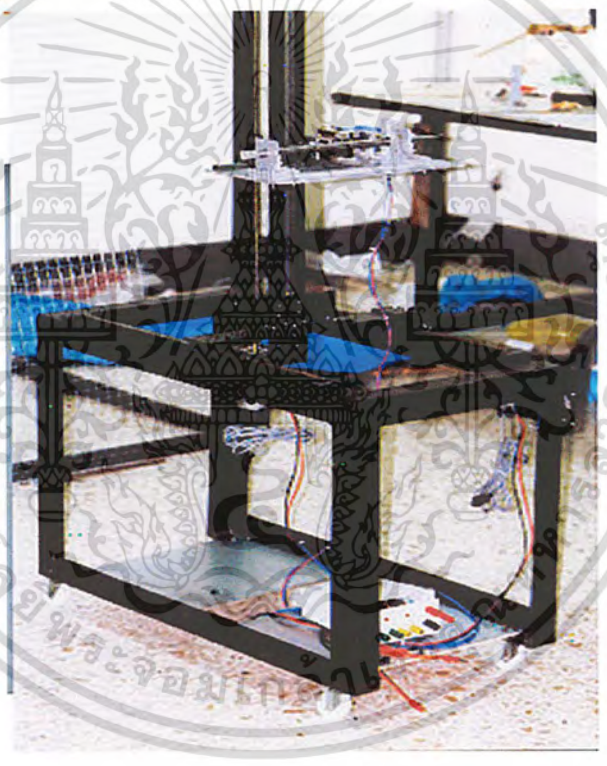




ภาคผนวก ง.

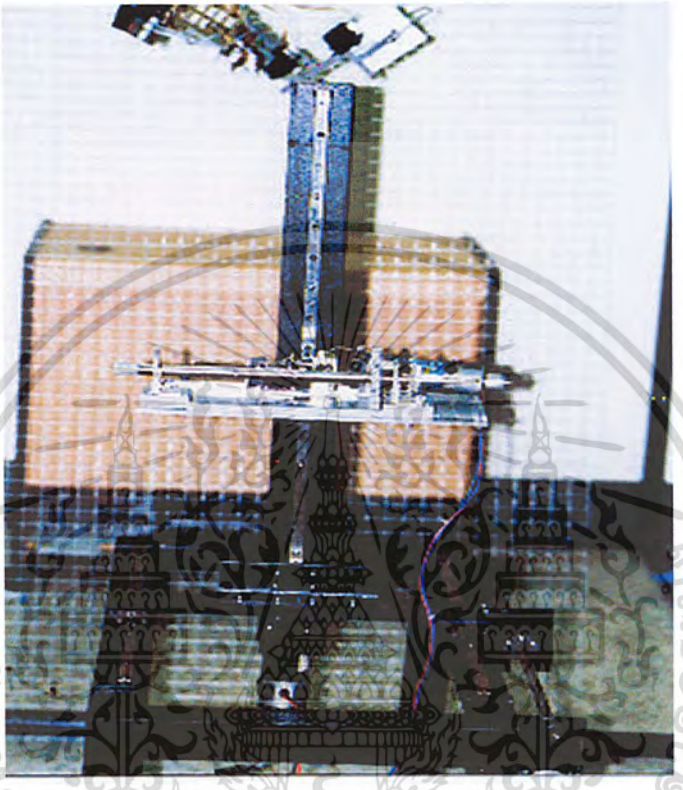
รูปถ่าย จริงของ PROJECT

ลักษณะโครงสร้างจริงของเครื่องทดสอบลูกเห็บที่สร้างเสร็จแล้วตามลักษณะภาพด้านหน้าของโครงงานแสดงดังรูป ง.1



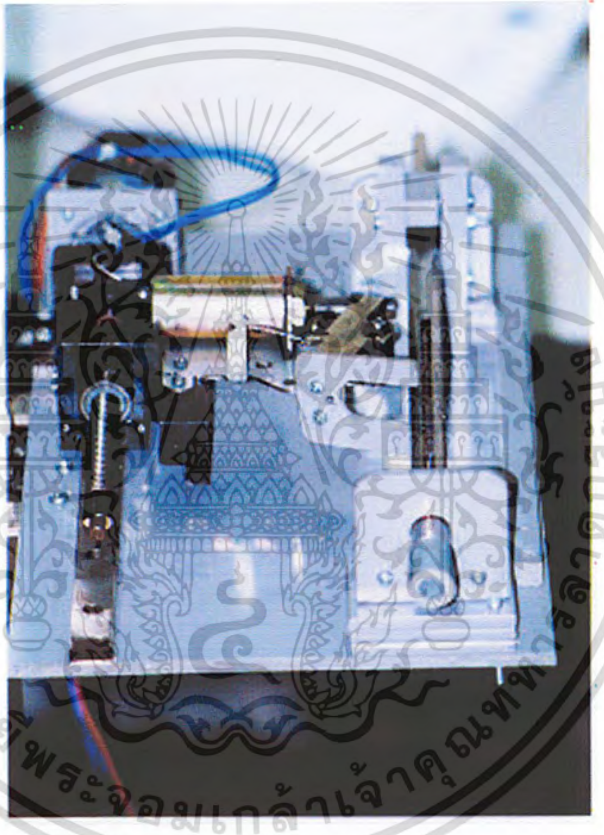
รูป ง.1แสดงภาพด้านหน้าของโครงงาน

ลักษณะโครงสร้างจริงของเครื่องทดสอบลูกเห็บที่สร้างเสร็จแล้ว ตามลักษณะภาพด้านข้างของโครงงานแสดงดังรูป ง.2



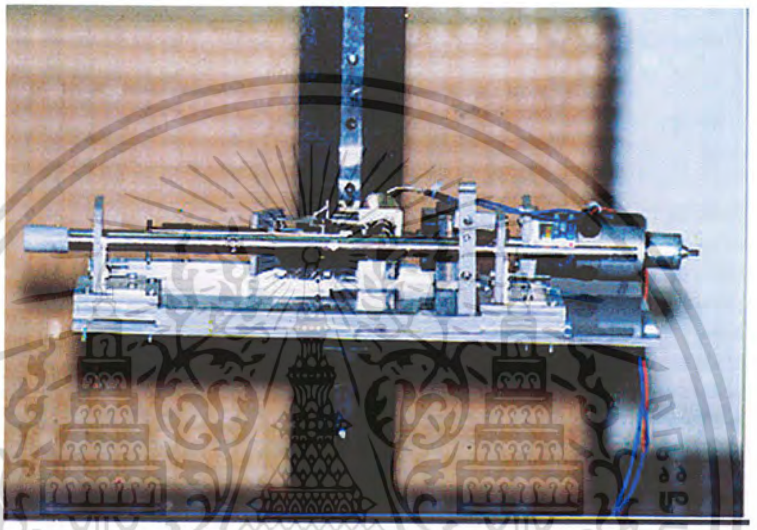
รูป ง.2 ลักษณะภาพด้านข้างของโครงงาน

ลักษณะโครงสร้างจริงของเครื่องทดสอบลูกเห็บที่สร้างเสร็จแล้ว ตามลักษณะภาพด้านหน้าชุดยิงของโครงงานแสดงดังรูป ง.3



รูป ง.3 แสดงภาพด้านหน้าชุดยิงของโครงงาน

ลักษณะโครงสร้างจริงของเครื่องทดสอบลูกเห็บที่สร้างเสร็จแล้ว ตามลักษณะภาพด้านข้างชุดยิงของโครงงานแสดงดังรูป ง.4



รูป ง.4 แสดงภาพด้านข้างชุดยิงของโครงงาน

บรรณานุกรม

1. ชัยวัฒน์ ลิ้มพรจิตวิไล,วรพจน์ กรแก้ววัฒนกุล, เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51,Innovative Experiment Co,Ltd

2.พรชัยยศ ศรีปัญญาพงศ์,เอกสารประกอบการอบรม ไมโครคอนโทรลเลอร์ 8051,ศูนย์บริการและพัฒนาวิศวกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

3.ธาริน สิทธิธรรมชารีมคู่มือการเขียนโปรแกรม Microsoft Visual Basic6.0 ,Succes Media Co,Ltd

4.กฤษฎา ใจเย็น,อรรถพล บุญยะโภคา,ชัยวัฒน์ ลิ้มพรจิตวิไล,เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม,Innovative Experiment Co,Ltd

5.คู่มือการใช้งาน CP-S8252 V2.0 Control Board , บริษัท อีทีที จำกัด