ระบบรักษาความปลอดภัยแบบกระจายผ่านอินเตอร์เน็ต

DISTRIBUTED SECURITY SYSTEM VIA INTERNET

โดย

นายกรันต์ เกียรติสิริกำพล

นายกิตติ เพชรหยอย

นายเกียรติยศ อุทัยรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ผ่านการตรวจชิ้นงานแล้ว

(ลงชื่อ).....ผู้ตรวจ

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

ระบบรักษาความปลอดภัยแบบกระจายผ่านอินเตอร์เน็ต

DISTRIBUTED SECURITY SYSTEM VIA INTERNET

โดย

นายกรันต์	เกียรติสิริกำพล	45010011
นายกิตติ	เพชรหยอย	45010045
นายเกียรติยศ	อุทัยรัตน์	45010069

อาจารย์ที่ปรึกษา

รศ.ดร.ยุทธพงษ์ รังสรรค์เสรี

รศ.ดร.ปัญญา ฐิติมัชฌิมา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบรักษาความปลอดภัยแบบกระจายผ่านอินเตอร์เน็ต**

: DISTRIBUTED SECURITY SYSTEM VIA INTERNET

ผู้จัดทำ

นายกรันต์	**เกียรติสิริกำพล**	**45010011**
นายกิตติ	**เพชรหยอย**	**45010045**
นายเกียรติยศ	**อุทัยรัตน์**	**45010069**

..... อาจารย์ที่ปรึกษา

(รศ.ดร. ภูทธพงษ์ รังธรรศ์เสรี)

..... อาจารย์ที่ปรึกษา

(รศ.ดร. ปัญญา ฐิติมัชฌิมา)

ระบบรักษาความปลอดภัยแบบกระจายผ่านอินเตอร์เน็ต

DISTRIBUTED SECURITY SYSTEM VIA INTERNET

โดย นายกรันต์ เกียรติสิริกำพล 45010011
นายกิตติ เพชรหยอย 45010045
นายเกียรติยศ อุทัยรัตน์ 45010069

อาจารย์ที่ปรึกษา รศ.ดร.ยุทธพงษ์ รังสรรค์เสรี
รศ.ดร.ปัญญา ฐิติมัชฌิมา

บทคัดย่อ

โครงงานนี้จัดทำขึ้นเพื่อรักษาความปลอดภัยหลายๆจุดผ่านอินเตอร์เน็ต โดยมีไมโครคอนโทรลเลอร์เป็นตัวควบคุมในแต่ละจุด ไมโครคอนโทรลเลอร์จะทำหน้าที่รับข้อมูลจากเซนเซอร์ และทำการส่งข้อมูลให้กับเซิร์ฟเวอร์ ซึ่งเป็นตัวติดต่อผู้ใช้ผ่านอินเตอร์เน็ต

ABSTRACT

This project is created for the purpose of the security on the internet. By using Micro controller runs into each position. Micro controller would receive the data from sensor and send that data to the server where connect with all users through the internet.

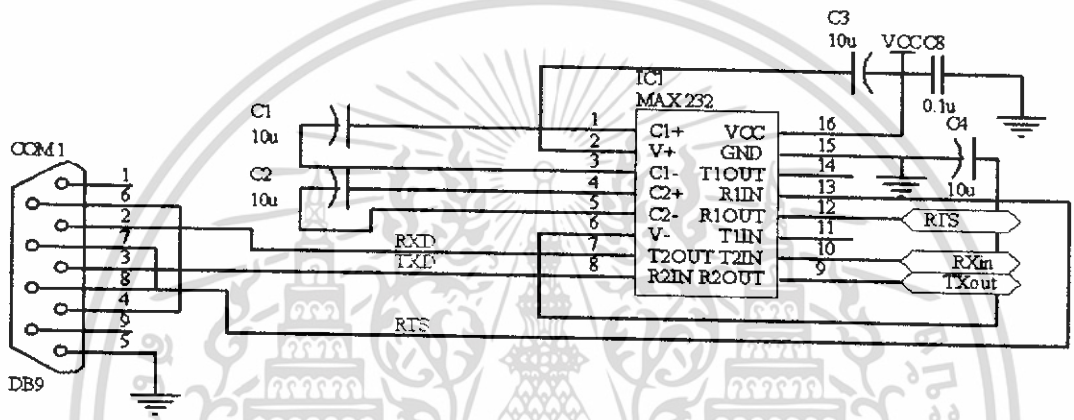
บทที่ 3

การคำนวณและการสร้าง

การคำนวณและการสร้างแบ่งเป็นสองส่วน คือ ส่วนของฮาร์ดแวร์และส่วนของซอฟท์แวร์ การคำนวณและการสร้างส่วนฮาร์ดแวร์

3.1 วงจรแปลงสัญญาณแบบ RS-232

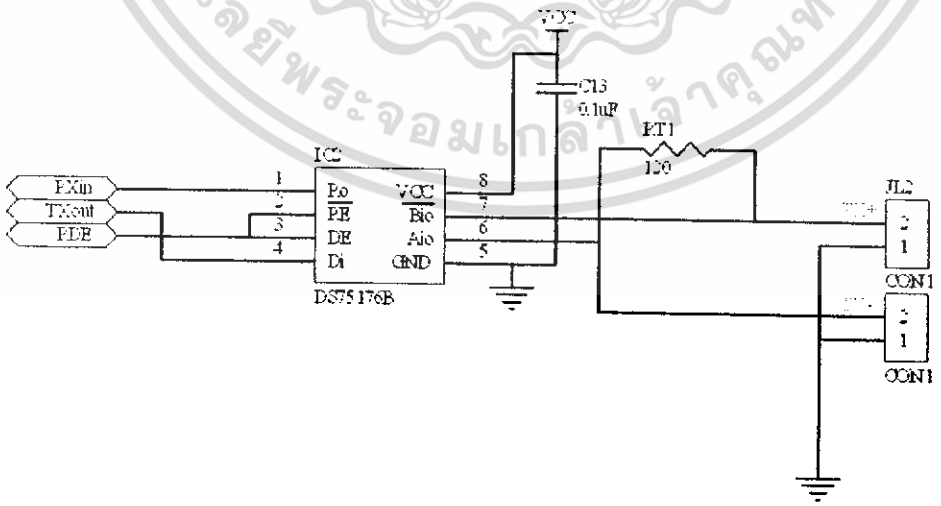
ใช้ไอซี Max 232 ซึ่งทำหน้าที่ปรับระดับสัญญาณให้เหมาะสม เชื่อมต่อกับคอมพิวเตอร์ที่ขา RxD และขา TxD ของ Com 1 หรือ Com 2 มีข้อดีคือมีทั้ง driver และ receiver ในตัวเอง และใช้โวลท์เตจระดับเดียวคือ 5 โวลต์ การนำเอาไอซี Max 232 ไปใช้นั้น ต้องมีการต่อตัวเก็บประจุเข้าไปอีกเล็กน้อย ซึ่งค่าของตัวเก็บประจุที่ใช้ในการทดลองคือ C = 10 uF ดังรูปที่ 3.1



รูปที่ 3.1 วงจรแปลงสัญญาณแบบ RS-232

3.2 วงจร RS-485 อแดปเตอร์

วงจรของตัวแปลงสัญญาณจากมาตรฐาน RS-232 ไปเป็นสัญญาณตามมาตรฐาน RS-485 นั้น สามารถแสดงให้เห็นได้ดังรูปที่ 3.2

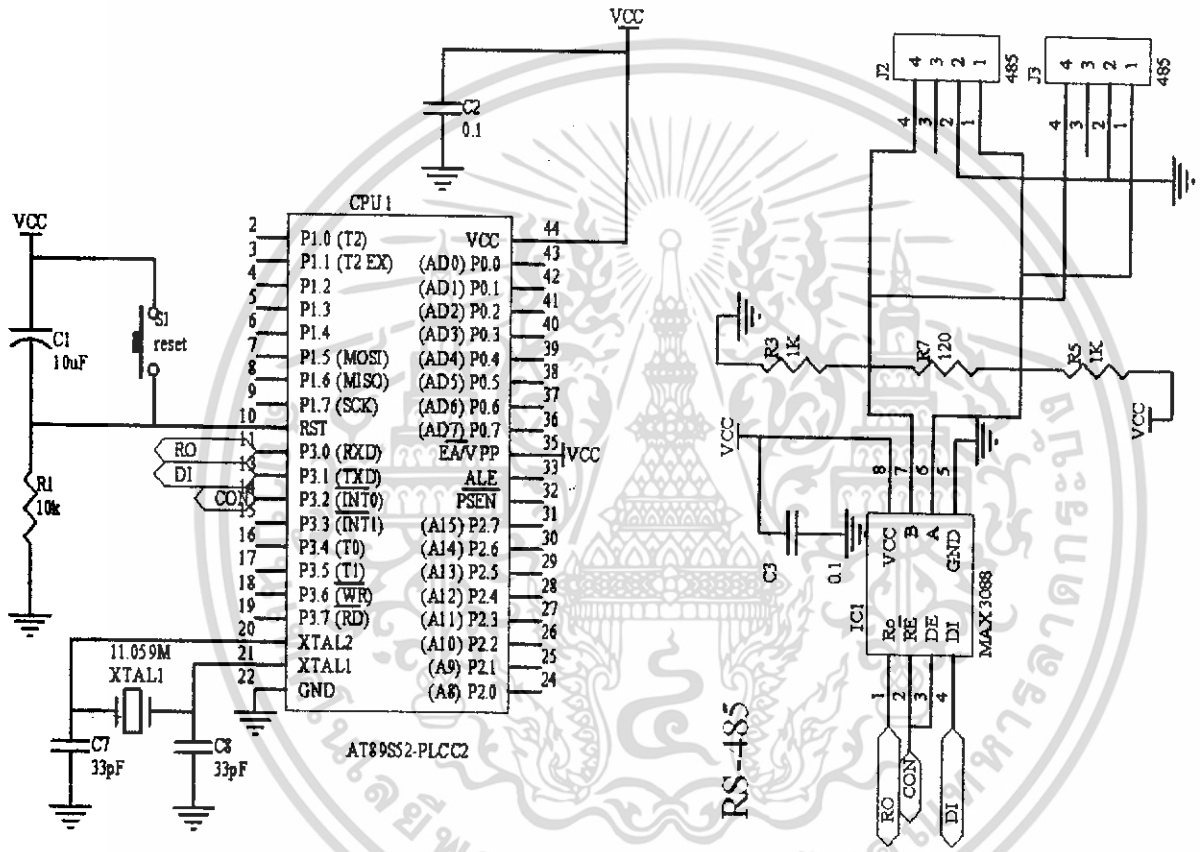


รูปที่ 3.2 แสดงวงจรแดปเตอร์

ในส่วนวงจรของตัวแปลงสัญญาณจะมีการเชื่อมต่อขาที่ทำหน้าที่ในการส่งและรับสัญญาณของพอร์ตสื่อสารแบบอนุกรรมเข้ากับขาของ IC MAX-232 ทำหน้าที่แปลงไฟ ระดับ 12 V เป็นระดับ TTL เพื่อให้ใช้ได้กับ IC DS 75176B เพื่อแปลงจากมาตรฐาน RS-232 เป็นมาตรฐาน RS-485

3.3 วงจรไมโครคอนโทรลเลอร์

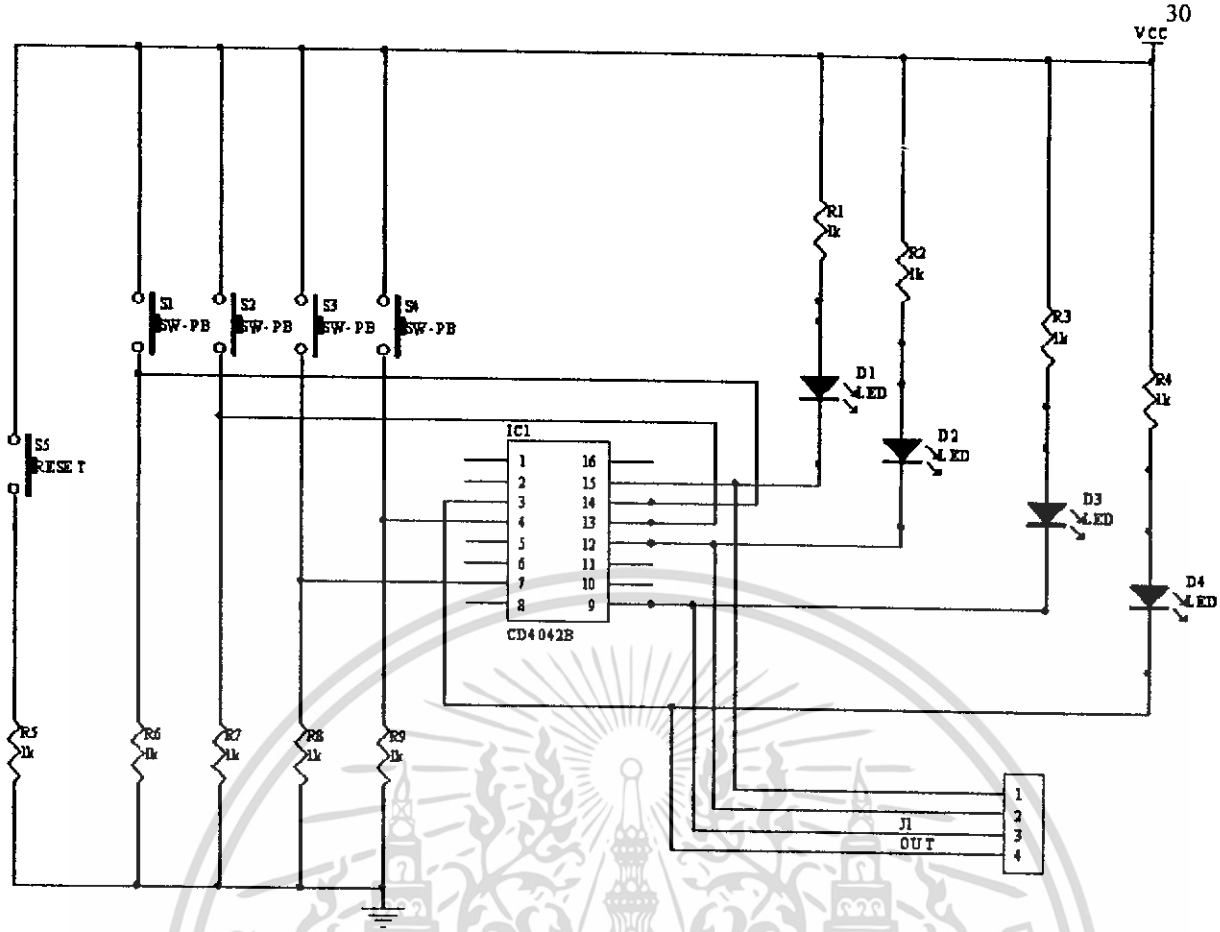
วงจรไมโครคอนโทรลเลอร์เป็นวงจรควบคุมการทำงานต่าง ๆ ของวงจรแต่ละชุดโดยจะต่ออยู่กับ RS-485 อแดปเตอร์เพื่อต่อกับ RS-485 อีกชุดหนึ่งที่ต่ออยู่กับ RS-232 โดยไมโครคอนโทรลเลอร์จะทำหน้าที่รับข้อมูลจากวงจรและส่งข้อมูลไปให้กับคอมพิวเตอร์โดยจะมีรูปวงจรดังนี้



รูปที่ 3.3 วงจรไมโครคอนโทรลเลอร์

3.4 วงจรสวิทช์

เป็นวงจรที่ทำให้เราทราบว่ามีผู้บุกรุกเข้ามาในบ้านในทางใด ถ้าสวิทช์ตัวถูกกด สมมติว่าเป็น S1 ถูกกดขา 4 จะมีสภาวะเป็น "High" และขา 3 จะมีสภาวะเป็น "Low" ทำให้ LED1 สว่างและถ้า S2 ถูกกดขา 7 จะมีสภาวะเป็น "High" และขา 9 จะมีสภาวะเป็น "Low" ทำให้ LED2 สว่าง และถ้า S3 ถูกกดขา 13 จะมีสภาวะเป็น "High" และขา 12 จะมีสภาวะเป็น "Low" ทำให้ LED3 สว่าง และถ้าสุดท้าย S4 ถูกกดขา 14 จะมีสภาวะเป็น "High" และขา 15 จะมีสภาวะเป็น "Low" ทำให้ LED4 สว่าง ตามลำดับ วงจรสวิทช์แสดงได้ดังรูป



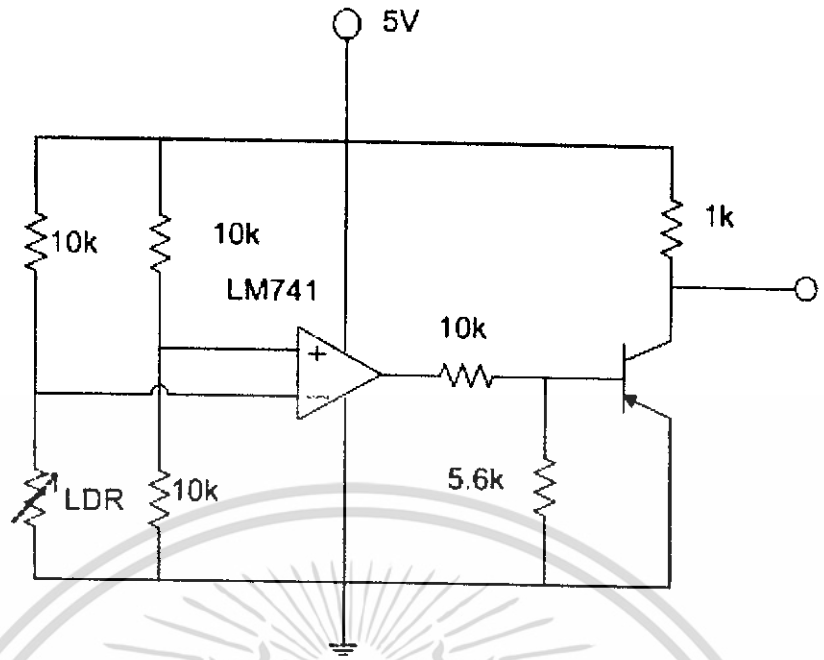
รูปที่ 3.4 วงจรสวิทช์

3.5 วงจรตรวจจับทางแสง

ในวงจรนี้จะใช้ตัวตรวจจับทางแสดงที่เรียกว่า Light Depentdent Resistance (LDR) กล่าวคือ ความต้านทานจะเปลี่ยนไปตามความเข้มของแสงนั้นเอง ถ้าความเข้มของแสงมากความต้านทานของมันจะต่ำ หากกลับกัน ความเข้มแสงน้อยความต้านทานจะสูง

ทำการวัดค่าความต้านทานของ LDR ในช่วงเวลาตามตารางจะได้ค่าดังนี้

ช่วงเวลาในการวัดความต้านทาน	ค่าความต้านทาน
กลางวัน	20 กิโลโอห์ม
กลางคืน	550 กิโลโอห์ม



รูปที่ 3.5 วงจรตรวจจับแสง

จากรูปที่ 3.5 เป็นวงจรที่ใช้ตรวจจับแสงโดยใช้ไอซี LM 741 ต่อเป็นวงจรเปรียบเทียบแรงดัน

ที่สถานะมืด ทรานซิสเตอร์ไม่ทำงาน ทำให้ O/P เท่ากับ 5 Volt

ที่สถานะสว่าง ทรานซิสเตอร์ทำงาน ทำให้ O/P เท่ากับ 0 Volt

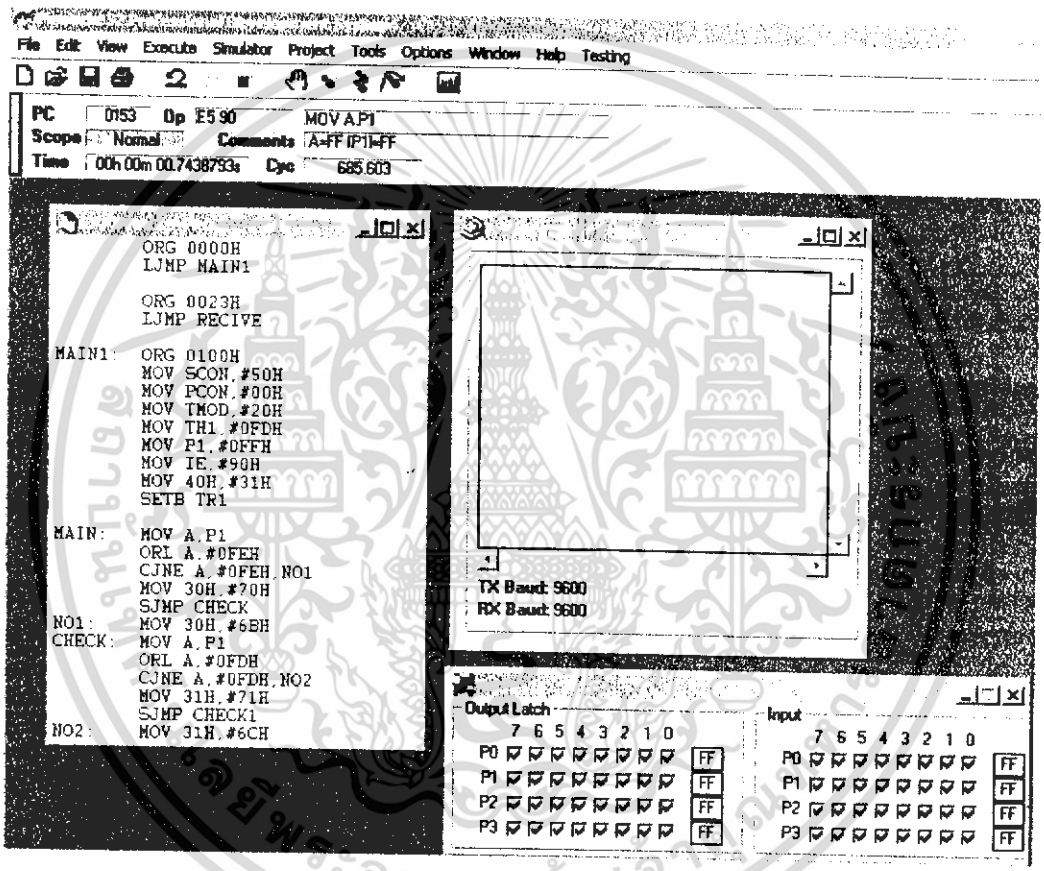
3.6 การคำนวณและการสร้างส่วนซอร์ฟแวร์

ส่วนโปรแกรมควบคุมการทำงานไมโครคอนโทรลเลอร์

โดยขั้นตอนการสร้างจะใช้

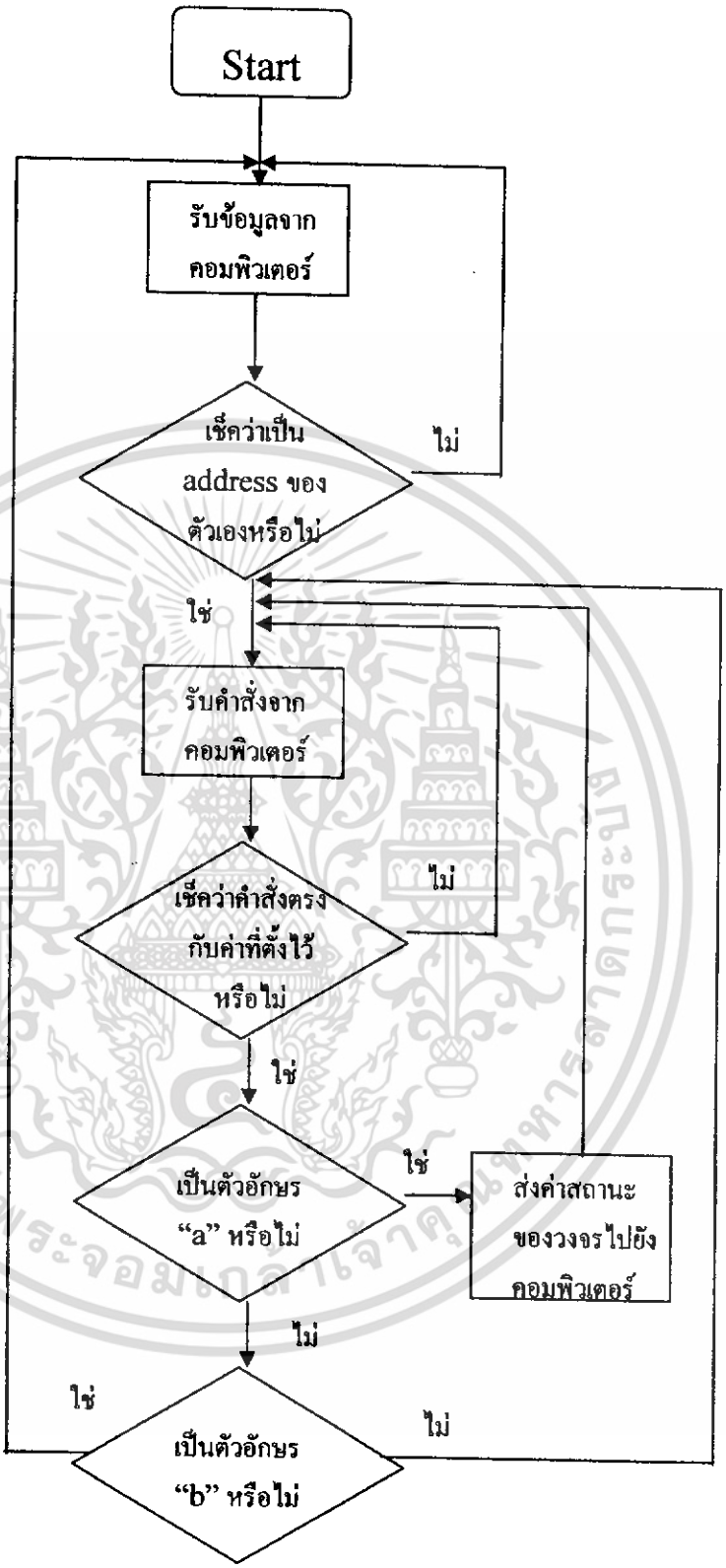
โปรแกรม pinnacle52

เป็นโปรแกรมจำลองการทำงานของไมโครคอนโทรลเลอร์โดยใช้ภาษา Assemble (.asm) ซึ่งจะดีกว่าโปรแกรม SXA51 เพราะสามารถจำลอง input ของ Hyper terminal , input และ output port ซึ่งง่ายแก่การทดสอบโปรแกรมดังรูปที่ 3.6



รูปที่ 3.6 แสดงโปรแกรม Pinnacle52

โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์โดยใช้ภาษา Assembly

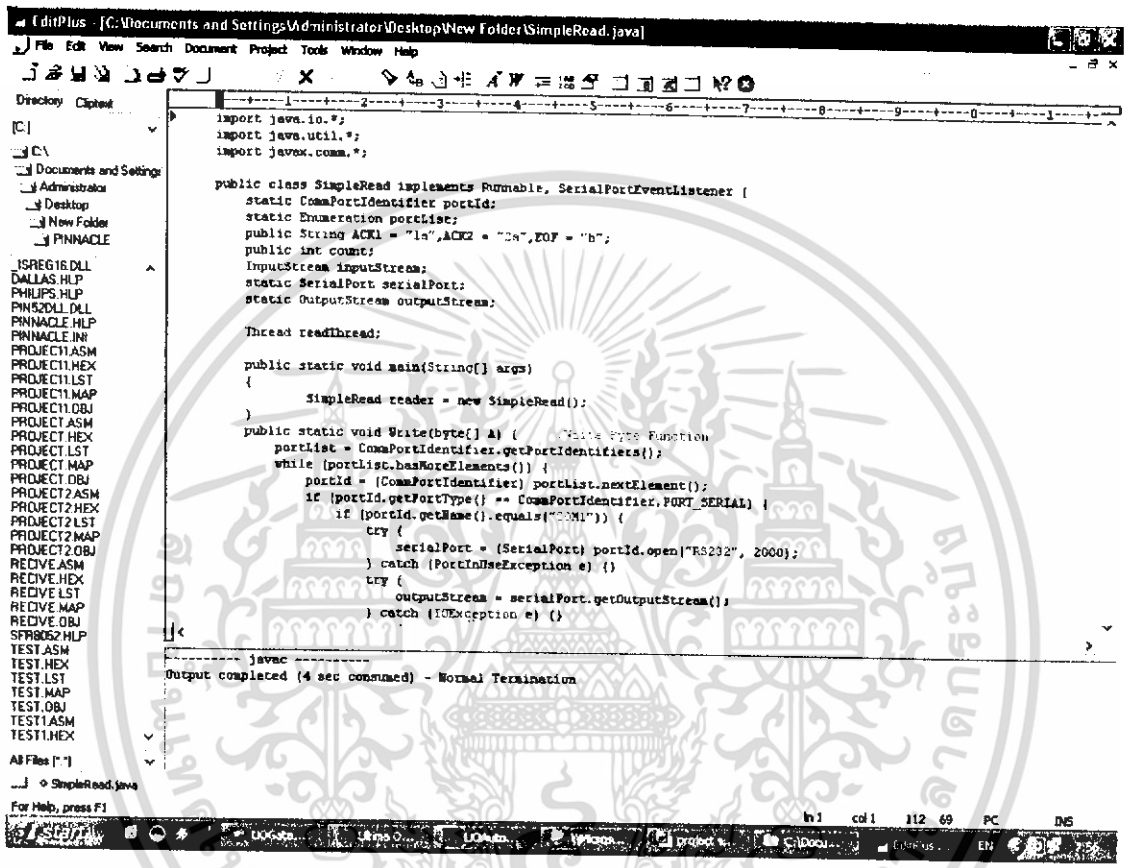


รูปที่ 3.7 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรม

ส่วนโปรแกรมควบคุมการทำงานของคอมพิวเตอร์

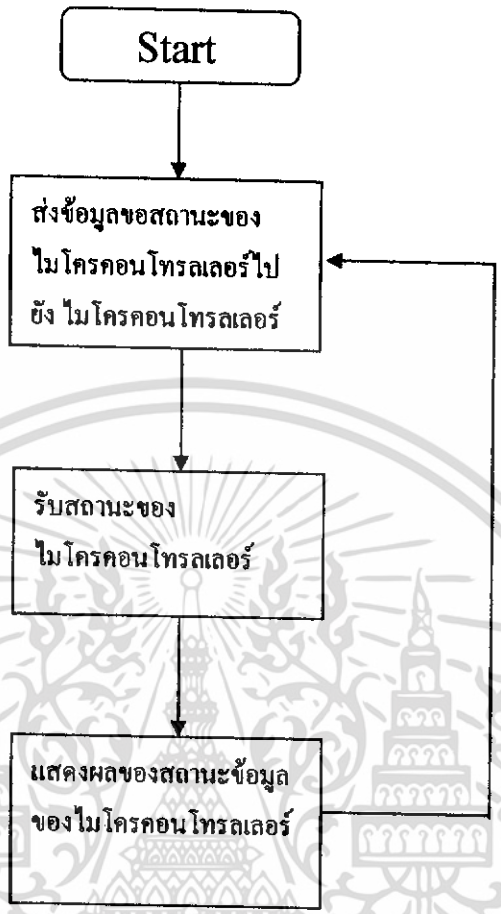
โดยขั้นตอนการสร้างจะใช้โปรแกรม Editplus

โปรแกรม Editplus เป็นโปรแกรมให้ความสะดวกสบายในการแก้ไขข้อมูลเนื่องจากมีสีแบ่งคำสั่งต่างๆ และยังสามารถ compile และ Run ภาษาจาวา



รูปที่ 3.8 แสดงโปรแกรม Editplus

โฟลว์ชาร์ตการทำงานของคอมพิวเตอร์โดยใช้ภาษาจาวา



รูปที่ 3.9 โฟลว์ชาร์ตแสดงการทำงานของคอมพิวเตอร์ด้วยจาวา

สารบัญ

สารบัญรูป

หน้า

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 วัตถุประสงค์

1. ศึกษาการส่งข้อมูลผ่านพอร์ตอนุกรม
2. ศึกษาการเชื่อมต่อระหว่างพอร์ต RS-232 กับ RS-485
3. ศึกษาการทำงานของวงจรไมโครคอนโทลเลอร์
4. ศึกษาการรับส่งข้อมูลผ่านพอร์ตอนุกรมและประมวลผลโดยใช้จาวา

1.2 แนวความคิดและที่มา

ในอดีต COMPUTER ทำงานโดยเก็บข้อมูลขนาดใหญ่ๆ อยู่ในตัวมันซึ่งเป็นการเก็บที่มีระเบียบแบบแผนค้นหาง่าย ต่อมา COMPUTER ก็ได้พัฒนา SOFTWARE และ HARDWARE ที่มีความสามารถขึ้นเรื่อยๆ ความสะดวกสบายนี้เอง COMPUTER จึงได้เข้ามามีบทบาทในชีวิตมากขึ้นและในปัจจุบันมีการใช้ COMPUTER ติดต่อสื่อสารทางอินเตอร์เน็ตซึ่งครอบคลุมได้ทั่วโลก คุณประโยชน์นี้เองจึงได้เป็นแนวทางในการประยุกต์ใช้อินเตอร์เน็ตรับส่งข้อมูลกับวงจรรักษาความปลอดภัย โดยโครงงานประกอบด้วย 2 ส่วนใหญ่ คือ

HARDWARE

อุปกรณ์ที่เชื่อมต่อกับ COMPUTER SERVER โดยผ่านพอร์ตสื่อสาร (Communication Port: Com Port) ของคอมพิวเตอร์ ซึ่งในที่นี้ใช้ DB-25 (COM Port 1) ในการเชื่อมต่อ โดยการสื่อสารของระบบควบคุมจะเป็นดังนี้คือ สัญญาณที่ออกจากพอร์ตสื่อสารจะถูกแปลงจากสัญญาณมาตรฐาน RS-232 ให้เป็นสัญญาณตามมาตรฐาน RS-485 โดย อแดปเตอร์ (Adapter) จากนั้นสัญญาณจะถูกส่งผ่านสายสัญญาณ (RS-485 BUS) ไปยัง HARDWARE

SOFTWARE

โปรแกรมรับส่งข้อมูลด้วยภาษา JAVA ไว้ที่เครื่อง computer โดยรับส่งข้อมูลผ่านทาง com port 1 และพอร์ต RS-232 จะถูกแปลงเป็นพอร์ต RS-485 เพื่อรับส่งข้อมูลที่แสดงสถานะควบคุมส่งกลับมายังผู้ใช้ และเมื่อมีปัญหาจะมีการส่ง mail เข้า E-mail ที่ได้กำหนดเอาไว้ เพื่อเตือนเจ้าของบ้าน

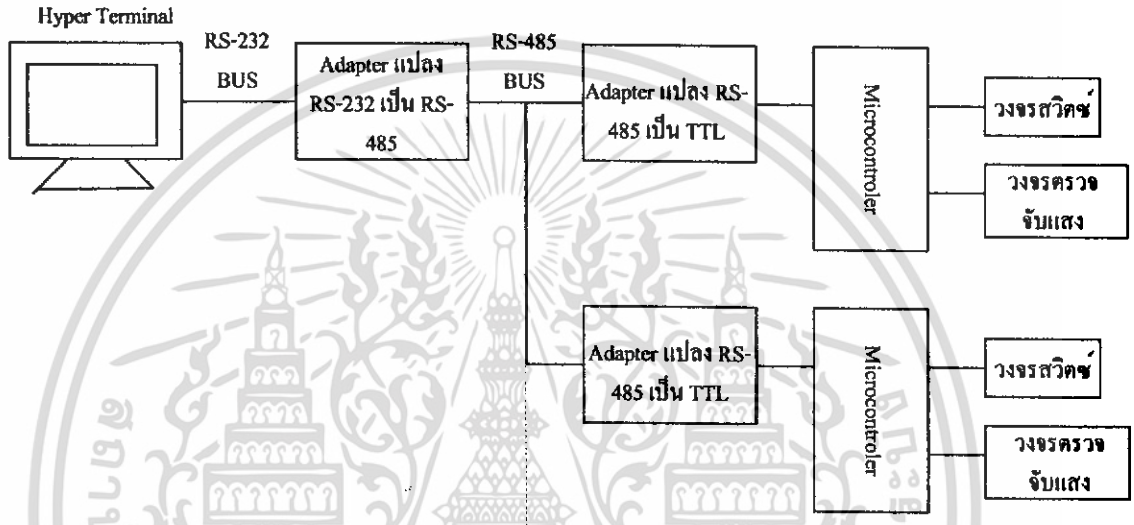
1.3 ขอบเขตของโครงงาน

โครงงานนี้ได้ใช้เวลาทำ 2 เทอม ในช่วงแรกจะทำทางด้าน Hardware ซึ่งประกอบด้วยวงจร RS-232 , RS-485 , วงจรสวิทช์ , วงจรตรวจจับแสง และวงจร microcontroller และในส่วนอีกเทอมหนึ่งจะทำทางด้าน Software ซึ่งใช้ภาษา Java ในการเขียน program ที่ทำการเชื่อมต่อระหว่าง microcontroller กับ computer โดยจะผ่าน RS-485 และโปรแกรมเตือนที่ส่งทาง mail เข้า E-mail ตามที่ได้ตั้งเอาไว้

บทที่ 2
ทฤษฎีและหลักการ

2.1 หลักการทำงาน

คอมพิวเตอร์จะทำการรับ-ส่ง ข้อมูลโดยผ่าน Hyper terminal ผ่านพอร์ต RS-232 (com1) ไปยัง adapter เพื่อทำการเปลี่ยนเป็นการส่งแบบ RS-485 เมื่อข้อมูลมาถึงไมโครคอนโทรเลอร์โดย ไมโครคอนโทรเลอร์จะทำการเก็บข้อมูลสถานะของวงจร เช่น วงจรสวิทช์และวงจรตรวจจับแสงและส่ง สถานะต่างๆของวงจรกลับไปยังคอมพิวเตอร์เป็นดังรูปที่ 2.1 ซึ่งหลักการต่างๆจะกล่าวถัดไปจากนี้



รูปที่ 2.1 แผนภาพแสดงการทำงานของโครงงาน

2.2 ลักษณะของการสื่อสาร

ในการสื่อสารหรือการส่งข้อมูลโดยใช้คอมพิวเตอร์ส่วนบุคคลนั้นมีรูปแบบในการสื่อสารที่ สำคัญอยู่ 2 แบบคือ

1.การสื่อสารแบบขนาน

2.การสื่อสารแบบอนุกรม

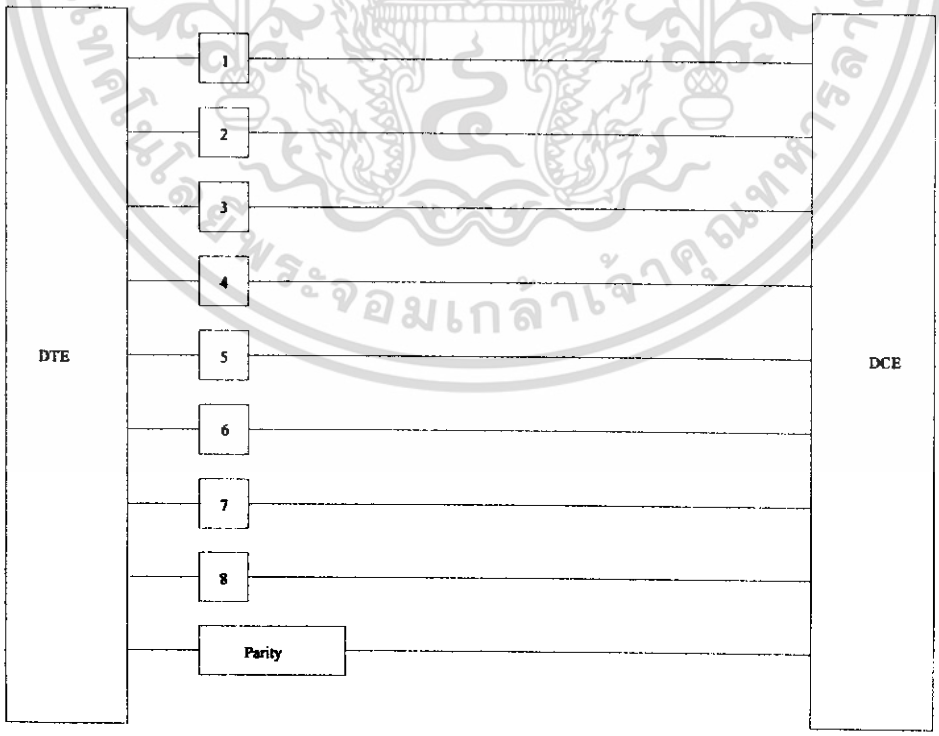
แต่ก่อนที่จะทำความรู้จักกับการสื่อสารทั้งสองควรจะมีความเข้าใจเกี่ยวกับคำสั่ง หรือข้อมูลที่อยู่ ในรูปของบิต ซึ่งประกอบด้วยหลายบิตมาประกอบกัน ข้อมูลในการสื่อสารแต่ละข้อมูลจะถูกแปลงให้อยู่ ในรูปแบบของเลขฐานสอง แล้วนำมาประกอบกัน เช่น ถ้าข้อมูลที่ประกอบด้วย 4 บิต เราจะเรียกว่า 1 นิบเบิล หรือถ้าหากข้อมูลที่ประกอบด้วย 8 บิต เราจะเรียกว่า 1 ไบต์ เป็นต้น

การสื่อสารแบบขนาน

การสื่อสารแบบขนานจะมีรูปแบบการส่งข้อมูลครั้งละ 1 ไบต์ก็คือจะทำการส่งข้อมูลครั้งละ 8 บิตนั่นเอง ซึ่งในการส่งต้องใช้สายไฟในการส่งข้อมูล 8 เส้น แล้วยังต้องใช้สายไฟอีก 1 เส้นในการควบคุมเช่นใช้เป็นพาริตี้บิต หรืออาจจะมีมากกว่านั้นเพื่อใช้ในการควบคุมการโต้ตอบของการทำงาน (Hand-shake) ซึ่งรายละเอียดจะบอกให้ทราบต่อไปในเรื่องของ Hand-shake จึงสรุปได้ว่าในการสื่อสารแบบขนานนั้นต้องใช้สายอย่างน้อยที่สุด 9 เส้น

ดังนั้นในการส่งข้อมูลทีละ 1 ไบต์นั้นทำให้ข้อมูลทั้ง 8 บิตมาถึงปลายทางพร้อมกัน ทำให้ข้อมูลแบบขนานสามารถทำได้ด้วยความเร็วที่สูงมาก แต่ปัญหาที่สำคัญของการส่งข้อมูลแบบขนาน คือคุณสมบัติของบิตกับแรงดัน เวลาที่บิตหรือแรงดันไฟฟ้าที่ใช้ในการเปลี่ยนสถานะจากหนึ่งเป็นศูนย์นั้นสั้นมาก โดยเร็วถึงระดับนาโนวินาที (หนึ่งในพันล้านของวินาที) การเปลี่ยนแปลงที่รวดเร็วนี้เป็นส่วนที่สำคัญมากต่อการส่งข้อมูล เพราะการเปลี่ยนแปลงระหว่างศูนย์และหนึ่งอย่างช้าๆจะไม่ถูกอ่านเป็นข้อมูลเลย และเมื่อสายไฟที่ส่งข้อมูลยาวขึ้น คุณสมบัติทางไฟฟ้าของสายไฟเช่นค่าความจุไฟฟ้าและค่าความเหนี่ยวนำจะจำกัดความเร็วในการเปลี่ยนแปลงระหว่างศูนย์และหนึ่งของบิต ซึ่งจะทำให้ข้อมูลอาจสูญหายหรือทำให้การส่งข้อมูลล้มเหลวได้ ดังนั้นการส่งข้อมูลบนสายยาวอาจจะเป็นปัญหาได้หากใช้วิธีการสื่อสารแบบขนาน

เนื่องจากข้อเสียของการส่งข้อมูลแบบขนานสองอย่างคือ ค่าใช้จ่ายที่สูงและการสูญหายของข้อมูลทำให้การใช้งานของมันถูกจำกัดอยู่กับอุปกรณ์เพียงไม่กี่ชนิด เช่น เครื่องพิมพ์ที่มักจะอยู่ใกล้กับเครื่องคอมพิวเตอร์และต้องทำงานที่ความเร็วสูง แต่เรายังใช้วิธีการส่งข้อมูลแบบขนานนี้ภายในเครื่องคอมพิวเตอร์ เนื่องจากไม่ต้องใช้สายไฟขนาดยาว



รูปที่ 2.2 แสดงการสื่อสารแบบขนาน

การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมเป็นการส่งข้อมูลทีละบิต และข้อมูลจะถูกต่อรวมเข้าเป็นไบต์ใหม่ด้วยวิธีการส่งข้อมูลทีละบิตนี้ ทำให้สามารถใช้สายไฟเพียงสองเส้นในการส่งข้อมูล ซึ่งช่วยให้เราประหยัดค่าสายไฟไปได้มาก แต่ก็ทำให้ประสิทธิภาพการทำงานลดลงไปด้วย เพราะการส่งข้อมูลวิธีนี้ต้องใช้เวลานานขึ้นอย่างน้อยแปดเท่าของการส่งข้อมูลแบบขนาน แต่ความเร็วที่ลดลงไปนี้ยังไม่ถือว่าเป็นข้อจำกัดที่สำคัญทางการใช้งานนัก เพราะหากว่าเราพิจารณาที่อุปกรณ์ทั่วไป จะพบว่าอุปกรณ์ส่วนใหญ่ที่ทำงานช้ามากเมื่อเทียบกับความเร็วในการทำงานในไมโครโปรเซสเซอร์ อุปกรณ์แต่ละตัวมีขั้นตอนการทำงานที่กินเวลานาน ซึ่งโดยทั่วไปมักจะเป็นกระบวนการทางกลไก (mechanic) ที่เป็นตัวจำกัดความเร็วของเครื่องลงไปอย่างมาก

ตัวอย่างเช่น ความเร็วของเครื่องพิมพ์ถูกจำกัดที่ความเร็วของหัวพิมพ์ (print head) ความเร็วของโมเด็มถูกจำกัด โดยขีดจำกัดความถี่ของสายโทรศัพท์ และความเร็วของดิสก์ไดร์ฟถูกจำกัดโดยอัตราเร็วการหมุนของไดร์ฟ ดังนั้นความเร็วที่ได้มาจากการส่งข้อมูลแบบขนานจะเสียไปโดยเปล่าประโยชน์เมื่อนำมาใช้งานได้ แม้ว่าอัตราเร็วการส่งข้อมูลจะลดลงแต่ก็ยังสามารถใช้งานร่วมกับอุปกรณ์ประเภทนี้ได้อย่างมีประสิทธิภาพ ข้อเสียจากความเร็วที่ลดลงนั้นไม่อาจเทียบได้กับผลพวงที่ได้จากคุณภาพการส่งและระยะทางการส่งข้อมูลที่เพิ่มขึ้น

ตารางที่ 2.1 ตารางเปรียบเทียบการทำงานระหว่างการสื่อสารข้อมูลแบบขนานและแบบอนุกรม

การใช้งาน	แบบขนาน	แบบอนุกรม
1.ระยะทาง	จะใช้งานได้ระยะไม่เกิน 100 ฟุต	จะสามารถใช้งานได้ตั้งแต่ในระยะใกล้ๆไปจะถึงระยะทางที่มากๆจนถึงหลักไมล์
2.ความเร็ว	อัตราความเร็วสูงมากในระยะที่ไม่ไกลมากนักกำหนดได้เป็นจำนวนบิตต่อวินาที	อัตราเร็วของข้อมูลที่ใช้กันอยู่ทั่วไปจะอยู่ในช่วง 0-2 ล้านบิตต่อวินาที
3.ระดับของสัญญาณ	ในการอินเตอร์เฟสจะใช้ระดับ TTL คือสัญญาณลอจิก 1 และ 0 จะแทนด้วยระดับแรงดัน +5V และ 0V	ในมาตรฐานของ EIA-RS-232c ระบุว่ามีระดับสัญญาณไฟฟ้าขนาน 12V หรือใช้มาตรฐาน 20 mV current loop
4.ความผิดพลาดของสัญญาณ	ในการใช้งานระยะไกลๆ ความผิดพลาดของข้อมูลจะเกิดขึ้นได้ง่าย	การใช้งานจะเกิดการผิดพลาดของสัญญาณจะมีน้อยมากและสิ้นเปลืองน้อยกว่ามาก

การใช้งาน	แบบขนาน	แบบอนุกรม
5. ค่าใช้จ่าย	ค่าใช้จ่ายจะสูงมากเพราะจะต้องใช้สายส่งสัญญาณหลายเส้น โดยเฉพาะการส่งในระยะทางไกลๆ	สิ้นเปลืองน้อยกว่ามากถึงแม้ว่าจะต้องใช้อุปกรณ์เปลี่ยนสัญญาณของข้อมูลจากขนานไปเป็นอนุกรมแล้วส่งผ่านสายส่ง แล้วกลับสัญญาณมาเป็นขนานอีกครั้งก็ตาม

2.3 ลักษณะสัญญาณที่ใช้ในการอินเตอร์เฟสตามมาตรฐานต่างๆ

การส่งข้อมูลอยู่ภายในเครื่องคอมพิวเตอร์สามารถกระทำได้ง่าย เนื่องจากเราสามารถคาดเดาสภาพแวดล้อมภายในเครื่องได้ แต่ในการส่งข้อมูลสู่ภายนอก เราไม่สามารถคาดการณ์ได้ว่าเครื่องคอมพิวเตอร์และตัวข้อมูลจะต้องพบกับสภาพเช่นไรและจะมีผลกระทบต่อตัวข้อมูลและเครื่องคอมพิวเตอร์อย่างไร ดังนั้นในการออกแบบวงจรจึงมีสิ่งที่ต้องพิจารณา คือ จะต้องหาวิธีการในการแยกข้อมูลออกจาสภาพแวดล้อมและสัญญาณรบกวนโดยเฉพาะอย่างยิ่งในกรณีที่มีสายส่งข้อมูลขนาดยาว สิ่งที่จะต้องพิจารณาอีกสิ่งหนึ่งก็คือ จะต้องหาวิธีการป้องกันคอมพิวเตอร์จากสภาพแวดล้อมที่ไม่พึงประสงค์ด้วย นั้นก็คือจะต้องมีตัวอินเตอร์เฟส ซึ่งจะมีหน้าที่เป็นจุดเชื่อมต่อระหว่างคอมพิวเตอร์กับสิ่งแวดล้อมภายนอก และเนื่องจากอุปกรณ์อินเตอร์เฟสมีหน้าที่คล้ายกับเป็นประตูของเครื่องคอมพิวเตอร์ บางครั้งมันจึงถูกเรียกว่า I/O พอร์ต (I/O Port) หรือบางครั้งเรียกสั้นๆว่าพอร์ต (PORT)

วัตถุประสงค์หลักของการอินเตอร์เฟสก็คือ การใช้อุปกรณ์อินเตอร์เฟสเป็นสื่อกลางของการส่งข้อมูล และวัตถุประสงค์ที่สำคัญอีกอย่างหนึ่งของการอินเตอร์เฟสก็คือ ความง่ายต่อการใช้และเมื่อเราสามารถทำการอินเตอร์เฟสได้สำเร็จ ก็จะสามารถที่จะส่งข้อมูลสู่ภายนอกได้ โดยในโครงงานได้ใช้มาตรฐานการอินเตอร์เฟส 2 แบบ คือ มาตรฐานการอินเตอร์เฟสแบบ RS-232 และมาตรฐานการอินเตอร์เฟสแบบ RS-485

มาตรฐานการอินเตอร์เฟส RS-232

มาตรฐานการอินเตอร์เฟส RS-232 ได้เกิดขึ้นตั้งแต่ปี 1969 EIA (Electronic Industries Association) ห้องวิจัย Bell และบรรดาผู้ผลิตอุปกรณ์สื่อสารได้ร่วมกันจัดมาตรฐาน EIA RS-232 ซึ่งต่อมาไม่นานนักก็ได้มีการปรับปรุงแก้ไขอีกเล็กน้อยกลายเป็น RS-232 C และเมื่อไม่นานมานี้ก็ได้ออกมาตรฐาน RS-232 D

เพื่อทำความเข้าใจกับวัตถุประสงค์หลักของ RS-232 ได้ดียิ่งขึ้น ควรทำความเข้าใจกับวัตถุประสงค์หลักของ RS-232 ก่อน ซึ่งได้แสดงไว้อย่างชัดเจนในหัวข้อของเอกสารคือ

Interface between Data Terminal Equipment and Data Communications Equipment Employing Serial Binary Data Interchange (การอินเตอร์เฟสระหว่างอุปกรณ์เทอร์มินัล และ อุปกรณ์สื่อสารข้อมูลที่ใช้วิธีการแลกเปลี่ยนข้อมูลไบนารีแบบอนุกรม)

ซึ่งในหัวข้อสำคัญจะอธิบายถึงการอินเตอร์เฟสระหว่างเทอร์มินัล (Data Terminal Equipment หรือ DTE) กับโมเด็ม (Data Communications Equipment หรือ DCE) เพื่อใช้ส่งข้อมูลแบบอนุกรม โดยจะประกอบด้วย 4 ส่วนคือ

- คุณสมบัติทางไฟฟ้าของสัญญาณ (Electrical Signal Characteristics) ซึ่งจะอธิบายถึงรูปแบบของสัญญาณไฟฟ้าที่ตัวอินเตอร์เฟสจะส่งออก และรับเข้ามาจากภายนอกระดับแรงดันไฟฟ้าที่แสดงถึงตรรกะ 0 และ 1 ก็จะมีกำหนดไว้ในส่วนนี้ด้วย

- คุณสมบัติทางกลไกการอินเตอร์เฟส : คอนเน็กเตอร์ (Interface Mechanical Characteristics Connectors) ซึ่งเป็นข้อกำหนดเกี่ยวกับตัวอินเตอร์เฟส ประกอบด้วย ส่วนที่เป็นปลั๊ก (plug) และเต้าเสียบ (receptacle) โดยเต้าเสียบจะอยู่บน DCE สำหรับ RS-232 A-C ไม่ได้มีการกำหนดคอนเน็กเตอร์รูปตัว D (D-Shaped) ซึ่งมีใช้กันอยู่ทั่วไปนั้น ทั้งนี้เพราะว่าอุปกรณ์ตัวนี้ได้รับการคุ้มครองโดยลิขสิทธิ์ และเมื่อสิทธิบัตรนั้นหมดอายุลงใน RS-232 D จึงได้เพิ่มข้อกำหนดคอนเน็กเตอร์ DB-25 เข้าไว้ในมาตรฐานด้วย

- หน้าที่การทำงานของวงจรการแลกเปลี่ยน (Functional Description of Interchange Circuit) ในส่วนนี้กำหนดหน้าที่และตั้งชื่อให้กับสัญญาณไฟฟ้าต่างๆที่นำมาใช้ เช่น Transmitted Data (ข้อมูลส่งออก) ได้ถูกกำหนดไว้ให้กับขา 2

- มาตรฐานการอินเตอร์เฟสสำหรับระบบการสื่อสารเฉพาะอย่าง (Standard Interfaces for Selected Communications System Configurations) ในส่วนนี้เป็นรายละเอียดต่างๆ สำหรับการติดต่อระหว่างโมเด็มกับเทอร์มินัลทั่วไป

พื้นฐานการอินเตอร์เฟส RS-232

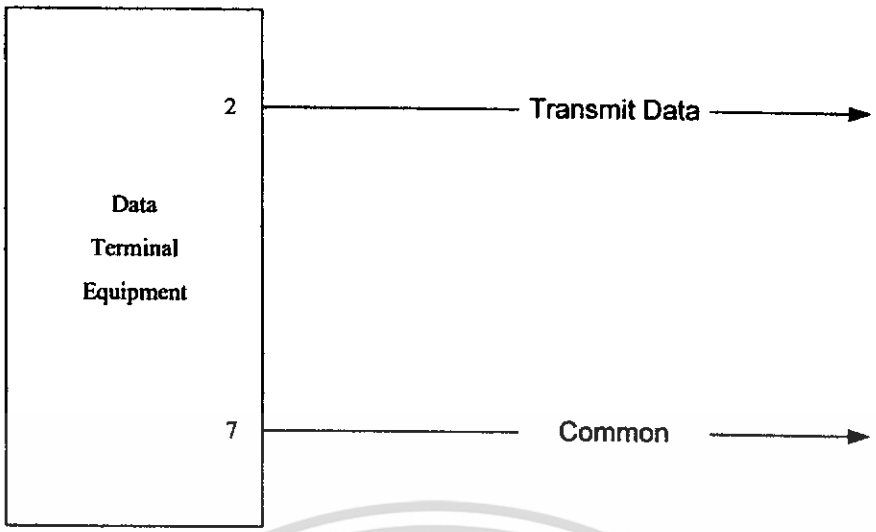
ก่อนที่จะทำความรู้จักกับพื้นฐานการอินเตอร์เฟส RS-232 ก่อนอื่นต้องทำความเข้าใจกับคำว่า DTE และ DCE ก่อน

DTE = Data Terminal Equipment ซึ่งก็คือ คอมพิวเตอร์ นั่นเอง (ตัวส่งข้อมูล)

DCE = Data Communications Equipment อุปกรณ์เหล่านี้ ได้แก่ โมเด็ม ,TA อแดปเตอร์พล็อตเตอร์ เป็นต้น (ตัวรับข้อมูล)

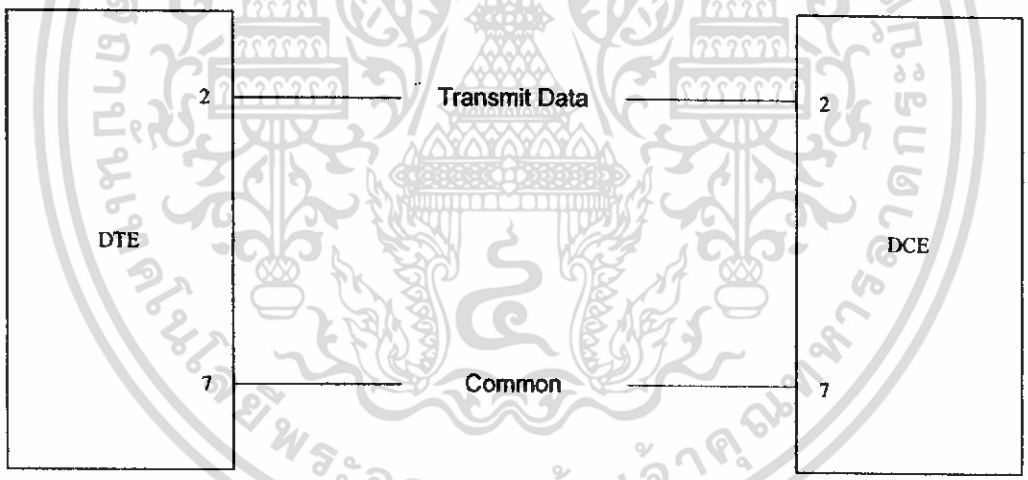
ตามมาตรฐานของ EIA ได้กำหนดไว้ว่า อุปกรณ์ DTE หมายถึงอุปกรณ์ที่ข้อมูลมาสิ้นสุดและอุปกรณ์ DCE เป็นอุปกรณ์ที่ใช้ส่งผ่านข้อมูล ดังนั้นคอมพิวเตอร์ซึ่งสามารถทำได้ทั้งส่งผ่านข้อมูลและรับข้อมูล จึงไม่อาจระบุได้ว่าเป็นอุปกรณ์ DCE หรือ DTE

ในการพิจารณาโครงสร้างเบื้องต้นของการอินเตอร์เฟส RS-232 นั้นจะประกอบด้วยเส้นสายไฟเพียง 2 เส้น เส้นหนึ่งใช้ในการส่งข้อมูล และอีกเส้นหนึ่งสำหรับอ้างอิงแรงดันของวงจรอินเตอร์เฟส (circuit common) ซึ่งมักจะมีความเข้าใจผิดว่าคือกราวด์ (ground) แต่แท้จริงแล้วไม่ใช่ ดังรูปที่ 2.3



รูปที่ 2.3 แสดงอุปกรณ์ DTE เบื้องต้น

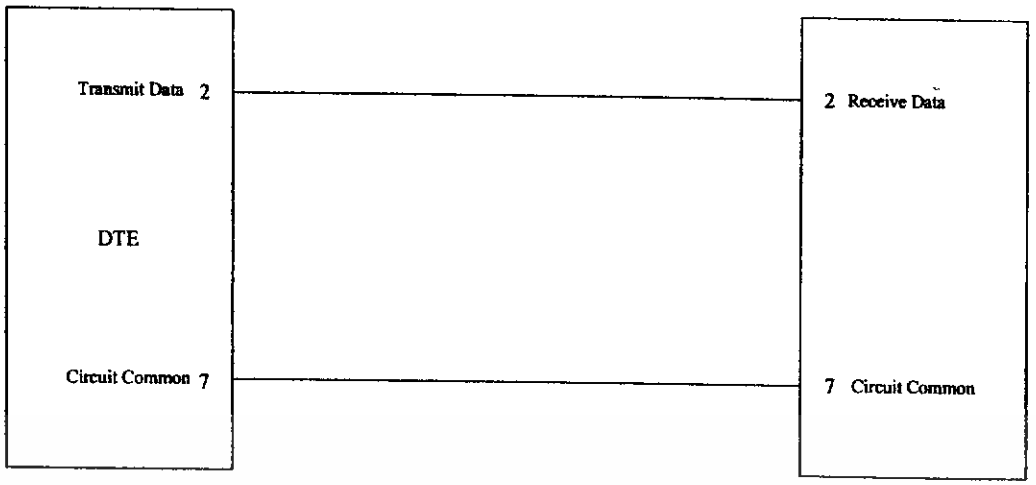
จากรูปที่ 2.3 เราจะพบว่ามีเพียงตัวส่งข้อมูล แต่ในความจริงแล้วในการอินเตอร์เฟสข้อมูลใดๆ จะต้องมีส่วนที่รับข้อมูลด้วย ซึ่งก็คือ DCE นั่นเอง ดังรูป2.4



รูปที่ 2.4 แสดงอุปกรณ์ทั้ง DTE และ DCE

เมื่อพิจารณารูปที่ 2.4 จะพบว่าข้อมูลที่ถูกส่งออกจากขา 2 ของ DTE จะรับเข้าไปยังขา 2 ของ DCE เช่นกัน โดยข้อมูลที่รับทาง DCE แน่นอนจะต้องเป็นข้อมูลเดียวกันที่ส่งออกมาจาก DTE ที่ให้สรุปได้ว่า "Transmit Data" มิได้มีส่วนในการกำหนดว่าอุปกรณ์ใดเป็นต้นทางหรือปลายทาง แต่จะขึ้นอยู่กับว่าจะพิจารณาเช่นไร

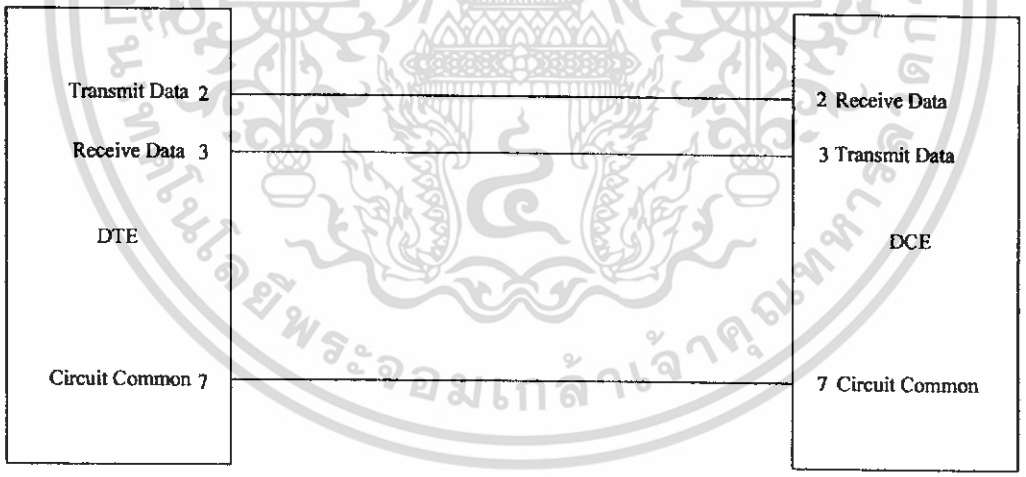
ในที่นี้จะใช้การพิจารณาดังนี้คือ สัญญาณที่ส่งออกไปเรียกว่า "เอาต์พุต" และสัญญาณที่รับเข้ามาเรียกว่า "อินพุท" โดยถือว่าสัญญาณเป็นกิจกรรมทางไฟฟ้าที่เกิดขึ้นในกระบวนการอินเตอร์เฟส ดังรูปที่ 2.5



รูปที่ 2.5 แสดงอุปกรณ์ DTE และ DCE ซึ่งเป็นคู่อุปกรณ์ที่ทำงานตรงข้ามกัน

การรับส่งข้อมูลสองทิศทาง

เมื่อพิจารณารูปที่ 2.5 แล้วเราจะพบว่าเมื่ออุปกรณ์ DTE รับข้อมูลแล้วจะต้องส่งผ่านไปยังอุปกรณ์ DCE (โมเด็ม) แล้วโมเด็มก็จะทำหน้าที่ในการส่งข้อมูลต่อออกไปยังสายโทรศัพท์ ซึ่งจะเป็นได้ว่าอุปกรณ์ DTE และอุปกรณ์ DCE สามารถเป็นได้ทั้งอุปกรณ์ส่งและรับข้อมูล และยังสามารถส่งและรับข้อมูลในทิศทางตรงข้ามได้อีกด้วย ดังรูปที่ 2.6



รูปที่ 2.6 แสดงการทำงานของอุปกรณ์ที่สามารถส่งและรับข้อมูลได้ทั้งสองทิศทาง

ซึ่งจากรูปที่ 2.6 ก็คงจะพอสรุปความแตกต่างระหว่างอุปกรณ์ DTE และ DCE ได้ดังนี้

DTE ส่งเอาต์พุตที่ขา 2 และรับอินพุทที่ขา 3

DCE ส่งเอาต์พุตที่ขา 3 และรับอินพุทที่ขา 2

โดยในการอินเตอร์เฟสนั้นจะต้องทำการตรวจสอบทิศทางของสัญญาณข้อมูลที่ขา 2 และที่ขา 3 ก่อนเสมอ

การแฮนด์เช็คใน RS-232

การแฮนด์เช็คหมายถึง กระบวนการที่อุปกรณ์หนึ่งใช้ตรวจสอบสถานะของอีกอุปกรณ์ที่ต่อเข้าด้วยกัน และตอบสนอง สถานะนั้นอย่างเหมาะสมและถูกจังหวะเวลา ซึ่งก็คือวิธีการควบคุมการทำงานของอุปกรณ์สองตัวให้สัมพันธ์กันในการรับส่งข้อมูลนั้นเอง การแฮนด์เช็คนั้นแบ่งออกเป็น 2 ประเภทคือ การแฮนด์เช็คทางฮาร์ดแวร์ และการแฮนด์เช็คทางซอฟต์แวร์

การแฮนด์เช็คทางซอฟต์แวร์ (software handshaking) เป็นวิธีการหนึ่ง ในการควบคุมการทำงานของอุปกรณ์รับข้อมูลโดยส่งผ่านสัญญาณควบคุมไปพร้อมกับตัวข้อมูลที่ต้องการส่ง

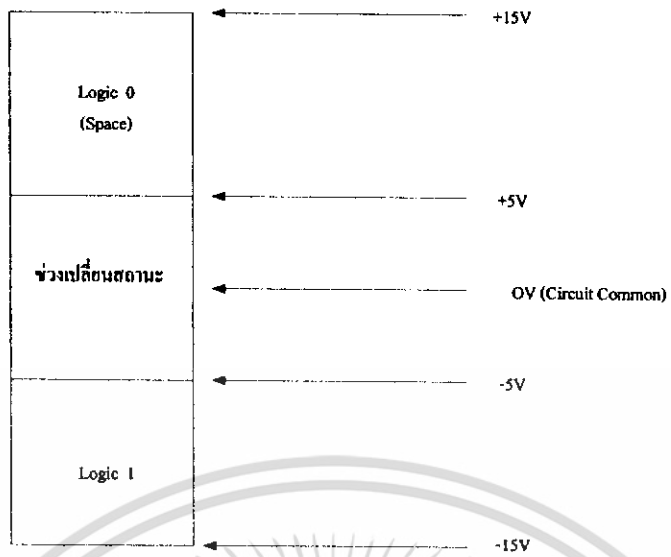
การแฮนด์เช็คทางฮาร์ดแวร์ (hardware handshaking) สามารถควบคุมได้ตั้งแต่ระดับฮาร์ดแวร์ โดยการเปลี่ยนระดับแรงดันในสายสัญญาณควบคุมเป็นตัวระงับไม่ให้คอมพิวเตอร์ส่งข้อมูลเพิ่มเข้ามาอีก ซึ่งเป็นการหลีกเลี่ยงใช้รหัสหรือโปรแกรม แต่การแฮนด์เช็คทางฮาร์ดแวร์นั้นมีข้อจำกัด คือจำเป็นต้องมีสายสัญญาณควบคุมต่างหากโดยเฉพาะ ทำให้วิธีนี้ไม่เหมาะที่จะนำมาใช้ในการอินเตอร์เฟสกับโมเด็ม

ขอบเขตความคอมแพติเบิล (Compatible) กับ RS-232

คุณสมบัติทางไฟฟ้า (ระดับแรงดัน ฯลฯ) ของการอินเตอร์เฟสได้รับการตรวจรับรอง ถ้าอุปกรณ์นั้นถูกอ้างว่าคอมแพติเบิลกับ RS-232 ย่อมหมายความว่าเราสามารถนำอุปกรณ์นั้นไปติดต่อกับอุปกรณ์อื่นที่คอมแพติเบิลกับ RS-232 ได้ โดยไม่ทำให้เกิดความเสียหายแก่อุปกรณ์ทั้งคู่ ซึ่งเงื่อนไขนี้ช่วยให้เรามั่นใจได้ว่าอุปกรณ์ทั้งคู่มีระบบไฟฟ้าที่ทำงานด้วยกันได้ โดยไม่ทำให้เกิดความบกพร่อง ในการรับส่งข้อมูล ระดับแรงดันสำหรับค่า "ศูนย์" และ "หนึ่ง" ต้องเป็นไปตามที่ระบุไว้ในมาตรฐาน

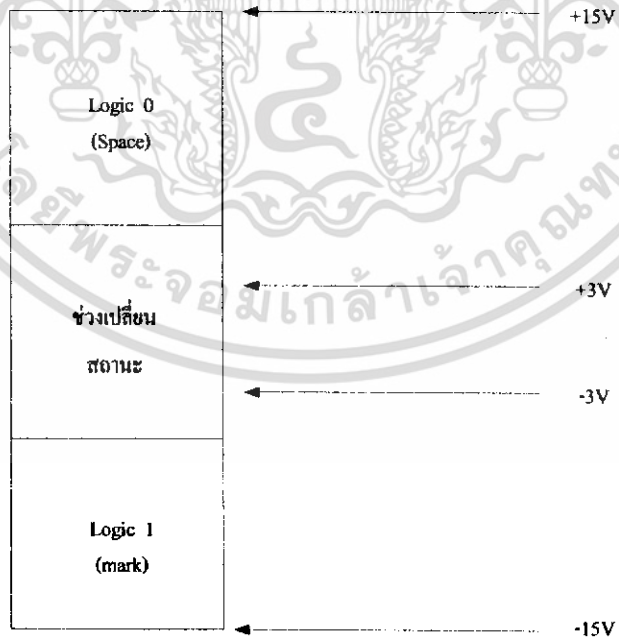
มาตรฐานระดับตรรกะ (Logic) ใน RS-232

การส่งข้อมูลจากวงจรอินเตอร์เฟสมีลักษณ์ "กลับหัว" กับวงจรที่ใช้งานอยู่ทั่วไป โดยความสัมพันธ์ระหว่างแรงดันกับค่าตรรกะบนอินเตอร์เฟสคือ แรงดันบวกอินเตอร์เฟสจะถูกแทนด้วย 0 ในขณะที่แรงดันลบแทนด้วยค่า 1 ดังรูปที่ 2.7



รูปที่ 2.7 แสดงคำจำกัดความค่าตรรกะที่เอาต์พุตของ RS-232

ขอให้สังเกตค่าตรรกะที่กลับกันให้ดีคือ แรงดันลบแทนด้วยค่า 1 และแรงดันบวกแทนด้วยค่า 0 เพื่อให้แน่ใจค่าตรรกะ 0 แรงดันไฟที่เอาต์พุตจะต้องอยู่ในช่วง +5V ถึง +15V และในทำนองเดียวกันในการแทนระดับ 1 ระดับแรงดันที่เอาต์พุตจะต้องอยู่ในช่วง -5V ถึง -15V สำหรับช่องว่างหรือ dead-band ที่อยู่ในช่วง +5V ถึง -5V มีชื่อว่าเรียกว่า ช่วงเปลี่ยนสถานะ (transition region) เป็นที่ไม่สามารถกำหนดค่าตรรกะได้ ซึ่งหมายความว่าค่าแรงดันเอาต์พุตในช่วง +5V ถึง -5V นั้นอาจถูกแปลความหมายให้เป็น 0 หรือ 1 ก็ได้ ดังรูปที่ 2.8



รูปที่ 2.8 แสดงคำจำกัดความค่าตรรกะที่อินพุทของ RS-232

ข้อแตกต่างเพียงข้อเดียวระหว่างคำจำกัดความสำหรับอินพุทกับเอาต์พุตคือ ความกว้างของช่วงเปลี่ยนสถานะ (Transition region) โดยช่วงที่ไม่สามารถกำหนดค่าตรรกะได้ของอินพุทกว้างเพียง 6V (+3Vถึง-3V) ในขณะที่ช่วงเดียวกันนี้สำหรับเอาต์พุตกว้างถึง 10 V (+5V ถึง -5V) ซึ่งความแตกต่างนี้มีความสำคัญอย่างมากทีเดียว

ช่วงการยอมรับสัญญาณรบกวน

ความแตกต่างระหว่างคำจำกัดความของแรงดันต่ำสุดที่วงจรยอมรับได้เรียกว่า ช่วงการยอมรับสัญญาณรบกวน (noise margin) ซึ่งหมายความว่าวงจรยอมให้มีสัญญาณรบกวนออกจากเอาต์พุตเข้าสู่อินพุทได้โดยไม่ส่งผลต่อค่าตรรกะที่อินพุท ซึ่งคุณสมบัติข้อนี้มีประโยชน์มากในเวลาที่จำเป็นต้องเดินสายข้อมูลผ่านอุปกรณ์ที่เป็นตัวสร้างสัญญาณรบกวน เช่น มอเตอร์ไฟฟ้า หลอดไฟฟลูโอเรสเซนต์ วงจรหรี่ไฟ และอุปกรณ์การสื่อสารต่างๆ

ส่วนต่างระหว่างช่วงเปลี่ยนสถานะของอินพุทและเอาต์พุตนอกจากจะทำหน้าที่เป็นช่วงยอมรับสัญญาณรบกวนแล้ว ยังทำหน้าที่เป็นช่วงปลอดภัย (safety margin) ด้วย โดยการให้แรงดันเผื่อสำหรับแรงดันที่ตกคร่อมสายเคเบิล ทำให้วงจรสามารถรับแรงดันที่ลดลงจากเอาต์พุตได้ถึงสอง โดยข้อมูลไม่ตกเข้าสู่ช่วงที่กำหนดตรรกะไม่ได้ของอินพุท

เนื่องจากแรงดันไฟกระแสตรง (direct current voltage) สูญเสียไปน้อยมาก ในสายเคเบิลจนสามารถตัดทิ้งไปได้ แม้ในสายไฟขนาดยาวๆ ดังนั้นมาตรฐาน RS-232 จึงมีข้อกำหนดสำหรับสัญญาณควบคุมน้อยกว่าสัญญาณข้อมูล เนื่องจากสัญญาณควบคุมและสัญญาณการแฮนด์เชคเป็นสัญญาณ

Mark และ Space

การรักษาระดับกระแสไฟให้คงที่ในระหว่างช่วงรอทำงาน (Idle) ซึ่งเป็นช่วงที่ไม่มีการส่งข้อมูลออกมาของการส่งข้อมูล สามารถเพิ่มความน่าเชื่อถือของอุปกรณ์จะเพิ่มขึ้นอย่างมาก การส่งข้อมูลสามารถทำได้โดยการขัดจังหวะ (interrupt) กระแสไฟขณะหยุดรอการทำงานนี้ (มีกระแสไฟไหล) ได้มีการกำหนดให้มีชื่อว่า Mark ในทางกลับกันเมื่อไม่มีกระแสไหล (เช่นในช่วงที่มีการส่งข้อมูลจริง) สถานะทางตรรกะได้ถูกกำหนดให้มีค่าเป็น 0 หรือสถานะ Space (Mark = logic “1” , Space = logic “0”)

ตารางที่ 2.2 ตารางมาตรฐานของการใช้แรงดันไฟฟ้า RS-232

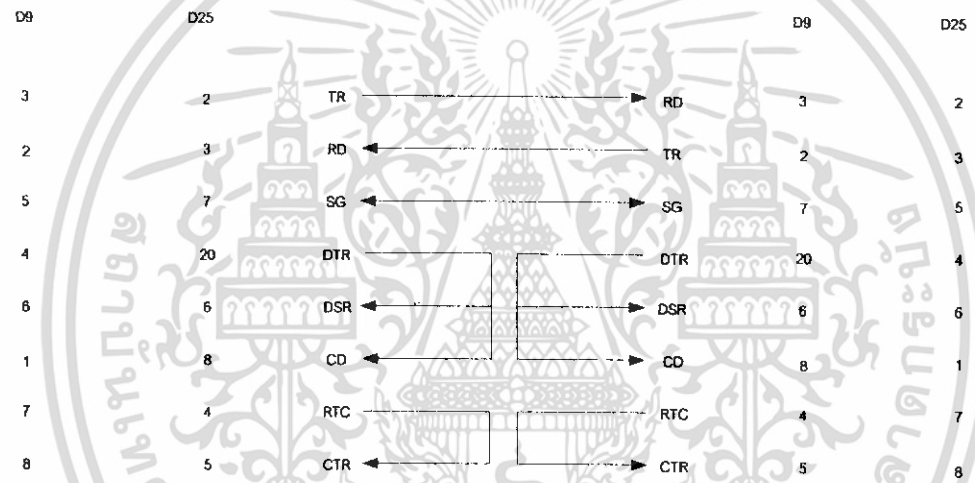
แรงดันไฟฟ้า	สถานภาพลอจิก	สถานภาพของสัญญาณ	ฟังชันก์ในการควบคุม
บวก	0	Mark	on
ลบ	1	Space	off

UART (Universal Asynchronous Receiver/Transmitter) เป็นอุปกรณ์ที่มีความสำคัญมากในการสื่อสารแบบอนุกรมมีหน้าที่อยู่ 3 อย่างที่สำคัญคือ

- Transmitter (ตัวส่งข้อมูล) ทำหน้าที่แปลงข้อมูลขนาด 8 bit ไปเป็นอนุกรมของข้อมูล 8 bit
- Receiver (ตัวรับข้อมูล) จะทำหน้าที่ตรงกันข้ามกับ Transmitter คือทำหน้าที่แปลงข้อมูลขนาด 8 bit ไปเป็นไบต์ข้อมูล
- Control and Status มีหน้าที่ในการเฝ้าดูสถานะทางตรรกะของขาอินพุทต่างๆ และเมื่อโปรแกรมถูกเรียก ก็จะทำหน้าที่เปลี่ยนสถานะทางตรรกะของขาเอาท์พุท

ลักษณะการเชื่อมต่อและหน้าที่การทำงานของแต่ละขาที่สำคัญ

ในปัจจุบันพอร์ตอนุกรมนั้นจะมีอยู่ด้วยกัน 2 ขนาดคือคอนเน็คเตอร์แบบ D-type ตัวผู้ขนาด 25 Pin และตัวผู้ขนาด 9 Pin ซึ่งแสดงการเชื่อมต่อดังรูปที่ 2.9



รูปที่ 2.9 แสดงการเชื่อมต่อทั้งแบบ 25 Pin และ 9 Pin

หน้าที่การทำงานของแต่ละขา (ของชนิด 25 Pin) โดยพิจารณาด้าน DTE

- ขา 2 Transmitted Data (TD) ส่งข้อมูลจาก DTE ไปยัง DCE
- ขา 3 Received Data (RD) ส่งข้อมูลจาก DCE ไป DTE
- ขา 4 Request to Send (RTS) เอาท์พุทอเนกประสงค์ สามารถนำไปประยุกต์ใช้ได้หลากหลายในโมเด็มแบบ half duplex ใช้สัญญาณนี้แสดงความต้องการส่งข้อมูล
- ขา 5 Clear to Send (CTS) อินพุทอเนกประสงค์ นำไปใช้งานหลากหลาย ในโมเด็มแบบ half duplex สัญญาณนี้ใช้อนุญาตให้ส่งข้อมูลได้
- ขา 6 Data Set Ready (DSR) อินพุทอเนกประสงค์ที่ใช้แจ้ง DTE ว่าอุปกรณ์ DCE มีไฟเลี้ยงและพร้อมที่จะทำงาน
- ขา 7 Signal Ground (SG) จุดอ้างอิงแรงดันสำหรับทุกสัญญาณในกระบวนการอินเฟส (ต้องมี)

- ขา 8 Data Carrier Detect (CD) สำหรับโมเด็มจะส่งสัญญาณ DCD เมื่อมันรับรู้การติดต่อกับโมเด็มที่อยู่ห่างออกไป สำหรับ DTE สัญญาณ DCD จะถูกนำไปใช้ในการยกเลิกการรับข้อมูล
- ขา 20 Data Terminal Ready (DTR) เอาท์พุทอเนกประสงค์โดยทั่วไปใช้เป็นสัญญาณบอก DCE ว่าอุปกรณ์ DTE ที่มันอินเตอร์เฟสด้วยมีไฟเลี้ยงและพร้อมที่จะทำงาน

นอกจาก 9 ขาที่กล่าวข้างต้น ยังมีขาอื่นๆ อีกที่ใช้ในการอินเตอร์เฟส แต่สัญญาณสำคัญต่างๆที่มีการนำไปใช้เป็นประจำก็มักมาจาก 9 ขานี้เท่านั้น ซึ่งขาของคอนเน็คเตอร์ ขา 2,3,4,5,6,7,8 และ 20 ว่ากลุ่ม "BIGEIGHT" ส่วนขาสัญญาณอื่นๆ มีไว้สำหรับเป็นทางเลือกที่ผู้ผลิต แต่ละรายจะนำไปประยุกต์ใช้ได้ตามความต้องการ

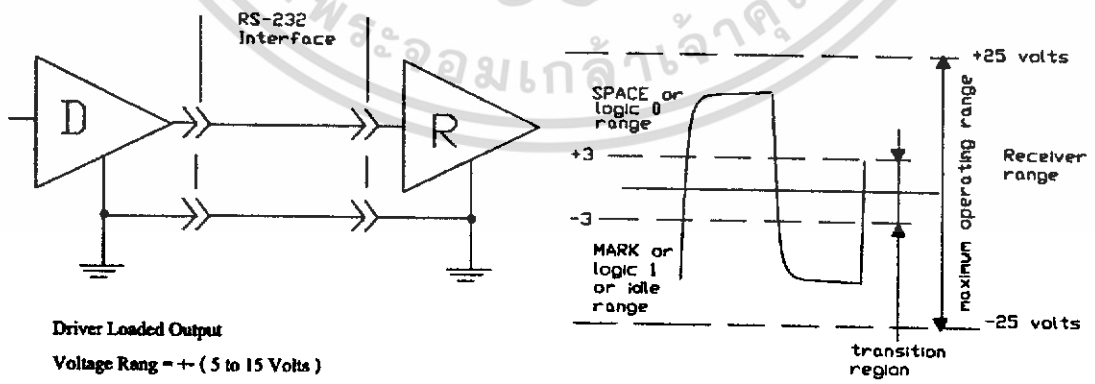
ลักษณะการอินเตอร์เฟสที่ใช้ในมาตรฐาน RS-485

เนื่องจาก RS-422 และ RS-485 เป็นการส่งข้อมูลในระบบสมดุล (Balanced System) เหมือนกัน จึงนำมากล่าวไว้ในบทเดียวกัน และต่อไปนี้จะแทนมาตรฐาน EIA/TIA-422 ด้วย RS-422 และแทนมาตรฐาน EIA/TI-485 ด้วย RS-485

การส่งข้อมูล (Data Transmission Signals)

ตัวส่งแบบไม่สมดุล (Unbalanced line driver)

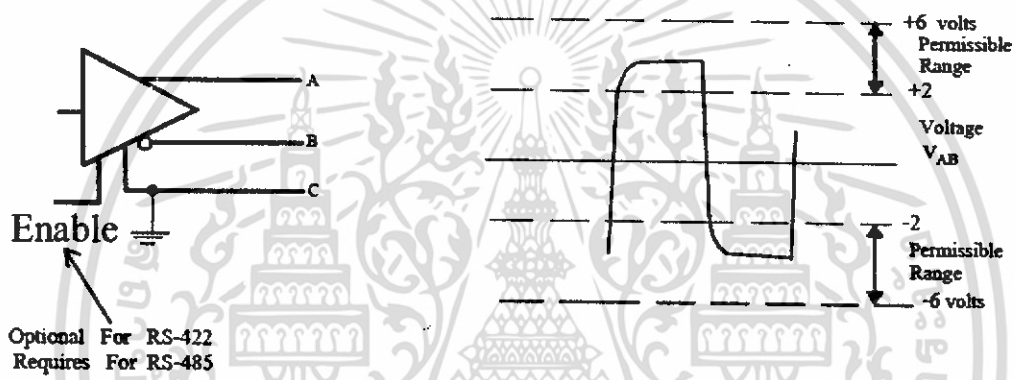
ระดับสัญญาณที่ส่งในระบบไม่สมดุล RS-232 เป็นระบบที่วัดระดับแรงดันสัญญาณเทียบกับสายกราวด์ ยกตัวอย่าง เช่น ในการส่งข้อมูล (Transmission Data : TD) จากอุปกรณ์ (Data Terminal Equipment :DTE) ผ่านหัวต่อแบบ DB-25 จะส่งสัญญาณข้อมูลทางขา 2 โดยการวัดระดับแรงดันของสัญญาณจะวัดเทียบกับสายกราวด์ (Signal ground) ที่ขา 7 ถ้าสายส่งข้อมูลอยู่ในสถานะ idle ระดับแรงดันจะมีค่าเป็นลบ และเมื่อทำการส่งข้อมูลระดับแรงดันจะเปลี่ยนแปลงในช่วงค่าบวกและค่าลบ โดยมีระดับแรงดันอยู่ในช่วง +- 5 V ถึง +- 15 V รูปที่ 2.10 แสดงการทำงานที่ตัวรับของ RS-232 โดยจะทำงานในระดับแรงดันช่วง +3 ถึง +12 และ -3 ถึง -12



รูปที่ 2.10 RS-232 Interface Circuit

ตัวส่งแบบสมดุล (Balanced line driver)

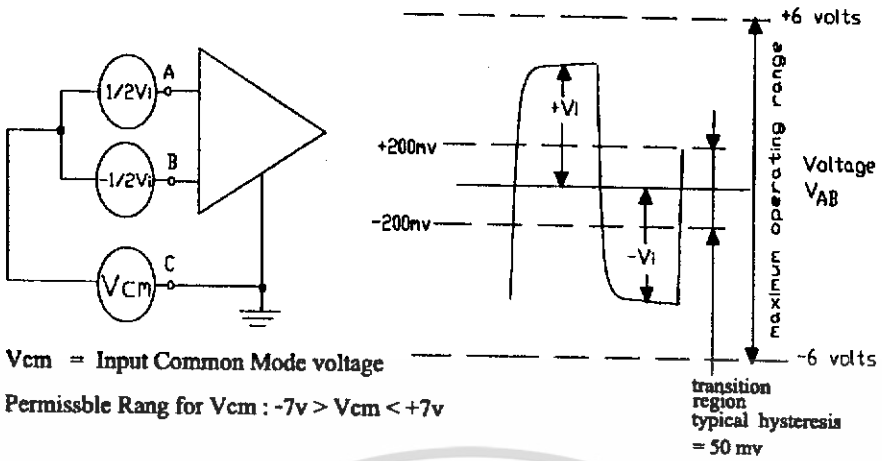
ระบบสมดุลจะส่งสัญญาณผ่านสาย 2 เส้น รูปที่ 2.11 แสดงรูปแบบและระดับแรงดัน ตัวส่งจะส่งแรงดันช่วง 2-6 V ที่เอาท์พุทระหว่าง A และ B ระบบจะมีการเชื่อมต่อสายกราวด์ 1 เส้น สายกราวด์ในระบบนี้ไม่ได้ใช้ในการส่งสัญญาณหรือหาสถานะลอจิกของข้อมูล แต่สายกราวด์มีความสำคัญคือใช้เป็นจุดอ้างอิงของระบบ การวัดสัญญาณข้อมูลจะถูกวัดเทียบกับสายกราวด์ ในการส่งข้อมูลตัวส่งจะได้รับสัญญาณหนึ่งเรียกว่าสัญญาณควบคุม หรือ Enable signal ซึ่งเชื่อมต่อระหว่างตัวส่งและเทอร์มินอล A และ B ที่เอาท์พุท ตัวส่งจะส่งสัญญาณควบคุมซึ่งเปรียบเสมือนเป็นเอาท์พุทที่ 3 นอกจากการส่งสถานะลอจิก 1 หรือ 0 ให้กับเทอร์มินอล ถ้าสัญญาณควบคุมอยู่ในสถานะ OFF จะหมายถึงตัวส่งตัวนั้นไม่ได้ต่ออยู่กับสายสัญญาณและไม่สามารถส่งสัญญาณข้อมูลได้ หรืออยู่ในสถานะ disable หรือ tri-state



รูปที่ 2.11 Balanced Differential Output Line Driver

ตัวรับแบบสมดุล (Balanced line receiver)

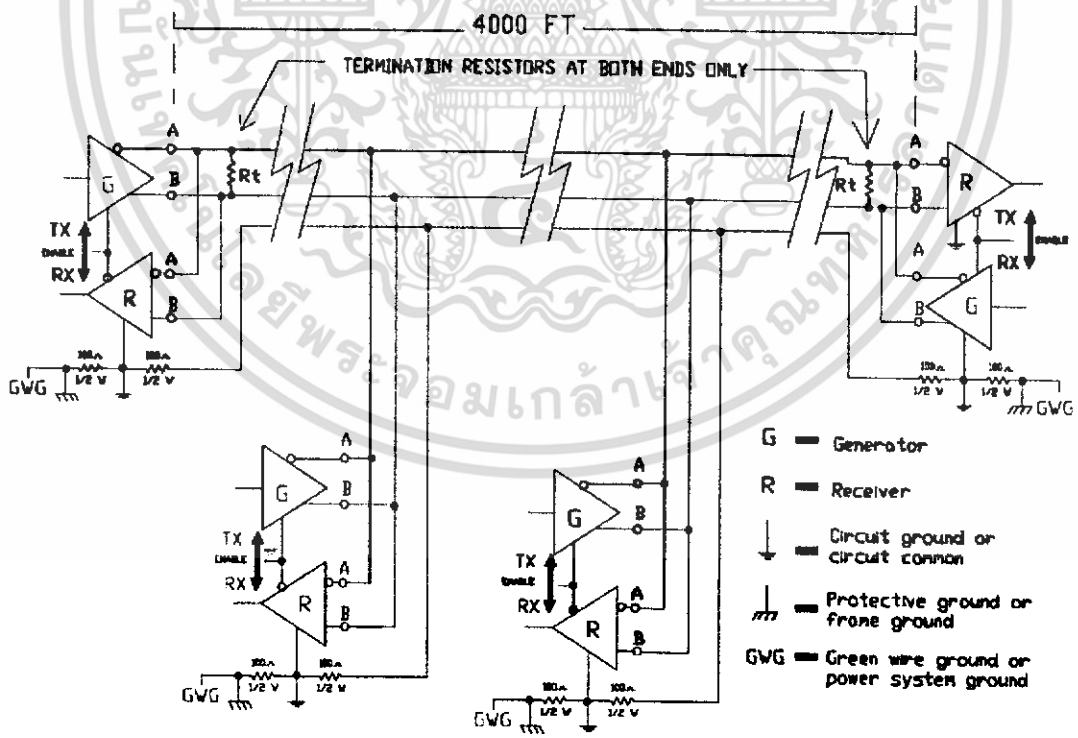
ตัวรับในระบบสมดุลจะถูกเชื่อมต่อด้วยสายส่งสัญญาณข้อมูลและสายกราวด์ โดยจะรับสถานะของสัญญาณได้โดยวัดความแตกต่างของระดับแรงดันที่สายอินพุทของตัวรับ หรือ วัดระหว่างเทอร์มินอล A และ B (V_{AB}) ดังแสดงในรูปที่ 2.12 ถ้าแรงดันมีขนาดมากกว่า +200 mV จะถูกกำหนดให้มีสถานะเป็นลอจิกค่าหนึ่ง และถ้ามีขนาดน้อยกว่า -200 mV จะถูกกำหนดให้มีสถานะตรงข้าม เนื่องจากอาจเกิดการลดทอนสัญญาณ (Attenuate) ขึ้นในสายส่ง ที่ตัวรับจึงถูกออกแบบให้สามารถรับความแตกต่างระดับแรงดันของสัญญาณได้ในช่วงที่กว้างกว่าตัวส่งคือ +- 200 mV ถึง +- 6 V ในขณะที่แรงดันที่ตัวส่งอยู่ในช่วง +- 2 V ถึง +- 6V



รูปที่ 2.12 Balance Differential Input Line Receiver

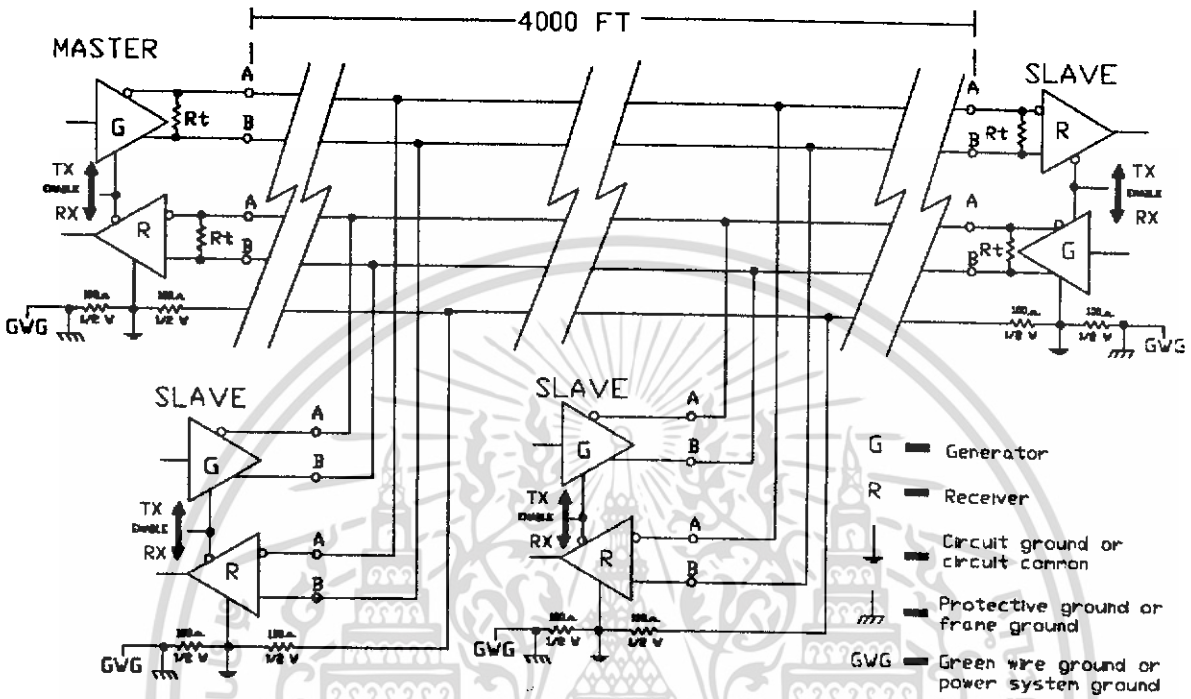
การส่งข้อมูลตามมาตรฐาน RS-485 (EIA Standard RS-485 Data Transmission)

RS-485 เป็นการส่งข้อมูลในระบบสมดุล สายสัญญาณ 1 คู่สายสามารถติดต่ออุปกรณ์ได้ถึง 32 ตัว คุณสมบัติของอุปกรณ์ในระบบ RS-422 และ RS-485 มีลักษณะคล้ายคลึงกัน สำหรับในระบบ RS-485 สามารถทนแรงดันระหว่างสายสัญญาณและสายกราวด์ หรือ Common Mode Voltage หรือ V_{cm} ได้ในช่วง - 7 V ถึง +12 V ซึ่งมากกว่าอุปกรณ์ในระบบ RS-422



รูปที่ 2.13 Typical RS - 485 Two Wire Multidrop Network

รูปที่ 2.13 แสดงระบบเครือข่ายที่เรียกการต่อในลักษณะนี้ว่า two-wire multidrop จากรูปสังเกตได้ว่ามีการต่อความต้านทานขั้วที่โหนดปลายทั้งสองด้านของสายส่งแต่ไม่มีการต่อขั้วปลายที่โหนดที่อยู่ระหว่างสายส่ง การต่อขั้วปลาย (termination) มักใช้กับระบบที่มีอัตราการส่งข้อมูลสูงและระยะทางยาว



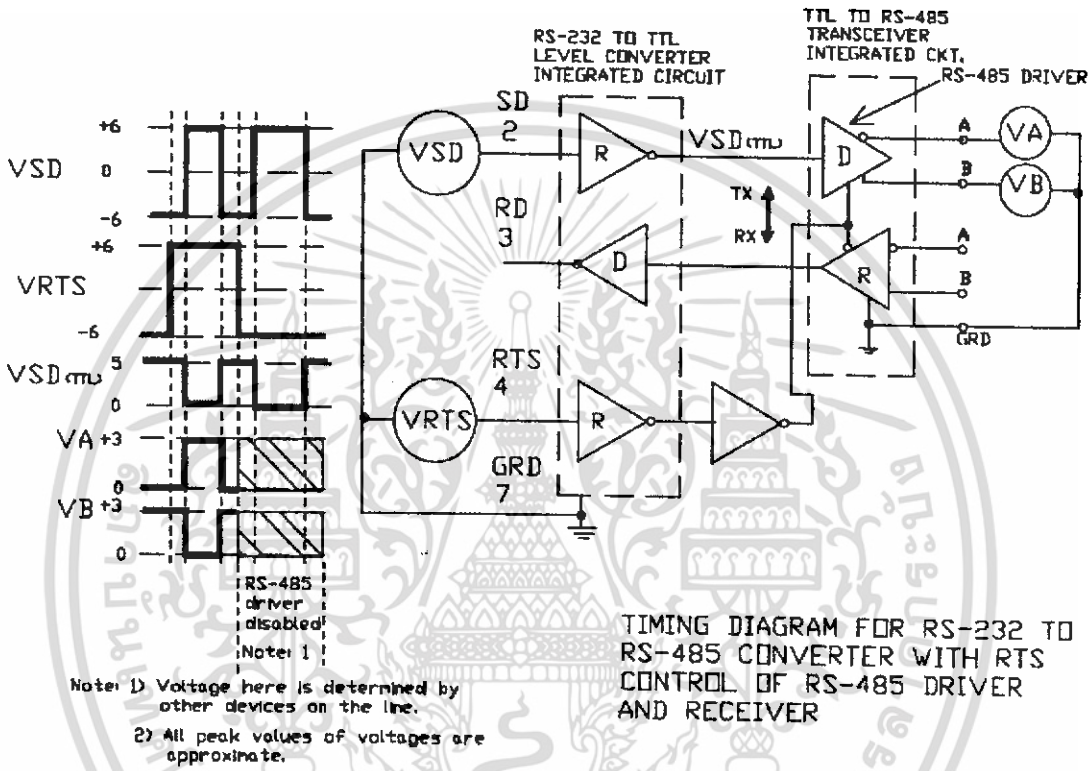
รูปที่ 2.14 Typical RS-485 Four-Wire Multidrop Network

รูปที่ 2.14 แสดงระบบ RS-485 แบบ 4 สาย ประกอบด้วยสายสัญญาณ 4 เส้น และสายกราวด์ 1 เส้น มีลักษณะการติดต่อแบบโหนดแม่ (master node) และโหนดลูก (slave node) ตัวส่งของตัวแม่จะต่อถึงตัวรับของตัวลูกทุกตัวผ่านสายสัญญาณ (โดยมากใช้เป็นสายคู่ตีเกลียว) 1 คู่ และตัวส่งของตัวลูกทุกตัวจะต่อเข้ากับตัวรับของตัวแม่ผ่านสายสัญญาณอีก 1 คู่ โดยในการติดต่อนั้นตัวแม่จะสามารถติดต่อกับตัวลูกได้ทุกตัวแต่ตัวลูกทุกตัวจะติดต่อกับตัวแม่ได้เท่านั้น ไม่สามารถติดต่อกับตัวลูกตัวอื่นหรือติดต่อกันเองได้ และในการติดต่อจากตัวลูกไปยังตัวแม่นั้นจะทำได้ทีละ 1 ตัว เมื่อมีตัวใดตัวหนึ่งติดต่อหรือทำการส่งข้อมูลอยู่ ตัวลูกตัวอื่นที่เหลือในระบบจะไม่สามารถส่งข้อมูลได้โดยต้องอยู่ในสถานะ disable หรือ tri-state เนื่องจากตัวลูกจะไม่สนใจการติดต่อของตัวลูกตัวอื่นกับตัวแม่ ถือเป็นข้อดีคือทำให้สามารถต่ออุปกรณ์ที่มีโปรโตคอลต่างกันเข้าไว้ในระบบเดียวกันได้

การควบคุม Tri-state ของอุปกรณ์ RS-485 โดยการใช้สัญญาณ RTS (Tri-state control of an RS-485 Device using RTS)

ในระบบ RS-485 เป็นการส่งผ่านข้อมูลผ่านสายส่งชุดเดียวกันโดยผลัดกันส่ง เมื่ออุปกรณ์ตัวใดตัวหนึ่งต้องการส่งข้อมูลจะทำการเชื่อมต่อตัวส่งเข้ากับสายส่ง และจะตัดตัวส่งออกจากสายส่งเมื่อส่งข้อมูลเสร็จ โดยส่วนมากมักใช้อุปกรณ์ที่เปลี่ยน RS-232 เป็น RS-485 หรือการ์ดอนุกรม RS-485 ต่อเข้ากับ

ระบบ เพื่อใช้เป็นตัวให้สัญญาณควบคุมการส่ง หรือเรียกว่า RTS : Request To Send โดยต่อจาก Asynchronous serial port ไปยังขา Enable ของตัวส่งผ่านสาย RTS และอาจกำหนดการทำงานของขา Enable โดยถ้าได้รับสถานะ High หรือลอจิก 1 ให้ตัวส่งต่อเข้ากับสายส่งและทำการส่งข้อมูลได้ ถ้าได้รับสถานะ Low หรือลอจิก 0 ให้ตัวส่งตัดการต่อออกจากสายส่ง หรือเรียกว่า tri-state และยอมให้ตัวส่งอื่นที่ได้รับลอจิก 1 ต่อเข้ากับสายส่งและส่งข้อมูลผ่านสายส่งได้ ดังนั้นในขณะที่มีตัวส่งตัวเดียวได้รับลอจิก 1 และต่อเข้ากับสายส่ง ตัวส่งตัวอื่นๆที่เหลือจะต้องได้รับลอจิก 0



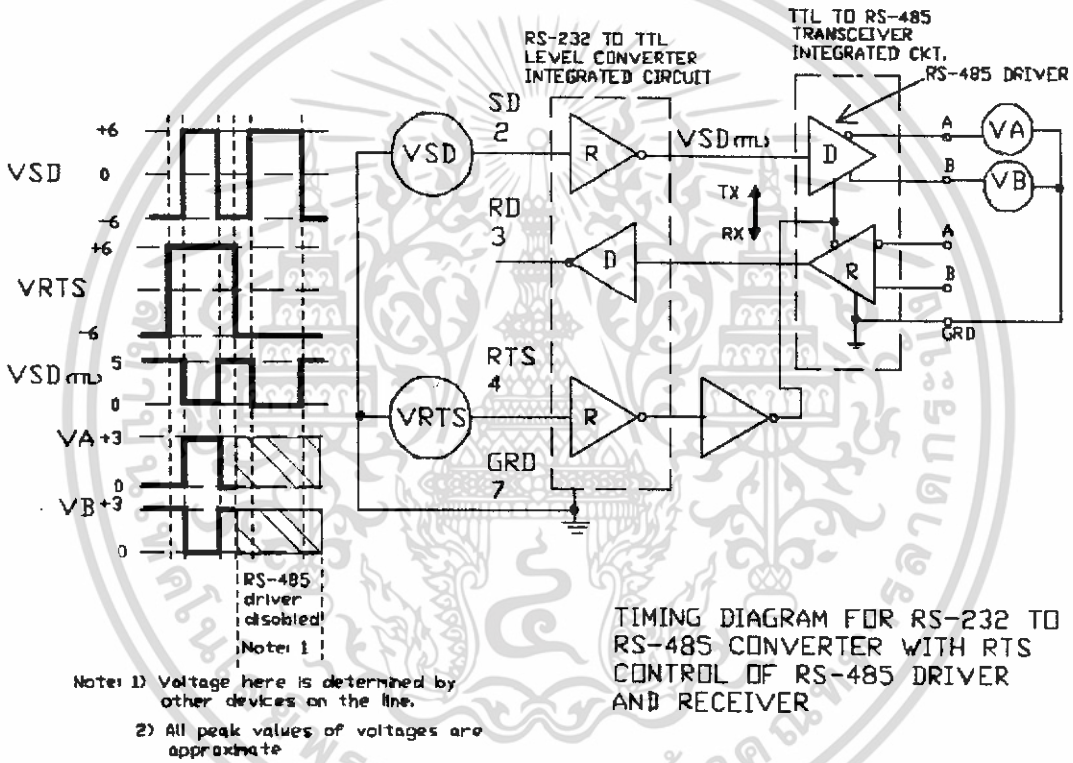
รูปที่ 2.15 Timing Diagram for RS-232 to RS-485 Converter with RTS control of RS-485 Driver and Receiver

รูปที่ 2.15 แสดง Timing diagram ของตัวเปลี่ยน RS-232 เป็น RS-485 (RS-232 to RS-485 Converter) จากรูปคลื่น (waveform) แสดงให้เห็นถึงความผิดพลาดเนื่องจากสัญญาณที่ควบคุมการส่งหรือสัญญาณที่ส่งให้กับขา Enable หรือ VRTS ที่มีขนาดแคบกว่าสัญญาณข้อมูล (VSD) มีผลทำให้เกิดการสูญหายของข้อมูลในส่วนหลังนับจากที่สัญญาณ VRTS อยู่ในสถานะ low ดังนั้นต้องมั่นใจว่าสัญญาณควบคุมหรือ VRTS ต้องมีขนาดกว้างกว่าสัญญาณข้อมูล (VSD) หรือกล่าวได้ว่าสัญญาณ RTS จะต้องเป็น High ก่อนข้อมูลจะถูกส่ง และจะต้องเป็น Low หลังจากส่งบิทสุดท้ายแล้ว ซึ่งช่วงเวลาการทำงานดังกล่าวนี้จะถูกทำโดย software ที่ทำหน้าที่ควบคุมผ่านพอร์ตอนุกรมในการ์ดอนุกรม RS-485

ดังนั้นในการต่ออุปกรณ์หรือการเชื่อมต่อระบบเครือข่าย RS-485 ในลักษณะ 2 สายที่ตัวรับแต่ละตัวจะถูกต่อเข้ากับสายส่งข้อมูลดังในรูปที่ 2.15 จำเป็นต้องศึกษาถึงวิธีการจัดการในส่วนของ Enable function ของแต่ละตัวอุปกรณ์

การควบคุมการส่งข้อมูลของอุปกรณ์ RS-485 (Send Data Control of a RS-485 Device)

ผลิตภัณฑ์ตัวเปลี่ยน RS-232 เป็น RS-485 (RS-232 to RS-485 Converter) และการ์ดอนุกรม RS-485 (RS-485 serial cards) มีวงจรพิเศษในการใช้สัญญาณข้อมูลเป็นตัว enable ให้กับตัวส่ง (RS-485 driver) รูปที่ 2.16 แสดง Timing diagram ของสัญญาณที่ใช้ควบคุมตัวเปลี่ยน (Converter) จาก diagram แสดงให้เห็นว่ามีช่วงเวลาคงที่เวลาหนึ่งหลังการส่งข้อมูลบิทสุดท้ายและก่อนการ disable ของตัวส่ง ถ้าระยะเวลานั้นสั้นเกินไปหรือตัวส่งอาจถูก disable ก่อนที่ตัวส่งจะส่งข้อมูลเสร็จ อาจทำให้ข้อมูลในส่วนท้ายสูญหายได้ และถ้ามีขนาดยาวเกินไปจะทำให้ระบบเปลี่ยนสายสัญญาณจากส่งเป็นรับก่อนที่โหนดพร้อมจะรับข้อมูลหรือตัวรับเชื่อมต่อเข้ากับสายส่งและรับข้อมูลช้าไม่ทันต่อสัญญาณที่ส่งมาถึง ทำให้ไม่ได้รับข้อมูลในส่วนแรก ดังนั้นควรกำหนดช่วงเวลาดังกล่าวให้เหมาะสม โดยส่วนมากมักเท่ากับความยาวหนึ่งตัวอักษรที่อัตราการส่งข้อมูลนั้นๆ



รูปที่ 2.16 Timing Diagram for RS-232 to RS-485 Converter with Send Data (SD) Control of RS-485 Driver and Receiver

2.4 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล MCS-51 เป็นไมโครคอนโทรลเลอร์แบบ 8 บิตซึ่งมีอยู่ด้วยกันหลายเบอร์ และเบอร์ที่เรียกใช้แตกต่างกันออกไปขึ้นอยู่กับโครงสร้างภายในและเทคโนโลยีที่ใช้ในการสร้าง โดยสรุปได้ดังนี้

803x : ไม่มีหน่วยความจำภายใน (Internal Memory) เช่น 8031,8032 เป็นต้น

805x : มีหน่วยความจำภายใน

8xxx : ใช้เทคโนโลยี NMOS

8xcxx : ใช้เทคโนโลยี CMOS

89xx : มีหน่วยความจำภายในเป็นแบบ Flash PEROM

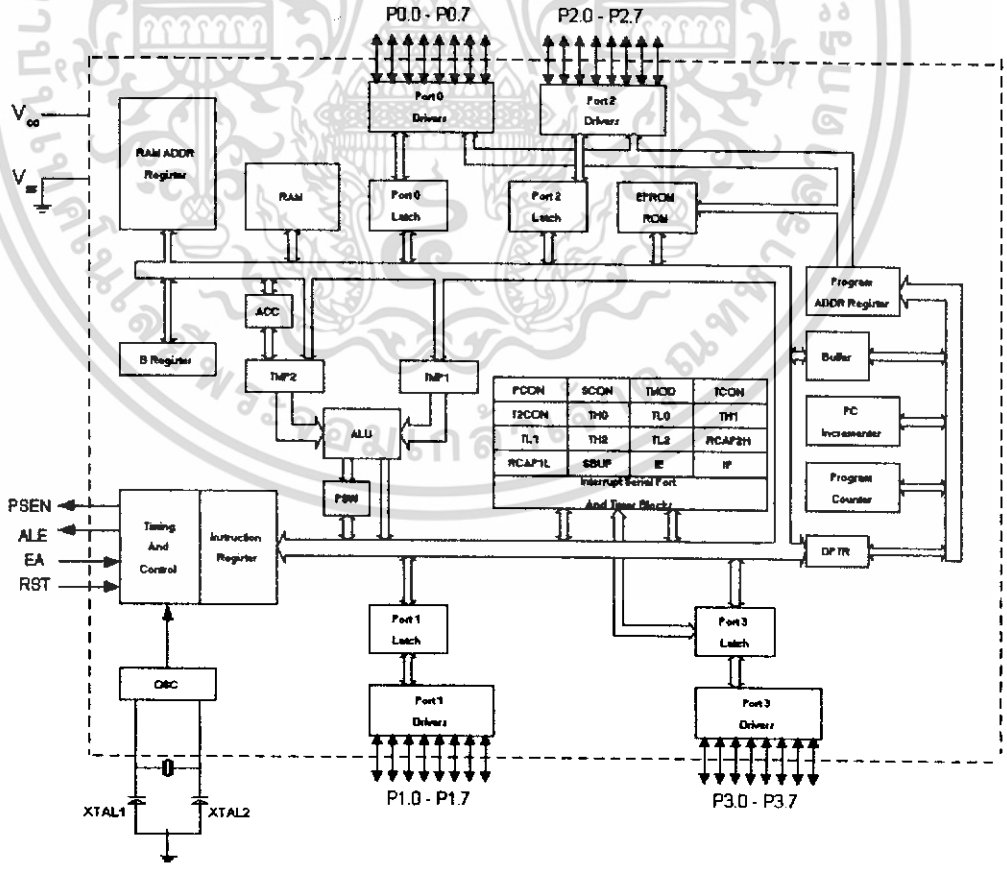
8xx1 : มีหน่วยความจำโปรแกรมภายใน 4 กิโลไบต์ (kByte) , RAM ภายใน 128 ไบต์ (Byte) , ตัวจับเวลา (Timer) และมีตัวนับ (Counter) เป็น 16 บิต (Bit) 2 ชุด

8xx2 : มีหน่วยความจำโปรแกรมภายใน 8 กิโลไบต์ , RAM ภายใน 256 ไบต์ , ตัวจับเวลา และมีตัวนับเป็น 16 บิต 3 ชุด

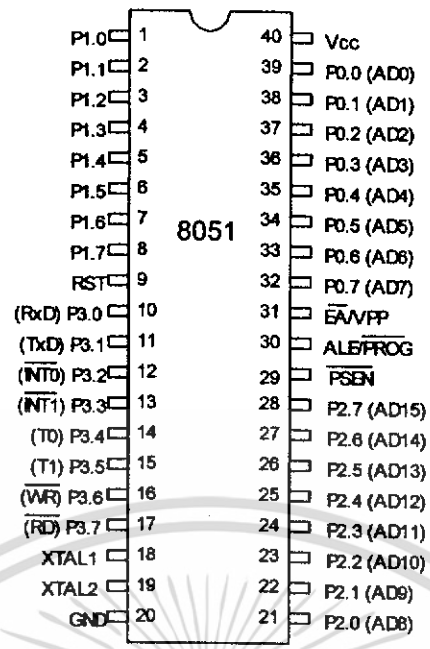
คุณสมบัติของไมโครคอนโทรลเออร์ MCS-51

- ต้องการแหล่งจ่ายไฟ +5V
- มีหน่วยความจำโปรแกรมภายใน 4 กิโลไบต์สำหรับ 8xx1 และ 8 กิโลไบต์สำหรับ 8xx2 ส่วน 803x จะไม่มีหน่วยความจำชุดนี้
- มีหน่วยความจำภายในสำหรับเก็บข้อมูล 128 ไบต์สำหรับ 8xx1 และ 256 ไบต์สำหรับ 8xx2
- มีหน่วยความจำภายนอกสำหรับเก็บโปรแกรมและข้อมูล 64 กิโลไบต์
- คำสั่งที่ใช้เวลาน้อยที่สุดประมาณ 1 μs เมื่อทำงานที่ความถี่ 12 MHz
- มีตัวจับเวลาและตัวนับ 16 บิต ซึ่งทำงานได้ 4 โหมด (Mode)
- มีพอร์ต (Port) รับส่งข้อมูลอนุกรมแบบฟูลดูเพล็ก (full duplex) ตามมาตรฐาน UART (Universal Synchronous Asynchronous Receiver Transmitter)
- สามารถที่จะรับการขัดจังหวะ (Interrupt) ได้

โครงสร้างภายในของ MCS-51 เป็นดังนี้



รูปที่ 2.17 โครงสร้างภายในของ MCS-51

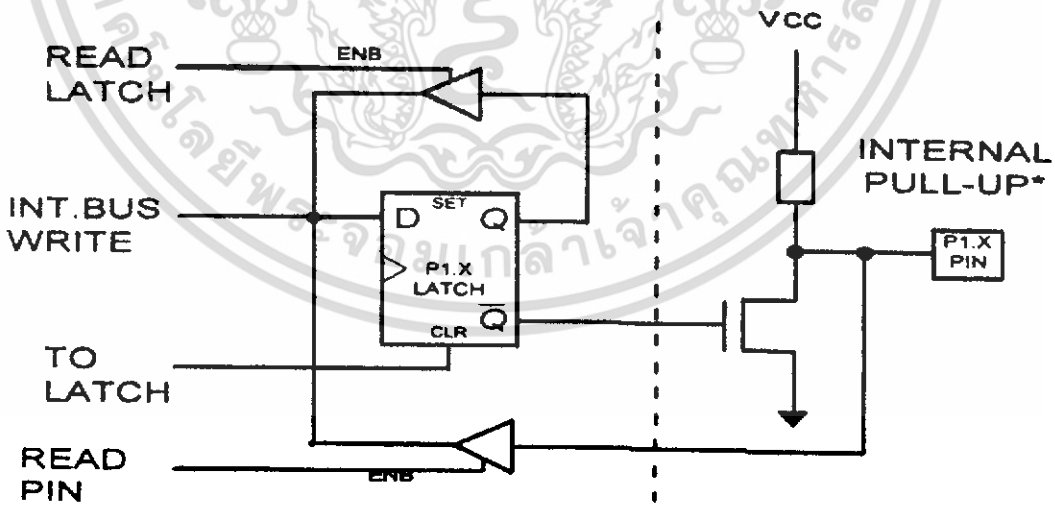


รูปที่ 2.18 การจัดวางขาของ 8051

พอร์ตต่างๆ ของ 8051 ดังนี้

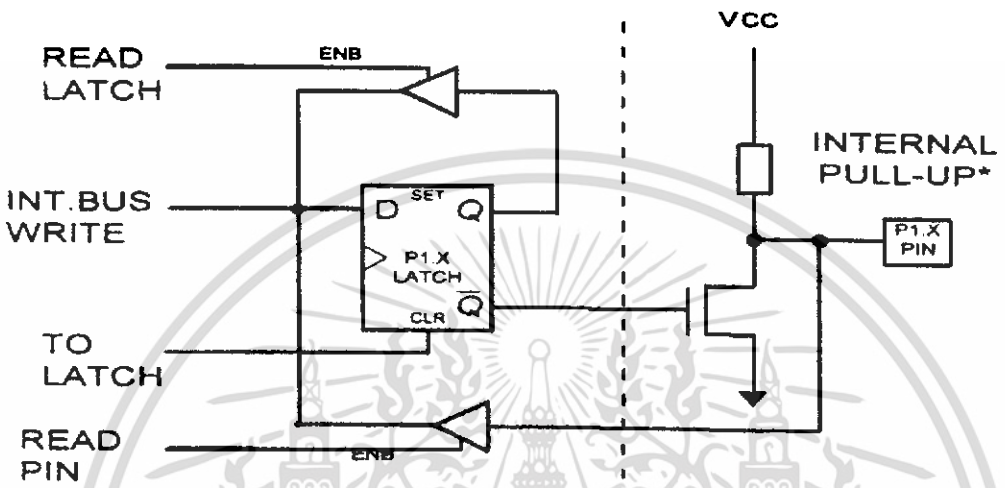
- V_{cc} (ขาที่ 40) ต่อกับ +5V
- V_{ss} (ขาที่ 20) ต่อลง GND

- Port 0 (ขาที่ 32 -39) มีทั้งหมด 8 บิต (P0.0 – P0.7) นอกจากจะใช้งานเป็นพอร์ตอินพุทเอาท์พุท (I/O)แล้ว ยังถูกใช้งานเป็นขาแอดเดรสและขาข้อมูลด้วย โดยแต่ละบิตจะมีชื่อ AD0-AD7 ถ้าต้องการให้พอร์ตนี้ทำหน้าที่เป็น I/O ต้องป้อนลอจิก (logic) 1 ลงไปทุกบิตของพอร์ต 0 จึงจะทำงานได้



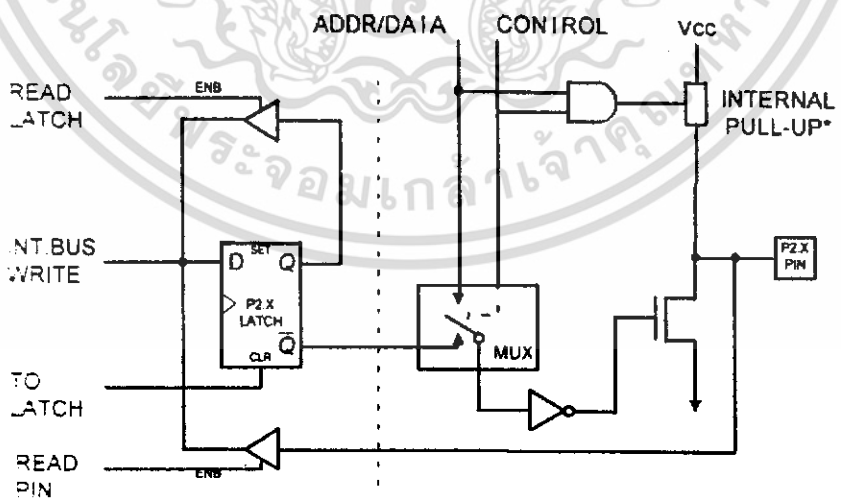
รูปที่ 2.19 โครงสร้างของ Port 0

- Port 1 (ขาที่ 1 -8) มี 8 บิต (P1.0 - P1.7) โครงสร้างภายในจะมีตัวต้านทานต่อแบบ pull-up อยู่ ทำให้สามารถใช้งาน เป็นพอร์ตอินพุทเอาท์พุทได้ทันที ถ้าต้องการให้พอร์ตนี้ทำหน้าที่เป็น I/O ต้องป้อนลอจิก 1 ลงไปทุกบิต



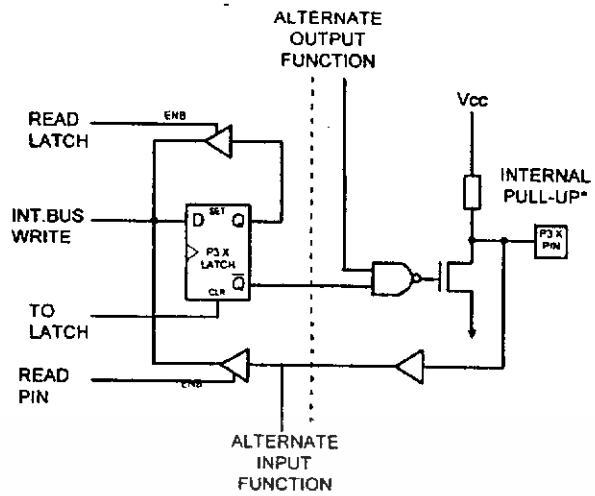
รูปที่ 2.20 โครงสร้างของ Port 1

- Port 2 (ขาที่ 21 -28) มี 8 บิต (P2.0 – P2.7) โครงสร้างภายในคล้าย 0 และมีตัวต้านทานต่อแบบ พูลอัพ (pull-up) อยู่ ถ้าต้องการให้พอร์ตนี้ทำหน้าที่เป็น I/O ต้องป้อนลอจิก 1 ลงไปทุกบิต นอกจากนี้ยังใช้งานเป็นขาแอดเดรสสูงด้วย คือ AD8-AD15



รูปที่ 2.21 โครงสร้างของ Port 2

- Port 3 (ขาที่ 10 -17) มี 8 บิต (P3.0 – P3.7) โครงสร้างคล้ายพอร์ต 1 ซึ่ง ในแต่ละบิตได้ออกแบบไว้ให้ใช้งานเป็นพอร์ตควบคุม โดยมีหน้าที่ดังนี้



รูปที่ 2.22 โครงสร้างของ Port 3

บิต	ชื่อฟังก์ชั่น	ขาที่	หน้าที่
P3.0	RxD	10	ใช้เป็นขารับข้อมูลเข้าแบบอนุกรม
P3.1	TxD	11	ใช้เป็นขาส่งข้อมูลแบบอนุกรม
P3.2	INT0	12	ใช้เป็นขารับสัญญาณขัดจังหวะจากภายนอกหมายเลข 0
P3.3	INT1	13	ใช้เป็นขารับสัญญาณขัดจังหวะจากภายนอกหมายเลข 1
P3.4	T0	14	ใช้รับสัญญาณจากภายนอกของตัวนับหมายเลข 0
P3.5	T1	15	ใช้รับสัญญาณจากภายนอกของตัวนับหมายเลข 1
P3.6	WR	16	เป็นสัญญาณที่ใช้เขียนข้อมูลลงหน่วยความจำภายนอก
P3.7	RD	17	เป็นสัญญาณที่ใช้อ่านข้อมูลจากหน่วยความจำภายนอก

ตารางที่ 2.3 หน้าที่ต่างๆของพอร์ต 3 ในแต่ละบิต

RST (ขาที่ 9) เป็นขา reset ซึ่งตัว CPU จะ reset เมื่อเราป้อนลอจิก 1 นานอย่างน้อย 2 แมชชีนไซเคิล (Machine Cycle) ค่ารีจีสเตอร์ (Register) ต่างๆหลังไมโครคอนโทรลเลอร์ถูกรีเซ็ต (reset) จะมีค่าดังนี้

Register	ค่าเริ่มต้น
PC	0000
ACC	0000
B	0000
PSW	0000
SP	0007
DPTR	0000

ตารางที่ 2.4 ค่ารีจีสเตอร์ต่างๆ หลังถูก reset

$\overline{EA}$: External Access (ขาที่ 31) ถ้าขานี้มีลอจิกเป็น 1 หมายความว่าให้อ่านโปรแกรมจากหน่วยความจำภายในชิพ แต่ถ้าเป็น MCS-51 ที่มีหน่วยความจำอยู่ภายนอก ต้องต่อขา $\overline{EA}$ นี้ลง GND เป็นลอจิก 0 เพื่อให้ตัวซีพียู (CPU) อ่านหน่วยความจำภายนอกชิพ

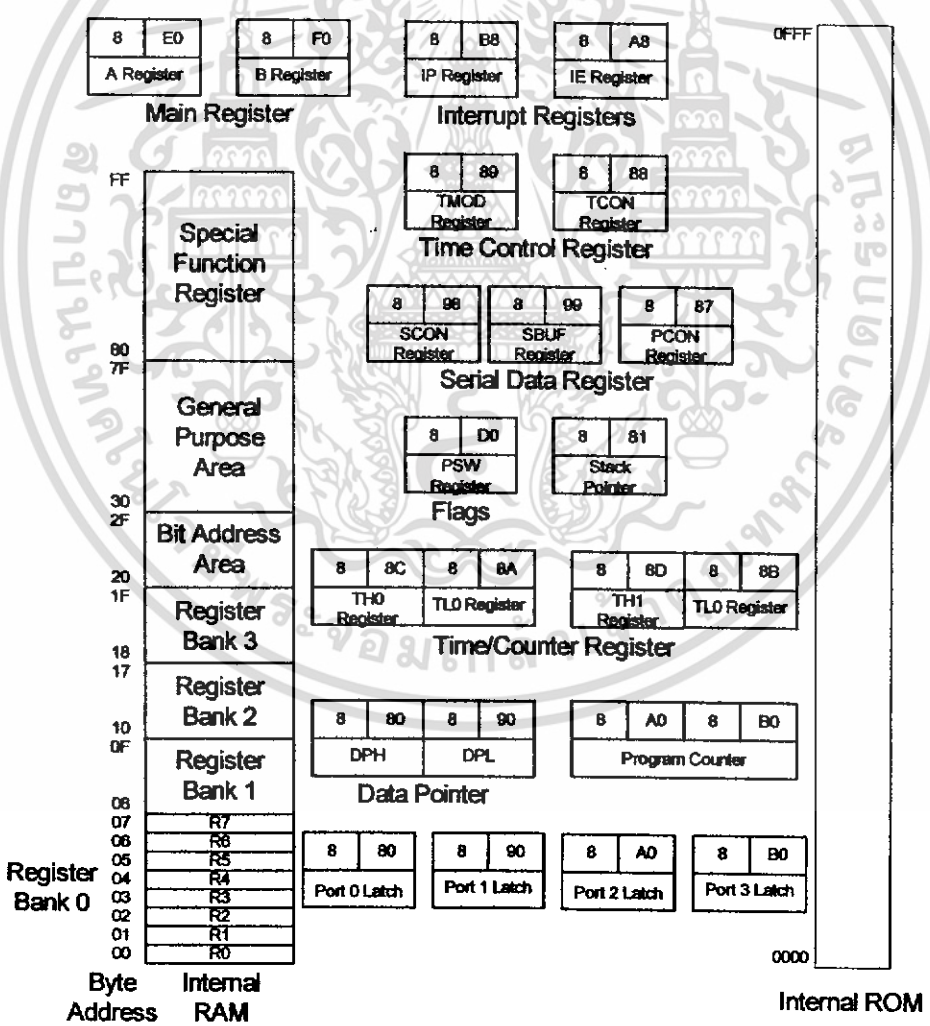
$\overline{PSEN}$: Program Store Enable (ขาที่ 29) ขานี้จะทำงานเมื่อลอจิกเป็น 0 คือ ต้องการอ่านค่าจากหน่วยความจำโปรแกรมภายนอก (หน่วยความจำประเภท EPROM) แต่ถ้าเป็นการอ่านจากหน่วยความจำภายใน ขานี้จะไม่มีสัญญาณออกมา

ALE : Address Latch Enable (ขาที่ 30) เป็นขาเอาท์พุทที่ทำงานเมื่อลอจิกเป็น 1 โดย CPU จะส่งสัญญาณนี้ออกมาเอง เนื่องจากพอร์ต P0 ทำงานได้ 2 หน้าที่คือ ขาแอดเดรสต่ำและขาข้อมูล ตัว MCS-51 จะใช้สัญญาณขา ALE นี้ มัลติเพล็กซ์ (multiplex) กับ P0 ว่าในขณะนั้นจะใช้งานเป็นขาแอดเดรส หรือขาข้อมูล โดยทั่วไปแล้วในวงจรระบบ MCS-51 ขา ALE นี้มักจะต่อกับขา G ของชิพ 74LS373

XTAL1 (ขาที่ 19) ใช้ต่อคริสตอล (Crystal) ภายนอกโดยเป็นอินพุทเข้าสู่วงจรกำเนิดสัญญาณ

XTAL2 (ขาที่ 20) ใช้ต่อคริสตอลภายนอกโดยเป็นเอาท์พุทของวงจรกำเนิดสัญญาณ

โครงสร้างของ Memory และ Register ภายใน 8xx1

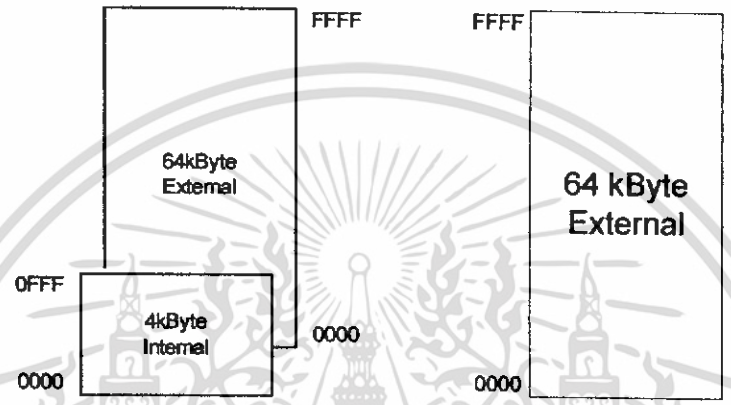


รูปที่ 2.23 ตำแหน่งของ Memory และ Register ของ 8051

หน่วยความจำของไมโครคอนโทรลเลอร์ ซึ่งสามารถแบ่งได้เป็น

หน่วยความจำภายใน (Internal Memory) ซึ่งจะแบ่งเป็นหน่วยความจำโปรแกรม (Internal Program Memory) ซึ่งก็คือ รอม (ROM) ภายในนั่นเอง ถ้าเป็น 8xx1 จะมี 4 กิโลไบต์ (0000 – 0FFF) และ 8xx2 จะมี 8 กิโลไบต์ (0000 – 1FFF) ในหน่วยความจำส่วนนี้จะเอาไว้เก็บโปรแกรมสั่งงานภายในตัวชิพ MCS-51 และอีกส่วนเป็นหน่วยความจำข้อมูล (Internal Data Memory) ซึ่งคือ แรม (RAM) ถ้าเป็น 8xx1 จะมี 128 ไบต์ (00 – 7F) ส่วน 8xx2 จะมี 256 ไบต์ (00 – FF)

หน่วยความจำภายนอก (External Memory) ซึ่งของ MCS-51 มีได้ 64 กิโลไบต์ สามารถแบ่งใช้งานได้ดังนี้



ก.ใช้งานเป็น Internal กับ External ข.แบ่งเป็น External Memory อย่างเดียว

รูปที่ 2.24 (ก.) , (ข.) การแบ่งใช้งานหน่วยความจำของ 8051

หน้าที่ต่างๆ ของรีจิสเตอร์ฟังก์ชั่นพิเศษ

Function	Name	Bit Address
ACC	Accumulator	E0H
B	B Register	F0H
PSW	Program Status Word	D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
P0	Port 0	80H
P1	Port 1	90H
P2	Port 2	A0H
P3	Port 3	B0H
IP	Interrupt Priority Control	B8H

Function	Name	Bit Address
IE	Interrupt Enable Control	A8H
TMOD	Timer/Counter Mode Control	89H
TCON	Timer/Counter Control	88H
TCON 2 (8052)	Timer/Counter 2 Control	C8H
TH 0	Timer/Counter 0 High Byte	8CH
TL 0	Timer/Counter 0 Low Byte	8AH
TH 1	Timer/Counter 1 High Byte	8DH
TL 1	Timer/Counter 1 Low Byte	8BH
TH 2 (8052)	Timer/Counter 2 High Byte	CDH
TL 2 (8052)	Timer/Counter 2 Low Byte	CCH
RCAP2H (8052)	T/C Capture Reg. High Byte	CBH
RCAP2L (8052)	T/C Capture Reg. Low Byte	CAH
SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

ตารางที่ 2.5 หน้าที่ต่างของฟังก์ชั่นต่างๆ

หน้าที่ของรีจิสเตอร์ต่างๆ มีดังนี้

- Accumulator เป็นรีจิสเตอร์ 8 บิต ที่ใช้ประมวลผลทางคณิตศาสตร์ หรือเป็นทางผ่านในการเคลื่อนย้ายข้อมูล

- B Register เป็นรีจิสเตอร์ 8 บิต ที่ใช้คำสั่งคูณและหารร่วมกับ Accumulator

- Program Status Word เป็นรีจิสเตอร์ 8 บิต แต่ใช้เพียง 7 บิต โดยเป็นแฟล็ก (flag) 5 บิต และอีก 2 บิต เป็นบิตเลือก Register Bank (Bank 0-4) ดังรูป

PSW.7 PSW.0

CY	AC	F0	RS 1	RS 0	OV	-	P

รูปที่ 2.25 โครงสร้างของ PSW

โดยที่ CY : แฟล็กทด (carry flag)เกิดการทดที่บิต D7

AC : แฟล็กช่วยทด เมื่อมีการทดจากบิต D3 ไปยัง D4

F0 : แฟล็กที่ผู้ใช้กำหนดเองอย่างอิสระ

RS 1 : บิตเลือก Register Bank บิต 1

RS 0 : บิตเลือก Register Bank บิต 0

OV : โอเวอร์โฟลวแฟล็ก (overflow flag) ถูกเซตเมื่อผลลัพธ์ออกมามีค่ามากกว่าที่รีจิสเตอร์จะเก็บได้

\- : ไม่ได้ใช้งาน

P : พาริตี้แฟล็ก (Parity flag) ใช้แสดงจำนวนลอจิก 1 ในข้อมูล ถ้าเป็น 0 เป็นพาริตี้คู่ และ 1 เป็นพาริตี้คี่

RS 0	RS 1	Register Bank	Address
0	0	Bank 0	00H – 07H
0	1	Bank 1	08H – 0FH
1	0	Bank 2	10H – 17H
1	1	Bank 3	18H – 1FH

ตารางที่ 2.6 ค่า RS ในการเลือก Register Bank

- Stack Pointer เป็นรีจิสเตอร์ 8 บิตที่ใช้ชี้ตำแหน่งของ Stack ทุกครั้ง (ซึ่ง Stack คือ RAM ที่ CPU จอง เพื่อจะเก็บค่าและจำตำแหน่งของโปรแกรม)

- Data Pointer Register เป็นรีจิสเตอร์ 16 บิตแบ่งเป็น DPL (8 บิตล่าง) และ DPH (8บิตบน) มีประโยชน์ในการชี้แอดเดรสของหน่วยความจำ

- Interrupt Priority Register เป็นรีจิสเตอร์ควบคุมการขัดจังหวะ

- Interrupt Enable Register เป็นรีจิสเตอร์ที่กำหนดว่าจะให้ขัดจังหวะได้หรือไม่ แต่ละบิตมีดังนี้

รูปที่ 2.26 โครงสร้างของ IE

โดยที่ EA : ถ้ามีค่าเป็น 0 คือ ไม่สามารถที่จะขัดจังหวะได้ ถ้าเป็น 1 แสดงว่าสามารถขัดจังหวะได้

\- : ไม่ใช้งาน

ET 2 : Enable/Disable Interrupt Timer 2 (8052)

ES : Enable/Disable Interrupt พอร์ตอนุกรม

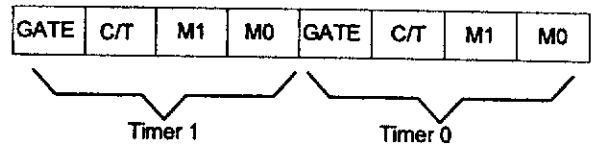
ET 1 : Enable/Disable Interrupt Timer 1

EX 1 : Enable/Disable External Interrupt 1

ET 0 : Enable/Disable Interrupt Timer 0

EX 0 : Enable/Disable External Interrupt 0

- Timer/Counter Mode Control Register เป็นรีจิสเตอร์ 8 บิตที่เลือกการทำงานได้ 4 โหมด โดยเลือกที่บิต M0 กับ M1



รูปที่ 2.27 โครงสร้างของ TMOD

โดยที่ GATE : เป็นบิตที่เซตให้วงจรทำงาน

C/T : เป็นบิตที่ใช้เลือกระหว่างตัวจับเวลากับตัวนับ โดยถ้ามีค่า 0 เลือก timer และถ้าค่าเป็น 1 เลือก counter

M0 : Mode Bit 0

M1 : Mode Bit 1

M0	M1	Mode	ลักษณะการทำงาน
0	0	0	เป็น Timer แบบ 13 บิต โดย TH = 8 และ TL= 5
0	1	1	เป็น Timer แบบ 16 บิต
1	0	2	เป็น Timer แบบ 16 บิตและมี Auto-Reload
1	1	3	เป็น Timer ทั้งแบบ 16 บิต 1 ตัวหรือแบบ 8 บิต 2 ตัว

ตารางที่ 2.7 การเซตค่า M0 และ M1 ในการเลือก Mode ทำงาน

- Timer/Counter Control Register เป็นรีจิสเตอร์ 8 บิต ใช้ควบคุมให้ Timer/Counter เริ่มนับ โดยแต่ละบิตจะมีความหมายดังนี้

รูปที่ 2.28 โครงสร้างของ TCON

โดยที่ TF1 : Timer1 จะแสดงผลของการเกิด overflow

TR1 : ใช้เซตค่าเพื่อควบคุมให้ Timer/Counter 1 เริ่มนับ

TF0 : Timer0 จะแสดงผลของการเกิด overflow

TR0 : ใช้เซตค่าเพื่อควบคุมให้ Timer/Counter 1 เริ่มนับ

IE1 : แสดงสถานะเมื่อมีการขัดจังหวะจากภายนอก บิต 1

IT1 : เป็นบิตเลือกประเภทของขัดจังหวะบิต 1

IE0 : แสดงสถานะเมื่อมีการขัดจังหวะจากภายนอก บิต 0

IT0 : เป็นบิตเลือกประเภทของขัดจังหวะบิต 0

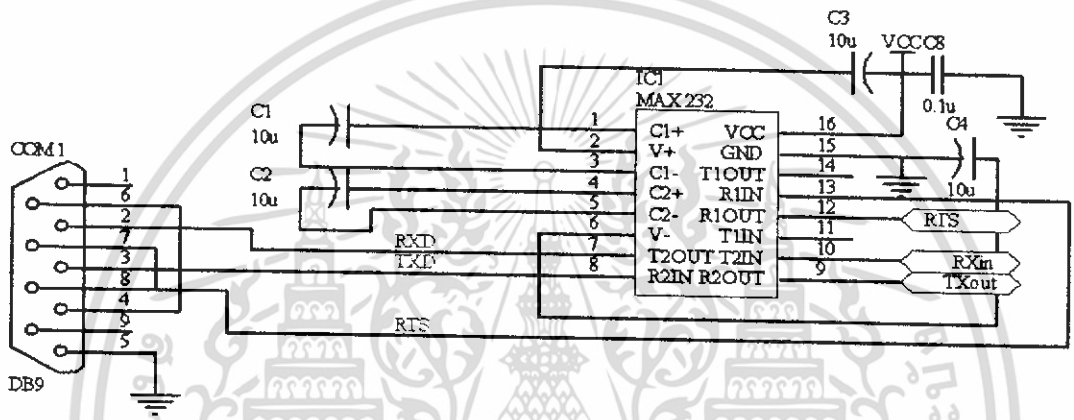
บทที่ 3

การคำนวณและการสร้าง

การคำนวณและการสร้างแบ่งเป็นสองส่วน คือ ส่วนของฮาร์ดแวร์และส่วนของซอฟท์แวร์ การคำนวณและการสร้างส่วนฮาร์ดแวร์

3.1 วงจรแปลงสัญญาณแบบ RS-232

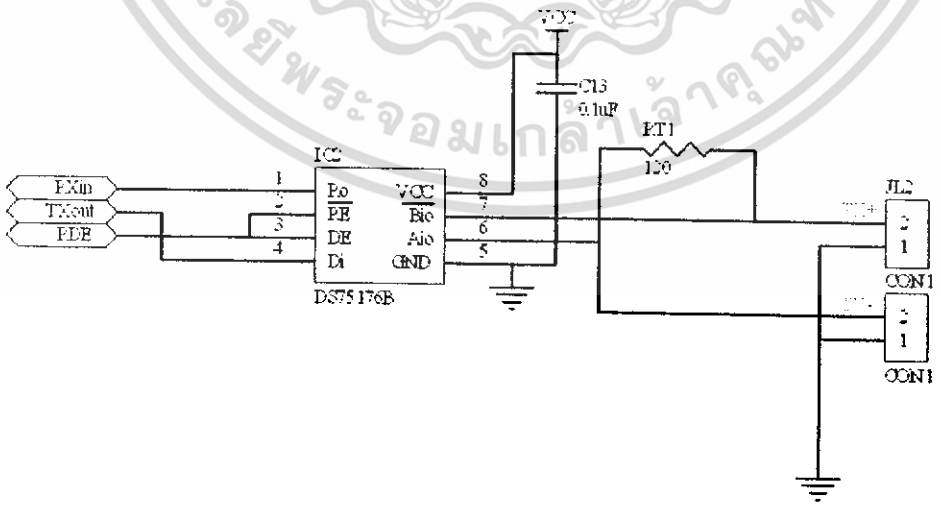
ใช้ไอซี Max 232 ซึ่งทำหน้าที่ปรับระดับสัญญาณให้เหมาะสม เชื่อมต่อกับคอมพิวเตอร์ที่ขา RxD และขา TxD ของ Com 1 หรือ Com 2 มีข้อดีคือมีทั้ง driver และ receiver ในตัวเอง และใช้โวลท์เตจระดับเดียวคือ 5 โวลต์ การนำเอาไอซี Max 232 ไปใช้นั้น ต้องมีการต่อตัวเก็บประจุเข้าไปอีกเล็กน้อย ซึ่งค่าของตัวเก็บประจุที่ใช้ในการทดลองคือ C = 10 uF ดังรูปที่ 3.1



รูปที่ 3.1 วงจรแปลงสัญญาณแบบ RS-232

3.2 วงจร RS-485 อแดปเตอร์

วงจรของตัวแปลงสัญญาณจากมาตรฐาน RS-232 ไปเป็นสัญญาณตามมาตรฐาน RS-485 นั้น สามารถแสดงให้เห็นได้ดังรูปที่ 3.2

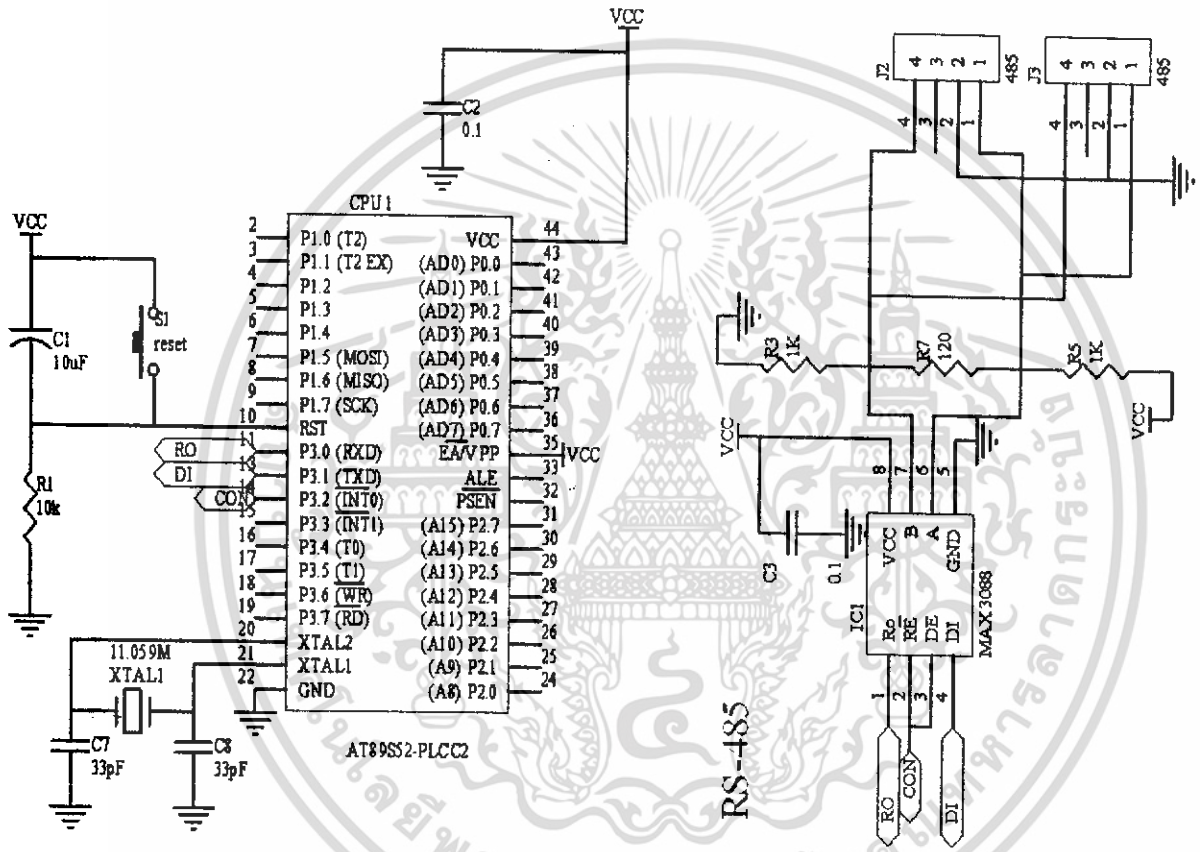


รูปที่ 3.2 แสดงวงจรแดปเตอร์

ในส่วนวงจรของตัวแปลงสัญญาณจะมีการเชื่อมต่อขาที่ทำหน้าที่ในการส่งและรับสัญญาณของพอร์ตสื่อสารแบบอนุกรรมเข้ากับขาของ IC MAX-232 ทำหน้าที่แปลงไฟ ระดับ 12 V เป็นระดับ TTL เพื่อให้ใช้ได้กับ IC DS 75176B เพื่อแปลงจากมาตรฐาน RS-232 เป็นมาตรฐาน RS-485

3.3 วงจรไมโครคอนโทรลเลอร์

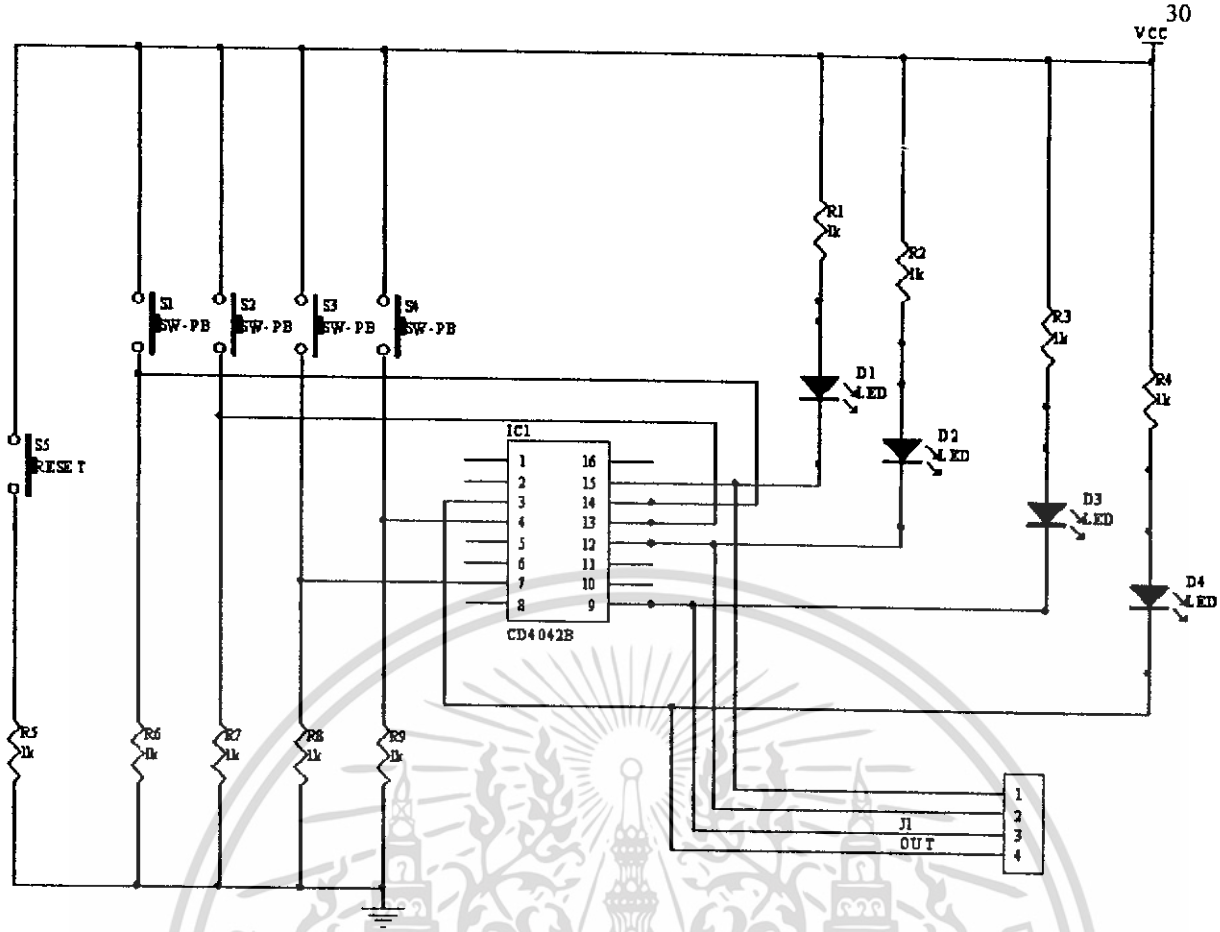
วงจรไมโครคอนโทรลเลอร์เป็นวงจรควบคุมการทำงานต่าง ๆ ของวงจรแต่ละชุดโดยจะต่ออยู่กับ RS-485 อแดปเตอร์เพื่อต่อกับ RS-485 อีกชุดหนึ่งที่ต่ออยู่กับ RS-232 โดยไมโครคอนโทรลเลอร์จะทำหน้าที่รับข้อมูลจากวงจรและส่งข้อมูลไปให้กับคอมพิวเตอร์โดยจะมีรูปวงจรดังนี้



รูปที่ 3.3 วงจรไมโครคอนโทรลเลอร์

3.4 วงจรสวิทช์

เป็นวงจรที่ทำให้เราทราบว่ามีผู้บุกรุกเข้ามาในบ้านในทางใด ถ้าสวิทช์ตัวถูกกด สมมติว่าเป็น S1 ถูกกดขา 4 จะมีสภาวะเป็น "High" และขา 3 จะมีสภาวะเป็น "Low" ทำให้ LED1 สว่างและถ้า S2 ถูกกดขา 7 จะมีสภาวะเป็น "High" และขา 9 จะมีสภาวะเป็น "Low" ทำให้ LED2 สว่าง และถ้า S3 ถูกกดขา 13 จะมีสภาวะเป็น "High" และขา 12 จะมีสภาวะเป็น "Low" ทำให้ LED3 สว่าง และถ้าสุดท้าย S4 ถูกกดขา 14 จะมีสภาวะเป็น "High" และขา 15 จะมีสภาวะเป็น "Low" ทำให้ LED4 สว่าง ตามลำดับ วงจรสวิทช์แสดงได้ดังรูป



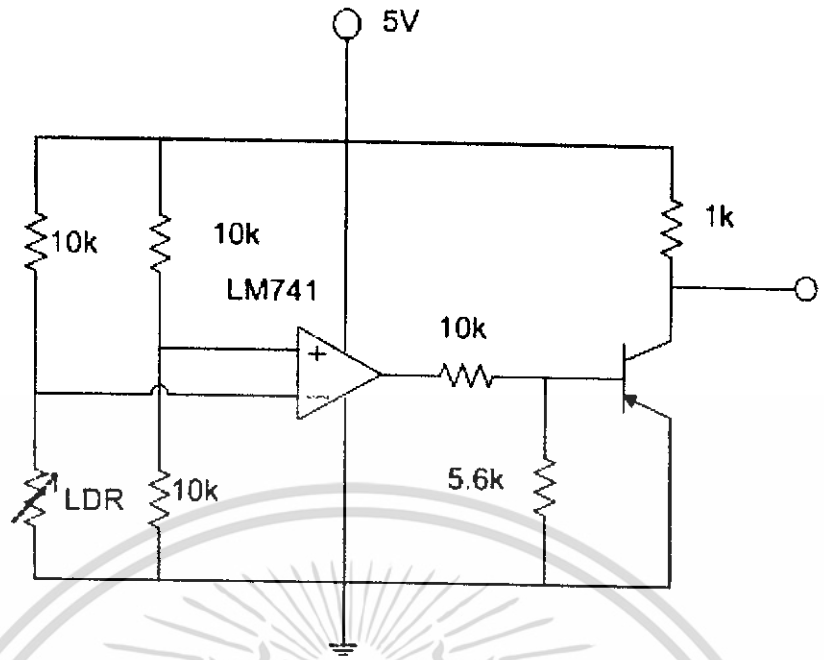
รูปที่ 3.4 วงจรสวิทช์

3.5 วงจรตรวจจับทางแสง

ในวงจรนี้จะใช้ตัวตรวจจับทางแสดงที่เรียกว่า Light Depentdent Resistance (LDR) กล่าวคือ ความต้านทานจะเปลี่ยนไปตามความเข้มของแสงนั้นเอง ถ้าความเข้มของแสงมากความต้านทานของมันจะต่ำ หากกลับกัน ความเข้มแสงน้อยความต้านทานจะสูง

ทำการวัดค่าความต้านทานของ LDR ในช่วงเวลาตามตารางจะได้ค่าดังนี้

ช่วงเวลาในการวัดความต้านทาน	ค่าความต้านทาน
กลางวัน	20 กิโลโอห์ม
กลางคืน	550 กิโลโอห์ม



รูปที่ 3.5 วงจรตรวจจับแสง

จากรูปที่ 3.5 เป็นวงจรที่ใช้ตรวจจับแสงโดยใช้ไอซี LM 741 ต่อเป็นวงจรเปรียบเทียบแรงดัน

ที่สถานะมืด ทรานซิสเตอร์ไม่ทำงาน ทำให้ O/P เท่ากับ 5 Volt

ที่สถานะสว่าง ทรานซิสเตอร์ทำงาน ทำให้ O/P เท่ากับ 0 Volt

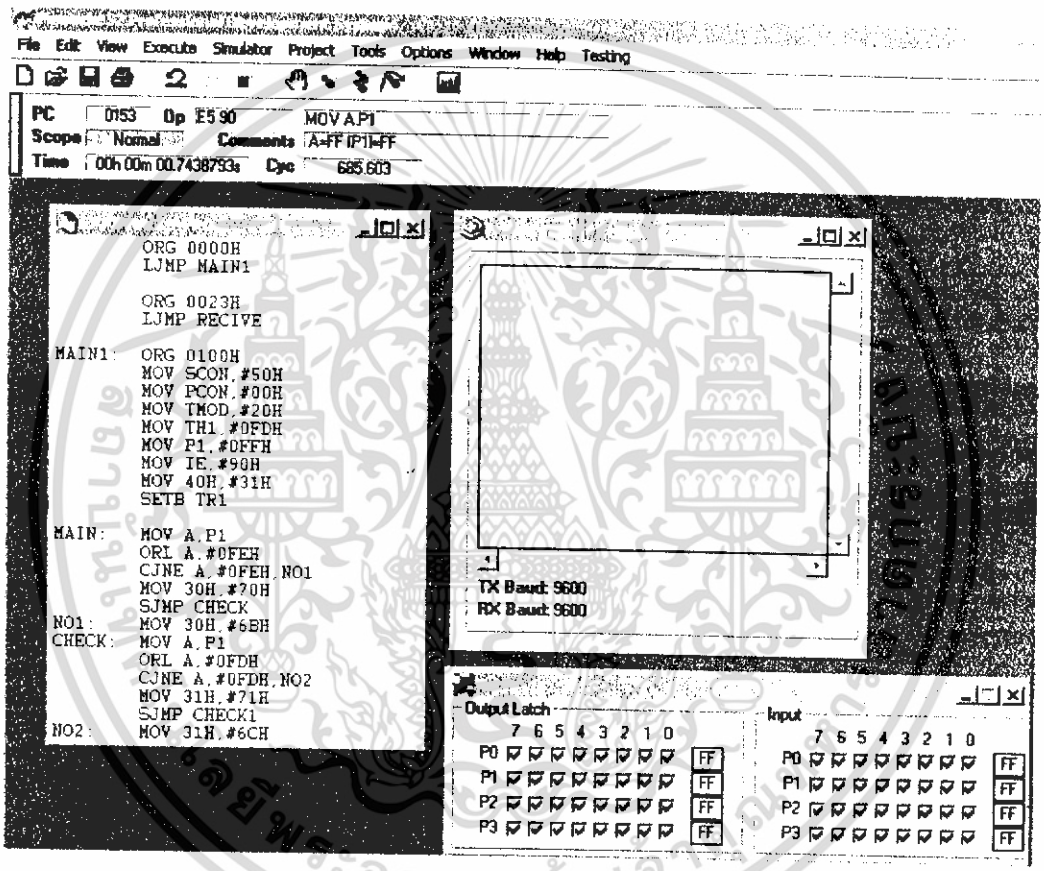
3.6 การคำนวณและการสร้างส่วนซอร์ฟแวร์

ส่วนโปรแกรมควบคุมการทำงานไมโครคอนโทรลเลอร์

โดยขั้นตอนการสร้างจะใช้

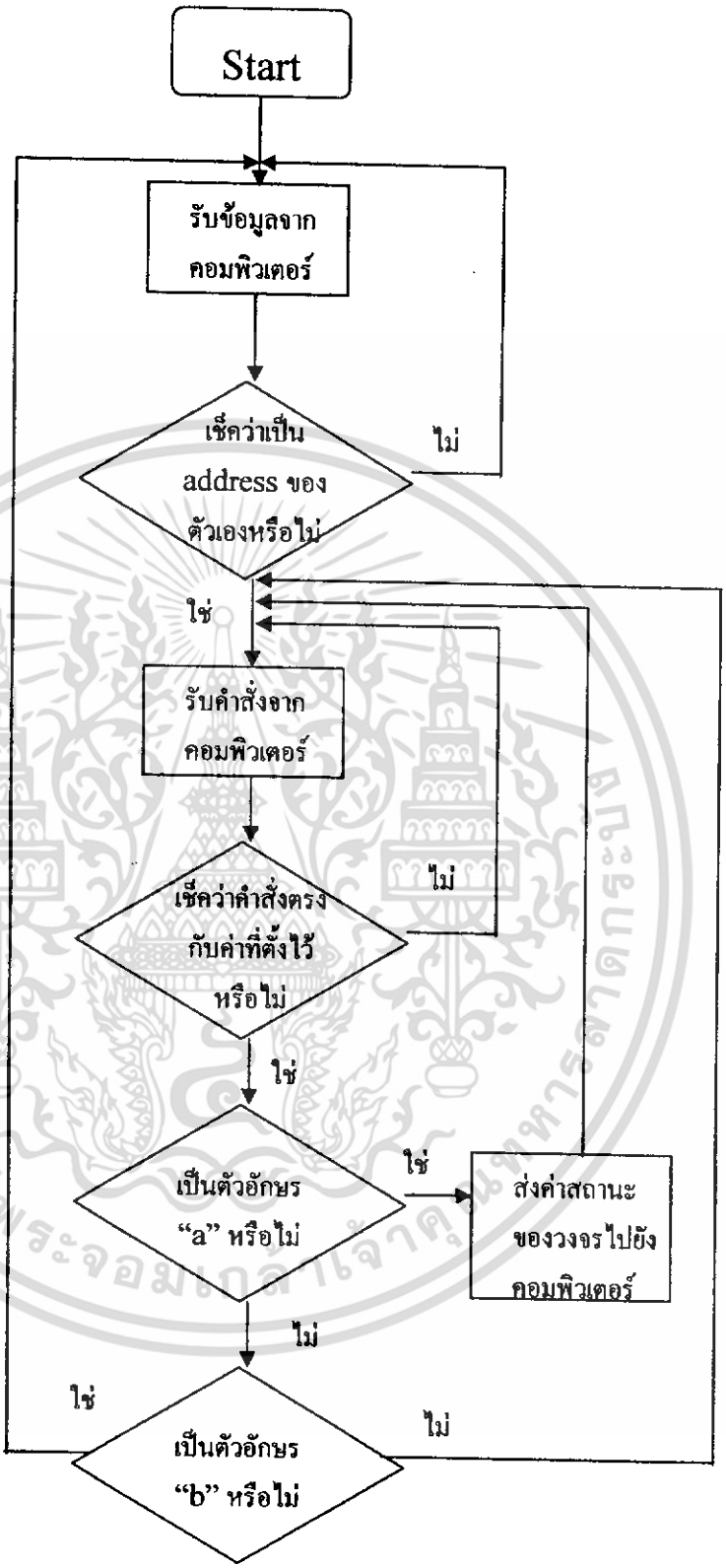
โปรแกรม pinnacle52

เป็นโปรแกรมจำลองการทำงานของไมโครคอนโทรลเลอร์โดยใช้ภาษา Assemble (.asm) ซึ่งจะดีกว่าโปรแกรม SXA51 เพราะสามารถจำลอง input ของ Hyper terminal , input และ output port ซึ่งง่ายแก่การทดสอบโปรแกรมดังรูปที่ 3.6



รูปที่ 3.6 แสดงโปรแกรม Pinnacle52

โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์โดยใช้ภาษา Assembly

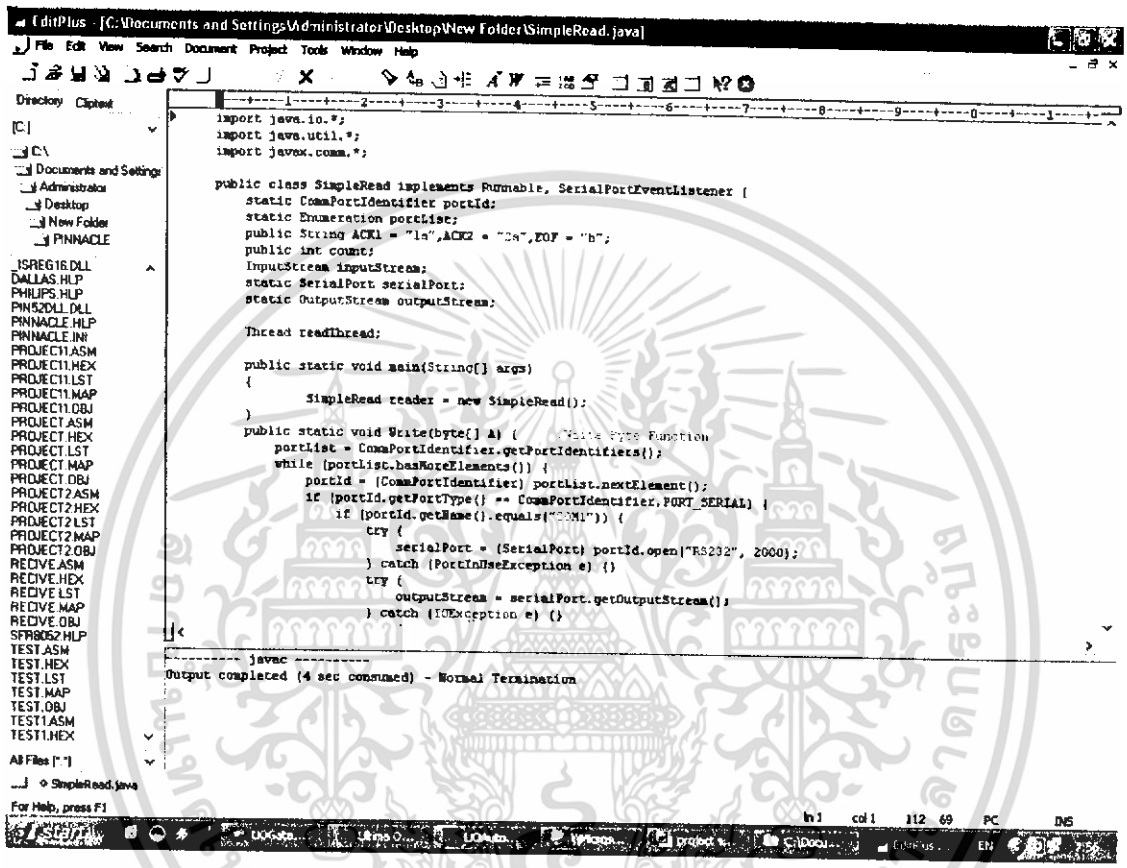


รูปที่ 3.7 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรม

ส่วนโปรแกรมควบคุมการทำงานของคอมพิวเตอร์

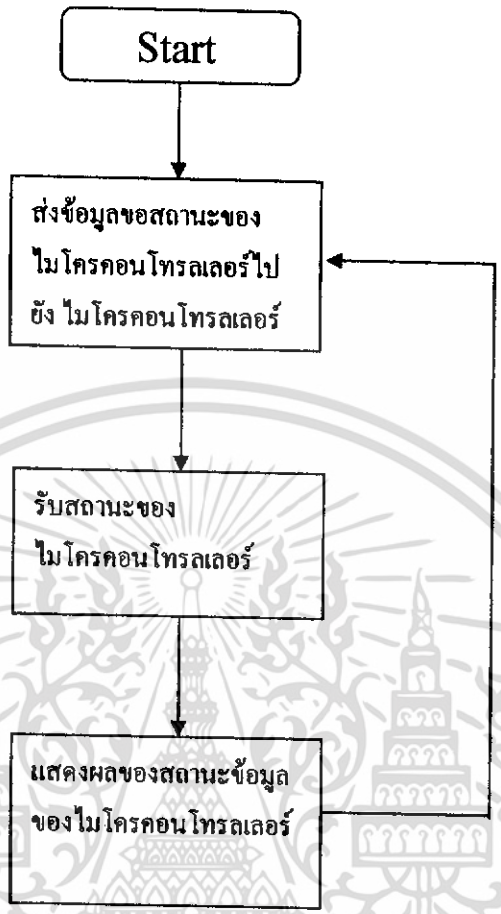
โดยขั้นตอนการสร้างจะใช้โปรแกรม Editplus

โปรแกรม Editplus เป็นโปรแกรมให้ความสะดวกสบายในการแก้ไขข้อมูลเนื่องจากมีสีแบ่งคำสั่งต่างๆ และยังสามารถ compile และ Run ภาษาจาวา



รูปที่ 3.8 แสดงโปรแกรม Editplus

โฟลว์ชาร์ตการทำงานของคอมพิวเตอร์โดยใช้ภาษาจาวา



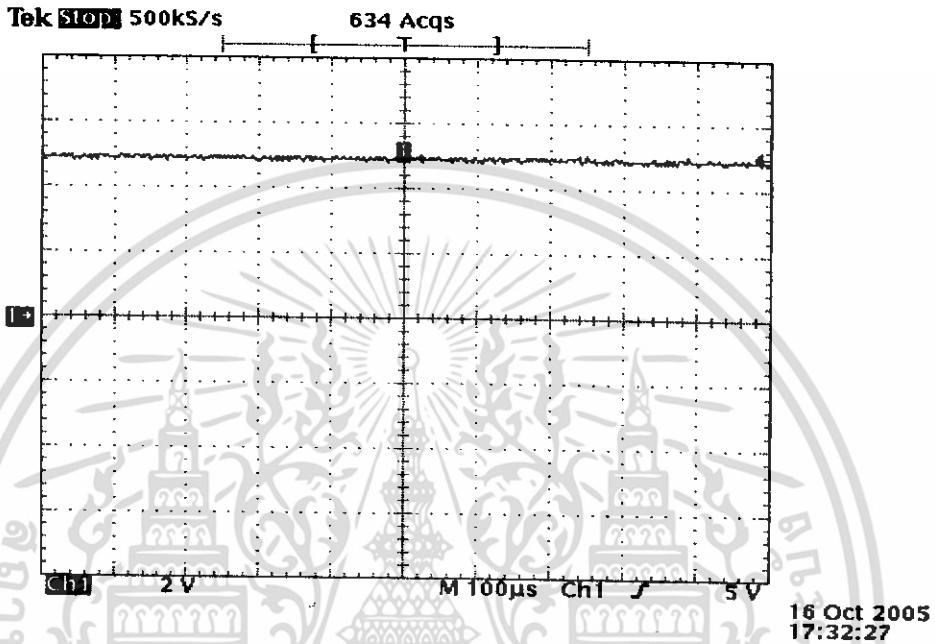
รูปที่ 3.9 โฟลว์ชาร์ตแสดงการทำงานของคอมพิวเตอร์ด้วยจาวา

บทที่ 4

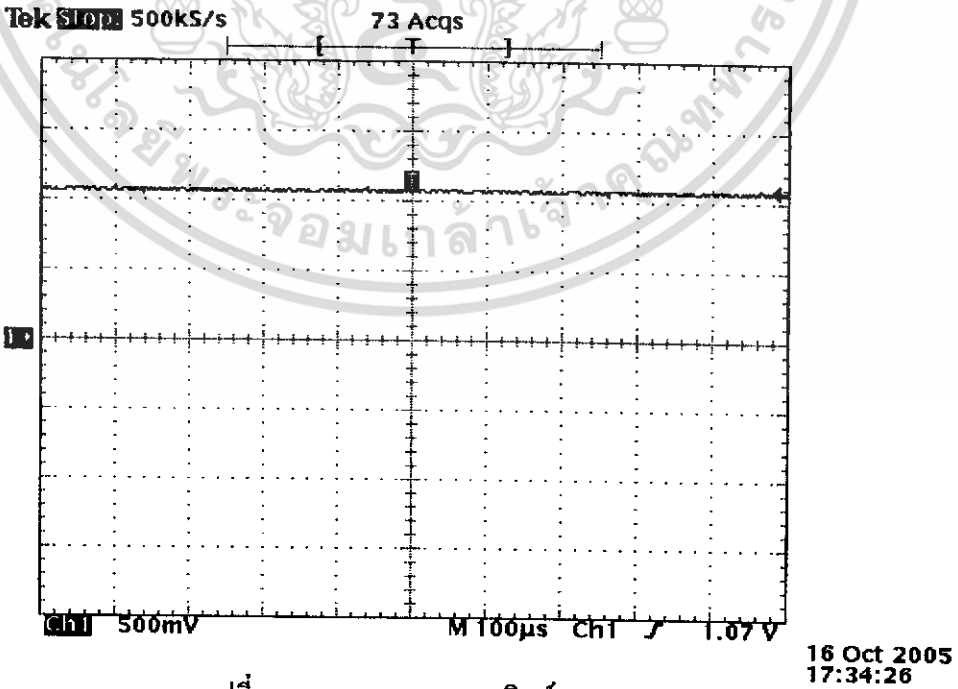
การทดลองและผลการทดลอง

4.1 วงจรสวิทช์

ต่อวงจรตามรูปที่ 3.4 โดยทดสอบเอาท์พุทที่ ขา 3, 9, 12, 15 ของไอซี CD4042B ในสภาวะปกติ จะได้เอาท์พุทเท่ากับ 5 โวลต์ จากนั้นทดสอบเมื่อกดสวิทช์โดยทดสอบเอาท์พุท จะได้เอาท์พุทเท่ากับ 1.07 โวลต์ ดังรูป



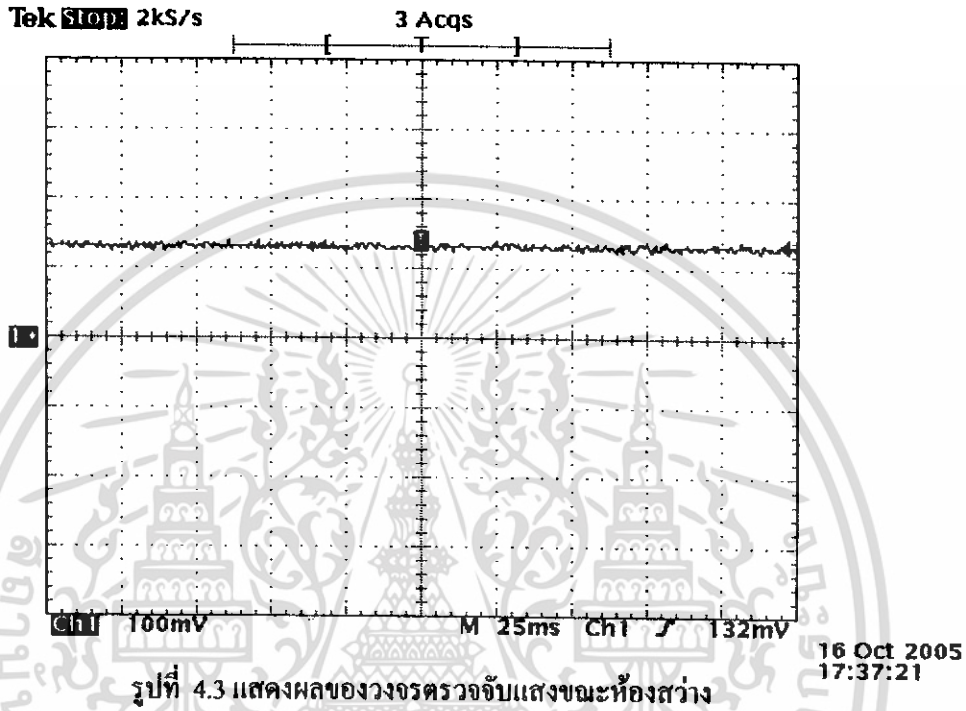
รูปที่ 4.1 แสดงผลของวงจรสวิทช์ในสถานะปกติ



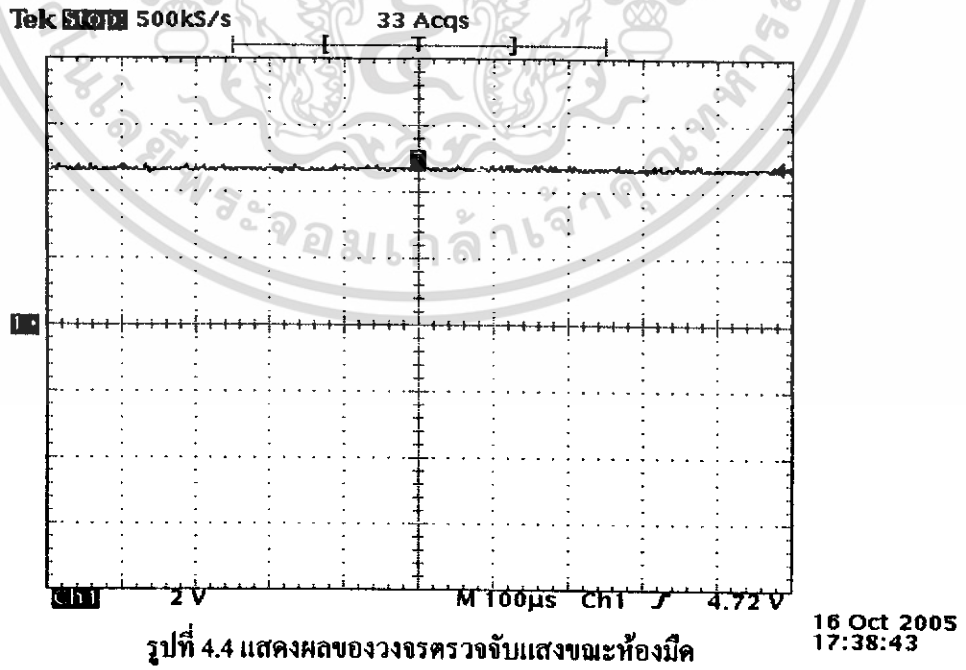
รูปที่ 4.2 แสดงผลของวงจรสวิทช์ถูกกด

4.2 วงจรตรวจจับทางแสง

ต่อวงจรดังรูปที่ 3.7 ในสภาวะปกติในห้องที่สว่างจะได้เอาท์พุทที่ขา 7 ของไอซี HA7741 จะได้เอาท์พุทเท่ากับ 0.13 โวลต์ซึ่งจะมีสถานะเป็น "Low" เมื่อห้องมืดลงจะได้เอาท์พุทเท่ากับ 4.72 โวลต์ ซึ่งจะมีสถานะเป็น "High" ผลการทดลองเป็นดังรูปที่ 4.3 และ 4.4



รูปที่ 4.3 แสดงผลของวงจรตรวจจับแสงขณะห้องสว่าง



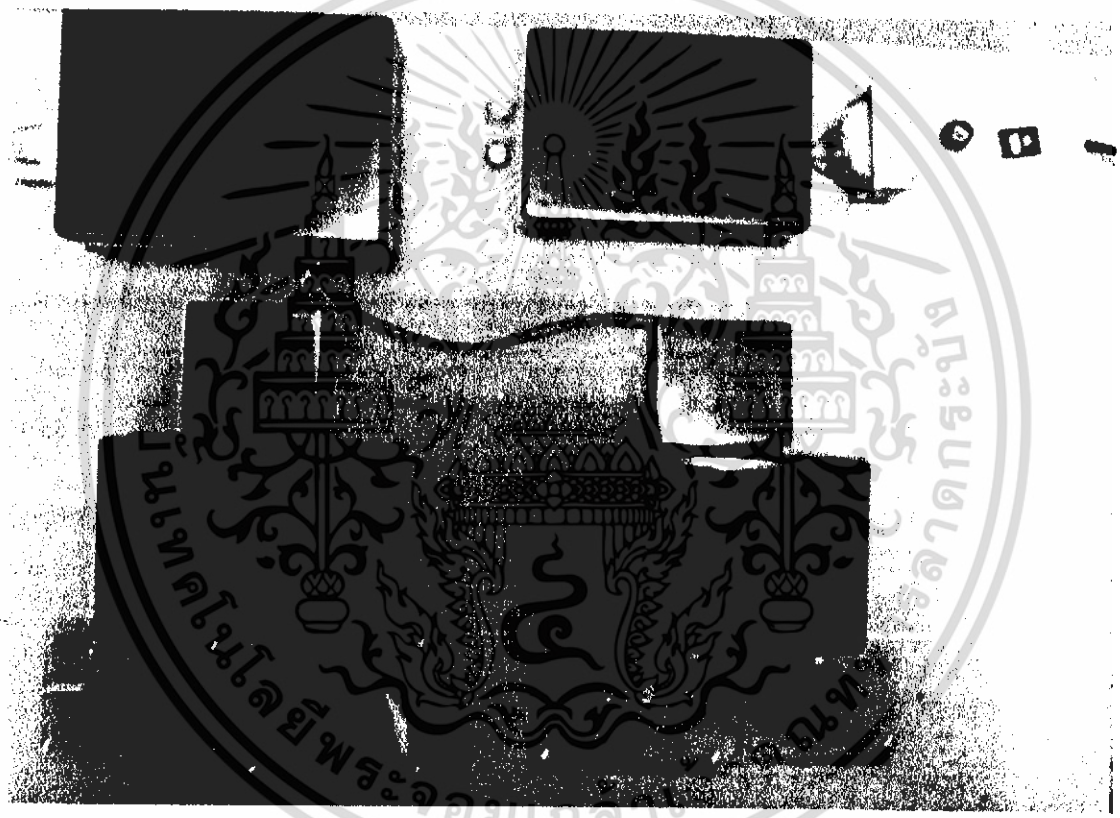
รูปที่ 4.4 แสดงผลของวงจรตรวจจับแสงขณะห้องมืด

4.3 แสดงการติดต่อระหว่างวงจรกับคอมพิวเตอร์โดยผ่านทางจาวา

การแสดงการติดต่อระหว่างวงจรกับคอมพิวเตอร์โดยผ่านทางจาวา โดยใช้โปรแกรม Free Serial Port Monitor ในการตรวจจับข้อมูลที่ผ่านเข้า-ออก พอร์ตคอม 1 ซึ่งโปรแกรมนี้จะแบ่งหน้าจอออกเป็น 2 ส่วน คือ

1. ส่วนที่อยู่ด้านบนแสดงการรับข้อมูลจากภายนอกผ่านพอร์ตคอม 1 มายังคอมพิวเตอร์
2. ส่วนที่อยู่ด้านล่างแสดงการส่งข้อมูลที่ออกจากคอมพิวเตอร์ผ่านพอร์ตคอม 1 ไปยังภายนอก

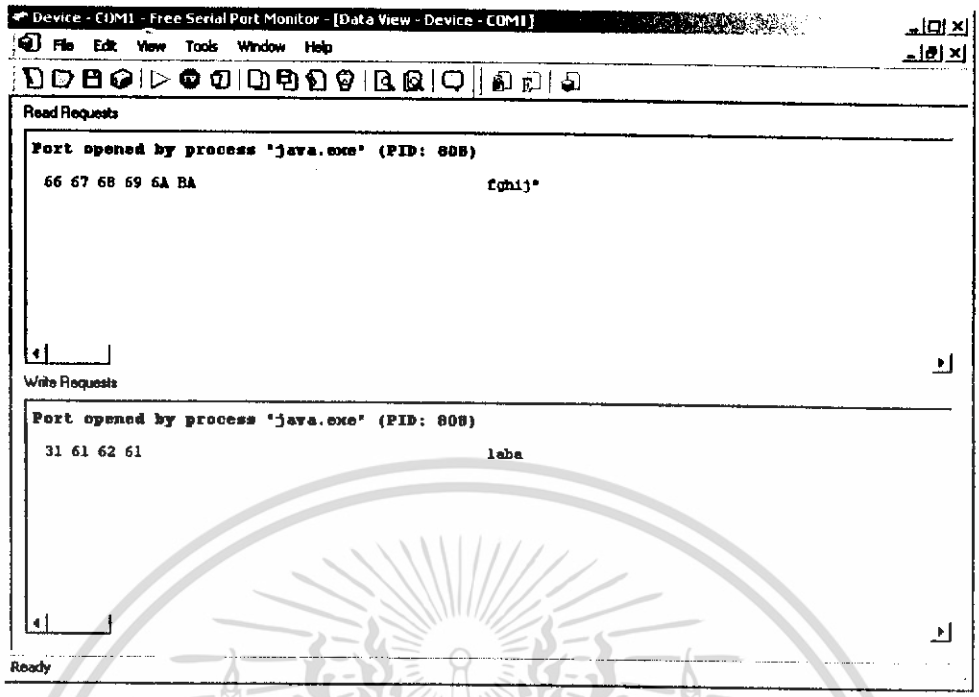
ซึ่งโครงงานนี้จะแบ่งไมโครคอนโทรลเลอร์เป็น 2 ชุด โดยแต่ละชุดจะมีสวิทช์ทั้งหมด 4 ตัวและวงจรตรวจจับแสง 1 วงจร ดังรูปที่ 4.5



รูปที่ 4.5 แสดงวงจรของโครงงาน

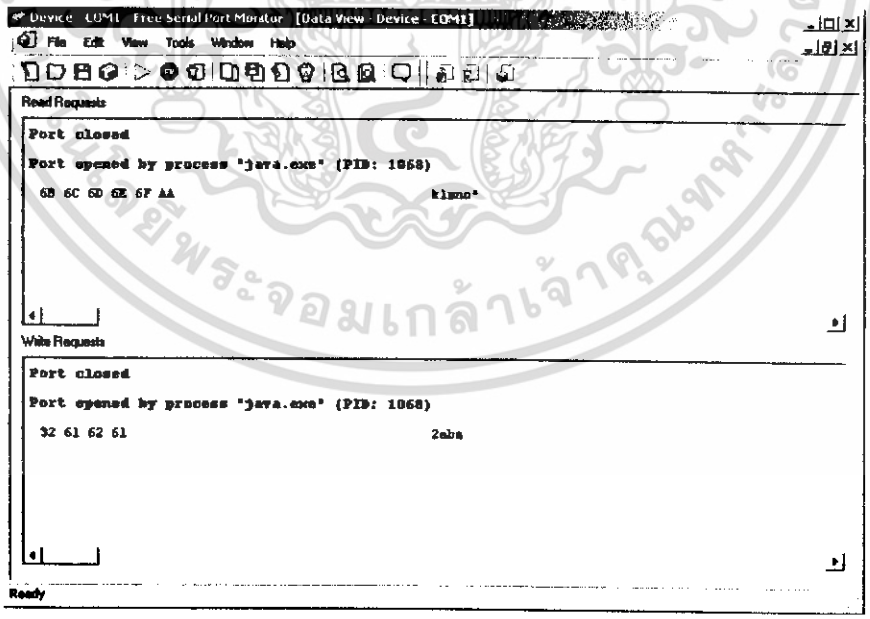
โดยคำสั่ง 1a เป็นการขอสถานะของวงจรไมโครคอนโทรลเลอร์ชุดที่ 1 และ 2a เป็นการขอสถานะของวงจรไมโครคอนโทรลเลอร์ชุดที่ 2 เมื่อรันโปรแกรมจาวา จาวาจะส่งข้อมูล 1a ไปยังไมโครคอนโทรลเลอร์ชุดที่ 1 แล้วจะรับข้อมูลจากไมโครคอนโทรลเลอร์ซึ่งทางไมโครคอนโทรลเลอร์จะส่งข้อมูลสถานะของวงจรมาให้ เมื่อคอมพิวเตอร์ได้รับสถานะจากไมโครคอนโทรลเลอร์

ซึ่งเป็นสถานะที่ไม่ถูกกดคือตัวอักษร “fghij” แล้วจาวาจะทำการส่งข้อมูล b ไปเพื่อที่จะเตรียมตัวทำการขอสถานะของวงจรไมโครคอนโทรลเลอร์ชุดที่ 2 ดังรูปที่ 4.6



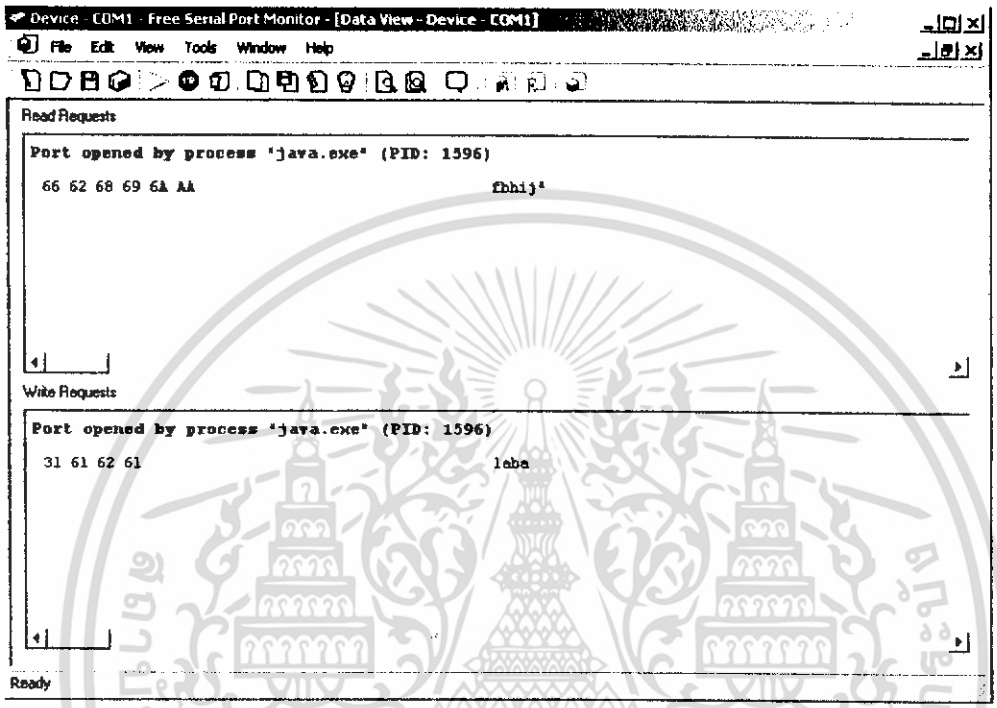
รูปที่ 4.6 แสดงการรับส่งข้อมูลผ่านจาวาโดยใช้ Free Serial Port Monitor

แล้วทำการการเช็คว่าได้ส่งข้อมูล 1a หรือ 2a ออกไป ถ้าส่ง 1a ไปแล้วก็จะทำการส่งข้อมูล 2a เพื่อขอสถานะของวงจรไมโครคอนโทรลเลอร์ รูปที่ 4.7



รูปที่ 4.7 แสดงสถานะของวงจรไมโครคอนโทรลเลอร์ชุดที่ 2

แต่ถ้าเมื่อมีการส่งข้อมูล 2a ออกไปแล้วก็จะส่ง 1a ออกไปและเมื่อมีการสวิทช์ของวงจรไมโครคอนโทรลเลอร์ชุดที่ 1 ถูกกด โดยสวิทช์ที่ถูกกดคือ สวิทช์ที่ 2 ข้อมูลสถานะของวงจรไมโครคอนโทรลเลอร์ที่คอมพิวเตอร์ได้รับจะเห็นว่าข้อมูลสถานะเปลี่ยนไปเป็น "fbhij" ดังรูปที่ 4.8



รูปที่ 4.8 รูปที่แสดงสถานะของวงจรไมโครคอนโทรลเลอร์ที่สวิทช์ตัวที่ 2 ถูกกด

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 สรุปผลการทดลอง

จากการออกแบบ และใช้งานวงจร พบว่าสามารถส่งข้อมูลระหว่างคอมพิวเตอร์และไมโครคอนโทรลเลอร์ได้

5.2 ปัญหาที่เกิดขึ้นและแนวทางแก้ไข

1. เราสามารถส่งข้อมูลได้แต่เวลาดูข้อมูลแล้วจะดูยาก ควรใช้รูปภาพในการแสดงผลจะทำให้ดูสถานะของวงจรได้ง่าย

2. เนื่องจากเป็นการส่งแบบ Half-Duplex ทำให้เวลาส่งข้อมูลไม่สามารถส่งพร้อมกันได้เรา ซึ่งมีวิธีแก้ไข 2 ทาง

- ทางซอฟต์แวร์ คือ เขียนโปรแกรมบนให้เช็คว่าสายข้อมูลถูกใช้ก่อนทำการรับ-ส่งข้อมูลทั้งบนไมโครคอนโทรลเลอร์และคอมพิวเตอร์
- ทางฮาร์ดแวร์ คือ ให้ใช้ขา RTS เป็นตัวกำหนดการรับ-ส่งข้อมูลหรือไม่ก็เปลี่ยน IC ที่ใช้ในการส่งข้อมูลแบบ RS-485 ซึ่งมี IC ที่สามารถส่งข้อมูลเป็นแบบ Full-Duplex ได้

5.3 ข้อเสนอแนะ

จากโครงงานนี้ มีแค่วงจรสวิทช์กับวงจรตรวจจับแสง ถ้าเราเพิ่มวงจรอื่นเพิ่มขึ้นไปอีกจะทำให้ประสิทธิภาพของระบบรักษาความปลอดภัยดีขึ้น

บรรณานุกรม

1. ดร.อิน มาน หยาง , ทศพล ปราชญสมพงษ์ , "การออกแบบระบบไมโครคอนโทรลเลอร์ตระกูล 8051" ศูนย์หนังสือมหาลัยรังสิต , 2539
2. รศ.สมยศ จุณณะปิยะ , "การประยุกต์ใช้งานไมโครคอนโทรเลอร์ ตระกูล MCS—51" คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง 2541

ภาคผนวก

ภาคผนวก ก.

โปรแกรมที่เขียนด้วยภาษา Assembly

โปรแกรมของไมโครคอนโทรเลอร์ตัวที่ 1

```
            ORG 0000H
            LJMP MAIN1

            ORG 0023H
            LJMP RECIVE

MAIN1:      ORG 0100H
            MOV SCON,#50H
            MOV PCON,#00H
            MOV TMOD,#20H
            MOV TH1,#0FDH
            MOV P1,#0FFH
            MOV IE,#90H
            SETB TR1

MAIN:       MOV A,P1
            ORL A,#0FEH
            CJNE A,#0FEH,NO1
            MOV 30H,#61H
            SJMP CHECK
NO1:        MOV 30H,#66H
CHECK:      MOV A,P1
            ORL A,#0FDH
            CJNE A,#0FDH,NO2
            MOV 31H,#62H
            SJMP CHECK1
NO2:        MOV 31H,#67H
CHECK1:     MOV A,P1
            ORL A,#0FBH
            CJNE A,#0FBH,NO3
            MOV 32H,#63H
            SJMP CHECK2
NO3:        MOV 32H,#68H
CHECK2:     MOV A,P1
            ORL A,#0F7H
            CJNE A,#0F7H,NO4
            MOV 33H,#64H
            SJMP CHECK3
NO4:        MOV 33H,#69H
CHECK3:     MOV A,P1
            ORL A,#0EFH
            CJNE A,#0EFH,NO5
            MOV 34H,#65H
            LJMP MAIN
NO5:        MOV 34H,#6AH
            LJMP MAIN

RECIVE:     CLR RI
            PUSH ACC
            MOV A,SBUF
            CJNE A,#31H,END1
WAIT:       JNB RI,WAIT
            CLR RI
            MOV A,SBUF
            CJNE A,#61H,SENT1
            MOV R1,#06H
```

```
            MOV R0,#30H
SENT:       MOV A,@R0
            MOV SBUF,A
WAIT1:      JNB TI,WAIT1
            CLR TI
            INC R0
            DJNZ R1,SENT
            SJMP WAIT
SENT1:      CJNE A,#62H,WAIT
            POP ACC
            RETI
END1:       POP ACC
            RETI
```

ภาคผนวก ข.

โปรแกรมของไมโครคอนโทรเลอร์ตัวที่ 2

```
                ORG 0000H
                LJMP MAIN1

                ORG 0023H
                LJMP RECIVE

MAIN1:          ORG 0100H
                MOV SCON,#50H
                MOV PCON,#00H
                MOV TMOD,#20H
                MOV TH1,#0FDH
                MOV P1,#0FFH
                MOV IE,#90H
                SETB TR1
MAIN:           MOV A,P1
                ORL A,#0FEH
                CJNE A,#0FEH,NO1
                MOV 30H,#70H
                SJMP CHECK
NO1:            MOV 30H,#6BH
CHECK:          MOV A,P1
                ORL A,#0FDH
                CJNE A,#0FDH,NO2
                MOV 31H,#71H
                SJMP CHECK1
NO2:            MOV 31H,#6CH
CHECK1:         MOV A,P1
                ORL A,#0FBH
                CJNE A,#0FBH,NO3
                MOV 32H,#72H
                SJMP CHECK2
NO3:            MOV 32H,#6DH
CHECK2:         MOV A,P1
                ORL A,#0F7H
                CJNE A,#0F7H,NO4
                MOV 33H,#73H
                SJMP CHECK3
NO4:            MOV 33H,#6EH
CHECK3:         MOV A,P1
                ORL A,#0EFH
                CJNE A,#0EFH,NO5
                MOV 34H,#74H
                LJMP MAIN
NO5:            MOV 34H,#6FH
                LJMP MAIN

RECIVE:         CLR RI
                PUSH ACC
                MOV A,SBUF
                CJNE A,#32H,END1
WAIT:           JNB RI,WAIT
                CLR RI
                MOV A,SBUF
                CJNE A,#61H,SENT1
                MOV R1,#06H
                MOV R0,#30H
SENT:           MOV A,@R0
```

ภาคผนวก ค.

โปรแกรมที่เขียนด้วยภาษา Java

โปรแกรมที่ติดตั้งบน Computer (File SimpleRead.java)

```
import java.io.*;
import java.util.*;
import javax.comm.*;

public class SimpleRead implements Runnable, SerialPortEventListener
{
    static CommPortIdentifier portId;
    static Enumeration portList;
      public String ACK1 = "1",ACK2 = "2",EOF = "b",SentD = "a";
      public int count;
    InputStream inputStream;
    static SerialPort serialPort;
    static OutputStream outputStream;
      int c[] = {1, 1, 1, 1, 1};
      int d[] = {1, 1, 1, 1, 1};
      int n;
      int s = 0;
      int j = 0;
      int t = 0;
      String msg,msg2;
      String a1[] = {"a", "b" , "c", "d", "j"};
      String a2[] = {"p", "q" , "r", "s", "o"};
    Thread readThread;
      public static void main(String[] args)
      {
                  SimpleRead reader = new SimpleRead();


      }
      public static void Write(byte[] A) {      //Write Byte Function
            portList = CommPortIdentifier.getPortIdentifiers();
        while (portList.hasMoreElements()) {
            portId = (CommPortIdentifier) portList.nextElement();
            if (portId.getPortType() ==
CommPortIdentifier.PORT_SERIAL) {
            if (portId.getName().equals("COM1")) {
            try {
                  serialPort = (SerialPort) portId.open("RS232",
3000);
            } catch (PortInUseException e) {}
            try {
                  outputStream = serialPort.getOutputStream();
                              } catch (IOException e) {}
            try {

                        serialPort.setSerialPortParams(9600
                                    serialPort.DATABITS_8,
                                    serialPort.STOPBITS_1,
                                    serialPort.PARITY_NONE);
            } catch (UnsupportedCommOperationException e) {}
            try {
                  outputStream.write(A);

            } catch (IOException e) {}
                        }
            }
```

```
        }
      }
        public SimpleRead() {
            count = 0;
            Write(ACK1.getBytes());
            Write(SentD.getBytes());
            for(int p=0;p<500000;p++){}
            count++;
      portList = CommPortIdentifier.getPortIdentifiers();
        while (portList.hasMoreElements()) {
            portId = (CommPortIdentifier) portList.nextElement();
            if (portId.getPortType() ==
CommPortIdentifier.PORT_SERIAL) {
                        if (portId.getName().equals("COM1")) {
        try {
            serialPort = (SerialPort) portId.open("RS232", 3000);
        } catch (PortInUseException e) {}
        try {
            inputStream = serialPort.getInputStream();
        } catch (IOException e) {}
      try {
            serialPort.addEventListener(this);
      } catch (TooManyListenersException e) {}
        serialPort.notifyOnDataAvailable(true);
        try {
            serialPort.setSerialPortParams(9600,
                SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,
                SerialPort.PARITY_NONE);
        } catch (UnsupportedCommOperationException e) {}
        readThread = new Thread(this);
        readThread.start();
                    }
            }
        }
    }

    public void run() {}
            public void serialEvent(SerialPortEvent event) {
            switch(event.getEventType()) {
        case SerialPortEvent.BI:
        case SerialPortEvent.OE:
        case SerialPortEvent.FE:
        case SerialPortEvent.PE:
        case SerialPortEvent.CD:
        case SerialPortEvent.CTS:
        case SerialPortEvent.DSR:
        case SerialPortEvent.RI:
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
            break;
        case SerialPortEvent.DATA_AVAILABLE:
            byte[] readBuffer = new byte[6];

            try {
                while (inputStream.available() > 0) {
                    int numBytes = inputStream.read(readBuffer);
                }
                String str = new String(readBuffer);
                        int a = str.length();
                        String num[] = new String[a];
                        for(int i=0;i<str.length();i++)
```

```
                        {
                              num[i] = str.substring(i,i+1);
                              System.out.print(num[i]+" ");


                        }
                        System.out.println();
                        sent m = new sent();

if(j==0){
          for(n=0;n<5;n++)
          {
                if(num[n].equals(a1[n]))
                {
                        if(c[n]==1)
                        {
                        if(num[0].equals("a"))
                        {
                        m.text = m.text+" 00000";
                        s = 1;
                        c[0]=0;
                        }
                        if(num[1].equals("b"))
                        {
                        m.text = m.text+" 11111";
                        s = 1;
                        c[1]=0;
                        }
                        if(num[2].equals("c"))
                        {
                        m.text = m.text+" 22222";
                        s = 1;
                        c[2]=0;
                        }
                        if(num[3].equals("d"))
                        {
                        m.text = m.text+" 33333";
                        s = 1;
                        c[3]=0;
                        }
                        if(num[4].equals("j"))
                        {
                        m.text = m.text+" 44444";
                        s = 1;
                        c[4]=0;
                        }
                        }else{}
                }
                else
                {
                c[n] = 1;
                }

          }
          j = 1;
}
    else if (j==1)
    {

          for(n=0;n<5;n++)
          {
```

```
                if(num[n].equals(a2[n]))
                {
                        if(d[n]==1)
                        {
                        if(num[0].equals(a2[0]))
                        {
                        m.text = m.text+" 55555";
                        s = 2;
                        d[0]=0;
                        }
                        if(num[1].equals(a2[1]))
                        {
                        m.text = m.text+" 66666";
                        s = 2;
                        d[1]=0;
                        }
                        if(num[2].equals(a2[2]))
                        {
                        m.text = m.text+" 77777";
                        s = 2;
                        d[2]=0;
                        }
                        if(num[3].equals(a2[3]))
                        {
                        m.text = m.text+" 88888";
                        s = 2;
                        d[3]=0;
                        }
                        if(num[4].equals(a2[4]))
                        {
                        m.text = m.text+" 99999";
                        s = 2;
                        d[4]=0;
                        }
                        }else{}
                }
                else
                {
                d[n] = 1;
                }
        }
        j=0;
}


if(s==1)
{
m.mail();
m.text = "";
}
s=0;



                        Write(EOF.getBytes());
                        if (count%2==1)
                        {
                                Write(ACK2.getBytes());
                                Write(SentD.getBytes());
                                t = 1;
```

```
                        for(int p=0;p<500000;p++){for(int
q=0;q<1000;q++){}}
                    }
                    else
                        {
                        Write(ACK1.getBytes());
                        Write(SentD.getBytes());

                        for(int p=0;p<500000;p++){for(int
q=0;q<1000;q++){}}
                        }
                    count++;
            } catch (IOException e) {}
                break;
             }
        }
}
```

ภาคผนวก ง.

โปรแกรมที่ติดตั้งบน Computer (File Sent.java)

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import javax.activation.*;
public class sent{
      static String text = "";
      public static void mail(){
            String input = "gankung@gmail.com";
            String host = "mail.kmitl.ac.th";
            String sender = "gankung@hotmail.com";
            String receiver = input;
            String subject = "Test ja";
            Properties p = new Properties();
            p.put("mail.smtp.host",host);
            p.put("mail.debug",true);
            Session ses = Session.getInstance(p,null);

try{
                  Message msg = new MimeMessage(ses);
                  msg.setFrom(new InternetAddress(sender));
                  msg.setRecipient(Message.RecipientType.TO,new
InternetAddress(receiver));
                  msg.setSubject(subject);
                  msg.setSentDate(new Date());
                  msg.setText(text);
                  Transport.send(msg);
            }catch(Exception e){
                  e.printStackTrace();
            }
      }
}
```