การประยุกต์ใช้งาน RS-485

RS-485 APPLICATION

โดย

นายธนายุทธ วิริยะพงศ์

นายสิทธิพงษ์ เลิศวจนะ

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

การประยุกต์ใช้งาน RS-485

RS-485 APPLICATION

นายธนายุทธ วิริยะพงศ์ 39014221

นายสิทธิพงษ์ เลิศวจนะ 39014573

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รศ.ดร.มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

ปริญญานิพนธ์ ปีการศึกษา 2542

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งาน RS-485

ผู้จัดทำ

1. นายธนายุทธ วิริยะพงศ์ 39014221
2. นายสิทธิพงษ์ เลิศวจนะ 39014573

.....

(รศ.ดร.มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

การประยุกต์ใช้งาน RS-485

นายธนายุทธ วิริยะพงศ์
นายสิทธิพงษ์ เลิศวจนะ
รศ.ดร.มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

ในปัจจุบัน เทคโนโลยีในการสื่อสารข้อมูลนั้นได้รับการพัฒนาขึ้นอย่างรวดเร็ว เนื่องจากความต้องการที่จะใช้ข้อมูลโดยมีการส่งและรับข้อมูลที่ถูกต้องแม่นยำ รวดเร็ว และติดต่อกันได้แม้จะอยู่ห่างไกลก็ตาม ทั้งเพื่อผลทางเศรษฐกิจ การเมือง หรือการพัฒนาทางวิทยาศาสตร์และเทคโนโลยี ล้วนแต่จำเป็นที่จะต้องใช้การสื่อสารข้อมูลทั้งสิ้น สำหรับโครงงานนี้จะเป็นการศึกษาถึงระบบการสื่อสารข้อมูลแบบอนุกรม ตามมาตรฐาน RS-485 และนำมาประยุกต์ใช้งานในการควบคุมระบบต่างๆภายในตัวอาคารเพื่อเพิ่มประสิทธิภาพและความสะดวกสบายในชีวิตประจำวัน โดยใช้คอมพิวเตอร์ส่วนบุคคลเป็นตัวควบคุมหลักเพื่อที่จะตรวจสอบสถานะของอุปกรณ์ต่างๆและบอกแก่เจ้าของโดยผ่านทางเครือข่ายทางอินเทอร์เน็ตโดยใช้ระบบเพจเจอร์เตือน

RS-485 APPLICATION

Mr. Tanayoot Wiriyapong
Mr. Sittipong Lertvajana
Assoc.Prof.Dr.Manas Sangworasil Advisor
1998

ABSTRACT

In the recent day more and more business government agencies and academic departments require a fast and more realiable transmission of information and data. Thus , the field of data communications have become a rapidly growing area. The main purpose of this project is to study the serial transmission by RS-485 standard, and its application to control the building system by control the data transmission between micro computer (or personal computer), that can check status and tell the user via the internet

สารบัญ

เรื่อง **หน้า**

สารบัญ

(ต่อ)

สารบัญ

(ต่อ)

สารบัญภาพ

สารบัญภาพ

(ต่อ)

บทที่ 1
บทนำ

1.1 แนวความคิดของระบบควบคุม

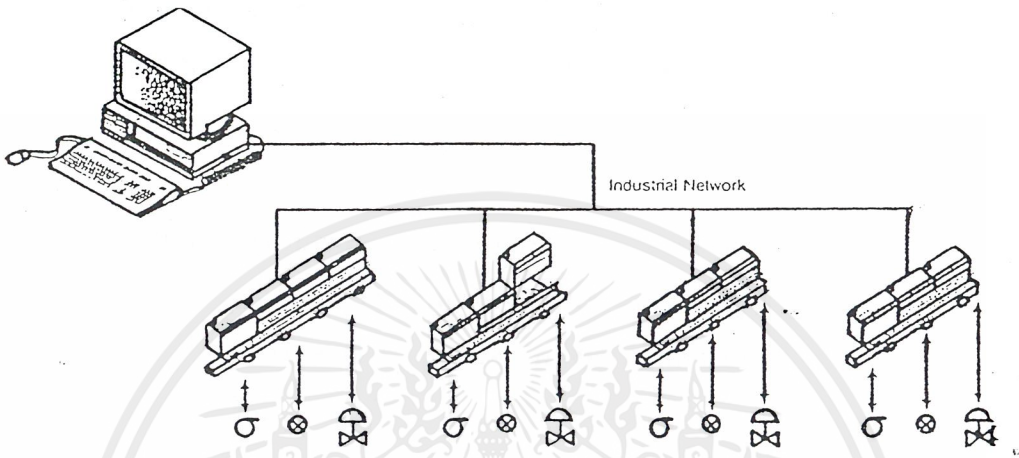
ในปัจจุบันการสื่อสารและการควบคุมอุปกรณ์มีความจำเป็นมากขึ้นเนื่องจากจำนวนของอุปกรณ์ที่มากขึ้น ความต้องการในการใช้งาน ความสะดวกสบาย ความรวดเร็วในการทำงานที่มีเพิ่มมากขึ้น แนวทางที่จะตอบสนองความต้องการดังกล่าวแนวทางทางหนึ่งคือการสร้างระบบควบคุมที่สามารถทำงานได้ดี กล่าวคือ ใช้งานได้ง่าย มีความยืดหยุ่น มีความผิดพลาดเกิดขึ้นน้อย อีกทั้งยังต้องสามารถขยายระบบที่ต้องการควบคุมออกไปได้ จึงทำให้เกิดแนวคิดที่จะสร้างระบบควบคุมโดยการประยุกต์ใช้งาน RS-485 ขึ้น

การกำหนดขอบเขตของระบบควบคุมนั้นจะถือว่าสิ่งที่ต้องการควบคุมทั้งหมดนั้นเป็นส่วนหนึ่งของระบบควบคุม ในที่นี้จะขอยกตัวอย่างเป็นห้องหลายๆห้องในอาคารชั้นหนึ่ง ซึ่งภายในห้องๆหนึ่งนั้นจะมีอุปกรณ์ที่ต้องการควบคุมอาทิ หลอดไฟ เครื่องปรับอากาศ และเครื่องใช้ไฟฟ้าอื่น ๆ โดยแต่ละอุปกรณ์ที่ต้องการควบคุม(Controlled Device: CD) จะต่อเข้ากับตัวลูกข่าย (Module Box : MB) ที่ใช้ควบคุมอุปกรณ์นั้น และตัวลูกข่ายหลาย ๆ ตัวจะต่อรวมเข้ากับตัวควบคุม (Master Control Box : MCB) ซึ่งตัวควบคุมนี้จะทำหน้าที่เสมือนกับเป็นสถานีจ่ายงานย่อย เนื่องจากระบบควบคุมนี้สามารถเชื่อมต่อตัวควบคุมได้มากกว่าหนึ่งตัวทำให้สามารถแบ่งการควบคุมอุปกรณ์ออกเป็นกลุ่มย่อย(Section, Zone) ทำให้ง่ายและสะดวกต่อการควบคุมอุปกรณ์ที่ต้องการควบคุมทีเดียวทั้งหมด

แนวคิดของระบบควบคุมสามารถแสดงให้เห็นได้ดังรูปที่ 1.1 โดยจะมีการแบ่งห้องต่าง ๆ ในชั้นของตัวอาคารออกเป็นกลุ่มซึ่งอุปกรณ์ที่ต้องการควบคุมหนึ่งตัวจะทำการเชื่อมต่อเข้ากับตัวลูกข่ายหนึ่งตัว และตัวลูกข่ายจะถูกแบ่งเป็นกลุ่มตามการเชื่อมต่อเข้ากับตัวควบคุม ดังจะเห็นได้จากในรูปที่มีการแบ่งห้องในชั้นออกเป็น 4 กลุ่ม คือ

1. กลุ่มตัวควบคุมที่ 1(MCB-1 Area) เชื่อมต่อกับตัวลูกข่ายที่ 1-4 (MB 1-4)
2. กลุ่มตัวควบคุมที่ 2(MCB-2 Area) เชื่อมต่อกับตัวลูกข่ายที่ 5-7 (MB 5-7)
3. กลุ่มตัวควบคุมที่ 3(MCB-3 Area) เชื่อมต่อกับตัวลูกข่ายที่ 8-9 (MB 8-9)
4. กลุ่มตัวควบคุมที่ 4(MCB-4 Area) เชื่อมต่อกับตัวลูกข่ายที่ 10-13 (MB 10-13)

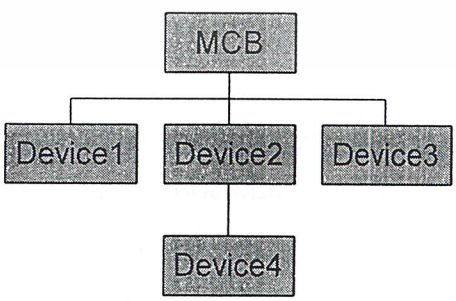
โดยอุปกรณ์ที่ต้องการควบคุมจะเป็นอุปกรณ์ประเภทรับข้อมูล(Input) หรือแสดงผล(Output) ก็ได้เนื่องจากระบบควบคุมนี้เป็นการสื่อสารแบบสองทางในคนละเวลา(Half-duplex) ทำให้ระบบควบคุมนี้สามารถรับและส่งข้อมูลได้บนคู่สายคู่หนึ่ง



รูปที่ 1.1 แสดงแนวความคิดของระบบควบคุม

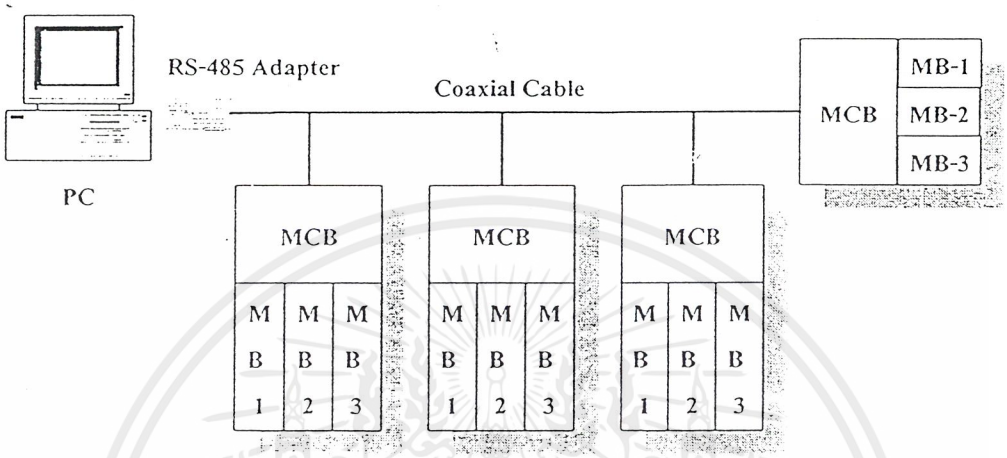
1.2 โครงสร้างของระบบควบคุม

การทำงานของระบบควบคุมทั้งหมดจะถูกควบคุมโดยคอมพิวเตอร์ส่วนบุคคล (Personal Computer : PC) โดยจะเป็นการควบคุมแบบรวมศูนย์(Centralize Control) ซึ่งคอมพิวเตอร์ส่วนบุคคลทำหน้าที่เป็นตัวหลักในการติดต่อกับผู้ใช้งาน (User Interface) และควบคุมระบบโดยการจ่ายงานให้กับตัวควบคุมที่เชื่อมต่อกันเป็นเครือข่าย

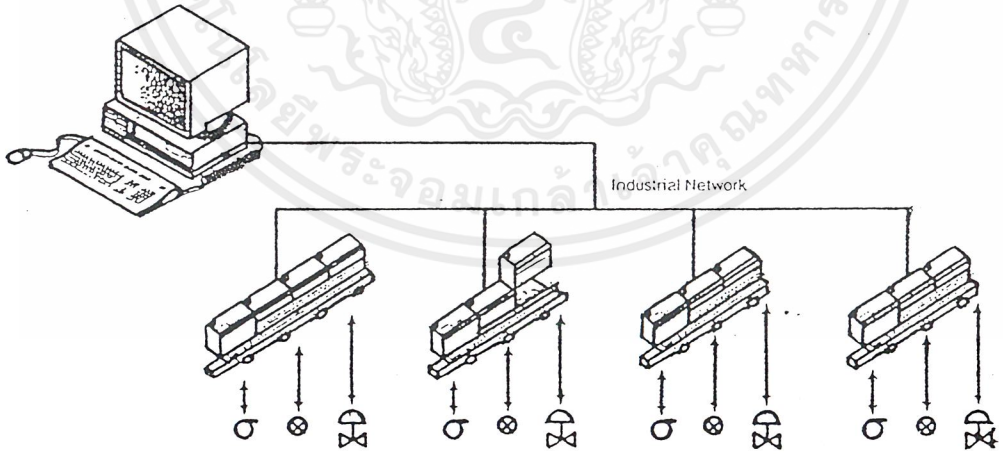


รูปที่ 1.2 แสดงโครงสร้างของระบบควบคุม

ตัวอย่างการประยุกต์ใช้งานแสดงดังรูปที่ 1.5 และ 1.6



รูปที่ 1.5 แสดงการประยุกต์ใช้งานในการควบคุมระบบงานทั่วๆไป



รูปที่ 1.6 แสดงการประยุกต์ใช้งานในการควบคุมระบบทางอุตสาหกรรม

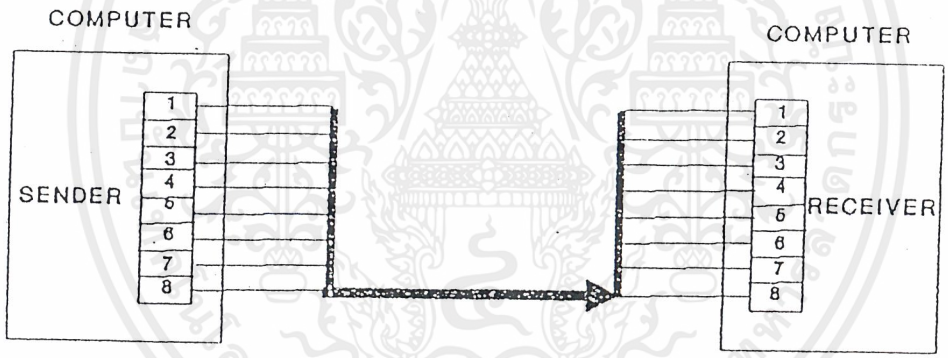
บทที่2
พื้นฐานในการออกแบบระบบควบคุม

2.1 ลักษณะของการสื่อสาร

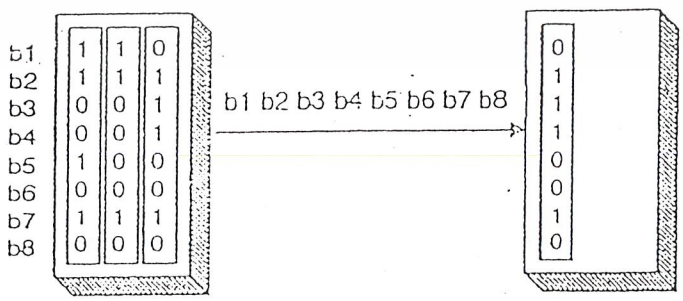
ในการสื่อสารหรือการส่งข้อมูลโดยใช้คอมพิวเตอร์ส่วนบุคคลนั้นมีรูปแบบในการสื่อสารที่สำคัญอยู่ 2 แบบคือ

2.1.1 การสื่อสารแบบอนุกรม(Serial Communication)

เป็นการสื่อสารโดยการส่งข้อมูลทีละบิท(bit)ผ่านสายสัญญาณเส้นเดียวจนครบทั้ง 8บิทหรือไบท์(byte)โดยจะส่งบิทต่ำ(LSB)ออกไปก่อน ซึ่งการสื่อสารและการส่งข้อมูลแบบอนุกรมจะแสดงให้เห็นได้ดังรูป2.1และ2.2



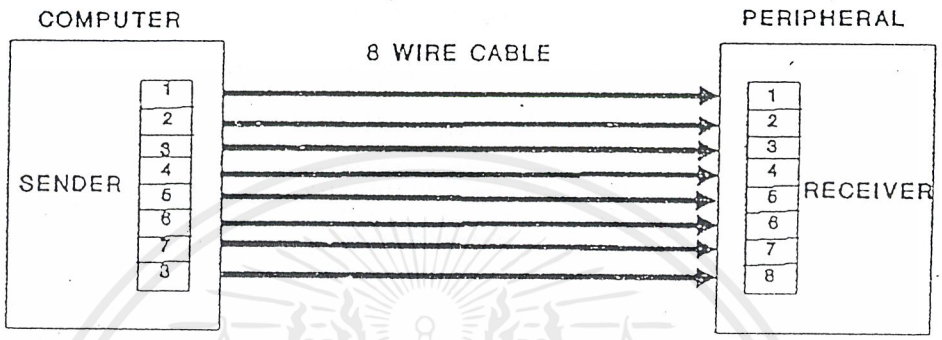
รูปที่ 2.1 แสดงลักษณะการสื่สารแบบอนุกรม



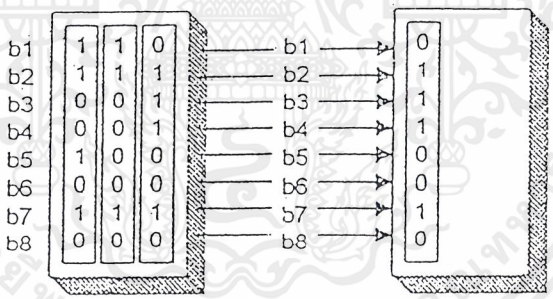
รูปที่ 2.2 แสดงการส่งข้อมูลแบบอนุกรม

2.1.2 การสื่อสารแบบขนาน(Parallel Communication)

เป็นการสื่อสาร โดยการส่งข้อมูลไปทีเดียวพร้อมๆกันทั้ง8บิทผ่านสายสัญญาณทั้ง 8เส้น ซึ่งการสื่อสารและการส่งข้อมูลแบบขนานจะแสดงให้เห็นได้ดังรูป 2.3 และ 2.4



รูปที่ 2.3 แสดงลักษณะการสื่อสารแบบขนาน



รูปที่ 2.4 แสดงการส่งข้อมูลแบบขนาน

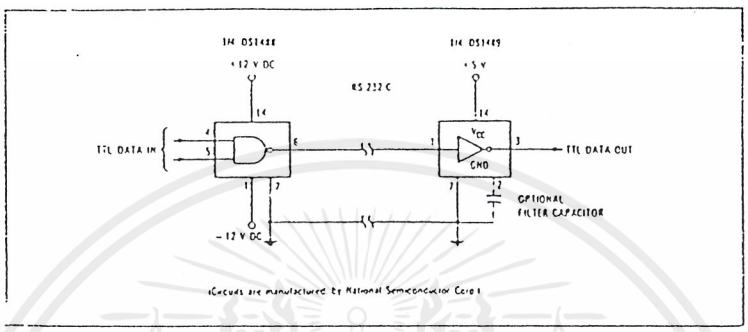
จะเห็นได้ว่าการสื่อสารแบบขนานมีข้อดีคือทำให้สามารถส่งข้อมูลได้ทีละมากๆ และรวดเร็วกว่าการส่งแบบอนุกรม แต่การสื่อสารแบบขนานมีข้อจำกัดคือไม่สามารถส่งข้อมูลในระยะไกลๆได้และยังต้องใช้สายสัญญาณหลายเส้นในการส่งข้อมูลทำให้สิ้นเปลืองกว่าการสื่อสารแบบอนุกรม รวมทั้งทำให้ไม่สะดวกในการใช้งาน

ตัวอย่างของการสื่อสารแบบอนุกรมเช่น การสื่อสารด้วย MODEM และ เมาส์

ตัวอย่างของการสื่อสารแบบขนานเช่น เครื่องพิมพ์(Printer) และ การสื่อสารทางพอร์ทขนาน(ECP Printer Port)เป็นต้น

วงจรแปลงระดับแรงดัน

ตัวอย่างของวงจรที่ใช้แปลงระดับสัญญาณ TTL ไปเป็นระดับสัญญาณที่กำหนดไว้ในมาตรฐาน RS-232-C และแปลงกลับจากระดับแรงดันในมาตรฐาน RS-232-C ไปเป็นระดับสัญญาณ TTL แสดงไว้ในรูปที่ 2.5



รูปที่ 2.5 วงจรขับและรับสัญญาณที่ใช้กับมาตราฐาน RS-232-C

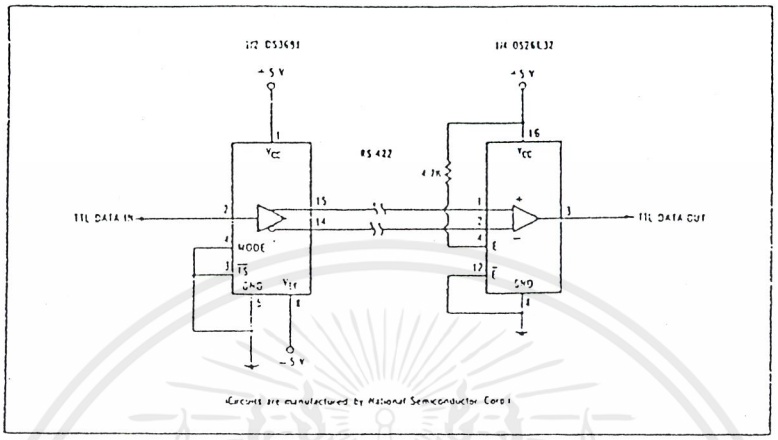
2.2.2 ลักษณะการอินเทอร์เฟสที่ใช้มาตรฐาน RS-423

มาตรฐาน RS-423 เป็นมาตรฐานที่ได้รับการพัฒนามาจากมาตรฐาน RS-232-C อุปกรณ์ที่ผลิตขึ้นมาใหม่ ๆ มักจะใช้การอินเทอร์เฟสแบบนี้ โดยเฉพาะแปกรณ์ที่ต้องการให้อัตราเร็วในการส่งข้อมูลมีค่าสูง

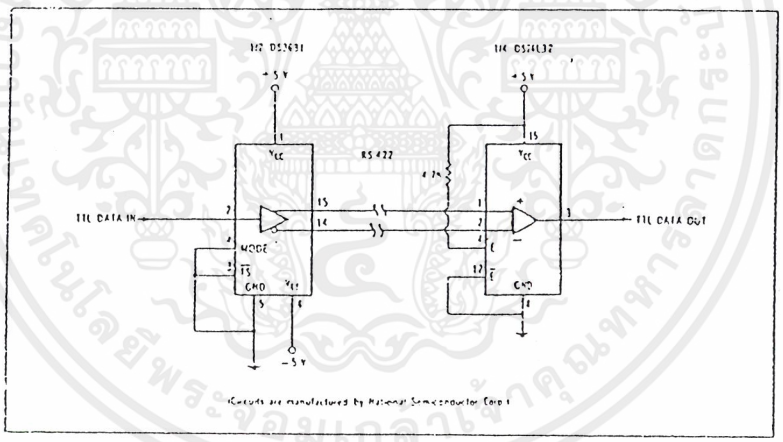
ลักษณะสัญญาณที่ใช้ในการอินเทอร์เฟส

มาตรฐาน RS-423 ใช้สายสัญญาณเส้นเดียวในการส่งสัญญาณ โดยสัญญาณที่ส่ง ๆ ไปได้ทิศทางเดียว อัตราเร็วในการส่งข้อมูลมีค่าสูงถึง 100k bps ที่ระยะห่าง 40 ฟุต ตัวรับข้อมูลเป็นแบบ balanced-1 (ดูรูปที่ 2.6) ดังนั้นตัวรับข้อมูล (receiver) จึงรับข้อมูลแบบขยายความแตกต่างของสัญญาณระหว่างสายกราวนด์กับตัวขับสัญญาณ (driver) การทำเช่นนี้ช่วยแก้ปัญหาในกรณีที่เกิดความแตกต่างระหว่างแรงดันที่ กราวนด์ของตัวรับข้อมูล กับตัวขับสัญญาณ สำหรับการแทนระดับแรงดันนั่น ลอจิก “1” แทนระดับแรงดันที่อยู่ระหว่าง +4 โวลท์ ถึง +6 โวลท์ ส่วนลอจิก “0” แทนระดับแรงดันที่มีค่าระหว่าง -4 โวลท์ ถึง -6 โวลท์

เดียว (single pair) ซึ่งเป็นมาตราฐานใหม่คือ RS-485 คุณสมบัติข้อนี้ทำให้เราสามารถใช้มาตรฐาน RS-485 ในเน็ตเวิร์คที่มีโครงสร้างเป็นแบบ multidrop ซึ่งอุปกรณ์หลาย ๆ ตัวสามารถรับและส่งข้อมูลแบบ half duplex บนสายคู่เดียวได้



รูปที่ 2.7 (ก) วงจรขับและรับสัญญาณที่ใช้มาตรฐาน RS-422

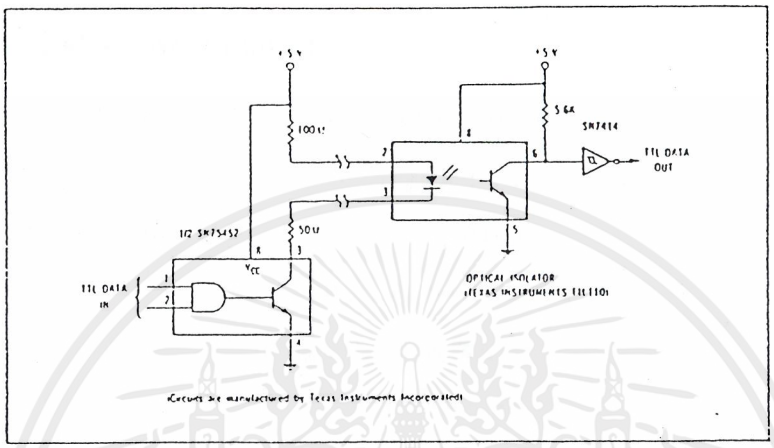


รูปที่ 2.7 (ข) วงจรขับและรับสัญญาณที่ใช้มาตรฐาน RS-485

2.2.4 ลักษณะการอินเทอร์เฟสแบบ current-loop

การอินเทอร์เฟสแบบ current-loop มีข้อดีคือสามารถส่งข้อมูลได้ในระยะทางไกล ๆ และค่าใช้จ่ายในการอินเทอร์เฟสแบบนี้มีราคาไม่สูงนัก หลักการของการอินเทอร์เฟสแบบนี้มีดังนี้ เมื่อลูปถูกปิดวงจร (closed-loop) ระดับแรงดันจะถูกเปลี่ยนเป็นกระแสตามสมการ V=IR เมื่อลูปถูกเปิด วงจรจะไม่มีกระแสไหลในลูป เนื่องจากวงจรของการอินเทอร์เฟสแบบนี้มีค่าอิมพีแดนซ์ต่ำดังนั้นจึงทนต่อสัญญาณรบกวนได้ดี เรามักใช้การอินเทอร์เฟสแบบนี้ในกรณีที่ต้องเดินสายผ่านบริเวณที่มีสัญญาณรบกวนมาก ๆ นอกจากนี้เราสามารถใช้การอินเทอร์เฟสแบบนี้แยกกราวนด์ของระบบ

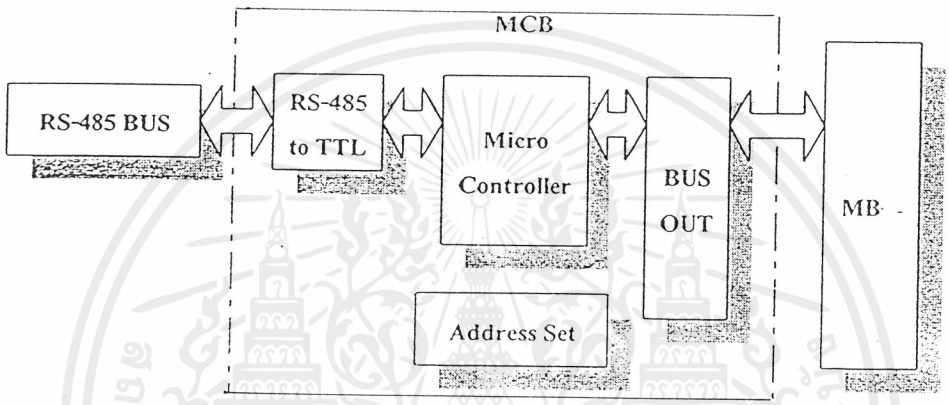
สองระบบออกจากกัน ดังแสดงในรูป 2.8 ซึ่งเป็นการแยก current-loop ของวงจรผลิตสัญญาณ (transmitter) ออกจากวงจรรับสัญญาณ (receiver) วงจรนี้สามารถใช้ส่งข้อมูลได้ในอัตราเร็ว 50k bps โดยระยะทางที่ใช้ส่งข้อมูลมีค่าได้ไม่เกิน 3000 ฟุต ตัวจำกัดระยะทางที่ใช้ในการส่งข้อมูลคือความต้านทานของสายที่ประกอบกันอยู่เป็นลูป ซึ่งไม่ควรมีค่าเกิน 30 โอห์ม (สำหรับวงจรในรูปที่ 2.8)



รูปที่ 2.8 วงจรที่ใช้การอินเทอร์เฟสแบบ current-loop

3.3 โครงสร้างของตัวควบคุม

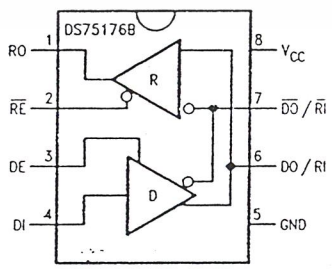
ตัวควบคุมมีหน้าที่แปลงสัญญาณข้อมูลจาก PC ไปยังส่วนแสดงผล และรับข้อมูลจากส่วนรับข้อมูลส่งไปยัง PC ตามมาตรฐาน RS-485 ดังรูป 3.3 แสดงโครงสร้างตัวควบคุม



รูปที่ 3.3 แสดงโครงสร้างตัวควบคุม

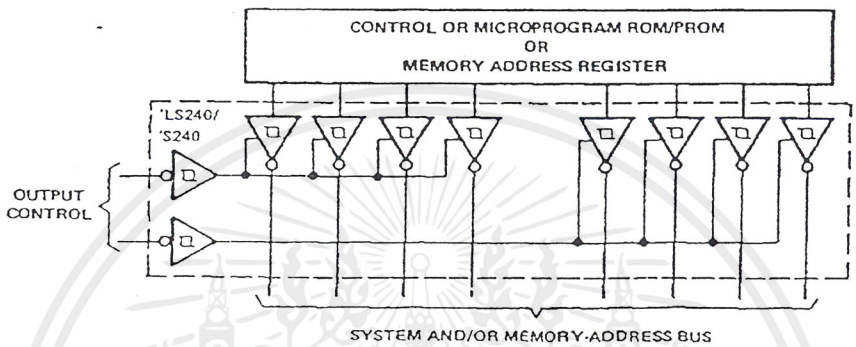
3.4 ส่วนประกอบของตัวควบคุม

1. IC DS 75176B ทำหน้าที่แปลงสัญญาณมาตรฐาน RS-485 กับ RS-232C ระดับมาตรฐาน TTL ที่ 89C51 ทำงานได้



รูปที่ 3.4 แสดงรายละเอียดของไอซี DS 75176B

2. IC 74LS244 ทำหน้าที่ ช่วยเพิ่มกระแสของพอร์ท 89C51 ให้ LED



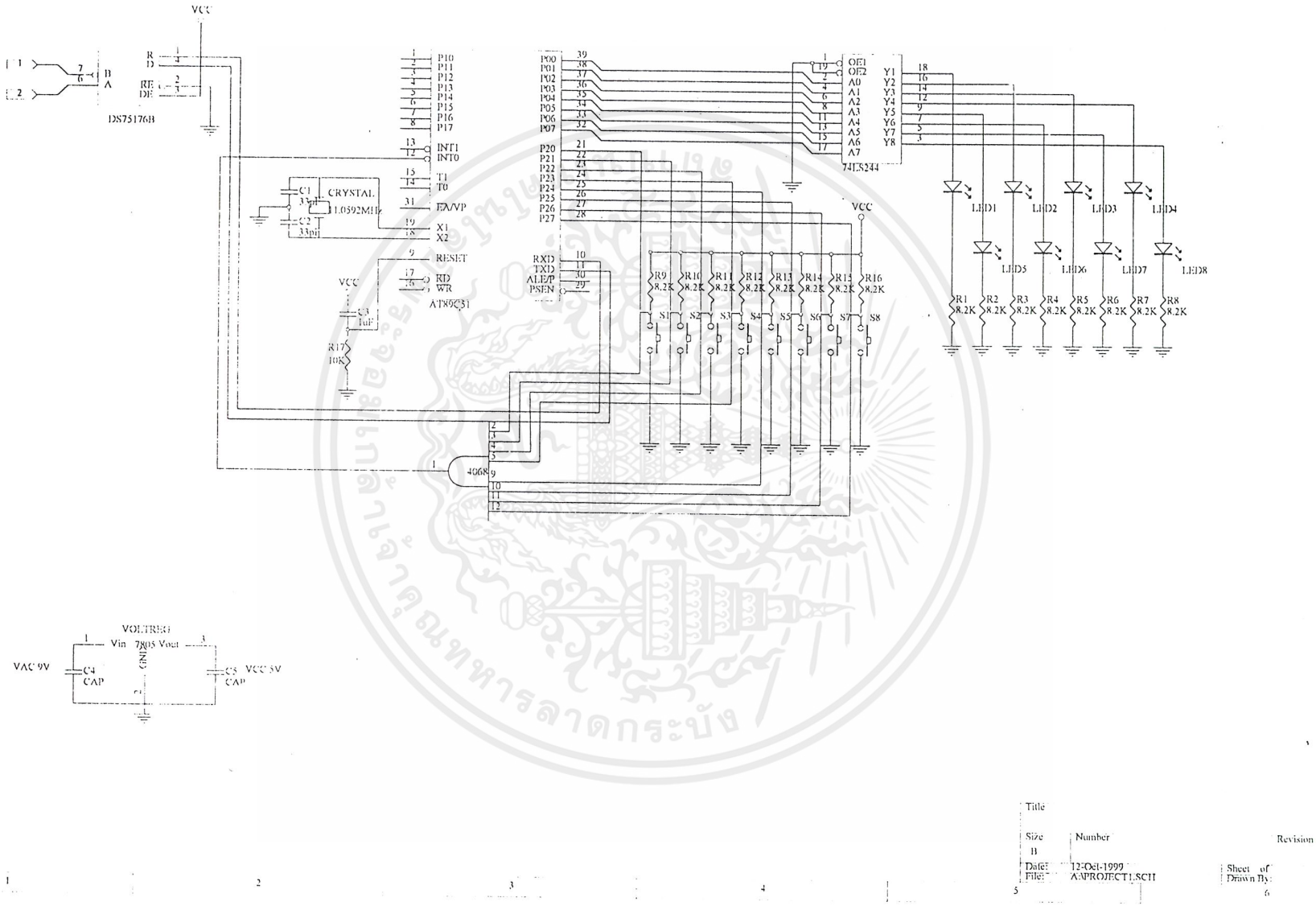
รูปที่ 3.5 แสดงรายละเอียดของไอซี 74LS244

3. IC 4068 ทำหน้าที่ AND สัญญาณ INPUT เพื่อเป็นสัญญาณ INTERRUPT ให้ 89C51

4. LED และ SWITCH เป็น INPUT และ OUTPUT

5. IC 89C51 ทำหน้าที่ประมวลผลและติดต่อข้อมูล

แสดงวงจรของการแปลงสัญญาณ



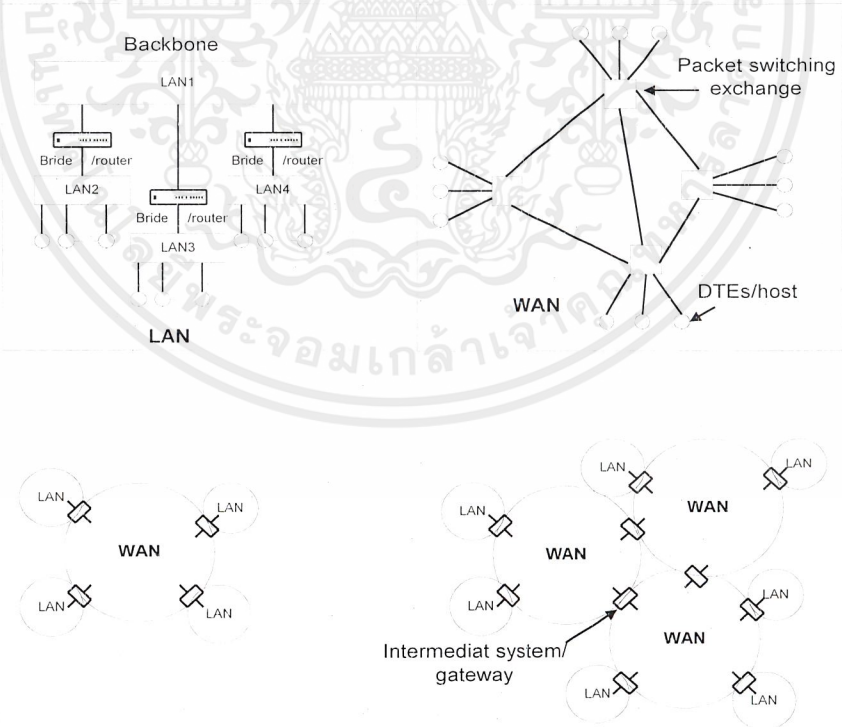
แสดงวงจรของตัวทำงานหลัก

บทที่ 4
เครือข่ายอินเตอร์เนต

Internet คือ การที่เครือข่าย 2 หรือมากกว่า เชื่อมต่อเข้าด้วยกัน และทำงานเสมือนเป็นเครือข่ายเดียวกัน โดย network ที่เป็นส่วนประกอบของ internet คือ Subnetwork (Subnet) ซึ่งอาจจะเป็นเครือข่าย Local area network (LAN) หรือ Wide area network (WAN) อุปกรณ์ที่ใช้ในการเชื่อมต่อ 2 เครือข่าย เข้าด้วยกันก็คือ intermediate system (IS) หรือ internetworking unit (IWU) การเชื่อมโยงระหว่างระบบที่แตกต่างกัน จำเป็นต้องมีมาตรฐานในการติดต่อกัน ซึ่งเรียกเป็นศัพท์เฉพาะว่าโพรโตคอล (Protocol)

4.1 สถาปัตยกรรม อินเตอร์เน็ต (Internet Architectures)

สถาปัตยกรรมพื้นฐานของ Internet แสดงดังรูปที่ 4.1

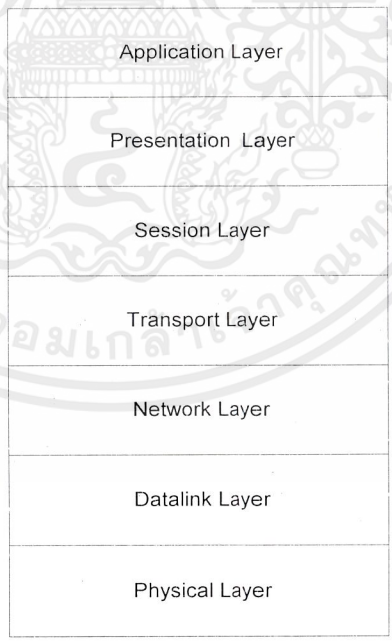


รูปที่ 4.1 แสดงสถาปัตยกรรมของ Internet (a) Single LAN and WAN (b) Interconnected LAN/WAN

ในรูป (a) แสดงตัวอย่าง 2 ตัวอย่างของเครือข่ายเดี่ยว (Single network) ซึ่งอย่างแรกเป็น site-wide LAN ซึ่งประกอบขึ้นมาจากชุดของ LANs ซึ่งถูกต่อเข้ากับเครือข่ายหลัก (backbone) ซึ่งอุปกรณ์ที่ใช้ต่อ LAN เข้ากับเครือข่ายหลัก ถ้า LAN ทุกเครือข่ายมีระบบเดียวกันก็จะใช้ bridge ถ้าเป็น LAN ที่แตกต่างกันก็จะใช้ router ตัวอย่างที่ 2 เป็นตัวอย่างของ WAN เดี่ยว ๆ ในรูป (b) แสดงถึงเครือข่ายอินเตอร์เน็ต ซึ่งประกอบด้วย network ทั้ง 2 ชนิด ข้างต้น

4.2 OSI โมเดล

องค์กรมาตรฐานสากล ISO (International Organization for Standardization) ได้กำหนดมาตรฐานของเครือข่าย โดยจัดแบ่งกิจกรรมของเครือข่าย ออกเป็นงานย่อย ๆ และกำหนดโมเดลแบ่งเป็นชั้น ๆ ตามลำดับเรียกว่ามาตรฐาน OSI (Open System Interconnection) โดยที่จะแบ่งกิจกรรมที่ซับซ้อนในเครือข่าย ออกเป็นงานย่อย ๆ ก็จะช่วยในการออกแบบ และการใช้งานเครือข่าย รวมถึงการเชื่อมโยงกัน เป็นไปได้ด้วยความสะดวก และมีวิธีการทำงานอยู่ในกรอบเดียวกัน ดังในรูปที่ 4.2



รูปที่ 4.2 แสดงการแบ่งการทำงานของเครือข่ายออกเป็น OSI model

ในแต่ละชั้นของ OSI model จะมีการติดต่อสื่อสารกันเป็นชั้น ๆ ตามลำดับลงมาเช่น Application Layer ก็จะติดต่อสื่อสารกับ Presentation Layer ตามลำดับไปจนถึงชั้นแรกสุดคือ Physical Layer

Application Layer เป็นชั้นบนสุดของ โมเดลเป็นส่วนที่จะทำให้การติดต่อระหว่างเครือข่ายกับผู้ใช้ เป็นไปได้ตามต้องการ ตัวอย่างแอปพลิเคชั่นของเครือข่าย เช่น ระบบ e-mail, การโอนถ่ายข้อมูล (File Transfer), การขอเข้าใช้ระบบคอมพิวเตอร์ในเครือข่ายเป็นต้น

Presentation Layer มีการกำหนดหน้าที่ไม่ชัดเจนนักและมีการนำไปใช้ไม่มาก ซึ่งหน้าที่หลักก็คือ เป็นส่วนที่จัดรูปแบบและนำเสนอข้อมูล ให้เป็นไปตามต้องการ รวมไปถึงการจัดแปลงข้อมูล ในรูปมาตรฐาน ASCII หรือ EBCDIC, การลดขนาดข้อมูล (data compression) การเข้ารหัสหรือถอดรหัสของข้อมูล แต่ส่วนใหญ่แล้ว แอปพลิเคชั่นจะจัดการแทนได้

Session Layer เป็นชั้นที่จัดการในเรื่อง "การติดต่อแต่ละครั้ง" หรือ session ให้ระบบคอมพิวเตอร์ทั้งสองฝั่ง โดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลในการส่งผ่านข้อมูล ในการติดต่อครั้งนั้น ๆ เป็นไปได้โดยไม่มีปัญหา จนถึงเลิกการติดต่อเมื่อเสร็จงาน

Transport Layer ทำหน้าที่ควบคุมปริมาณ และรายละเอียดวิธีการรับส่งข้อมูล ให้เป็นไปตามกำหนดที่ตั้งไว้ และจัดการให้การเชื่อมโยงเครือข่ายเป็นไปด้วยความราบรื่น Transport Layer จะเป็นชั้นสุดท้าย ที่จัดการเรื่องเส้นทางในการส่งข้อมูล และจัดการตรวจสอบความผิดพลาดของข้อมูล ซึ่งส่วนของ TCP (Transmission Control Protocol) ในโพรโตคอล TCP/IP ทำงานที่ระดับนี้

Network Layer ทำหน้าที่ควบคุมวิธีการส่งผ่านข้อมูลระหว่างเครือข่ายให้ถูกต้อง และเป็นไปตามเส้นทางที่กำหนด โดยจะจัดการส่งผ่าน packet ข้อมูล ผ่านอุปกรณ์ต่าง ๆ ไปยังเครือข่ายย่อยได้อย่างถูกต้องตามที่ต้องการ นอกจากนี้ยังจัดการดูแลเส้นทางในการส่งข้อมูล (Routing table) และกั้นหรือกรอง packet ข้อมูลที่ส่งไปยังเครือข่ายเดียวกันไม่ให้ข้ามไปยังเครือข่ายอื่น ซึ่งจะช่วยลดปริมาณข้อมูลที่จะวิ่งบนเครือข่ายได้ส่วนหนึ่ง โพรโตคอล IP, TCP/IP และ IPx เป็นโพรโตคอลที่ทำงานอยู่ใน layer นี้

Data link Layer ทำหน้าที่เรียกใช้หรือกำหนดช่องทาง ในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet, Tokenring หรือ FDDI เป็นต้น รวมถึงการลำดับและอัตราการรับส่งข้อมูลหรือ flow control และสถานที่ ที่จะส่งข้อมูลไป (address) ทั้งนี้ Data link layer จะเป็นชั้นแรกที่จัดการแปลงข้อมูลจาก bit ให้เป็น packet โดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบความถูกต้อง ในกรณีที่ส่งข้อมูลออกไป หรือในกรณีที่อ่านข้อมูลที่เข้ามา ก็จะตรวจสอบผ่าน checksum เพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วน และถ้าได้รับ packet ข้อมูลที่ไม่ถูกต้องก็จะไม่เอาข้อมูลนั้นไปใช้งาน และจะบอกให้ต้นทางส่งข้อมูลเดิมมาใหม่

Physical Layer รับผิดชอบดูแลในรายละเอียดในการส่งข้อมูลในด้าน hardware เช่น การควบคุม Network Interface Card การส่งสัญญาณผ่านสายสัญญาณแบบต่าง ๆ การเชื่อมต่อเข้ากับเครือข่ายแบบต่าง ๆ โดยใช้ Physical Layer จะจัดสร้างสัญญาณทางไฟฟ้า, สัญญาณเสียง หรือ

สัญญาณที่จำเป็นในการสื่อสารโดยตรง เนื่องจาก Network Layer เป็นชั้นที่โพรโตคอล IP, TCP/IP ทำงานอยู่จะกล่าวโดยละเอียดต่อไป

4.3 โครงสร้างชั้น เน็ทเวิร์ค(Network layer structure)

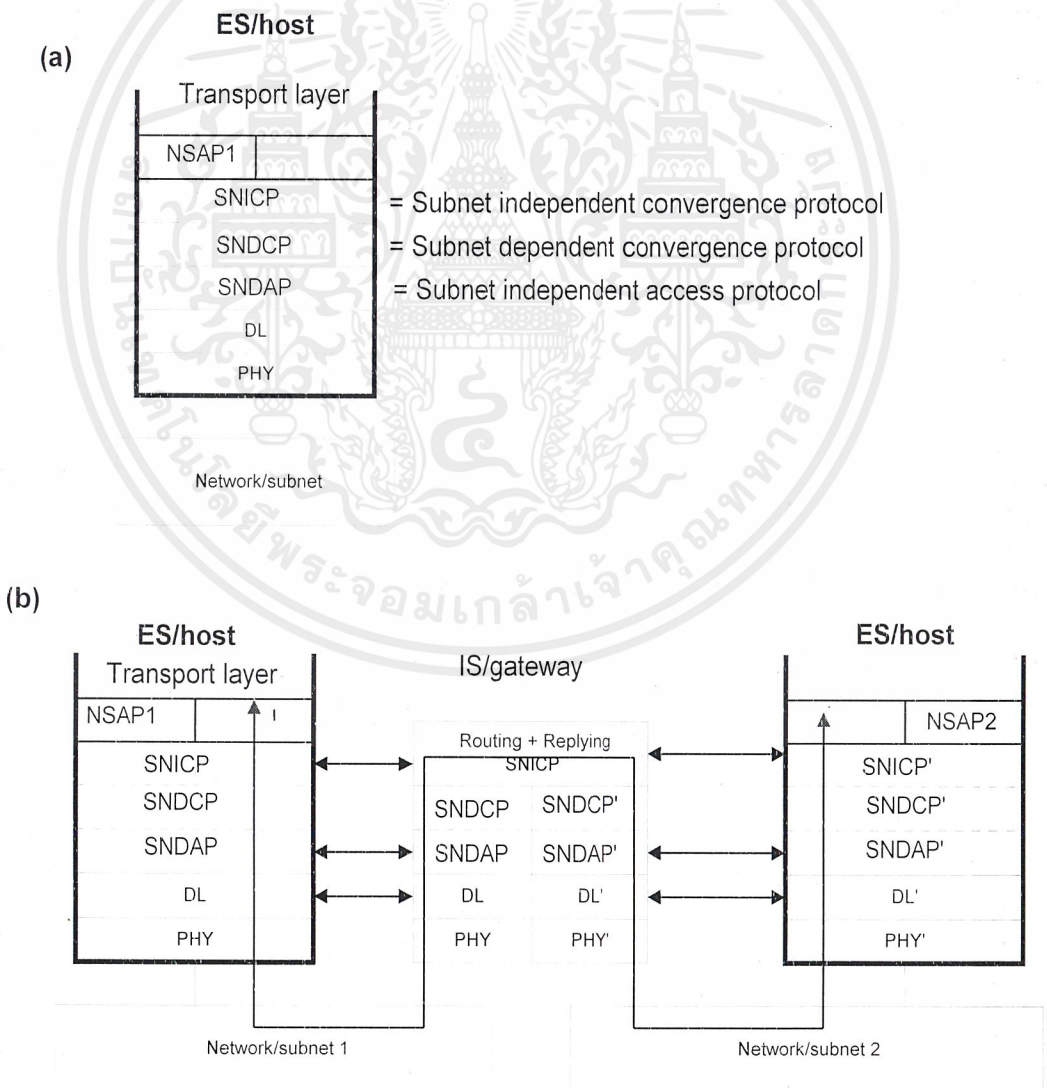
หน้าที่ของ Network Layer ในแต่ละ End System (ES) จะเป็นตัวจัดการการติดต่อแบบ end-to-end ของการบริการ internetwide ไปยังผู้ใช้บริการ (NS-User)

โดย ISO ได้จัด network layer เป็น 3 (sublayer) protocol ซึ่งจะทำงานร่วมกัน ได้แก่

Subnetwork independent convergence protocol (SNICP)

Subnetwork dependent convergence protocol (SNDCP)

Subnetwork dependent access protocol (SNDAP) โดยที่ โครงสร้างแสดงดังรูปที่ 4.3



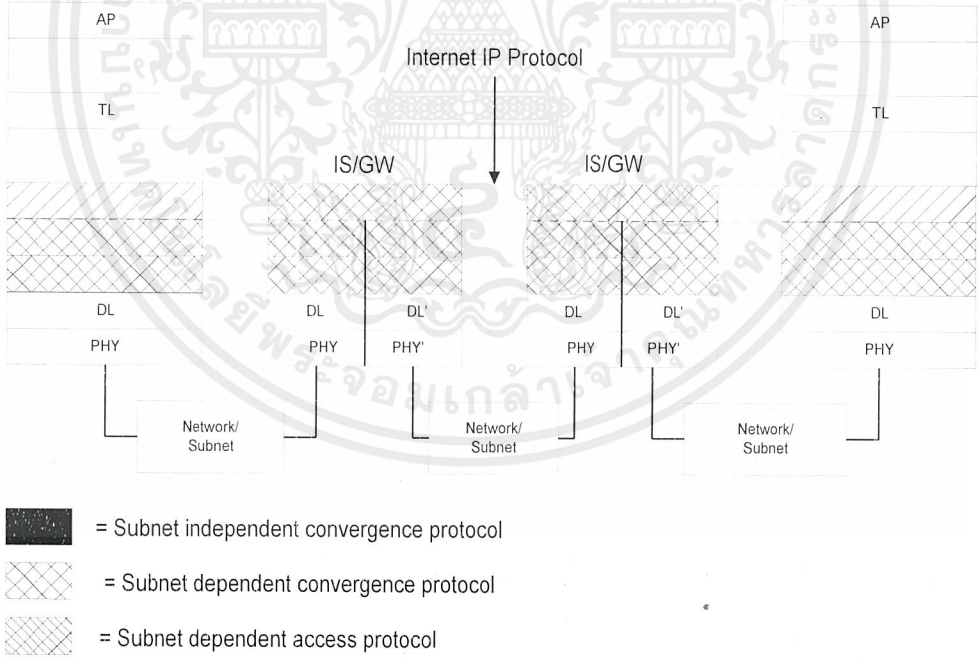
รูปที่ 4.3 Network layer structure (a) Sublayer Protocol ; (b) IS structure

โดยที่ SNICP จะเป็นตัวสนับสนุนจัดการให้ผู้ใช้บริการ (NS-user) สามารถ interface กับ Internet ซึ่งมันจะมีหน้าที่ เป็นตัวประสานฟังก์ชันต่าง ๆ ที่จำเป็นในการเลือกเส้นทางและถ่ายทอดข้อมูลของผู้ใช้ข้าม Internet ซึ่งการทำงานของมันไม่ขึ้นอยู่กับคุณสมบัติเฉพาะของ เครือข่ายย่อย (subnet)

SNDAP จะเป็นตัวโพรโตคอลที่ติดต่อกับเครือข่ายย่อย (subnet) ที่มีลักษณะเฉพาะใน Internet เช่น X.25 packet layer protocol สำหรับเครือข่าย X.25 ซึ่งใช้บ่อยใน LAN เพราะว่าการบริการและการทำงานของ SNDAP แตกต่างจาก network แบบอื่น ๆ sublayer ที่อยู่ตรงกลางคือ SNDCP จะเป็นตัวจัดการระหว่าง SNICP และ SNDAP

4.4 Internet protocol standards

อินเตอร์เน็ตโพรโตคอลซึ่งถูกใช้ในอินเตอร์เน็ตมี โพรโตคอลก็คือ TCP/IP (Transfer Control Protocol / Internet Protocol) ซึ่งรวมถึง transport และ application โพรโตคอล ซึ่งทั้งหมดของ TCP/IP จะกำหนด ให้เหมาะกับการใช้ในเชิงสาธารณะ ซึ่งรูปแบบโดยทั่วไปแสดงดังรูปที่ 4.4



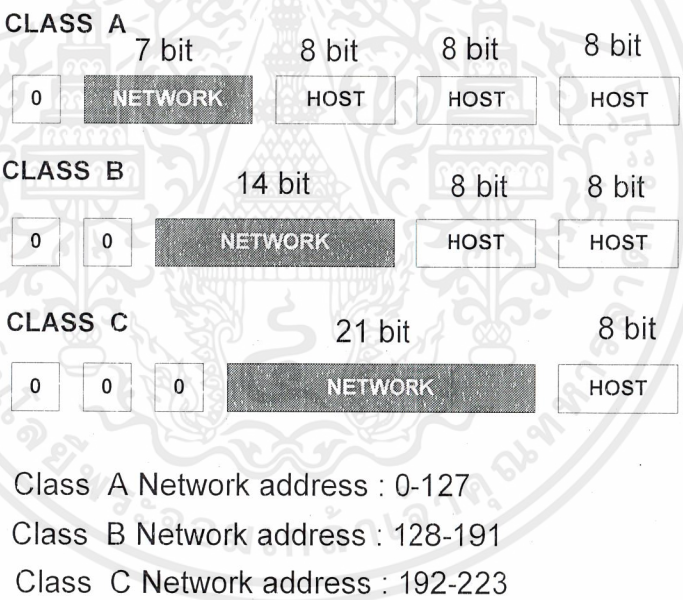
รูปที่ 4.4 Internetwide IP schematic

IP เป็น internetwide protocol ซึ่งทำให้สอง transport protocol ที่ต่างสถานที่และต่าง ESs / hosts กันสามารถแลกเปลี่ยนหน่วยข้อมูล (NSDUS) กันได้ ซึ่งหมายถึงว่า หลาย ๆ network / subnet และ ISs / gateways ที่แตกต่างกันสามารถ ติดต่อสื่อสารกันได้อย่างสมบูรณ์

4.5 Internet IP

4.5.1 โครงสร้างแอดเดรส(Address structure)

ในศัพท์ของ ISO เมื่อ 2 network ติดต่อกันด้วย host/ES ที่ต่อกับอินเตอร์เน็ต network เหล่านี้ ติดต่อกันได้โดยใช้ network service access point (NSAP) address และ subnet point of attachment (SNPA) สำหรับใน TCP/IP ก็จะมี IP address และ NPA address ตามลำดับ โดย NPA address จะแตกต่างกันในแต่ละชนิดของ network/subnet ขณะที่ IP address



รูปที่ 4.5 โครงสร้างของ Address ที่ใช้ใน class ต่าง ๆ ของเครือข่ายทั้งหมดความยาว 32 bit

IP address นี้มีการจัดแบ่งออกเป็นทั้งหมด 5 ระดับ (class) แต่ที่ใช้งานทั่วไปจะมีเพียง 3 ระดับคือ Class A, Class B และ Class C ซึ่งจะแบ่งตามขนาดความใหญ่ของเครือข่าย ถ้าเครือข่ายใดมีจำนวนเครื่องคอมพิวเตอร์ที่เชื่อมต่ออยู่มาก จะมีหมายเลขอยู่ใน Class A และลดหลั่นกันมาใน Class B และ Class C ตามลำดับ

จากรูปจะเห็นว่าหมายเลข IP ของ Class A มีตัวแรกเป็น 0 และหมายเลขของเครือข่าย (network number) ขนาด 7 bit และมีหมายเลขเครื่องคอมพิวเตอร์ (Host number) ขนาด 24 bit ทำให้ในหนึ่งเครือข่ายของ Class A สามารถมีคอมพิวเตอร์เชื่อมต่ออยู่ในเครือข่ายได้ถึง 2^{24} =16 ล้าน

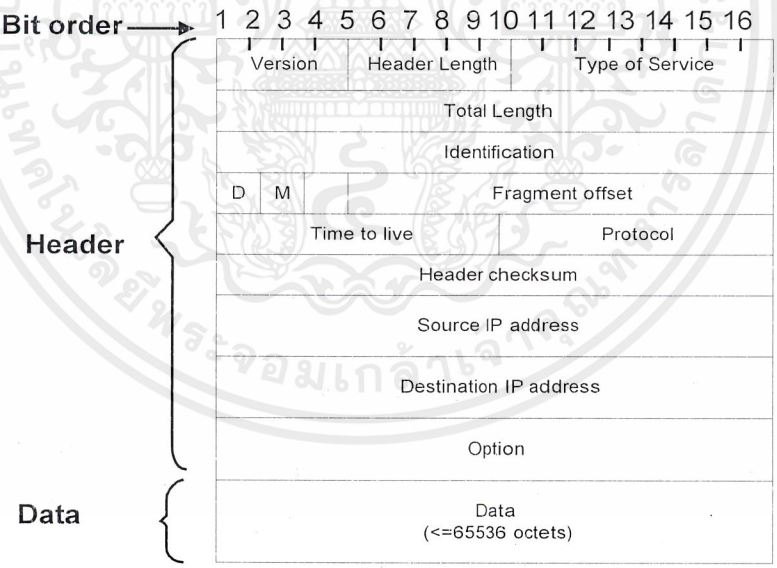
เครื่อง แต่ใน Class A นี้ จะมีหมายเลขเครือข่ายได้ 128 ตัวเท่านั้นทั่วโลก ซึ่งก็คือจะมีเครือข่ายใหญ่แบบนี้เพียง 128 เครือข่ายเท่านั้น

สำหรับ Class B จะมีหมายเลขเครือข่ายแบบ 14 bit และหมายเลขเครื่องคอมพิวเตอร์แบบ 16 bit (ส่วนอีก 2 bit ที่เหลือบังคับว่าต้องขึ้นต้นด้วย 10_2) ดังนั้นจึงสามารถมีคอมพิวเตอร์เชื่อมต่อในเครือข่าย Class B แต่ละเครือข่ายได้ถึง 2^{16} = กว่า 65,000 เครื่อง และสุดท้ายคือ Class C ซึ่งมีหมายเลขคอมพิวเตอร์แบบ 8 bit และมีหมายเลขเครือข่ายแบบ 21 bit ส่วน 3 bit แรกบังคับว่าต้องเป็น 110_2 ดังนั้นในแต่ละเครือข่าย Class C จะมีจำนวนเครื่องต่อเชื่อมได้เพียงไม่เกิน 254 เครื่องในแต่ละเครือข่าย (2^8=256 แต่หมายเลข 0 และ 255 จะไม่ถูกใช้งาน จึงเหลือเพียง 254)

จะเห็นได้ว่าเมื่อเครือข่ายและเครื่องคอมพิวเตอร์ ที่ต่ออยู่ในอินเตอร์เน็ตมีหมายเลข IP address ให้ใช้อ้างอิงได้ไม่ซ้ำกัน และมีความหมายให้ทราบถึงขนาดเครือข่ายแล้ว การติดต่อส่งผ่านข้อมูล จึงกระทำได้ไม่สับสน

4.5.2 รูปแบบของ ข้อมูล(Datagrams)

รูปแบบของ IP data unit ก็คือ datagrams ซึ่งโครงสร้างของ datagrams เป็นดังรูปที่ 4.6



รูปที่ 4.6 Internet datagrams format and contents

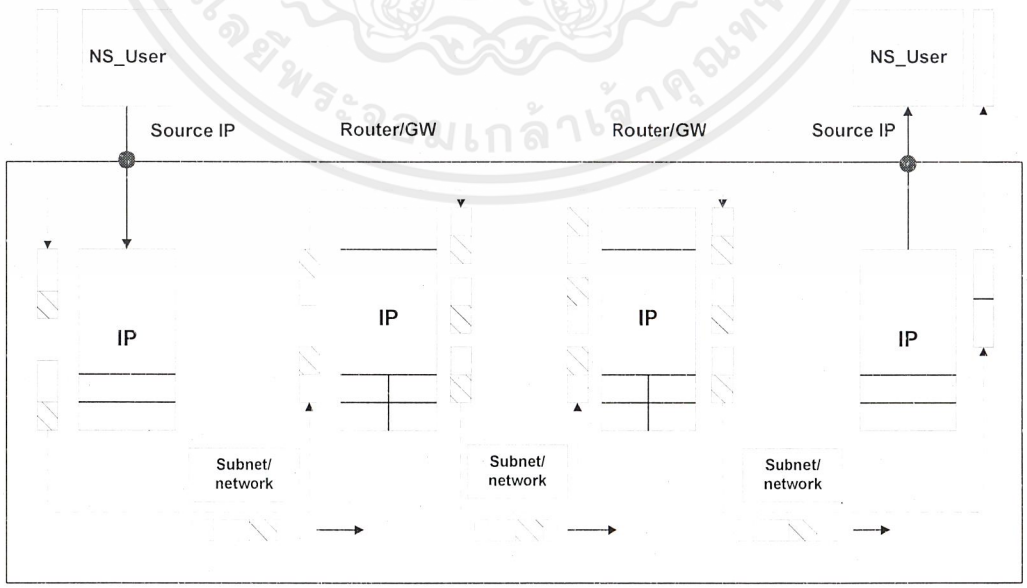
หน่วยข้อมูล IP (IP datagrams) แต่ละหน่วยจะประกอบด้วย ส่วนของข้อมูลที่รับมาจากส่วนของงาน TCP หรือ UDP และส่วนของข้อมูลนำทาง (Header) ซึ่งมีรายละเอียดดังนี้

- Version หมายเลขรุ่นของข้อกำหนด IP
- Header Length ความยาวของข้อมูลนำทาง

- Type of service วิธีการจัดการกับข้อมูล
- Total Length ความยาวของหน่วยข้อมูล
- Identification, Flags และ Fragment offset รายละเอียดที่เกี่ยวกับการแบ่งย่อยข้อมูล ซึ่งจะถูกนำมาใช้ในการรวบรวมข้อมูล
- Time to live เวลาสูงสุดที่ใช้ในการเดินทาง ซึ่งกำหนดมาจากต้นทาง เวลานี้จะลดลงเรื่อย ๆ ในระหว่างทาง ถ้าลดลงไปถึงศูนย์ หน่วยข้อมูลนั้นจะถูกกำจัดไป
- Protocol ชนิดของข้อมูลเป็น UDP หรือ TCP
- IP address หมายเลข internetwide IP (NSAP) ของเครื่องต้นทางและปลายทาง
- Option ข้อมูลอื่น ๆ เช่น ข้อมูลเกี่ยวกับการรักษาความปลอดภัย บันทึกเส้นทางเดินของข้อมูล และเวลาที่ข้อมูลเดินทางมาถึง เป็นต้น

4.5.3 การแบ่งส่วนย่อยของข้อมูล และการประกอบขึ้นใหม่ (Fragmentation and Reassembly)

ขนาดข้อมูลของผู้ใช้ซึ่งอ้างอิงกับ NSDU มีความจุได้ถึง 64k หรือ 65,536 bytes แต่ขนาดของหน่วยข้อมูล (packet size) ที่สามารถติดต่อกันในระบบที่ต่างกัน สามารถมีได้ตั้งแต่ 128 byte สำหรับระบบ X.25 packet switching จนถึง 8000 byte สำหรับบาง LAN ดังนั้นกระบวนการ Fragmentation และ Reassembly จึงถูกนำมาใช้เพื่อ ทำให้ขนาดของข้อมูลเล็กลง และสามารถส่งไปในระบบได้ และเมื่อถึงปลายทาง IP ก็จะทำการประกอบข้อมูล (reassembly) ขึ้นมาใหม่ก่อนที่จะส่งผ่านไปยังผู้ใช้ปลายทาง ดังรูปที่ 4.7



รูปที่ 4.7 internet fragmentation and reassembly

อันดับแรก IP ใน Host ต้นทางจะแยกข้อมูลของผู้ใช้ (NS -User), NSDU เป็น Datagram ซึ่งมี Address กำกับเป็นเฉพาะส่วน ๆ ไปซึ่งจะถูกออกคำสั่งโดย Network ที่มันติดต่ออยู่ด้วยและส่ง Datagram ไปยัง IP ใน Gateway ตัวแรก โดยที่ IP ใน Gateway จะไม่ Reassemble NSDU แต่จะปรับปรุงในขอบเขตที่เหมาะสม และส่ง Datagram ที่ได้รับตรงไปยัง Network ที่สอง (ถ้า Network ที่สองสามารถรองรับขนาด Datagram นี้) หรือทำการ Fragment datagram ให้มีขนาดเล็กลง ซึ่งขั้นตอนนี้จะถูกทำซ้ำที่ Gateway ตัวต่อไป โดยในรูปที่ 4.7 Network หลังสุดสามารถรองรับขนาดของ Packet ได้มากกว่า Packet ที่มันได้รับข้อมูลจึงถูกส่งได้โดยตรง โดยที่จะมีการปรับปรุงในบางส่วน Header ของ Datagram เท่านั้น จากนั้น IP ใน Host ปลายทางจะทำการประกอบข้อมูล (Reassemble) ที่มันได้รับขึ้นมาใหม่ และส่งผลที่ได้รับก็คือ NSDU ไปยังผู้ใช้ (NS-user)

ในการคิดค่าเวลาสูงสุดที่ Host ต้นทางกำหนดให้ Gateway รอ Datagram (NSDU) ระหว่างแต่ละการ Assembly ซึ่งก็คือ time-to-live ซึ่งจะถูกนำติดไปที่ส่วน Header ของ Datagram ซึ่งจะถูกตั้งค่าโดย IP ใน Host ต้นทาง ซึ่งจะมีค่าลดลงเรื่อยๆในแต่ละขั้นตอนของการ Process datagram ถ้า Datagram ถูก Fragment ค่าปัจจุบันจะถูกนำไปใส่ในส่วน Header ของ Datagram ตัวใหม่ ถ้ามันถึงค่า 0 ที่จุดใด ๆ ระหว่างการ Process ใน Gateway (หรือ Host) การ Reassembly ก็จะล้มเหลวและทุกๆการ Fragment ที่เกี่ยวกับ NSDU ก็จะถูกตัดทิ้ง

ค่า Time-to-live ในแต่ละ Datagram จะเป็นจำนวนเท่าของ 1 วินาที โดยที่จำนวนของมันจะถูกลดลงโดยแต่ละ IP ซึ่งจะเปลี่ยนแปลงไปตามค่าเวลาจริงในการส่งถ่ายข้อมูลของ Network ที่ติดต่อด้วย

4.5.4 Routing

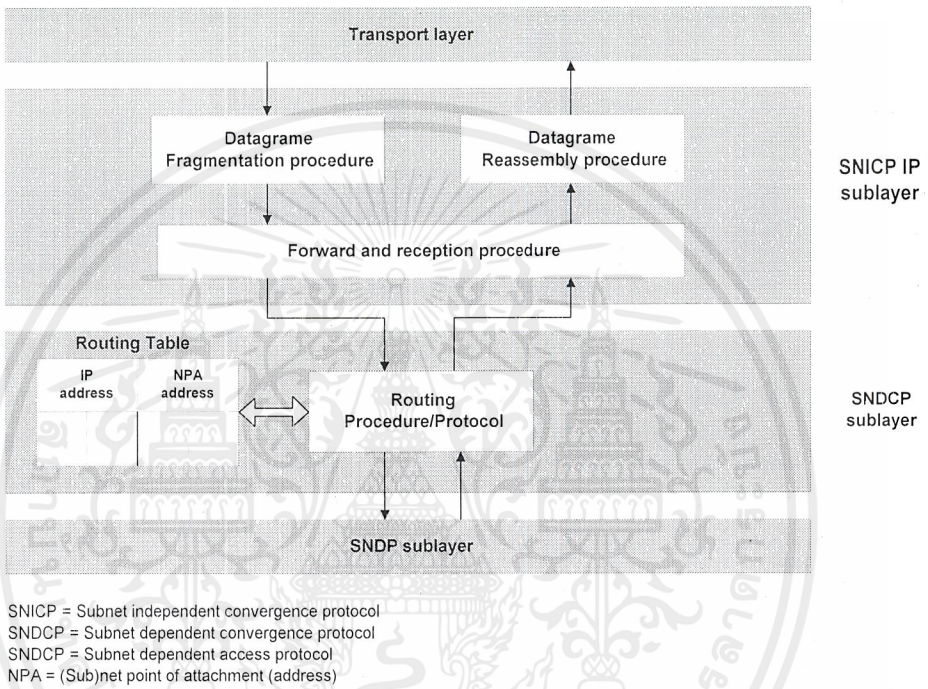
ในแต่ละ Network (หรือ subnet) ใน Internet จะมีชนิดของ PA address ที่แตกต่างกัน ซึ่งระบบ (system)-host หรือ gateway ที่ถูกต่อเข้ากับ network จะสามารถส่ง datagram ไปยังระบบอื่นได้โดยตรงเฉพาะ network ที่เหมือนกันเท่านั้น ในการเลือกเส้นทาง (routing) ให้ datagram ข้ามไปยังหลาย ๆ network IP ในแต่ละ internetwork gateway ต้องรู้ PA address ของ host ปลายทาง

ซึ่งมี 2 วิธีการพื้นฐานที่ถูกใช้ในการหาเส้นทางภายใน Internet คือ centralized และ distributed ด้วยวิธีการ centralized routing ข้อมูลเกี่ยวกับการเลือกเส้นทาง ที่เกี่ยวข้องกับแต่ละ gateway จะถูก download จาก site ส่วนกลางโดยใช้ข้อมูล network และ special network management โดย network management จะพยายามตรวจสอบ network และ host ที่ถูกเพิ่มเข้าและถอดออก และข้อบกพร่องที่จะถูกวินิจฉัยและตรวจซ่อม

ด้วยวิธีการ Distributed routing ทุก ๆ host และ gateway จะร่วมกันในการแบ่งปัน วิธีการในการรับประกันว่า ข้อมูลเกี่ยวกับการเลือกเส้นทางในแต่ละ system, host และ gateway จะถูกทำ

ให้ทันสมัย และสอดคล้องกัน ข้อมูลเกี่ยวกับการเลือกเส้นทางจะถูกจดจำไว้โดยแต่ละระบบ ในรูปของ routing table ซึ่งจะมี NPA address ไว้ใช้ในการส่งแต่ละ datagram ซึ่ง Internet จะใช้วิธีการแบบนี้

ขั้นตอนการ Routing ที่เกี่ยวกับ IP ขั้นตอนแรกจะอ่าน IP address (NSAP) ปลายทางจากภายใน datagram และใช้มันในการหาการตอบสนอง PA address ของ host หรือ gateway จาก



routing table ในส่วนที่เพิ่มเติมชุดของ routing protocol จะถูกใช้เพิ่มและรักษาส่วนที่อยู่ในแต่ละ routing table ในแบบของ distributed ซึ่งรูปแบบทั่วไปที่ถูกใช้ภายใน host IP แสดงดังรูปที่ 4.8

รูปที่ 4.8 รูปแบบทั่วไปของการเลือกเส้นทางภายใน Host

บทที่ 5
การออกแบบโปรแกรม

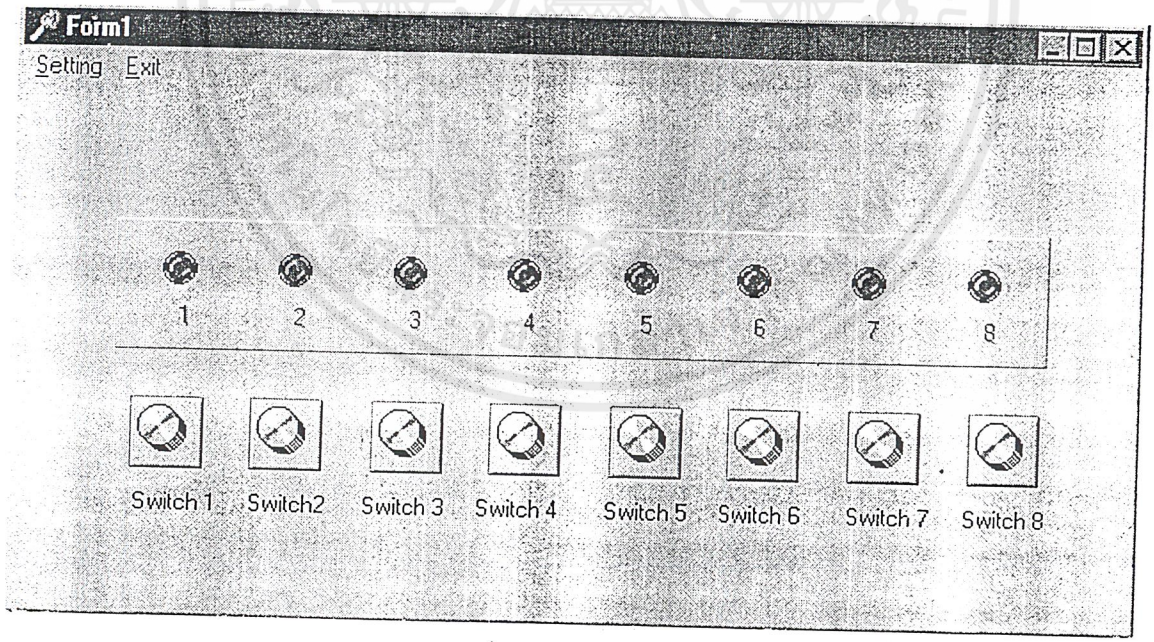
ระบบควบคุมที่มีประสิทธิภาพนั้นต้องประกอบด้วยสองส่วนใหญ่ ๆ คือ ส่วนของฮาร์ดแวร์ และซอฟต์แวร์ เพื่อให้ทั้งระบบนั้นสามารถทำงานประสานกันได้อย่างมีประสิทธิภาพอย่างเป็นระบบ โดยโปรแกรมที่ควบคุมระบบนี้ แบ่งออกเป็น 2 ส่วน คือ ส่วนโปรแกรมบนไมโครคอมพิวเตอร์ และส่วนของโปรแกรมบนไมโครคอนโทรลเลอร์

5.1 โปรแกรมบนไมโครคอมพิวเตอร์

เนื่องจากได้มีการเขียนโปรแกรมให้ใช้การคำนวณแบบเหมือนจริง (Visual Control) ซึ่งทำให้ผู้ใช้งานสามารถใช้งานได้ง่าย ประกอบด้วย

5.1.1 โปรแกรมหลัก

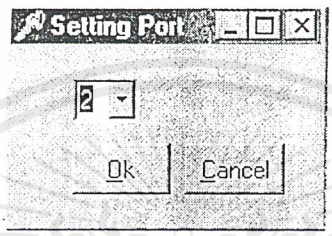
เป็นหน้าจอหลักเมื่อได้มีการเปิดโปรแกรมขึ้นมา



รูปที่ 5.1.1 แสดงหน้าจอหลักของโปรแกรม

5.1.2 ส่วนตั้งค่าของพอร์ทอนุกรม

ในการส่งข้อมูลทางพอร์ทอนุกรมนั้น ตัวส่งและตัวรับต้องมีการตั้งค่าในการสื่อสารให้สัมพันธ์กัน



รูปที่ 5.1.2 แสดงหน้าจอตั้งค่าพอร์ท

5.2 ส่วนโปรแกรมบนไมโครคอนโทรเลอร์

ใช้ไมโครคอนโทรเลอร์ 89C51 เป็นตัวประมวลผลประกอบด้วย

5.2.1 ส่วนติดต่อสื่อสารแบบอนุกรม

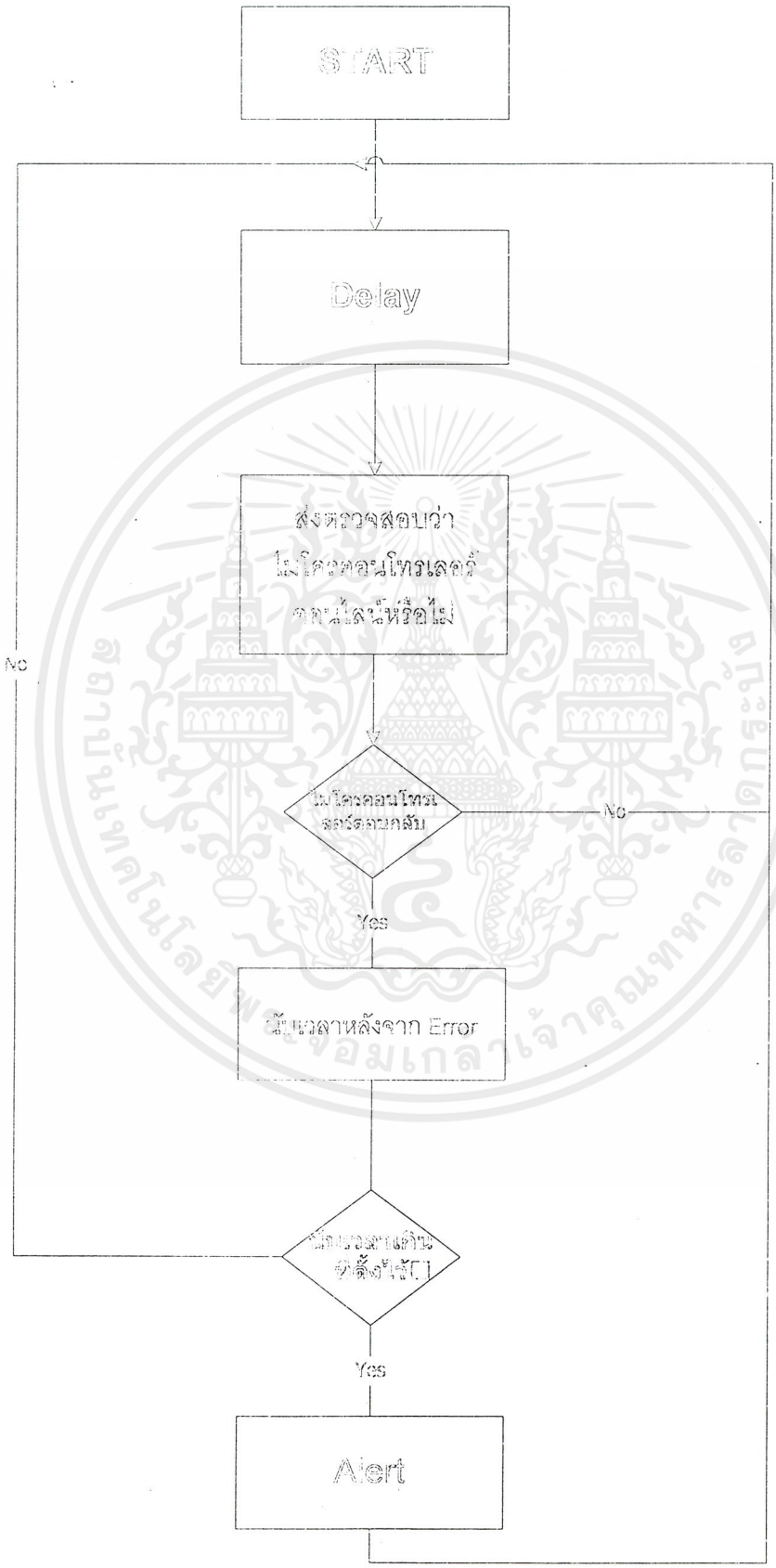
เป็นส่วนติดต่อสื่อสารอนุกรมตามมาตรฐาน RS-232C ที่ความเร็ว 9600 B/S ไม่มี PARITY BIT จะมีการส่งข้อมูลเมื่อมีการเปลี่ยนแปลงที่อุปกรณ์แสดงผลเนื่องจากผลของอุปกรณ์รับข้อมูลเฉพาะ และจะมีการวนรอบตรวจสอบว่ามีการส่งข้อมูลมาจากไมโครคอมพิวเตอร์ ซึ่งข้อมูลที่รับมาจะส่งไปยังส่วนประมวลผลและนำไปแสดงต่อไป

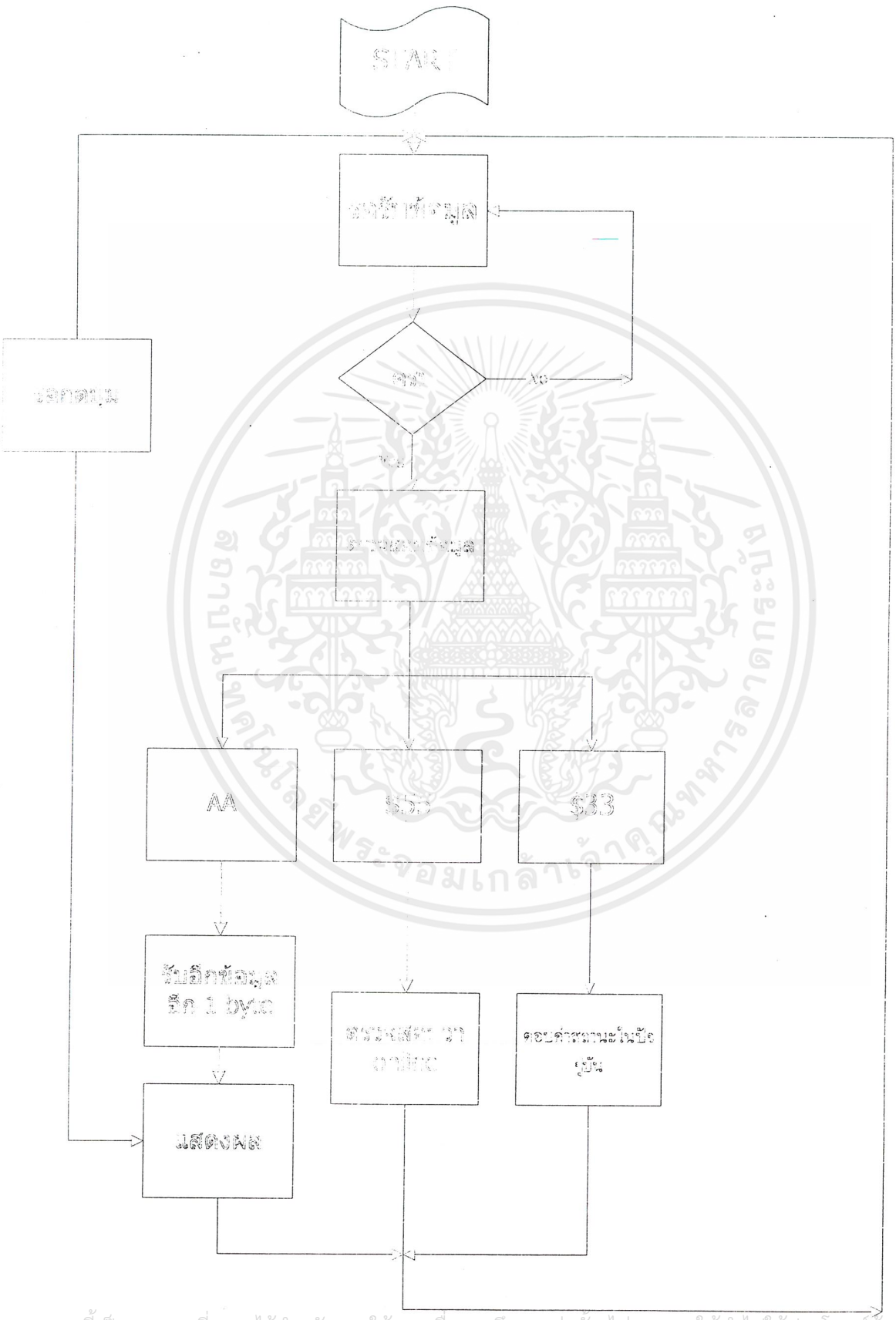
5.2.2 ส่วนรับค่าจากอุปกรณ์เฉพาะ

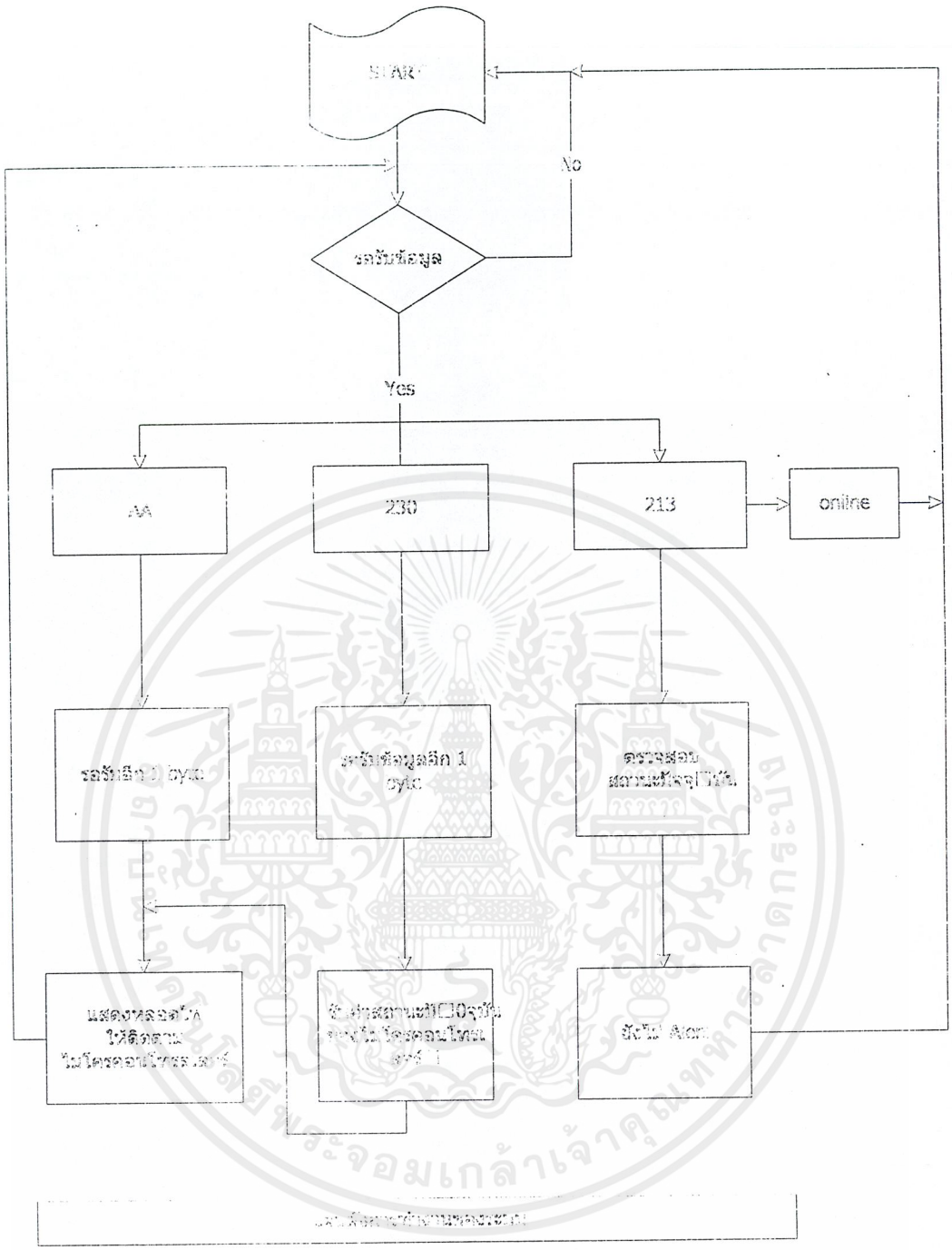
เป็นส่วนรับข้อมูลจากพอร์ทรับข้อมูลไปพักที่ REGISTER R0 เพื่อรอการประมวลผลต่อไป โดยจะมีการทำงานเมื่อมีสัญญาณ INTERRUPT

5.2.3 ส่วนประมวลผลและแสดงผล

เป็นส่วนที่ทำงานเมื่อมีการเรียกใช้จากโปรแกรมรับข้อมูล จากทั้งไมโครคอมพิวเตอร์และอุปกรณ์รับข้อมูลเฉพาะ มาวิเคราะห์และนำไปแสดงผลต่อไป







บทที่ 6
ผลการทดลอง

6.1 การทดลองในการส่งข้อมูล

เป็นการทดลองโดยส่งข้อมูลจากคอมพิวเตอร์ส่วนบุคคลผ่านทางอแดปเตอร์ส่งไปยังตัวควบคุมหลัก จากนั้นตัวควบคุมหลักจะทำการสลับไบท์สูงกับไบท์ต่ำ แล้วส่งค่ากลับมาที่คอมพิวเตอร์ส่วนบุคคล แล้วทำการเปรียบเทียบค่าที่ส่งกับค่าที่รับว่าเหมือนกันหรือไม่เพื่อทดสอบว่าเกิดความผิดพลาดในการส่งได้อย่างไร จะมีวิธีแก้ไขอย่างไร ได้ทำการทดสอบที่

- ใช้สายสัญญาณเป็นสายเกลียวคู่ความยาว 50 เมตร
- อัตราเร็วในการส่ง (Baud rate) 56.7 kbps

ได้ทำการทดสอบเพื่อหาเปอร์เซ็นต์ของความผิดพลาดในการส่ง โดยทำการส่งข้อมูลและทำการตรวจสอบการตอบกลับที่ผิดพลาด ในช่วงเวลาประมาณ 10 นาทีได้ผลออกมาเป็นดังนี้

เวลาที่ใช้ในการส่ง	จำนวนไบท์ที่ส่งไปทั้งหมด	Rate(Byte/s)	Err byte
604.78	1337678	2211.84	94(0.007%)
555.13	1231866	2219.06	79(0.0064%)
676.74	1439521	2127.14	0

ในการทดสอบถึงสายสัญญาณออกจากตัวควบคุมได้เกิด Error และทางด้านPC ซึ่งเป็นจอแสดงสถานะทั้งหมดของอุปกรณ์จะแจ้งข้อผิดพลาดดังรูป

6.2 ข้อจำกัดในการเชื่อมต่ออุปกรณ์ควบคุม

จากรูปแบบในการสื่อสารเป็นแบบแพ็กเกจ จึงมีข้อจำกัดในการต่ออุปกรณ์ควบคุมดังนี้

-ตัวควบคุมหลัก สามารถต่อพ่วงกันได้มากสุด 255 ตัวเนื่องจากมีแอดแดรส 1-255 ส่วน แอดแดรส 0 สงวนไว้สำหรับสั่งตัวควบคุมหลักทุกตัว

-ในแต่ละตัวควบคุมหลัก สามารถต่อตังลูกข่ายได้มากที่สุด 15 ตัว ตั้งแต่แอดเดรส 1-15 แอดเดรส 0 สงวนไว้สำหลับสั่งตัวลูกข่ายทุกตัว ที่ต่อกับตัวควบคุมหลักนั้น

บทที่ 7
สรุปผลและวิจารณ์

7.1 ปัญหาที่พบและการแก้ไข

จากการทดลองการประยุกต์ใช้งาน RS-485 ในการควบคุมระบบนั้น ได้พบปัญหาใหญ่ๆ 2 ปัญหาด้วยกันคือ

7.1.1 ปัญหาของสัญญาณรบกวนในการส่งข้อมูล

ในการส่งข้อมูลตามมาตราฐานการเชื่อมต่อ RS-485 นั้น เมื่อทำการแปลงสัญญาณจากพอร์ทสื่อสาร(Com Port DB-25) ปรากฏว่า เมื่อทำการส่งสัญญาณจะเกิดความผิดพลาดขึ้น เมื่อทำการตรวจสอบจึงทราบว่าเป็นเพราะ อแดปเตอร์ที่ใช้ทำการส่งเป็นแบบใช้พลังงานในการทำงานต่ำ (Low Power) ดังนั้นกำลังในการส่งจึงมีการดึงกำลังไฟจากพอร์ทสื่อสาร ซึ่งพอร์ทสื่อสารไม่สามารถจ่ายกำลังให้เพียงพอได้ แต่การดึงกำลังนี้จะเกิดขึ้นเฉพาะช่วงที่อแดปเตอร์มีการส่งสัญญาณเท่านั้น

แก้ไขปัญหาได้โดยเปลี่ยนหม้อแปลงไฟฟ้าให้จ่ายไฟแยกเฉพาะที่ ทำให้การส่งข้อมูลมีประสิทธิภาพมากขึ้น เนื่องจากสัญญาณรบกวนมีผลต่อสัญญาณที่ส่งน้อยลงและเมื่ออแดปเตอร์ทำการส่งสัญญาณก็มีกำลังเพียงพอในการส่ง โดยไม่ต้องพึ่งไฟเลี้ยงจากวงจรภายนอก

7.2 สรุปผลและวิจารณ์

จากการทดลองประยุกต์ใช้งาน RS-485 ในระบบควบคุมนี้ ทำให้ทราบว่าในการส่งสัญญาณนั้นกำลังในการส่งสัญญาณจะมีผลอย่างมากต่อสัญญาณรบกวน ซึ่งทำให้การสื่อสารผิดพลาดหรืออาจถึงขั้นล้มเหลวได้ นอกจากนี้ยังทำให้ได้เรียนรู้ปัญหาในการส่งสัญญาณผ่านสายสัญญาณและการแก้ไขปัญหาในการรับส่งข้อมูลก็มีส่วนสำคัญอย่างยิ่งในการสื่อสาร

และจากการที่ได้ศึกษาเกี่ยวกับการสื่อสารแบบอนุกรม ก็ได้นำเอาความสามารถของการเชื่อมต่อตามมาตรฐาน RS-485 มาประยุกต์ใช้งานในการออกแบบระบบควบคุมโดยมีศูนย์กลางที่คอมพิวเตอร์ ซึ่งเราสามารถจะขยายโครงข่ายออกให้ได้ และในระยะทางที่ไกลมากขึ้น

ในส่วนของการเขียนโปรแกรมบนคอมพิวเตอร์ ได้จัดทำเป็นระบบฐานข้อมูล ซึ่งสามารถตรวจสอบสถานะของอุปกรณ์และบอกให้ผู้ใช้ทราบถึงสถานะนั้นในเวลาขณะนั้น

บรรณานุกรม

1. Frederick F. Driscoll, "Data Communication" : Saunders College Publishing, 1992,313 p.
2. Byron W. Putman, "RS-232 Simplified" : Prentice-Hall,Inc.,1987,190 p.
3. Charles M. Gilmore, "Microprocessors: Principles and Application" : McGraw-Hall, Inc.,1989,508 p.
4. ธานินทร์ ถาวรศาสนวงศ์, "การอินเทอร์เฟส IBM/PC" : ฟิสิกส์เซ็นเตอร์ การพิมพ์,2536,250 หน้า.
5. สุเจตน์ จันทรังษ์, "ไมโครคอนโทรลเลอร์ ชิพเดี่ยว 8051" : โครงการตำราวิชาการวิทยาลัยมหานคร,2535,178 หน้า.
6. สุนทร วิทูสุรพจน์, "การใช้งานไมโครคอนโทรลเลอร์ ตระกูล 8051" : ซีเอ็ดยูชั่น จำกัด,180 หน้า.
7. สมาคมวิศวกรรมสถานแห่งประเทศไทย ในพระบรมราชูปถมภ์, "ศัพท์เทคนิควิศวกรรมอิเล็กทรอนิกส์" : โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, พิมพ์ครั้งที่ 7, 2539,184 หน้า.

กิตติกรรมประกาศ

รายงานและผลงาน การประยุกต์ใช้งานวงจรอิเล็กทรอนิกส์ที่ได้จัดทำสำเร็จลุล่วงได้เป็นอย่างดี ขอกราบขอบพระคุณบิดา-มารดา ที่คอยอบรมสั่งสอนและได้ให้โอกาสการศึกษาแก่ข้าพเจ้าอีกทั้งยังคอยให้กำลังใจอยู่เสมอ และทางด้านเนื้อหา ความรู้ในการทำงานครั้งนี้ได้รับการชี้นำและคำปรึกษา จาก รศ.ดร.มนัส สังวรศิลป์ ซึ่งเป็นอาจารย์ที่ปรึกษา ผู้จัดทำรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณ ชุมนุมอิเล็กทรอนิกส์ที่ได้ เอื้อเฟื้อและให้การสนับสนุนทางด้านอุปกรณ์ , เครื่องมือ และอื่นๆอีกมากมายในการทำงาน ขอขอบคุณพี่ ๆ น้อง ๆ ชุมนุมอิเล็กทรอนิกส์ที่ได้ให้คำปรึกษา , ช่วยเหลือและเป็นกำลังใจอย่างใกล้ชิดเสมอมา

สุดท้ายขอขอบคุณภาควิชาอิเล็กทรอนิกส์ ที่ได้สนับสนุนสถานที่และเครื่องมือต่าง ๆ รวมทั้งคำปรึกษาจากอาจารย์ทุกท่าน

คุณค่าและประโยชน์อันพึงเกิดมีจากรายงานฉบับนี้ ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

ธนายุทธ วิริยะพงศ์

(นายธนายุทธ วิริยะพงศ์)

สิทธิพงษ์ เลิศวจนะ

(นายสิทธิพงษ์ เลิศวจนะ)

ผู้จัดทำ

ภาคผนวก

```
unit uMain;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, zPanel, zLed, ComCtrls, zonzoff, OleCtrls, MSCommLib_TLB, Std2,
  ExtCtrls, NMHTML, Psock, NMHttp;

type
  TForm1 = class(TForm)
    StatusBar1: TStatusBar;
    GroupBox1: TGroupBox;
    zLed1: TzLed;
    zLed2: TzLed;
    zLed3: TzLed;
    zLed4: TzLed;
    zLed5: TzLed;
    zLed6: TzLed;
    zLed7: TzLed;
    MSComm1: TMSComm;
    zcSingleOnOff1: TzcSingleOnOff;
    zcSingleOnOff2: TzcSingleOnOff;
    zcSingleOnOff3: TzcSingleOnOff;
    zcSingleOnOff4: TzcSingleOnOff;
    zcSingleOnOff5: TzcSingleOnOff;
    zcSingleOnOff6: TzcSingleOnOff;
    zcSingleOnOff7: TzcSingleOnOff;
    zcSingleOnOff8: TzcSingleOnOff;
    zLed8: TzLed;
    Edit1: TEdit;
    Timer1: TTimer;
    Edit2: TEdit;
    NMHTTP1: TNMHTTP;
    Edit3: TEdit;
```

```
    Edit4: TEdit;
    Edit5: TEdit;
    Button2: TButton;
    GroupBox2: TGroupBox;
    Button1: TButton;
    CheckBox1: TCheckBox;
    Label1: TLabel;
    Label2: TLabel;
    CheckBox2: TCheckBox;
    GroupBox3: TGroupBox;
    Edit6: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    procedure zcSingleOnOff1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure zcSingleOnOff2Click(Sender: TObject);
    procedure zcSingleOnOff3Click(Sender: TObject);
    procedure zcSingleOnOff4Click(Sender: TObject);
    procedure zcSingleOnOff5Click(Sender: TObject);
    procedure zcSingleOnOff6Click(Sender: TObject);
    procedure zcSingleOnOff7Click(Sender: TObject);
    procedure zcSingleOnOff8Click(Sender: TObject);
    procedure MSComm1Comm(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure Edit6Change(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
    Led : byte;
  end;

var
  Form1: TForm1;
  status : byte;
  ledstat : array[1..8] of boolean;
  count : byte ;
  command : byte;
  CheckOnline:integer;
  AlertFlag : boolean;
implementation


{$R *.DFM}

procedure SendDataOut();
var Buff : byte;
begin
    Buff:=status;
    form1.MSComm1.RTSEnable :=false;
    form1.MSComm1.output:=chr($AA)+chr(Buff);
    sleep(1);
    form1.MSComm1.RTSEnable:=true;
    form1.Edit1.text:=form1.edit1.text+inttostr(Buff)+'-';
end;


procedure TForm1.zcSingleOnOff1Click(Sender: TObject);
begin
    Zled1.Enabled := zcSingleOnOff1.SwitchOn ;
    if Zled1.Enabled then
     begin
         status:= status or $80;
```

```
      end
    else
      begin
         status:= status and $7F;
      end;
    SendDataout();
end;

procedure TForm1.FormShow(Sender: TObject);
begin
 if not MSComm1.PortOpen then MSComm1.PortOpen :=true;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
    status:=0;
    count:=1;
    command:=0;
    checkOnline:=1000;
    edit2.text:='http://203.146.49.162/cgi-
bin/web2pager.cgi?namefrom='+edit6.text+'&subject='+edit7.Text+'&message='+edit8.t
ext;
end;

procedure TForm1.zcSingleOnOff2Click(Sender: TObject);
begin
    Zled2.Enabled := zcSingleOnOff2.SwitchOn ;
    if Zled2.Enabled then
     begin
        status:= status or $40;
     end
    else
     begin
        status:= status and $BF;
     end;
```

```
    SendDataout();

end;

procedure TForm1.zcSingleOnOff3Click(Sender: TObject);
begin
    Zled3.Enabled := zcSingleOnOff3.SwitchOn ;
    if Zled3.Enabled then
     begin
        status:= status or $20;
     end
    else
     begin
        status:= status and $DF;
     end;
    SendDataout();

end;

procedure TForm1.zcSingleOnOff4Click(Sender: TObject);
begin
    Zled4.Enabled := zcSingleOnOff4.SwitchOn ;
    if Zled4.Enabled then
     begin
        status:= status or $10;
     end
    else
     begin
        status:= status and $EF;
     end;
    SendDataout();

end;

procedure TForm1.zcSingleOnOff5Click(Sender: TObject);
```

```
begin
    Zled5.Enabled := zcSingleOnOff5.SwitchOn ;
    if Zled5.Enabled then
     begin
        status:= status or $08;
     end
    else
     begin
        status:= status and $F7;
     end;
    SendDataout();

end;

procedure TForm1.zcSingleOnOff6Click(Sender: TObject);
begin
    Zled6.Enabled := zcSingleOnOff6.SwitchOn ;
    if Zled6.Enabled then
     begin
        status:= status or $04;
     end
    else
     begin
        status:= status and $FB;
     end;
    SendDataout();

end;

procedure TForm1.zcSingleOnOff7Click(Sender: TObject);
begin
    Zled7.Enabled := zcSingleOnOff7.SwitchOn ;
    if Zled7.Enabled then
     begin
        status:= status or $02;
```

```
        end
      else
        begin
            status:= status and $FD;
        end;
      SendDataout();

  end;

  procedure TForm1.zcSingleOnOff8Click(Sender: TObject);
  begin
      Zled8.Enabled := zcSingleOnOff8.SwitchOn ;
      if Zled8.Enabled then
        begin
            status:= status or $1;
        end
      else
        begin
            status:= status and $FE;
        end;

        //      status:=strtoint(edit1.text);
      SendDataout();

  end;

  procedure TForm1.MSComm1Comm(Sender: TObject);
  Var InputBuff : Variant;
     Arraybuff : array of byte ;
     A,Buff : byte ;
  begin
      case MSComm1.CommEvent of
        comevreceive:
        begin
            inputbuff := MSComm1.input;
```

```
        arraybuff:= inputbuff;
        Buff:= inputbuff[0];
        edit3.text:=edit3.text+inttostr(buff)+'-';
//*********************
        case count of
        1:begin
            command:=buff ;
            case buff of
            $AA:count:=2;
            213:begin
                Dec(CheckOnline);
                if Label1.Caption <> 'OK' then
                 begin
                   Label1.Caption := 'OK';
                   Label1.Font.Color :=cllime;
                   AlertFlag:=false;
                 end;
                count:=1;
              end;
            230:count:=2;
            end;  /// case command
         end;
        2:
  begin       //***** begin case 2
    case command of
     $AA,230 :
      begin   ////   $AA
        if (Buff AND $1) <> (Status AND $1) then
          begin
            zLed8.Enabled := not zLed8.Enabled;
            zcSingleOnOff8.SwitchOn :=zLed8.Enabled;
            if Zled8.Enabled then status:= status or $1
              else  status:= status and $FE;
          end;
        if (Buff AND $2) <> (Status AND $2) then
```

```
    begin
        zLed7.Enabled := not zLed7.Enabled;
        zcSingleOnOff7.SwitchOn :=zLed7.Enabled;
        if Zled7.Enabled then status:= status or $2
          else  status:= status and $FD;
    end;
  if (Buff AND $4) <> (Status AND $4) then
    begin
        zLed6.Enabled := not zLed6.Enabled;
        zcSingleOnOff6.SwitchOn :=zLed6.Enabled;
        if Zled6.Enabled then status:= status or $4
          else  status:= status and $FB;
    end;
  if (Buff AND $8) <> (Status AND $8) then
    begin
        zLed5.Enabled := not zLed5.Enabled;
        zcSingleOnOff5.SwitchOn :=zLed5.Enabled;
        if Zled5.Enabled then status:= status or $8
          else  status:= status and $F7;
    end;
  if (Buff AND $10) <> (Status AND $10) then
    begin
        zLed4.Enabled := not zLed4.Enabled;
        zcSingleOnOff4.SwitchOn :=zLed4.Enabled;
        if Zled4.Enabled then status:= status or 16
          else  status:= status and $EF;
    end;
  if (Buff AND $20) <> (Status AND $20) then
    begin
        zLed3.Enabled := not zLed3.Enabled;
        zcSingleOnOff3.SwitchOn :=zLed3.Enabled;
        if Zled3.Enabled then status:= status or 32
          else  status:= status and $DF;
    end;
  if (Buff AND $40) <> (Status AND $40) then
```

```
            begin
                zLed2.Enabled := not zLed2.Enabled;
                zcSingleOnOff2.SwitchOn :=zLed2.Enabled;
                if Zled2.Enabled then status:= status or 64
                  else  status:= status and $BF;
            end;
          if (Buff AND $80) <> (Status AND $80) then
            begin
                zLed1.Enabled := not zLed1.Enabled;
                zcSingleOnOff1.SwitchOn :=zLed1.Enabled;
                if Zled1.Enabled then status:= status or 128
                  else  status:= status and $7F;
            end;
        end;  ///// case $AA
       213:
        begin
          showmessage('อันนี้ไม่น่าเกิดนะ');
        end;
      end;    /// sub case --- command in count=2
   count:=1;
   end; //***** end case 2
   end; //******* end case count of
      end;                    // end Commreceive
      end;                    // end case commevent
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
    form1.MSComm1.RTSEnable :=false;
    form1.MSComm1.output:=chr($33);
//    sleep(1);
//    form1.MSComm1.output:=chr(strtoint(edit5.text));
  sleep(1);
    form1.MSComm1.RTSEnable:=true;
```

```
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    form1.MSComm1.RTSEnable :=false;
    form1.MSComm1.output:=chr(strtoint(edit4.text));
//    sleep(1);
//    form1.MSComm1.output:=chr(strtoint(edit5.text));
  sleep(1);
    form1.MSComm1.RTSEnable:=true;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var buff : string;
begin
    form1.MSComm1.RTSEnable :=false;
    form1.MSComm1.output:=chr($55);
    Inc(CheckOnline);
  sleep(1);
    form1.MSComm1.RTSEnable:=true;
    if CheckOnline > 1005 then
      begin
       label1.caption:='Error !!';
       label1.Font.Color:=clred;
       if (checkbox2.Checked) and (not AlertFlag) then
         begin
            AlertFlag:=true;
            Buff:=edit2.text;
            buff:=buff +' Time:'+Formatdatetime('HH:MM:SS',Time);
            NMHTTP1.Get(buff);        // Send pager
         end;
       CheckOnline:=1000;
      end;
      edit5.text:=inttostr(CheckOnline);
end;
```

```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
    Timer1.Enabled :=checkBox1.Checked;
    checkOnline:=1000;
end;

procedure TForm1.Edit6Change(Sender: TObject);
begin
    edit2.text:='http://203.146.49.162/cgi-
bin/web2pager.cgi?namefrom='+edit6.text+'&subject='+edit7.Text+'&message='+edit8.t
ext;
end;

end.
```

The AT89C51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 0
Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 may also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming, and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1
Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port 2
Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3
Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C51 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

Port 3 also receives some control signals for Flash programming and verification.

RST
Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

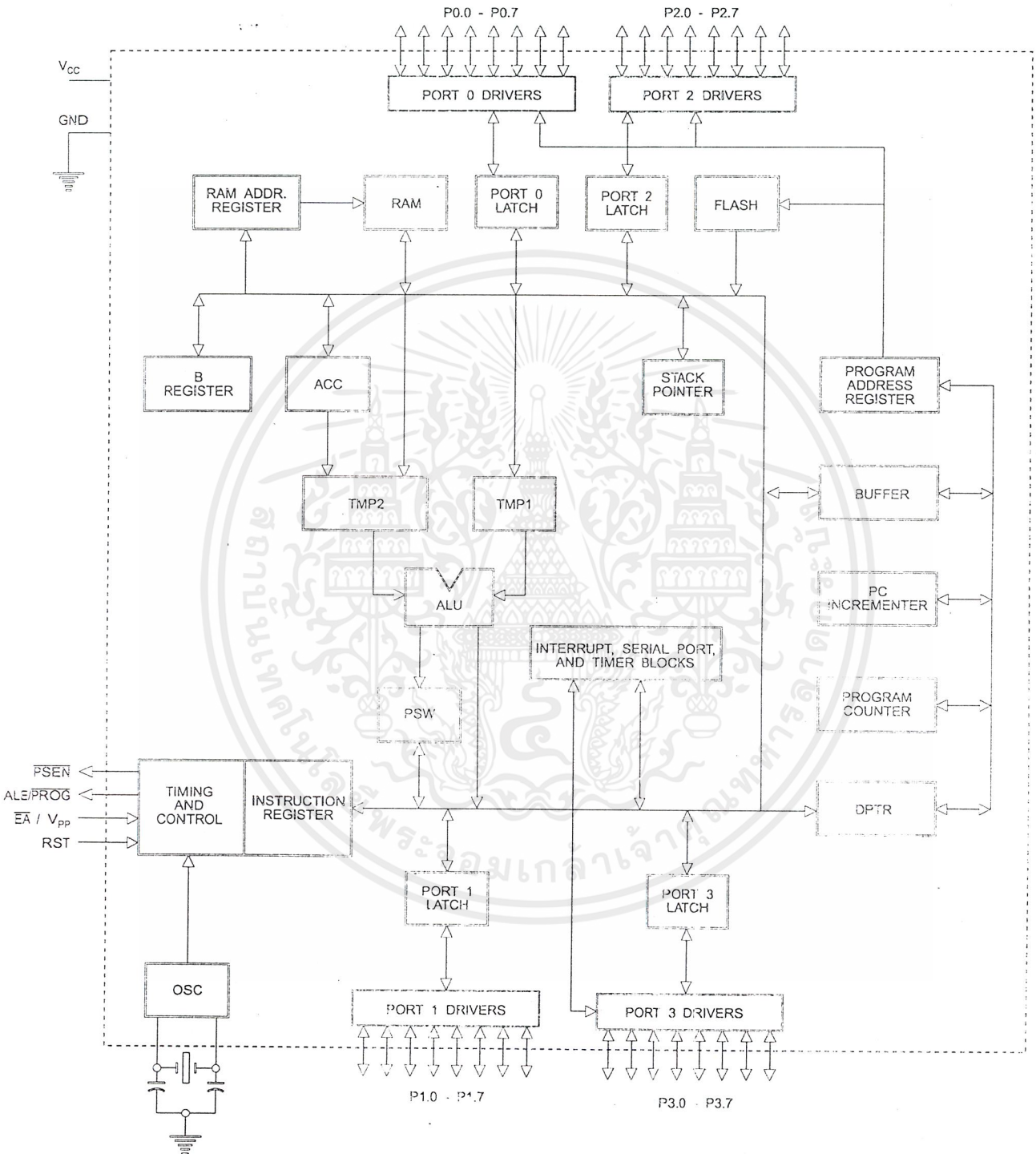
ALE/$\overline{PROG}$
Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{PSEN}$
Program Store Enable is the read strobe to external program memory.

BlockDiagram



Features

- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

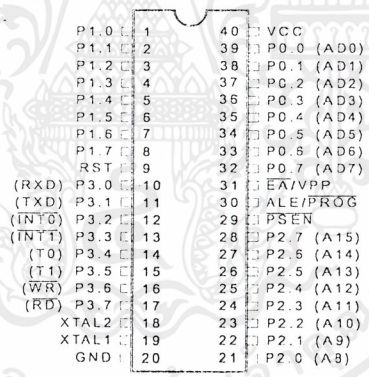
Description

The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

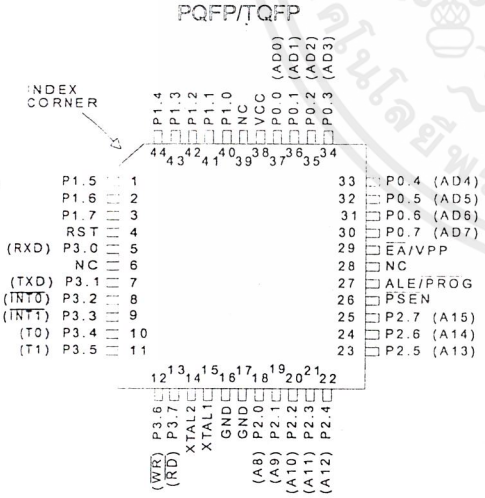
(continued)

Pin Configurations

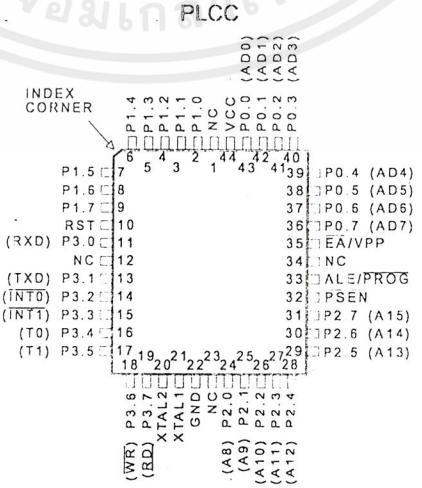
PDIP



PQFP/TQFP



PLCC



8-Bit Microcontroller with 4K Bytes Flash

AT89C51

0265F-A-12/97

When the AT89C51 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}/V_{PP}$

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming, for parts that require 12-volt V_{PP}.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

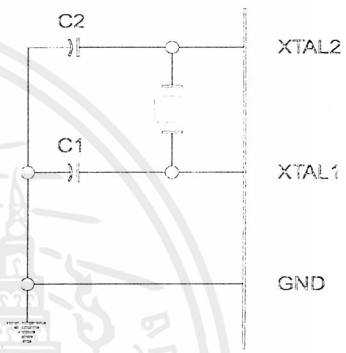
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

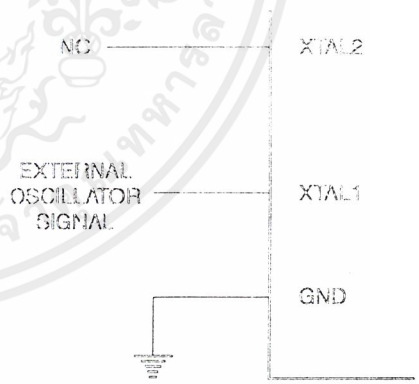
It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Figure 1. Oscillator Connections



Note: C1, C2 = 30pF ± 10pF for Crystals
= 40pF ± 10pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Status of External Pins During Idle and Power Down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

Power Down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Program Memory Lock Bits

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

Lock Bit Protection Modes

Program Lock Bits				Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash is disabled.
3	P	P	U	Same as mode 2, also verify is disabled.
4	P	P	P	Same as mode 3, also external execution is disabled.

Programming the Flash

The AT89C51 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage (V_{CC}) program enable signal. The low voltage programming mode provides a convenient way to program the AT89C51 inside the user's system, while the high-voltage programming mode is compatible with conventional third party Flash or EPROM programmers.

The AT89C51 is shipped with either the high-voltage or low-voltage programming mode enabled. The respective top-side marking and device signature codes are listed in the following table.

	V_{PP}=12V	V_{PP}=5V
Top-Side Mark	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
Signature	(030H)=1EH (031H)=51H (032H)=FFH	(030H)=1EH (031H)=51H (032H)=05H

The AT89C51 code memory array is programmed byte-by-byte in either programming mode. *To program any non-blank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.*

Programming Algorithm: Before programming the AT89C51, the address, data and control signals should be set up according to the Flash programming mode table and Figures 3 and 4. To program the AT89C51, take the following steps.

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}$ / V_{PP} to 12V for the high-voltage programming mode.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

Data Polling: The AT89C51 features Data $\overline{Polling}$ to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on PO.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. Data $\overline{Polling}$ may begin any time after a write cycle has been initiated.

Ready/$\overline{Busy}$: The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.4 is pulled low after ALE goes high during programming to indicate BUSY. P3.4 is pulled high again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The code array is written with all "1"s. The chip erase operation must be executed before the code memory can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H, 031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H)=1EH indicates manufactured by Atmel
(031H)=51H indicates 89C51
(032H)=FFH indicates 12V programming
(032H)=05H indicates 5V programming

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Flash Programming Modes

Mode		RST	PSEN	ALE/$\overline{PROG}$	EA/V_{PP}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L		H/12V	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit-1	H	L		H/12V	H	H	H	H
	Bit-2	H	L		H/12V	H	H	L	L
	Bit-3	H	L		H/12V	H	L	H	L
Chip Erase		H	L	(1)	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Note: 1. Chip Erase requires a 10-ms $\overline{PROG}$ pulse.

Figure3.ProgrammingtheFlash

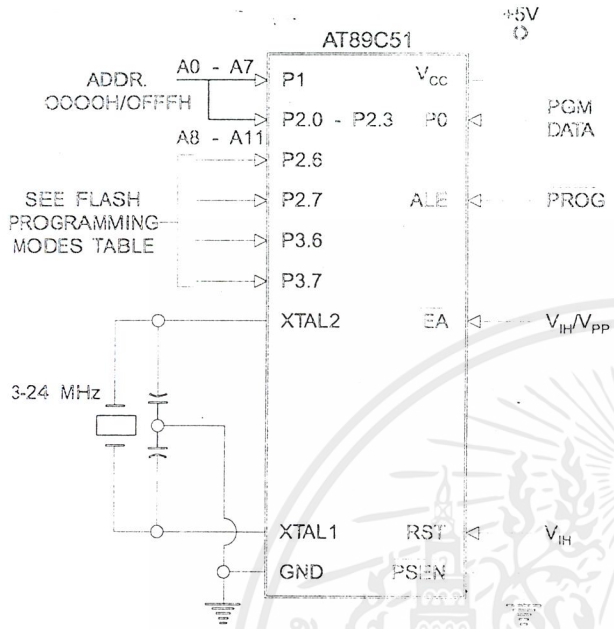
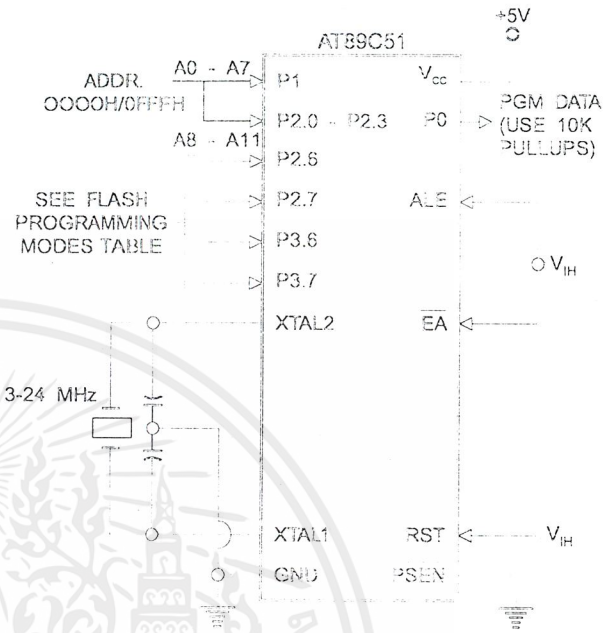


Figure4.VerifyingtheFlash



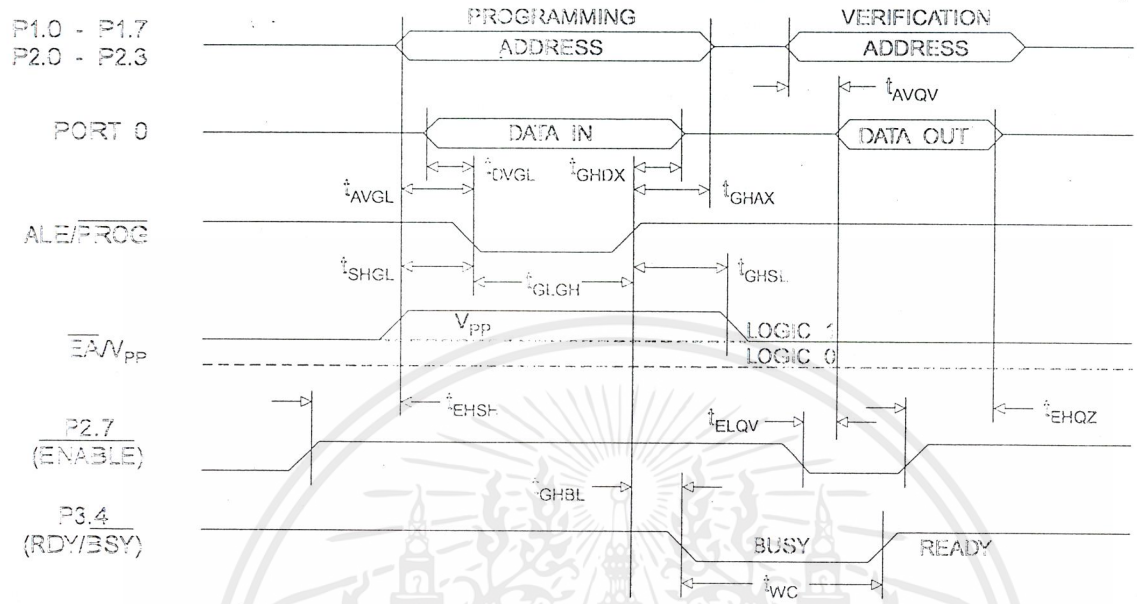
FlashProgrammingandVerificationCharacteristics

T_A=0°Cto70°C,V $_{CC}$=5.0 ±10%

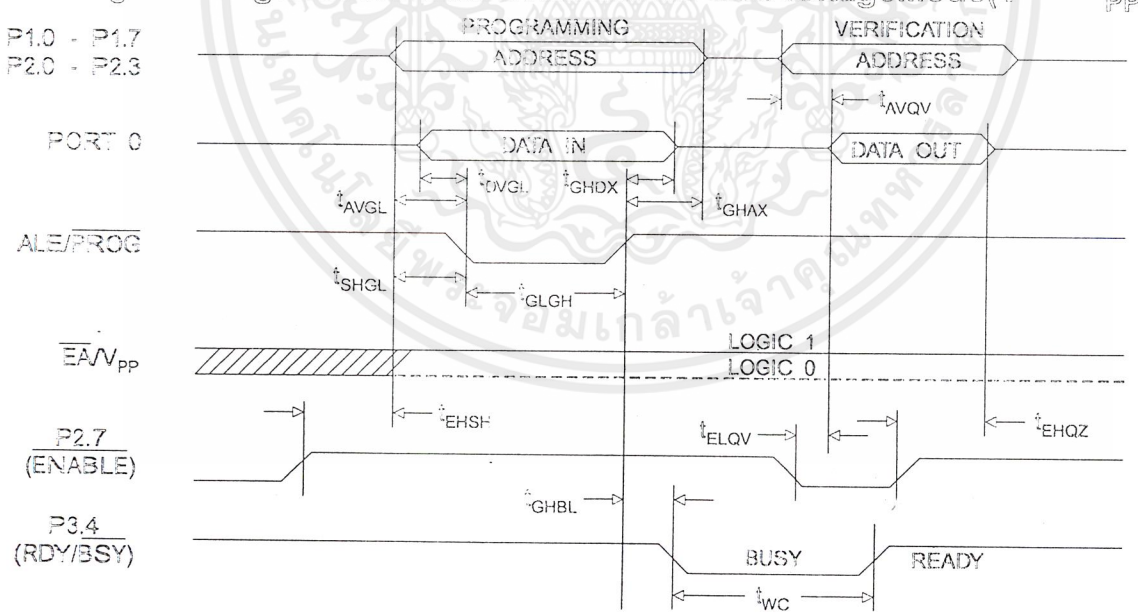
Symbol	Parameter	Min	Max	Units
V_{PP}(1)	ProgrammingEnableVoltage	11.5	12.5	V
I_{PP}(1)	ProgrammingEnableCurrent		1.0	mA
$1/t_{CLCL}$	OscillatorFrequency	3	24	MHz
t_{AVGL}	AddressSetuptoPROG Low	$48t_{CLCL}$		
t_{GHAX}	AddressHoldAfterPROG	$48t_{CLCL}$		
t_{DVGL}	DataSetuptoPROG Low	$48t_{CLCL}$		
t_{GHDX}	DataHoldAfterPROG	$48t_{CLCL}$		
t_{EHSH}	P2.7(ENABLE)HightoV $_{PP}$	$48t_{CLCL}$		
t_{SHGL}	V_{PP}SetuptoPROGLow	10		µs
t_{GHSL}(1)	V_{PP}HoldAfterPROG	10		µs
t_{GLGH}	PROGWidth	1	110	µs
t_{AVQV}	AddresstoDataValid		$48t_{CLCL}$	
t_{ELQV}	ENABLELowtoDataValid		$48t_{CLCL}$	
t_{EHQZ}	DataFloatAfterENABLE	0	$48t_{CLCL}$	
t_{GHBL}	PROGHightoBUSY Low		1.0	µs
t_{WC}	ByteWriteCycleTime		2.0	ms

Note: 1.Onlyusedin12-voltprogrammingmode.

Flash Programming and Verification Waveforms - High Voltage Mode (V_{PP} = 12V)



Flash Programming and Verification Waveforms - Low Voltage Mode (V_{PP} = 5V)



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	15.0mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_A=-40°C to 85°C, V_{CC} =5.0V±20% (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage	(Except EA)	-0.5	0.2V_{CC} -0.1	V
V_{IL1}	Input Low Voltage (EA)		-0.5	0.2V_{CC} -0.3	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2V_{CC} +0.9	V_{CC} +0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7V_{CC}	V_{CC} +0.5	V
V_{OL}	Output Low Voltage (1) (Ports 1,2,3)	I_{OL}=1.6mA		0.45	V
V_{OL1}	Output Low Voltage (1) (Port 0, ALE, PSEN)	I_{OL}=3.2mA		0.45	V
V_{OH}	Output High Voltage (Ports 1,2,3, ALE, PSEN)	I_{OH}=-60 µA, V_{CC}=5V ±10%	2.4		V
		I_{OH}=-25 µA	0.75V_{CC}		V
		I_{OH}=-10 µA	0.9V_{CC}		V
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	I_{OH}=-800 µA, V_{CC}=5V ±10%	2.4		V
		I_{OH}=-300 µA	0.75V_{CC}		V
		I_{OH}=-80 µA	0.9V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN}=0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN}=2V, VCC=5V± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, EA)	0.45<V_{IN}<V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq.=1MHz, T_A=25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12MHz		20	mA
		Idle Mode, 12MHz		5	mA
	Power Down Mode (2)	V_{CC}=6V		100	µA
		V_{CC}=3V		40	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 10mA
Maximum I_{OL} per 8-bit port: Port 0: 26mA
Ports 1,2,3: 15mA
Maximum total I_{OL} for all output pins: 71mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum V_{CC} for Power Down is 2V.

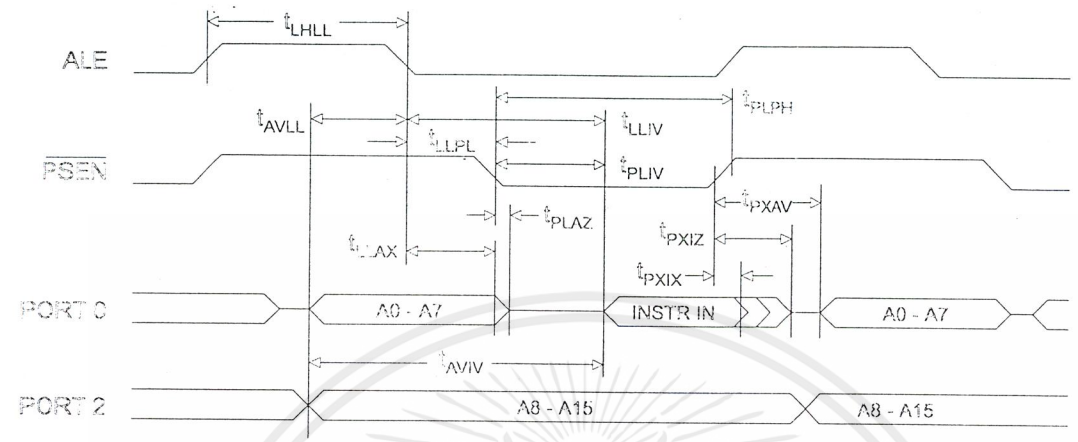
AC Characteristics

(Under Operating Conditions; Load Capacitance for Port 0, ALE/PROG, and PSEN = 100 pF; Load Capacitance for all other outputs = 80 pF)

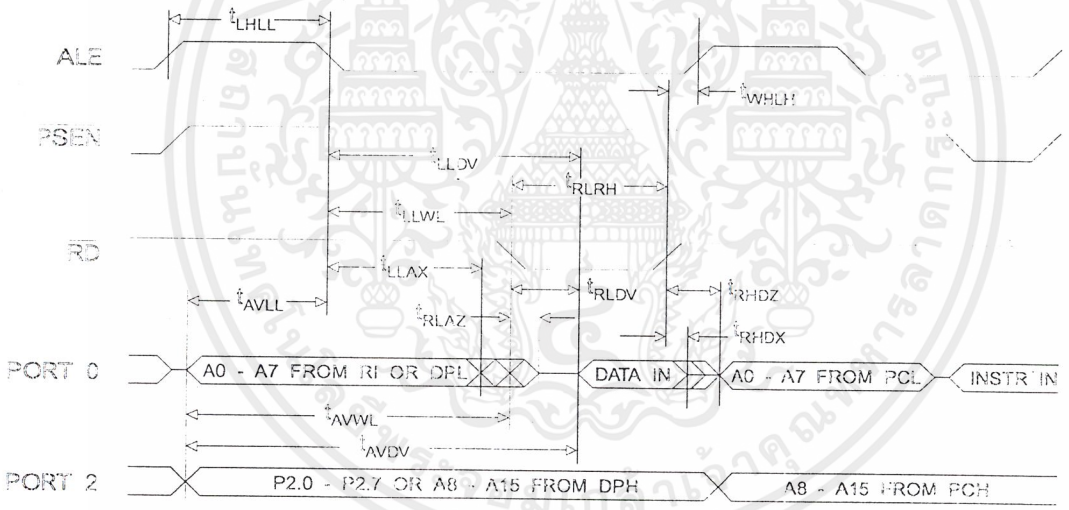
External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		16 to 24 MHz Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	24	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	Address Valid to ALE Low	43		$t_{CLCL}-13$		ns
t_{LLAX}	Address Hold After ALE Low	48		$t_{CLCL}-20$		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		$4t_{CLCL}-65$	ns
t_{LLPL}	ALE Low to PSEN Low	43		$t_{CLCL}-13$		ns
t_{PLPH}	PSEN Pulse Width	205		$3t_{CLCL}-20$		ns
t_{PLIV}	PSEN Low to Valid Instruction In		145		$3t_{CLCL}-45$	ns
t_{PXIX}	Input Instruction Hold After PSEN	0		0		ns
t_{PXIZ}	Input Instruction Float After PSEN		59		$t_{CLCL}-10$	ns
t_{PXAV}	PSEN to Address Valid	75		$t_{CLCL}-8$		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}-55$	ns
t_{PLAZ}	PSEN Low to Address Float		10		10	ns
t_{RLRH}	RD Pulse Width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	WR Pulse Width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	RD Low to Valid Data In		252		$5t_{CLCL}-90$	ns
t_{RHDX}	Data Hold After RD	0		0		ns
t_{RHDZ}	Data Float After RD		97		$2t_{CLCL}-28$	ns
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CLCL}-150$	ns
t_{AVDV}	Address to Valid Data In		585		$9t_{CLCL}-165$	ns
t_{LLWL}	ALE Low to RD or WR Low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	Address to RD or WR Low	203		$4t_{CLCL}-75$		ns
t_{QVWX}	Data Valid to WR Transition	23		$t_{CLCL}-20$		ns
t_{QVWH}	Data Valid to WR High	433		$7t_{CLCL}-120$		ns
t_{WHQX}	Data Hold After WR	33		$t_{CLCL}-20$		ns
t_{RLAZ}	RD Low to Address Float		0		0	ns
t_{WHLH}	RD or WR High to ALE High	43	123	$t_{CLCL}-20$	$t_{CLCL}+25$	ns

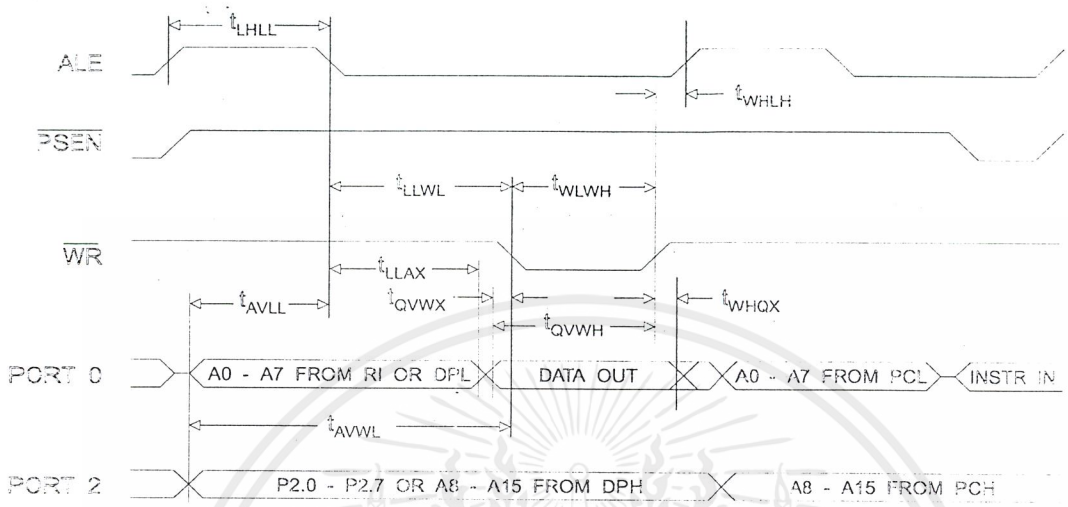
External Program Memory Read Cycle



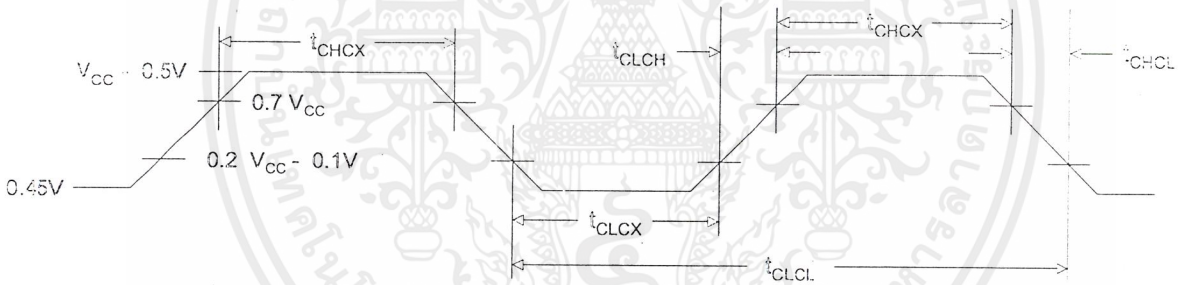
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

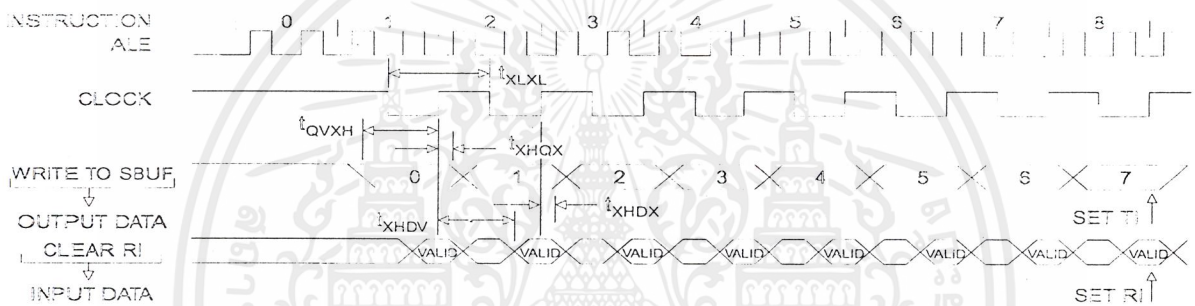
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

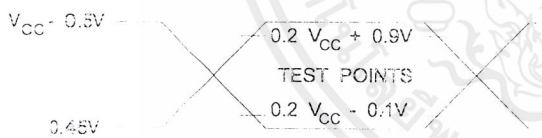
(V_{CC}=5.0V±20%; Load Capacitance=80pF)

Symbol	Parameter	12MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		12t_{CLCL}		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		10t_{CLCL}-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		2t_{CLCL}-117		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		10t_{CLCL}-133	ns

Shift Register Mode Timing Waveforms

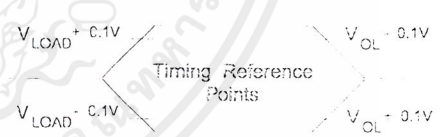


AC Testing Input/Output Waveforms (1)



Note: 1. AC Inputs during testing are driven at V_{CC}-0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

Float Waveforms (1)



Note: 1. For timing purposes, a port pin is no longer floating when a 100mV change from load voltage occurs. A port pin begins to float when 100mV change from the loaded V_{OH}/V_{OL} level occurs.

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	5V ±20%	AT89C51-12AC	44A	Commercial
		AT89C51-12JC	44J	(0°C to 70°C)
		AT89C51-12PC	40P6	
		AT89C51-12QC	44Q	
		AT89C51-12AI	44A	Industrial
		AT89C51-12JI	44J	(-40°C to 85°C)
		AT89C51-12PI	40P6	
		AT89C51-12QI	44Q	
		AT89C51-12AA	44A	Automotive
		AT89C51-12JA	44J	(-40°C to 105° C)
		AT89C51-12PA	40P6	
		AT89C51-12QA	44Q	
16	5V ±20%	AT89C51-16AC	44A	Commercial
		AT89C51-16JC	44J	(0°C to 70° C)
		AT89C51-16PC	40P6	
		AT89C51-16QC	44Q	
		AT89C51-16AI	44A	Industrial
		AT89C51-16JI	44J	(-40°C to 85 °C)
		AT89C51-16PI	40P6	
		AT89C51-16QI	44Q	
		AT89C51-16AA	44A	Automotive
		AT89C51-16JA	44J	(-40°C to 105 °C)
		AT89C51-16PA	40P6	
		AT89C51-16QA	44Q	
20	5V ±20%	AT89C51-20AC	44A	Commercial
		AT89C51-20JC	44J	(0°C to 70° C)
		AT89C51-20PC	40P6	
		AT89C51-20QC	44Q	
		AT89C51-20AI	44A	Industrial
		AT89C51-20JI	44J	(-40°C to 85 °C)
		AT89C51-20PI	40P6	
		AT89C51-20QI	44Q	