การควบคุมอุปกรณ์ไฟฟ้าภายในอาคารโดยผ่านโมเด็ม

(Distribute control system via modem)

โดย

นายฐิติกร ประดับสุข

นางสาวทวีพร ชูประยูร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การควบคุมอุปกรณ์ไฟฟ้าภายในอาคารโดยผ่านโมเด็ม

(Distribute control system via modem)

โดย

นายฐิติกร ประดับสุข เลขประจำตัว 40010196

นางสาวทวีพร ชูประยูร เลขประจำตัว 40010266

อาจารย์ที่ปรึกษา

รศ. ดร. วันชัย ริ้วรุจา

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมอุปกรณ์ไฟฟ้าภายในอาคารโดยผ่านโมเด็ม

(Distribute control system via modem)

ผู้จัดทำ

1. นายฐิติกร ประดับสุข
2. นางสาวทวีพร ชูประยูร

.....อาจารย์ที่ปรึกษา

(รศ.ดร.วันชัย ริ้วรุจา)

ระบบควบคุมอุปกรณ์ไฟฟ้าภายในอาคารโดยผ่านโมเด็ม

(Distribute control system via modem)

นายฐิติกร ประดับสุข
นางสาวทวีพร ชูประยูร
รศ.ดร.วันชัย ริ้วรุจา อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ระบบควบคุมอุปกรณ์ไฟฟ้าภายในอาคารโดยผ่านโมเด็มนี้จะใช้ระบบอินเตอร์เน็ตเข้ามาช่วยติดต่อกับอุปกรณ์ไฟฟ้าภายในอาคารโดยมีหลักการที่สำคัญ คือ ใช้โปรแกรม Delphi ในการควบคุมอุปกรณ์ไฟฟ้าติดต่อกับไมโครคอนโทรลเลอร์ซึ่งทำหน้าที่รับและส่งข้อมูลระหว่างเซิร์ฟเวอร์กับวงจรรีเลย์ และ ใช้โปรแกรม Delphi ในการติดต่อสื่อสารผ่านโมเด็มในระบบอินเตอร์เน็ตกับคอมพิวเตอร์ของผู้ใช้ โดยผู้ใช้จะใช้โปรแกรมเว็บเบราเซอร์ ทำการโหลดโปรแกรมควบคุมอุปกรณ์ไฟฟ้าจากเซิร์ฟเวอร์มายังเครื่องลูกข่าย หลังจากนั้นผู้ใช้จะสามารถเห็นสถานะและสามารถสั่งงานเปลี่ยนแปลงสถานะต่างๆของอุปกรณ์ไฟฟ้าภายในอาคารได้ตามต้องการ

ABSTRACT

Distribute control system via modem will use internet to connect electrical equipment in the building. The server is using Delphi programing to send and receive the data from microcontrollor , themain device controlling the electrical system, via modem transmission. User will use web browser program to load program from server to client's computer. After that user will be able to check and change status of electrical equipment in that building.

สารบัญ

เรื่อง	หน้า
บทที่ 1 บทนำ	1
บทที่ 2 ลักษณะของระบบ	2
บทที่ 3 บทบาทของโพรโตคอล HTTP	3
3.1 โพรโตคอล HTTP สำหรับเวิล์ดไวด์เว็บ	4
3.2 วิธีการติดต่อของโพรโตคอล HTTP	4
บทที่ 4 คอมมอนเกทเวย์อินเตอร์เฟส (Common Gateway Interface)	8
4.1 ภาษาที่ใช้เขียนซีจีไอ (CGI)	10
4.2 โปรแกรมซีจีไอหรือสคริปต์ซีจีไอ	11
บทที่ 5 พอร์ตสื่อสารข้อมูลแบบอนุกรม	12
5.1 การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรม	12
5.2 โหมดการทำงาน	13
5.3 อัตราเร็วในการรับและส่งข้อมูล (Baud Rate)	14
บทที่ 6 โครงสร้างของ MCS –51	17
6.1 โครงสร้างภายในของ MCS-51	18
6.2 พอร์ตของ MCS-51	20
6.3 วงจรคล็อคของ MCS-51	23
6.4 ผังเวลาของซีพียู (CPU Timing)	25
6.5 การต่อหน่วยความจำโปรแกรมและเก็บข้อมูลภายนอกชิพ	27
6.6 การแบ่งประเภทของหน่วยความจำ	27
บทที่ 7 โปรแกรม Delphi	33
7.1 การสร้างซีจีไอและแอปพลิเคชั่นด้วย Delphi	33
7.2 วิธีสร้างซีจีไอและแอปพลิเคชั่น	34
บทที่ 8 องค์ประกอบฮาร์ดแวร์ของระบบ	42
บทที่ 9 สรุปปัญหาและข้อเสนอแนะ	48
บรรณานุกรม	49
ภาคผนวก : ตัวอย่างโปรแกรม	
ดาต้าชีท	

สารบัญภาพ

รูปที่		หน้า
รูป 2.1	ลักษณะของระบบ	2
รูป 3.1	การติดต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์	5
รูป 3.2	ค้นหาเซิร์ฟเวอร์ ventura.lanna.com เพื่อสร้างการเชื่อมต่อ	6
รูป 3.3	ส่งคำร้องขอไปยังเซิร์ฟเวอร์ ventura.lanna.comและกำลังรอคำตอบ	6
รูป 3.4	เว็บเบราเซอร์กำลังส่งข้อมูลที่เซิร์ฟเวอร์ ventura.lanna.com ส่งมาให้	7
รูป 4.1	การร้องขอเอกสาร HTML จากเซิร์ฟเวอร์	8
รูป 4.2	การร้องขอข้อมูลจากเซิร์ฟเวอร์ผ่านซีจีไอ	9
รูป 6.1	8051 บล็อกไดอะแกรมของ MCS-51	18
รูป 6.2	ตำแหน่งต่างๆของริจิสเตอร์ต่างๆและหน่วยความจำเพื่อใช้ประกอบในการเขียนโปรแกรม	19
รูป 6.3	การจัดวางขาของ MCS-51	19
รูป 6.4	แสดงโครงสร้างของพอร์ต 0 (บิต)	20
รูป 6.5	แสดงโครงสร้างของพอร์ต 1 (บิต)	21
รูป 6.6	แสดงโครงสร้างของพอร์ต 2 (บิต)	21
รูป 6.7	แสดงโครงสร้างของพอร์ต 3 (บิต)	22
รูป 6.8	การต่อขารีเซ็ตให้กับ 8051	23
รูป 6.9	วงจรสร้างคล็อคของ 8051	24
รูป 6.10	ผังเวลาการทำงานของแต่ละคำสั่ง	25
รูป 6.11	แสดงผังเวลาการติดต่อกับหน่วยความจำภายนอก	26
รูป 6.12	การต่อหน่วยความจำโปรแกรมและดาต้าภายนอกชิพ	27
รูป 6.13	ผังหน่วยความจำสำหรับโปรแกรมสำหรับเบอร์ 8051	28
รูป 6.14	ผังหน่วยความจำสำหรับโปรแกรมสำหรับเบอร์ 8052	28
รูป 6.15	ผังหน่วยความจำสำหรับหน่วยความจำสำหรับเก็บข้อมูลเบอร์ 8051	28
รูป 6.16	ผังหน่วยความจำสำหรับหน่วยความจำสำหรับเก็บข้อมูลเบอร์ 8052	29
รูป 6.17	128 ไบต์ของแรมที่เข้าถึงข้อมูลแบบทางตรงและทางอ้อม	30
รูป 6.18	แสดงตำแหน่งบิตต่างๆในตำแหน่งแอดเดรส (20h-2Fh)	31
รูป 6.19	แสดงสัญลักษณ์ ชื่อและตำแหน่งต่างๆที่มีอยู่ใน SFR	32
รูป 7.1	แสดงหน้าต่าง Editing WebModule1.Actions/logon	35
รูป 7.2	แสดงหน้าต่าง Editing WebModule1.Actions/logUser	37

รูปที่		หน้า
รูป 7.3	แสดงโปรแกรมหน้าล๊อกออน	39
รูป 7.4	แสดงโปรแกรมหน้าล๊อกยูสเซอร์	39
รูป 7.5	โฟล์ชาร์ตของไมโครคอลโทรลเลอร์	40
รูป 7.6	โฟล์ชาร์ตของโปรแกรมเดลไฟ	41
รูป 8.1	การทำงานของวงจรเรียงกระแสเต็มคลื่นแบบบริดจ์	43
รูป 8.2	แสดงแรงดันตกคร่อม R_Lของวงจรบริดจ์	44
รูป 8.3	การฟิลเตอร์ในวงจรภาคจ่ายไฟ	45
รูป 8.4	ไดโอดเปล่งแสง	46
รูป 8.5	แสดงไดโอดเปล่งแสง	46
รูป 8.6	การทำงานของแอลอีดี	47
รูป 8.7	สเป็กตรัมของแอลอีดีในรูปแบบกราฟ	47

สารบัญตาราง

ตารางที่		หน้า
ตาราง 5.1	ค่าที่ต้องนำไปไว้ในริจิสเตอร์ของไทม์เมอร์1 เมื่อใช้อัตราการรับและส่งข้อมูลมาตรฐานต่างๆ	15
ตาราง 6.1	คุณสมบัติไมโครคอนโทรลเลอร์ตระกูล MCS-51	17

บทที่ 1
บทนำ

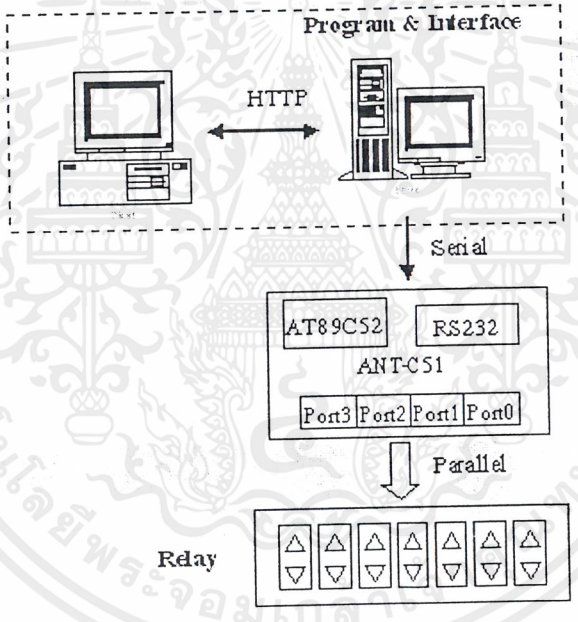
ปัจจุบันอินเตอร์เน็ต (Internet) หรืออภิมหาเครือข่ายคอมพิวเตอร์ กลายเป็นโลกยุคใหม่ใบเล็กๆและเป็นสถานที่สำหรับผู้คนที่ท่องอยู่ในโลกของเครือข่ายคอมพิวเตอร์ประมาณ 5 หมื่นเครือข่าย หรือกว่า 30 ล้านเครื่องต่อเชื่อมถึงกัน โดยลักษณะการเชื่อมต่อของเครือข่ายคอมพิวเตอร์ที่ประกอบไปด้วยเครือข่ายคอมพิวเตอร์ขนาดทั้งเล็กและใหญ่จำนวนมากเข้าด้วยกัน โดยมีข้อกำหนดว่าทุกเครือข่ายที่เชื่อมต่อถึงกัน จะต้องอยู่ภายใต้มาตรฐานของการเชื่อมต่อ(โพรโตคอล) ที่ถูกสร้างขึ้นมาเพื่อใช้งานบนเครือข่ายแบบนี้โดยเฉพาะ ซึ่งเรียกว่า TCP/IP เหมือนกันหมดทุกเครือข่าย โดยนำเสนอมูลข้อมูลที่เป็นแบบมัลติมีเดีย ซึ่งประกอบไปด้วยภาพกราฟฟิก เสียง ข้อมูลและสัญญาณวีดีโอที่ชื่อว่า World Wide Web

ในระบบเว็บจะประกอบไปด้วยเครื่องที่ให้บริการข้อมูล (Web Server) หรืออาจจะเรียกว่าเว็บไซต์ (Web Site) กับเครื่องรับข้อมูล (Web Client) โดยที่เว็บไซต์แต่ละที่จะต้องเชื่อมต่ออยู่กับผู้ให้บริการอินเตอร์เน็ต (Internet Service Provider) ที่ต่างโดยจะต้องจัดเตรียมข้อมูลในรูปแบบไฮเปอร์เท็กซ์เอาไว้ ซึ่งในปัจจุบันนิยมใช้ภาษามาตรฐานที่เรียกว่า HTML (Hyper Text Markup Language) ที่ประกอบด้วยไฟล์ข้อความที่บรรจุคำสั่งในการทำงานไว้ภายใน และเมื่อผู้ใช้ต้องการติดต่อกับเวิลด์ไวด์เว็บเพียงแค่มีซอฟต์แวร์ที่เรียกว่า เว็บบราวเซอร์ (Web Browser) หรือบริการออนไลน์และอุปกรณ์สื่อสาร หรือที่เรียกว่า โมเด็ม เพื่อติดต่อไปยังศูนย์บริการข้อมูลหรือเซิร์ฟเวอร์ จากนั้นชี้แล้วคลิกเลือกข้อมูลที่ผู้ใช้ต้องการค้นหา โดยใช้โปรแกรมเว็บบราวเซอร์จะเป็นตัวที่แปลงสัญญาณคำสั่งและแสดงผลเป็นข้อความ รูปภาพ เสียง ซึ่งเป็นข้อมูลที่ผู้ใช้ต้องการและข้อมูลจะถูกดาวน์โหลดมายังคอมพิวเตอร์ของผู้ใช้ เพราะความง่ายและสะดวกต่อผู้ใช้งาน จึงทำให้อินเตอร์เน็ตได้รับความนิยมอย่างรวดเร็วและมีผู้เข้ามาใช้บริการอย่างมากมาย

โครงงานนี้ถูกสร้างขึ้นเพื่อใช้ประโยชน์จากเว็บซึ่งมีขนาดครอบคลุมทั่วโลกในการควบคุมรีเลย์จากระยะไกล และอาจประยุกต์โครงงานนี้ในการควบคุมแบบอื่นๆได้อีกด้วย

บทที่ 2
ลักษณะของระบบ

ทางฝั่งผู้ใช้จะควบคุมการเปิดปิดอุปกรณ์ไฟฟ้าในอาคารได้โดย ใช้คอมพิวเตอร์ที่ต่ออยู่กับโมเด็มโดยผู้ใช้จะใช้โปรแกรมเว็บเบราเซอร์ติดต่อเข้ามาด้วย HTTP (Hypertext Transfer Protocol) เซิร์ฟเวอร์จะโหลดไฟล์ HTML ที่ใช้ควบคุมอุปกรณ์ไฟฟ้า ให้กับเครื่องของผู้ใช้ ผู้ใช้จะส่งข้อมูลสถานะใหม่กลับมายังเซิร์ฟเวอร์ เซิร์ฟเวอร์จะทำการอ่านข้อมูลและส่งต่อให้กับบอร์ดไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์จะควบคุมวงจรรีเลย์ให้เปิดปิดอุปกรณ์ไฟฟ้าได้ตามที่ผู้ใช้ต้องการ



รูป 2.1 ลักษณะของระบบ

บทที่ 3
บทบาทของโพรโตคอล HTTP

สำหรับโปรแกรมเมอร์หรือผู้ที่อยากจะเขียนโปรแกรมเพื่อติดต่อสื่อสารผ่านเว็บหรืออินเตอร์เน็ตแล้ว สิ่งหนึ่งที่จะต้องรู้และเข้าใจคือเรื่องของโพรโตคอล (Protocol) เช่นเว็บเบราเซอร์จะใช้โพรโตคอล HTTP (HyperText Transfer Protocol) ในการรับส่งข้อมูล การรับส่งไฟล์อาจจะใช้ HTTP หรือ FTP (File Transfer Protocol) ก็ได้ ฯลฯ จึงอาจมีคำถามว่า โพรโตคอลเป็นโปรแกรมหรือว่าเป็นอุปกรณ์ทางฮาร์ดแวร์หรือเปล่า มาพร้อมกับเว็บเบราเซอร์หรือเปล่า มันทำงานอย่างไร และจะไปดูได้ที่ไหน

ในความเป็นจริงแล้วโพรโตคอลเป็นเพียงข้อตกลงกันระหว่าง 2 ฝ่ายที่จะติดต่อหรือพูดคุยกัน ไม่ได้เป็นโปรแกรมหรืออุปกรณ์ฮาร์ดแวร์ใดๆ ถ้าตั้งคำถามว่าทำไมต้องมีโพรโตคอล คำตอบสั้นๆ คือ เพื่อให้การติดต่อระหว่าง 2 ฝ่ายเป็นไปอย่างราบรื่นและถูกต้องมากที่สุด เพราะมีกฎเกณฑ์หรือโพรโตคอลควบคุมอยู่ สิ่งที่พอจะเรียกได้ว่าเป็นโพรโตคอลในชีวิตประจำวันก็คือไฟจราจร เราตั้งกฎเกณฑ์กันว่าเห็นไฟเหลืองให้เตรียมพร้อม ไฟเขียวถึงจะออกตัวได้ ถ้าเป็นไฟแดงให้หยุดแล้วทุกคนก็ใช้สิ่งที่เป็นข้อตกลงร่วมกัน ปัญหาการจราจรจึงลดลงไปบ้าง

ในทางอินเตอร์เน็ตและคอมพิวเตอร์ก็เช่นเดียวกัน เมื่อต้องมีการสื่อสารระหว่างเครื่องคอมพิวเตอร์ ก็จำเป็นต้องกำหนดกฎเกณฑ์อะไรบางอย่างให้ตรงกันก่อน จึงจะพูดคุยแลกเปลี่ยนข้อมูลกันได้ ดังนั้นถ้าเครื่อง 2 เครื่องจะคุยกันก็ต้องเลือกโพรโตคอลใดโพรโตคอลหนึ่งมาเป็นตัวควบคุมการเจรจา สมมติว่าเราเขียนโปรแกรมเพื่อรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ 2 เครื่องผ่านพอร์ตขนาน (Parallel Port) แล้วตั้งกฎเกณฑ์ง่ายๆดังต่อไปนี้เป็นโพรโตคอล

1. เครื่องไหนที่จะเป็นตัวแม่ (Server) ต้องเปิดเครื่องรอ
2. ตัวลูก (Client) จะส่งข้อมูลขนาด 5 ไบต์ ไปถามตัวแม่
3. ถ้าตัวแม่พร้อมที่จะทำงาน จะนำข้อมูล 5 ไบต์นั้นส่งกลับให้ตัวลูก
4. ถ้าตัวลูกได้รับข้อมูล 5 ไบต์กลับมาจากตัวแม่ แสดงว่าตัวแม่พร้อมแล้วตัวลูกจะส่งชื่อไฟล์ที่ต้องการไปให้ตัวแม่
5. เมื่อตัวแม่ได้รับชื่อไฟล์แล้ว จะส่งไฟล์นั้นไปยังเครื่องตัวลูกที่รอรับ โดยส่งครั้งละ 52 ไบต์
6. ตัวลูกอ่านข้อมูลจากพอร์ตทีละ 52 ไบต์แล้วนำมาต่อกันจนได้เป็นไฟล์ที่สมบูรณ์

อาจเห็นว่าโพรโตคอลแค่นี้จะทำงานได้อย่างไร ไม่เห็นมีการตรวจสอบความผิดพลาดของข้อมูล จะรู้ได้ยังไงว่าได้รับข้อมูลครบแล้ว ที่ยกตัวอย่างมาก็เพื่อให้เห็นภาพของคำว่าโพรโตคอลเท่านั้น ถ้าจะเขียนโปรแกรมกันจริงๆ แล้วต้องมีรายละเอียดที่มากกว่านี้ นั่นคือต้องมากำหนดรายละเอียดของโพรโตคอลให้ชัดเจนในระดับบิตเลยทีเดียว เพราะทราบแล้วว่าในทาง

คอมพิวเตอร์นั้นไม่เหมือนกับชีวิตจริงอย่างการพูดคุยผ่านโทรศัพท์ ถ้าเสียงไม่ชัดเจนยังพอจะเดาได้ แต่ถ้าข้อมูลทางคอมพิวเตอร์ผิดพลาดแค่บิตเดียว มันอาจหมายถึงความผิดพลาดทั้งระบบเลยก็ได้

3.1 โพรโตคอล HTTP สำหรับ เวิลด์ไวด์เว็บ

กลับมาถึงเรื่องโพรโตคอล HTTP ยอดนิยมในอินเตอร์เน็ต โพรโตคอลนี้สร้างขึ้นสำหรับบริการที่เรียกว่าเวิลด์ไวด์เว็บ (World Wide Web หรือ WWW) ในเครือข่ายอินเตอร์เน็ตโดยเฉพาะ โพรโตคอลนี้จะเป็นตัวกำหนดวิธีส่งข้อมูลหรือไฟล์(ส่วนมากมักเรียกรวมว่า ทรัพยากร หรือ resource) ระหว่างเครื่องคอมพิวเตอร์ที่เป็นไคลเอนต์กับเครื่องคอมพิวเตอร์ที่เป็นเซิร์ฟเวอร์ รวมถึงกำหนดกฎระเบียบในการติดต่อด้วย

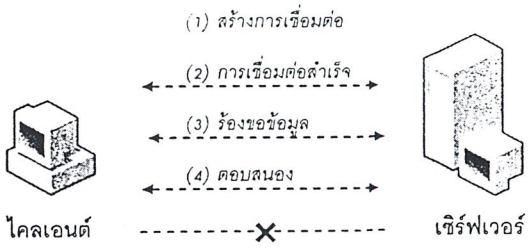
HTTP ถูกพัฒนาโดย นายเบอร์เนอร์ ลี (Berners-Lee) แห่ง CERN ในช่วงปี ค.ศ. 1990-1991 โพรโตคอลนี้ช่วยให้บริการเวิลด์ไวด์เว็บ ได้รับความนิยมและแพร่หลายมากขึ้นกว่าเดิม เพราะเป็นตัวควบคุมการรับส่งข้อมูลได้ทั้งภาพและเสียงจนเกือบจะเป็นมัลติมิเดีย จากเดิมที่แลกเปลี่ยนได้เพียงข้อความอย่างเดียว

ในอินเตอร์เน็ตนั้นนอกจากบริการเวิลด์ไวด์เว็บ แล้ว ยังมีบริการอีกหลายอย่าง เช่น FTP หรืออีเมล (E-mail) จึงมีคนสร้างโปรแกรมเพื่อให้ใช้บริการเหล่านี้ได้ง่ายขึ้นอย่างโปรแกรมชุดติดตั้งของเว็บเบราเซอร์รุ่นใหม่ๆ จะมีโปรแกรมใช้งานสารพัดเรียกได้ว่าครบเครื่อง เช่น Netscape Communicator จะมี Navigator เป็นเว็บเบราเซอร์สำหรับใช้บริการเวิลด์ไวด์เว็บ และ FTP และมี Netscape Messenger สำหรับใช้บริการอีเมล แต่บางโปรแกรมก็สร้างขึ้นมาสำหรับใช้บริการเดี่ยวๆ เช่น Eudora เป็นโปรแกรมสำหรับใช้บริการอีเมลเพียงอย่างเดียว เป็นต้น

โปรแกรมดังกล่าวเหล่านี้ ล้วนเป็นโปรแกรมสื่อสารผ่านเว็บหรืออินเตอร์เน็ตทั้งสิ้น ผู้เขียนโปรแกรมเหล่านี้ จึงต้องศึกษาเรื่องโพรโตคอลให้เข้าใจแล้วปฏิบัติตามโพรโตคอลอย่างเคร่งครัด ถ้าจะเขียนโปรแกรมสำหรับรับส่งอีเมล แต่ไม่ทำตามโพรโตคอลรับส่งอีเมลตามที่วางกฎเกณฑ์กันไว้ ก็คงต้องเตรียมตัวใช้โปรแกรมนั้นคนเดียวเพราะไปคุยกับคนอื่นจะไม่รู้เรื่อง ยกเว้นแต่ว่าสามารถสร้างโพรโตคอลใหม่ที่ดีกว่าเดิม แล้วนำมาเผยแพร่จนกระทั่งผู้ใช้ในอินเตอร์เน็ตยอมรับนำไปใช้งานและกลายเป็นมาตรฐาน ซึ่งก็อาจเป็นไปได้ เพราะทุกวันนี้ก็มีโพรโตคอลต่างๆออกมาให้ทดลองใช้กันอย่างมากมาย

3.2 วิธีการติดต่อของโพรโตคอล HTTP

ด้วยเหตุที่การทำงานของโพรโตคอล HTTP เป็นแบบ ไคลเอนต์และเซิร์ฟเวอร์ (client / server) ดังนั้นการติดต่อสื่อสารใดๆ ผ่านโพรโตคอลนี้จำเป็นต้องมีเครื่องคอมพิวเตอร์ตัวลูกกับตัวแม่ การสื่อสารจึงจะสมบูรณ์ได้ การติดต่อกันระหว่างไคลเอนต์ไปยังเซิร์ฟเวอร์ผ่านโพรโตคอล HTTP (เช่นการเปิดเว็บเพจผ่านเว็บเบราเซอร์) จะมีขั้นตอนอย่างไรบ้าง



รูป 3.1 การติดต่อสื่อสารระหว่างไคลเอนต์กับเซิร์ฟเวอร์

จากรูป ขั้นแรกคือไคลเอนต์ (ในตอนนี้คือเว็บเบราเซอร์)จะสร้าง การเชื่อมต่อ(Connection) กับเซิร์ฟเวอร์ผ่านสิ่งที่เรียกว่าซ็อกเก็ต(Socket) เมื่อซ็อกเก็ตทั้งสองฝั่งเสียบเชื่อมต่อกันได้สำเร็จ ไคลเอนต์จะส่ง คำร้องขอข้อมูล(Request) ไปยังเซิร์ฟเวอร์ จากนั้นเซิร์ฟเวอร์จะไปหาข้อมูลที่ไคลเอนต์ต้องการ ซึ่งไม่ว่าจะมีหรือไม่มีข้อมูลตามที่ไคลเอนต์ร้องขอ เซิร์ฟเวอร์ก็จะต้องส่ง ข้อมูลตอบสนอง (Response) กลับมายังไคลเอนต์เสมอ สุดท้ายการเชื่อมต่อจะถูกตัดขาดหรือปลดการเชื่อมต่อของซ็อกเก็ตทั้งสองฝั่งออกนั่นเอง

ด้วยการทำงานของโพรโตคอล HTTP ที่มีการเชื่อมต่อในระยะเวลาเพียงสั้นๆ หรือที่เรียกว่าเป็นแบบโพรโตคอลแบบ Connectionless ในลักษณะดังกล่าว ทำให้ช่วงเวลาหนึ่งๆ เซิร์ฟเวอร์ที่ให้บริการเวิลด์ไวด์เว็บ สามารถรองรับไคลเอนต์ได้จำนวนมากพร้อมๆกัน เพราะไม่มีใครทำการเชื่อมต่ออย่างถาวร

จากสถานการณ์ต่อไปนี้ หากสมมติว่าไฟล์โฮมเพจของเซิร์ฟเวอร์ ventura.lanna.com ถูกกำหนดไว้ให้เป็นชื่อ index.html และมีเนื้อหาในไฟล์ดังต่อไปนี้

```
<html>
<head><title>Infothai</title></head>
<body background = “linethai.gef”>
<img src=”under.gif”>
</body>
</html>
```

ถ้าเราสั่งให้เว็บเบราเซอร์ติดต่อไปยัง ventura.lanna.com เว็บเบราเซอร์จะต้องค้นหาเซิร์ฟเวอร์ ventura.lanna.com ให้ได้เสียก่อน เพื่อทำการเชื่อมต่อโดยจะปรากฏข้อความแจ้งให้ทราบที่แถบสถานะ(Status Bar) ของเว็บเบราเซอร์ดังรูป 3.2

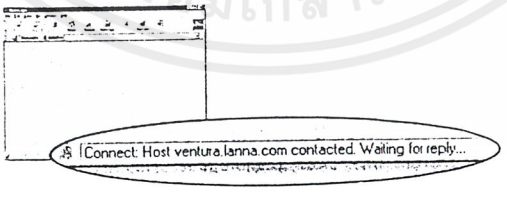
เมื่อเชื่อมต่อได้แล้ว เว็บเบราเซอร์จะส่งข้อความร้องขอไปยังเซิร์ฟเวอร์ดังรูป 3.3 แต่เนื่องจากเรามิได้บอกว่าต้องการไฟล์ชื่ออะไร (เพราะระบุเพียงแค่ ventura.lanna.com) เมื่อเซิร์ฟเวอร์รับทราบการร้องขอแล้วพบว่าไม่มีการบอกระบุชื่อไฟล์ ดังนั้นเว็บเบราเซอร์จะส่งไฟล์ที่เป็นไฟล์

โฮมเพจกลับมาให้แทน ซึ่งก็คือไฟล์ index.html ดังรูป 3.4 เมื่อเว็บเบราเซอร์ได้รับไฟล์มาแล้ว เซิร์ฟเวอร์จะตัดการเชื่อมต่อทันที

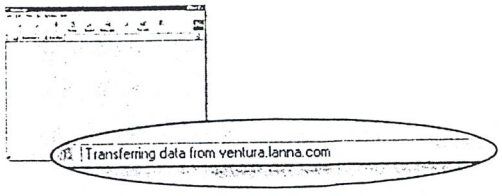
ต่อจากนั้น เมื่อเว็บเบราเซอร์เริ่มอ่านและตีความหมายของแท็กไฟล์ index.html จึงพบว่า รูปพื้นหลังของเว็บเพจระบุให้ใช้ไฟล์ linethai.gif ดังนั้นเว็บเบราเซอร์ต้องสร้างการเชื่อมต่อไปยัง ventura.lanna.com เป็นครั้งที่สองเพื่อขอไฟล์ linethai.gif จากเซิร์ฟเวอร์อีก เมื่อเซิร์ฟเวอร์ส่งไฟล์ linethai.gif ให้เว็บเบราเซอร์แล้ว การเชื่อมต่อก็ถูกตัดขาดลงอีกครั้งหนึ่ง จากนั้นเมื่อเว็บเบราเซอร์อ่านพบในบรรทัดต่อมาว่าต้องแสดงรูปที่ชื่อ under.gif ในเว็บเพจ เว็บเบราเซอร์ก็ต้องสร้างการเชื่อมเพื่อขอข้อมูลอีก การทำงานของเว็บเบราเซอร์จะเป็นอย่างนี้ไปเรื่อยๆ ดังนั้นการเปิดเว็บเพจหนึ่งๆ จึงอาจมีการเชื่อมต่อกับเซิร์ฟเวอร์เพื่อขอข้อมูลหลายครั้ง

สังเกตการทำงานนี้ได้ที่ตัวเลขเปอร์เซ็นต์การโหลดข้อมูลที่แถบสถานะของเว็บเบราเซอร์ เมื่อเว็บเบราเซอร์เริ่มโหลดรูปกราฟิกใดๆ เปอร์เซ็นต์จะเริ่มวิ่งจาก 0 นั่นแสดงว่ามีการเชื่อมต่อเพื่อขอรูปนั้นมาได้ และเซิร์ฟเวอร์กำลังทยอยส่งข้อมูลของรูปกราฟิกมาให้เว็บเบราเซอร์ จนกระทั่งได้ข้อมูลรูปภาพครบหลังจากนั้นหากมีการโหลดรูปหรือข้อมูลอื่นๆ อีก ตัวเลขเปอร์เซ็นต์การโหลดจะเริ่มต้นที่ 0 อีกครั้ง

รูป 3.2 ค้นหาเซิร์ฟเวอร์ ventura.lanna.com เพื่อสร้างการเชื่อมต่อ



รูป 3.3 ส่งคำร้องขอไปยังเซิร์ฟเวอร์ ventura.lanna.com และกำลังรอคำตอบ

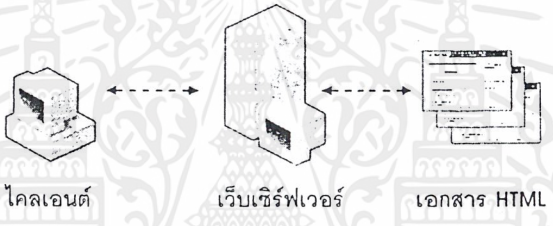


รูป 3.4 เว็บเบราเซอร์กำลังรับข้อมูลที่เซิร์ฟเวอร์ ventura.lanna.com ส่งมาให้

บทที่ 4
คอมมอนเกทเวย์อินเตอร์เฟส (Common Gateway Interface)

คำเต็มซีจีไอ (CGI) นั้นคือ Common Gateway Interface แต่ก็อาจไม่ทราบความหมายที่ชัดเจน รู้เพียงว่าเป็นประตูหรือทางสำหรับเชื่อมโยงอะไรบางอย่างเท่านั้น เพราะอาจเข้าใจว่าซีจีไอเป็นภาษาอีกภาษาหนึ่งเหมือนกับ C หรือ Pascal บ้างก็บอกว่าซีจีไอ คือ ภาษาPerl หรือไม่ก็บอกว่าซีจีไอ คือโปรแกรมที่รันอยู่ในเซิร์ฟเวอร์ เลยสรุปไม่ได้ว่า ซีจีไอเป็นภาษาหรือว่าเป็นอะไรกันแน่

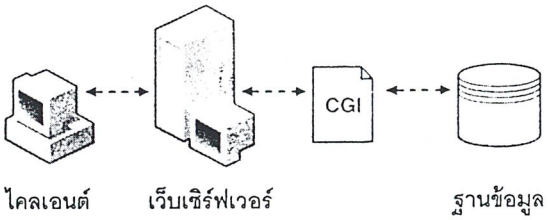
ซีจีไอเป็นแอปพลิเคชั่น ซึ่งทำงานเป็นตัวกลางระหว่างเซิร์ฟเวอร์กับไคลเอนต์ เราทราบแล้วว่าข้อมูลที่จะให้บริการผ่านเวิลด์ไวด์เว็บได้นั้น จะต้องถูกเก็บอยู่ในรูปแบบของเอกสาร HTML เมื่อเซิร์ฟเวอร์ได้รับการร้องขอไฟล์จากไคลเอนต์ เว็บเซิร์ฟเวอร์จะค้นหาและส่งไฟล์ที่ไคลเอนต์ต้องการกลับไปให้ดังรูป 4.1



รูป 4.1 การร้องขอเอกสาร HTML จากเซิร์ฟเวอร์

ปัญหาของข้อมูลที่เก็บด้วยรูปแบบ HTML คือ เมื่อจะต้องมีการอัปเดตข้อมูลจะเป็นงานที่ยุ่งยากและเสียเวลาเป็นอย่างมาก เพราะว่าไฟล์เอกสาร HTML มีลักษณะการจัดเก็บแบบตายตัว (Static) ยิ่งถ้ามีข้อมูลมากด้วยแล้ว การจัดเก็บข้อมูลแยกออกเป็นไฟล์ๆ ยิ่งจะทำให้ดูแลแก้ไขได้ยากมากขึ้นเงาตามตัว เป้าหมายของการใช้ซีจีไออย่างหนึ่งก็คือ ทำให้เอกสาร HTML ที่ผู้ใช้ร้องขอเข้ามามีความยืดหยุ่นหรือที่เรียกว่าเป็นแบบไดนามิก (Dynamic)

วิธีการทำเอกสาร HTML ให้มีความเป็นไดนามิกคือ แทนที่จะเก็บข้อมูลแยกเป็นไฟล์ HTML หลายๆไฟล์ ก็อาจจะเก็บข้อมูลทั้งหมดไว้ในไฟล์เดียวเมื่อผู้ใช้ต้องการข้อมูลอะไรสักอย่างก็กำหนดให้ป้อนเงื่อนไขที่ต้องการเข้ามาให้แก่ซีจีไอ หลังจากนั้นซีจีไอจะไปค้นหรือดึงเอาเฉพาะข้อมูลที่ตรงตามที่ผู้ใช้ต้องการจากนั้นจึงนำข้อมูลมาสร้างเป็นเอกสาร HTML แล้วส่งกลับไปแสดงให้ผู้ใช้ดู ดังนั้นเอกสาร HTML ที่ผู้ใช้แต่ละคนได้รับอาจไม่เหมือนกัน ขึ้นอยู่กับเงื่อนไขความต้องการของผู้ใช้ ในกรณีนี้ซีจีไอจะทำหน้าที่เป็นประตู (Gateway) ระหว่างฐานข้อมูลในเซิร์ฟเวอร์กับไคลเอนต์นั่นเอง ดังรูป 4.2



รูป 4.2 การร้องขอข้อมูลจากเซิร์ฟเวอร์ผ่านซีจีไอ

ลักษณะการทำงานของซีจีไอต้องอาศัยการประมวลผลที่เซิร์ฟเวอร์ แล้วสร้างคำตอบออกมาเป็นเนื้อหาแบบ HTML จากนั้นจึงส่งเนื้อหากลับไปให้ไคลเอนต์ เซิร์ฟเวอร์ใดที่ยอมให้มีการรันซีจีไอได้จึงต้องทำงานหนักกว่าเซิร์ฟเวอร์ที่ให้บริการเอกสาร HTML เพียงอย่างเดียว แนวความคิดการทำงานของซีจีไอจะเป็นแบบที่เรียกว่า รวมศูนย์ หรือ Centralize งานทุกอย่างต้องวิ่งเข้ามารันที่เซิร์ฟเวอร์หมด ไคลเอนต์เพียงแต่ทำหน้าที่ส่งคำร้องขอและรอรับผลการทำงานเท่านั้น ซึ่งดูเหมือนว่าซีจีไอจะขัดกับแนวความคิดในการทำงาน แบบกระจายศูนย์ หรือ Distribute ที่พยายามลดการเข้าไปรันงานในเซิร์ฟเวอร์ แต่ให้ไคลเอนต์ทำงานให้มากที่สุด

แนวคิดการทำงานแบบกระจายศูนย์ ทำให้โปรแกรมต่างๆที่ใช้ทำงานกับเวิลด์ไวด์เว็บในปัจจุบันอย่างเว็บเบราเซอร์ มีการพัฒนาความสามารถใหม่ๆ เพิ่มขึ้นมากมาย(โปรแกรมจึงมีขนาดใหญ่ขึ้นเรื่อยๆ) เพื่อทำงานบางอย่างทางฝั่งไคลเอนต์เลยโดยไม่ต้องส่งไปทำงานที่เซิร์ฟเวอร์ ตัวอย่างที่ชัดเจนอย่างหนึ่งคือ JavaScript ซึ่งเกิดขึ้นมาเพื่อช่วยลดจุดด้อยของ HTML งานบางอย่างที่เมื่อก่อนต้องส่งให้ซีจีไอในเซิร์ฟเวอร์ตรวจสอบ ก็เปลี่ยนมาใช้ JavaScript ในฝั่งไคลเอนต์ช่วยตรวจสอบให้หากใครที่เคยใช้อินเตอร์เน็ตมาก่อนที่จะมี JavaScript เกิดขึ้นจะทราบดี คือการป้อนข้อมูลอะไรสักอย่างให้แก่ฟอร์มในเว็บเพจ ถ้าเราป้อนข้อมูลที่ผิดลงไปโดยไม่ทราบหรือไม่ทันสังเกต แล้วคลิกปุ่มส่งข้อความเข้าเซิร์ฟเวอร์ บางทีใช้เวลานานหลายนาทีกว่าเซิร์ฟเวอร์จะตอบกลับมาเพียงว่า “คุณป้อนข้อมูลไม่ครบ กรุณาป้อนที่อยู่ของคุณด้วย” หรือ “คุณป้อนอีเมลแอดเดรสผิด” สาเหตุที่ช้าเพราะว่าเว็บเบราเซอร์ต้องส่งข้อมูลไปให้ซีจีไอ ในเซิร์ฟเวอร์เป็นผู้ตรวจสอบความถูกต้องทั้งหมด

แต่ถึงแม้ในปัจจุบัน JavaScript จะเข้ามาช่วยทำงานตรวจสอบข้อมูลลักษณะนี้ได้ก็ตาม ก็ไม่ได้หมายความว่าซีจีไอจะทำงานเบากว่าเดิม อาจหนักเท่าเดิมหรือมากกว่าด้วยซ้ำ เหตุการณ์เกี่ยวกับเรื่องความไว้วางใจให้ JavaScript กับ HTML ช่วยตรวจสอบข้อมูลในฟอร์ม แล้วทำให้เกิดปัญหาในการทำงานของซีจีไอมีซ้ำแล้วซ้ำเล่า

ตัวอย่างที่เห็นชัดๆ คือการบอมบ์ (Bomb) ข้อความเข้าเซิร์ฟเวอร์ การบอมบ์ที่ว่านี้คือ ส่งข้อความยาวมากๆ เข้าไปยังเซิร์ฟเวอร์ที่เป็นเว็บบอร์ด (Web Board) มีหลายเซิร์ฟเวอร์ที่คนเขียนซีจีไอไว้วางใจให้ JavaScript เป็นผู้ตรวจสอบความยาวของข้อมูลที่จะส่งเข้ามายังเซิร์ฟเวอร์ จึงไม่เขียน

ซีจีไอตรวจสอบความยาวข้อมูลอีกครั้งในซีจีไอ ผู้ใช้ที่ต้องการจะแกล้งบอมบ์ข้อความเข้าเซิร์ฟเวอร์ เพียงแต่คัดลอกซอร์ซโค้ด HTML ของเว็บเพจที่ใช้ป้อนข้อมูลออกมา แล้วนำมาสร้างเป็นไฟล์ HTML ใหม่โดยตัดการทำงานส่วนที่ตรวจสอบความยาวข้อความออกไปเสีย จากนั้นก็คัดลอก (Copy) ข้อความขยะจากไฟล์ก็ได้นำมาวาง (Paste) ลงในช่องรับข้อมูลหลายๆครั้ง แล้วคลิกส่งข้อความในฟอร์มให้ซีจีไอในเซิร์ฟเวอร์รับไปทำงาน

จะเห็นได้ว่า JavaScript หรือ ภาษาสคริปต์อื่นๆ ที่ทำงานร่วมกับ HTML ในฝั่งไคลเอนต์นั้น ไม่สามารถควบคุมข้อมูลให้ถูกต้อง 100% เพราะฉะนั้นการตรวจสอบด้วยซีจีไอที่เซิร์ฟเวอร์ก็ยังเป็นสิ่งที่จำเป็นอยู่ดี อีกประการหนึ่งคือโพรโตคอล HTTP เป็นโพรโตคอลที่เปิดกว้างมาก ไม่ว่าไคลเอนต์ใช้ระบบปฏิบัติการอะไร เครื่องรุ่นไหน ก็สามารถติดต่อกันได้หมด ข้อมูลแทบจะไม่มีอะไรเป็นความลับได้เลย หากนำมาใช้กับโพรโตคอลนี้ จึงเป็นเรื่องยากที่จะควบคุมไคลเอนต์ได้

งานที่ซีจีไอทำได้ดีและเป็นเป้าหมายของซีจีไอคือ การทำงานที่ติดต่อกับเซิร์ฟเวอร์ผ่านเว็บเบราเซอร์ เช่น การค้นหาหรือการเก็บ ข้อมูล นอกจากนี้ซีจีไอยังใช้งานอื่นๆ ได้อีกสารพัด เช่น สร้างตัวเลขเคาน์เตอร์ (Counter) ในเว็บเพจ สร้างสมุดเซ็นเยี่ยม (Guest Book) ฯลฯ และสามารถใช้ซีจีไอทำงานได้หลายอย่างเท่าที่ความสามารถของภาษาจะอำนวย การเขียนซีจีไอนั้นเพียงแต่เปลี่ยนวิธีการติดต่อกับผู้ใช้ (User Interface) จากคอมมานด์ไลน์ (Command Line) หรือวินโดว์ (Window) ที่คุ้นเคยมาเป็นเว็บเบราเซอร์และการรับส่งข้อมูลแบบพื้นฐานของโพรโตคอล HTTP เท่านั้นเอง

4.1 ภาษาที่ใช้เขียนซีจีไอ

เมื่อทราบว่าซีจีไอเป็นโปรแกรมที่จะต้องรันอยู่ในเซิร์ฟเวอร์ ดังนั้นสามารถใช้ภาษาใดๆ ในการเขียนเป็นซีจีไอ เช่น C,Pascal,Visual Basic,Delphi,Perl หรือ UNIX Shell ก็ได้ ขึ้นอยู่กับระบบปฏิบัติการของเซิร์ฟเวอร์ว่ายอมให้รันอะไรได้บ้าง และยังขึ้นอยู่กับความเหมาะสมในการใช้งาน ผสมกับความถนัดของผู้เขียนโปรแกรมด้วย เช่น ถ้าเป็นงานที่ต้องจัดการกับข้อความมากๆแล้ว Perl น่าจะเป็นทางเลือกที่น่าสนใจ เพราะ Perl มีฟังก์ชันจัดการเกี่ยวกับข้อความที่ดี สามารถเขียนสคริปต์ได้สั้นมาก

ถ้าเป็นงานด้านฐานข้อมูลแล้ว Visual Basic หรือ Delphi น่าจะเหมาะสมกว่า แต่ถึงอย่างไรก็ตามในปัจจุบันนี้เกือบทุกภาษาที่นำมาเขียนเป็นซีจีไอจะมี ไลบรารี (Library) หรือซอร์ซโค้ดเพิ่มเติมให้ นำไปใช้ประกอบการเขียนโปรแกรมได้งานขึ้น โดยไม่ต้องเขียนโปรแกรมเองทั้งหมด

ระบบปฏิบัติการ ก็เป็นสิ่งสำคัญต่อการเลือกใช้ภาษาที่จะเป็นซีจีไอด้วยเช่น Perl จะทำงานได้เต็มประสิทธิภาพมากที่สุดเมื่อรันใน UNIX ถึงแม้ว่าตอนนี้จะมี Perl เวอร์ชัน Win32 สำหรับระบบปฏิบัติการในตระกูล Windows ก็ตาม แต่จะใช้ความสามารถของ Perl ได้ไม่เต็มที่ เพราะบางคำสั่งของ Perl เชื่อมโยงกับการทำงานของ UNIX ด้วย ปัจจุบันส่วนใหญ่จะนิยมใช้ Perl เชื่อมโยงกับการทำงานของ UNIX ด้วยปัจจุบันส่วนใหญ่จะนิยมใช้ Perl เขียนซีจีไอเพราะเซิร์ฟเวอร์ให้

บริการเวิลด์ไวด์เว็บในอินเตอร์เน็ต มักจะเป็น UNIX เสียส่วนใหญ่ และ Perl มักจะถูกติดตั้งแถมมาพร้อมกับระบบปฏิบัติการอยู่แล้ว ไม่ต้องหามาติดตั้งเสริมแต่อย่างใด ถ้ามีเว็บเซิร์ฟเวอร์ที่รันอยู่ใน Windows 95 หรือ NT การใช้ภาษา C หรือ Pascal ก็เป็นทางเลือกที่น่าสนใจเหมือนกัน

4.2 โปรแกรมซีจีไอหรือสคริปต์ซีจีไอ

บางทีเรียกโปรแกรมซีจีไอที่รันอยู่ในเซิร์ฟเวอร์ว่าเขียนโปรแกรมซีจีไอแต่บางทีก็เรียกว่าเขียนสคริปต์ซีจีไอด้วย Perl หรือเขียน Perl Script ถ้ามองในแง่ของการทำงานแล้ว ทั้งสองอย่างก็ได้ผลการทำงานออกมาเหมือนกัน ต่างกันตรงที่เนื้อหาในไฟล์ของโปรแกรมซีจีไอกับสคริปต์ซีจีไอนั้นไม่เหมือนกัน

หากซอร์ซโค้ด (Source Code) ของซีจีไอตัวใดในเซิร์ฟเวอร์ถูกเขียนจากภาษาที่เป็นคอมไพเลอร์ (Compiler) เช่น C หรือ Delphi แล้วการที่จะรันโปรแกรมได้ ก็ต้องคอมไพล์ซอร์ซโค้ดให้กลายเป็นแบบไฟล์แบบรันได้ (Executable File) เสียก่อน คือได้ไฟล์ที่มีนามสกุลเป็น .exe แล้วนำไฟล์นี้ไปใช้งานเป็นซีจีไอทำงานอยู่ใน เซิร์ฟเวอร์ หากเป็นลักษณะนี้ก็พอจะเรียกได้ว่าเขียนโปรแกรมซีจีไอและรันโปรแกรมซีจีไอ

แต่บางภาษาที่ใช้เขียนในซีจีไอ เช่น Perl จะไม่มีการคอมไพล์ซอร์ซโค้ดให้เป็นไฟล์ที่รันได้อย่างอิสระ ดังนั้น จึงต้องนำซอร์ซโค้ดของโปรแกรมไปวางไว้ในเซิร์ฟเวอร์รอไคลเอนต์มาเรียกให้ทำงาน เมื่อซอร์ซโค้ดถูกเรียกให้ทำงาน เว็บเซิร์ฟเวอร์จะเรียกตัวแปลภาษา (Interpreter) ของ Perl ให้มารับเอาสคริปต์นั้นไปทำงานทีละคำสั่งตามที่เขียนไว้ในสคริปต์ ผลการทำงานของสคริปต์จะส่งไปให้เว็บเซิร์ฟเวอร์ เพื่อที่เว็บเซิร์ฟเวอร์จะส่งผลการทำงานนั้นกลับไปให้ไคลเอนต์อีกทีหนึ่ง ดังนั้นจึงต้องเรียกว่าซีจีไอนั้นว่า สคริปต์ซีจีไอไม่ใช่โปรแกรมซีจีไอ

ถ้าจะเปรียบเทียบโปรแกรมซีจีไอและสคริปต์ซีจีไอ กับการทำงานของโปรแกรมในระบบดอสแล้ว โปรแกรมซีจีไอก็เหมือนไฟล์นามสกุล .exe หรือ .com ที่เรียกมารันได้ทันที และสคริปต์ซีจีไอก็จะคล้ายกับการเขียนแบตซ์ไฟล์ (Batch File) นั่นเอง

เมื่อดูจากลักษณะการทำงานของโปรแกรมกับสคริปต์ ซีจีไอที่ได้จากการเขียนด้วยภาษาที่เป็นสคริปต์น่าจะทำงานได้ช้ากว่า เพราะเมื่อรันสคริปต์ทุกครั้งก็ต้องเรียกตัวแปลภาษามาแปลสคริปต์ตลอด ซึ่งปรากฏว่าถ้าเปรียบเทียบประสิทธิภาพความเร็วในการทำงานแล้ว ภาษาที่เป็นสคริปต์อาจทำงานช้ากว่าบ้างจริง แต่ภาษาที่เป็นแบบสคริปต์ก็มีข้อดีพอที่จะชดเชยข้อด้อยได้บ้าง อย่างน้อยถ้าเกิดข้อผิดพลาดในสคริปต์ขึ้นมาโดยสามารถแก้ไขและทดลองรันสคริปต์ได้ทันทีโดยไม่ต้องเสียเวลาในการคอมไพล์ นอกจากนี้ภาษาสคริปต์อย่าง Perl ก็มีฟังก์ชันสำหรับการเขียนโปรแกรมเครือข่ายได้ง่าย และยังรันได้ทั้งเครื่องที่เป็น UNIXและ Windows ในขณะที่บางภาษาอย่าง Delphi หรือ Visual Basic ไม่สามารถนำมารันในเซิร์ฟเวอร์ UNIX ได้

บทที่ 5
พอร์ตสื่อสารข้อมูลแบบอนุกรม

พอร์ตอนุกรม (Serial Communication Device) หรือเรียกว่า Serial Port โดยปกติเครื่องคอมพิวเตอร์จะมีพอร์ตชนิดนี้อยู่แล้ว 2 พอร์ต คือ พอร์ตขนาด 9 ขา (9- Pins) มีรูปร่างเหมือนสี่เหลี่ยมคางหมู มีเข็มยื่นออกมา 9 เข็ม เรียกหัวชนิดนี้ว่า DB-9 Connector Male Type และ พอร์ตขนาด 25 ขา (25-Pins) ซึ่งมีเข็มจำนวน 25 เข็มยื่นออกมา เรียกหัวชนิดนี้ว่า DB-25 Connector Male Type ซึ่งพอร์ตขนาด 9 ขาจะต่อกับเมาส์ เพื่อควบคุมเคอร์เซอร์บนระบบวินโดวส์

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีพอร์ตสำหรับสื่อสารข้อมูลแบบอนุกรมที่สามารถรับและส่งข้อมูลแบบอนุกรมได้ โดยที่ผู้ใช้ไม่จำเป็นต้องต่อชิพที่ทำหน้าที่รับหรือส่งข้อมูลแบบอนุกรมโดยเฉพาะเพิ่มแต่อย่างใดเลย การนำMCS-51ไปประยุกต์ใช้งานที่ต้องมีการติดต่อสื่อสารข้อมูลแบบอนุกรมกับวงจรภายนอกอื่นๆจึงทำได้สะดวกและมีความคล่องตัวสูงมาก

5.1 การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรม

พอร์ตสื่อสารข้อมูลแบบอนุกรมที่มีใน MCS-51 สามารถทำงานได้ในแบบฟูลดูเพลกซ์ (Full Duplex) หมายความว่า MCS-51 สามารถรับและส่งข้อมูลได้พร้อมๆกัน โดยในการรับข้อมูลจะมีการบัฟเฟอร์ข้อมูลให้ด้วย จึงทำให้ MCS-51 สามารถกำหนดการรับข้อมูลไบต์ที่สองซึ่งถูกส่งตามเข้ามาก่อนที่ไบต์แรกที่ได้รับเข้ามาจะถูกอ่านจากรีจิสเตอร์ใช้งานเฉพาะที่ใช้สำหรับรับข้อมูล (Receive Register) เพื่อนำไปเก็บไว้ในหน่วยความจำต่อไป (หากไบต์แรกยังไม่ถูกอ่านเมื่อได้รับไบต์ที่สองเรียบร้อยแล้ว ข้อมูลจะหายไปหนึ่งไบต์)

พอร์ตสื่อสารข้อมูลแบบอนุกรมใน MCS-51 ประกอบด้วยรีจิสเตอร์ขนาด 8 บิต จำนวนสองตัว แต่ละตัวมีชื่อเรียกตามหน้าที่ดังนี้คือ

1.รีจิสเตอร์สำหรับรับข้อมูลใช้รับข้อมูลที่ส่งเข้ามาจากภายนอก

2.รีจิสเตอร์สำหรับส่งข้อมูล (Transmit Register) ใช้ส่งข้อมูลจาก MCS-51 ออกไปภายนอก

รีจิสเตอร์ทั้งสองมีตำแหน่งเดียวกันในรีจิสเตอร์ใช้งานเฉพาะ คือ ตรงกับตำแหน่งของรีจิสเตอร์ใช้เฉพาะ SBUF (ตำแหน่ง 99H) ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิพที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ การเข้าถึงข้อมูลในรีจิสเตอร์แต่ละตัว MCS-51 จะทราบเองว่าผู้ใช้ต้องการติดต่อกับรีจิสเตอร์ตัวใดโดยตรวจสอบจากรหัสคำสั่ง ทั้งนี้เพราะในการเขียนข้อมูลไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ SBUF หมายถึง การโหลดข้อมูลไปที่รีจิสเตอร์สำหรับส่งข้อมูลเพื่อส่งข้อมูลออกไปภายนอก ส่วนการอ่านข้อมูลจากรีจิสเตอร์ใช้งานเฉพาะ SBUF จะหมายถึงนำค่าที่รับมาได้จากภายนอกที่เก็บไว้ในรีจิสเตอร์สำหรับรับข้อมูลมาใช้งาน

5.2 โหมดการทำงาน

การใช้งานพอร์ตสื่อสารแบบอนุกรมใน MCS-51 มีความสะดวกและคล่องสูง ทั้งนี้เนื่องจากผู้ใช้สามารถกำหนดการทำงานที่แตกต่างกันได้ถึง 4 ประเภท โดยสามารถกำหนดได้จากค่าของบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON การใช้งานที่แตกต่างกัน 4 ประเภทนี้มีจุดประสงค์เพื่อความคล่องตัวในการรับหรือส่งข้อมูลแบบอนุกรมแต่ละประเภทดังนี้

โหมด 0 การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 0 ขา RXD จะใช้สำหรับรับและส่งข้อมูล ส่วนขา TXD มีไว้เพื่อใช้สร้างสัญญาณ Shift Clock เพื่อกำหนดจังหวะในการรับและส่งข้อมูล (ข้อมูลจะถูกรับหรือส่งตามจังหวะของ Shift Clock) ในโหมดนี้การรับส่งข้อมูลจะเป็นแบบ 8 บิต (บิตข้อมูล 8 บิต) โดยเริ่มรับและส่งบิตต่ำสุดก่อน (LSB First) อัตราในการรับส่งข้อมูลในการทำงานโหมด 0 ถูกกำหนดไว้ที่ ½ ของความถี่ออสซิลเลเตอร์ที่ใช้ การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 0 จะไม่มีบิตเริ่มต้นของข้อมูล (Start Bit) และบิตสิ้นสุดของข้อมูล (Stop Bit) เพราะจังหวะในการรับและส่งข้อมูลถูกกำหนดจากสัญญาณ Shift Clock แล้ว

โหมด 1 การทำงานจะมีการรับและส่งข้อมูลครั้งละ 10 บิต ข้อมูลจะถูกส่งไปภายนอกผ่านทางขา TXD และรับข้อมูลเข้ามาทางขา RXD ข้อมูลทั้ง 10 บิตประกอบด้วย บิตเริ่มต้นของข้อมูล 1 บิต (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) และบิตสิ้นสุดของข้อมูลอีก 1 บิต (มีค่าเป็น 1 เสมอ) ในขณะทำการรับข้อมูล ค่าในบิตสิ้นสุดของข้อมูลที่รับได้จะไปอยู่ในบิต RB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON อัตราเร็วในการรับหรือส่งข้อมูลของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดนี้สามารถเปลี่ยนแปลงได้

โหมด 2 การทำงานโหมดนี้ จะมีการรับและส่งข้อมูลครั้งละ 11 บิต ข้อมูลจะถูกส่งออกภายนอกผ่านทางขา TXD และรับเข้ามาผ่านทางขา RXD ข้อมูลที่รับและส่งทั้ง 11 บิต ประกอบด้วยบิตเริ่มต้นของข้อมูล 1 บิต (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) ตามด้วยบิตที่ 9 (ต่อจากข้อมูลบิตสุดท้าย) ซึ่งเป็นบิตที่สามารถกำหนดให้มีค่าเป็นศูนย์หรือหนึ่งได้ (Programmable 9th Data Bit) และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล (มีค่าเป็น 1 เสมอ) ในขณะที่ทำการส่งข้อมูล บิตที่ 9 จะได้จากค่าในบิต TB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON บิตนี้สามารถถูกกำหนดให้มีค่าเป็น 0 หรือ 1 อย่างไรก็ได้ ส่วนใหญ่ในการใช้งานจริงมักจะใช้บิตนี้สำหรับตรวจสอบความถูกต้องของข้อมูลที่รับหรือส่ง (Parity Bit) โดยจะนำบิต P (Parity) ในรีจิสเตอร์ PSW ไปไว้ในบิต TB8 ส่วนในขณะรับข้อมูลบิตที่ 9 จะไปปรากฏอยู่ที่ RB8 ของรีจิสเตอร์ SCON โดยไม่สนใจบิตสิ้นสุดของข้อมูล ค่าอัตราเร็วในการรับหรือส่งข้อมูลโหมดนี้ถูกกำหนดไว้ที่ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้

โหมด 3 ในการทำงานในโหมดนี้ ข้อมูลจำนวน 11 บิตถูกส่งผ่านทางขา TXD และถูกรับเข้ามาทางขา RXD ข้อมูลทั้ง 11 บิตประกอบด้วยบิตเริ่มต้นของข้อมูล 1 บิต (เป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) ตามด้วยบิตที่ 9 ซึ่งเป็นบิตที่สามารถกำหนดค่าได้เหมือนในโหมด

2 และสุดท้ายคือบิตสิ้นสุดของข้อมูล (เป็น 1 เสมอ) อัตราเร็วในการรับหรือส่งข้อมูลสามารถเปลี่ยนแปลงได้ ดังนั้นจะเห็นว่ารูปแบบการรับส่งข้อมูลในโหมด 3 จะเหมือนกับโหมด 2 ทุกอย่าง แต่ในโหมดนี้สามารถกำหนดค่าอัตราเร็วในการรับหรือส่งข้อมูลได้ตามความต้องการของผู้ใช้

การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมทั้ง 4 โหมดที่กล่าวมานี้ การส่งข้อมูลจะเริ่มทันที เมื่อมีคำสั่งใดๆที่ใช้รีจิสเตอร์ใช้งานเฉพาะ SBUF เป็นรีจิสเตอร์ปลายทาง (Destination Register) เช่น MOV SBUF, A

ส่วนในการรับข้อมูลจะเริ่มต้นโดยมีเงื่อนไข ดังนี้

- ในโหมด 0 เริ่มเมื่อค่าในบิต RI = 1 และ REN = 1
- ในโหมดอื่นๆ การรับข้อมูลเริ่มเมื่อ MCS-51 ได้รับบิตเริ่มต้นของข้อมูลเข้ามา โดยที่บิต REN ในขณะนั้นต้องมีค่าเป็น 1

5.3 อัตราเร็วในการรับและส่งข้อมูล (Baud Rate)

หมายความถึงอัตราเร็วในการรับและส่งข้อมูล โดยในMCS-51 ค่าอัตราเร็วในการรับและส่งข้อมูลจะมีค่าเท่าใดขึ้นอยู่กับการทำงานในแต่ละโหมดของพอร์ตสื่อสารข้อมูลแบบอนุกรม ดังนี้

อัตราเร็วในการรับและส่งข้อมูลโหมด 0 = ความถี่ออสซิลเลเตอร์ / 12

หากใช้คริสตอลความถี่ 12 เมกะเฮิรตซ์ อัตราเร็วในการรับและส่งข้อมูลในโหมด 0 จะมีค่าสูงถึง 1 เมกะเฮิรตซ์

ในโหมด 2 อัตราเร็วในการรับและส่งข้อมูลขึ้นอยู่กับค่าของบิต SMOD ที่อยู่ในรีจิสเตอร์ใช้งานเฉพาะ PCON โดย

บิต SMOD = 0 อัตราเร็วในการรับและส่งข้อมูลเป็น 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้

บิต SMOD = 1 อัตราเร็วในการรับและส่งข้อมูลเป็น 1/32 ของความถี่ออสซิลเลเตอร์ที่ใช้

หลังจากการรีเซต MCS-51 ค่าในบิต SMOD จะเป็น 0 เสมอ และเราสามารถเขียนสูตรสำหรับคำนวณอัตราเร็วในการรับและส่งข้อมูลได้ ดังนี้

อัตราเร็วในการรับและส่งข้อมูลโหมด 2 = $[2^{SMOD}$* (ความถี่ออสซิลเลเตอร์)] / 64

หากใช้คริสตอลความถี่ 12 เมกะเฮิรตซ์ อัตราเร็วในการรับและส่งข้อมูลสูงสุดในการทำงานโหมดนี้คือ 375 กิโลเฮิรตซ์

อัตราเร็วในการรับและส่งข้อมูลในโหมด 1 และ 3 จะถูกกำหนดโดยอัตราการเกิดโอเวอร์โฟวล์ (Overflow) ของไทม์เมอร์ 1 (Timer 1 Overflow Rate)

เมื่อใช้ไทม์เมอร์ 1 เป็นตัวกำหนดอัตราเร็วในการรับและส่งข้อมูล สำหรับการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 1 และ 3 อัตราเร็วในการรับและส่งข้อมูลที่ได้จะถูกกำหนดด้วยอัตราการเกิดโอเวอร์โฟวล์ ของไทม์เมอร์ 1 และขึ้นอยู่กับบิต SMOD ในรีจิสเตอร์ PCON ซึ่งเขียนเป็นสมการที่ใช้คำนวณหาอัตราเร็วในการรับและส่งข้อมูลได้ ดังนี้

อัตราเร็วในการรับและส่งโหมด1และ3 = $[2^{SMOD}$*(อัตราการเกิดโอเวอร์โฟวล์ของไทม์เมอร์ 1)] / 32

เนื่องจากเมื่อเกิดโอเวอร์โฟวล์ในไทม์เมอร์ตัวใด จะทำให้เกิดการอินเตอร์รัปต์เพื่อบอกให้ซีพียูทราบ ดังนั้นเมื่อนำไทม์เมอร์ 1 มาเป็นตัวกำหนดอัตราเร็วในการรับและส่งข้อมูล จึงควรห้ามการเกิดอินเตอร์รัปต์ขึ้นในระหว่างการรับหรือส่งข้อมูล และเนื่องจากตัวไทม์เมอร์ 1 เองยังสามารถถูกกำหนดให้ทำงานเป็นไทม์เมอร์หรือเคาน์เตอร์อย่างใดอย่างหนึ่ง ซึ่งมีโหมดการทำงานย่อยไปอีก 4โหมด ดังนั้นในการใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมจึงต้องทำความเข้าใจในเรื่องไทม์เมอร์และเคาน์เตอร์

การใช้งานพอร์ตสื่อสารข้อมูลแบบอนุกรมที่พบบ่อยที่สุดนั้นไทม์เมอร์ 1 จะถูกกำหนดให้ทำงานเป็นไทม์เมอร์ในโหมด 2 (Auto-Reload) ในกรณีนี้อัตราเร็วในการรับและส่งข้อมูล จะถูกกำหนดโดยสมการ

อัตราเร็วในการรับและส่งข้อมูลโหมด 1 และ 3 = (2^{SMOD}*ความถี่ออสซิลเลเตอร์ที่ใช้) /

[32*12*(256-TH1)]

ดังนั้น ค่าที่ต้องโหลดไปไว้ยังรีจิสเตอร์ TH1 เพื่อให้ได้อัตราเร็วในการรับและส่งข้อมูลจะสามารถคำนวณได้ ดังนี้

TH1 = 256 – [(2^{SMOD}*ความถี่ออสซิลเลเตอร์) / (384*อัตราเร็วในการรับและส่งข้อมูล)]

อาจสามารถที่จะสร้างอัตราเร็วในการรับและส่งข้อมูลต่ำๆด้วยไทม์เมอร์ 1 ได้โดยปล่อยให้ไทม์เมอร์ 1 อินเตอร์รัปต์ซีพียูได้ และกำหนดการทำงานให้เป็นไทม์เมอร์ขนาด 16 บิต (โหมด 1) และใช้ไทม์เมอร์ 1 อินเตอร์รัปต์ซีพียูเพื่อโหลดค่าใหม่เองด้วยซอฟต์แวร์ขณะเกิดโอเวอร์โฟวล์ เนื่องจากในการทำงานโหมด 1 ของไทม์เมอร์1 ไม่สามารถโหลดค่าใหม่เองด้วยฮาร์ดแวร์ได้ (ไม่สามารถทำงานแบบออโต้รีโหลดได้)

ตารางที่ 5.1 ค่าที่ต้องนำไปไว้ในรีจิสเตอร์ของไทม์เมอร์ 1 เมื่อใช้อัตราเร็วในการรับและส่งข้อมูล มาตรฐานต่างๆ

baud rate	ความถี่ของคริสตอล	บิต SMOD	ไทม์เมอร์ 1		
			C/T	โหมด	ค่าที่ใช้โหลด
Mode 0 Max 1 MHZ	12 MHZ	X	X	X	X
Mode 2 Max 375K	12 MHZ	1	X	X	X
Modes 1,3 62.5K	12 MHZ	1	0	2	FFH
19.2K	11.059 MHZ	1	0	2	FDH
9.6K	11.059 MHZ	0	0	2	FDH
4.8K	11.059 MHZ	0	0	2	FAH
2.4K	11.059 MHZ	0	0	2	F4H
1.2K	11.059 MHZ	0	0	2	E8H
137.5K	11.986 MHZ	0	0	2	1DH
110K	6 MHZ	0	0	2	72H
110K	12 MHZ	0	0	1	FEEBH

จากตาราง 5.1 จะเห็นว่าในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0 จะมีความเร็วในการส่งมากที่สุดเมื่อเปรียบกับโหมดอื่นที่ความถี่คริสตอลค่าเดียวกัน และจะเห็นว่าหากเลือกใช้คริสตอลความถี่ 11.059 เมกะเฮิรตซ์ จะสามารถตั้งอัตราเร็วในการรับและส่งข้อมูลใน

โหมด 1 และ 3 เป็นค่ามาตรฐานที่ใช้กันทั่วไปได้ เช่น 1200,2400,4800,9600,19200 จึงเป็นเหตุผลสำคัญที่ในระบบควบคุมส่วนใหญ่เลือกใช้คริสตอลความถี่ 11.059 เมกะเฮิรตซ์มากกว่า 12 เมกะเฮิรตซ์

ในตาราง นอกจากจะแสดงอัตราเร็วในการรับและส่งข้อมูลค่าต่างๆ เปรียบเทียบให้เห็นแล้ว ตารางนี้ยังแสดงค่าที่ต้องโหลดไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ TH1 ที่อัตราเร็วในการรับและส่งข้อมูลมาตรฐานต่างๆให้ทราบอีกด้วย ผู้เขียนโปรแกรมสามารถนำค่านี้ไปใช้ได้เลย แต่หากต้องการใช้อัตราเร็วในการรับและส่งข้อมูลอื่นๆที่นอกเหนือไปจากตารางนี้ จะต้องคำนวณค่าที่จะต้องโหลดให้รีจิสเตอร์ใช้งานเฉพาะ TH1 จากสมการที่แสดงไปแล้ว

บทที่ 6
โครงสร้างของ MCS-51

ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล MCS-51 นี้ผลิตโดยบริษัทอินเทลมีอยู่ด้วยกันหลายเบอร์ซึ่งมีรายละเอียดดังตาราง 6.1

ตาราง 6.1 คุณสมบัติไมโครคอนโทรลเลอร์ตระกูล MCS-51

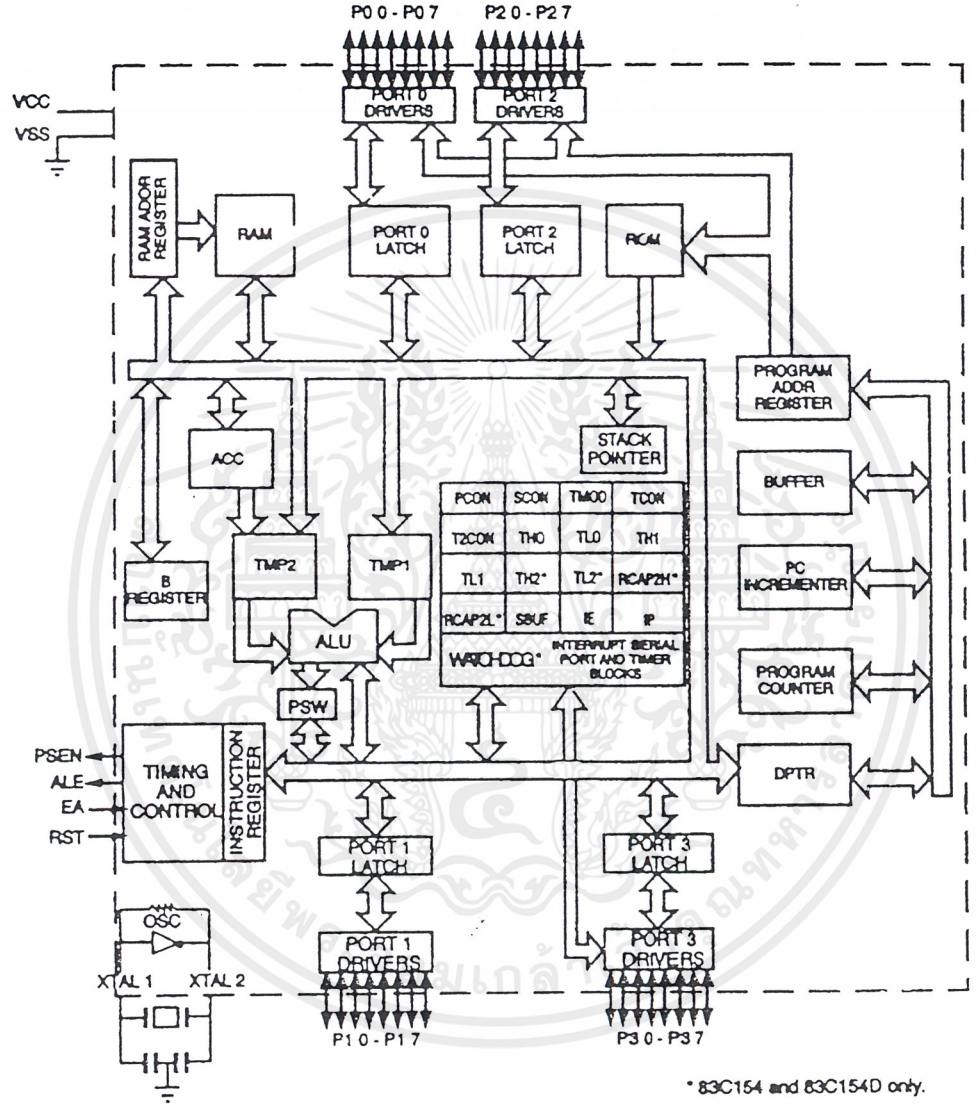
Device	ROM Version	EPROM Version	ROM Bytes	RAM Bytes	8-Bit I/O Ports	16-Bit Timer/ Counters	Programmable Counter Array (PCA)	UART	Serial Expansion Port (SEP)	Global Serial Channel (GSC)	DMA Channels	A/D Channels	Interrupt Sources/ Vectors	Power Down and Idle Modes
8051	8031	—	4K	128	4	2		✓					6/5	
8051AH	8031AH	8751H 8751BH	4K	128	4	2		✓					6/5	
8052AH	8032AH	8752BH	8K	256	4	3		✓					8/6	
80C51BH	80C31BH	87C51	4K	128	4	2		✓					6/5	✓
80C52	80C32	—	8K	256	4	3		✓					8/6	✓
83C51FA	80C51FA	87C51FA	8K	256	4	3	✓	✓					14/7	✓
83C51FB	80C51FA	87C51FB	16K	256	4	3	✓	✓					14/7	✓
83C152JA	80C152JA	—	3K	256	5	2		✓		✓	2		19/11	✓
—	80C152JB	—	—	256	7	2		✓		✓	2		19/11	✓
83C152JC	80C152JC	—	8K	256	5	2		✓		✓	2		19/11	✓
—	80C152JD	—	—	256	7	2		✓		✓	2		19/11	✓
83C452	80C452	87C452P	8K	256	5	2		✓					9/8	✓

คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MSC-51

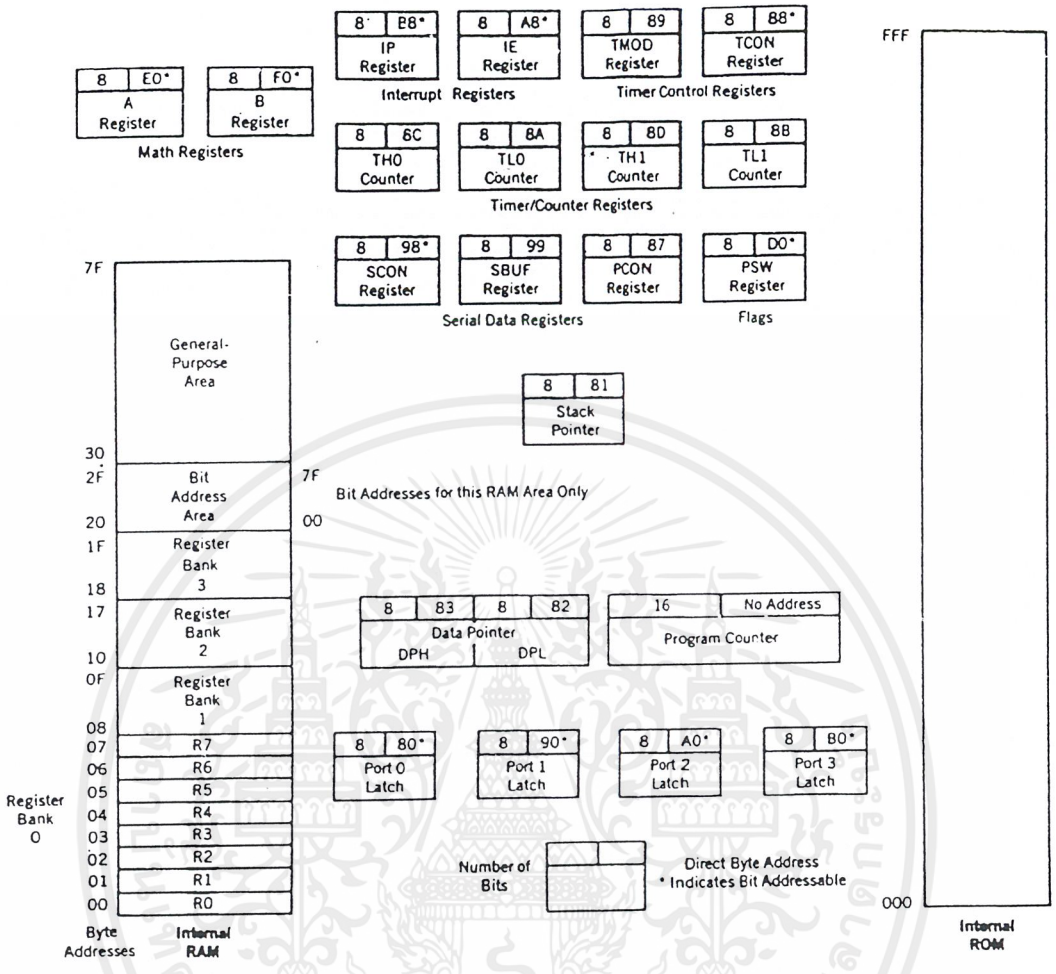
- ต้องการแหล่งจ่ายไฟ +5 V ชุดเดียว
- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์ สำหรับเบอร์ 8051 และ 8031 ,8032 ไม่มีหน่วยความจำชุดนี้ ส่วน 8052 มีหน่วยความจำถึง 8 กิโลไบต์
- มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับ 8052 มีถึง 256 ไบต์
- หน่วยความจำสำหรับโปรแกรมและเก็บข้อมูล ขนาด128 ไบต์ สำหรับ 8052 มีถึง 256 ไบต์
- หน่วยความจำสำหรับโปรแกรมและเก็บข้อมูล แยกจากกันอย่างละ 64 กิโลไบต์
- คำสั่งที่ใช้เวลาน้อยที่สุดประมาณ 1 ไมโครวินาที เมื่อทำงานที่ความถี่ 12 เมกะเฮิรตซ์
- มีไทม์เมอร์และเคาน์เตอร์ขนาด16 บิต 2 ชุด (สำหรับ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด
- รับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์
- มีพอร์ตรับส่งข้อมูลอนุกรม (UART) 2 พอร์ตทั้งรับและส่งในเวลาเดียวกันได้ (Full Duplex) เลือกรูปแบบการส่งข้อมูลได้ 4 โหมด
- มีคำสั่งในการทำ AND,ORหรือ COMPLEMENT ได้ทั้งแบบ 8 บิตและ 1 บิต

6.1 โครงสร้างภายในของ 8051

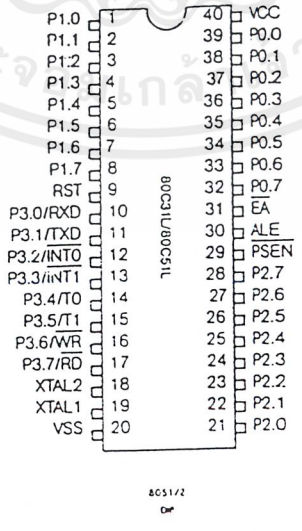
MCS-51 ใช้เทคโนโลยีในการผลิตเป็นแบบ NMOS และ CMOS เบอร์ 8032 และ 8052 จะมี ROM BASIC อยู่ภายในจึงสะดวก สำหรับผู้เขียนโปรแกรมที่จะเขียนโปรแกรมด้วยภาษาเบสิก โครงสร้างภายในสำหรับเบอร์ 8051 ดังแสดงในรูป 6.1 และ 6.2



รูป 6.1 8051 บล็อคไดอะแกรมของ MCS-51



รูป 6.2 ตำแหน่งต่างๆของรีจิสเตอร์และหน่วยความจำเพื่อใช้ประกอบในการเขียนโปรแกรม

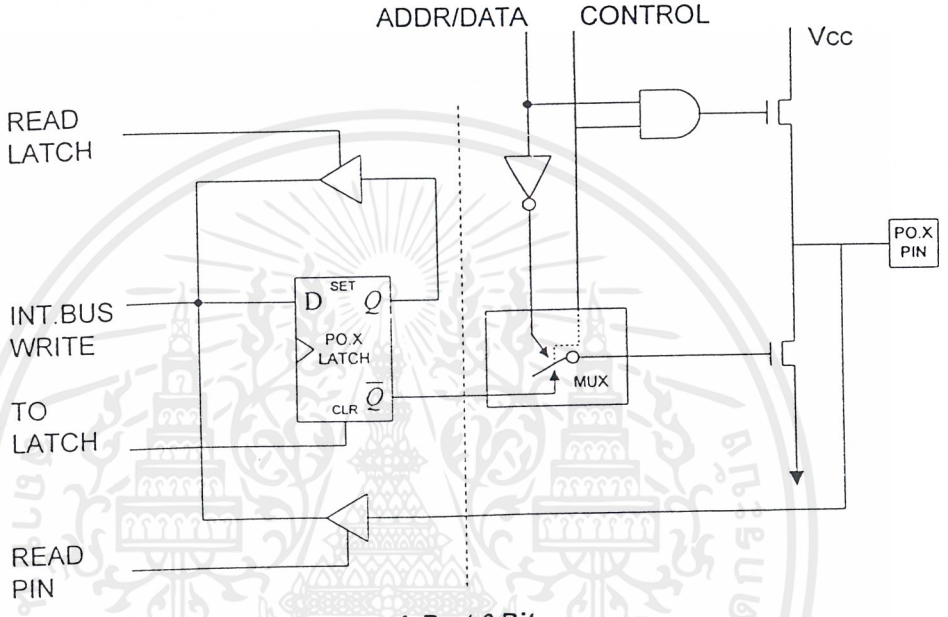


รูป 6.3 การจัดวางขาของ 8051

6.2 พอร์ตของ 8051

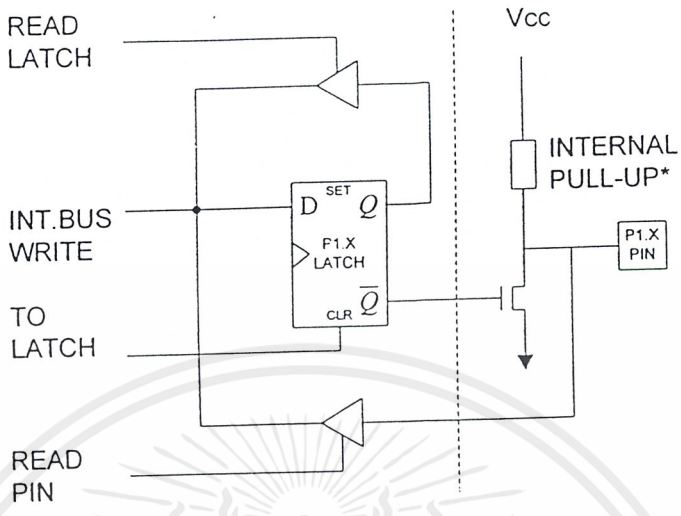
8051 เป็นไมโครคอนโทรลเลอร์ขนาด 40 ขาซึ่งมีขาต่างๆดังนี้

- Vcc (ขา40) ต่อกับ +5 V
- Vss (ขา 20) เป็นขา GND
- PORT.0 (ขา32-39) มีทั้งหมด 8 บิตคือ (P0.0-P0.7) มีโครงสร้างแบบ Open Drain Bi-Directional ดังรูป 6.4



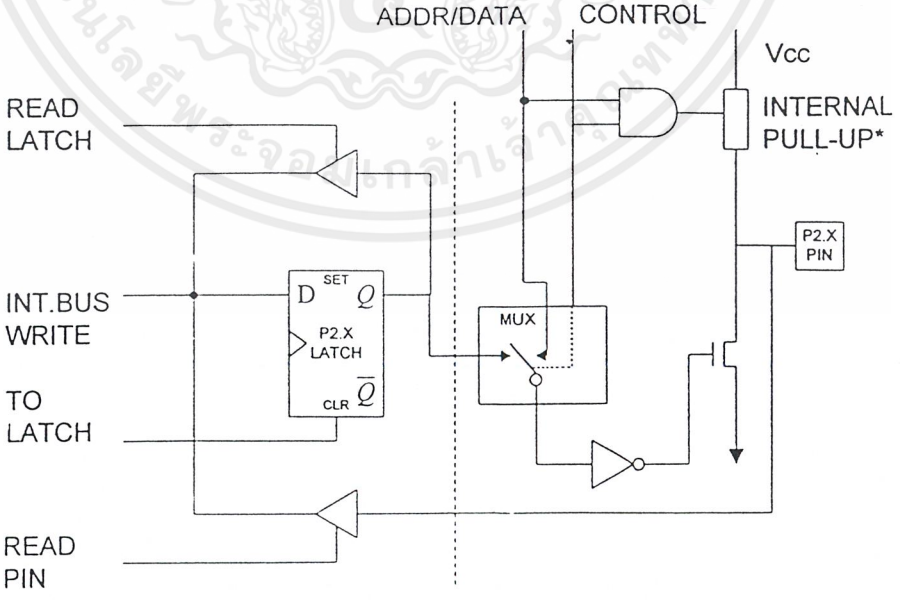
รูป 6.4 แสดงโครงสร้างของพอร์ต 0 (บิต)

- พอร์ต 0 (ขา 32- 39) มีทั้งหมด 8 บิต คือ (P0.0-0.7) ใช้งานได้ 2 หน้าที่ คือ ส่งแอดเดรสและดาต้าออกไปให้หน่วยความจำภายนอกเมื่อทำการเขียนข้อมูลลงในหน่วยความจำภายนอกควบคุมด้วยขา Control ดูรูปประกอบ และอีกหน้าที่หนึ่งก็คือเป็นพอร์ต I/O ถ้าต้องการให้ทำงานเป็นเอาท์พุทพอร์ตต้องส่งลอจิก “1” ไปยังพอร์ตนี้ จะมีผลให้ Q ของ D-FF เป็น “0 ” ทำให้ FET ตัวล่างมีสถานะ OFF สัญญาณที่ใช้อ่านอินพุทพอร์ต PIN (พอร์ต P0.X PIN) จะใช้สัญญาณ READ LACTH เมื่อถูกกระตุ้นที่ Tri-State Buffer ตัวบน
- พอร์ต 1 (ขา1-8) มีทั้งหมด 8 บิต คือ (P1.0-1.7) มีโครงสร้างคล้ายพอร์ต 0 แต่จะใช้ความต้านทานภายในพูลอัพแทน (Internal Pull Up Register) มีโครงสร้างดังรูป 6.5



รูป 6.5 แสดงโครสร้างของพอร์ต 1 (บิต)

- พอร์ต 2 (ขา 21-28) มีทั้งหมด 8 บิต คือ ขา (P2.0-2.7) มีโครงสร้างคล้ายพอร์ต 0 โดยมี FET ดังล่างตัวเดียว ส่วนด้านบนใช้ความต้านทานพูลอัพแทน พอร์ตนี้ทำงาน 2 หน้าที่ คือ สามารถใช้เป็นพอร์ตสำหรับส่งแอดเดรส 8 บิตบน (A8-A15) และเป็น I/O พอร์ต ใช้งานทั่วไป เมื่อจะใช้งานเป็นอินพุทพอร์ต ต้องส่งลอจิก "1"มาที่พอร์ตนี้ก่อน เพื่อบังคับให้ FET อยู่ในสภาวะ OFF ดังแสดงในรูป 6.6



รูป 6.6 โครงสร้างของพอร์ต 2 (บิต)

- พอร์ต 3 (ขา 10-17) มีทั้งหมด 8 บิต คือ ขา (P3.0-3.7) มีโครงสร้างคล้ายพอร์ต 1 พอร์ตนี้ทำหน้าที่ คือ เป็น I/O พอร์ต ถ้าจะให้พอร์ตนี้เป็น I/P พอร์ต ก็ให้ส่งลอจิก "1" มาที่พอร์ตนี้ก่อนและอีกหน้าที่หนึ่งก็คือ ส่งสัญญาณควบคุมออกมาและรับสัญญาณเข้าไป สัญญาณต่างๆมีดังนี้

  P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับข้อมูลแบบอนุกรม

  P3.1/RXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม

  P3.2/INT0 (External Interrupt) ใช้รับสัญญาณขัดจังหวะจากภายนอก

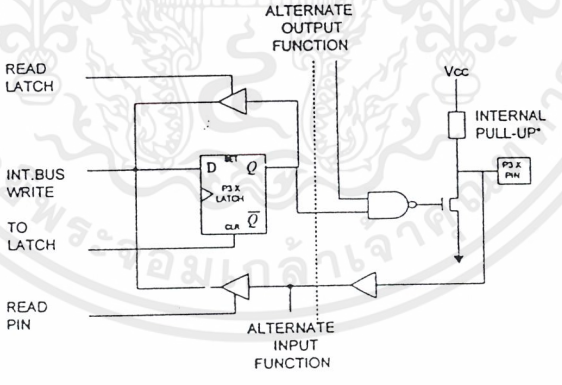
  P3.3/INT1 (External Interrupt) ใช้รับสัญญาณขัดจังหวะจากภายนอก

  P3.4/T0 (Timer/Counter 0 External Input) ขารับสัญญาณเข้าไปยังวงจร Timer/Counter 0 ที่ทำหน้าที่นับจำนวนไซเคิลของสัญญาณ T1 นี้หรือ สัญญาณนาฬิกาก็ได้

  P3.5/T1 (Timer/Counter 1 External Input) ขารับสัญญาณเข้าไปยัง Timer/Counter 1 ซึ่งมีการทำงานเหมือนกับ T0

  P3.6/WR (External Data Memory Write Strobe) ขาสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก 8051

  P3.7/RD (External Data Memory Read Strobe) ขาสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำสำหรับข้อมูลภายนอก

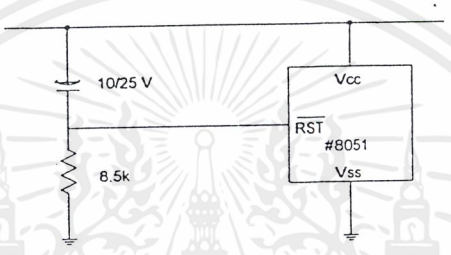


รูป 6.7 โครงสร้างของพอร์ต 3 (บิต)

- ALE (ขา 30) เป็นขาส่งสโตรบสำหรับใช้ในการแลตช์แอดเดรสไบต์ต่ำ (A0-A7) ที่ส่งออกมาจาก (พอร์ต 0) สัญญาณนี้จะแอตทีฟทุกๆ 2 ครั้ง ใน 1 แมชชีนไซเคิล (1/16 ของสัญญาณนาฬิกา)
- PSEN (ขา 29) เป็นขาที่ใช้ส่งสโตรบสำหรับอ่านข้อมูลจากหน่วยความจำสำหรับโปรแกรมภายนอก (หน่วยความจำประเภท ROM EPROM) สัญญาณนี้จะส่งออกมา 2 ครั้ง ในแต่ละแมชชีนไซเคิล แต่ถ้าเป็นการอ่านหน่วยความจำสำหรับโปรแกรมภายใน

จะไม่มีสัญญาณออกที่ขานี้

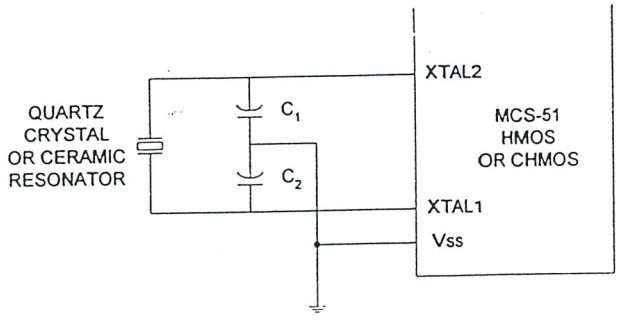
- EA (ขา 30) ถ้าป้อนลอจิก “0” เข้าที่ขานี้ ซีพียูจะอ่านค่าจากหน่วยความจำสำหรับโปรแกรมภายนอกชิพเท่านั้น แต่ถ้าถูกป้อนด้วยลอจิก “1 ” ก็จะอ่านโปรแกรมภายในชิพ
- RST (ขา 9) เป็นขารีเซ็ต ซีพียูจะรีเซ็ตได้ก็ต่อเมื่อป้อนลอจิก “1” เข้าที่ขานี้นานอย่างน้อย 2 แมชชีนไซเคิล เมื่อซีพียูถูกรีเซ็ตค่าต่างๆในรีจิสเตอร์ใดๆจะมีคำสั่งดังตาราง 6.1
- XTAL1(ขา 19) ใช้ต่อคริสตอลภายนอกโดยเป็นอินพุทเข้าสู่วงจรออสซิลเลเตอร์
- XTAL2 (ขา 18) ใช้ต่อคริสตอลภายนอกโดยเป็นเอาท์พุทของวงจรออสซิลเลเตอร์



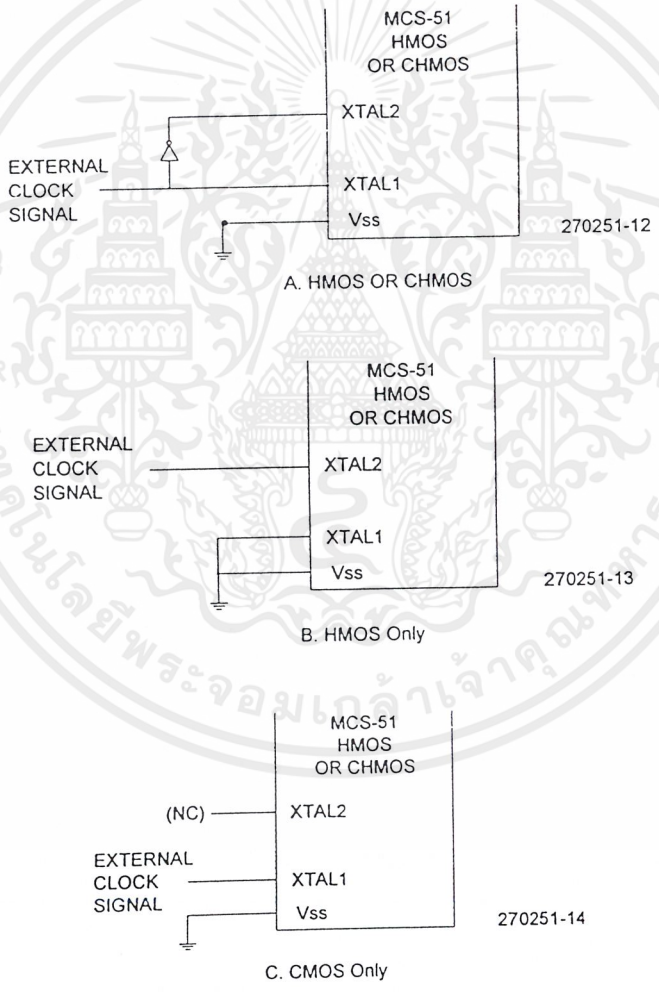
รูป 6.8 การต่อขารีเซ็ทให้กับ 8051

6.3 วงจรคล็อคของ 8051

การต่อมีด้วยกัน 2 รูปแบบ คือ แบบคล็อคภายในและคล็อคจากภายนอกมีรูปแบบการต่อดังรูป 6.9



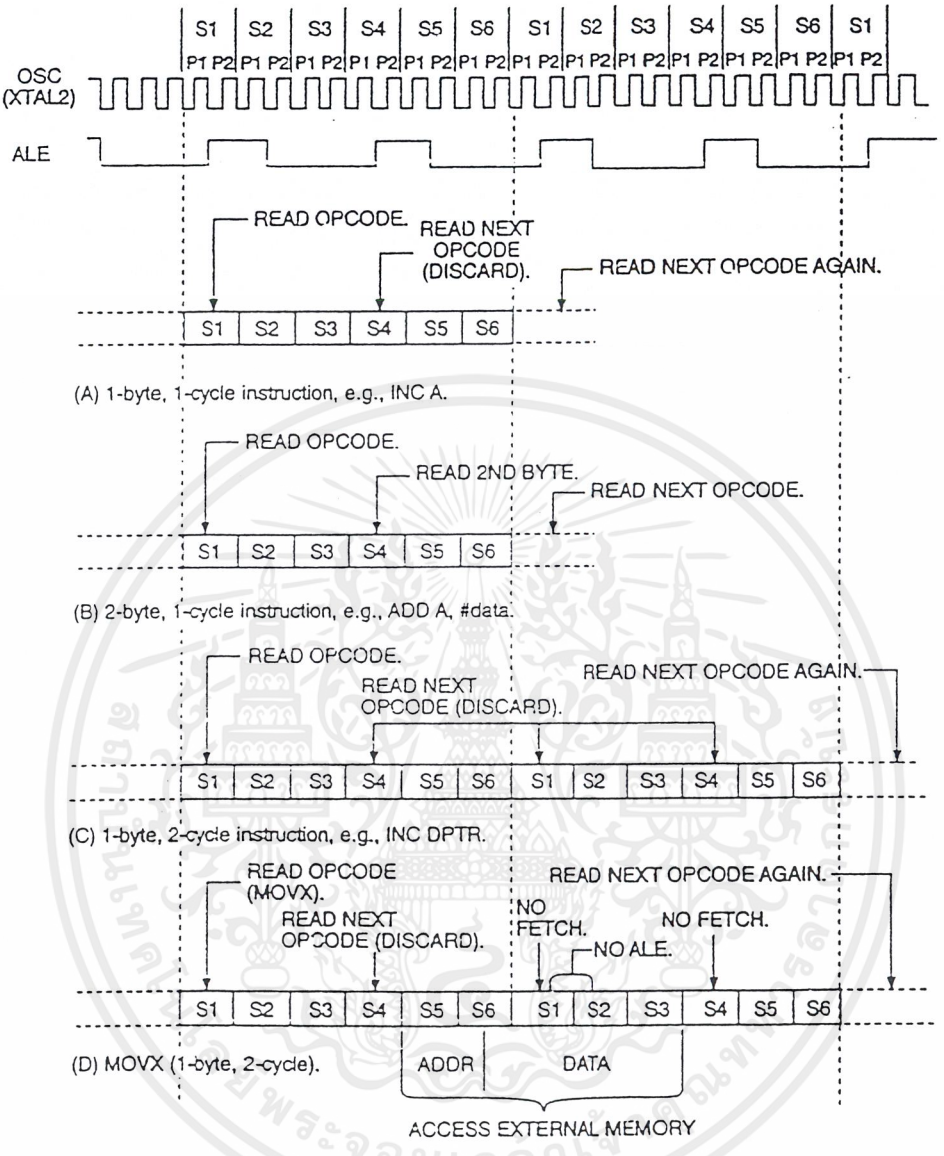
(a) Using the on-chip Oscillator



(b) Using the External Clock

รูป 6.9 วงจรสร้างคล็อคของ 8051

6.4 ผังเวลาของซีพียู (CPU Timing)



รูป 6.10 ผังเวลาการทำงานของแต่ละคำสั่ง

การทำงานใน 1 คำสั่งต่ำสุดจะใช้เวลาเพียง 1 ไมโครวินาที เช่น คำสั่ง INC A ซึ่งเป็นคำสั่ง 1 ไบต์ 1 Cycle Instruction ซึ่งจะเป็นคล็อคไปเท่ากับ 12 ลูก โดยคล็อคลูกที่ 1 และ 2 จะอยู่ในช่วง S1 เฟส 1 และ S1 เฟส 2 และคล็อคลูกที่ 12 ก็จะอยู่ในช่วง S6P2 นั่นเอง (ปกติแล้ว ซีพียูจะรันด้วยความเร็วเท่ากับ 12 เมกะเฮิรตซ์ ดังนั้น คล็อค 12 ลูกจะกินเวลาเท่ากับ (1/12)*12 = 1 ไมโครวินาที

คำว่า 1 แมชชีนไซเคิล คือช่วงการทำงานตั้งแต่ S1 จนถึง S6

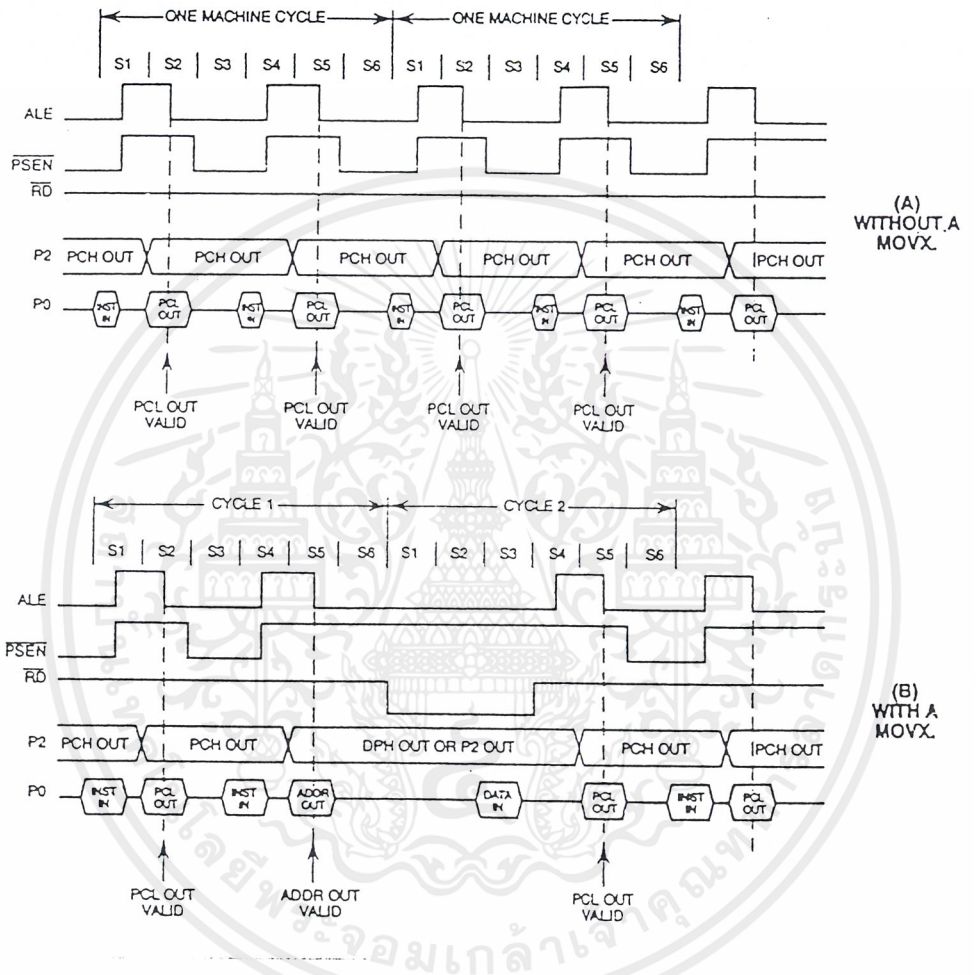
รูป 6.10A แสดงการทำงานของคำสั่ง INC A ซึ่งเป็นคำสั่ง 1 ไบต์ทำงานเสร็จภายใน 1 แมชชีนไซเคิล

รูป 6.10B แสดงการทำงานของคำสั่ง ADD A#Data ซึ่งเป็นคำสั่ง 2 ไบต์แต่ทำงานเสร็จใน 1

แมชชีนไซเคิล

รูป 6.10C แสดงการทำงานของคำสั่ง INC DPTR ซึ่งเป็นคำสั่ง 1 ไบต์ แต่ทำงานเสร็จภายใน 2 แมชชีนไซเคิล

รูป 6.10D แสดงการทำงานของคำสั่ง MOVX ซึ่งเป็นคำสั่ง 1 ไบต์ แต่ทำงานเสร็จใน 2 แมชชีนไซเคิล



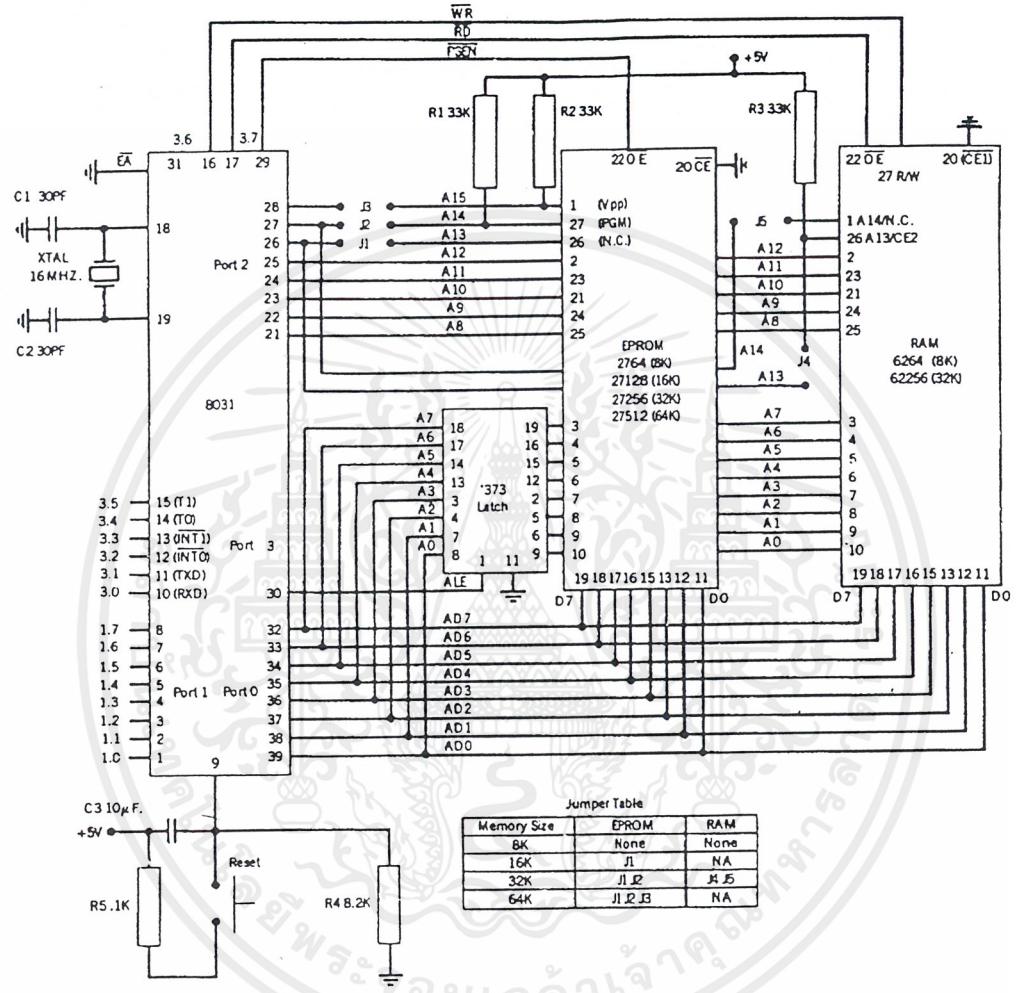
รูป 6.11 แสดงผังเวลาการติดต่อกับหน่วยความจำภายนอก

รูป 6.11A เป็นผังเวลาของสัญญาณซึ่งเกี่ยวข้องกับเฟิทซ์เมื่อส่วนของหน่วยความจำสำหรับโปรแกรมอยู่ภายนอก ดังนั้น สัญญาณที่จะนำไปใช้อ่าน OP-Code จากหน่วยความจำสำหรับโปรแกรมก็คือ *PSEN* ซึ่งจะแอกทีฟ 2 ครั้ง ใน 1 แมชชีนไซเคิล ดังนั้น สัญญาณที่ใช้อ่านข้อมูลจากหน่วยความจำสำหรับโปรแกรม จะใช้สัญญาณ *PSEN*

รูป 6.11B เป็นผังเวลาของสัญญาณที่เกี่ยวข้องกับการอ่านข้อมูลจากหน่วยความจำสำหรับเก็บข้อมูล สัญญาณ *PSEN* จะมีเพียง 1 ลูก เพราะช่วงเวลาถัดมาจะเป็นช่วงเวลาในการอ่านข้อมูลจากหน่วยความจำสำหรับข้อมูล โดยใช้สัญญาณ *RD*

อาจสรุปง่ายๆว่าการอ่านข้อมูลจากหน่วยความจำสำหรับโปรแกรมจะใช้สัญญาณ *PSEN* และการอ่านข้อมูลจากหน่วยความจำสำหรับเก็บข้อมูลจะใช้สัญญาณจาก *RD* ส่วนสัญญาณ ALE คือ สัญญาณที่ใช้ในการ Latch Address A0 – A7 นั่นเอง

6.5 การต่อหน่วยความจำ หน่วยความจำสำหรับโปรแกรมและ เก็บข้อมูลภายนอกชิพ

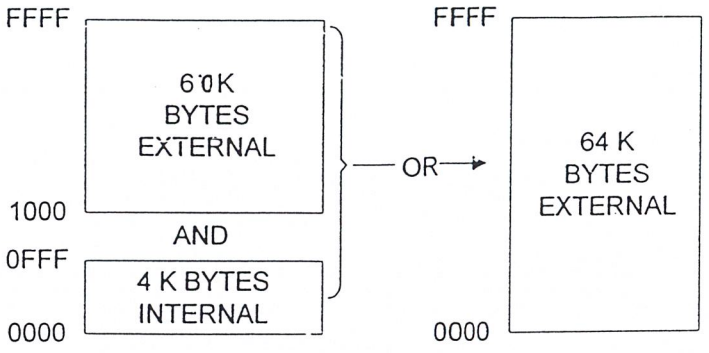


รูป 6.12 การต่อหน่วยความจำโปรแกรมและดาต้าภายนอกชิพ

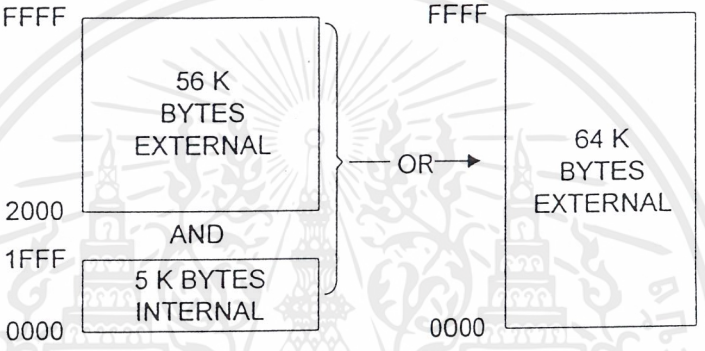
6.6 การแบ่งประเภทของหน่วยความจำ

หน่วยความจำที่ใช้กับ MCS-51 มีอยู่ด้วยกัน 2 ชนิด คือ หน่วยความจำสำหรับโปรแกรมและเก็บข้อมูล

หน่วยความจำสำหรับโปรแกรม ซึ่งเป็นหน่วยความจำที่ใช้เก็บโปรแกรมสั่งงานบรรจุอยู่ในชิพ 8051 ส่วนที่เป็นหน่วยความจำสำหรับโปรแกรม ก็คือรอม(ROM) ขนาด 4 กิโลไบต์นั่นเอง แต่ถ้าเป็นเบอร์ 8052 ก็คือรอมขนาด 8 กิโลไบต์ ดังแสดงในรูป 6.13 และ 6.14

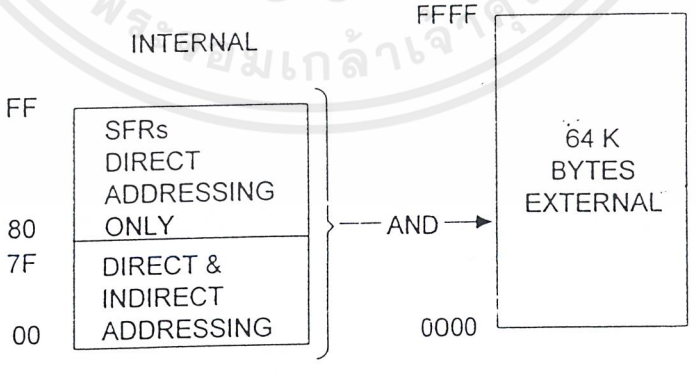


รูป 6.13 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับเบอร์ 8051

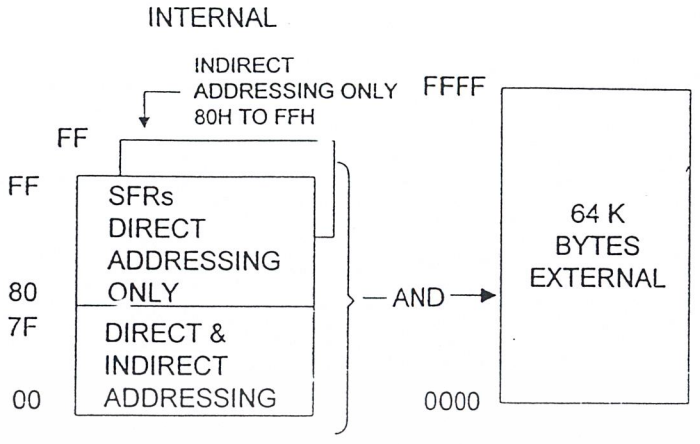


รูป 6.14 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับเบอร์ 8051

หน่วยความจำสำหรับเก็บข้อมูล เป็นหน่วยความจำที่ใช้เก็บข้อมูล หน่วยความจำนี้สามารถเขียนข้อมูลลงไปและอ่านข้อมูลออกมาได้ ซึ่งเป็นหน่วยความจำภายในชิพมีเพียง 128 ไบต์ สำหรับเบอร์ 8051 และ 256 ไบต์สำหรับเบอร์ 8052 ส่วนหน่วยความจำภายนอกชิพมี 64 กิโลไบต์ ดังแสดงในรูป 6.15 และ 6.16



รูป 6.15 ผังหน่วยความจำสำหรับโปรแกรม เบอร์ 8051



รูป 6.16 ผังหน่วยความจำสำหรับเก็บโปรแกรม เบอร์ 8052

บางครั้งอาจจะสงสัยว่าตำแหน่งของหน่วยความจำสำหรับโปรแกรมและข้อมูลมีตำแหน่งที่ซ้อนกัน ซีพียูจะรู้ได้อย่างไรว่าติดต่อกับหน่วยความจำที่เป็นโปรแกรมและดาต้า บริษัทอินเทลได้ออกแบบแยกคำสั่งออกเป็น 3 ส่วน คือ

MOV ใช้ติดต่อกับ RAM ภายใน

MOVC ใช้ติดต่อ หน่วยความจำสำหรับเก็บโปรแกรม

MOVX ใช้ติดต่อ หน่วยความจำสำหรับเก็บข้อมูลภายนอกชิพ โดยระบุตำแหน่งผ่านรีจิสเตอร์ DPTR

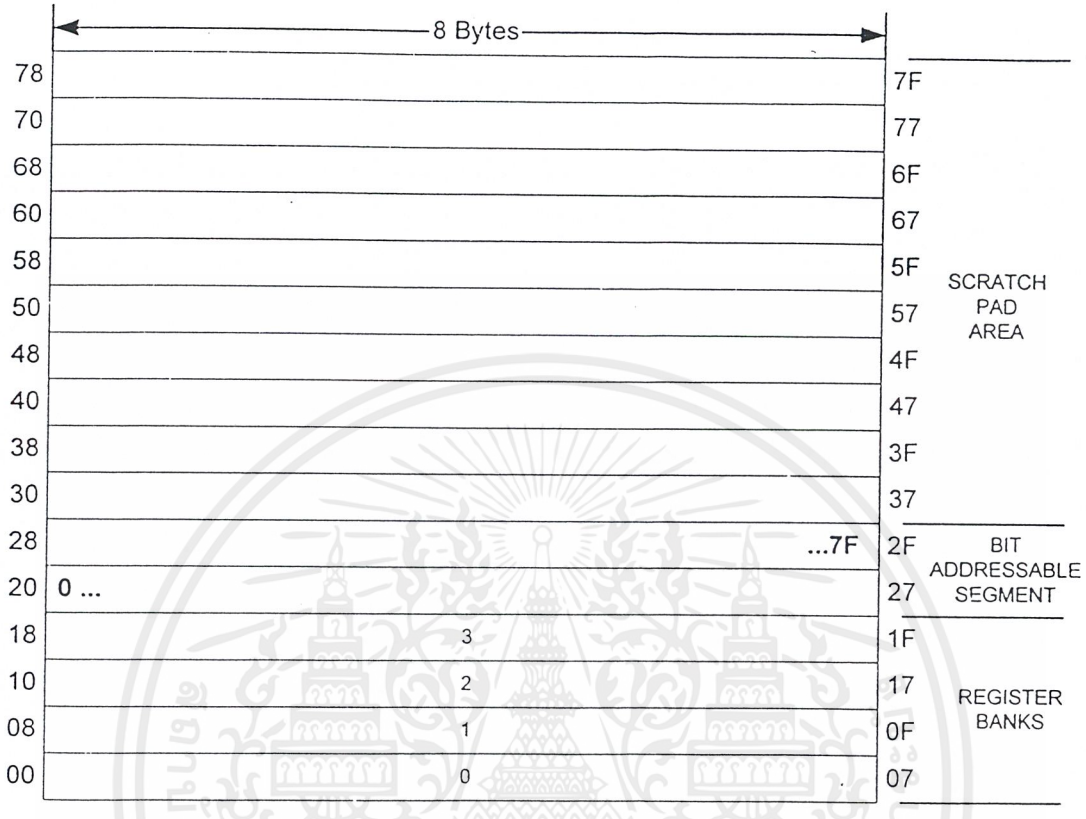
ชิพเบอร์ 8052 จะมีพื้นที่บริเวณ 80h-FFh ซึ่งถ้าจะเขียนอ่านข้อมูล ณ บริเวณนี้จะเข้าถึงข้อมูลโดยทางอ้อมเท่านั้น

พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยทางอ้อมเท่านั้น (Indirect Address Area)

พื้นที่หน่วยความจำบริเวณ(80h – FFh) เป็นพื้นที่ซ้อนกันอยู่อย่างละ 128 ไบต์ โดยส่วนแรกจะเป็น SFR และพื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยทางอ้อมเท่านั้น ดังนั้น ผู้เขียนโปรแกรมถ้าจะติดต่อกับ SFR จะต้องใช้คำสั่งแบบเข้าถึงข้อมูลโดยตรงเท่านั้น (Direct Address Area) ส่วนพื้นที่อีกส่วนหนึ่งจะเข้าถึงข้อมูลแบบทางอ้อมเท่านั้น (Indirect Address Only)

พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยตรงและทางอ้อม (Direct and Indirect Address Area)

พื้นที่ 128 ไบต์ล่างสุดจะแบ่งเป็น 3 ส่วนดังรูป 6.17



รูป 6.17 128 ไบต์ของแรมที่เข้าถึงข้อมูลแบบทางตรงและทางอ้อม

1.รีจิสเตอร์แบงค์ (Register Banks 0-3)

ตั้งแต่ตำแหน่ง (00 – 1Fh) จะเป็นส่วนของรีจิสเตอร์แบงค์ (0-3) โดยแบ่งเป็นแบงค์ละ 8 ไบต์ รวมแล้วได้ 32 ไบต์ ถ้าซีพียูทำงานอยู่ที่แบงค์ 3 เมื่อถูกรีเซ็ท ก็จะกลับมาทำงานที่แบงค์ 0 เสมอ และสแตคพอยเตอร์(SP) จะมาเริ่มต้นที่ตำแหน่ง 07h ทันที

2.บริเวณหน่วยความจำที่ใช้คำสั่งเขียน/อ่านเกี่ยวกับบิตได้ (Bit Address Area)

พื้นที่ตั้งแต่แอดเดรส (20h – 7Fh) จำนวน 16 ไบต์หรือถ้านับเป็นบิตจะได้เท่ากับ 128 บิต ซึ่งตำแหน่งบิต 00,01,02,03,04,05,06,07 ก็คือ ตำแหน่งหน่วยความจำตำแหน่ง 20h ที่บิต 0,1,2,3,4,5,6,7 ตามลำดับ ดูรูป 6.18เช่น ต้องการเซ็ทบิต D0 ของตำแหน่ง 20h ก็จะต้องเขียนคำสั่งว่า SET 00h

3.บริเวณหน่วยความจำที่ใช้งานทั่วไป (Scratch Pad Area)

พื้นที่ตั้งแต่ (30h – 7Fh) จะเขียนข้อมูลได้ทีละไบต์เท่านั้น ไม่สามารถใช้คำสั่งเกี่ยวกับบิตได้ ถ้าย้ายเนื้อที่สแตคมาบริเวณนี้โปรดระมัดระวังในการเขียนข้อมูลมาทับสแตค

RAM

Byte	(MSB)							(LSB)
7FH								
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00

รูป 6.18 แสดงตำแหน่งบิตต่างๆในตำแหน่งแอดเดรส (20h-2Fh)

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
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	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*+T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer 7 Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
+TH2	Timer/Counter 2 High Byte	0CDH
+TL2	Timer/Counter 2 Low Byte	0CCH
+RCAP2H	T/C 2 Capture Reg. High	0CBH
+RCAP2L	Byte	0CAH
*SCON	T/C 2 Capture Reg Low	98H
SBUF	Byte	99H
PCON	Serial Control	87H
	Serial Data Buffer	
	Power Control	

ตาราง 6.19 แสดงสัญลักษณ์ที่ชื่อและตำแหน่งต่างๆที่มีอยู่ใน SFR

บทที่ 7

DELPHI

Delphi เป็นเครื่องมือสำหรับสร้างแอปพลิเคชั่นสำหรับรันบนวินโดว์ 95/98/2000 ที่ผลิตโดยบริษัท Inprise (ชื่อเดิมคือ Borland) ซึ่งเป็นบริษัทที่แวดวงของนักพัฒนาแอปพลิเคชั่นรู้จัก และยอมรับในผลิตภัณฑ์เป็นอย่างดี

Delphi เป็นเครื่องมือพัฒนาแอปพลิเคชั่นแบบ Visual Programming เหมือนกับ Visual Basic,Visaul C++ ฯลฯ) ซึ่งทำให้เราสามารถเห็นผลลัพธ์การทำงานไปพร้อมๆกับการลงมือสร้างแอปพลิเคชั่น

จุดเด่นที่สำคัญมากของความเป็น Visual Programming คือช่วยลดเวลาของการสร้างแอปพลิเคชั่น โดยให้เป็นหน้าที่ของ Delphi แทน

7.1 การสร้าง CGI Application ด้วย Delphi

สิ่งที่จำเป็น

1. โปรแกรม Delphi 3 /4/5
2. โปรแกรมเว็บเซิร์ฟเวอร์

อาจจะใช้ Microsoft's Personal Web Server (PWS) ซึ่งสามารถเอามาใช้ได้ฟรีจากเว็บไซต์ให้บริการอยู่ หรืออาจจะมีโปรแกรม PWS อยู่แล้วบนเครื่อง ถ้าเครื่องเป็น Win98 Edition2 ซึ่งจะมีโปรแกรม Personal Web Server 4.0.ให้อยู่แล้ว และยังสามารถใช้ Microsoft's IIS บน Windows NT/ Windows 2000 ซึ่งจะต้องใช้งานโดยกำหนดการทำงานเป็นแบบโปรแกรมซีจีไอแทนซึ่งการทำงาน 2 แบบนั้นคือ ไอเอสเอพีไอ (ISAPI) และซีจีไอจะใช้ทรัพยากรไม่เหมือนกัน รวมทั้งผลเรื่องความเร็วก็ไม่เหมือนกันและ ต้องแน่ใจว่าทั้งทรัพยากร และความเร็วจะต้องเพียงพอที่เอาไปพัฒนากับแอปพลิเคชั่น โปรแกรมซีจีไอและยังต้องแน่ใจว่าไม่มีปัญหาเวลารันโปรแกรม นอกจากนี้ ยังสามารถใช้ Omni httpd หรืออาจจะใช้ Falcon Web Server ซึ่งเป็นโปรแกรมเว็บเซิร์ฟเวอร์ที่เขียนโดยDelphi

3. ความรู้พื้นฐานของ HTML

   HTML เป็นสคริปต์ที่ใช้แท็กแยกความแตกต่างของส่วนต่างๆออกจากเอกสาร โดยที่มีรูปแบบ <Tagname> เพื่อเปิด แท็กและต้องมีการปิดแท็ก</Tagname> บางแท็กอาจมีการกำหนดค่าต่างๆเช่น <FONT COLOR> เป็นต้น

4. ความรู้ในDelphi

   อธิบายพื้นฐานก่อนที่จะเริ่มทำ

   โปรแกรมเว็บเบราเซอร์ของไคลเอนต์จะส่งคำร้องขอข้อมูลเว็บเพจ ไปยังเครื่องเซิร์ฟเวอร์ซึ่งอาจจะเป็น ข้อมูล HTML หรือ ไฟล์ดาต้าเบส หรือทั้ง 2 อย่างก็ได้

คำร้องขอข้อมูลโดยเว็บเบราเซอร์จะไปยังเครื่องเซิร์ฟเวอร์ที่มีโปรแกรมเว็บเซิร์ฟเวอร์ โดยที่เว็บเซิร์ฟเวอร์จะทำการส่งคำร้องขอข้อมูลไปยังโปรแกรมซีจีไอ

เมื่อโปรแกรมซีจีไอแอปพลิเคชั่นได้รับคำร้องขอข้อมูล อาจจะมีค่าพารามิเตอร์ส่งมาด้วย ซึ่งจะต้องตอบรับคำต้องขอข้อมูลโดยการส่งข้อมูลตอบสนองโดยผ่านไปยังเว็บเซิร์ฟเวอร์ โดยที่ข้อมูลจะอยู่ในรูป HTML เสมอ

หลังจาก อินสตอล โปรแกรมเว็บเซิร์ฟเวอร์ลงในเครื่องแล้วจะรันโปรแกรมนี้ได้โดยใส่ URL ในโปรแกรมเว็บเบราเซอร์ของเราเพื่อผ่านเข้า โดเมนเนม (Domain Name)โดยจะใช้ชื่อของคอมพิวเตอร์ของผู้ใช้ ถ้าชื่อคอมพิวเตอร์คือ MyPC ต้องพิมพ์ http:// MyPC หรือ http:// MyPC.default.htm โปรแกรมเว็บเบราเซอร์จะคอยหาไฟล์ที่เรียกว่า default.htm (หรือ default.asp) เสมอในโฟลเดอร์รูท ของเว็บเซิร์ฟเวอร์ (ในกรณี เว็บเซิร์ฟเวอร์ 4.0 และ IISโฟล์เดอร์รูทอยู่ที่ C:\interpub\wwwroot)

รูทโฟลเดอร์จะเหมือนกับโฟลเดอร์ C:\ ของฮาร์ดไดร์ฟ รูทโฟลเดอร์จะเป็นโฟลเดอร์ย่อยๆของฮาร์ดไดร์ฟ เว็บเซิร์ฟเวอร์จะไม่เห็นโฟลเดอร์ที่สูงกว่าโฟลเดอร์รูท รูทโฟลเดอร์คือจุดเริ่มของเว็บเซิร์ฟเวอร์ และสามารถเห็นทุกโฟลเดอร์และซับโฟลเดอร์ใน ทรีของมันโฟลเดอร์ส่วนมากจะอ่านได้อย่างเดียว ซับโฟลเดอร์หนึ่งของ เว็บเซิร์ฟเวอร์คือ /Scripts (หรือ/Cgi-bin) ที่สามารถอ่าน/เขียน/execute access โดยที่ซีจีไอจะอยู่ในโฟลเดอร์นี้

และสามารถเข้าไปหน้าอื่นๆ สมมติ MyHTML.htm จากรูทเว็บเซิร์ฟเวอร์ โดยพิมพ์ htttp://MyPC/MyHTML.htm ที่ URL จะเข้าไปในสคริปต์ที่ชื่อ MyCgi.exe เพราะฉะนั้นใน URL จะเป็น http://MyPC/scripts/ MyCgi.exe ดังนั้นจึงไม่ต้องมีอะไรพิเศษ default " action" ในเว็บแอปพลิเคชั่น จะถูกรัน

ถ้าเรามี หน้าล็อกออน(หมายถึงที่ต้องการล็อกอินเข้าไปโดยมี ชื่อผู้ใช้ USERNAME กับรหัสผ่าน PASSWORD) ที่ต้องการให้แสดงจะใช้ "/logon" ถ้าจะให้ขึ้น logon page ในURL จะเป็น http://MyPC/scripts/MyCgi.exe/logon คำร้องขอข้อมูลจะถูกส่งมายัง PWS และจะรับคำร้องขอข้อมูลในหน่วยความจำและจะส่งให้ logon action

จุดสำคัญคือ งานที่จะส่งหน้า HTML กลับในรูปของ สตริงดังนั้นในการสื่อสารระหว่างไคลเอนต์และเซิร์ฟเวอร์จะแน่นอน ถ้าเข้าใจในหลักการนี้จะสามารถสร้างซีจีไอโดยใช้Delphi แอปพลิเคชั่นได้

7.2 วิธีสร้างซีจีไอแอปพลิเคชั่น

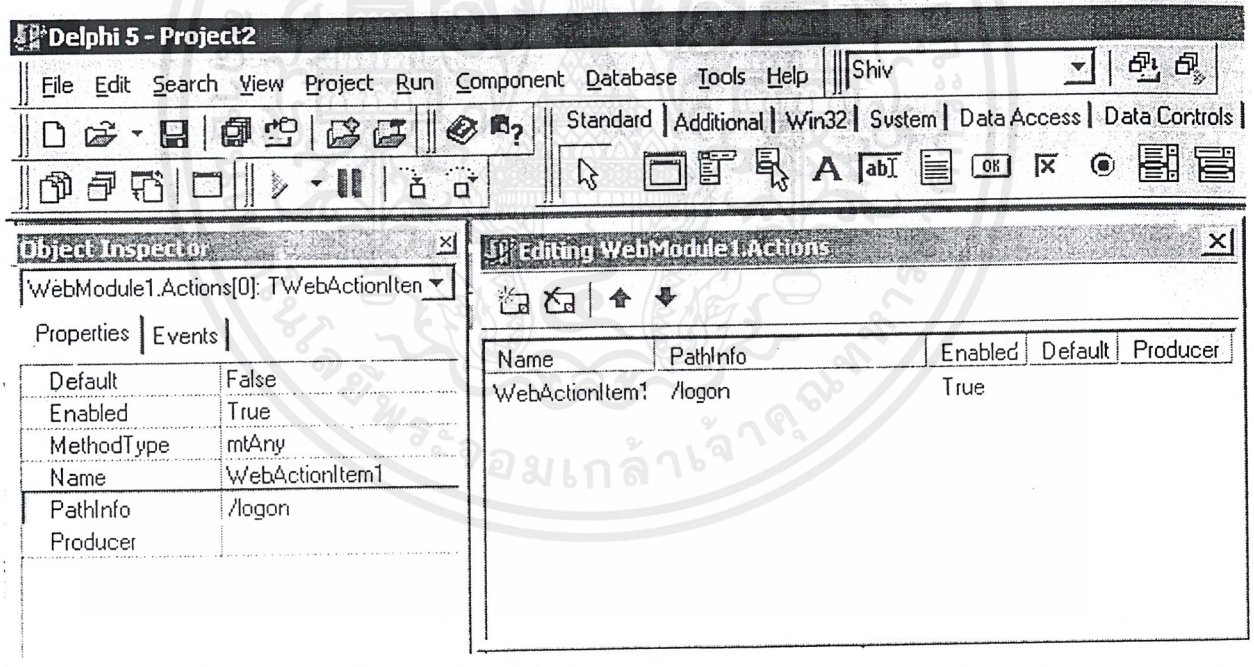
- เปิดโปรแกรม
- เลือก File – New
- จากไดอะล็อกให้เลือก เว็บเซิร์ฟเวอร์แอปพลิเคชั่นแล้วคลิก Ok

- ไดอะล็อกต่อไป ทำการเลือกซีจีไอโดยจะมีให้เลือกจากไอเอสเอพีไอ ,ซีจีไอ และ วิน-ซีจีไอ (WIN-CGI)

ตอนนี้จะเห็น New Project กับ Web Module และยูนิตของมัน

ข้อแนะนำ ควรไปที่ Project Options Dialog และเซ็ตค่าที่ " Output Directory" บน ไดเร็กทอรี่และ คอนดิชั่น เพจ ไปที่สคริปต์ โฟลเดอร์ของเว็บเซิร์ฟเวอร์ นั่นคือ C:\Interpub\Scritps เหตุผลคือ เมื่อคอมไพล์ แอปพลิเคชั่นแล้ว ไฟล์เว็บเพจจะถูกทำให้เกิดในโฟลเดอร์ที่ต้องการ

- Save Project ด้วยชื่อ MyCgi Web Module คือที่ที่จะใส่คอมโพเนนต์
- ทำการเพิ่มแอคชั่นในเว็บแอปพลิเคชั่น คลิกบนแอคชั่นพรอพเพอร์ตี้ในออพชั่น อินสเป็กเตอร์ หรือ ดับเบิ้ลคลิกบนเว็บโมดูล จะแสดงหน้าต่าง แอคชั่น อีดิตเตอร์
- เพิ่มแอคชั่นในเว็บโมดูล ถ้ามี แอคชั่น อีดิตเตอร์ แอคทีฟ อยู่ให้คลิกบน Add New ที่ แถบเครื่องมือของแอคชั่น อีดิตเตอร์
- ที่แอคชั่นที่แอดเข้าไปใหม่ให้ดูใน พร็อพเพอร์ตี้ ใน อ็อพชั่น อินสเปคเตอร์ ที่ช่อง พาร์ทอินโฟ ในกรณีนี้เรา เซ็ตในส่วนของ logon ดังนั้นให้เปลี่ยนเป็น / logon
- ต่อไปไปที่หน้าอีเวนต์ และที่หน้าอีเวนต์จะขึ้นเป็น OnAction



รูป 7.1 แสดงหน้าต่าง Editing WebActionItem1.Actions/logon

ที่การเขียนสำหรับ OnAction Event ควรจะสังเกตุจากตัวแปรที่ทำการผ่านค่า หมายถึงตัวแปรที่ทำการขอข้อมูล และ ตอบสนองทั้งคู่เป็นออบเจ็กต์และมีการ เซ็ตพรอพเพอร์ตี้และ เมดธอด เป็นของตัวเอง

" logon" Action ต้องการจะ ส่งค่าการล็อกออนกลับไปยังเว็บเบราเซอร์ ดังนั้นที่ HTML Tag ที่จะสร้างหน้า ล็อกออน จะต้องรู้ว่าแท็ก ต้องการอะไรที่จะส่งค่ากลับไปยังเบราเซอร์เพื่อแสดงหน้าโฮมเพจเราใช้ request object เพื่อหาว่า คำร้องขอข้อมูลของเราคืออะไรและ ทำการส่งค่ากลับไปยังเว็บเบราเซอร์ โดยมีวิธีเขียนดังต่อไปนี้

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
 Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
 Response.Content :=
  '<html>' +
  '<head>' +
  '<title>User Log On</title>' +
  '</head>' +
  '<body>' +
  { Remember to change the URL from MyPC to your PC's name !! }
  '<form action="http://MyPC/scripts/MyCgi.exe/logUser" method="post">' +
  '<BR/>UserID<input name="txtUserID">' +
  '<BR/>Password<input name="txtPassword">' +
  '<BR/><BR/>' +
  '<input type="submit" value="Submit">' +
  '<input type="reset" value="Reset">' +
  '</form>' +
  '</body>' +
  '</html>';
end;
```

อะไรจะเกิดขึ้นเมื่อ ปุ่มซับมิตถูกคลิกในฟอร์มนี้

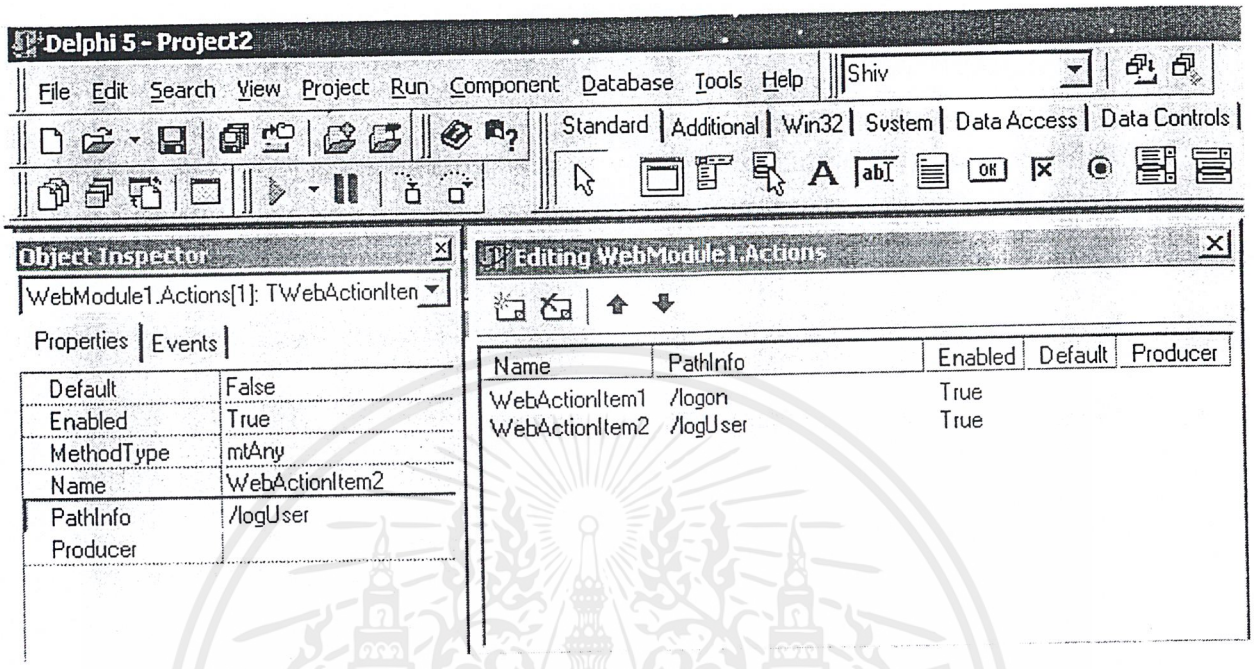
<form action=http://MyPC/scripts/MyCgi.exe/loguser method = post >

เมื่อปุ่ม ซับมิต ถูกคลิกในกรณีนี้ คำร้องขอข้อมูล จะถูกทำการเรียกไปยังซีจีไอแอปพลิเคชั่น (MyCgi.exe) ด้วย "/loguser" Action ส่งไป ดังนั้นเราต้องมีหน้า "/loguser". Action ในแอปพลิเคชั่นด้วย

ดังนั้นย้อนกลับไปดู

1. เพิ่ม New Action บนเว็บโมดูล
2. เซ็ต พาร์ดอินโฟ เป็น / logUser

3. สร้าง OnAction Event ใช้ใน Action นี้



รูป 7.2 แสดงหน้าต่าง Editing WebActionItem1.Actions/logUser

อะไรใช้ตรวจสอบว่าชื่อผู้ใช้ และ รหัสผ่านถูกต้องใช้ logUser Action สังเกตว่า คำสั่งของ< Form> คือ Post เมื่อใช้ Post เมดธอดในฟอร์ม HTML ตัวแปรของฟอร์มจะหาได้ใน Contentfields Property ใน Request Object ส่งให้เหมือนตัวแปรใน OnAction Event Contentfields property เป็นสตริง Tstring Object Method ใช้ในการดึงค่าที่ต้องการใช้ตาม Code นี้

SUserID := Request.ContentFields.values ['txtUserId']

SPassword := Request.ContentFields.values['txtPassword']

เมื่อ SUserID และ Spassword อยู่ใน local string variables ดังนั้นอีเวนต์ของ / logUser จะมีแอคชั่นดังนี้

```
procedure TWebModule1.WebModule1WebActionItem2Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  sUserID: string;
  sPassword: string;
begin
  { Extract the values for the forms' fields }
  sUserID := Request.ContentFields.Values['txtUserID'];
```

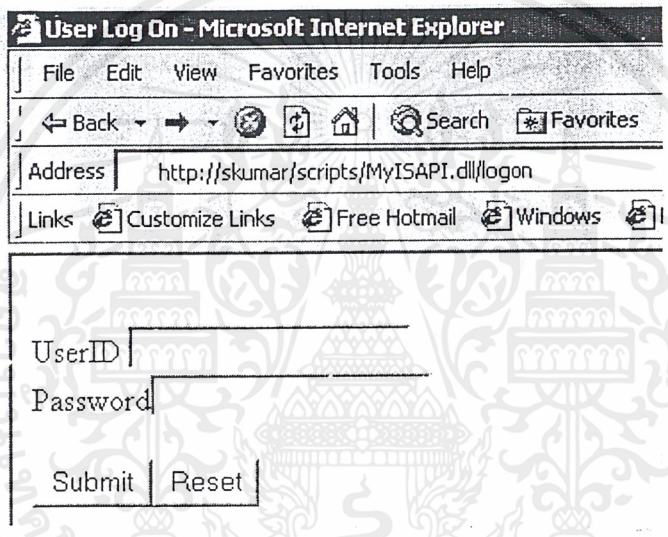
```
sPassword := Request.ContentFields.Values['txtPassword'];

{ Validate the UserID and Password fields }
if (sUserID = '') or (sPassword = '') then
begin
  Response.Content :=
    '<html>' +
    '<head>' +
    '<title>Log On Error</title>' +
    '</head>' +
    '<body>' +
    'Either the UserID or Password was blank.' +
    'Please make sure your user information is entered correctly' +
    '</body>' +
    '</html>';
end
else
begin
  { At this point you could verify the userid and password with
    data in a database or any other way. Assuming the userid
    and password are valid. You can then send back an HTML page
    indicating that to the user.
  }
  Response.Content :=
    '<html>' +
    '<head>' +
    '<title>User is Valid</title>' +
    '</head>' +
    '<body>' +
    'Welcome. Your profile has been recognized.' +
    '</body>' +
    '</html>';
```

```
 end;
end;
```

เราต้องใช้ Request Object ในกรณีนี้ไม่ใช่ Request Object Request Object ต้องใช้ Get ในการขอข้อมูล ส่วน Response Object ใช้ส่งข้อมูลกลับไปที่ผู้ขอคำร้องข้อมูล

การทดสอบ MyCgi

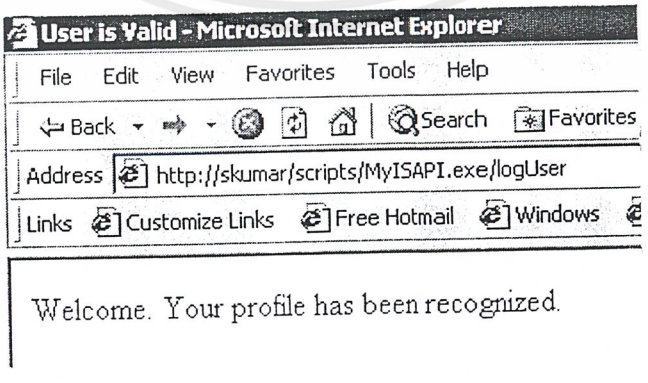
1. เตรียมคอมไพล์ และทดสอบMyCgi.exe ถ้าเราเปลี่ยน Output Directory ไปที่สคริปต์โฟลเดอร์ของ เว็บเซิร์ฟเวอร์ แล้วจะเห็น ไฟล์ MyCgi.exe ในโฟลเดอร์ที่คอมไพล์ไปแล้วครั้งหนึ่ง
2. เปิดเว็บเบราเซอร์และพิมพ์ URL ชื่อ MyPC เป็นชื่อของ PC ดังนั้น URL คือ `http://MyPC/scripts/MyCgi.exe/logon`
3 หน้าจอจะขึ้นมาดังนี้



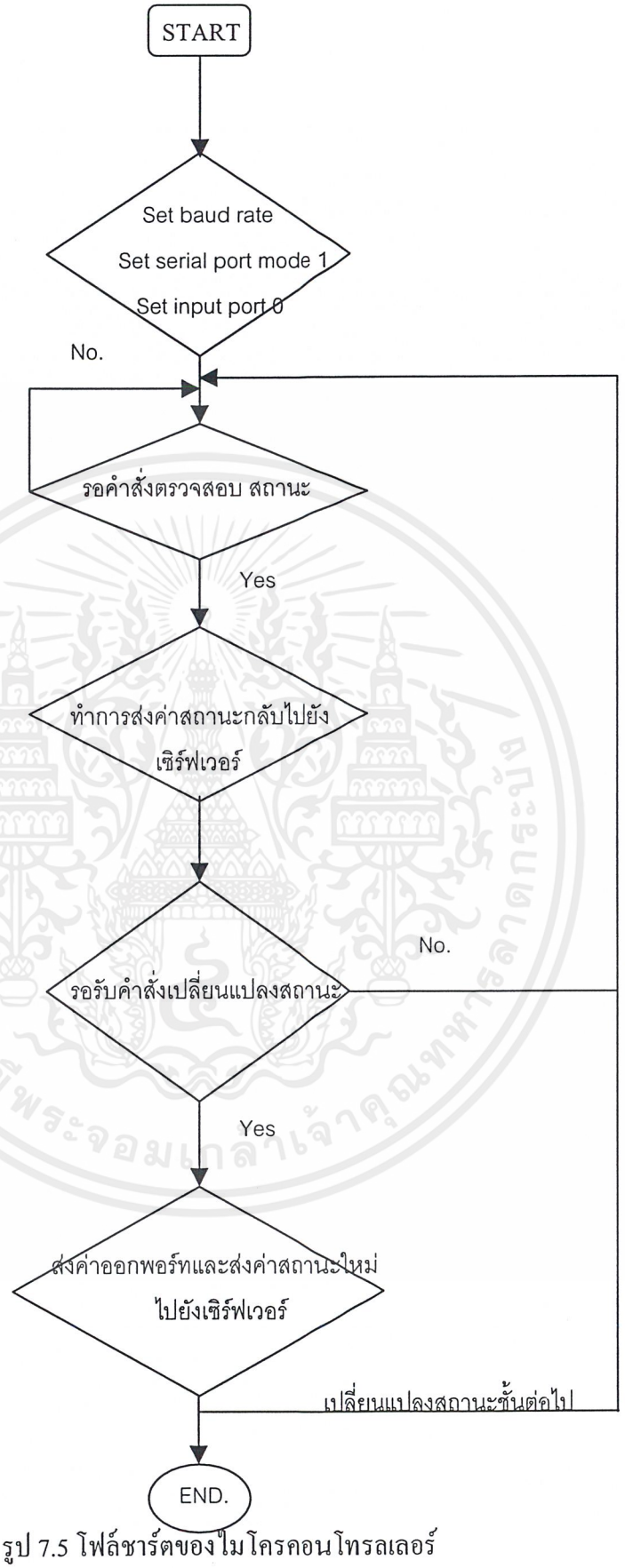
รูป 7.3 แสดงโปรแกรมซีจีไอหน้าล๊อกออน

หลังจากเติมในช่องว่างและกดปุ่มซับมิตถูกต้องแล้วที่

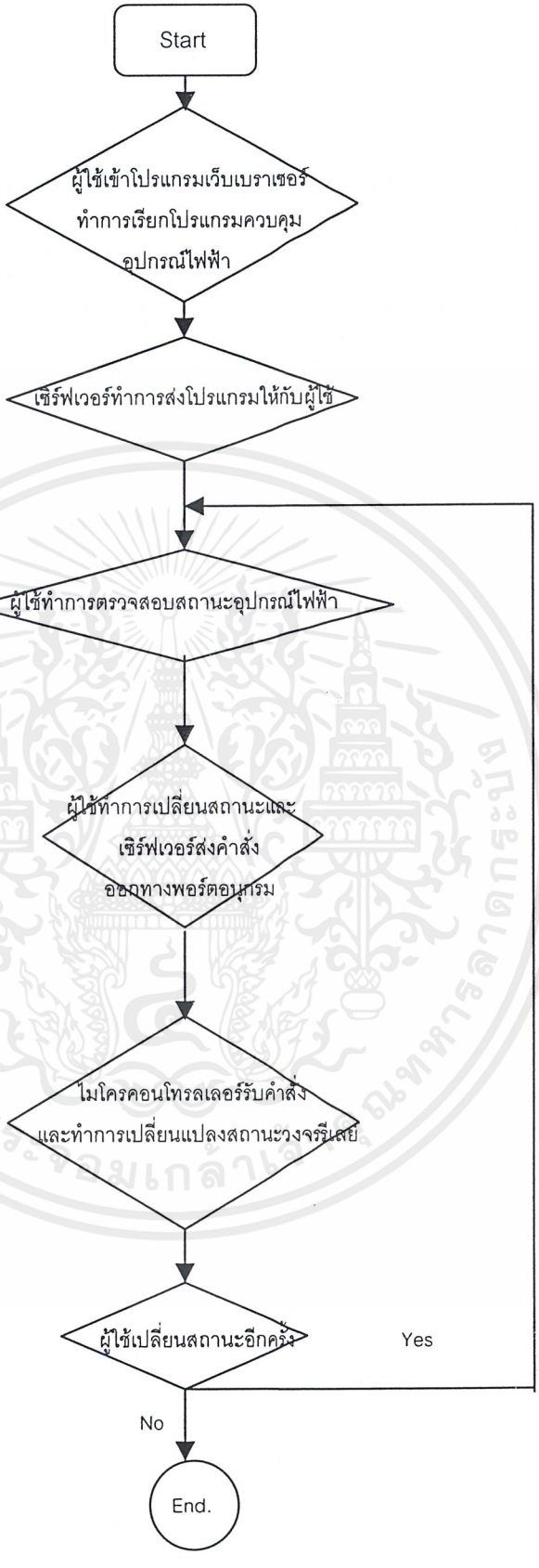
address `http://MyPC/Scripts/MyCgi.exe/logUser`



รูป 7.4 แสดงโปรแกรมซีจีไอหน้าล๊อกยูสเซอร์



รูป 7.5 โฟล์ชาร์ตของไมโครคอนโทรลเลอร์



รูป 7.6 โฟล์ชาร์ตของโปรแกรมDelphi

บทที่ 8
องค์ประกอบฮาร์ดแวร์ของระบบ

วงจรไมโครคอนโทรลเลอร์ จะทำงานเป็นวงจรรับข้อมูลอนุกรม จากเครื่องเซิร์ฟเวอร์โดยผ่านคอม 1 แล้วแปลงเป็นข้อมูลขนานเพื่อส่งต่อไปยังวงจรรีเลย์ ในโครงงานนี้เราจะใช้บอร์ด ANT-C51 เป็นวงจรไมโครคอนโทรลเลอร์

ANT-C51

ใช้ SINGLE CHIP เบอร์ 89C51 ของบริษัท ATMEL เป็นซีพียูซึ่งเป็นโครงสร้างพื้นฐานภายในของตัวชิพจะเหมือนกับเบอร์ 8751(ชิพตระกูล MCS51 ของบริษัท INTEL) ทุกประการแต่อย่างไรก็ตามผู้ใช้สามารถเลือกใช้งาน SINGLE CHIPS เบอร์ต่างๆในตระกูล MCS51 หรือ COMPATBLE ร่วมกับบอร์ด ANT-C51 ได้คือเบอร์87(C)51,87(C)52,89C51,89C52,89C55 และ DS5000(T) ซึ่งจะทำให้ได้คุณสมบัติเป็นไปตามโครงสร้างของเบอร์นั้นๆ

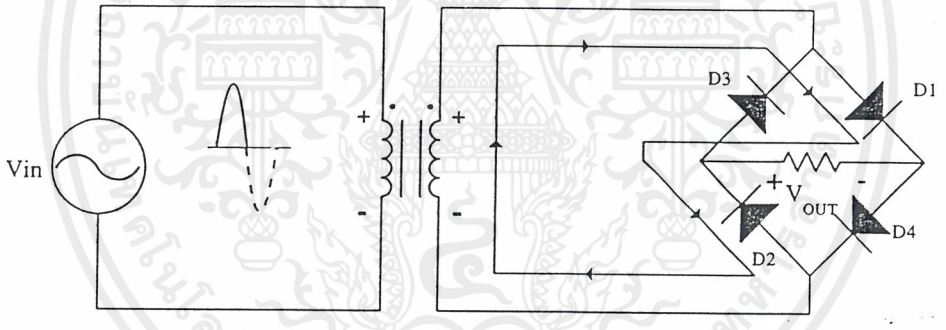
<u>คุณสมบัติของบอร์ด</u>

CPU	89C51 Uc (แต่ในโครงการนี้เราทำการเปลี่ยนชิพเป็น 89C52)
CLOCK	11.0592 เมกะเฮิรตซ์
MEMORY	4 กิโลไบต์ (MCU FLASH MEMORY)
PORT	32 BIT I/O (MCU I/O PORT)
CONNECTOR	26 PIN HEADER (8255 PORT COMPATIBLE)
	16 PIN HEADER (LCD PORT COMPATIBLE)
	3 PIN RS232 (DS275)
	2 PIN RS485 (75176)
	2 PIN 5VDC
ONBOARD	1 POWER LED
	1 RESET SWITCH (DS1232LP)
	1 2-WAYS JUMPER (DS5000 RUN/PROG)
	1 2-WAYS JUMPER (WATCH-DOG EN/DIS)
	2 TEST POINS FOR CLIP (VCC,GND)
	3 4x3.8 INCH WORKING AREA PCB
OPTION	DS275 (RS232 PORT)
	75176 (RS485 PORT)
	93C46 (64x16 SERIAL READ/WRITE EEPROM)
POWER	5 VDC CURRENT 60 มิลลิแอมแปร์ (FULL OPTION)

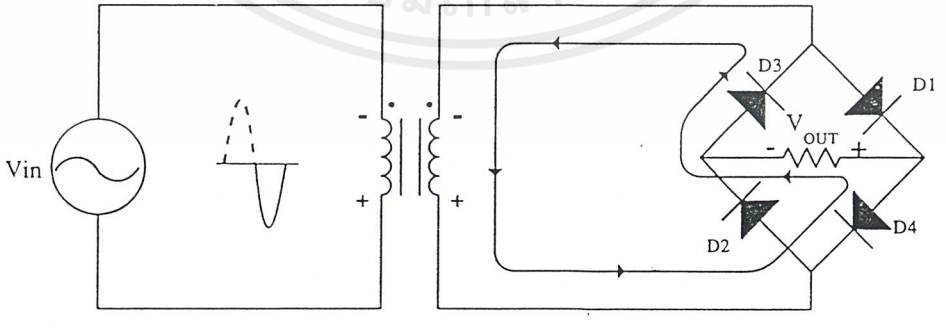
SIZE 4x5.5 นิ้ว

วงจรเรียงกระแสแบบบริดจ์

การเรียงกระแสแบบเต็มคลื่นหรือฟูลเวฟเร็กติฟายเออร์ (Full – wave Rectifier) แม้ว่าจะสามารถให้กระแสออกไปอย่างต่อเนื่องแต่หากไดโอดตัวหนึ่งตัวใดขาด วงจรนั้นจะกลายเป็นวงจรเรียงกระแสแบบครึ่งคลื่น (Half-wave Rectifier) และหากไดโอดเกิดการรั่วไหลหรือลัดวงจรกระแสไฟสลับจะไม่ถูกส่งไปยังโหลดให้เกิดเสียหายได้ ที่สำคัญระบบนี้ยังต้องใช้หม้อแปลงขนาดใหญ่เพราะมีเซ็นเตอร์แท็ป จำนวนรอบของขดลวดจึงเพิ่มขึ้นเป็นเท่าตัว จึงได้มีการออกแบบวงจรเรียงกระแสเต็มคลื่นขึ้นมาใหม่ เรียกว่าวงจรบริดจ์ (The Full-wave Bridge Rectifier) ดังได้แสดงวงจรไว้ในรูป ก. สมมติว่ามีแรงดันเฟสบวกส่งเข้ายังวงจรมีผลทำให้ไดโอด D1,D2 เป็นฟอร์เวิร์ดสามารถนำกระแสตามทิศทางดังแสดงไว้ เกิดแรงดันตกไฟคร่อมโหลด RL ดังแสดงไว้ ในขณะที่ตอนนี้ D3,D4 อยู่ในสภาวะรีเวิร์ส เมื่อแรงดันไฟขาเข้าสลับเฟสเป็นเฟสลบดังแสดงไว้ในรูป ข. มีผลทำให้ไดโอด D3 ,D4 ฟอร์เวิร์ด D1,D2 รีเวิร์ส ไดโอด D3,D4 จึงนำกระแสไหลผ่านไปยัง RL ได้ดังทิศทางตามรูป โดยทิศทางกระแสดังกล่าวนี้เป็นทิศทางเดียวกับเมื่อมีเฟสบวกเข้ามา การทำงานจึงต้องมีการเรียงกระแสเต็มคลื่นได้

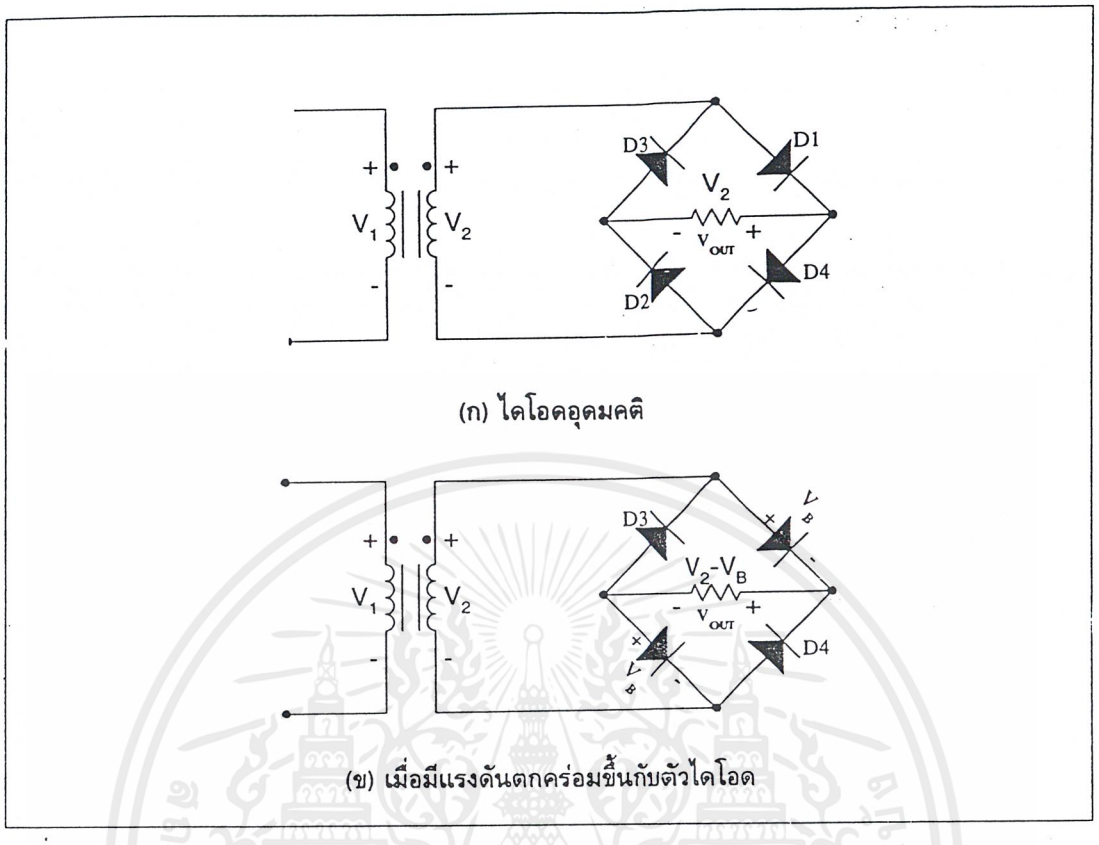


(ก) **เมื่อครึ่งไซเกิ้ลแรกมีแรงดันเฟสบวกเข้ามาทำให้ D_1 , D_2 เป็นฟอร์เวิร์ด และ D_3 , D_4 เป็นรีเวิร์ส**



(ข) **เมื่อครึ่งไซเกิ้ลแรกมีแรงดันเฟสบวกเข้ามาทำให้ D_3 , D_4 เป็นฟอร์เวิร์ด และ D_1 , D_2 เป็นรีเวิร์ส**

รูป 8.1 การทำงานของวงจรเรียงกระแสเต็มคลื่นแบบบริดจ์



รูป 8.2 แสดงแรงดันตกคร่อม R_L ของวงจรบริดจ์

แรงดันขาออกของบริดจ์ หากจะพิจารณาถึงแรงดันไฟตกคร่อมโหลดของวงจรบริดจ์ สามารถพิจารณาได้จากรูป เมื่อแรงดันไฟเฟสดังแสดงไว้ในรูป 8.2 ก ทำให้ไดโอด D1,D2 ฟอร์เวิร์ด หากไม่คำนึงแรงดันตกคร่อม ไดโอดที่เป็นศักย์โพเทนเชี่ยล แรงดันตกคร่อมรีซิสเตอร์จะเท่ากับแรงดันไฟของหม้อแปลงขดทุติยภูมิหรือเท่ากับค่า V 2 และเช่นเดียวกันหากแรงดันไฟมีการสลับเฟสเป็นตรงกันข้ามทำให้ไดโอด D3,D4 เป็นฟอร์เวิร์ดบ้าง ผลที่ออกมาจะเหมือนกัน

$$V_{OUT} = V_2$$

และหากคำนึงถึงค่าแรงดันตกคร่อมไดโอด เมื่อการไหลของกระแสในแต่ละคราวต้องส่งผ่านไดโอดคราวละ 2 ตัวจึงทำให้แรงดันที่ปรากฏที่รีซิสเตอร์ต้องลดลงเท่ากับค่าแรงดันตกคร่อมไดโอด 2 ตัวในแต่ละครั้ง ดังแสดงผลไว้ในรูป 8.2 ข

$$V_{OUT} = V_2 - 2V_B$$

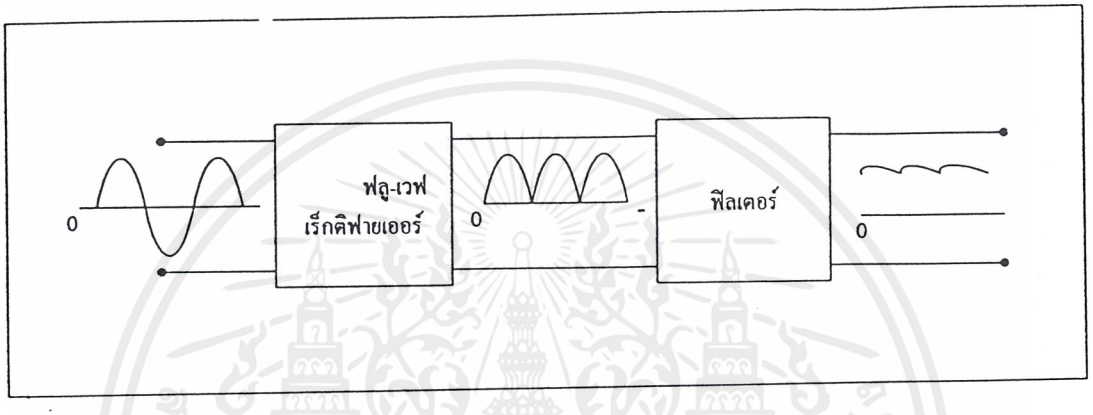
วงจรฟิลเตอร์-การกรองกระแสให้เรียบ

วัตถุประสงค์ของวงจรเร็กติฟายเออร์หรือวงจรเรียงกระแส คือความพยายามในการแปลงแรงดันไฟกระแสสลับให้เป็นแรงดันไฟกระแสตรง เพื่อเอาไปใช้กับอุปกรณ์อิเล็กทรอนิกส์ต่างๆ อย่างไรก็ดีเมื่อศึกษาวงจรเรียงกระแสจะพบว่าวงจรดังกล่าวยังให้ผลท้ายสุดเป็นได้เพียงแรงดันไฟตรงที่เรียกว่า “ พัลเซติ้ง ดีซี (Pulsating DC) หากต้องการทำให้แรงดันไฟดังกล่าวเรียบเหมือน

กับแรงดันไฟที่มาจากแบตเตอรี่ ต้องใช้อุปกรณ์กรองแรงดันที่เรียกว่า " เร็กติฟายเออร์ ฟิลเตอร์ (Rectifier Filter)

วงจรฟิลเตอร์หรือวงจรกรองไฟให้เรียบสามารถเขียนแนวคิดของวงจรได้ เมื่อมีการเร็กติฟายแรงดันไฟออกมาเป็นแบบเต็มคลื่นทำให้เกิดพัลเซติ้งขึ้น หากความถี่ที่ส่งเข้าเป็นความถี่ 50 เฮิรตซ์ เมื่อเร็กติฟายหรือเรียงกระแสแบบเต็มคลื่นแล้วจะได้พัลซ์เซติ้งความถี่ 100 เฮิรตซ์

เมื่อนำพัลซ์เซติ้งดังกล่าวผ่านไปยังวงจรฟิลเตอร์ทำให้สัญญาณทางออกมามีความเรียบขึ้น ดังแสดงไว้ในรูป 8.3



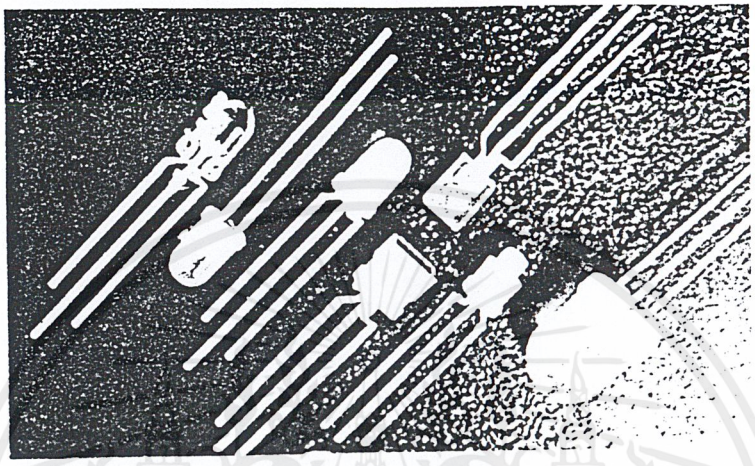
รูป 8.3 การฟิลเตอร์ในวงจรภาคจ่ายไฟ

ไลต์-อิมิตติ้ง-ไดโอด หรือ แอลอีดี

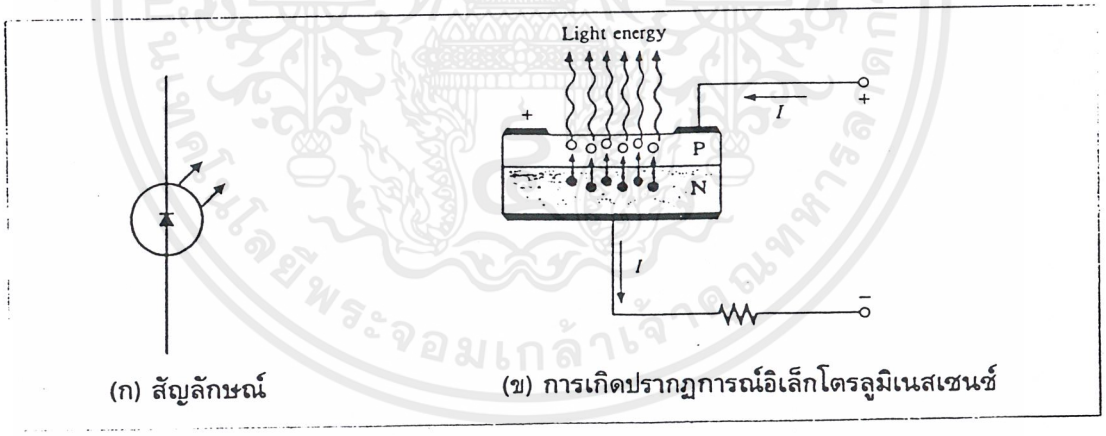
อุปกรณ์ไดโอดที่จัดเป็นอุปกรณ์ทางแสงหรือที่เรียกว่าออปติคอลไดโอด (Optical Diode) มีมากมายหลายตัวเหมือนกัน ไดโอดเปล่งแสงหรือไลต์อีมิตติ้งไดโอด (Light-Emitting-Diode) หรือแอลอีดี (LED) เป็นอุปกรณ์ทางแสงตัวหนึ่งที่เป็นที่รู้จักอย่างกว้างขวาง แอลอีดีเป็นอุปกรณ์ที่ต้องการไบอัสแบบฟอร์เวิร์ด เพื่อทำให้เกิดอิเล็กตรอนวิ่งข้ามจังชั่นพีเอ็นจากสารชนิด (N-Type) ไปหาโฮลในสารชนิดพี (P-type) เมื่ออิเล็กตรอนสามารถก้าวข้ามผ่านไปได้กลายเป็นอิเล็กตรอนอิสระ จะทำให้สารกึ่งตัวนำนั้นข้ามเข้าสู่แถบความนำได้ พลังงานจะมีมากกว่าระดับโฮลในแถบวาเลนซ์ พลังงานที่เกิดขึ้นตามติดมาจากผลของระบบนี้คือพลังงานที่เป็นความร้อนกับพลังงานที่เป็นแสง ทั้งนี้ยังมีองค์ประกอบของผิวหน้าของเนื้อสารอีกกรณีหนึ่งด้วย หากมีพื้นที่ผิวหน้ามากสารกึ่งตัวนำสั้นสามารถให้พลังงานแสงที่เรามองเห็นได้ ปรากฏการณ์ดังกล่าวเรียกว่าปรากฏการณ์ " อิเล็กโตรลูมิเนสเซนซ์" (Electroluminescence)

อาจมีการนำเอาสารกึ่งตัวนำบางชนิดมาทำเป็นแอลอีดีอย่างเช่นแกลเลี่ยมเซไนด์ (GaAs) หรือแกลเลี่ยมอาร์เซไนด์ฟอสไฟด์ (GaAsP) หรือแกลเลี่ยมฟอสไฟด์ (GaP) เราไม่ใช้ซิลิกอนหรือเยอรมันเนียมเพราะสารเหล่านี้จะเกิดความร้อน เมื่อพลังงานถูกแจกแจงไปในความร้อน ย่อมทำให้เกิดการพลังงานลดต่ำลง แอลอีดีที่ทำมาจากแกลเลี่ยมอาร์เซไนด์จะให้พลังงานแสงต่ำกว่าสีแดง

ปกติพลังงานแสงที่ต่ำกว่าแสงสีแดงตาของคนจะมองไม่เห็นและแสงที่สูงกว่าสีม่วงตาของคนก็มองไม่เห็นเช่นกันแสงที่ต่ำกว่าแดงเรียกทับศัพท์ว่า “ อินฟราเรด” (Infrared หรือ IR) หากแอลอีดีนั้นสร้างมาจากแกลเลี่ยมอาร์เซไนด์ฟอสไฟต์มันจะให้แสงสีเหลืองหรือแสงสีแดงออกมาให้เห็น หากสร้างมาจากแกลเลี่ยมฟอสไฟต์จะให้แสงสีแดงหรือเขียว จากการศึกษาเรื่องพลังงานแสงที่คนมองเห็นพบว่าแสงสีแดงมีช่วงกว้างความถี่มากกว่าแสงสีอื่น



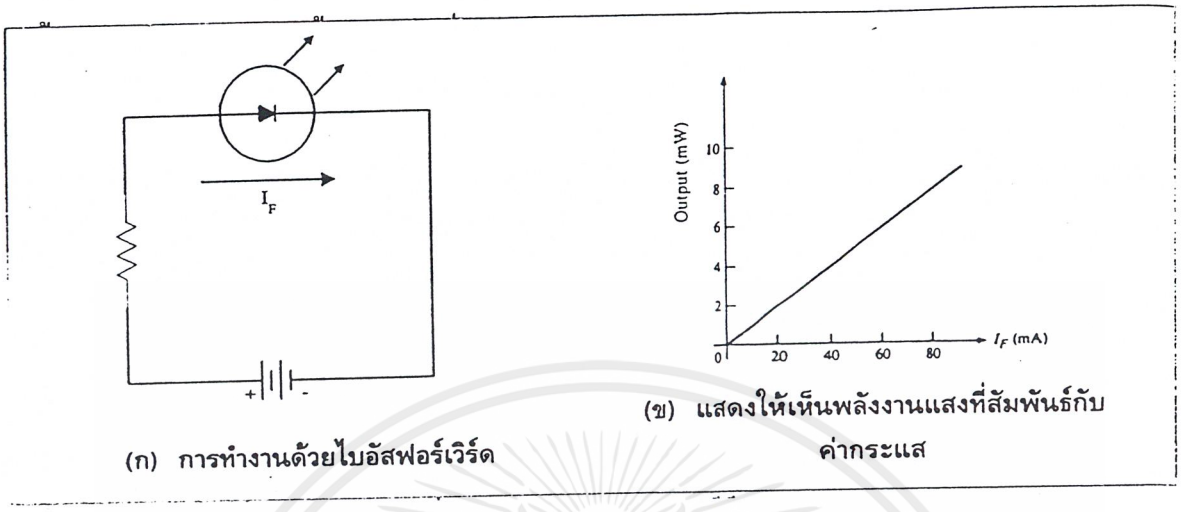
รูป 8.4 ไดโอดเปล่งแสง



(ก) สัญลักษณ์ (ข) การเกิดปรากฏการณ์อิเล็กโตรลูมิเนสเซนซ์

รูป 8.5 แสดงถึงไดโอดเปล่งแสง

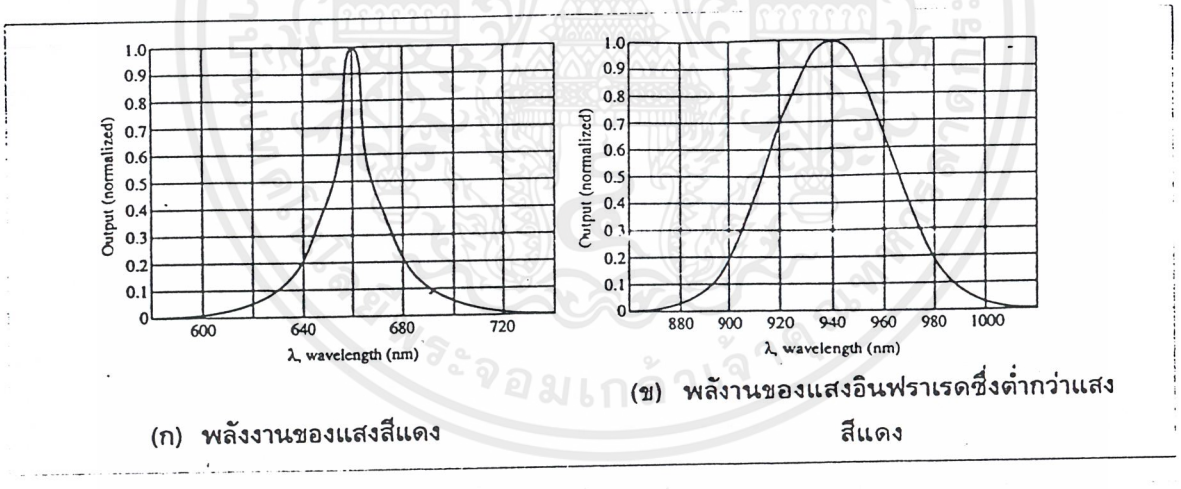
การเปล่งแสงของแอลอีดีจะมีผลออกมาเป็นสัดส่วนตรงต่อค่ากระแส ดังแสดงไว้ในรูป ความยาวคลื่นของพลังงานแสงจะเป็นตัวบอกถึงการมองเห็นหรือมองไม่เห็นอย่างเช่นอินฟราเรดเป็นแสงที่มองไม่เห็น ในขณะที่ LED ทั่วไปให้แสงสีแดงที่ตามองเห็น ดังแสดงในรูป 8.7 โดยทั่วไปของแอลอีดีที่มีความสามารถเปล่งพลังงานแสงซึ่งต้องเกี่ยวกับลักษณะเลนส์ของตัวแอลอีดี หากเลนส์ในตัวของแอลอีดีสร้างให้มีมุมแคบย่อมให้แสงเข้มส่วนในรูป เป็นการแสดงให้เห็นรูปแบบต่างๆที่เป็นตัวถังของแอลอีดี



(ก) การทำงานด้วยไบอัสฟอร์เวิร์ด

(ข) แสดงให้เห็นพลังงานแสงที่สัมพันธ์กับค่ากระแส

รูป 8.6 การทำงานของแอลอีดี



(ก) พลังงานของแสงสีแดง

(ข) พลังงานของแสงอินฟราเรดซึ่งต่ำกว่าแสงสีแดง

รูป 8.7 สเปกตรัมของแอลอีดีในรูปแบบของกราฟ

บทที่ 9
สรุป ปัญหาและข้อเสนอแนะ

การควบคุมอุปกรณ์ไฟฟ้าภายในอาคารโดยผ่านโมเด็มนี้ เป็นการใช้โปรแกรมควบคุมการจ่ายไฟให้อุปกรณ์ไฟฟ้าในระยะไกลโดยผ่านทางอินเตอร์เน็ต ซึ่งจำเป็นต้องใช้ความรู้ในด้าน ซีจีไอ การส่งข้อมูลผ่านพอร์ตอนุกรม ไมโครคอนโทรลเลอร์ และ วงจรรีเลย์ เป็นต้น

ปัญหา ที่พบคือในระยะแรกคือ การส่งข้อมูลผ่านพอร์ตอนุกรม RS232 เนื่องจากมีปัญหาในการรับและส่งค่าระหว่าง เครื่องเซิร์ฟเวอร์และไมโครคอนโทรลเลอร์ เนื่องจากความเร็วของ คอมพิวเตอร์ในปัจจุบันเร็วกว่า ไมโครคอนโทรลเลอร์ค่อนข้างมาก จึงทำให้ค่าที่รับและส่งผิดพลาดอยู่เสมอ ซึ่งจะแก้ได้โดยเขียนโปรแกรมให้หน่วงเวลา (Delay)ในคอมพิวเตอร์ของเครื่องเซิร์ฟเวอร์ เพื่อให้ไมโครคอนโทรลเลอร์ ประมวลผลได้ทันกับเครื่องเซิร์ฟเวอร์ ปัญหาต่อมาที่พบคือ ในการควบคุมโปรแกรมให้เป็นซีจีไอ ให้รันได้บนอินเตอร์เน็ตนั้น มีให้เลือกใช้ได้หลายโปรแกรม ซึ่งไม่ว่าโปรแกรมตัวไหนก็สามารถนำมาเขียนเป็นซีจีไอ ได้แต่ทั้งนี้ก็ขึ้นอยู่กับผู้ใช้และโอเปอเรชั่นซิสเต็ม (Operation System หรือ OS) ของระบบเป็นโปรแกรมใดจึงจะสามารถ เขียนออกมาได้ดีที่สุด ปัญหาสุดท้ายคืออุปกรณ์ ฮาร์ดแวร์ที่นำมาใช้ในการควบคุมระบบไฟฟ้าคือ วงจรรีเลย์และเลือกวงจรตรวจสอบกระแสไฟฟ้าโดยใช้การเปรียบเทียบมาช่วยในการตรวจสอบสถานะของอุปกรณ์ไฟฟ้า ซึ่งในอุปกรณ์ไฟฟ้าที่จะนำมาควบคุม 1 ชุดนั้น ก็จำเป็นต้องใช้วงจรควบคุมระบบการจ่ายไฟฟ้าและวงจรตรวจสอบสถานะของอุปกรณ์ไฟฟ้าอย่างละ 1 วงจร ซึ่งอาจจะทำให้เกิดการสิ้นเปลืองได้ ถ้านำมาใช้ควบคุมอุปกรณ์ไฟฟ้าภายในอาคารที่มีจำนวนมาก ในห้องๆเดียว ดังนั้นผู้ออกแบบวงจร จึงต้องคำนึงถึงความเหมาะสมที่จะเลือกนำระบบนี้มาควบคุมอุปกรณ์ไฟฟ้าให้เกิดประโยชน์สูงสุด

บรรณานุกรม

1. ชูชัย ธนสารตั้งเจริญ, " ทฤษฎีวงจรอิเล็กทรอนิกส์" , ฟิสิกส์เซ็นเตอร์,กรุงเทพ
2. เจน สงสมพันธุ์, " วงจรอิเล็กทรอนิกส์" ,สถาบันอิเล็กทรอนิกส์กรุงเทพรังสิต,2540
3. ปรเมษฐ์ ประนยานันท์และปิยพงศ์ เผ่าวณิช , " คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51 , ซีเอ็ดยูเคชั่น, 2540
4. สมยศ จุณณะปิยะ " การใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51" , คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ,2537
5. ทรงเกียรติ ภาวดี, "แกะรอย CGI" , ซีเอ็ดยูเคชั่น,2542
6. สัจจะ จรัสรุ่งรวีวร และจักรพงษ์ สุขประเสริฐ , " คู่มือการสร้างแอปพลิเคชั่นด้วย Delphi 5.0 ฉบับสมบูรณ์" อินโฟเพรส,2543
7. เยาวภา สงวนวรรณ และวิทยา สงวนวรรณ, " Design Web Graphics with HTML 3.2" ,เฟิสท์แปซิฟิกมิเดีย,2540
8. www.matlus.com

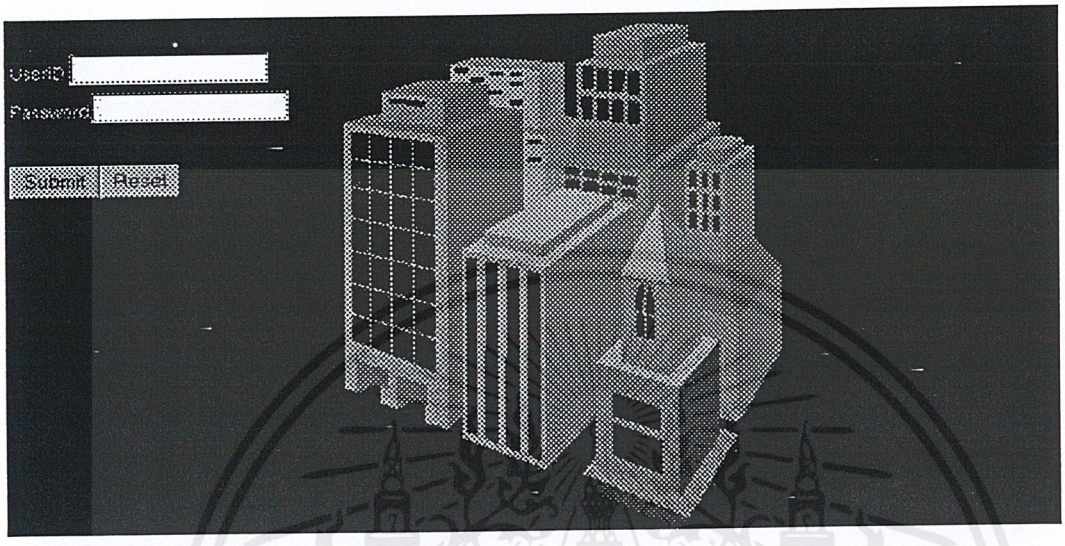
ตัวอย่างโปรแกรมไมโครคอนโทรลเลอร์

```
        org  000h
        jmp 100h
        org  100h
setbaudrate:
        mov p2,#0FFh   ;p2 = Input
        mov pcon,#00h  ;smod = 0
        mov scon,#50h  ;serial (mode1)ren = 1
        mov tmod,#20h  ;timer1 mode2
        mov th1,#0fdh   ; 9600 buad
        setb tr1       ;starttimer1
        mov p1,#00h    ;set พอร์ต1 เป็น0
        mov r6,#00h
        mov r7,#00h
    main:
        jnb ri,main
        clr  ri
          clr   ti
        mov a,sbuf
        cjne a,#105,status2
    status1:
        mov a,p2
        mov r6,a
        xrl   a,#0FFh ;เปลี่ยนค่า a ให้เป็นเหมือนเดิม
        mov sbuf,a
    wait1:
        jnb ti,wait1
        clr ti
        jmp main
    status2:
```

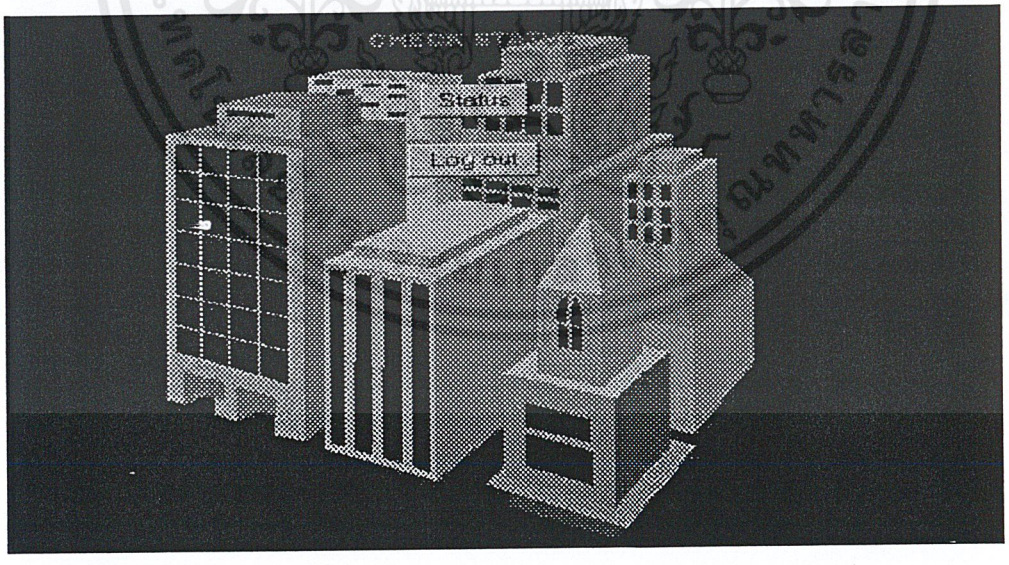
```
        cjne a,#100,closelight
openlight:
        jnb ri,openlight
        clr ri
        mov a,sbuf
        mov r7,a
        mov a,#0ffh
        clr c
openshiftport1:
        rlc a
        djnz r7,openshiftport1
        anl     a,r6
        mov     p1,a
        mov       r6,a
        jmp main
closelight:
        jnb     ri,closelight
        clr     ri
        mov     a,sbuf
        mov     r7,a
        mov     a,#00h
        setb    c
closeshiftport1:
        rlc      a
        djnz    r7,closeshiftport1
        orl     a,r6
        mov     p1,a
        mov       r6,a
        jmp main
end
```

ตัวอย่าง โปรแกรมควบคุมระบบไฟฟ้าภายในอาคาร

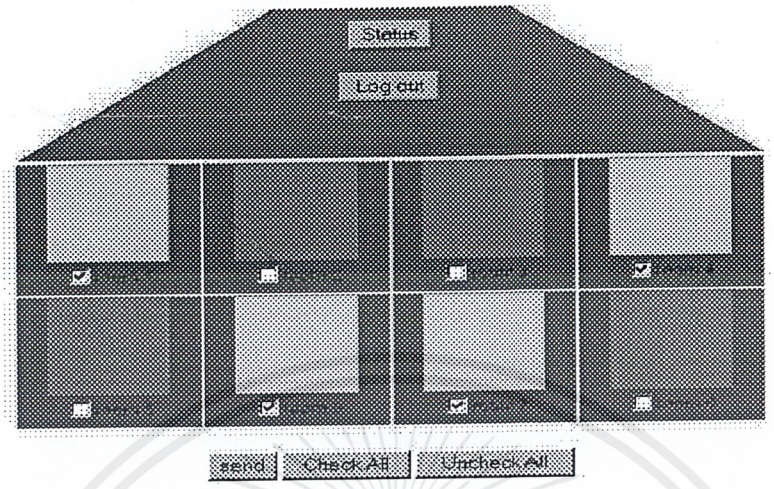
1 เปิดโปรแกรม Internet explorer หรือ Nescapte เปิด URL รัน 161.246.7.52/scripts/myisapi.exe/logon จะได้หน้าเวบเพจเป็นรูปดังนี้



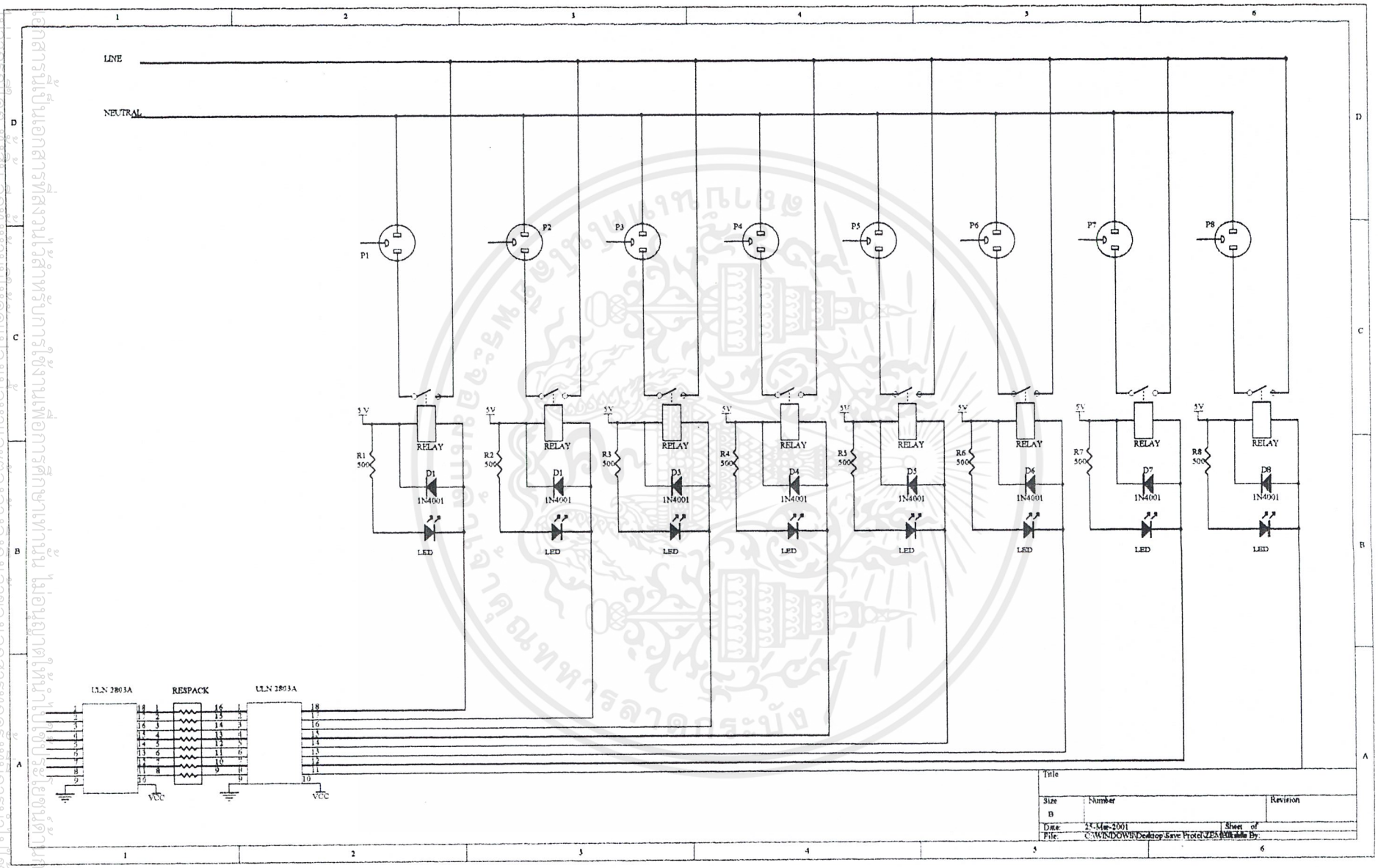
2. ทำการ ป้อน USER NAME และ PASSWORD คือ CONTROL และ KMITL ตามลำดับจะได้หน้าเวบเพจเป็นดังนี้

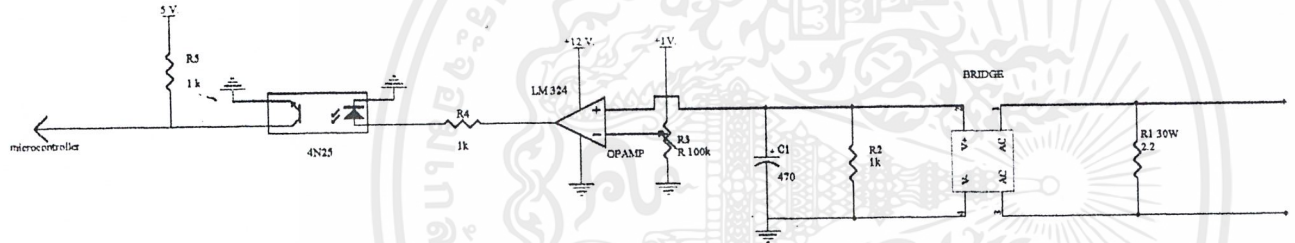


3. ทำการคลิกปุ่ม STATUS จะสามารถตรวจสอบสถานะของอุปกรณ์ไฟฟ้าภายในห้องได้



จะเห็นว่าจากรูป นี้คือมีห้องที่ 1 ,4 ,6,7 ที่เปิดอุปกรณ์ไฟฟ้าอยู่ เราสามารถ เปิดปิดอุปกรณ์ไฟฟ้าในห้องต่างๆ โดย คลิกเช็กซ์บอกซ์ในแต่ละห้องได้ตามต้องการ


LINE
NEUTRAL
P1
P2
P3
P4
P5
P6
P7
P8
5V
RELAY
R1 500
R2 500
R3 500
R4 500
R5 500
R6 500
R7 500
R8 500
D1 1N4001
D3 1N4001
D4 1N4001
D5 1N4001
D6 1N4001
D7 1N4001
D8 1N4001
LED
ULN 2803A
RESPACK
ULN 2803A
VCC
VCC
Title
Size
B
Number
Revision
Date: 25-Mar-2001
Sheet of
File: C:\WINDOWS\Desktop\Save Protel\ZEMAX.ddb
Drawn By:



Title		
Size B	Number	Revision
Date: 25-Mar-2001		Sheet of
File: C:\WINDOWS\Desktop\Save Protel\ZES.ddb		Drawn By:

Features

- Compatible with MCS-51™ Products
- 2 Kbytes of Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- 2.7 V to 6 V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Two-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 15 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Five Interrupt Sources
- Programmable Serial UART Channel
- Direct LED Drive Outputs
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes

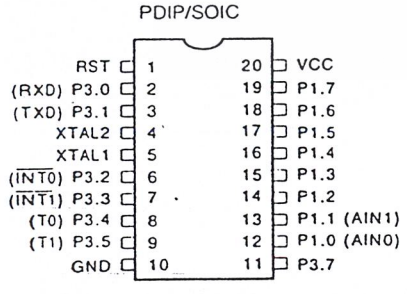
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2 Kbytes 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. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2 Kbytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five source two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 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 Configuration

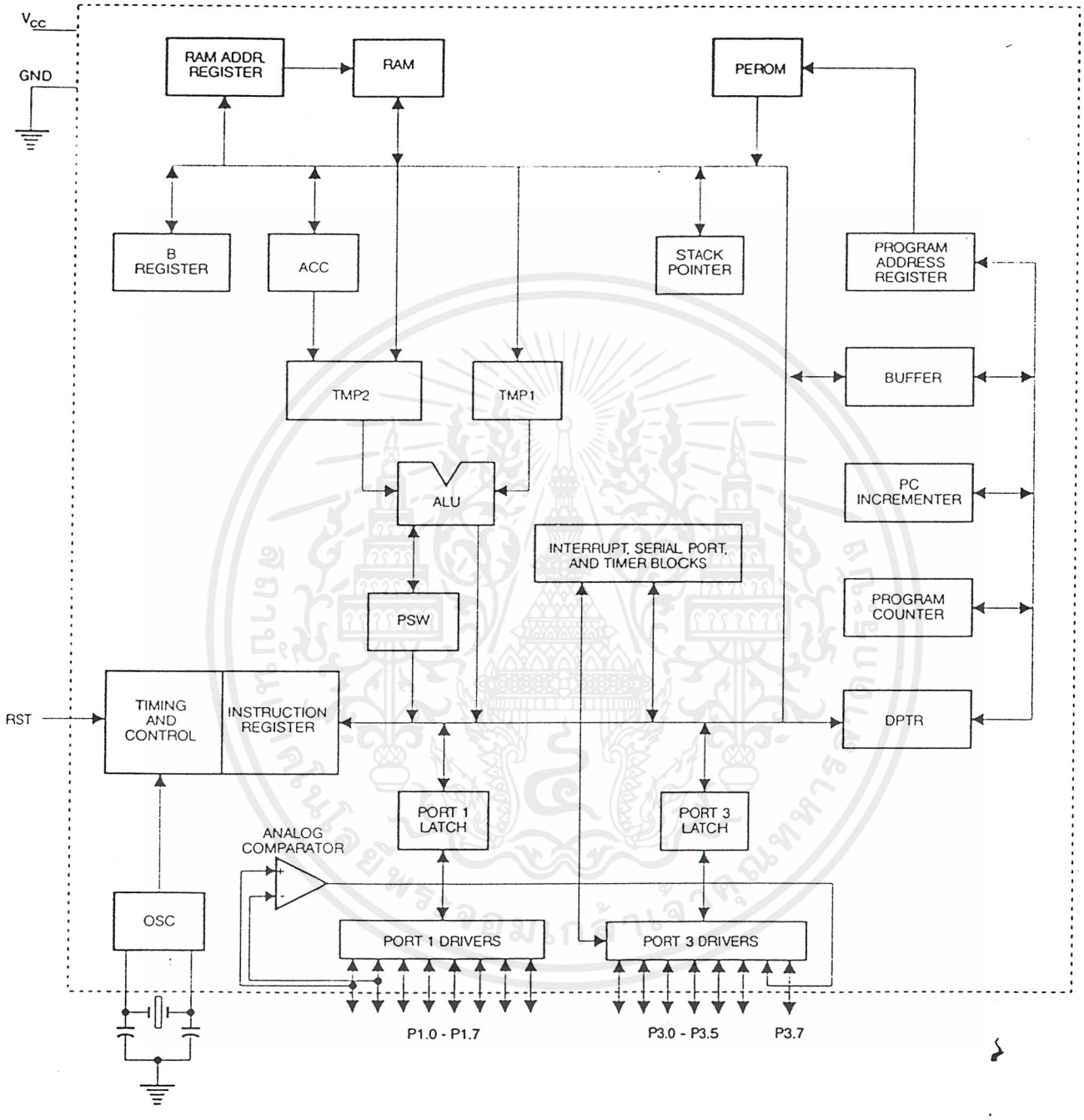
PDIP/SOIC



8-Bit Microcontroller with 2 Kbytes Flash

AT89C2051

Block Diagram



Pin Description

Vcc

Supply voltage.

GND

Ground.

Port 1

Port 1 is an 8-bit bidirectional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (I_{IL}) because of the internal pullups.

Port 1 also receives code data during Flash programming and program verification.

Port 3

Port 3 pins P3.0 to P3.5, P3.7 are seven bidirectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. 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 AT89C2051 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)

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

RST

Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the oscillator is running resets the device.

Each machine cycle takes 12 oscillator or clock cycles.

XTAL1

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

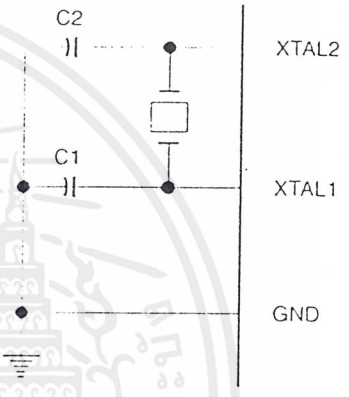
XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

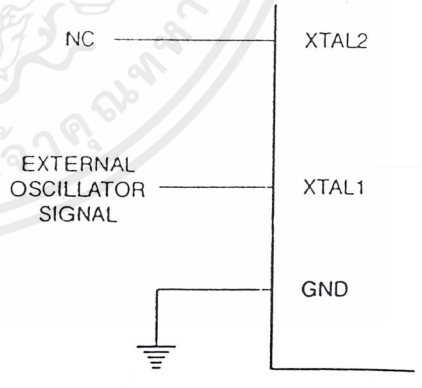
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected 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.

Figure 1. Oscillator Connections



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

Figure 2. External Clock Drive Configuration



Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in the table below.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Table 1. AT89C2051 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H									0CFH
0C0H									0C7H
0B8H	IP X0000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 00000000								0AFH
0A0H									0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H		SP 00000111	DPL 00000000	DPH 00000000				PCON 00000000	87H

Restrictions on Certain Instructions

The AT89C2051 and AT89C1051 are economical and cost-effective members of Atmel's growing family of microcontrollers. They contain 2 Kbytes and 1 Kbyte of flash program memory, respectively. Both are fully compatible with the MCS-51 architecture, and can be programmed using the MCS-51 instruction set. However, there are a few considerations one must keep in mind when utilizing certain instructions to program these devices.

All the instructions related to jumping or branching should be restricted such that the destination address falls within the physical program memory space of the device, which is 2K for the AT89C2051. This should be the responsibility of the software programmer. For example, LCALL 7E0H would be valid for the AT89C2051 (with 2K of memory), but LCALL 900H would not.

1. Branching instructions:

LCALL, LJMP, ACALL, AJMP, SJMP, JMP @A+DPTR

These unconditional branching instructions will execute correctly as long as the programmer keeps in mind that the destination branching address must fall within the physical boundaries of the program memory size (locations 00H to 7FFH for the 89C2051). Violating the physical space limits may cause unknown program behavior.

CJNE [...], DJNZ [...], JB, JNB, JC, JNC, JBC, JZ, JNZ With these conditional branching instructions the same rule above applies. Again, violating the memory boundaries may cause erratic execution.

For applications involving interrupts the normal interrupt service routine address locations of the 80C51 family architecture have been preserved.

2. MOVX-related instructions, Data Memory:

The AT89C2051 contains 128 bytes of internal data memory, while the AT89C1051 has 64 bytes. Thus, in the AT89C1051 the stack depth is limited to 64 bytes, the amount of available RAM. External DATA memory access is not supported in either device, nor is external PROGRAM memory execution. Therefore, no MOVX [...] instructions should be included in the program.

A typical 80C51 assembler will still assemble instructions, even if they are written in violation of the restrictions mentioned above. It is the responsibility of the controller user to know the physical features and limitations of the device being used and adjust the instructions used correspondingly.

Program Memory Lock Bits

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

Lock Bit Protection Modes[1]

Program Lock Bits			
	LB1	LB2	Protection Type
1	U	U	No program lock features.
2	P	U	Further programming of the Flash is disabled.
3	P	P	Same as mode 2, also verify is disabled.

Note: 1. The Lock Bits can only be erased with the Chip Erase operation

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.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

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.

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.

P1.0 and P1.1 should be set to '0' if no external pullups are used, or set to '1' if external pullups are used.

Programming The Flash

The AT89C2051 is shipped with the 2 Kbytes of on-chip PEROM code memory array in the erased state (i.e., contents = FFH) and ready to be programmed. The code memory array is programmed one byte at a time. *Once the array is programmed, to re-program any non-blank byte, the entire memory array needs to be erased electrically.*

Internal Address Counter: The AT89C2051 contains an internal PEROM address counter which is always reset to 000H on the rising edge of RST and is advanced by applying a positive going pulse to pin XTAL1.

Programming Algorithm: To program the AT89C2051, the following sequence is recommended.

1. Power-up sequence:
   Apply power between V_{CC} and GND pins
   Set RST and XTAL1 to GND
   With all other pins floating, wait for greater than 10 milliseconds
2. Set pin RST to 'H'
   Set pin P3.2 to 'H'
3. Apply the appropriate combination of 'H' or 'L' logic levels to pins P3.3, P3.4, P3.5, P3.7 to select one of the programming operations shown in the PEROM Programming Modes table.

To Program and Verify the Array:

4. Apply data for Code byte at location 000H to P1.0 to P1.7.
5. Raise RST to 12V to enable programming.
6. Pulse P3.2 once to program a byte in the PEROM array or the lock bits. The byte-write cycle is self-timed and typically takes 1.2 ms.
7. To verify the programmed data, lower RST from 12V to logic 'H' level and set pins P3.3 to P3.7 to the appropriate levels. Output data can be read at the port P1 pins.
8. To program a byte at the next address location, pulse XTAL1 pin once to advance the internal address counter. Apply new data to the port P1 pins.
9. Repeat steps 5 through 8, changing data and advancing the address counter for the entire 2 Kbytes array or until the end of the object file is reached.
10. Power-off sequence:
    set XTAL1 to 'L'
    set RST to 'L'
    Float all other I/O pins
    Turn Vcc power off

$\overline{\text{Data}}$ Polling: The AT89C2051 features $\overline{\text{Data}}$ 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 P1.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{\text{Data}}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{\text{Busy}}$: The Progress of byte programming can also be monitored by the RDY/$\overline{\text{BSY}}$ output signal. Pin P3.1 is pulled low after P3.2 goes High during programming to indicate BUSY. P3.1 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed code data can be read back via the data lines for verification:

1. Reset the internal address counter to 000H by bringing RST from 'L' to 'H'.
2. Apply the appropriate control signals for Read Code data and read the output data at the port P1 pins.
3. Pulse pin XTAL1 once to advance the internal address counter.
4. Read the next code data byte at the port P1 pins.
5. Repeat steps 3 and 4 until the entire array is read.

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 PEROM array (2 Kbytes) and the two Lock Bits are erased electrically by using the proper combination of control signals and by holding P3.2 low for 10 ms. The code array is written with all "1"s in the Chip Erase operation and must be executed before any non-blank memory byte can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 001H, and 002H, except that P3.5 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(001H) = 21H indicates 89C2051

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	P3.2/$\overline{\text{PROG}}$	P3.3	P3.4	P3.5	P3.7
Write Code Data[1,3]	12V	⎍	L	H	H	H
Read Code Data[1]	H	H	L	L	H	H
Write Lock Bit - 1	12V	⎍	H	H	H	H
Bit - 2	12V	⎍	H	H	L	L
Chip Erase	12V	⎍[2]	H	L	L	L
Read Signature Byte	H	H	L	L	L	L

Notes: 1. The internal PEROM address counter is reset to 000H on the rising edge of RST and is advanced by a positive pulse at XTAL1 pin.
2. Chip Erase requires a 10 ms $\overline{\text{PROG}}$ pulse.
3. P3.1 is pulled Low during programming to indicate RDY/$\overline{\text{BSY}}$.

Figure 3. Programming the Flash Memory

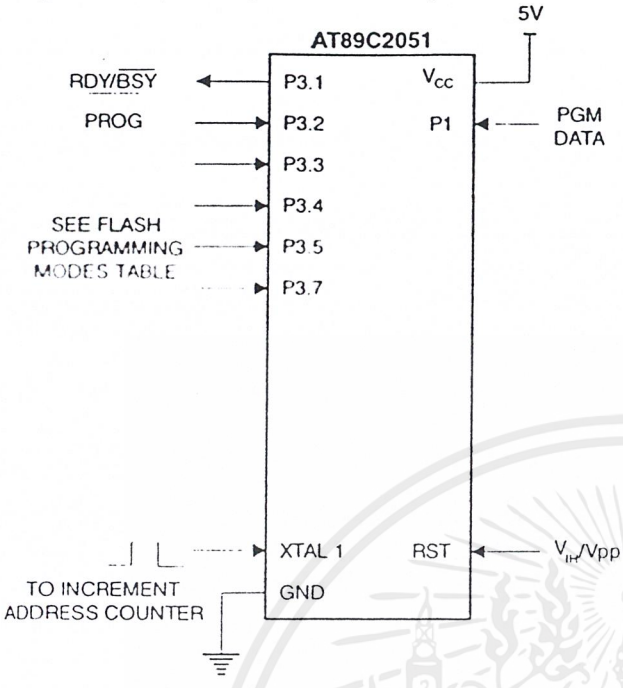
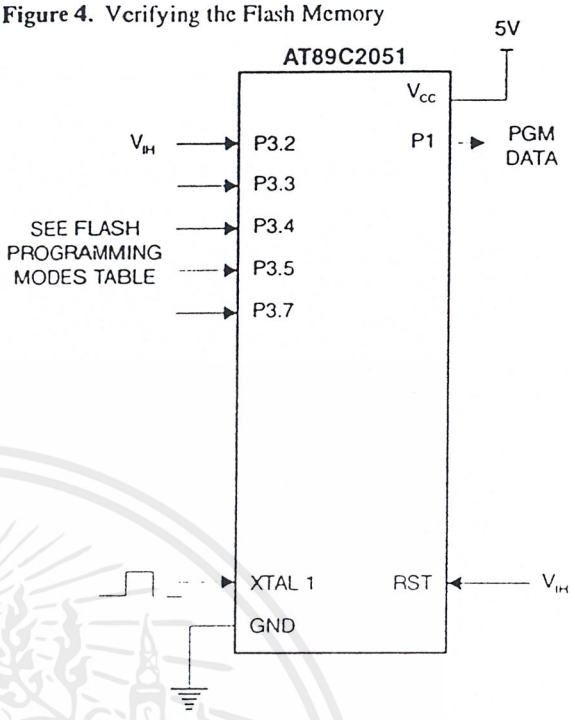


Figure 4. Verifying the Flash Memory

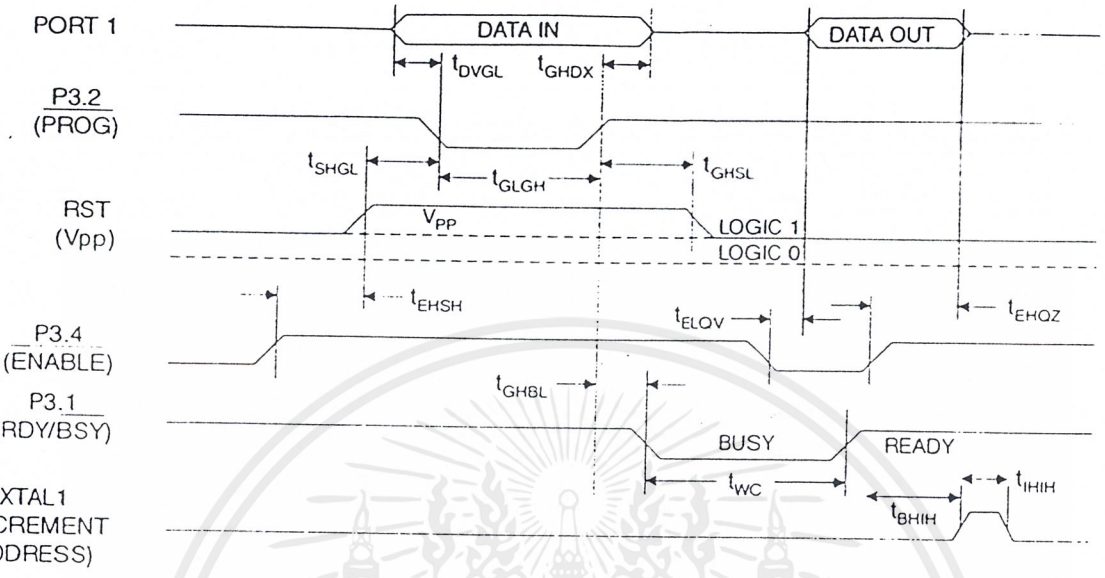


Flash Programming and Verification Characteristics

T_A = 21°C to 27°C, V_{CC} = 5.0 ± 10%

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Enable Voltage	11.5	12.5	V
I_{PP}	Programming Enable Current		250	µa
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	1.0		µs
t_{GHDX}	Data Hold After $\overline{PROG}$	1.0		µs
t_{EHSH}	P3.4 ($\overline{ENABLE}$) High to V_{PP}	1.0		µs
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		1.0	µs
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	1.0	µs
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		50	ns
t_{WC}	Byte Write Cycle Time		2.0	ms
t_{BHIH}	RDY/$\overline{BSY}$\ to Increment Clock Delay	1.0		µs
t_{IHIL}	Increment Clock High	200		ns

Flash Programming and Verification Waveforms



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	-1.0 V to +7.0 V
Maximum Operating Voltage	6.6 V
DC Output Current	25.0 mA

*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.

D.C. Characteristics

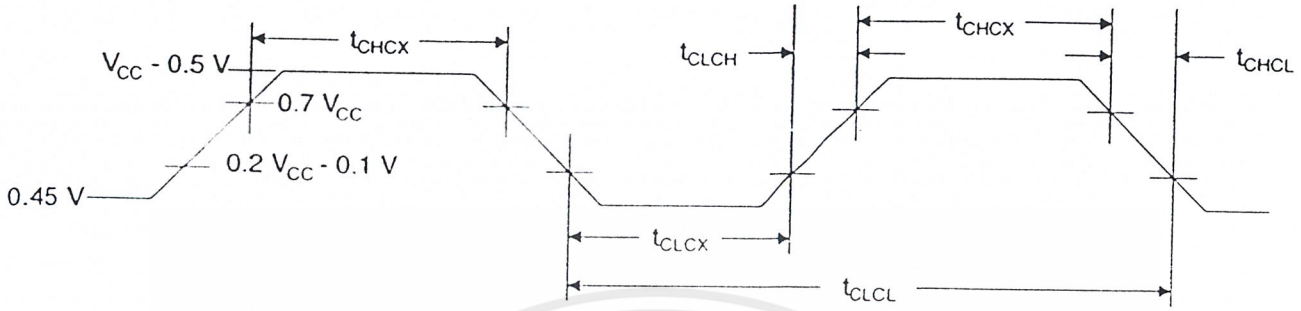
T_A = -40°C to 85°C, V_{CC} = 2.7 V to 6.0 V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage		-0.5	0.2 V_{CC}-0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
V_{OL}	Output Low Voltage[1] (Ports 1, 3)	I_{OL} = 20 mA, V_{CC} = 5 V I_{OL} = 10 mA, V_{CC} = 2.7 V		0.5	V
V_{OH}	Output High Voltage (Ports 1, 3)	I_{OH} = -80 µA, V_{CC} = 5 V ± 10%	2.4		V
		I_{OH} = -30 µA	0.75 V_{CC}		V
		I_{OH} = -12 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3)	V_{IN} = 0.45 V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3)	V_{IN} = 2 V		-750	µA
I_{LI}	Input Leakage Current (Port P1.0, P1.1)	0 < V_{IN} < V_{CC}		±10	µA
V_{OS}	Comparator Input Offset Voltage	V_{CC} = 5 V		20	mV
V_{CM}	Comparator Input Common Mode Voltage		0	V_{CC}	V
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz, V_{CC} = 6 V/3 V		15/5.5	mA
		Idle Mode, 12 MHz, V_{CC} = 6 V/3 V P1.0 & P1.1 = 0V or V_{CC}		5/1	mA
	Power Down Mode[2]	V_{CC} = 6 V P1.0 & P1.1 = 0V or V_{CC}		100	µA
		V_{CC} = 3 V P1.0 & P1.1 = 0V or V_{CC}		20	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin:20 mA
Maximum total IOL for all output pins:80 mA
If IOL exceeds the test condition, VOL 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 2 V.

External Clock Drive Waveforms



External Clock Drive

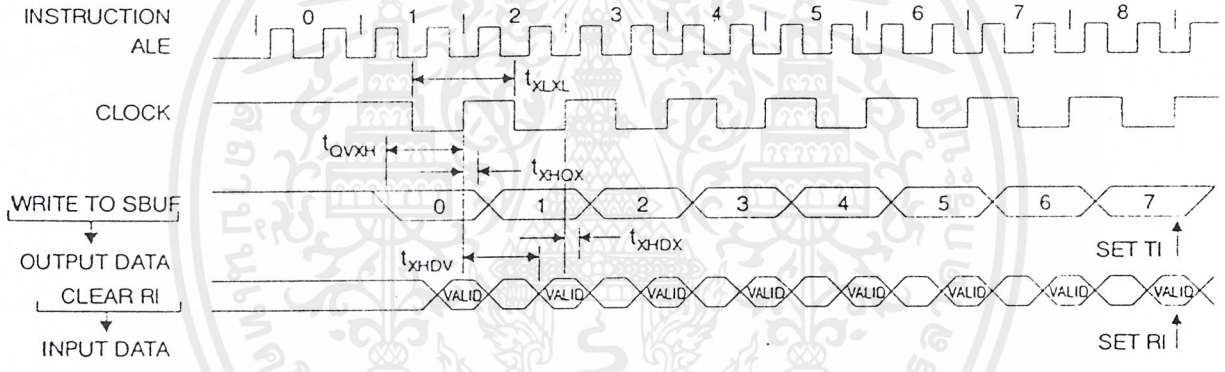
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

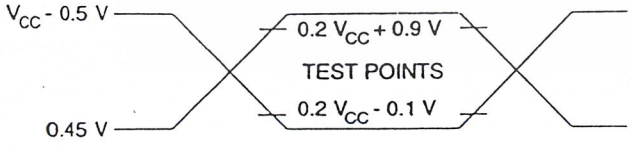
(V_{CC} = 5.0 V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz 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		$2t_{CLCL}$-33		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		10tCLCL-133	ns

Shift Register Mode Timing Waveforms

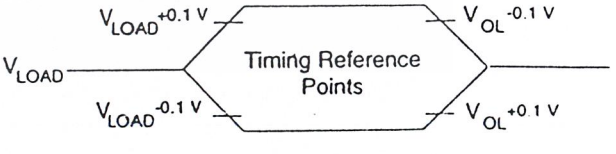


AC Testing Input/Output Waveforms [1]



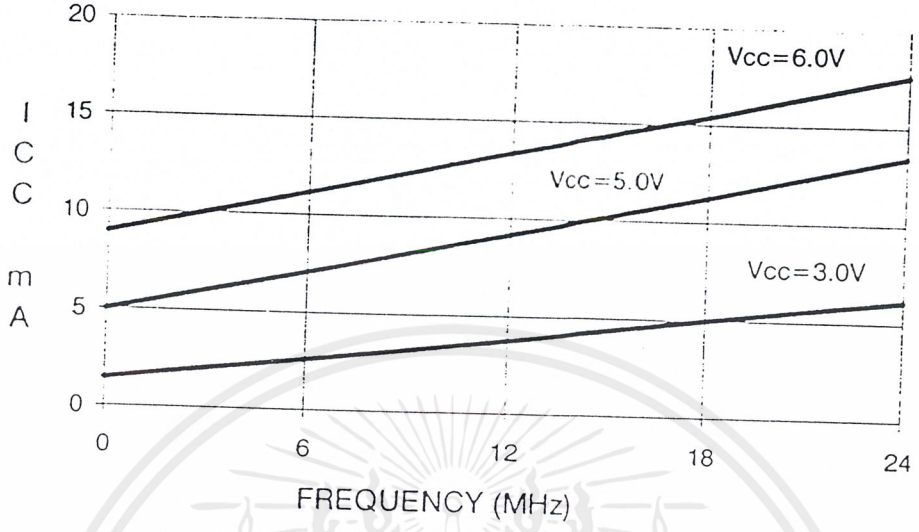
Note: 1. AC Inputs during testing are driven at V_{CC} - 0.5 V for a logic 1 and 0.45 V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

Float Waveforms [1]

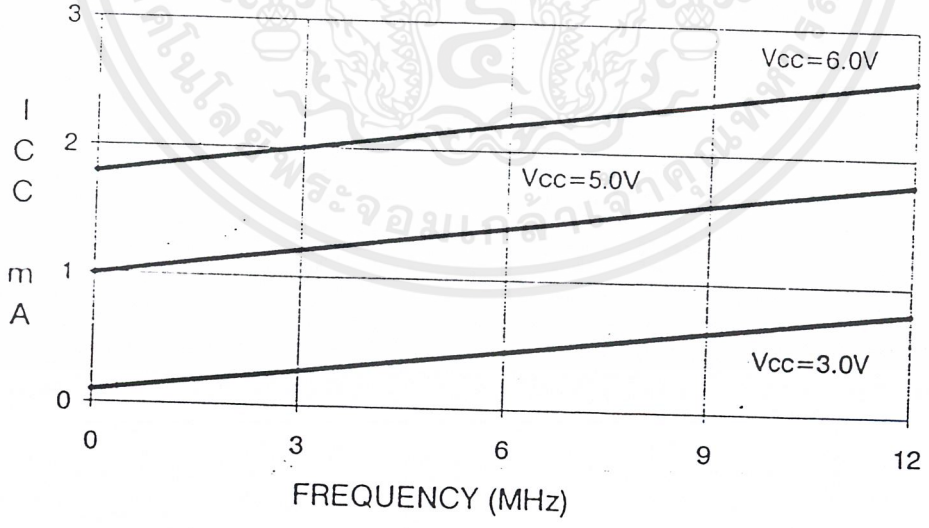


Note: 1. For timing purposes, a port pin is no longer floating when 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

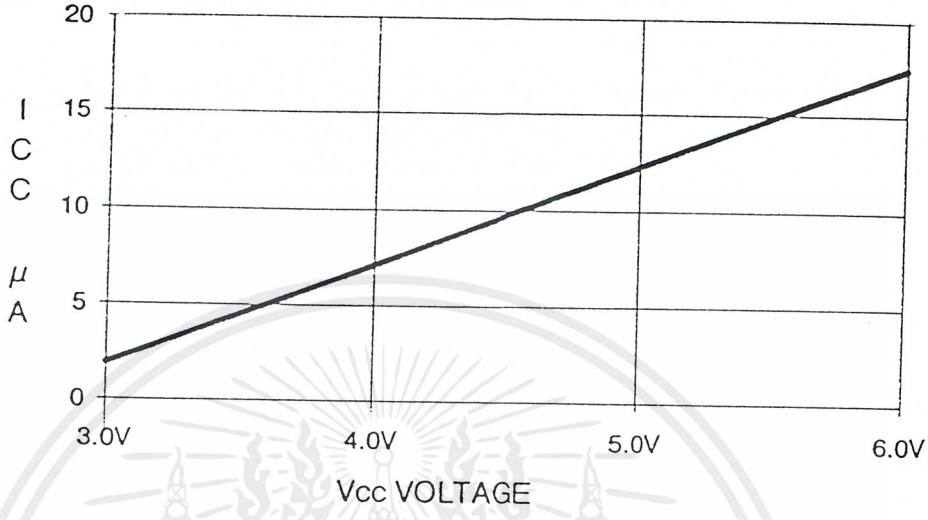
AT89C2051
TYPICAL ICC - ACTIVE (85°C)



AT89C2051
TYPICAL ICC - IDLE (85°C)



TYPICAL ICC vs. VOLTAGE - POWER DOWN (85°C)



Note: 1. XTAL1 tied to GND for I_{CC} (power down).
2. P.1.0 and P1.1 = V_{CC} or GND.
3. Lock bits programmed.

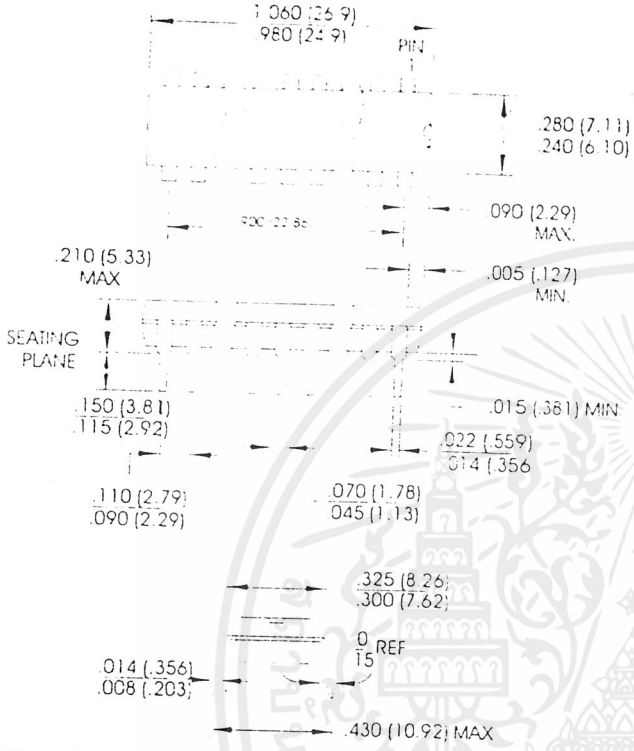
Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	2.7 V to 6.0 V	AT89C2051-12PC AT89C2051-12SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-12PI AT89C2051-12SI	20P3 20S	Industrial (-40°C to 85°C)
24	4.0 V to 6.0 V	AT89C2051-24PC AT89C2051-24SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-24PI AT89C2051-24SI	20P3 20S	Industrial (-40°C to 85°C)

Package Type	
20P3	20 Lead, 0.300" Wide, Plastic Dual In-line Package (PDIP)
20S	20 Lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)

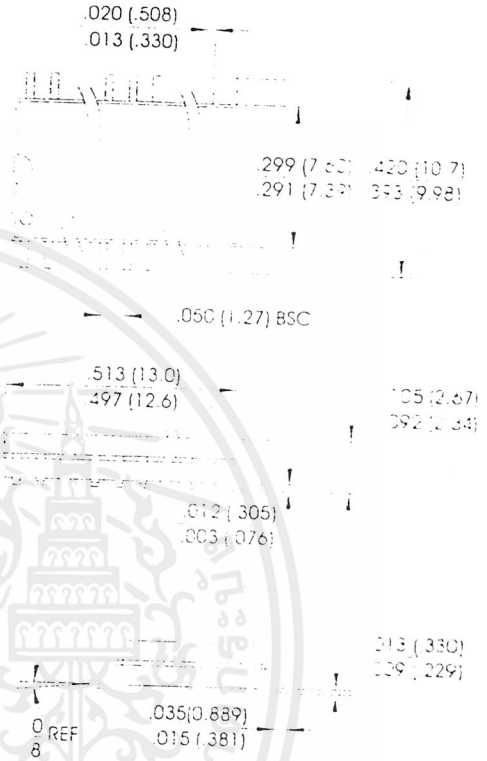
Packaging Information

20P3, 20 Lead, 0.300" Wide,
Plastic Dual Inline Package (PDIP)



*Controlling dimension: millimeters

20S, 20 Lead, 0.300" Wide,
Plastic Gull Wing Small Outline (SOIC)

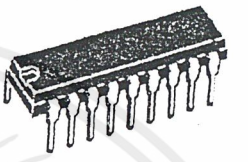


SGS-THOMSON MICROELECTRONICS

ULN2801A ULN2802A - ULN2803A ULN2804A - ULN2805A

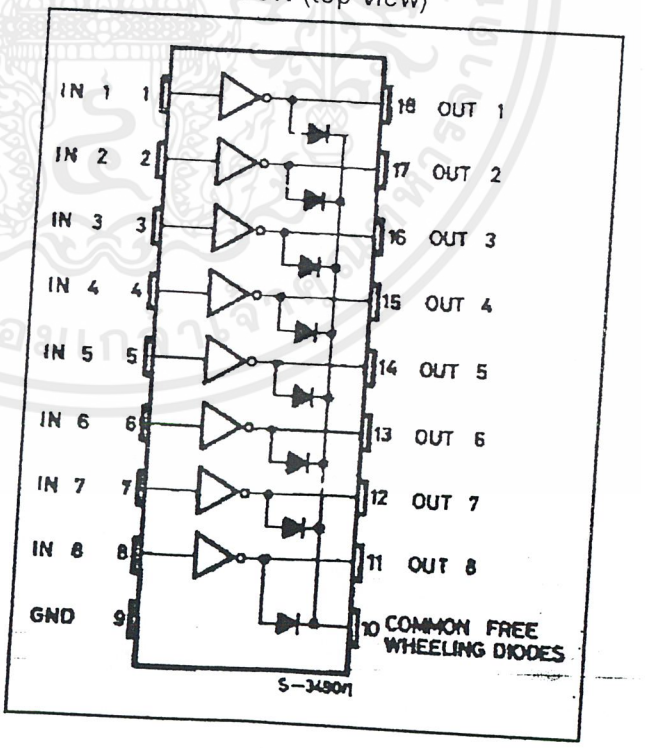
EIGHT DARLINGTON ARRAYS

- EIGHT DARLINGTONS WITH COMMON EMITTERS
- OUTPUT CURRENT TO 500 mA
- OUTPUT VOLTAGE TO 50 V
- INTEGRAL SUPPRESSION DIODES
- VERSIONS FOR ALL POPULAR LOGIC FAMILIES
- OUTPUT CAN BE PARALLELED
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY BOARD LAYOUT



DIP18

PIN CONNECTION (top view)



DESCRIPTION

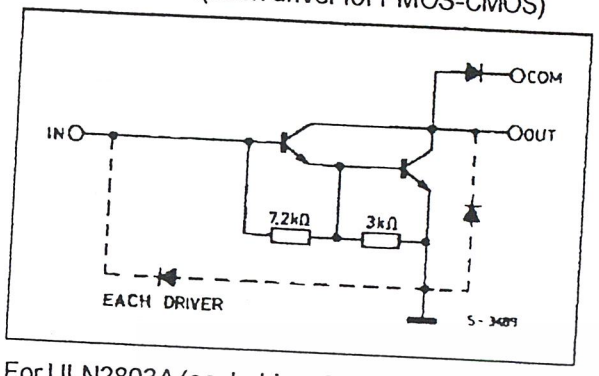
The ULN2801A-ULN2805A each contain eight darlington transistors with common emitters and integral suppression diodes for inductive loads. Each darlington features a peak load current rating of 600mA (500mA continuous) and can withstand at least 50V in the off state. Outputs may be paralleled for higher current capability.

Five versions are available to simplify interfacing to standard logic families : the ULN2801A is designed for general purpose applications with a current limit resistor; the ULN2802A has a 10.5kΩ input resistor and zener for 14-25V PMOS; the ULN2803A has a 2.7kΩ input resistor for 5V TTL and CMOS ; the ULN2804A has a 10.5kΩ input resistor for 6-15V CMOS and the ULN2805A is designed to sink a minimum of 350mA for standard and Schottky TTL where higher output current is required.

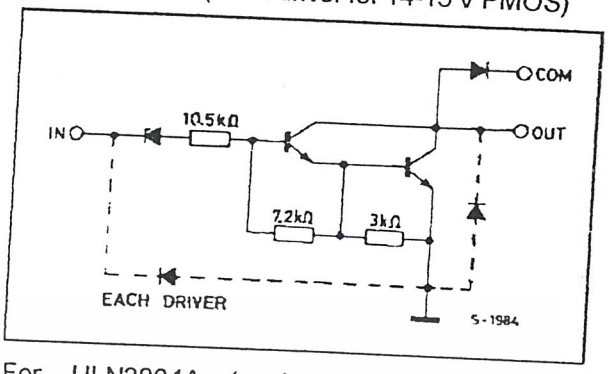
All types are supplied in a 18-lead plastic DIP with a copper lead from and feature the convenient input-opposite-output pinout to simplify board layout.

SCHEMATIC DIAGRAM AND ORDER CODES

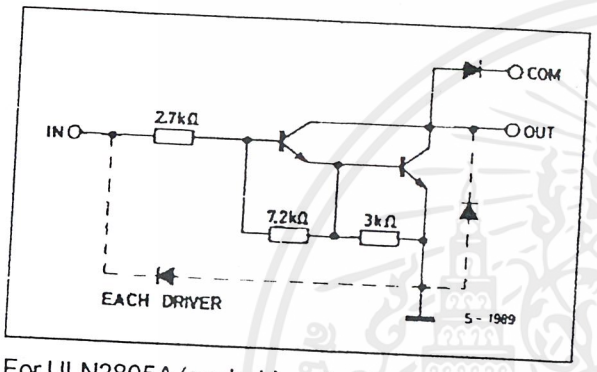
For ULN2801A (each driver for PMOS-CMOS)



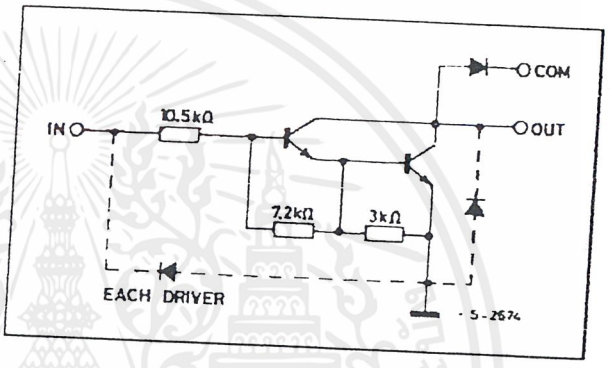
For ULN2802A (each driver for 14-15 V PMOS)



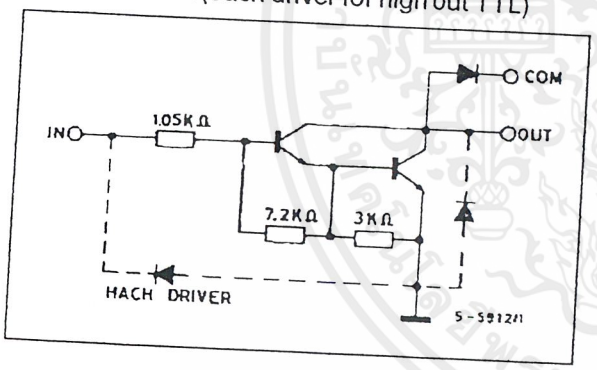
For ULN2803A (each driver for 5 V, TTL/CMOS)



For ULN2804A (each driver for 6-15 V CMOS/PMOS



For ULN2805A (each driver for high out TTL)



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_o	Output Voltage	50	V
V_i	Input Voltage for ULN2802A, UL2803A, ULN2804A for ULN2805A	 30 15	V
I_C	Continuous Collector Current	500	mA
I_B	Continuous Base Current	25	mA
P_{tot}	Power Dissipation (one Darlington pair) (total package)	 1.0 2.25	W
T_{amb}	Operating Ambient Temperature Range	– 20 to 85	°C
T_{stg}	Storage Temperature Range	– 55 to 150	°C
Tj	Junction Temperature Range	– 20 to 150	°C

THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	55	°C/W

ELECTRICAL CHARACTERISTICS (T_{amb} = 25°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit	Fig.
I_{CEX}	Output Leakage Current	V_{CE} = 50V T_{amb} = 70°C, V_{CE} = 50V T_{amb} = 70°C for ULN2802A V_{CE} = 50V, V_i = 6V for ULN2804A V_{CE} = 50V, V_i = 1V			50 100 500 500	µA µA µA µA	1a 1a 1b 1b
$V_{CE(sat)}$	Collector-emitter Saturation Voltage	I_C = 100mA, I_B = 250µA I_C = 200mA, I_B = 350µA I_C = 350mA, I_B = 500µA		0.9 1.1 1.3	1.1 1.3 1.6	V V V	2
$I_{i(on)}$	Input Current	for ULN2802A V_i = 17V for ULN2803A V_i = 3.85V for ULN2804A V_i = 5V V_i = 12V for ULN2805A V_i = 3V		0.82 0.93 0.35 1 1.5	1.25 1.35 0.5 1.45 2.4	mA mA mA mA mA	3
$I_{i(off)}$	Input Current	T_{amb} = 70°C, I_C = 500µA	50	65		µA	4
$V_{i(on)}$	Input Voltage	V_{CE} = 2 V for ULN2802A I_C = 300mA for ULN2803A I_C = 200mA I_C = 250mA I_C = 300mA for ULN2804A I_C = 125mA I_C = 200mA I_C = 275mA I_C = 350mA for ULN2805A I_C = 350mA			 13 2.4 2.7 3 5 6 7 8 2.4	 V V V V V V V V V	5
h_{FE}	DC Forward Current Gain	for ULN2801A V_{CE} = 2V, I_C = 350mA	1000			–	2
C_i	Input Capacitance			15	25	pF	–
t_{PLH}	Turn-on Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	–
t_{PHL}	Turn-off Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	–
I_R	Clamp Diode Leakage Current	V_R = 50V T_{amb} = 70°C, V_R = 50V			50 100	µA µA	6 6
V_F	Clamp Diode Forward Voltage	I_F = 350mA		1.7	2	V	7

TEST CIRCUITS

Figure 1a.

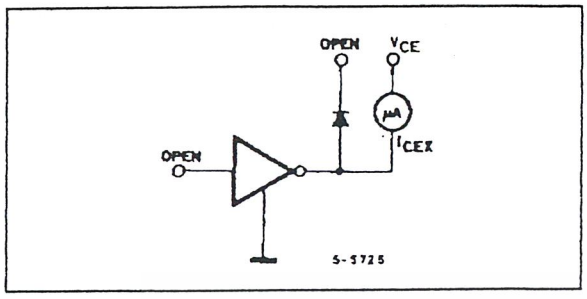


Figure 1b.

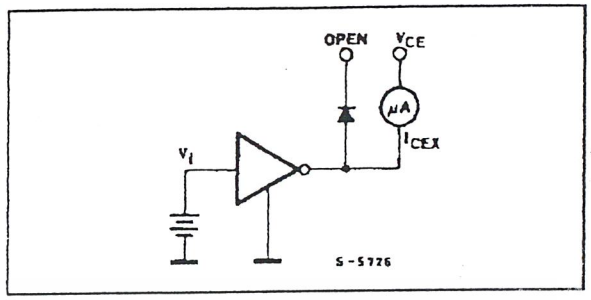


Figure 2.

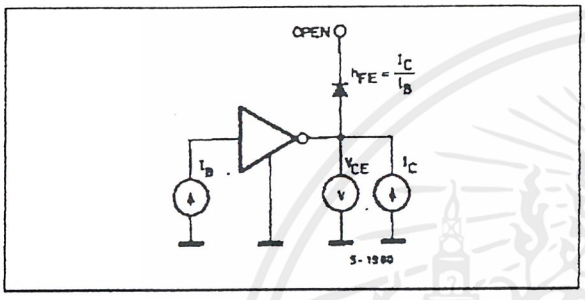


Figure 3.

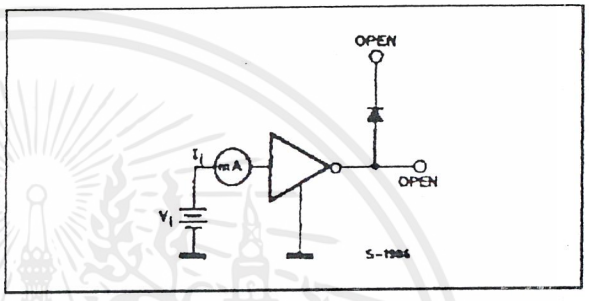


Figure 4.

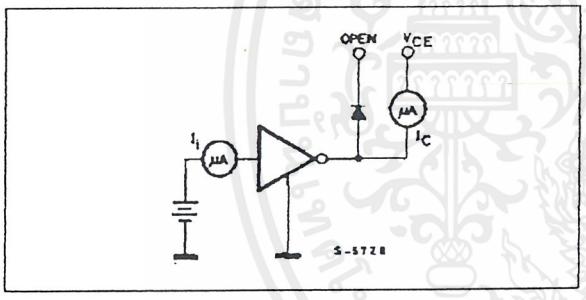


Figure 5.

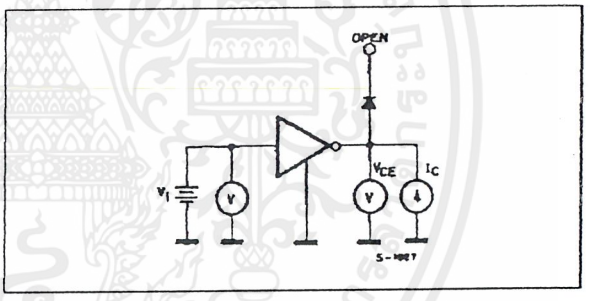


Figure 6.

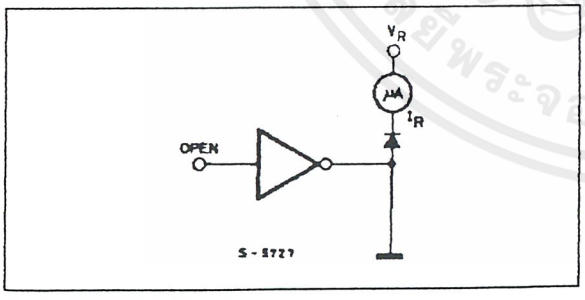


Figure 7.

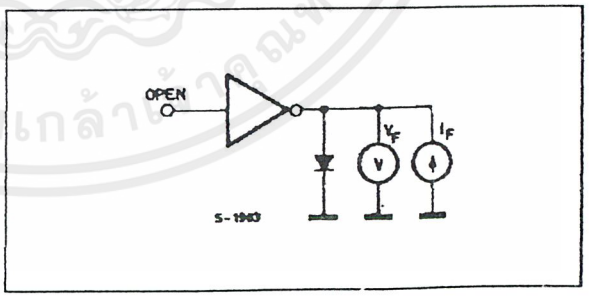


Figure 8 : Collector Current as a Function of Saturation Voltage.

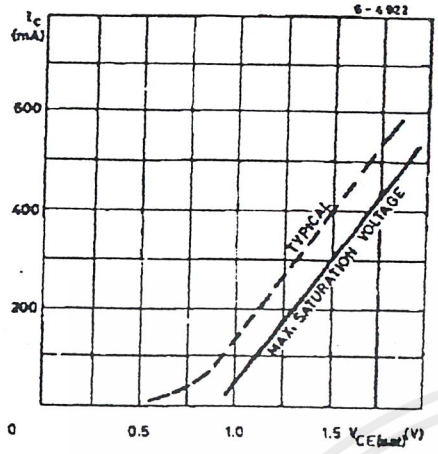


Figure 9 : Collector Current as a Function of Input Current.

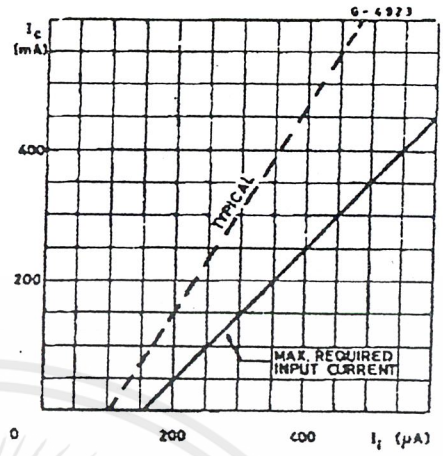


Figure 10 : Allowable Average Power Dissipation as a Function of Ambient Temperature.

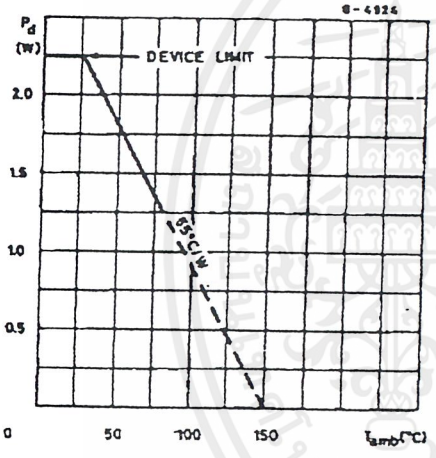


Figure 11 : Peak Collector Current as a Function of Duty Cycle.

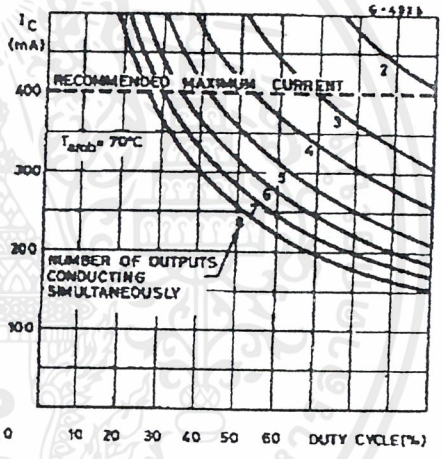


Figure 12 : Peak Collector Current as a Function of Duty.

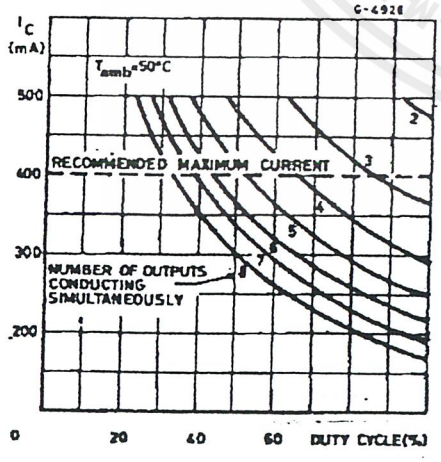


Figure 13 : Input Current as a Function of Input Voltage (for ULN2802A).

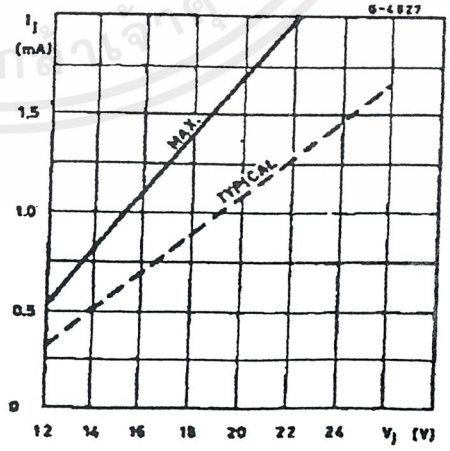


Figure 14 : Input Current as a Function of Input Voltage (for ULN2804A)

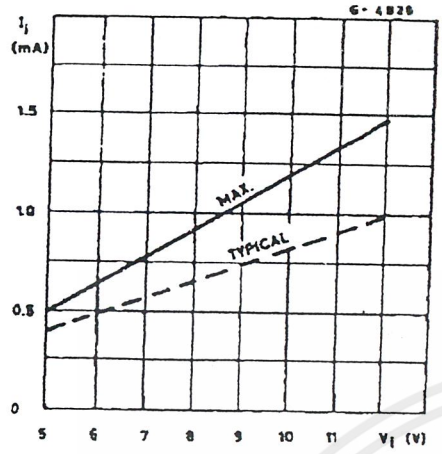


Figure 15 : Input Current as a Function of Input Voltage (for ULN2803A)

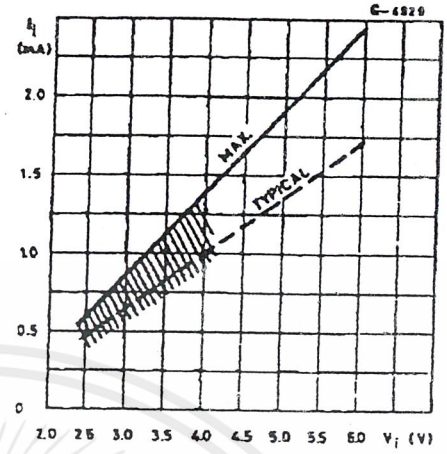
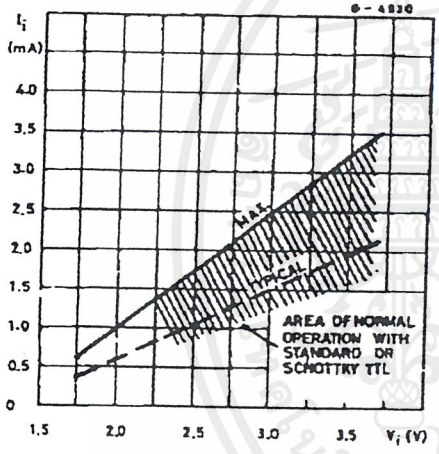


Figure 16 : Input Current as a Function of Input Voltage (for ULN2805A)



DIP18 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.254			0.010		
B	1.39		1.65	0.055		0.065
b		0.46			0.018	
b1		0.25			0.010	
D			23.24			0.915
E		8.5			0.335	
e		2.54			0.100	
e3		20.32			0.800	
F			7.1			0.280
I			3.93			0.155
L		3.3			0.130	
Z		1.27	1.59		0.050	0.063