ชุดพัฒนาไมโครคอนโทรลเลอร์ MCS – 51 โดยใช้ไอซี P89C51RD2

(Microcontroller Development Tool MCS – 51 by P89C51RD2)

จัดทำโดย

นายสกุล นิมมานนท์ รหัสนักศึกษา 41014436

นายอภิชาต ตั้งตรงสุนทร รหัสนักศึกษา 41014514

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ชุดพัฒนาไมโครคอนโทรลเลอร์ MCS – 51 โดยใช้ไอซี P89C51RD2

(Microcontroller Development Tool MCS – 51 by P89C51RD2)

จัดทำโดย

นายสกุล นิมมานนท์ รหัสนักศึกษา 41014436

นายอภิชาต ตั้งตรงสุนทร รหัสนักศึกษา 41014514

อาจารย์ที่ปรึกษา

อาจารย์เกียรติวรรณ ทรงสัตย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดพัฒนาไมโครคอนโทรลเลอร์ MCS- 51 โดยใช้ ไอซี P89C51RD2

(Microcontroller Development Tool MCS-51 by P89C51RD2)

จัดทำโดย

1. นายสกุล นิมมานนท์
2. นายอภิชาต ตั้งตรงสุนทร

..... อาจารย์ที่ปรึกษา

(อาจารย์เกียรติวรรณ ทรงสัตย์)

ชุดพัฒนาไมโครคอนโทรลเลอร์ MCS-51 โดยใช้ไอซี P89C51RD2

(Microcontroller Development Tool MCS-51 by P89C51RD2)

นายสกุล นิมมานนท์
นายอภิชาต ตั้งตรงสุนทร
อาจารย์ที่ปรึกษา
อาจารย์ เกียรติวรรณ ทรงสัตย์
ปีการศึกษา 2544

บทคัดย่อ

อินซิสเตมโปรแกรมมิง (In-System Programming) เป็นการพัฒนาการบรรจุโปรแกรมให้แก่ไมโครคอนโทรลเลอร์ที่มีหน่วยความจำภายในเป็นแฟลชเมโมรี (Flash Memory) โดยทำการส่งโปรแกรมจากคอมพิวเตอร์ ผ่านทางพอร์ตอนุกรม (Serial Port) ไปยังตัวไมโครคอนโทรลเลอร์ที่มีอินซิสเตมโปรแกรมมิง อยู่ภายในได้ทันที โครงงานนี้ได้ทำการสร้างอุปกรณ์ที่สนับสนุนการบรรจุโปรแกรมแบบ อินซิสเตมโปรแกรมมิง ซึ่งประกอบไปด้วยบอร์ดที่เสมือนเป็นไมโครคอนโทรลเลอร์ตัวหนึ่งซึ่งรับการป้อนข้อมูลจากพอร์ตอนุกรมของคอมพิวเตอร์ แล้วทำหน้าที่ควบคุมการทำงานของอุปกรณ์ต่างๆ โดยผู้ใช้งานสามารถทำการส่งข้อมูลจากคอมพิวเตอร์มายังบอร์ดได้บนระบบปฏิบัติการวินโดวส์ (Windows) จากซอฟต์แวร์ (Software) ที่ได้จัดทำขึ้น นอกจากนี้ยังมีส่วนสนับสนุนในการเขียนโปรแกรมและตรวจสอบโปรแกรม อันประกอบไปด้วย ส่วนของ อีดิเตอร์ (Editor) , คอมไพเลอร์ (Compiler) และ ซีมูเลเตอร์ (Simulator) ที่รวมกันอยู่เป็นชุดการใช้งาน เพื่อให้ใช้งานได้สะดวกยิ่งขึ้น

ABSTRACT

In-System Programming is the development that use for program microcontroller which has Flash Memory by send program from computer through Serial Port to microcontroller. This project presents tools that support for In-System Programming consist of Board which operate like microcontroller for control Target Board and software that can operate in Windows for send program to microcontroller. Besides , there is software that support for write program and test program consist of Editor , Compiler and Simulator . All of software are included in package for easy to use.

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

ในอดีตการจะนำโปรแกรมมาบรรจุเข้าไปในหน่วยความจำแบบ อีพรอม (Eprom) ถือเป็นเรื่องที่ยุ่งยากพอสมควร เพราะต้องใช้เครื่องโปรแกรมที่มีราคาสูง และยังต้องใช้เวลาในการโปรแกรมที่นานเมื่อจะทำการทดลองโปรแกรมหลาย ๆ ครั้ง ขบวนการเหล่านี้ก็กลายเป็นเรื่องที่ใช้เวลาอยู่มาก คือต้องบรรจุโปรแกรม แล้วล้างและทำซ้ำ ๆ ไปเรื่อย ๆ จนกว่าจะได้โปรแกรมที่สมบูรณ์ สมัยต่อ ๆ มาจึงมีเครื่องมือประเภท แรม (Ram) 2 ทาง หรือประเภท อีพรอม อีมูเลเตอร์ (Eprom Emulator) เพื่อช่วยให้ขั้นตอนต่าง ๆ ทำได้ง่ายขึ้น

ต่อมาได้มีการพัฒนาความก้าวหน้าในด้าน แฟลชเมโมรี (Flash Memory) ซึ่งทำให้ขบวนการล้างทำได้รวดเร็วขึ้น กล่าวคือ เป็นการล้างทางไฟฟ้าเท่านั้น ไม่ต้องใช้แสงยูวี (Ultra Violet : UV) เหมือนการล้างตัวอีพรอม การล้างแฟลชเมโมรีจะใช้เวลาเพียงเล็กน้อย เรียกได้ว่าเพียงไม่กี่วินาทีในขณะที่การล้างด้วยแสงยูวีจะกินเวลา 3-5 นาที และยิ่งถ้าใช้ไปนาน ๆ เข้าการล้างอาจจะต้องใช้เวลาถึง 30 นาทีเลย นอกจากนั้นแฟลชเมโมรี ยังมีจำนวนครั้งของการใช้งานได้มากกว่า เริ่มต้นที่ 1,000 ครั้งจนปัจจุบันสามารถขึ้นไปได้ถึง 10,000 ครั้งในขณะที่ตัวอีพรอม ใช้ได้ถึง 20 ครั้งก็นับว่าดีมากแล้ว ด้วยคุณสมบัติทั้งปวงนี้ ถือได้ว่า แฟลชเมโมรี มีส่วนในการปฏิวัติวงการไมโครคอนโทรลเลอร์ให้พัฒนาขึ้น แต่ว่าข้อเสียของการใช้ไมโครคอนโทรลเลอร์แบบแฟลชเมโมรี คือ ในการบรรจุโปรแกรมให้แก่ ไมโครคอนโทรลเลอร์ที่ใช้แฟลชเมโมรีนั้นจะต้องนำตัวไมโครคอนโทรลเลอร์มาบรรจุโปรแกรมที่เครื่องโปรแกรมเท่านั้น ดังนั้นในปัจจุบันได้มีการพัฒนาขบวนการบรรจุโปรแกรมให้แก่ไมโครคอนโทรลเลอร์แบบ แฟลชเมโมรี ให้สะดวกยิ่งขึ้น ซึ่งก็คือ อินซิสเตมโปรแกรมมิ่ง หรือ ไอเอสพี (IN-SYSTEM PROGRAMMING : ISP) นั่นเอง

ไอเอสพี เป็นการส่งข้อมูลแบบอนุกรม เราใช้ ไอเอสพี ในการติดต่อไมโครคอนโทรลเลอร์กับตัวคอมพิวเตอร์ ผ่านทางพอร์ตอนุกรม (Serial Port) โดยตัวไมโครคอนโทรลเลอร์จะรับคำสั่งและข้อมูลต่าง ๆ จากตัว คอมพิวเตอร์ เช่น การบรรจุโปรแกรม , การลบโปรแกรม ใน แฟลชเมโมรีภายใน ของไมโครคอนโทรลเลอร์ เป็นต้น ผ่านทางขาสัญญาณ ลงบนตัวไอซีได้เลย โดยไม่ต้องแกะตัวไอซีที่บัดกรีลงบนบอร์ดแล้วมาบรรจุโปรแกรมใหม่ในเครื่องโปรแกรมแบบทั่ว ๆ ไป ที่เรียกกว่าการโปรแกรมแบบขนาน (Parallel) เราสามารถบรรจุโปรแกรมผ่านทางขาสัญญาณเพียงไม่กี่เส้น และใช้การส่งข้อมูลในแบบอนุกรม (Serial) แทน นั่นหมายความว่า ตัวไอซีที่ถูกบัดกรีลงไปในบอร์ดแล้ว ก็ยังสามารถถูกนำมาบรรจุโปรแกรมใหม่ได้ โดยขอให้มีขาสัญญานดังกล่าวต่อออกมาได้ จึงเรียกกว่าเป็น อินซิสเตมโปรแกรมมิ่ง นั่นเอง

ในปัจจุบันนี้มีไมโครคอนโทรลเลอร์มากมายที่ถูกทำการพัฒนาให้มีคุณสมบัติแบบ ไอเอสพี นี้ไม่ว่าจะเป็นตระกูล MCS - 51 หรือ PIC (Microchip) หรืออื่น ๆ อีกมากมาย ซึ่งในโครงงานนี้ได้เลือกใช้

คอนโทรลเลอร์ของ ฟิลิปส์ รุ่น P89C51RD2 ซึ่งใช้ขาสัญญาณในการโปรแกรม 2 เส้น โดยเป็น พอร์ตอนุกรม มาตรฐานนั่นเอง คือ P3.0 (RX) P3.1 (TX) การโปรแกรมใช้แรงดันไฟฟ้าเพียงแค่ 5 โวลท์ เท่านั้น และไอซีรุ่นนี้ยังมีจุดเด่นคือ สามารถทำงานได้เร็วกว่า 1 เท่าด้วยค่า Xtal ค่าเดิม กล่าวคือ 1 คำสั่งของ MCS-51 ทั่วไปจะใช้สัญญาณนาฬิกา 12 ลูก แต่ P89C51RD2 จะใช้เพียง 6 ลูก เท่านั้น การโปรแกรมผ่านทางพอร์ตอนุกรมมาตรฐานนี้ จะทำให้ขบวนการต่าง ๆ ทำได้ง่ายขึ้นมาก เพราะสามารถต่อโดยตรงเข้ากับเครื่อง คอมพิวเตอร์ ได้ทันที แต่ยังคงต้องผ่านไอซีปรับแรงไฟ นอกจากนี้ P89C51RD2 นี้ยังมีราคาที่ถูกกว่ารุ่นอื่น ๆ ที่มีคุณสมบัติ ไอเอสพีอีกด้วย

ในการพัฒนาการใช้ไมโครคอนโทรลเลอร์ให้ใช้ผู้ใช้สามารถได้อย่างสะดวกนั้น จะพัฒนาเพียงในส่วนของการโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์อย่างเดียวคงไม่พอ ต้องมีส่วนที่ให้ผู้ใช้สามารถเขียนโปรแกรมและทำการตรวจสอบโปรแกรมได้ ในโครงงานนี้จึงแบ่งได้ออกเป็น 4 ส่วนดังนี้

ส่วนของ อีดิเตอร์ (Editor) ให้ผู้ใช้งานสามารถเขียนโปรแกรมภาษาแอสเซมบลีให้อยู่ในรูปของเท็กซ์ไฟล์ (Text File) เพื่อใช้ในการควบคุมไมโครคอนโทรลเลอร์ในการใช้งาน

ส่วนของ แอสเซมเบลอร์ (Assembler) ซึ่งใช้ในการแปลงรหัส แอสเซมบลี (Assembly Code) ให้เป็นภาษาเครื่อง (Machine Code) ในรูปของ Intel Hex File เพื่อใช้ในการโปรแกรมไมโครคอนโทรลเลอร์

ส่วนของ ซีมูเลเตอร์ (Simulator) เพื่อให้ผู้ใช้งานสามารถตรวจสอบการทำงานของโปรแกรมที่เขียนไป ว่ามีความถูกต้องหรือไม่

ส่วนของ อินซิสเตมโปรแกรมมิ่ง (In-System Programming) ให้ผู้ใช้ทำการโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ ผ่านพอร์ตอนุกรม จากตัวคอมพิวเตอร์

โดยทั้ง 4 ส่วนที่กล่าวมานี้ได้จัดทำให้อยู่รวมกันเป็นชุด (Package) ซึ่งสามารถปฏิบัติงานได้บนระบบปฏิบัติงานวินโดวส์ (Windows) เพื่อให้ผู้ใช้สามารถนำไปใช้งานได้อย่างสะดวกและรวดเร็ว

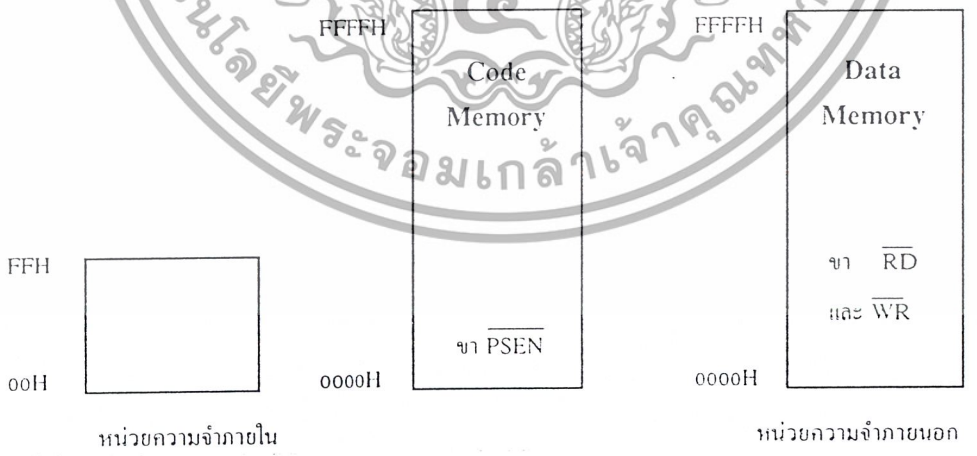
บทที่ 2
ทฤษฎี และ หลักการ

2.1 ทฤษฎีของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ P89C51RD2 เป็นไมโครคอนโทรลเลอร์ที่มีหลักการทำงานเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 ดั้งนั้นจึงขอนำเสนอหลักการทำงานของ MCS-51 บางประการที่สำคัญ และใช้ในโครงงาน ดังนี้

2.1.1 โครงสร้างหน่วยความจำ

หน่วยความจำสำหรับไมโครคอนโทรลเลอร์ จะมี 2 ชนิดคือ หน่วยความจำที่ใช้เก็บโปรแกรม หรือ รอม (ROM) กับหน่วยความจำที่ใช้เก็บข้อมูลในการประมวลผล หรือ แรม (RAM) ไมโครคอนโทรลเลอร์บางเบอร์เช่น 8051 , 8052 จะมีหน่วยความจำโปรแกรมภายในไอซี และ ไมโครคอนโทรลเลอร์ ทุกเบอร์สามารถอ้างหน่วยความจำโปรแกรมภายนอกได้มากที่สุด 64 กิโลไบต์ สำหรับหน่วยความจำแรมภายใน จะประกอบไปด้วยพื้นที่ใช้งานทั่วไป , รีจีสเตอร์แบงค์ ,พื้นที่ใช้งานระดับบิต และรีจีสเตอร์ฟังก์ชันพิเศษ เราอาจเขียนไดอะแกรมของหน่วยความจำของ 8031 ได้ดังรูปที่ 2.1 โดยในรูปจะบอกด้วยว่าขาใดจะแอคทีฟ



รูปที่ 2.1 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์

ใน 8031 จะมีหน่วยความจำภายในตั้งแต่ตำแหน่ง 00H ถึง FFH และสามารถอ้างหน่วยความจำโปรแกรมภายนอกได้ 64 กิโลไบต์ ตำแหน่ง ถ้าอ่านข้อมูลจากหน่วยความจำโปรแกรมขา $\overline{\text{PSEN}}$ จะแอคทีฟ นอกจานี้ 8031 สามารถ อ้างหน่วยความจำข้อมูลภายนอกได้ 64 กิโลไบต์ ตำแหน่ง โดยการติดต่อกับหน่วยความจำนี้ ขา $\overline{\text{RD}}$ และ $\overline{\text{WR}}$ จะแอคทีฟ สำหับหน่วยความจำข้อมูลภายในนั้นจะแบ่งออกได้ดังนี้

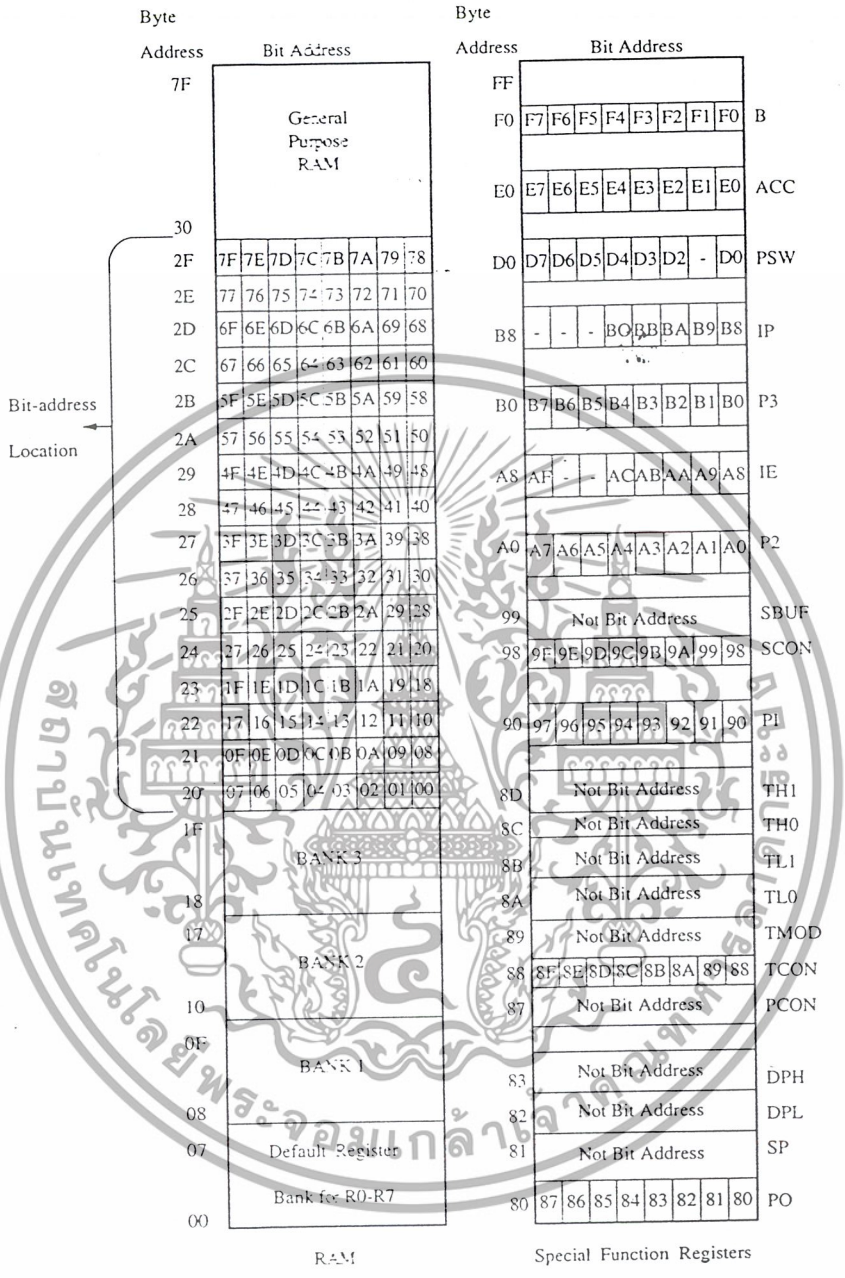
1) ชุดรีจีสเตอร์ 4 ชุด แต่ละชุดเรียกว่ารีจีสเตอร์แบงค์ ที่ตำแหน่ง 00H ถึง 1FH โดยแต่ละชุดประกอบด้วยรีจีสเตอร์ R0 ถึง R7

2) หน่วยความจำที่สามารถเข้าถึงข้อมูลระดับบิตได้ ตำแหน่ง 20H ถึง 2FH

3) หน่วยความจำใช้งานทั่วไปตำแหน่ง 30H ถึง 7FH

4) รีจีสเตอร์ ฟังก์ชันพิเศษ ตำแหน่ง 80H ถึง FFH

แผนผังการจัดหน่วยความจำข้อมูลภายในแสดงได้ดังรูปที่ 2.2 จากแผนผังจะเห็นว่าการอ้างตำแหน่ง หน่วยความจำภายในจะอ้างได้สองแบบ คือ การอ้างไปที่ตำแหน่งของไบต์ (เขียนหมายเลขตำแหน่งด้านนอก) หรือการอ้างไปที่ตำแหน่งของบิต (เขียนหมายเลขตำแหน่งด้านใน) โดยตำแหน่งของหน่วยความจำที่อ้างเป็นแบบบิตได้จะมีตำแหน่งบิตที่แน่นอน



รูปที่ 2.2 ตำแหน่งของหน่วยความจำทั้งแบบไบต์และแบบบิต

2.1.2 รีจีสเตอร์ฟังก์ชันพิเศษ (Spacial Function Rangister)

ใน ไมโครคอนโทรลเลอร์ รีจีสเตอร์จะใช้หน่วยความจำ แรม ภายในไอซี โดยส่วนหนึ่งเป็น รีจีสเตอร์พิเศษ (Spacial Function Register : SFR) ซึ่งมีทั้งหมด 21 ตัว โดยรีจีสเตอร์พิเศษต่าง ๆ จะเริ่มที่หน่วยความจำตั้งแต่ 80H ถึง FFH ซึ่งมีทั้งหมด 128 ตำแหน่ง แต่จะเป็นรีจีสเตอร์ฟังก์ชันพิเศษ เพียง 21 ตำแหน่ง แต่ถ้าเป็น 8032 / 8051 จะใช้ 26 ตำแหน่งหรือมี รีจิสเตอร์พิเศษ 26 ตัว

2.2 คุณสมบัติพิเศษของP89C51RD2 8-bit Flash microcontroller family

ไมโครคอนโทรลเลอร์รุ่น P89C51RD2 เป็นไมโครคอนโทรลเลอร์ที่อยู่ในตระกูลของ MCS-51 เหมือนไมโครคอนโทรลเลอร์ทั่ว ๆ ไป แต่จะมีคุณสมบัติพิเศษซึ่งต่างจากไมโครคอนโทรลเลอร์ตัวอื่น ๆ ก็คือ สามารถทำการโปรแกรมแบบ อินซิสเต็ม โปรแกรมมิ่ง ได้นอกจากนั้นยังมีข้อดีอื่นๆที่เหนือกว่าไมโครคอนโทรลเลอร์อื่น ๆ อีก เช่น

2.2.1. เลือกการทำงานได้ 2 แบบ คือ แบบ 6 สัญญาณนาฬิกา และ 12 สัญญาณนาฬิกา

P89C51RD2 สามารถเลือกที่จะทำงานเร็วขึ้นอีก 1 เท่าด้วย Xtal ค่าเดิม กล่าวคือ สามารถเลือกได้ว่าจะทำงานในโหมด 12 สัญญาณนาฬิกา หรือ 6 สัญญาณนาฬิกา ถ้าเลือก 12 สัญญาณนาฬิกา การทำคำสั่ง 1 คำสั่งก็จะใช้ 12 สัญญาณนาฬิกา ตามปกติ ถ้าเลือกโหมด 6 สัญญาณนาฬิกา การทำคำสั่ง 1 คำสั่งก็จะใช้เพียง 6 สัญญาณนาฬิกา เท่านั้น โดยปกติทางโรงงานจะกำหนดให้ไอซีทำงานในโหมด 6 สัญญาณนาฬิกา มาอยู่แล้ว แต่ผู้ใช้ยังคงสามารถโปรแกรมให้ทำงานแบบ 12 สัญญาณนาฬิกา ตามเดิมก็ได้ เพื่อที่จะนำมาใช้ กับเครื่องโปรแกรมแบบขนาน (Parallel) และที่สำคัญคือ เมื่อเราเปลี่ยนให้กลายเป็นการโปรแกรมให้เป็นแบบ 12 สัญญาณนาฬิกา จะไม่สามารถกำหนดย้อนกลับไปเป็นแบบ 6 สัญญาณนาฬิกา ได้อีก เพราะฉะนั้น ต้องตัดสินใจเลือกให้เด็ดขาดก่อน

2.2.2 มี ONCE™ Mode

ONCE ("On-Circuit Emulation") โหมดเป็นโหมดที่ให้การสนับสนุนในการทดสอบระบบโดยที่ไม่ต้องถอดตัวไอซีออกมาจากวงจร ในการใช้ ONCE โหมดต้องกำหนดค่าต่างๆดังนี้

1. ให้ ALE เป็น LOW ขณะที่กำลัง รีเซ็ต วงจร และให้ PSEN เป็น HIGH
2. คงค่า ALE ให้เป็น LOW ไว้จน RST สิ้นสุดลง

ขณะที่วงจรอยู่ใน ONCE โหมด เราสามารถใช้ อีมูเลเตอร์ ในการขับวงจร การกลับสู่การทำงานปกติทำได้โดยการ รีเซ็ต

2.2.3 มี แหล่งกำเนิดอินเทอร์รัปต์ 7 แหล่ง 4 ระดับ (Interrupt Priority Structure 7 source 4 level)

ปกติในไมโครคอนโทรลเลอร์ 8051 ทั่วไป จะมี 2 ระดับอินเทอร์รัปต์ แต่ในไมโครคอนโทรลเลอร์ เบอร์ P89C1RD2 จะมีถึง 4 ระดับอินเทอร์รัปต์ โดยจะมี รีจิสเตอร์ฟังก์ชั่นพิเศษ 3 ตัวที่เป็นตัวสร้าง อินเทอร์รัปต์ คือ IE , IP และ IPH โดย IPH จะอยู่ที่ตำแหน่ง B7H ของ รีจิสเตอร์ฟังก์ชั่นพิเศษ โดยการกำหนดค่า ระดับความสำคัญ จะกำหนดโดยค่าใน รีจิสเตอร์ฟังก์ชั่นพิเศษ IPH กับ รีจิสเตอร์ฟังก์ชั่นพิเศษ IP ดังตารางที่ 2.1

PRIORITY BITS		INTERRUPT PROIRITY LEVEL
IPH.x	IP.x	
0	0	Level 0 (lowest priority)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (Highest priority)

ตารางที่ 2.1 แสดงค่าการตั้งค่าระดับความสำคัญ ของ อินเทอร์รัปต์

2.2.4 มี ดาต้าพอยน์เตอร์ (DPTR) 2 ตัว

ใน P89C51RD2 มี ดาต้าพอยน์เตอร์ 2 ตัว โดยเราสามารถเลือกว่าจะใช้ตัวใดได้โดย เลือกเซ็ตบิต DPS ในรีจิสเตอร์ฟังก์ชั่นพิเศษ AUXR1 (ตำแหน่ง A2H) โดย

เลือก DPTR0 ; DPS = 0

เลือก DPTR1 ; DPS = 1

2.2.5 แฟลช อีพรอม เมโมรี (FLASH EPROM MEMORY)

ในไมโครคอนโทรลเลอร์ P89C51RD2 มีหน่วยความจำ แฟลชเมโมรี 64 กิโลไบต์ ซึ่งแบ่งเป็น 5 บล็อก โดยสองบล็อกแรกจะมีขนาด 8 กิโลไบต์ โดยมีแอดเดรสตั้งแต่ 0000F ถึง 3FFFH และอีก 3 บล็อกที่เหลือมีขนาด 16 กิโลไบต์ โดยมีแอดเดรสตั้งแต่ 4000H ถึง FFFFH

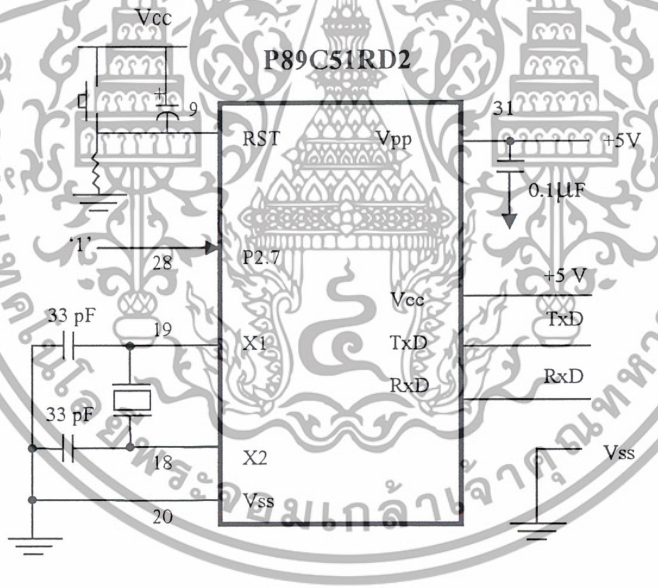
บู๊ตรอม (Boot ROM)เป็น แฟลชเมโมรี ที่มีขนาด 1 กิโลไบต์ อยู่ที่แอดเดรส FC00H ถึง FFFFH ซึ่ง บู๊ตรอม สามารถปิด (ไม่นำมาใช้งาน) ได้ ดังนั้นเมื่อเราไม่ได้ใช้งาน บู๊ตรอมแฟลชเมโมรี (Boot ROM Flash memory) 1 กิโลไบต์บนก็จะนำไปใช้เป็น แฟลชเมโมรี ได้ตามปกติ

รูปที่ 2.3 แสดงแผนภาพ แฟลชเมโมรีของ P89C51RD2

2.3 อินซิสเตม โปรแกรมมิ่ง : ไอเอสพี (IN – SYSTEM PROGRAMMING : ISP)

เป็นการส่งข้อมูลแบบอนุกรม เราใช้ ไอเอสพี ในการติดต่อไมโครคอนโทรลเลอร์กับตัวคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม โดยตัวไมโครคอนโทรลเลอร์จะรับคำสั่งและข้อมูลต่าง ๆ จากคอมพิวเตอร์ เช่น การบรรจุโปรแกรม , การลบโปรแกรม ในแฟรชเมโมรีของไมโครคอนโทรลเลอร์เป็นต้น ผ่านทางขาสัญญาณ ลงบนตัวไอซีได้เลย โดยไม่ต้องแกะตัวไอซีที่บัดกรีลงบนบอร์ดแล้วมาบรรจุโปรแกรมใหม่

ในไอซีไมโครคอนโทรลเลอร์ของฟิลิปส์ จะมี รอม จำนวน 1 กิโลไบต์ ซึ่งอยู่ด้านบนสุดของหน่วยความจำ อยู่ในตำแหน่ง FC00 ถึง FFFF ของไมโครคอนโทรลเลอร์ P89C51RD2 (มีแฟรชเมโมรี 64 กิโลไบต์) รอม 1 กิโลไบต์นี้เรียกว่า "บู๊ตรอม (BOOT ROM) " ซึ่งบู๊ตรอมนี้จะเป็นตัวเก็บคำสั่งในการบรรจุโปรแกรมและลบโปรแกรมในแฟรชเมโมรี ของไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรม



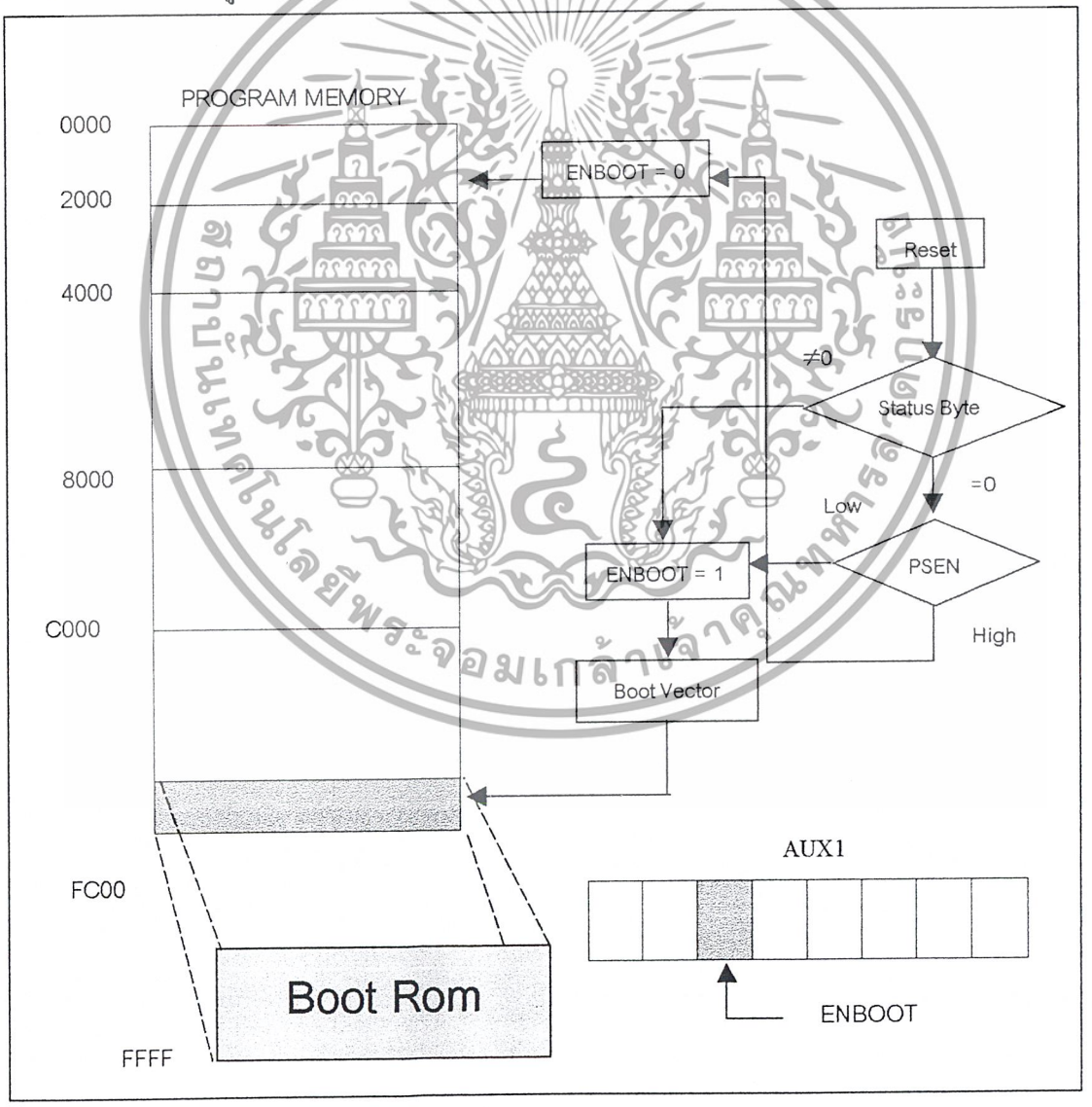
รูปที่ 2.4 การต่อขาต่าง ๆ ของ P89C51RD2 ในการใช้งานแบบ ไอเอสพี

ในการที่เราจะเข้าใจการทำงานของ ไอเอสพี ขั้นแรกเราต้องทำความเข้าใจกับ แฟรชรีจิสเตอร์พิเศษ (special Flash register) 2 ตัวเสียก่อน ซึ่งก็คือ บู๊ตเวคเตอร์ (BOOT VECTOR)และไบต์สถานะ (STATUS BYTE) เมื่อเกิดการรีเซ็ต ไมโครคอนโทรลเลอร์จะทำการตรวจสอบไบต์สถานะ ถ้าไบต์สถานะ ถูกกำหนดเป็นศูนย์ ในการเริ่มทำงานครั้งต่อไปเมื่อจ่ายไฟ (power – up) จะเริ่มต้นที่ตำแหน่ง 0000H ซึ่งเป็นตำแหน่งเริ่มต้นปกติ แต่ถ้าไบต์สถานะ ถูกกำหนดเป็นค่าอื่นที่ไม่ใช่ศูนย์ ค่าของ บู๊ตเวคเตอร์ จะถูกนำมาใช้เป็นแอดเดรสไบต์สูงในการกำหนดตำแหน่งเริ่มต้นใน

การทำงานครั้งต่อไป โดยแอดเดรสไบต์ต่ำจะเป็นถูกกำหนดเป็น 00H โดยปกติทางโรงงานจะกำหนดค่า บู๊ตเวคเตอร์ ไว้เป็น 0FCH ซึ่งจะเป็นแอดเดรส 0FC00H ซึ่งเป็นตำแหน่งของ ไอเอสพี บู๊ตโหลดเดอร์ (ISP boot loader) หรือ บู๊ตรอม เราสามารถกำหนด บู๊ตโหลดเดอร์ ได้โดยกำหนดบู๊ตเวคเตอร์ นั่นเอง

ข้อควรทราบ : เวลาเราลบ ไบต์สถานะ หรือ บู๊ตเวคเตอร์ ทั้ง 2 ไบต์ต้องถูกลบในเวลาเดียวกัน และเมื่อลบแล้วเราจำเป็นต้องใส่ค่าของบู๊ตเวคเตอร์ และใส่ค่าที่เปลี่ยนใหม่ของไบต์สถานะ ด้วย

นอกจากนี้ บู๊ตโหลดเดอร์ ยังสามารถทำงานได้โดย ให้ขา PSEN เป็น LOW , P2.7 และ ALE เป็น HIGH (หรือปล่อยลอยไว้) ขณะที่เกิดการ รีเซ็ต ซึ่งจะให้ผลเหมือนกับการเซ็ตค่าใน ไบต์สถานะ ไว้ไม่ให้เป็นศูนย์ โฟลวชาร์ตแสดงการทำงานของ ไอเอสพี แสดงดังรูปที่ 2.5



รูปที่ 2.5 แสดงโฟลวชาร์ตการทำงานของ ไอเอสพี

การใช้ ไอเอสพี ถูกทำให้สำเร็จโดย ซีเรียล บู๊ตโหลดเดอร์ สับรูทีน (serial boot loader subroutines) ซึ่งอยู่ใน บู๊ตรอม ในการทำงานนั้นจะใช้ รหัสคำสั่งเลขฐานสิบหก (Intel hex records) ในการรับคำสั่งและข้อมูลจากตัว คอมพิวเตอร์

ในการบรรจุโปรแกรมแบบ ไอเอสพี เราสามารถทำได้ตามลำดับขั้นตอนดังนี้

1. กำหนดการใช้ ไอเอสพี โดยใช้วิธีใดวิธีหนึ่งตามการอธิบายข้างต้น (เช่น กำหนด ไบต์สถานะ ให้ไม่เท่ากับศูนย์ , ให้ PSEN เป็น LOW เป็นต้น)
2. ส่งตัวอักษร "U" จาก คอมพิวเตอร์ เข้าสู่ตัวไมโครคอนโทรลเลอร์เพื่อกำหนดอัตราการส่งข้อมูล (autobaud)
3. ส่งค่าที่จะใช้กำหนดค่าความถี่ (oscillator) จาก คอมพิวเตอร์ สู่ไมโครคอนโทรลเลอร์
4. ส่งค่าตำแหน่งที่ต้องการจะลบบล็อก จาก คอมพิวเตอร์ สู่ไมโครคอนโทรลเลอร์
5. ส่งข้อมูลที่ต้องการจะ โปรแกรม จาก คอมพิวเตอร์ สู่ไมโครคอนโทรลเลอร์
6. ส่งค่าที่ใช้ในการลบ ไบต์สถานะ และ บู๊ตเวคเตอร์ เมื่อกระบวนการ ไอเอสพี สิ้นสุดลง
7. ส่งค่าที่ต้องการจะ โปรแกรม บู๊ตเวคเตอร์ กลับไปยังค่า 0FCH ถ้ายังต้องการใช้งานแบบไอเอสพี ต่อไป
8. ส่งค่า 00H ไปยัง ไบต์สถานะ เพื่อให้โปรแกรมเริ่มทำงานที่แอดเดรส 0000H หลังจากการรีเซต

ในการส่งข้อมูลแบบ ไอเอสพี นั้น จะส่งในรูปของ รหัสคำสั่งเลขฐานสิบหก ซึ่งประกอบด้วยตัวรหัสแอสกี (ASCII) ซึ่งสรุปได้ดังนี้

:NNAAAARRDD..DDCC<crlf>

- "NN" แสดงถึงจำนวนของไบต์ข้อมูลในเรคคอร์ด (record)ในตัว P89C51RD2 จะรับได้ถึง 16 (10H)ไบต์ข้อมูล
- "AAAA" แสดงถึงค่าแอดเดรสของไบต์แรกใน เรคคอร์ด ถ้าใน เรคคอร์ด มีศูนย์ไบต์ ค่านี้จะถูกเซ็ตเป็น 0000
- "RR" แสดงถึงชนิดของ เรคคอร์ด เช่นค่า "00" หมายถึงเป็นข้อมูล ค่า "01" หมายถึง จบไฟล์ (end-of-file mark)
- "DD…DD" แสดงถึง ข้อมูล
- "CC" แสดงถึง ตัว เชคซัม (checksum)

2.4 มาตรฐานพอร์ตอนุกรมแบบ RS-232

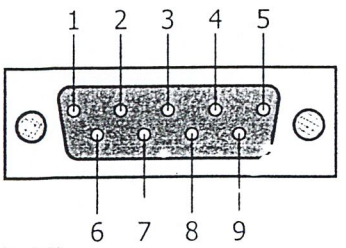
มาตราฐานการเชื่อมต่อแบบอนุกรม RS-232 แป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งออยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรม อิเล็กทรอนิกส์ (EIA) ได้วางมาตราฐานที่มีชื่อเรียกว่าEIA RS-232 มาตรฐาน นี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ –3 ถึง –12 V แสดงว่ามีข้อมูล และ +3 ถึง +12 V แสดงว่าเป็นช่องว่าง มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล(Data Terminal Equipment : DTE) กับ วงจรข้อมูลปลายทาง(Data Circuit Terminating : DCE) ได้ว่า อุปกรณ์ ดีทีอี (DTE) จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์ หรือ ไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ ดีซีอี (DCE) จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจากทาง ดีทีอี เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองผ่านทางมาตรฐาน RS-232

ข้อแตกต่างของอุปกรณ์ ดีทีอี และอุปกรณ์ ดีซีอี อย่างหนึ่งที่เห็นได้ชัดเจนคือ คอนเน็กเตอร์ของ ดีทีอี จะเป็นตัวผู้ สว่านคอนเน็กเตอร์ของ ดีซีอีจะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ ดีทีอี ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ ดีซีอี

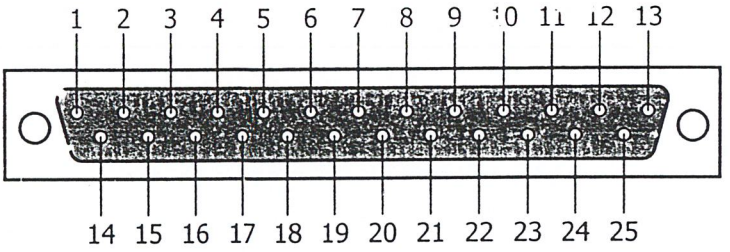
สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุดถึง 20 เมตร

2.4.1 คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้ หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 2.6



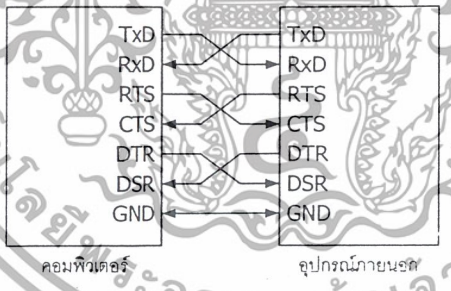
(ก) คอนเน็กเตอร์อนุกรม 9 ขา หรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)



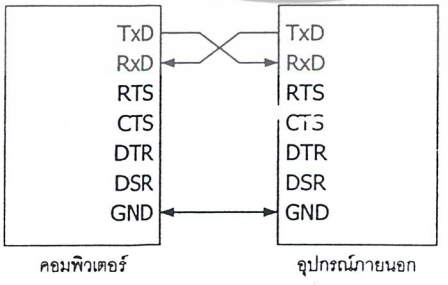
(ข) คอนเน็กเตอร์อนุกรม 25 ขา หรือแบบ DB-25 (มองจากด้านหลังคอมพิวเตอร์)

รูปที่ 2.6 การจัดขาของคอนเน็กเตอร์พอร์ตอนุกรม
ตามมาตรฐาน RS-232 ทั้งแบบ DB-9 และ DB-25

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกแสดงดังรูปที่ 2.7 ลูกศรในรูแสดงถึงทิศทางของข้อมูล ในรูที่ 2 (ก) เป็นการเชื่อแบบ นูลโมเดม (Null Modem) หรือการเชื่อมโดยตรงโดยไม่ต้องผ่าน โมเด็ม โดยมีการตรวจสอบหรือแฮนเช็กเติมรูปแบบ ส่วนในรูปที่ 2(ข) เป็นการเชื่อมต่อแบบ นูลโมเด็ม ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์ สำหรับรายละเอียดหน้าที่การทำงานในแต่ละขาของพอร์ตอนุกรม RS-232 มีดังนี้



(ก) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์
แบบ Null modem



(ข) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์
แบบ RS-232 โดยใช้สายสัญญาณเพียง 3 เส้น

รูปที่ 2.7 การต่อปุกรณ์ภายนอกกับพอร์ตอนุกรมของคอมพิวเตอร์ในลักษณะต่างๆ

1) ดาต้า แคเรีย ดีเทค (Data Carrier Detect : DCD) หรืออาจเรียกว่า แคเรีย ดีเทค (CarrierDetect :CD) ขานี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก

2) ขารับข้อมูล (Receive Data : RD) หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้เก็บไว้ใน รีจิสเตอร์บัฟเฟอร์

3) ขาส่งข้อมูล (Transmitted Data : TD) หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

4) ดาต้า เทอร์มินอล เรดดี้ (Data Terminal Ready :DTR) เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และ ขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อเป็นแบบ นูลโมเด็ม ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์

5) ซิกแนล กราวด์ (Signal Ground : GND) ขากราวด์ของระบบ

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

7) รีเควส ทู เซนด์ (Request To Send : RTS) เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือ ขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ นูล โมเด็ม 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกันเพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

8) เคลียร์ ทู เซนด์ (Clear To Send : CTS) ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงพร้อมที่จะรับข้อมูลหรือไม่

9.) ริง อินดิเคเตอร์ (Ring Indicator : RI) ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ปกติในการสื่อสาร โดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็ม โปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

2.4.2 UART

UART ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั่นเอง สำหรับการสื่อสารแบบอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

หน้าที่หลักของ UART คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรู)แบบอนุกรมแบบอะซิงโครนัส แล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์ ยังแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์รับทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล(บอร์ดเรต), รูปแบบการส่งข้อมูล , ความผิดพลาดที่เกิดขึ้นระหว่างกราถ่ายทอดข้อมูล(ผิดพลาดจากพาริตี้, เฟรมข้อมูล, โอเวอร์รัน) เป็นต้น

ภายใน UART จะมีส่วนของวงจรสร้างบอดเรตแบบโปรแกรมได้ โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART เบอร์นี้จะมีบัฟเฟอร์สำหรับรับและส่งข้อมูลตำแหน่งเดียวกัน ทำให้การรับและส่งข้อมูลถูกจำกัดความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาทีเท่านั้น แต่ UART เบอร์นี้ก็คือว่าเป็นต้นแบบของ UART ที่ใช้ในคอมพิวเตอร์ โดยคอมพิวเตอร์ทุกๆรุ่นจะต้องสนับสนุนการทำงานตามรูปแบบของ UART เบอร์นี้

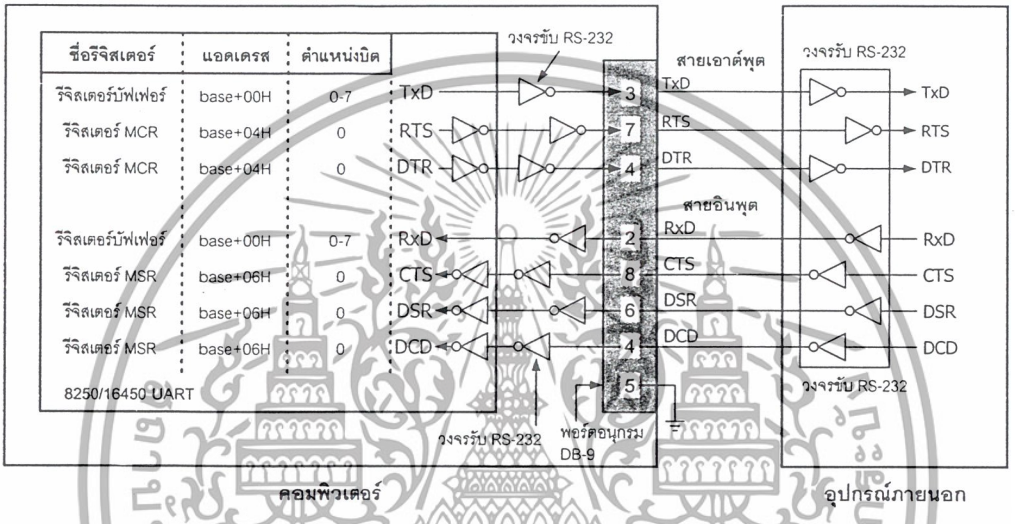
UART อีกเบอร์หนึ่งคือ 16450 มีความสามารถรับส่งข้อมูลได้ที่ความเร็ว115,200 บิตต่อวินาที และเพิ่มรีจิสเตอร์สำหรับพักข้อมูลสำหรับ UART นอกจากนั้นยังเพิ่มส่วนของชิฟต์ รีจิสเตอร์แบบFIFO (First In First Out) ขนาด 16 ไบต์เข้าไป ทำให้สามารถสนับสนุนความเร็ว ในการรับส่งข้อมูลที่ 256 กิโลบิตต่อวินาที โดยคอมพิวเตอร์ในปัจจุบัน ใช้ UART เบอร์นี้หรือใหม่กว่า เช่น เบอร์ TL16C750 ซึ่งมีรีจิสเตอร์แบบ FIFO ขนาด 64 ไบต์ ทำงานได้ที่ระดับแรงดัน +5 โวลต์ และ +3 โวลต์ มีโหมดประหยัดพลังงาน สามารถรับส่งข้อมูลได้ที่ความเร็ว 1 เมกะบิตต่อวินาทีเมื่อใช้สัญญาณนาฬิกา 16 เมกกะเฮิร์ต

อย่างไรก็ตาม ความเร็วในการส่งข้อมูลที่มากมายของ UART เบอร์ใหม่ๆก็ไม่ได้ช่วยให้การรับส่งข้อมูลของคอมพิวเตอร์เร็วขึ้น เนื่องจากว่าคอมพิวเตอร์ยังใช้ความถี่ของสัญญาณนาฬิกาในการแปลงข้อมลเพียง 1.8432 เมกะเฮิร์ต เท่านั้น

2.4.3 ลักษณะสัญญาณอินพุตและเอาต์พุตของพอร์ต RS-232

สัญญาณเอาต์พุตที่ใช้ควบคุม (RTS และ DTR) และสัญญาณแสดงสถานะอินพุต (CTS, DSR และ DCD)ของพอร์ตอนุกรม RS-232 จะถูกกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่งและรับจะไม่ถูกกลับสถานะ UART จะให้ระดับสัญญาณเอาต์พุตออกมาเป็นแบบทีทีแอลเท่านั้น ดังนั้นเมื่อสัญญาณถูกส่งออกมาจาก

UART จึงต้องส่งเข้าสู่วงจรขยายเพื่อปรับระดับแรงดันให้ได้ระดับสัญญาณเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกันเพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน แต่วงจรขับที่ใช้ทั้งภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้นจะถูกกลับสถานะ ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 ไดอะแกรมแสดงโครงสร้างทางฮาร์ดแวร์ของพอร์ตอนุกรม

แอดเดรสของพอร์ตอนุกรมมี 4 ตำแหน่งดังนี้

COM1 : 3F8H

COM2 : 2F8H

COM3 : 3E8H

COM4 : 2E8H

เมื่อเริ่มเปิดเครื่องเพื่อใช้งานคอมพิวเตอร์ ไบออสภายในคอมพิวเตอร์จะทำการตรวจสอบแอดเดรสของพอร์ตอนุกรมทั้งหมด ถ้าไบออสตราวจพบแอดเดรสของพอรตอนุกรมไบออสจะนำแอดเดรสที่ตรวจเจอไปเก็บไว้ในหน่วยความจำขนาด 2 ไบต์ สำหรับพอร์ตอนุกรม COM1 จะเก็บไว้ที่ แอดเดรส 0000:0400H และ 0000;040H ส่วนตำแหน่งอื่นๆมีรายละเอียดดังนี้

COM2 = 0000:0402H - 0000:0403H

COM3 = 0000:0404H - 0000:0405H

COM4 = 0000:0406H - 0000:0407H

นอกจากนี้ที่หน่วยความจำแอดเดรส 0000:0411H ยังใช้สำหรับแสดงจำนวนของพอร์ตอนุกรมที่มีใช้อยู่ในคอมพิวเตอร์อีกด้วย ดังตารางที่ 2.2

บิต 3	บิต 2	บิต 1	จำนวนพอร์ต
0	0	0	ไม่มีพอร์ตอนุกรม
0	0	1	มีพอร์ตอนุกรม 1 พอร์ต
0	1	0	มีพอร์ตอนุกรม 2 พอร์ต
0	1	1	มีพอร์ตอนุกรม 3 พอร์ต
1	0	0	มีพอร์ตอนุกรม 4 พอร์ต

ตารางที่ 2.2 แสดงข้อมูลในแอดเดรส 0000:0411H ที่ใช้แจ้งจำนวนพอร์ตอนุกรม

2.4.4 การส่งข้อมูลผ่านทาง พอร์ตอนุกรม ด้วยโปรแกรม วิชวลเบสิก

สำหรับการใช้งาน วิชวลเบสิกคัสตอมคอนโทรลสำหรับการสื่อสารอนุกรมผ่านทาง พอร์ตอนุกรมที่มีชื่อว่า MSCOMM32.OCX โดยคัสตอมตัวนี้ได้ถูกออกแบบมาให้ใช้งานกับระบบปฏิบัติการขนาด 32 บิต

MSComm จัดเตรียมทางเลือกไว้ 2 ทางเพื่อความสะดวกในการติดต่อสื่อสารข้อมูล ทางแรกคือ การสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำาหรับการตอบสนองแบบาทันทีทันใด เช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรมหรือเกิดการเปลี่ยนแปลงที่ขา ตรวจสอบการส่งข้อมูล หรือขา การร้องขาส่งข้อมูล เหตุการณ์ที่เกิดขึ้นขึ้นของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าต่างๆหลังจากให้โปรแกรมทำงานในฟังก์ชั่นต่างๆไปเรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก โดยคอนโทรล MSComm 1 ตัวสามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ตถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ตจะต้องใช้คอนโทรล

MSComm มากกว่า 1 ตัวเพื่อควบคุมพอร์ตอนุกรมในแต่ละพอร์ต แอดเดรสของพอร์ตอนุกรมและแอดเดรสของการเกิดอินเตอร์รัปต์สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ คอนโทรลพานอล (Control Panel) ถึงแม้ว่า คอนโทรล MSComm จะมีคุณสมบัติมากมาย แต่คำสั่งที่จำเป็นจะต้องนำมาใช้งานมีดังต่อไปนี้

CommPort

ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ติดต่อยู่ (COM1,COM2,COM3,COM4)

รูปแบบการใช้งาน

object .CommPort [=Value]

โดยค่า Value เป็นค่าของพอร์ตอนุกรมชนิดของข้อมูลเป็น จำนวนเต็ม ค่า Value สามารถกำนดได้ ในช่วง 1 – 16 (โดยค่าเริ่มต้นกำหนดไว้ที่ 1) เมื่อมีการกำหนดค่าแล้วทำการเปิดพอร์ตโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSComm จะสร้างสัญญาณแสดงข้อผิดพลาด error 68 ขึ้นมาซึ่งหมายถึงอุปกรณ์ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียน โปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่ใช้คำสั่ง OpenPort

Setting

ใช้ในการกำหนดและอ่านค่าอัตราบอด ,พาริตี้, จำนวนของบิตข้อมูล, จำนวนบิตปิดท้าย

รูปแบบการใช้งาน

object.Settings [= Value]

ค่า Value มีชนิดข้อมูลเป็นแบบ String มีรูปแบบเป็น “BBBB,P ,D,S” โดย “BBBB” เป็นค่าอัตราการบอด, P เป็นค่า พาริตี้ ,D เป็นจำนวนของบิตข้อมูล และ S เป็นจำนวนของบิตปิดท้าย ปกติแล้วค่านี้ถูกกำหนดไว้เป็น “9600,N,8,1”

ค่าบอดเรตมาตรฐานที่ใช้กับ MSComm มีดังนี้คือ

110 บิตต่อวินาที

300 บิตต่อวินาที

600 บิตต่อวินาที

1200 บิตต่อวินาที

2,400 บิตต่อวินาที

9,600 บิตต่อวินาที (ค่าปกติ)

14,400 บิตต่อวินาที

19,200 บิตต่อวินาที

28,800 บิตต่อวินาที

38,400 บิตต่อวินาที (สงวน)

56,000 บิตต่อวินาที (สงวน)

128,000 บิตต่อวินาที (สงวน)

256,000 บิตต่อวินาที (สงวน)

สำหรับค่ามาตรฐานในการกำหนดค่าพาริตี้บิต มีดังนี้

สัญลักษณ์	รายละเอียด
E	พาริตี้คู่ (Even)
M	ลอจิก "1" (Mark)
N	ไม่ใช้ (ค่าปกติ)
O	พาริตี้คี่ (Odd)
S	ลอจิก "0" (Space)

ค่าที่ใช้ในการกำหนดจำนวนบิตมี 5 ค่าคือ 4 , 5 , 6 ,7 ,8 (โดย 8 เป็นค่าปกติ)

ค่าที่ระบุจำนวนบิตปิดท้าย มี 3 ค่า คือ 1 , 1.5 และ 2 (โดย 1 เป็นค่าปกติ)

ตัวอย่างการใช้งานคำสั่ง Settings โดยจะเป็นการกำหนดค่าบอดเรตเท่ากับ 9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สามารถเขียนโปรแกรมได้ดังนี้

MSComm1.Settings = "9600 , N , 8 , 1"

หมายเหตุ สาเหตุที่ค่าที่กำหนดจะต้องอยู่ภายในเครื่องหมายคำพูด "" เนื่องจาก ค่าที่กำหนดนี้อยู่ในรูปตัวแปรสตริง String

PortOpen

ใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

รูปแบบการใช้งาน

object.PortOpen [= value]

ค่า Value มีชนิดข้อมูลเป็นแบบบูลีนคือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรม และ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm จะปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen จะต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ มิเช่นนั้น MSComm จะแสดงข้อผิดพลาด Error 68 แจ้งแก่ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดเอาไว้แล้ว โปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นเดียวกัน

ตัวอย่างการใช้คำสั่งเปิดพอร์ตอนุกรม

MSComm1.PortOpen = True

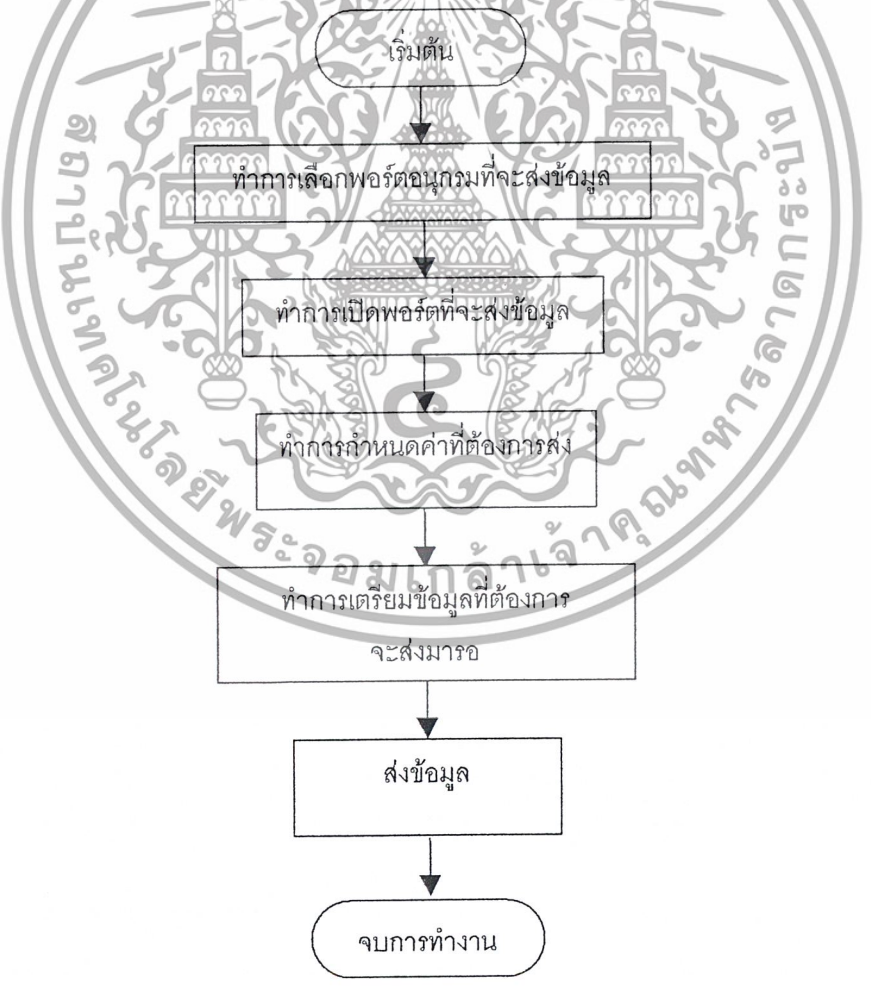
Output

ใช้ในการส่งขบวนของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

object.Output [= Value]

ค่า Value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลเป็นแบบ แวเรียนท์ (Varient) และมีข้อมูลภายในเป็นแบบ สตริง สำหรับการส่งข้อมูลไบนารี่จะต้องกำหนดชนิดของข้อมูลเป็นแบบ แวเรียนท์ และข้อมูลภายในเป็น ไบต์



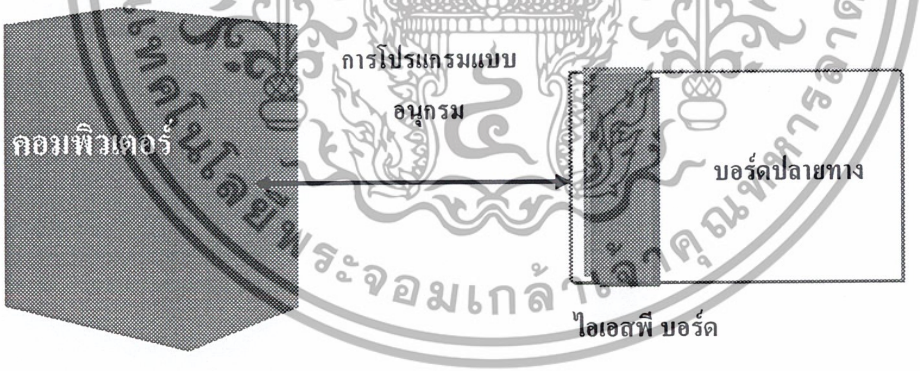
รูปที่ 2.9 โฟลวชาร์ตแสดงขั้นตอนการส่งข้อมูลผ่านทาง พอร์ตอนุกรม ด้วยโปรแกรม วิชวลเบสิก

บทที่ 3
การออกแบบ และ การสร้าง

ในโครงงานนี้ได้แบ่งการทำงานออกเป็น 2 ส่วน คือ ส่วนของฮาร์ดแวร์ (Hardware) และส่วนของซอฟต์แวร์ (Software) ซึ่งรายละเอียดมีดังต่อไปนี้

3.1 ส่วนของฮาร์ดแวร์

ในส่วนของฮาร์ดแวร์นั้นได้ทำ ไอเอสพีบอร์ด ซึ่งทำหน้าที่เป็นบอร์ดที่ใช้ในการควบคุมการทำงานของบอร์ดปลายทาง (Target Board) ซึ่งบน ไอเอสพีบอร์ด ประกอบไปด้วย ตัวไมโครคอนโทรลเลอร์ P89C51RD2 และ ไอซี MAX 232 ซึ่งเป็นไอซีที่เป็นบัฟเฟอร์ของการส่งข้อมูลแบบอนุกรม โดยบอร์ด ไอเอสพี นี้จะใช้ในการโปรแกรมแบบ ไอเอสพี ผ่านพอร์ตอนุกรมจากตัวคอมพิวเตอร์มายังตัวไมโครคอนโทรลเลอร์ P89C51RD2 ก่อนที่จะนำไปเสียบลงบนบอร์ดที่ใช้ในการทำงาน สามารถแสดงการทำงานของ ไอเอสพีบอร์ด ได้ดังรูปที่ 3.1



รูปที่ 3.1 แสดงการทำงานของ ไอเอสพีบอร์ด

รูปที่ 3.2 แสดงการต่อวงจรบนบอร์ด ไอเอสพี

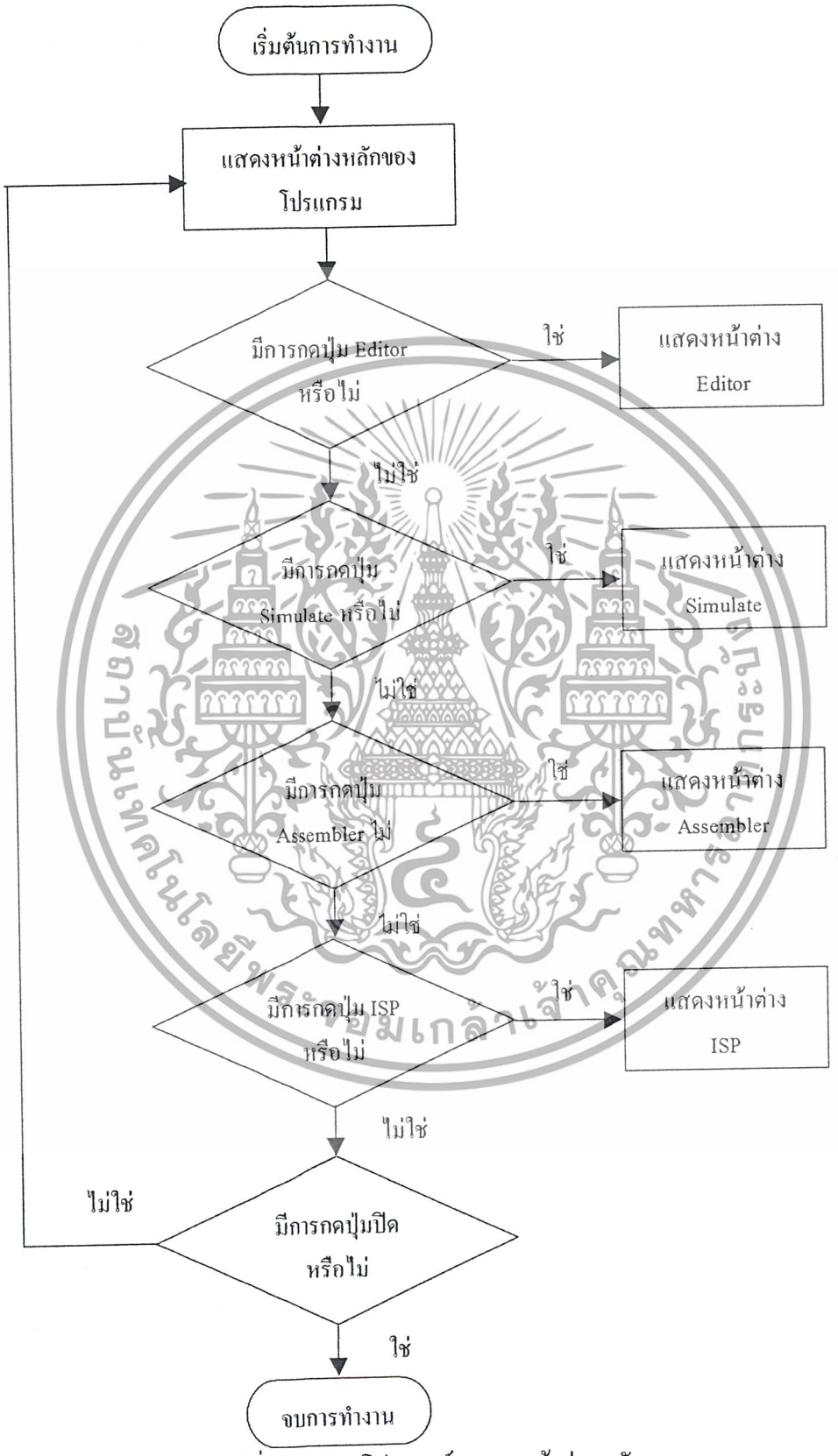
3.2 ส่วนของซอฟต์แวร์

ในส่วนของซอฟต์แวร์ของโครงงานนี้ได้จัดทำส่วนสนับสนุนในการส่งข้อมูลแบบ ไอเอสพี และส่วนสนับสนุนการเขียนโปรแกรม ควบคุมไมโครคอนโทรลเลอร์ โดยใช้โปรแกรม วิชวลเบสิก 6 (Visual Basic 6) ในการเขียนโปรแกรมในการเขียนโปรแกรมให้สามารถแสดงผลและติดต่อสื่อสาร ระหว่างผู้ใช้กับคอมพิวเตอร์บนระบบปฏิบัติการวินโดวส์ (Windows) ดังรูปที่ 3.3

3.2.1 ส่วนของหน้าต่างหลัก

ในส่วนของหน้าต่างหลัก จะประกอบไปด้วย เมนูคำสั่งต่าง ๆ เพื่อเรียกใช้การทำงานส่วนต่าง ๆ ซึ่งประกอบไปด้วย อีดิเตอร์ , แอสเซมเบลอร์ , ซีมูเลเตอร์ , และ ไอเอสพี ส่วนของหน้าต่างหลักมีรูปแบบดังนี้

รูปที่ 3.3 แสดงหน้าต่างหลักของโปรแกรม



รูปที่ 3.4 แสดงโฟลวชาร์ตของ หน้าต่างหลัก

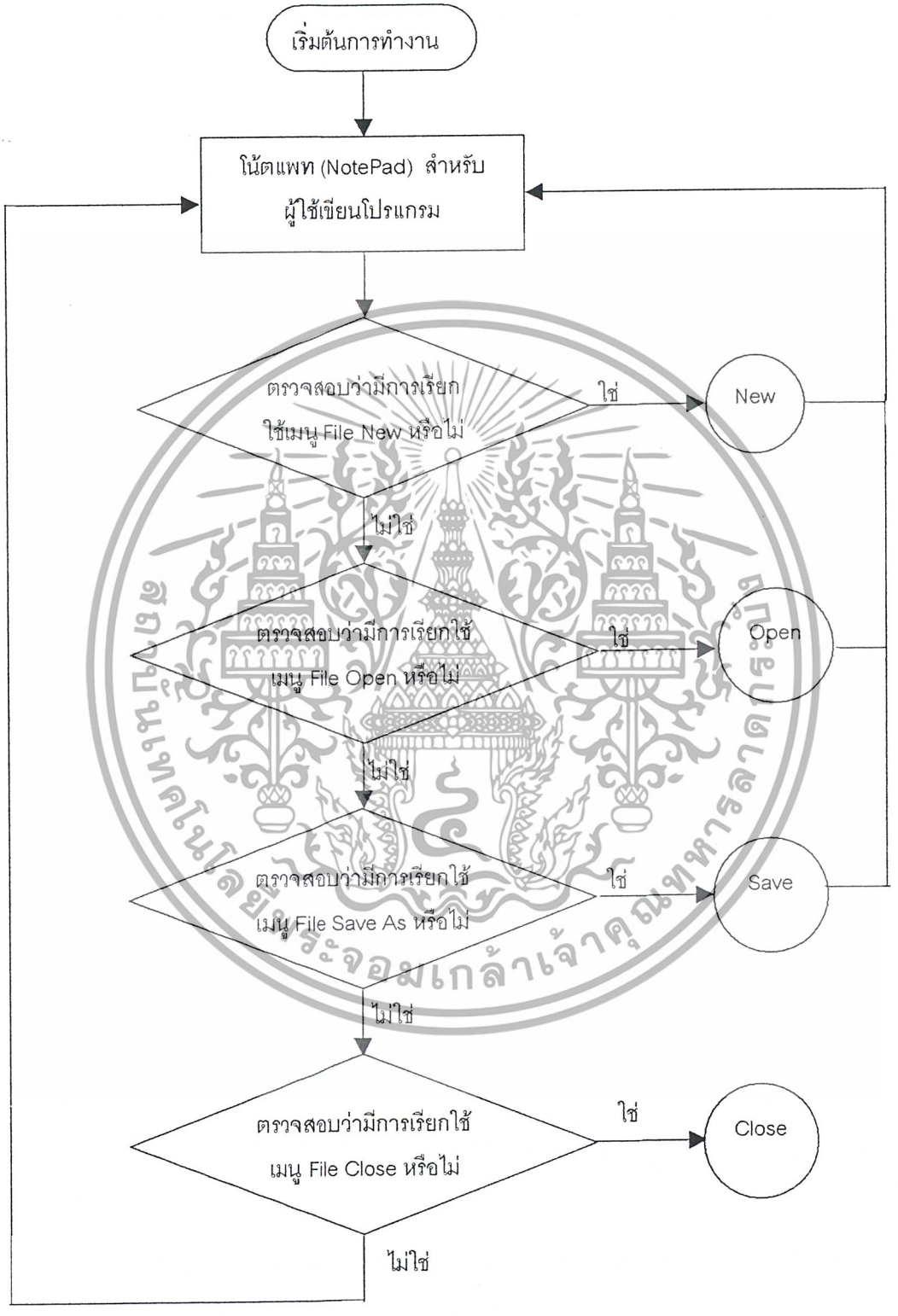
3.2.2 ส่วนหน้าต่างของอีดิเตอร์

ในส่วนหน้าต่างของ อีดิเตอร์ ประกอบด้วยเท็กซ์บ็อกซ์ (Text Box) ซึ่งอยู่ในรูป โน้ตแพท (NotePad) ไว้ให้ผู้ใช้เขียนโปรแกรม และมีเมนูไฟล์ต่าง ๆ ดังนี้

- เมนูไฟล์ New ใช้ในการสร้าง โน้ตแพทใหม่ขึ้นมาเพื่อให้ผู้ใช้เขียนโปรแกรมใหม่
- เมนูไฟล์ Open ใช้ในการเปิดไฟล์ที่มีอยู่แล้วในคอมพิวเตอร์
- เมนูไฟล์ Save As ใช้ในการบันทึกไฟล์
- เมนูไฟล์ Close ใช้ในการปิดการทำงานของอีดิเตอร์

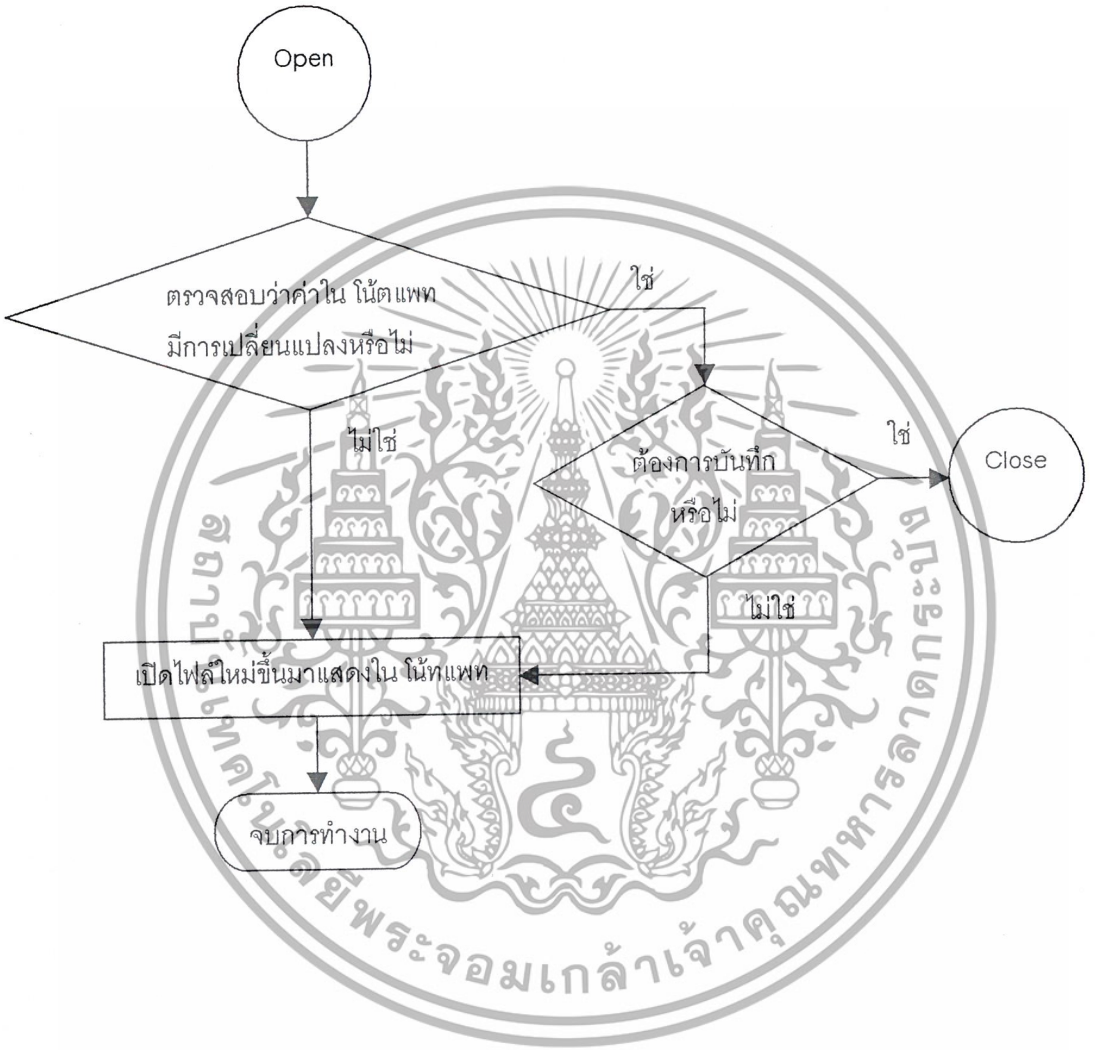
รูปแบบหน้าต่างของอีดิเตอร์เป็นดังนี้

รูปที่ 3.5 แสดงหน้าต่างอีดิเตอร์

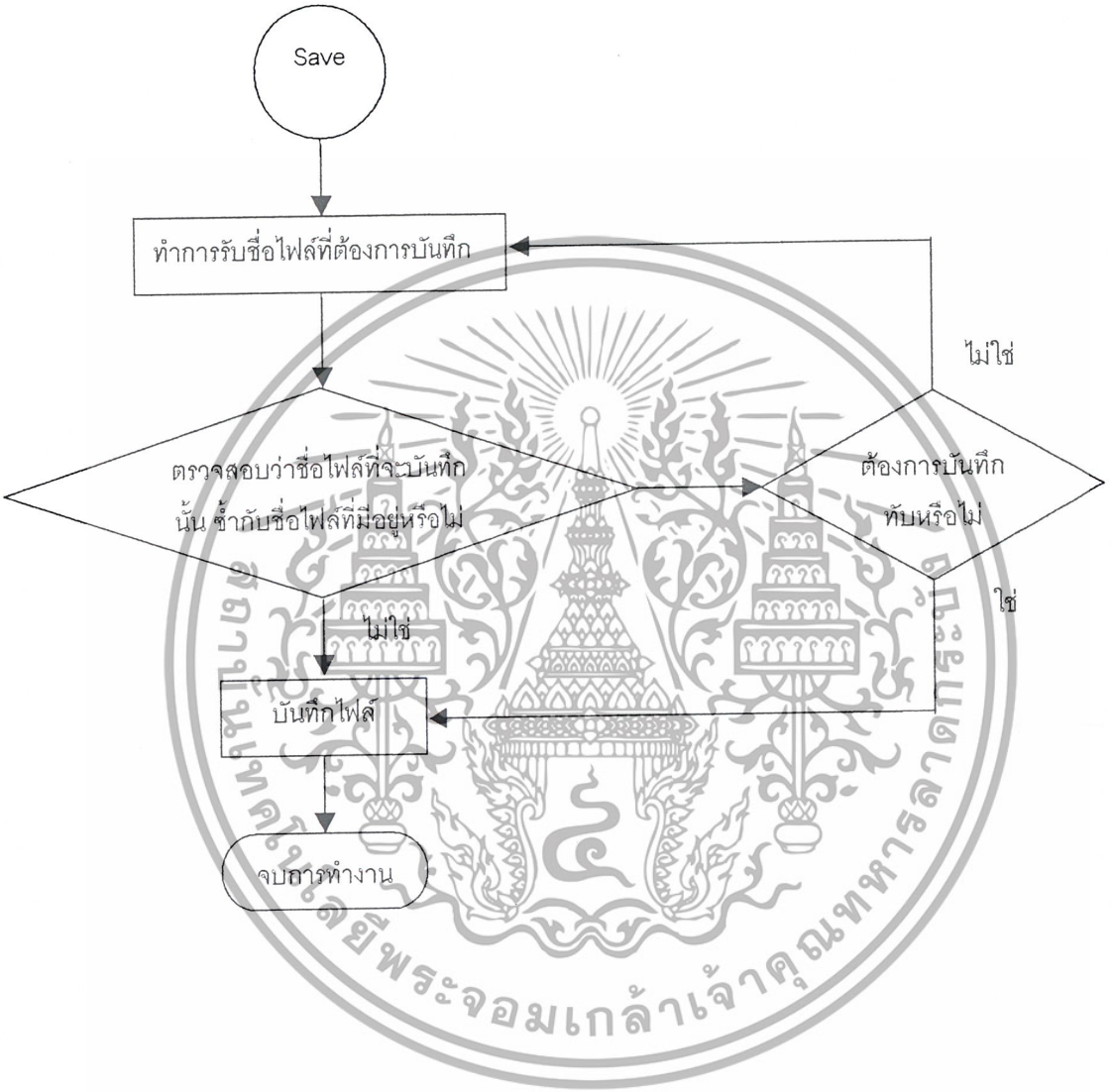


รูปที่ 3.6 โฟลวชาร์ตแสดงการเรียกใช้เมนู อีดิเตอร์

รูปที่ 3.6 (ต่อ)โฟลว์ชาร์ตแสดงการเรียกใช้เมนู อีดิเตอร์



รูปที่ 3.6 (ต่อ) โฟลว์ชาร์ตแสดงการเรียกใช้เมนู อีดิเตอร์

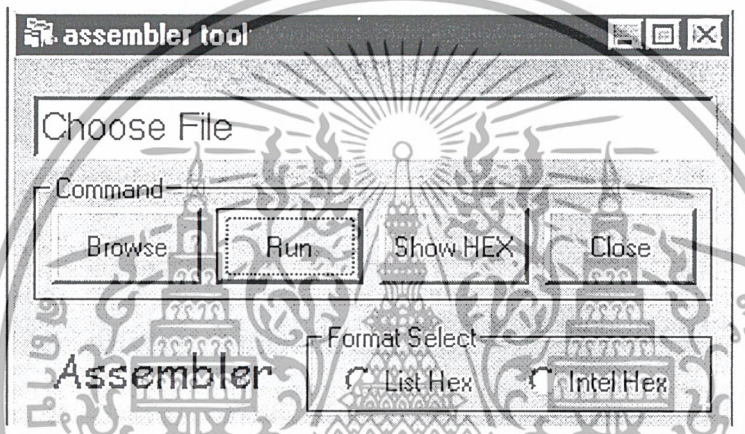


รูปที่ 3.6 (ต่อ) โฟลว์ชาร์จแสดงการเรียกใช้เมนู อีดิเตอร์

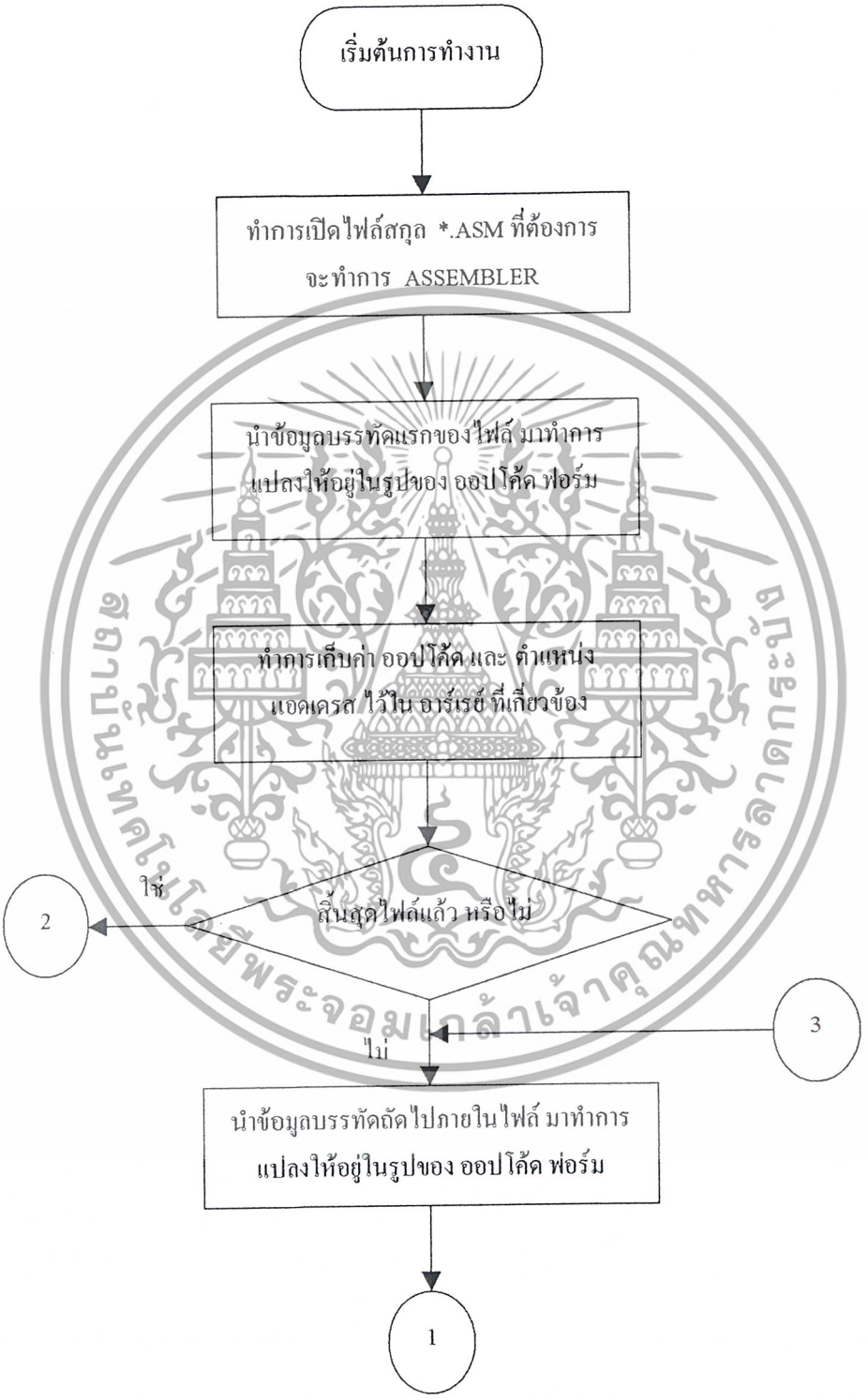
รูปที่ 3.6 (ต่อ)โฟลว์ชาร์จแสดงการเรียกใช้เมนู อีดิเตอร์

3.2.3 ส่วนของหน้าต่างแอสเซมเบลอร์

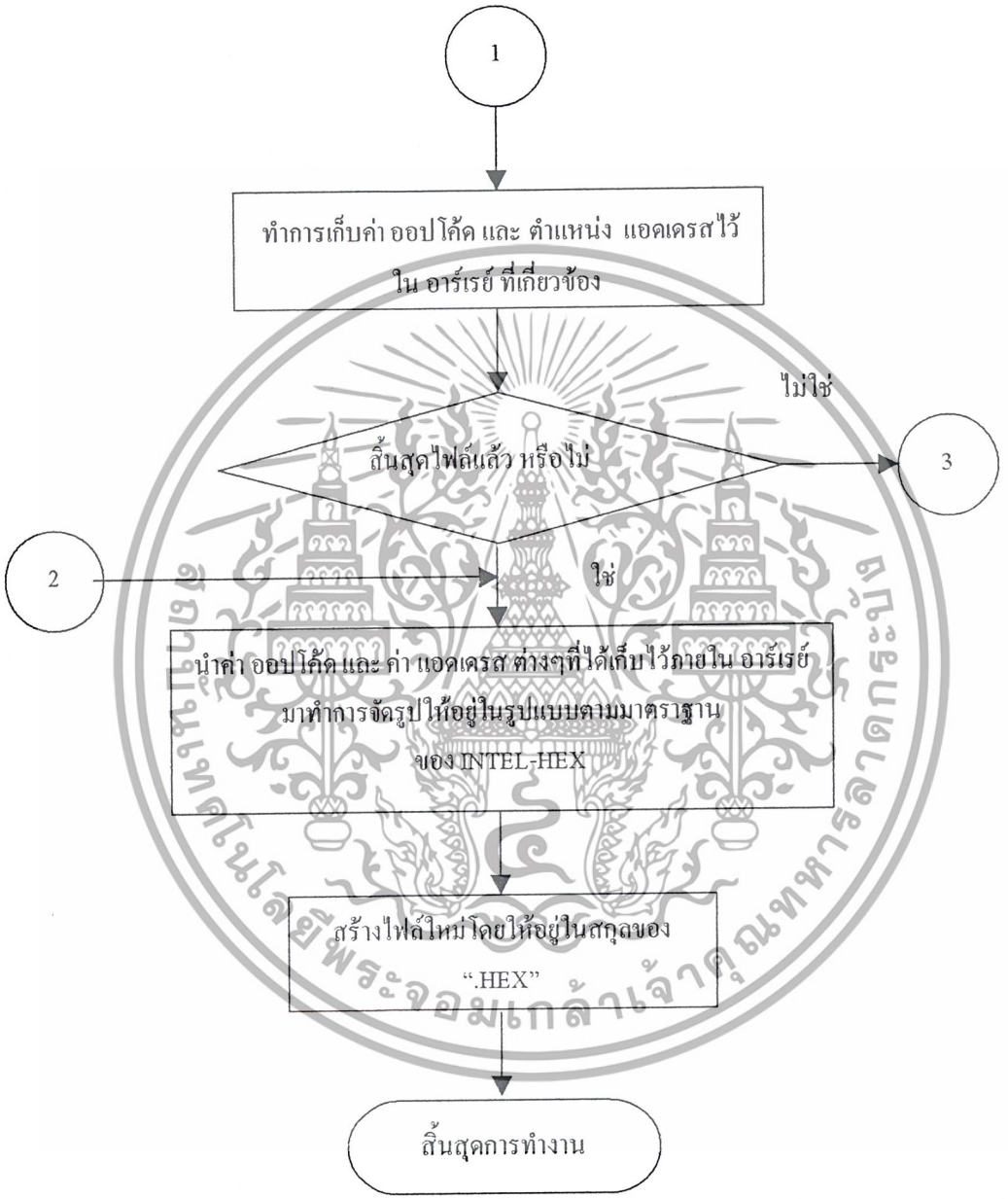
ส่วนของ (Assembler) ซึ่งใช้ในการแปลงรหัส แอสเซมบลี (Assembly Code) ให้เป็นรหัสฐานสิบหก (Hex Code) เพื่อใช้ในการโปรแกรมไมโครคอนโทรลเลอร์ จะมีส่วนให้เลือกชื่อไฟล์ที่ต้องการแอสเซมเบลอร์ และส่วนแสดงไฟล์ฐานสิบหกที่ได้จากการแอสเซมเบลอร์ ดังรูป 3.7



รูปที่ 3.7 แสดงหน้าต่างของแอสเซมเบลอร์



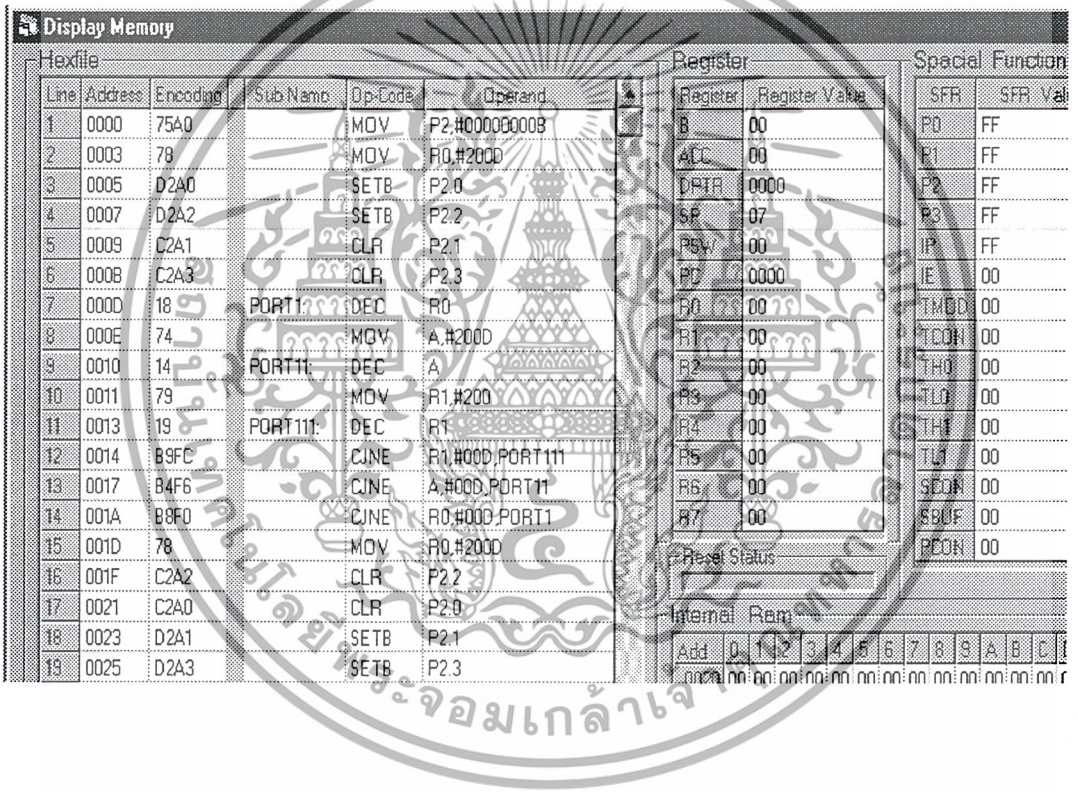
รูปที่ 3.8 โฟลวชาร์ตของกระบวนการแอสเซมเบลอร์



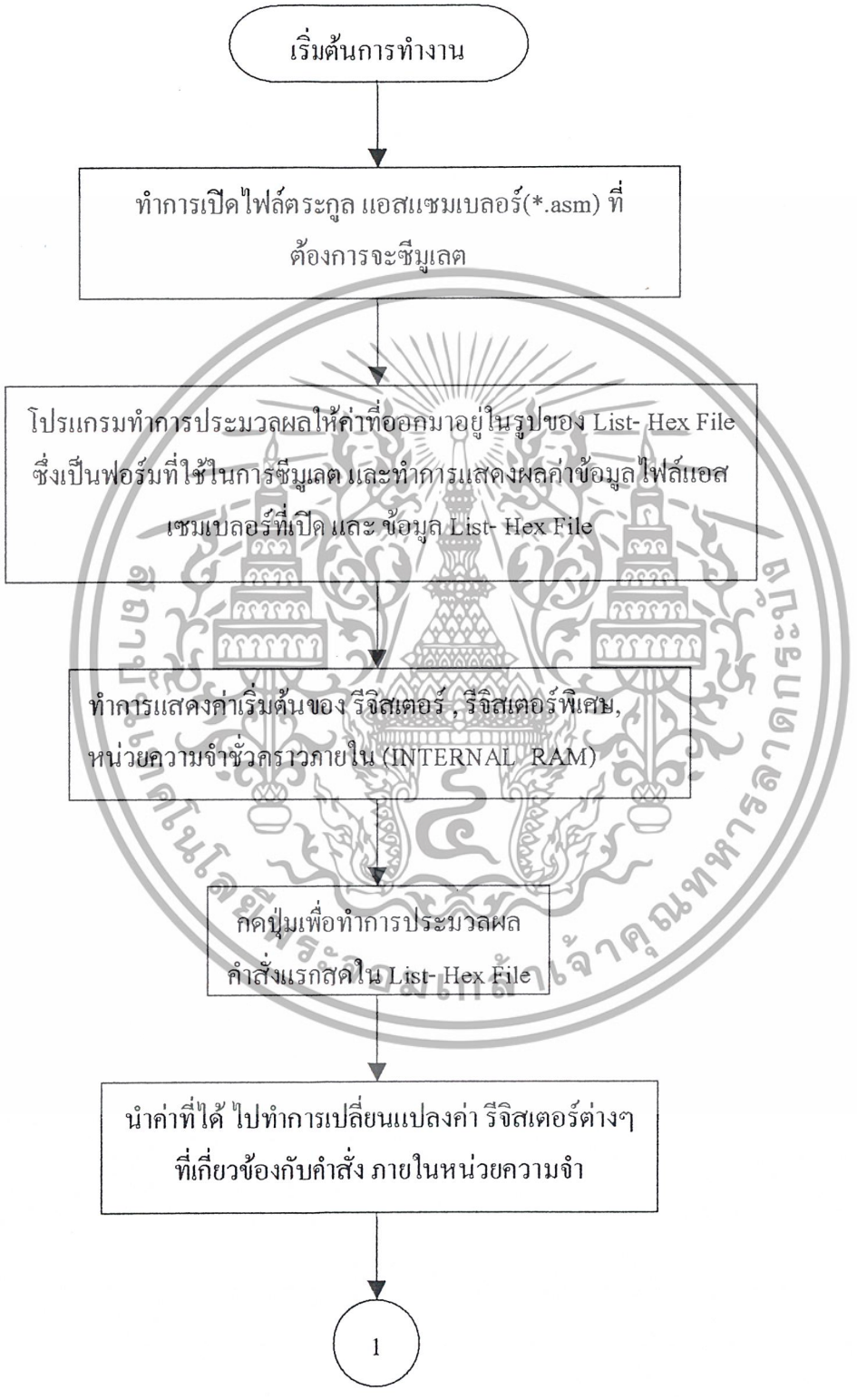
รูปที่ 3.8 (ต่อ) โฟลวชาร์ตของกระบวนการแอสเซมเบลอร์

3.2.4 ส่วนหน้าต่างของซีมูเลเตอร์

ในส่วนของหน้าต่างซีมูเลเตอร์นั้น เมื่อมีการเลือกเมนูไฟล์ ซีมูเลท (Simulate) ในหน้าต่างหลักแล้ว ก็จะปรากฏเมนูย่อย Load File ไว้ให้ผู้ใช้ทำการเลือกไฟล์ที่ต้องการทดสอบขึ้นมา และมีฟอร์มที่แสดงค่าของ รีจิสเตอร์ต่าง ๆ , แรมภายใน , รีจิสเตอร์พิเศษ แสดงขึ้นมาเพื่อให้ผู้ใช้ได้ทำการตรวจสอบโปรแกรม หน้าต่างของซีมูเลเตอร์แสดงได้ดังรูปที่ 3.9



รูปที่ 3.9 แสดงหน้าต่างของส่วนซีมูเลเตอร์



รูปที่ 3.10 โฟลวชาร์ตแสดงกระบวนการทำงานของ แอสเซมเบลอร์

รูปที่ 3.10 (ต่อ) โฟลว์ชาร์ตแสดงขั้นตอนการทำงานของเอมูเลตชั่นบอร์ด

รูปที่ 3.10 (ต่อ)โฟลวชาร์ตแสดงกระบวนการทำงานของ แอสเซมเบลอร์

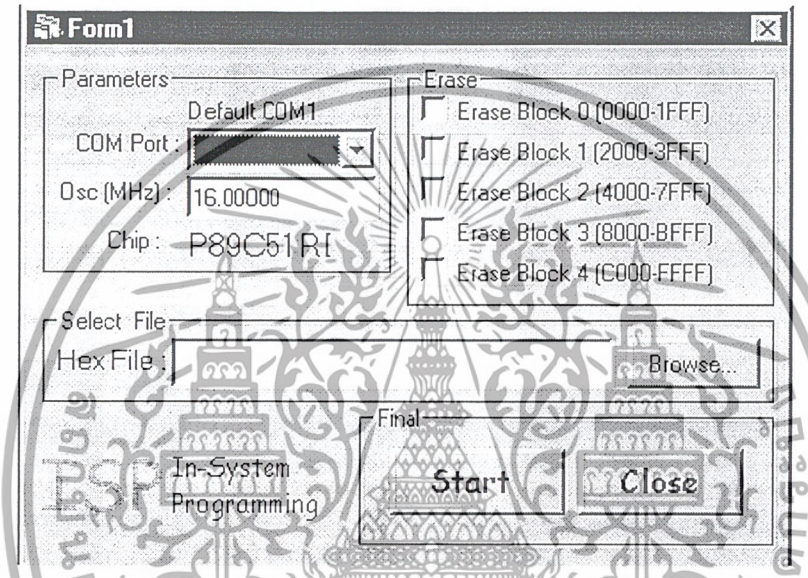
รูปที่ 3.10 (ต่อ)โฟลวชาร์ตแสดงกระบวนการทำงานของ แอสเซมเบลอร์

รูปที่ 3.10 (ต่อ) โฟลวชาร์ตแสดงกระบวนการทำงานของ แอสเซมเบลอร์

รูปที่ 3.10 (ต่อ) โฟลวชาร์ตแสดงกระบวนการทำงานของ แอสเซมเบลอร์

รูปที่ 3.10 (ต่อ)โฟลวชาร์ตแสดงกระบวนการทำงานของ แอสเซมเบลอร์

3.2.5 ส่วนหน้าต่างของ อินซิสเตม โปรแกรมมิ่ง (In-System Programming) ให้ผู้ใช้ทำการโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ ผ่านพอร์ตอนุกรม จากตัวคอมพิวเตอร์ หน้าต่างของส่วนนี้แสดงดังรูปที่ 3.11



รูปที่ 3.11 แสดงหน้าต่างของส่วน อินซิสเตม โปรแกรมมิ่ง

รูปที่ 3.12 โฟลวชาร์ตแสดงกระบวนการส่งข้อมูลแบบ ไอเอสพี สู่ ไมโครคอนโทรลเลอร์

รูปที่ 3.12 (ต่อ) โฟลวชาร์ตแสดงกระบวนการส่งข้อมูลแบบ ไอเอสพี สู่ไมโครคอนโทรลเลอร์

บทที่ 4

การทดลอง และ ผลการทดลอง

ในการทดลองนั้นได้แบ่งการทดลองออกเป็น 2 ส่วนคือ ส่วนซอฟต์แวร์และส่วนฮาร์ดแวร์

4.1 ส่วนของซอฟต์แวร์

4.1.1 อีดิเตอร์ (EDITOR)

หลังจากทำการเขียนโปรแกรม อีดิเตอร์ เรียบร้อยแล้ว จะต้องมีการทดลองเพื่อทำการตรวจสอบโปรแกรม อีดิเตอร์ ว่าสามารถทำงานได้อย่างถูกต้อง

วิธีการทดลอง

1.) ทำการกดที่เมนูบาร์ ดังรูป เพื่อเรียกโปรแกรม อีดิเตอร์ ขึ้นมา

2.) ทำการเปิดไฟล์ข้อมูลที่ต้องการแก้ไข

3.) ทำการแก้ไขข้อมูลภายใน ไฟล์นั้นๆ

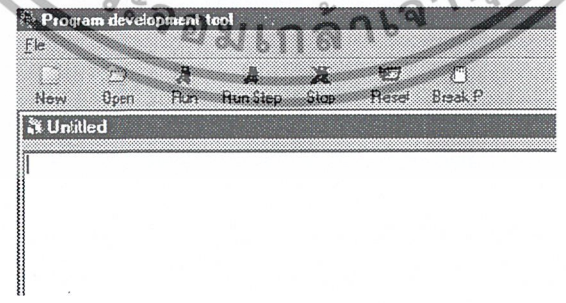
4.) ทำการบันทึก ไฟล์ข้อมูลที่ได้ทำการแก้ไขแล้วดังรูปที่ 4.1

5.) ปิดไฟล์ข้อมูลนั้น

6.) ทำการเปิดไฟล์ข้อมูลที่ได้ปิดไป และสังเกตข้อมูลภายใน ไฟล์ข้อมูลเพื่อเปรียบเทียบข้อมูลกับข้อมูลเดิมที่ได้ปิดไปแล้ว

ผลการทดลอง และ สรุป

เมื่อทำการเปรียบเทียบข้อมูลที่เปิดขึ้น กับ ข้อมูลเก่าที่ปิดไป ผลที่ปรากฏคือ ได้ข้อมูลภายในไฟล์เหมือนกันทั้งหมด ทำให้เราสามารถสรุปได้ว่า ในส่วนของ อีดิเตอร์ สามารถที่จะใช้งานได้อย่างถูกต้อง



รูปที่ 4.1 การใช้งานฟังก์ชั่นการบันทึกข้อมูล

4.1.2. แอสเซมเบลอร์ (ASSEMBLER)

เมื่อทำการเขียนโปรแกรมแอสเซมเบลอร์ เรียบร้อยแล้วจะต้องนำโปรแกรม มาทดลองเพื่อทำการตรวจสอบว่าโปรแกรมแอสเซมเบลอร์ สามารถทำงานได้อย่างถูกต้อง

วิธีการทดลอง

1.) ทำการกดที่เมนูบาร์ ดังรูป เพื่อเรียกโปรแกรมแอสเซมเบลอร์ขึ้นมา

2.) ทำการเลือกไฟล์ที่จะต้องทำการแอสเซมเบลอร์โดยการกดที่ปุ่ม Browse

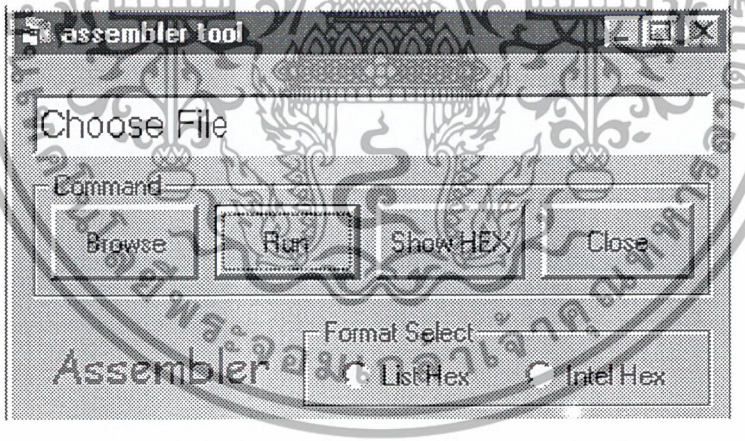
3.) เมื่อเราได้ทำการเลือกไฟล์แล้ว ชื่อและตำแหน่งข้อมูลจะถูกแสดงที่เท็กซ์บล็อก

4.) ทำการกดปุ่ม Run ณ. โปรแกรม เพื่อทำการแอสเซมเบลอร์

5.) ทำการเลือกปุ่ม Option List-Hex หรือปุ่ม Option Intel-Hex

6.) ทำการกดปุ่ม Show HEX เพื่อทำการดูค่าที่แปลงแล้ว

7.) ปิดโปรแกรม



รูปที่ 4.2 แสดงส่วนต่างๆของ โปรแกรมแอสเซมเบลอร์

ผลการทดลอง และ สรุป

เมื่อเราทำการเลือกไฟล์ที่เราต้องการจะทำการแอสเซมเบลอร์ พร้อมทั้ง ชื่อและตำแหน่ง ของไฟล์ปรากฏ ณ. เท็กซ์บล็อกเรียบร้อย เมื่อเรากด RUN แล้วจะปรากฏบล็อกแจ้งขึ้นถ้าหากการผิดพลาดภายในคำสั่ง ว่าจะให้ทำการแอสเซมเบลอร์ หรือ จะให้หยุดเพื่อทำการแก้ไขจนกว่าจะแอสเซมเบลอร์ จนจบไฟล์

หลังจากนั้นหากเราเลือกปุ่ม Option List-Hex แล้วก็ Show Hex จะปรากฏเท็กซ์บล็อกที่แสดงค่าเฮ็กซ์โค๊ด(Hexcode) ให้อยู่ในรูปของ List Form แต่ถ้าเลือก Option Intel-Hex และกด Show Hex แล้วจะปรากฏเท็กซ์ที่แสดง ค่าเฮ็กซ์โค๊ด ให้อยู่ใน รูป4.1 ของ Intel Hex Form และเมื่อทำการกดปุ่ม Close จะเป็นการปิดโปรแกรม

ดังนั้นสามารถสรุปได้ว่าโปรแกรมแอสเซมเบลอร์ นี้สามารถที่จะทำงานในการแปลงค่าได้อย่างถูกต้อง

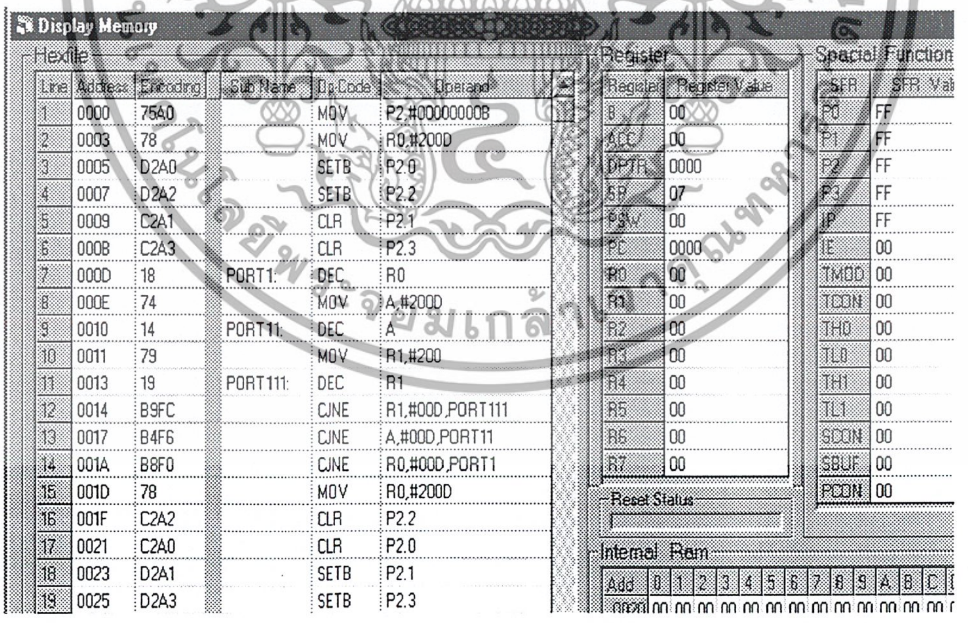
4.1.3. ซิมูเลเตอร์ (SIMULATOR)

หลังจากที่เราได้ทำการเขียนโปรแกรม ซิมูเลเตอร์เป็นที่เรียบร้อยแล้วเราจะต้องทำการทดสอบโปรแกรมเพื่อให้แน่ใจว่าโปรแกรมสามารถทำงานได้อย่างมีประสิทธิภาพ และถูกต้อง

วิธีการทดลอง

1.) ทำการเป็น ไฟล์ข้อมูลที่ต้องการซิมูเลต ดังรูป

2.) ให้ทดลองกดปุ่ม Run Step ,ปุ่ม Run All และ ปุ่ม Interrupt ดังรูป สังเกตการเปลี่ยนแปลงหลังกดปุ่ม



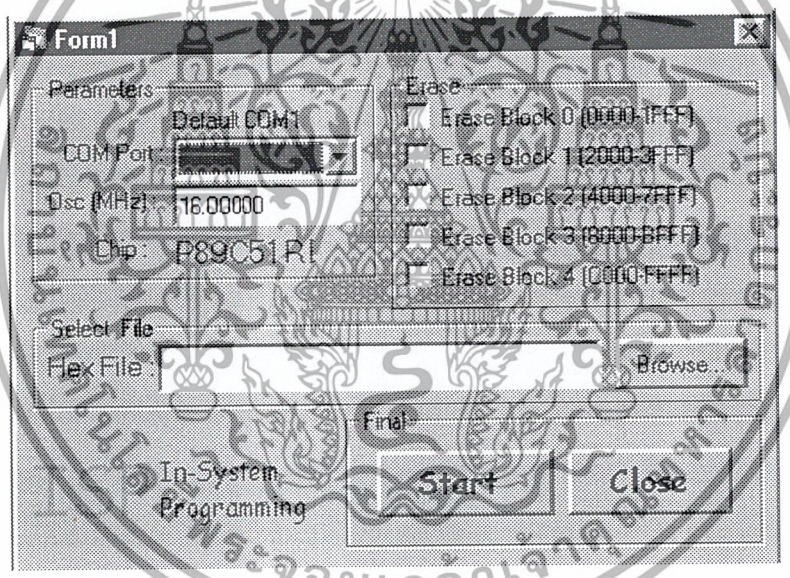
รูปที่ 4.3 แสดงภาพขณะกำลังประมวลผลของซิมูเลเตอร์

ผลการทดลอง

เมื่อเราได้ทำการกดปุ่ม Run Step แล้วผลจะปรากฏออกมาดังรูป คือค่าภายใน Register และหน่วยความจำภายในนั้นมีการเปลี่ยนแปลงไปจากค่าเริ่มต้นเดิมที่มีค่าเป็น 00 ทั้งหมด กลายเป็นค่าที่เกิดจากคำสั่งที่ ทำการประมวลผลขณะนั้นๆ แต่ถ้าหากว่าเรากดด้วย ปุ่ม Run All แล้ว ผลที่ได้จะปรากฏออกมาเป็นค่า เอาท์พุทสุดท้าย ทันที ส่วนปุ่ม Interrupt จะมีให้เลือกว่าเราจะใช้ Interrupt Source แหล่งไหน และขั้นตอนจะเป็นไปตาม โฟลวชาร์ตที่ได้กล่าวมาในส่วนของทฤษฎี

4.1.4 ไอเอสพี (ISP : IN SYSTEM PROGRAMING)

เป็นโปรแกรมที่ใช้สำหรับส่งข้อมูลผ่านทางพอร์ตอนุกรมเพื่อทำการบรรจุโปรแกม



รูปที่ 4.4 แสดงหน้าตาของโปรแกรมไอเอสพี

วิธีการทดลอง

1.) ทำการเปิดโปรแกรมขึ้นมาโดยการคลิ๊กที่คำว่า "ไอเอสพี "บนทูลบาร์จะปรากฏหน้าต่างไอเอสพีขึ้น ดังรูป 4.4

2.) ทำการเลือก คอมพอร์ตที่ต้องการติดต่อด้วย, ทำการเลือกบอร์ดเรต (นิยม 9,600)กำหนดความถี่เท่ากับบอร์ดที่เรานำไปใช้งานด้วย เลือกค่าสเตตัสไบท์ให้มีค่า = 00

3.) ทำการเลือกไฟล์ที่ต้องการส่ง

4.) กดปุ่ม Write และตามด้วย กดปุ่ม Start เพื่อทำการส่ง

ข้อควรระวัง!!! ไม่ควรที่จะเปลี่ยน ค่าของ บู๊ตเวกเตอร์ เด็ดขาด

ผลการทดลอง

เมื่อเราทำการกด Start แล้วไม่ปรากฏหน้าจอ ผิดพลาดแสดงออกมาให้เราเห็น ทำให้รู้ว่าสามารถส่งข้อมูลผ่านทาง พอร์ตอนุกรมได้อย่างเรียบร้อย

4.2 ส่วนของ Hardware

ได้ทำการทดลองการติดต่อแบบ ไอเอสพี ระหว่างตัว คอมพิวเตอร์ กับ ไมโครคอนโทรลเลอร์ P89C51RD2 โดยทำการสร้าง ไอเอสพี บอร์ดและเขียนโปรแกรมควบคุมการ เปิด-ปิด หลอด LED จาก คอมพิวเตอร์ แล้วส่งข้อมูลไปให้ไมโครคอนโทรลเลอร์ทำการควบคุมการ เปิด-ปิด

ในส่วนของวงจรเริ่มต้นจาก ต่อตัวไมโครคอนโทรลเลอร์และตัวไอซี MAX 232 ซึ่งเป็น IC ที่ใช้ปรับแรงดัน +5 Vจากตัวคอมพิวเตอร์ เป็นสัญญาณ TTL ส่งให้ตัวไมโครคอนโทรลเลอร์ ลงบนบอร์ดอเนกประสงค์ ในการส่งข้อมูลเข้าสู่บอร์ดนั้นเราส่งผ่านพอร์ตอนุกรม (COM1) ผ่านหัวต่อสายอนุกรม DB9 โดยต่อออกมาจากตัวคอมพิวเตอร์ วงจรการต่อพอร์ตอนุกรมผ่านตัว MAX 232 มายังตัวไมโครคอนโทรลเลอร์

จากนั้นทำการป้อนโปรแกรมเข้าสู่ตัวไมโครคอนโทรลเลอร์โดยใช้ โปรแกรมสำเร็จรูปของ บริษัท ฟิลิปส์ คือ โปรแกรม WIN ISP เป็นตัวส่งข้อมูลผ่านพอร์ทอนุกรมไปยังไมโครคอนโทรลเลอร์ พบว่าเมื่อทำการโปรแกรมเสร็จแล้ว ไมโครคอนโทรลเลอร์สามารถนำไปใช้งานได้ตรงตามกับที่เขียนโปรแกรมไว้ ในส่วนของ ฮาร์ดแวร์ จึงถือว่าทำงานได้ตามเป้าหมายที่ต้องการ

บทที่ 5
บทวิจารณ์ และ สรุปโครงงาน

ผลที่ออกมาของโครงงาน ชุดพัฒนาไมโครคอนโทรลเลอร์ MCS –51 โดยใช้ไอซี P89C51RD2 นับว่าได้รับความพอใจระดับหนึ่งจากผู้จัดทำ แต่ก็ยังมีจุดที่สามารถจะพัฒนาให้ผู้ใช้งานได้อย่างสะดวกยิ่งขึ้นอีก โดยจะทำการวิเคราะห์ข้อผิดพลาดและปัญหาที่เกิดขึ้นรวมถึงกรณีที่สามารถพัฒนาให้ก้าวหน้าต่อไปได้

5.1 ทางด้านฮาร์ดแวร์ (บอร์ดไอเอสพี)

สามารถที่จะส่งข้อมูลได้อย่างถูกต้องและมีความเสถียรสูง โดยข้อผิดพลาดและปัญหาที่ยังเกิดขึ้นอยู่ตอนนี้คือ มีสวิตซ์บางตัวที่จะต้องไปทำการควบคุมจากบนบอร์ดทำให้ผู้ใช้งานเสียเวลาในการเดินไป-เดินกลับ โดยทางผู้จัดทำได้คิดแนวทางในการแก้ไขและพัฒนาส่วนนี้ไว้แล้วคือ ให้เราส่งสัญญาณควบคุมออกมาทางพอร์ตอนุกรมแทน โดยหลักการคือเมื่อมีสัญญาณมากระทำ จะทำให้ทรานซิสเตอร์ทำงานส่งกระแสให้แก่ขานั้นให้มีสถานะเป็น + 5 โวลท์ และหากไม่มีสัญญาณมากระทำจำทำให้ขานั้นมีสถานะเป็น 0 โวลท์ เป็นการควบคุมจาก คอมพิวเตอร์แทนการเดินไปปรับสวิตซ์เอง

5.2 โปรแกรม แอสเซมเบลอร์

สามารถที่จะแปลงคำสั่งต่างๆจากภาษา แอสเซมบลีให้อยู่ในรูปของมาตราฐาน อินเทล เฮ็กซ์ ได้ประมาณ 85 % โดยยังขาดชุดคำสั่ง หรือตัวแปรบางตัวที่ผู้จัดทำไม่เคยพบ ดังนั้นแนวทางในการแก้ไขคือ ให้มีการศึกษาชุดคำสั่ง และตัวแปรบางตัว นอกเหนือจากที่ผู้จัดทำได้สร้างขึ้นมา และทำการเพิ่มเติมเข้าไป ในการพัฒนาโปรแกรมในส่วนของโปรแกรมแอสเซมเบลอร์ต่อไปในอนาคต

5.3 โปรแกรม อิดิเตอร์

ในส่วนของโปรแกรมอีดิเตอร์นับว่าสามารถใช้ได้อย่างมีประสิทธิภาพ ในด้านการแก้ไข , การเปิดข้อมูล, การบันทึกข้อมูล ฯลฯ ในส่วนของปัญหายังมีอยู่บ้าง แต่สำหรับในการพัฒนาในส่วนของอีดิเตอร์นี้คือสามารถใส่สี ลงไป ณ คำสั่งต่างๆ ของไฟล์ที่ต้องการ แก้ไขเพื่อให้ง่ายต่อการเขียนและพัฒนาโปรแกรม

5.4 โปรแกรมส่งข้อมูลแบบ ไอเอสพี

ในส่วนของโปรแกรมนี้ได้จักทำขึ้นเพื่อให้ผู้ใช้เห็นและเข้าใจง่ายโดยเรากำหนดให้ผู้ใช้กำหนดค่าเพียงไม่กี่ส่วนเท่านั้น แต่ปัญหาก็ยังเกิดขึ้นเกี่ยวกับการไม่เข้าใจการทำงานแบบไอเอสพี ของผู้ใช้ การแก้ไขตรงส่วนนี้ผู้พัฒนาคาดว่า ควรจะทำฟอร์มที่ใช้ในการช่วยเหลือ(Help Form) ให้ผู้ใช้สามรถที่จะเรียกขึ้นมาอ่านเมื่อไม่เข้าใจการใช้งานได้

5.5 โปรแกรมการจำลองการทำงานของไมโครคอนโทรลเลอร์

สามารถที่จะทำงานได้ในลักษณะการ ประมวลผลแบบทีละขั้นตอน กับการประมวลผลรวดเดียวจบ และสามารถทำการอินเตอร์รัปต์จากแหล่งได้ แต่สิ่งที่ยังขาดอยู่คือการต่อหน่วยความจำภายนอกเพิ่มเข้าไปยัง โปรแกรมจำลองการทำงานไมโครคอนโทรลเลอร์ เนื่องจากผู้จัดทำไม่ได้เล็งเห็นตรงส่วนนี้เป็นสำคัญแต่ถ้าหากมีการนำเอา โปรแกรมพัฒนานี้กลับมาปรับปรุงอีกคิดว่าคงจะเป็นส่วนที่สำคัญในอนาคตเนื่องจากเทคโนโลยีที่พัฒนาไปทำให้ขนาดโปรแกรมที่ทำการจำลองการทำงานเกินหน่วยความจำที่ได้กำหนดไว้ในโปรแกรมนี้ จึงควรนำไปพัฒนาสำหรับอนาคตอันใกล้นี้

ภาคผนวก

DATA SHEET

P89C51RB2/P89C51RC2/P89C51RD2

80C51 8-bit Flash microcontroller family

16KB/32KB/64KB ISP/IAP Flash with 512B/512B/1KB RAM

Preliminary specification
Supersedes data of 2000 Nov 28
IC28 Data Handbook

2001 Jan 11

Philips
Semiconductors

PHILIPS

80C51 8-bit Flash microcontroller family
16KB/32KB/64KB ISP/IAP Flash with 512B/512B/1KB RAM

P89C51RB2/P89C51RC2/ P89C51RD2

DESCRIPTION

The P89C51RB2/RC2/RD2 device contains a non-volatile 16kB/32kB/64kB Flash program memory that is both parallel programmable and serial In-System and In-Application Programmable. In-System Programming (ISP) allows the user to download new code while the microcontroller sits in the application. In-Application Programming (IAP) means that the microcontroller fetches new program code and reprograms itself while in the system. This allows for remote programming over a modem link. A default serial loader (boot loader) program in ROM allows serial In-System programming of the Flash memory via the UART without the need for a loader in the Flash code. For In-Application Programming, the user program erases and reprograms the Flash memory by use of standard routines contained in ROM.

This device executes one machine cycle in 6 clock cycles, hence providing twice the speed of a conventional 80C51. An OTP configuration bit lets the user select conventional 12 clock timing if desired.

This device is a Single-Chip 8-Bit Microcontroller manufactured in advanced CMOS process and is a derivative of the 80C51 microcontroller family. The instruction set is 100% compatible with the 80C51 instruction set.

The device also has four 8-bit I/O ports, three 16-bit timer/event counters, a multi-source, four-priority-level, nested interrupt structure, an enhanced UART and on-chip oscillator and timing circuits.

The added features of the P89C51RB2/RC2/RD2 makes it a powerful microcontroller for applications that require pulse width modulation, high-speed I/O and up/down counting capabilities such as motor control.

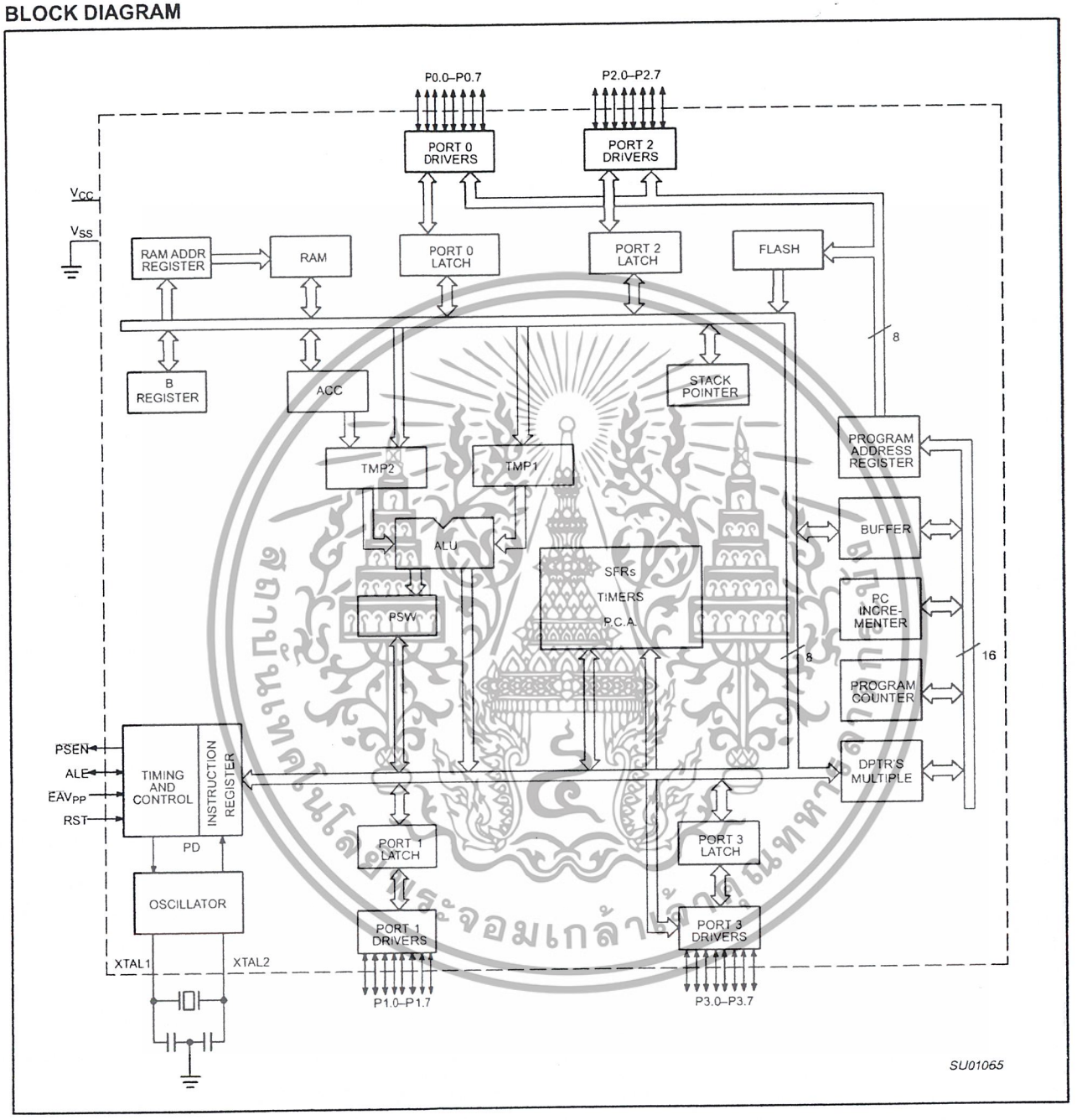
FEATURES

- 80C51 Central Processing Unit
- On-chip Flash Program Memory with In-System Programming (ISP) and In-Application Programming (IAP) capability
- Boot ROM contains low level Flash programming routines for downloading via the UART
- Can be programmed by the end-user application (IAP)
- 6 clocks per machine cycle operation (standard)
- 12 clocks per machine cycle operation (optional)
- Speed up to 20 MHz with 6 clock cycles per machine cycle (40 MHz equivalent performance); up to 33 MHz with 12 clocks per machine cycle
- Fully static operation
- RAM expandable externally to 64 kB
- 4 level priority interrupt
- 7 interrupt sources
- Four 8-bit I/O ports
- Full-duplex enhanced UART
  - Framing error detection
  - Automatic address recognition
- Power control modes
  - Clock can be stopped and resumed
  - Idle mode
  - Power down mode
- Programmable clock out
- Second DPTR register
- Asynchronous port reset
- Low EMI (inhibit ALE)
- Programmable Counter Array (PCA)
  - PWM
  - Capture/compare

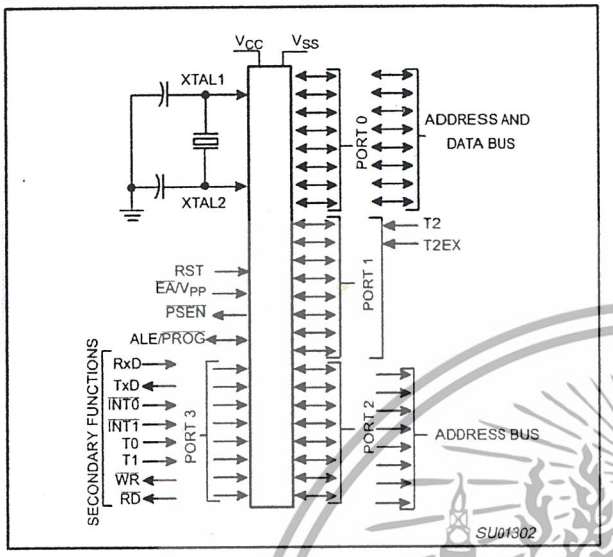
ORDERING INFORMATION

	PHILIPS (EXCEPT NORTH AMERICA) PART ORDER NUMBER PART MARKING	PHILIPS NORTH AMERICA PART ORDER NUMBER	MEMORY		TEMPERATURE RANGE (°C) AND PACKAGE	VOLTAGE RANGE	FREQUENCY (MHz)		DWG #
			FLASH	RAM			6 CLOCK MODE	12 CLOCK MODE	
1	P89C51RB2HBA	P89C51RB2BA	16 kB	512 B	0 to +70, PLCC	4.5–5.5 V	0 to 20 MHz	0 to 33 MHz	SOT187-2
2	P89C51RB2HBBD	P89C51RB2BBD	16 kB	512 B	0 to +70, LQFP	4.5–5.5 V	0 to 20 MHz	0 to 33 MHz	SOT389-1
3	P89C51RC2HBP	P89C51RC2BN	32 kB	512 B	0 to +70, PDIP	4.5–5.5 V	0 to 20 MHz	0 to 33 MHz	SOT129-1
4	P89C51RC2HBA	P89C51RC2BA	32 kB	512 B	0 to +70, PLCC	4.5–5.5 V	0 to 20 MHz	0 to 33 MHz	SOT187-2
5	P89C51RC2HFA	P89C51RC2FA	32 kB	512 B	–40 to +85, PLCC	4.75–5.25 V	0 to 20 MHz	0 to 33 MHz	SOT187-2
6	P89C51RC2HBBD	P89C51RC2BBD	32 kB	512 B	0 to +70, LQFP	4.5–5.5 V	0 to 20 MHz	0 to 33 MHz	SOT389-1
7	P89C51RC2HFBD	P89C51RC2FBD	32 kB	512 B	–40 to +85, LQFP	4.75–5.25 V	0 to 20 MHz	0 to 33 MHz	SOT389-1
8	P89C51RD2HBP	P89C51RD2BN	64 kB	1 kB	0 to +70, PDIP	4.5–5.5 V	0 to 20 MHz	0 to 33 MHz	SOT129-1
9	P89C51RD2HBA	P89C51RD2BA	64 kB	1 kB	0 to +70, PLCC	4.5–5.5 V	0 to 20 MHz	0 to 33 MHz	SOT187-2
10	P89C51RD2HBBD	P89C51RD2BBD	64 kB	1 kB	0 to +70, LQFP	4.5–5.5 V	0 to 20 MHz	0 to 33 MHz	SOT389-1

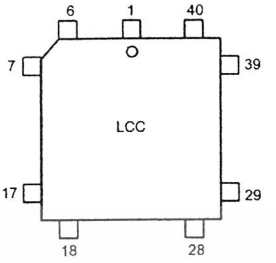
BLOCK DIAGRAM



LOGIC SYMBOL



Plastic Leaded Chip Carrier



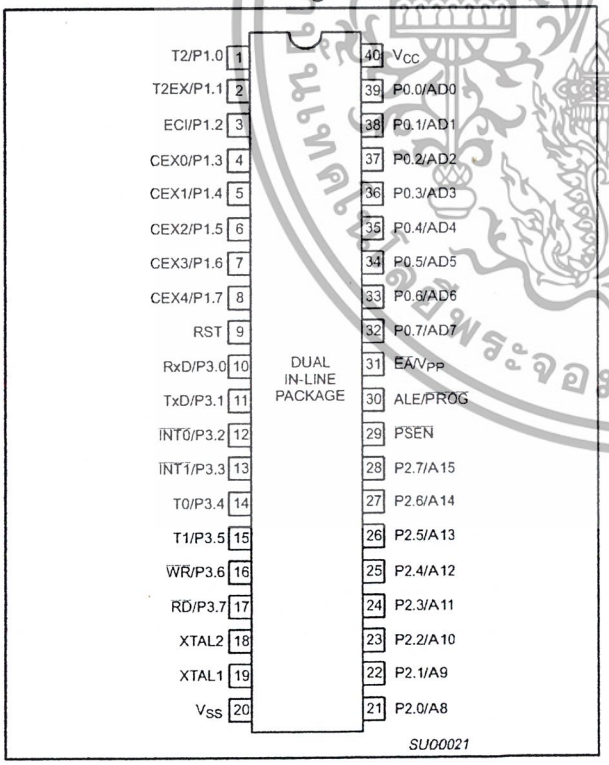
Pin	Function	Pin	Function	Pin	Function
1	NIC*	16	P3.4/T0	31	P2.7/A15
2	P1.0/T2	17	P3.5/T1	32	PSEN
3	P1.1/T2EX	18	P3.6/WR	33	ALE/PROG
4	P1.2/ECI	19	P3.7/RD	34	NIC*
5	P1.3/CEX0	20	XTAL2	35	EA/VPP
6	P1.4/CEX1	21	XTAL1	36	P0.7/AD7
7	P1.5/CEX2	22	VSS	37	P0.6/AD6
8	P1.6/CEX3	23	NIC*	38	P0.5/AD5
9	P1.7/CEX4	24	P2.0/A8	39	P0.4/AD4
10	RST	25	P2.1/A9	40	P0.3/AD3
11	P3.0/RxD	26	P2.2/A10	41	P0.2/AD2
12	NIC*	27	P2.3/A11	42	P0.1/AD1
13	P3.1/TxD	28	P2.4/A12	43	P0.0/AD0
14	P3.2/INT0	29	P2.5/A13	44	VCC
15	P3.3/INT1	30	P2.6/A14		

* NO INTERNAL CONNECTION

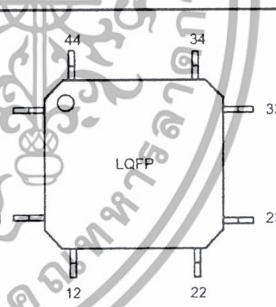
SU00023

PINNING

Plastic Dual In-Line Package



Plastic Quad Flat Pack



Pin	Function	Pin	Function	Pin	Function
1	P1.5/CEX2	16	VSS	31	P0.6/AD6
2	P1.6/CEX3	17	NIC*	32	P0.5/AD5
3	P1.7/CEX4	18	P2.0/A8	33	P0.4/AD4
4	RST	19	P2.1/A9	34	P0.3/AD3
5	P3.0/RxD	20	P2.2/A10	35	P0.2/AD2
6	NIC*	21	P2.3/A11	36	P0.1/AD1
7	P3.1/TxD	22	P2.4/A12	37	P0.0/AD0
8	P3.2/INT0	23	P2.5/A13	38	VCC
9	P3.3/INT1	24	P2.6/A14	39	NIC*
10	P3.4/T0	25	P2.7/A15	40	P1.0/T2
11	P3.5/T1	26	PSEN	41	P1.1/T2EX
12	P3.6/WR	27	ALE/PROG	42	P1.2/ECI
13	P3.7/RD	28	NIC*	43	P1.3/CEX0
14	XTAL2	29	EA/VPP	44	P1.4/CEX1
15	XTAL1	30	P0.7/AD7		

* NO INTERNAL CONNECTION

SU01400

PIN DESCRIPTIONS

MNEMONIC	PIN NUMBER			TYPE	NAME AND FUNCTION
	PDIP	PLCC	LQFP		
V_{SS}	20	22	16	I	**Ground:** 0 V reference.
V_{CC}	40	44	38	I	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0–0.7	39–32	43–36	37–30	I/O	**Port 0:** Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application, it uses strong internal pull-ups when emitting 1s.
P1.0–P1.7	1–8	2–9	40–44, 1–3	I/O	**Port 1:** Port 1 is an 8-bit bidirectional I/O port with internal pull-ups on all pins except P1.6 and P1.7 which are open drain. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 1 pins that are externally pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}).
					Alternate functions for 89C51RB2/RC2/RD2 Port 1 include:
	1	2	40	I/O	**T2 (P1.0):** Timer/Counter 2 external count input/Clockout (see Programmable Clock-Out)
	2	3	41	I	**T2EX (P1.1):** Timer/Counter 2 Reload/Capture/Direction Control
	3	4	42	I	**ECI (P1.2):** External Clock Input to the PCA
	4	5	43	I/O	**CEX0 (P1.3):** Capture/Compare External I/O for PCA module 0
	5	6	44	I/O	**CEX1 (P1.4):** Capture/Compare External I/O for PCA module 1
	6	7	1	I/O	**CEX2 (P1.5):** Capture/Compare External I/O for PCA module 2
	7	8	2	I/O	**CEX3 (P1.6):** Capture/Compare External I/O for PCA module 3
	8	9	3	I/O	**CEX4 (P1.7):** Capture/Compare External I/O for PCA module 4
P2.0–P2.7	21–28	24–31	18–25	I/O	**Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 2 pins that are externally being pulled low will source current because of the internal pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOV @Ri), port 2 emits the contents of the P2 special function register. P2.7 must be a "1" to program and erase the device.
P3.0–P3.7	10–17	11, 13–19	5, 7–13	I/O	**Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, port 3 pins that are externally being pulled low will source current because of the pull-ups. (See DC Electrical Characteristics: I_{IL}). Port 3 also serves the special features of the 89C51RB2/RC2/RD2, as listed below:
	10	11	5	I	**RxD (P3.0):** Serial input port
	11	13	7	O	**TxD (P3.1):** Serial output port
	12	14	8	I	**INT0 (P3.2):** External interrupt
	13	15	9	I	**INT1 (P3.3):** External interrupt
	14	16	10	I	**T0 (P3.4):** Timer 0 external input
	15	17	11	I	**T1 (P3.5):** Timer 1 external input
	16	18	12	O	**$\overline{WR}$ (P3.6):** External data memory write strobe
	17	19	13	O	**$\overline{RD}$ (P3.7):** External data memory read strobe
RST	9	10	4	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}.
ALE	30	33	27	O	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted twice every machine cycle, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. ALE can be disabled by setting SFR auxiliary.0. With this bit set, ALE will be active only during a MOVX instruction.

MNEMONIC	PIN NUMBER			TYPE	NAME AND FUNCTION
	PDIP	PLCC	LQFP		
$\overline{PSEN}$	29	32	26	O	**Program Store Enable:** The read strobe to external program memory. When executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
$\overline{EA}/V_{PP}$	31	35	29	I	**External Access Enable/Programming Supply Voltage:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations. If $\overline{EA}$ is held high, the device executes from internal program memory. The value on the $\overline{EA}$ pin is latched when RST is released and any subsequent changes have no effect. This pin also receives the programming supply voltage (V_{PP}) during Flash programming.
XTAL1	19	21	15	I	**Crystal 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	O	**Crystal 2:** Output from the inverting oscillator amplifier.

NOTE:
To avoid "latch-up" effect at power-on, the voltage on any pin (other than V_{PP}) must not be higher than V_{CC} + 0.5 V or less than V_{SS} – 0.5 V.

Table 1. Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
AUXR#	Auxiliary	8EH	–	–	–	–	–	–	EXTRAM	AO	xxxxxx00B
AUXR1#	Auxiliary 1	A2H	–	–	ENBOOT	–	GF2	0	–	DPS	xxxxxxx0B
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
CCAP0H#	Module 0 Capture High	FAH									xxxxxxxxB
CCAP1H#	Module 1 Capture High	FBH									xxxxxxxxB
CCAP2H#	Module 2 Capture High	FCH									xxxxxxxxB
CCAP3H#	Module 3 Capture High	FDH									xxxxxxxxB
CCAP4H#	Module 4 Capture High	FEH									xxxxxxxxB
CCAP0L#	Module 0 Capture Low	EAH									xxxxxxxxB
CCAP1L#	Module 1 Capture Low	EBH									xxxxxxxxB
CCAP2L#	Module 2 Capture Low	ECH									xxxxxxxxB
CCAP3L#	Module 3 Capture Low	EDH									xxxxxxxxB
CCAP4L#	Module 4 Capture Low	EEH									xxxxxxxxB
CCAPM0#	Module 0 Mode	DAH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
CCAPM1#	Module 1 Mode	DBH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
CCAPM2#	Module 2 Mode	DCH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
CCAPM3#	Module 3 Mode	DDH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
CCAPM4#	Module 4 Mode	DEH	–	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	x0000000B
			DF	DE	DD	DC	DB	DA	D9	D8	
CCON*#	PCA Counter Control	D8H	CF	CR	–	CCF4	CCF3	CCF2	CCF1	CCF0	00x00000B
CH#	PCA Counter High	F9H									00H
CL#	PCA Counter Low	E9H									00H
CMOD#	PCA Counter Mode	D9H	CIDL	WDTE	–	–	–	CPS1	CPS0	ECF	00xxx000B
DPTR:	Data Pointer (2 bytes)										
DPH	Data Pointer High	83H									00H
DPL	Data Pointer Low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt Enable 0	A8H	EA	EC	ET2	ES	ET1	EX1	ET0	EX0	00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt Priority	B8H	–	PPC	PT2	PS	PT1	PX1	PT0	PX0	x0000000B
			B7	B6	B5	B4	B3	B2	B1	B0	
IPH#	Interrupt Priority High	B7H	–	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	x0000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	CEX4	CEX3	CEX2	CEX1	CEX0	ECI	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TxD	RxD	FFH
PCON#[1]	Power Control	87H	SMOD1	SMOD0	–	POF	GF1	GF0	PD	IDL	00xxx000B

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.
– Reserved bits.
1. Reset value depends on reset source.

Table 1. Special Function Registers (Continued)

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program Status Word	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	00000000B
RCAP2H#	Timer 2 Capture High	CBH									00H
RCAP2L#	Timer 2 Capture Low	CAH									00H
SADDR#	Slave Address	A9H									00H
SADEN#	Slave Address Mask	B9H									00H
SBUF	Serial Data Buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial Control	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack Pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer Control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
T2CON*	Timer 2 Control	C8H	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	00H
T2MOD#	Timer 2 Mode Control	C9H	–	–	–	–	–	–	T2OE	DCEN	xxxxxx00B
TH0	Timer High 0	8CH									00H
TH1	Timer High 1	8DH									00H
TH2#	Timer High 2	CDH									00H
TL0	Timer Low 0	8AH									00H
TL1	Timer Low 1	8BH									00H
TL2#	Timer Low 2	CCH									00H
TMOD	Timer Mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00H
WDTRST	Watchdog Timer Reset	A6H									

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.
– Reserved bits.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. Minimum and maximum high and low times specified in the data sheet must be observed.

This device is configured at the factory to operate using 6 clock periods per machine cycle, referred to in this datasheet as "6 clock mode". (This yields performance equivalent to twice that of standard 80C51 family devices). It may be optionally configured on commercially-available EPROM programming equipment to operate at 12 clocks per machine cycle, referred to in this datasheet as "12 clock mode". Once 12 clock mode has been configured, it cannot be changed back to 6 clock mode.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (12 oscillator periods in 6 clock mode, or 24 oscillator periods in 12 clock mode), while the oscillator is running. To ensure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{CC} and RST must come up at the same time for a proper start-up. Ports 1, 2, and 3 will asynchronously be driven to their reset condition when a voltage above V_{IH1} (min.) is applied to RESET.

The value on the $\overline{EA}$ pin is latched when RST is deasserted and has no further effect.

LOW POWER MODES

Stop Clock Mode

The static design enables the clock speed to be reduced down to 0 MHz (stopped). When the oscillator is stopped, the RAM and Special Function Registers retain their values. This mode allows step-by-step utilization and permits reduced system power consumption by lowering the clock frequency down to any value. For lowest power consumption the Power Down mode is suggested.

Idle Mode

In the idle mode (see Table 2), the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

Power-Down Mode

To save even more power, a Power Down mode (see Table 2) can be invoked by software. In this mode, the oscillator is stopped and the instruction that invoked Power Down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values down to 2.0 V and care must be taken to return V_{CC} to the minimum specified operating voltages before the Power Down Mode is terminated.

Either a hardware reset or external interrupt can be used to exit from Power Down. Reset redefines all the SFRs but does not change the on-chip RAM. An external interrupt allows both the SFRs and the on-chip RAM to retain their values.

To properly terminate Power Down, the reset or external interrupt should not be executed before V_{CC} is restored to its normal operating level and must be held active long enough for the oscillator to restart and stabilize (normally less than 10 ms).

With an external interrupt, INT0 and INT1 must be enabled and configured as level-sensitive. Holding the pin low restarts the oscillator but bringing the pin back high completes the exit. Once the interrupt is serviced, the next instruction to be executed after RETI will be the one following the instruction that put the device into Power Down.

POWER OFF FLAG

The Power Off Flag (POF) is set by on-chip circuitry when the V_{CC} level on the P89C51RB2/RC2/RD2 rises from 0 to 5 V. The POF bit can be set or cleared by software allowing a user to determine if the reset is the result of a power-on or a warm start after powerdown. The V_{CC} level must remain above 3 V for the POF to remain unaffected by the V_{CC} level.

Design Consideration

- When the idle mode 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 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.

ONCE™ Mode

The ONCE ("On-Circuit Emulation") Mode facilitates testing and debugging of systems without the device having to be removed from the circuit. The ONCE Mode is invoked by:

1. Pull ALE low while the device is in reset and $\overline{PSEN}$ is high;
2. Hold ALE low as RST is deactivated.

While the device is in ONCE Mode, the Port 0 pins go into a float state, and the other port pins and ALE and $\overline{PSEN}$ are weakly pulled high. The oscillator circuit remains active. While the device is in this mode, an emulator or test CPU can be used to drive the circuit. Normal operation is restored when a normal reset is applied.

Programmable Clock-Out

A 50% duty cycle clock can be programmed to come out on P1.0. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed:

1. to input the external clock for Timer/Counter 2, or
2. to output a 50% duty cycle clock ranging from 122 Hz to 8 MHz at a 16 MHz operating frequency (61 Hz to 4 MHz in 12 clock mode).

To configure the Timer/Counter 2 as a clock generator, bit C/$\overline{T2}$ (in T2CON) must be cleared and bit T20E in T2MOD must be set. Bit TR2 (T2CON.2) also must be set to start the timer.

The Clock-Out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L) as shown in this equation:

$$\frac{\text{Oscillator Frequency}}{n \times (65536 - \text{RCAP2H, RCAP2L})}$$

$n = \begin{cases} 2 & \text{in 6 clock mode} \\ 4 & \text{in 12 clock mode} \end{cases}$

Where (RCAP2H,RCAP2L) = the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

In the Clock-Out mode Timer 2 roll-overs will not generate an interrupt. This is similar to when it is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and the Clock-Out frequency will be the same.

Table 2. External Pin Status During Idle and Power-Down Mode

MODE	PROGRAM MEMORY	ALE	$\overline{PSEN}$	PORT 0	PORT 1	PORT 2	PORT 3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

TIMER 2 OPERATION

Timer 2

Timer 2 is a 16-bit Timer/Counter which can operate as either an event timer or an event counter, as selected by C/T2* in the special function register T2CON (see Figure 1). Timer 2 has three operating modes: Capture, Auto-reload (up or down counting), and Baud Rate Generator, which are selected by bits in the T2CON as shown in Table 3.

Capture Mode

In the capture mode there are two options which are selected by bit EXEN2 in T2CON. If EXEN2=0, then timer 2 is a 16-bit timer or counter (as selected by C/T2* in T2CON) which, upon overflowing sets bit TF2, the timer 2 overflow bit. This bit can be used to generate an interrupt (by enabling the Timer 2 interrupt bit in the IE register). If EXEN2= 1, Timer 2 operates as described above, but with the added feature that a 1- to -0 transition at external input T2EX causes the current value in the Timer 2 registers, TL2 and TH2, to be captured into registers RCAP2L and RCAP2H, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and EXF2 like TF2 can generate an interrupt (which vectors to the same location as Timer 2 overflow interrupt. The Timer 2 interrupt service routine can interrogate TF2 and EXF2 to determine which event caused the interrupt). The capture mode is illustrated in Figure 2 (There is no reload value for TL2 and TH2 in this mode. Even when a capture event occurs from T2EX, the counter keeps on counting T2EX pin transitions or osc/6 pulses (osc/12 in 12 clock mode).).

Auto-Reload Mode (Up or Down Counter)

In the 16-bit auto-reload mode, Timer 2 can be configured (as either a timer or counter [C/T2* in T2CON]) then programmed to count up or down. The counting direction is determined by bit DCEN (Down Counter Enable) which is located in the T2MOD register (see Figure 3). When reset is applied the DCEN=0 which means Timer 2 will default to counting up. If DCEN bit is set, Timer 2 can count up or down depending on the value of the T2EX pin.

Figure 4 shows Timer 2 which will count up automatically since DCEN=0. In this mode there are two options selected by bit EXEN2 in T2CON register. If EXEN2=0, then Timer 2 counts up to 0FFFFH and sets the TF2 (Overflow Flag) bit upon overflow. This causes the Timer 2 registers to be reloaded with the 16-bit value in RCAP2L and RCAP2H. The values in RCAP2L and RCAP2H are preset by software means.

If EXEN2=1, then a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at input T2EX. This transition also sets the EXF2 bit. The Timer 2 interrupt, if enabled, can be generated when either TF2 or EXF2 are 1.

In Figure 5 DCEN=1 which enables Timer 2 to count up or down. This mode allows pin T2EX to control the direction of count. When a logic 1 is applied at pin T2EX Timer 2 will count up. Timer 2 will overflow at 0FFFFH and set the TF2 flag, which can then generate an interrupt, if the interrupt is enabled. This timer overflow also causes the 16-bit value in RCAP2L and RCAP2H to be reloaded into the timer registers TL2 and TH2.

When a logic 0 is applied at pin T2EX this causes Timer 2 to count down. The timer will underflow when TL2 and TH2 become equal to the value stored in RCAP2L and RCAP2H. Timer 2 underflow sets the TF2 flag and causes 0FFFFH to be reloaded into the timer registers TL2 and TH2.

The external flag EXF2 toggles when Timer 2 underflows or overflows. This EXF2 bit can be used as a 17th bit of resolution if needed. The EXF2 flag does not generate an interrupt in this mode of operation.

Symbol	Position	Name and Significance
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.
C/T2	T2CON.1	Timer or counter select. (Timer 2) 0 = Internal timer (OSC/6 in 6 clock mode or OSC/12 in 12 clock mode) 1 = External event counter (falling edge triggered).
CP/RL2	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

SU01251

Figure 1. Timer/Counter 2 (T2CON) Control Register

Table 3. Timer 2 Operating Modes

RCLK + TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud rate generator
X	X	0	(off)

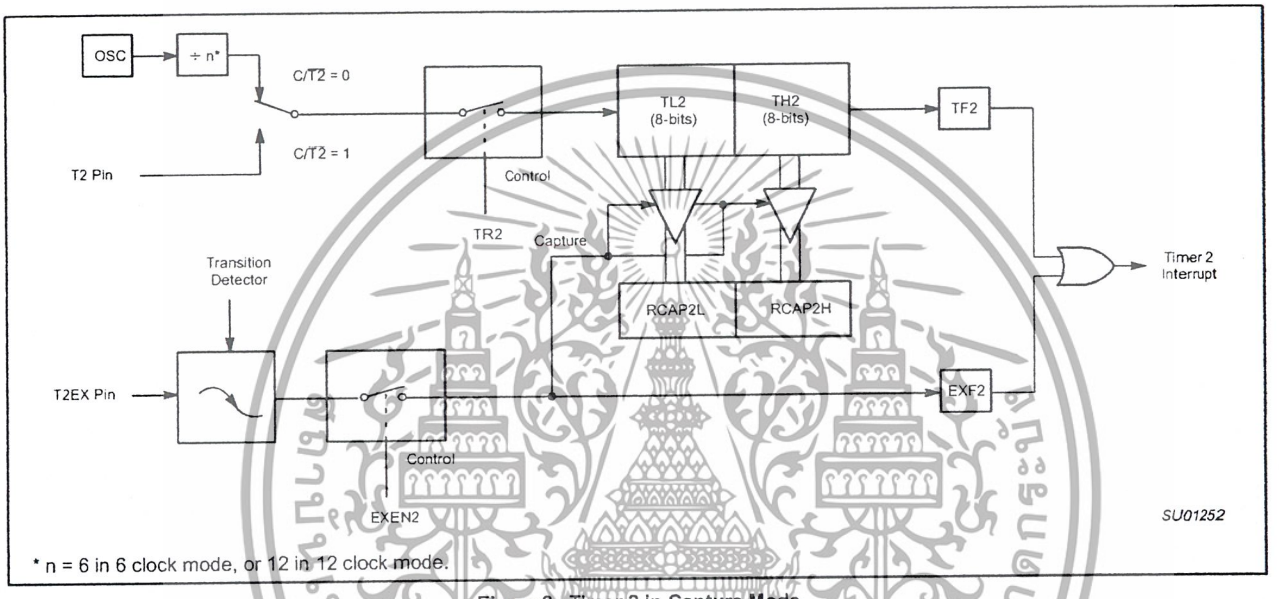


Figure 2. Timer 2 in Capture Mode

T2MOD Address = 0C9H Reset Value = XXXX XX00B

Not Bit Addressable

—	—	—	—	—	—	T2OE	DCEN
Bit 7	6	5	4	3	2	1	0

Symbol	Function
—	Not implemented, reserved for future use.*
T2OE	Timer 2 Output Enable bit.
DCEN	Down Count Enable bit. When set, this allows Timer 2 to be configured as an up/down counter.

* User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

SU00729

Figure 3. Timer 2 Mode (T2MOD) Control Register

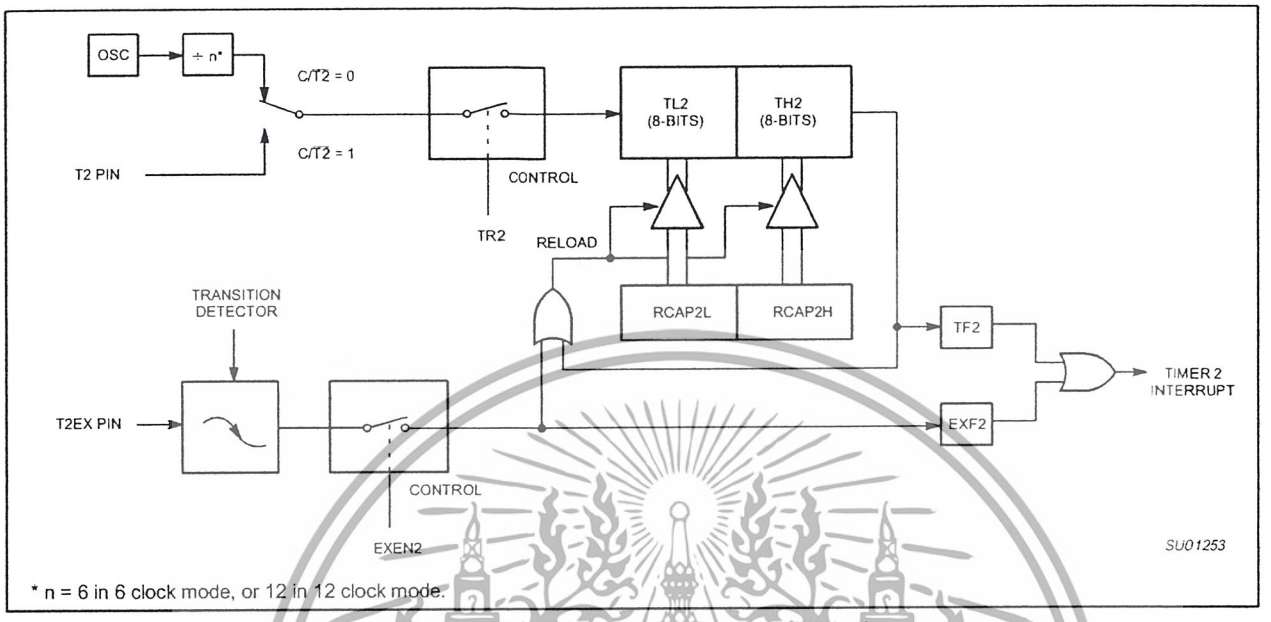


Figure 4. Timer 2 in Auto-Reload Mode (DCEN = 0)

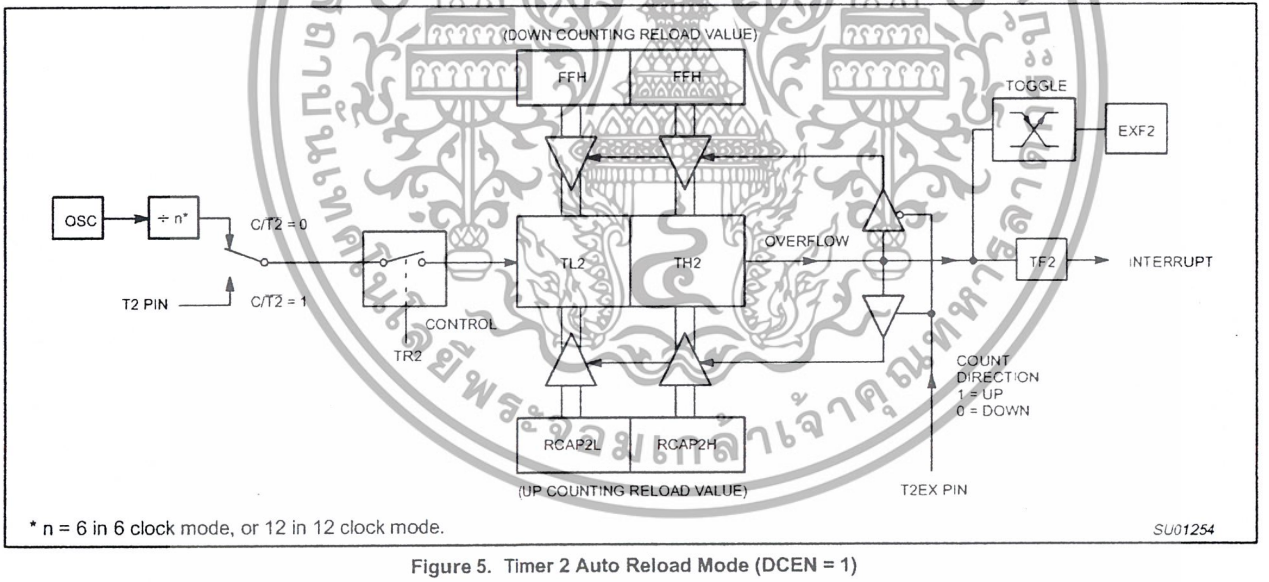


Figure 5. Timer 2 Auto Reload Mode (DCEN = 1)

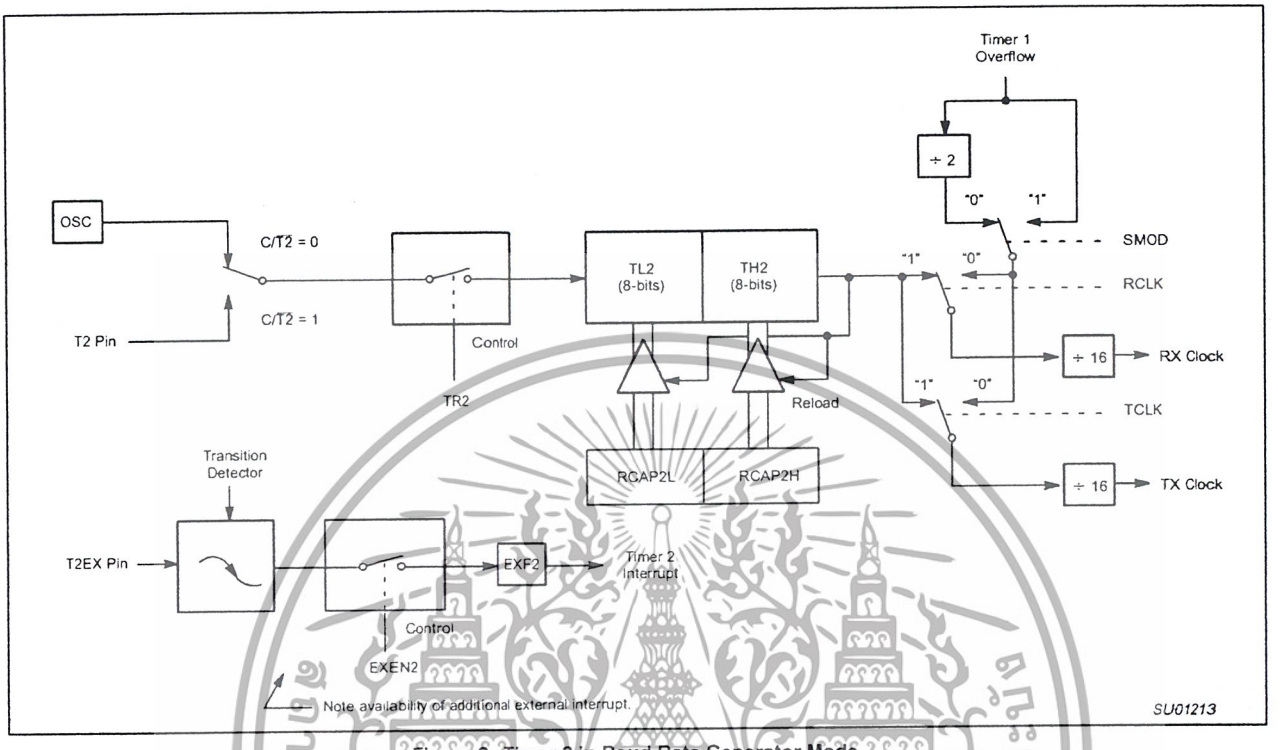


Figure 6. Timer 2 in Baud Rate Generator Mode

Table 4. Timer 2 Generated Commonly Used Baud Rates

Baud Rate		Osc Freq	Timer 2	
12 clock mode	6 clock mode		RCAP2H	RCAP2L
375 k	750 k	12 MHz	FF	FF
9.6 k	19.2 k	12 MHz	FF	D9
2.8 k	5.6 k	12 MHz	FF	B2
2.4 k	4.8 k	12 MHz	FF	64
1.2 k	2.4 k	12 MHz	FE	C8
300	600	12 MHz	FB	1E
110	220	12 MHz	F2	AF
300	600	6 MHz	FD	8F
110	220	6 MHz	F9	57

Baud Rate Generator Mode

Bits TCLK and/or RCLK in T2CON (Table 4) allow the serial port transmit and receive baud rates to be derived from either Timer 1 or Timer 2. When TCLK= 0, Timer 1 is used as the serial port transmit baud rate generator. When TCLK= 1, Timer 2 is used as the serial port transmit baud rate generator. RCLK has the same effect for the serial port receive baud rate. With these two bits, the serial port can have different receive and transmit baud rates – one generated by Timer 1, the other by Timer 2.

Figure 6 shows the Timer 2 in baud rate generation mode. The baud rate generation mode is like the auto-reload mode,in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in modes 1 and 3 are determined by Timer 2's overflow rate given below:

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The timer can be configured for either "timer" or "counter" operation. In many applications, it is configured for "timer" operation (C/T2*=0). Timer operation is different for Timer 2 when it is being used as a baud rate generator.

Usually, as a timer it would increment every machine cycle (i.e., $^1/_6$ the oscillator frequency in 6 clock mode, $^1/_{12}$ the oscillator frequency in 12 clock mode). As a baud rate generator, it increments at the oscillator frequency in 6 clock mode ($^{OSC}/_2$ in 12 clock mode). Thus the baud rate formula is as follows:

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Oscillator Frequency}}{[\,n^* \times [65536 - (\text{RCAP2H, RCAP2L})]]}$$

* n = 16 in 6 clock mode
32 in 12 clock mode

Where: (RCAP2H, RCAP2L)= The content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

The Timer 2 as a baud rate generator mode shown in Figure 6, is valid only if RCLK and/or TCLK = 1 in T2CON register. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Thus, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Also if the EXEN2 (T2 external enable flag) is set, a 1-to-0 transition in T2EX (Timer/counter 2 trigger input) will set EXF2 (T2 external flag) but will not cause a reload from (RCAP2H, RCAP2L) to (TH2,TL2). Therefore when Timer 2 is in use as a baud rate generator, T2EX can be used as an additional external interrupt, if needed.

When Timer 2 is in the baud rate generator mode, one should not try to read or write TH2 and TL2. As a baud rate generator, Timer 2 is incremented every state time (osc/2) or asynchronously from pin T2; under these conditions, a read or write of TH2 or TL2 may not be accurate. The RCAP2 registers may be read, but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

Table 4 shows commonly used baud rates and how they can be obtained from Timer 2.

Summary of Baud Rate Equations

Timer 2 is in baud rate generating mode. If Timer 2 is being clocked through pin T2(P1.0) the baud rate is:

$$\text{Baud Rate} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

If Timer 2 is being clocked internally, the baud rate is:

$$\text{Baud Rate} = \frac{f_{OSC}}{[\,n^* \times [65536 - (\text{RCAP2H, RCAP2L})]]}$$

* n = 16 in 6 clock mode
32 in 12 clock mode

Where f_{OSC}= Oscillator Frequency

To obtain the reload value for RCAP2H and RCAP2L, the above equation can be rewritten as:

$$\text{RCAP2H, RCAP2L} = 65536 - \left(\frac{f_{OSC}}{n^* \times \text{Baud Rate}}\right)$$

Timer/Counter 2 Set-up

Except for the baud rate generator mode, the values given for T2CON do not include the setting of the TR2 bit. Therefore, bit TR2 must be set, separately, to turn the timer on. see Table 5 for set-up of Timer 2 as a timer. Also see Table 6 for set-up of Timer 2 as a counter.

Table 5. Timer 2 as a Timer

MODE	T2CON	
	INTERNAL CONTROL (Note 1)	EXTERNAL CONTROL (Note 2)
16-bit Auto-Reload	00H	08H
16-bit Capture	01H	09H
Baud rate generator receive and transmit same baud rate	34H	36H
Receive only	24H	26H
Transmit only	14H	16H

Table 6. Timer 2 as a Counter

MODE	TMOD	
	INTERNAL CONTROL (Note 1)	EXTERNAL CONTROL (Note 2)
16-bit	02H	0AH
Auto-Reload	03H	0BH

NOTES:
1. Capture/reload occurs only on timer/counter overflow.
2. Capture/reload occurs on timer/counter overflow and a 1-to-0 transition on T2EX (P1.1) pin except when Timer 2 is used in the baud rate generator mode.

Enhanced UART

The UART operates in all of the usual modes that are described in the first section of *Data Handbook IC20, 80C51-Based 8-Bit Microcontrollers*. In addition the UART can perform framing error detect by looking for missing stop bits, and automatic address recognition. The UART also fully supports multiprocessor communication as does the standard 80C51 UART.

When used for framing error detect the UART looks for missing stop bits in the communication. A missing bit will set the FE bit in the SCON register. The FE bit shares the SCON.7 bit with SM0 and the function of SCON.7 is determined by PCON.6 (SMOD0) (see Figure 7). If SMOD0 is set then SCON.7 functions as FE. SCON.7 functions as SM0 when SMOD0 is cleared. When used as FE SCON.7 can only be cleared by software. Refer to Figure 8.

Automatic Address Recognition

Automatic Address Recognition is a feature which allows the UART to recognize certain addresses in the serial bit stream by using hardware to make the comparisons. This feature saves a great deal of software overhead by eliminating the need for the software to examine every serial address which passes by the serial port. This feature is enabled by setting the SM2 bit in SCON. In the 9 bit UART modes, mode 2 and mode 3, the Receive Interrupt flag (RI) will be automatically set when the received byte contains either the "Given" address or the "Broadcast" address. The 9-bit mode requires that the 9th information bit is a 1 to indicate that the received information is an address and not data. Automatic address recognition is shown in Figure 9.

The 8 bit mode is called Mode 1. In this mode the RI flag will be set if SM2 is enabled and the information received has a valid stop bit following the 8 address bits and the information is either a Given or Broadcast address.

Mode 0 is the Shift Register mode and SM2 is ignored.

Using the Automatic Address Recognition feature allows a master to selectively communicate with one or more slaves by invoking the Given slave address or addresses. All of the slaves may be contacted by using the Broadcast address. Two special Function Registers are used to define the slave's address, SADDR, and the address mask, SADEN. SADEN is used to define which bits in the SADDR are to b used and which bits are "don't care". The SADEN mask can be logically ANDed with the SADDR to create the "Given" address which the master will use for addressing each of the slaves. Use of the Given address allows multiple slaves to be recognized while excluding others. The following examples will help to show the versatility of this scheme:

Slave 0	SADDR	=	1100 0000
	SADEN	=	1111 1101
	Given	=	1100 00X0
Slave 1	SADDR	=	1100 0000
	SADEN	=	1111 1110
	Given	=	1100 000X

In the above example SADDR is the same and the SADEN data is used to differentiate between the two slaves. Slave 0 requires a 0 in bit 0 and it ignores bit 1. Slave 1 requires a 0 in bit 1 and bit 0 is ignored. A unique address for Slave 0 would be 1100 0010 since slave 1 requires a 0 in bit 1. A unique address for slave 1 would be 1100 0001 since a 1 in bit 0 will exclude slave 0. Both slaves can be selected at the same time by an address which has bit 0 = 0 (for slave 0) and bit 1 = 0 (for slave 1). Thus, both could be addressed with 1100 0000.

In a more complex system the following could be used to select slaves 1 and 2 while excluding slave 0:

Slave 0	SADDR	=	1100 0000
	SADEN	=	1111 1001
	Given	=	1100 0XX0
Slave 1	SADDR	=	1110 0000
	SADEN	=	1111 1010
	Given	=	1110 0X0X
Slave 2	SADDR	=	1110 0000
	SADEN	=	1111 1100
	Given	=	1110 00XX

In the above example the differentiation among the 3 slaves is in the lower 3 address bits. Slave 0 requires that bit 0 = 0 and it can be uniquely addressed by 1110 0110. Slave 1 requires that bit 1 = 0 and it can be uniquely addressed by 1110 and 0101. Slave 2 requires that bit 2 = 0 and its unique address is 1110 0011. To select Slaves 0 and 1 and exclude Slave 2 use address 1110 0100, since it is necessary to make bit 2 = 1 to exclude slave 2.

The Broadcast Address for each slave is created by taking the logical OR of SADDR and SADEN. Zeros in this result are trended as don't-cares. In most cases, interpreting the don't-cares as ones, the broadcast address will be FF hexadecimal.

Upon reset SADDR (SFR address 0A9H) and SADEN (SFR address 0B9H) are leaded with 0s. This produces a given address of all "don't cares" as well as a Broadcast address of all "don't cares". This effectively disables the Automatic Addressing mode and allows the microcontroller to use standard 80C51 type UART drivers which do not make use of this feature.

SCON Address = 98H — Reset Value = 0000 0000B

Bit Addressable

	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
Bit:	7	6	5	4	3	2	1	0

(SMOD0 = 0/1)*

Symbol	Function
FE	Framing Error bit. This bit is set by the receiver when an invalid stop bit is detected. The FE bit is not cleared by valid frames but should be cleared by software. The SMOD0 bit must be set to enable access to the FE bit.
SM0	Serial Port Mode Bit 0, (SMOD0 must = 0 to access bit SM0)
SM1	Serial Port Mode Bit 1

SM0	SM1	Mode	Description	Baud Rate**
0	0	0	shift register	$f_{OSC}/6$ (6 clock mode) or $f_{OSC}/12$ (12 clock mode)
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	$f_{OSC}/32$ or $f_{OSC}/16$ (6 clock mode) or $f_{OSC}/64$ or $f_{OSC}/32$ (12 clock mode)
1	1	3	9-bit UART	variable

SM2	Enables the Automatic Address Recognition feature in Modes 2 or 3. If SM2 = 1 then RI will not be set unless the received 9th data bit (RB8) is 1, indicating an address, and the received byte is a Given or Broadcast Address. In Mode 1, if SM2 = 1 then RI will not be activated unless a valid stop bit was received, and the received byte is a Given or Broadcast Address. In Mode 0, SM2 should be 0.
REN	Enables serial reception. Set by software to enable reception. Clear by software to disable reception.
TB8	The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
RB8	In modes 2 and 3, the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
TI	Transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
RI	Receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

NOTE:
*SMOD0 is located at PCON6.
**f_{OSC} = oscillator frequency

SU01255

Figure 7. SCON: Serial Port Control Register

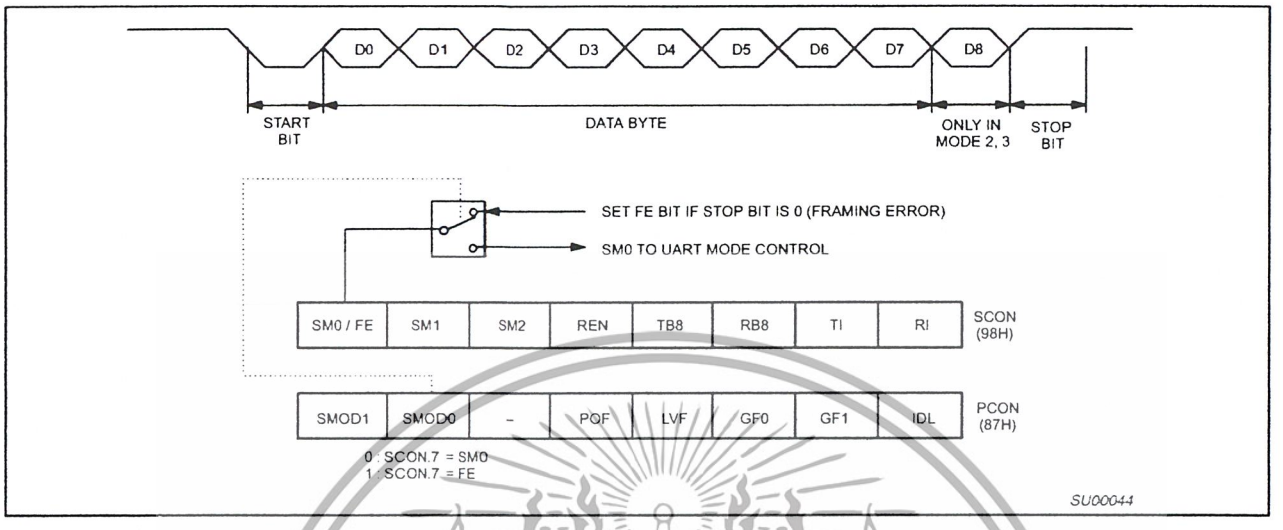


Figure 8. UART Framing Error Detection

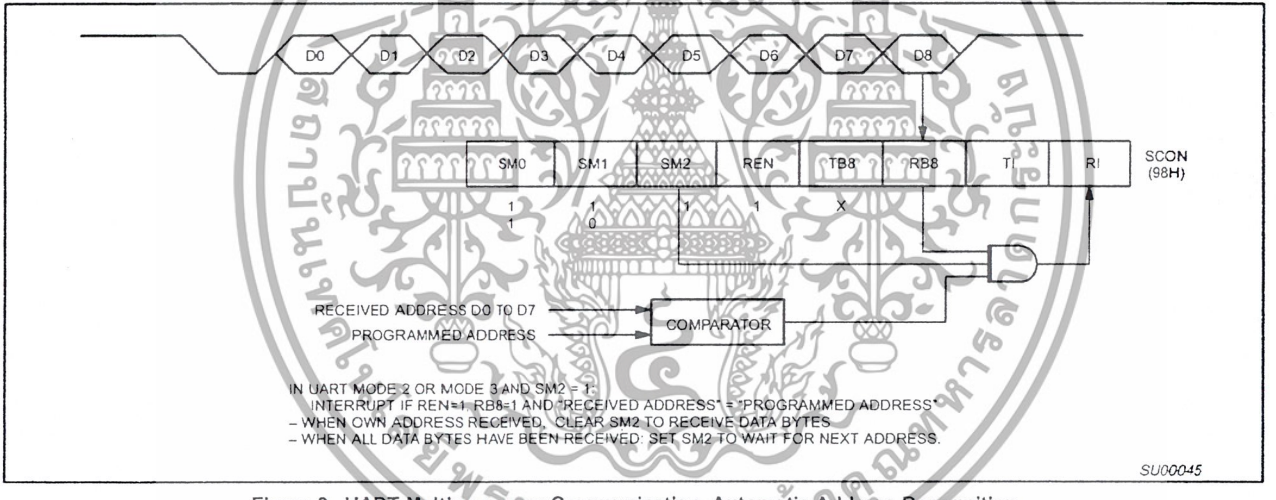


Figure 9. UART Multiprocessor Communication, Automatic Address Recognition

Interrupt Priority Structure

The P89C51RB2/RC2/RD2 has a 7 source four-level interrupt structure (see Table 7).

There are 3 SFRs associated with the four-level interrupt. They are the IE, IP, and IPH. (See Figures 10, 11, and 12.) The IPH (Interrupt Priority High) register makes the four-level interrupt structure possible. The IPH is located at SFR address B7H. The structure of the IPH register and a description of its bits is shown in Figure 12.

The function of the IPH SFR, when combined with the IP SFR, determines the priority of each interrupt. The priority of each interrupt is determined as shown in the following table:

PRIORITY BITS		INTERRUPT PRIORITY LEVEL
IPH.x	IP.x	
0	0	Level 0 (lowest priority)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (highest priority)

The priority scheme for servicing the interrupts is the same as that for the 80C51, except there are four interrupt levels rather than two as on the 80C51. An interrupt will be serviced as long as an interrupt of equal or higher priority is not already being serviced. If an interrupt of equal or higher level priority is being serviced, the new interrupt will wait until it is finished before being serviced. If a lower priority level interrupt is being serviced, it will be stopped and the new interrupt serviced. When the new interrupt is finished, the lower priority level interrupt that was stopped will be completed.

Table 7. Interrupt Table

SOURCE	POLLING PRIORITY	REQUEST BITS	HARDWARE CLEAR?	VECTOR ADDRESS
X0	1	IE0	N (L)[1] Y (T)[2]	03H
T0	2	TP0	Y	0BH
X1	3	IE1	N (L) Y (T)	13H
T1	4	TF1	Y	1BH
PCA	5	CF, CCFn n = 0–4	N	33H
SP	6	RI, TI	N	23H
T2	7	TF2, EXF2	N	2BH

NOTES:
1. L = Level activated
2. T = Transition activated

IE (0A8H)

7	6	5	4	3	2	1	0
EA	EC	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.
Enable Bit = 0 disables it.

BIT	SYMBOL	FUNCTION
IE.7	EA	Global disable bit. If EA = 0, all interrupts are disabled. If EA = 1, each interrupt can be individually enabled or disabled by setting or clearing its enable bit.
IE.6	EC	PCA interrupt enable bit
IE.5	ET2	Timer 2 interrupt enable bit.
IE.4	ES	Serial Port interrupt enable bit.
IE.3	ET1	Timer 1 interrupt enable bit.
IE.2	EX1	External interrupt 1 enable bit.
IE.1	ET0	Timer 0 interrupt enable bit.
IE.0	EX0	External interrupt 0 enable bit.

SU01290

Figure 10. IE Registers

	7	6	5	4	3	2	1	0
IP (0B8H)	–	PPC	PT2	PS	PT1	PX1	PT0	PX0

Priority Bit = 1 assigns high priority
Priority Bit = 0 assigns low priority

BIT	SYMBOL	FUNCTION
IP.7	–	–
IP.6	PPC	PCA interrupt priority bit
IP.5	PT2	Timer 2 interrupt priority bit.
IP.4	PS	Serial Port interrupt priority bit.
IP.3	PT1	Timer 1 interrupt priority bit.
IP.2	PX1	External interrupt 1 priority bit.
IP.1	PT0	Timer 0 interrupt priority bit.
IP.0	PX0	External interrupt 0 priority bit.

SU01291

Figure 11. IP Registers

	7	6	5	4	3	2	1	0
IPH (B7H)	–	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H

Priority Bit = 1 assigns higher priority
Priority Bit = 0 assigns lower priority

BIT	SYMBOL	FUNCTION
IPH.7	–	–
IPH.6	PPCH	PCA interrupt priority bit
IPH.5	PT2H	Timer 2 interrupt priority bit high.
IPH.4	PSH	Serial Port interrupt priority bit high.
IPH.3	PT1H	Timer 1 interrupt priority bit high.
IPH.2	PX1H	External interrupt 1 priority bit high.
IPH.1	PT0H	Timer 0 interrupt priority bit high.
IPH.0	PX0H	External interrupt 0 priority bit high.

SU01292

Figure 12. iPH Registers

Reduced EMI Mode

The AO bit (AUXR.0) in the AUXR register when set disables the ALE output.

Reduced EMI Mode

AUXR (8EH)

7	6	5	4	3	2	1	0
–	–	–	–	–	–	EXTRAM	AO

AUXR.1 EXTRAM
AUXR.0 AO Turns off ALE output.

Dual DPTR

The dual DPTR structure (see Figure 13) is a way by which the chip will specify the address of an external data memory location. There are two 16-bit DPTR registers that address the external memory, and a single bit called DPS = AUXR1/bit0 that allows the program code to switch between them.

- New Register Name: AUXR1#
- SFR Address: A2H
- Reset Value: xxxxxxx0B

AUXR1 (A2H)

7	6	5	4	3	2	1	0
–	–	ENBOOT	–	GF2	0	–	DPS

Where:
DPS = AUXR1/bit0 = Switches between DPTR0 and DPTR1.

Select Reg	DPS
DPTR0	0
DPTR1	1

The DPS bit status should be saved by software when switching between DPTR0 and DPTR1.

The GF2 bit is a general purpose user-defined flag. Note that bit 2 is not writable and is always read as a zero. This allows the DPS bit to be quickly toggled simply by executing an INC AUXR1 instruction without affecting the GF2 bit.

The ENBOOT bit determines whether the BOOTROM is enabled or disabled. This bit will automatically be set if the status byte is non zero during reset or $\overline{PSEN}$ is pulled low, ALE floats high, and EA > V_{IH} on the falling edge of reset. Otherwise, this bit will be cleared during reset.

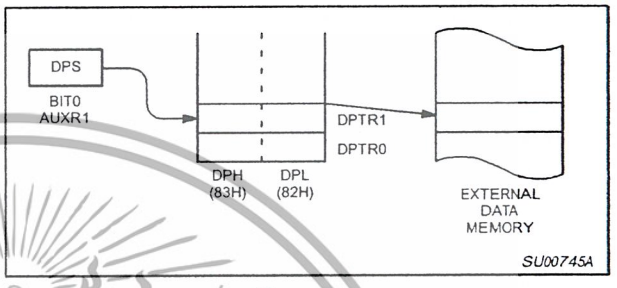


Figure 13.

DPTR Instructions

The instructions that refer to DPTR refer to the data pointer that is currently selected using the AUXR1/bit 0 register. The six instructions that use the DPTR are as follows:

INC DPTR	Increments the data pointer by 1
MOV DPTR, #data16	Loads the DPTR with a 16-bit constant
MOV A, @ A+DPTR	Move code byte relative to DPTR to ACC
MOVX A, @ DPTR	Move external RAM (16-bit address) to ACC
MOVX @ DPTR , A	Move ACC to external RAM (16-bit address)
JMP @ A + DPTR	Jump indirect relative to DPTR

The data pointer can be accessed on a byte-by-byte basis by specifying the low or high byte in an instruction which accesses the SFRs. See *Application Note AN458* for more details.

Programmable Counter Array (PCA)

The Programmable Counter Array available on the 89C51RB2/RC2/RD2 is a special 16-bit Timer that has five 16-bit capture/compare modules associated with it. Each of the modules can be programmed to operate in one of four modes: rising and/or falling edge capture, software timer, high-speed output, or pulse width modulator. Each module has a pin associated with it in port 1. Module 0 is connected to P1.3(CEX0), module 1 to P1.4(CEX1), etc. The basic PCA configuration is shown in Figure 14.

The PCA timer is a common time base for all five modules and can be programmed to run at: 1/6 the oscillator frequency, 1/2 the oscillator frequency, the Timer 0 overflow, or the input on the ECI pin (P1.2). The timer count source is determined from the CPS1 and CPS0 bits in the CMOD SFR as follows (see Figure 17):

CPS1	CPS0	PCA Timer Count Source
0	0	1/6 oscillator frequency (6 clock mode); 1/12 oscillator frequency (12 clock mode)
0	1	1/2 oscillator frequency (6 clock mode); 1/4 oscillator frequency (12 clock mode)
1	0	Timer 0 overflow
1	1	External Input at ECI pin

In the CMOD SFR are three additional bits associated with the PCA. They are CIDL which allows the PCA to stop during idle mode, WDTE which enables or disables the watchdog function on module 4, and ECF which when set causes an interrupt and the PCA overflow flag CF (in the CCON SFR) to be set when the PCA timer overflows. These functions are shown in Figure 15.

The watchdog timer function is implemented in module 4 (see Figure 24).

The CCON SFR contains the run control bit for the PCA and the flags for the PCA timer (CF) and each module (refer to Figure 18). To run the PCA the CR bit (CCON.6) must be set by software. The PCA is shut off by clearing this bit. The CF bit (CCON.7) is set when the PCA counter overflows and an interrupt will be generated if the ECF bit in the CMOD register is set, The CF bit can only be cleared by software. Bits 0 through 4 of the CCON register are the flags for the modules (bit 0 for module 0, bit 1 for module 1, etc.) and are set by hardware when either a match or a capture occurs. These flags also can only be cleared by software. The PCA interrupt system shown in Figure 16.

Each module in the PCA has a special function register associated with it. These registers are: CCAPM0 for module 0, CCAPM1 for module 1, etc. (see Figure 19). The registers contain the bits that control the mode that each module will operate in. The ECCF bit (CCAPMn.0 where n=0, 1, 2, 3, or 4 depending on the module) enables the CCF flag in the CCON SFR to generate an interrupt when a match or compare occurs in the associated module. PWM (CCAPMn.1) enables the pulse width modulation mode. The TOG bit (CCAPMn.2) when set causes the CEX output associated with the module to toggle when there is a match between the PCA counter and the module's capture/compare register. The match bit MAT (CCAPMn.3) when set will cause the CCFn bit in the CCON register to be set when there is a match between the PCA counter and the module's capture/compare register.

The next two bits CAPN (CCAPMn.4) and CAPP (CCAPMn.5) determine the edge that a capture input will be active on. The CAPN bit enables the negative edge, and the CAPP bit enables the positive edge. If both bits are set both edges will be enabled and a capture will occur for either transition. The last bit in the register ECOM (CCAPMn.6) when set enables the comparator function. Figure 20 shows the CCAPMn settings for the various PCA functions.

There are two additional registers associated with each of the PCA modules. They are CCAPnH and CCAPnL and these are the registers that store the 16-bit count when a capture occurs or a compare should occur. When a module is used in the PWM mode these registers are used to control the duty cycle of the output.

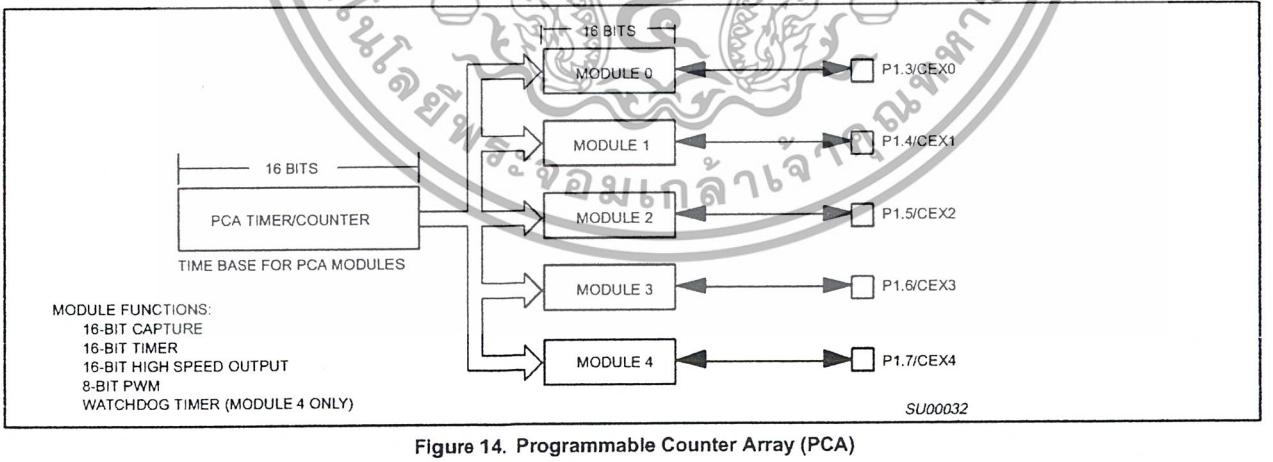


Figure 14. Programmable Counter Array (PCA)

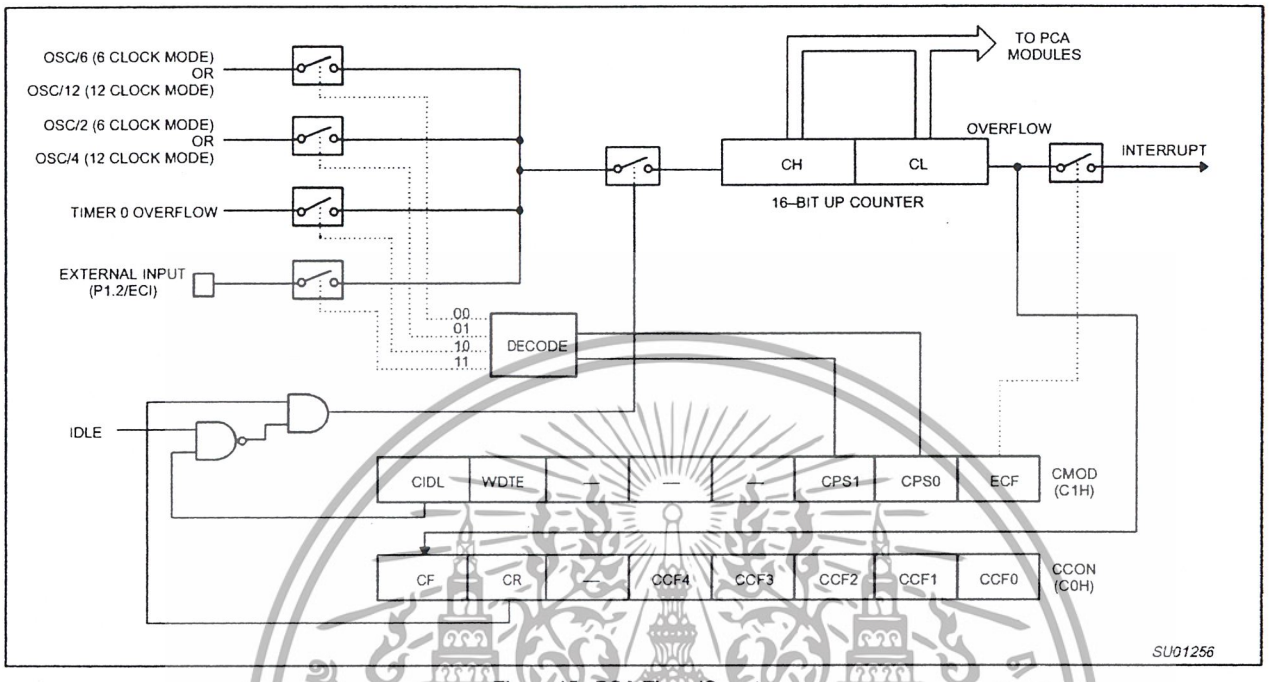


Figure 15. PCA Timer/Counter

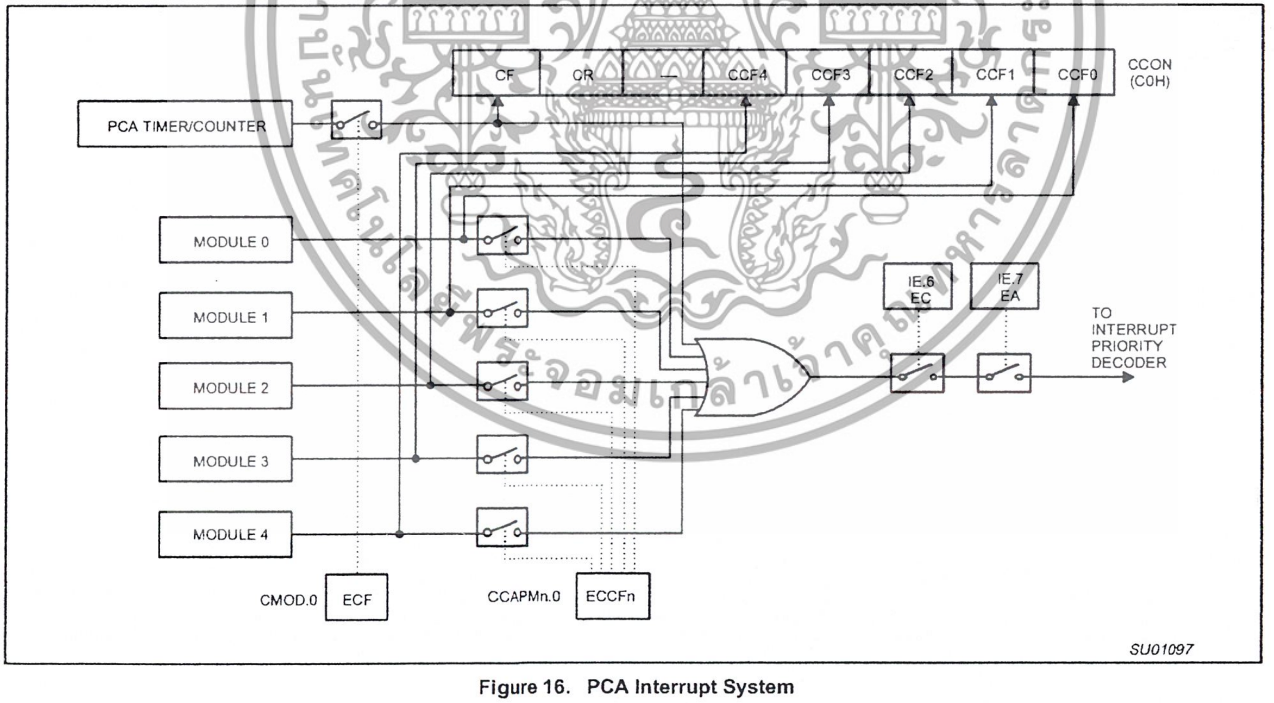


Figure 16. PCA Interrupt System

CMOD Address = D9H — Reset Value = 00XX X000B

Bit:	7	6	5	4	3	2	1	0
	CIDL	WDTE	–	–	–	CPS1	CPS0	ECF

Symbol	Function
CIDL	Counter Idle control: CIDL = 0 programs the PCA Counter to continue functioning during idle Mode. CIDL = 1 programs it to be gated off during idle.
WDTE	Watchdog Timer Enable: WDTE = 0 disables Watchdog Timer function on PCA Module 4. WDTE = 1 enables it.
–	Not implemented, reserved for future use.*
CPS1	PCA Count Pulse Select bit 1.
CPS0	PCA Count Pulse Select bit 0.

CPS1	CPS0	Selected PCA Input**	
0	0	0	Internal clock, f_{OSC}/6 in 6 clock mode (f_{OSC}/12 in 12 clock mode)
0	1	1	Internal clock, f_{OSC}/2 in 6 clock mode (f_{OSC}/4 in 12 clock mode)
1	0	2	Timer 0 overflow
1	1	3	External clock at ECI/P1.2 pin (max. rate = f_{OSC}/4 in 6 clock mode, f_{OCS}/8 in 12 clock mode)

ECF	PCA Enable Counter Overflow interrupt: ECF = 1 enables CF bit in CCON to generate an interrupt. ECF = 0 disables that function of CF.

NOTE:
* User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.
** f_{OSC} = oscillator frequency

SU01318

Figure 17. CMOD: PCA Counter Mode Register

CCON Address = D8H — Reset Value = 00X0 0000B

Bit Addressable

Bit:	7	6	5	4	3	2	1	0
	CF	CR	–	CCF4	CCF3	CCF2	CCF1	CCF0

Symbol	Function
CF	PCA Counter Overflow flag. Set by hardware when the counter rolls over. CF flags an interrupt if bit ECF in CMOD is set. CF may be set by either hardware or software but can only be cleared by software.
CR	PCA Counter Run control bit. Set by software to turn the PCA counter on. Must be cleared by software to turn the PCA counter off.
–	Not implemented, reserved for future use*.
CCF4	PCA Module 4 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF3	PCA Module 3 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF2	PCA Module 2 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF1	PCA Module 1 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.
CCF0	PCA Module 0 interrupt flag. Set by hardware when a match or capture occurs. Must be cleared by software.

NOTE:
* User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

SU01319

Figure 18. CCON: PCA Counter Control Register

CCAPMn Address CCAPM0 0DAH
CCAPM1 0DBH
CCAPM2 0DCH
CCAPM3 0DDH
CCAPM4 0DEH

Reset Value = X000 0000B

Not Bit Addressable

	–	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
Bit:	7	6	5	4	3	2	1	0

Symbol	Function
–	Not implemented, reserved for future use*.
ECOMn	Enable Comparator. ECOMn = 1 enables the comparator function.
CAPPn	Capture Positive, CAPPn = 1 enables positive edge capture.
CAPNn	Capture Negative, CAPNn = 1 enables negative edge capture.
MATn	Match. When MATn = 1, a match of the PCA counter with this module's compare/capture register causes the CCFn bit in CCON to be set, flagging an interrupt.
TOGn	Toggle. When TOGn = 1, a match of the PCA counter with this module's compare/capture register causes the CEXn pin to toggle.
PWMn	Pulse Width Modulation Mode. PWMn = 1 enables the CEXn pin to be used as a pulse width modulated output.
ECCFn	Enable CCF interrupt. Enables compare/capture flag CCFn in the CCON register to generate an interrupt.

NOTE:
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

SU01320

Figure 19. CCAPMn: PCA Modules Compare/Capture Registers

–	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	MODULE FUNCTION
X	0	0	0	0	0	0	0	No operation
X	X	1	0	0	0	0	X	16-bit capture by a positive-edge trigger on CEXn
X	X	0	1	0	0	0	X	16-bit capture by a negative trigger on CEXn
X	X	1	1	0	0	0	X	16-bit capture by a transition on CEXn
X	1	0	0	1	0	0	X	16-bit Software Timer
X	1	0	0	1	1	0	X	16-bit High Speed Output
X	1	0	0	0	0	1	0	8-bit PWM
X	1	0	0	1	X	0	X	Watchdog Timer

Figure 20. PCA Module Modes (CCAPMn Register)

PCA Capture Mode
To use one of the PCA modules in the capture mode either one or both of the CCAPM bits CAPN and CAPP for that module must be set. The external CEX input for the module (on port 1) is sampled for a transition. When a valid transition occurs the PCA hardware loads the value of the PCA counter registers (CH and CL) into the module's capture registers (CCAPnL and CCAPnH). If the CCFn bit for the module in the CCON SFR and the ECCFn bit in the CCAPMn SFR are set then an interrupt will be generated. Refer to Figure 21.

16-bit Software Timer Mode
The PCA modules can be used as software timers by setting both the ECOM and MAT bits in the modules CCAPMn register. The PCA timer will be compared to the module's capture registers and when a match occurs an interrupt will occur if the CCFn (CCON SFR) and the ECCFn (CCAPMn SFR) bits for the module are both set (see Figure 22).

High Speed Output Mode
In this mode the CEX output (on port 1) associated with the PCA module will toggle each time a match occurs between the PCA counter and the module's capture registers. To activate this mode the TOG, MAT, and ECOM bits in the module's CCAPMn SFR must be set (see Figure 23).

Pulse Width Modulator Mode
All of the PCA modules can be used as PWM outputs. Figure 24 shows the PWM function. The frequency of the output depends on the source for the PCA timer. All of the modules will have the same frequency of output because they all share the PCA timer. The duty cycle of each module is independently variable using the module's capture register CCAPLn. When the value of the PCA CL SFR is less than the value in the module's CCAPLn SFR the output will be low, when it is equal to or greater than the output will be high. When CL overflows from FF to 00, CCAPLn is reloaded with the value in CCAPHn. the allows updating the PWM without glitches. The PWM and ECOM bits in the module's CCAPMn register must be set to enable the PWM mode.

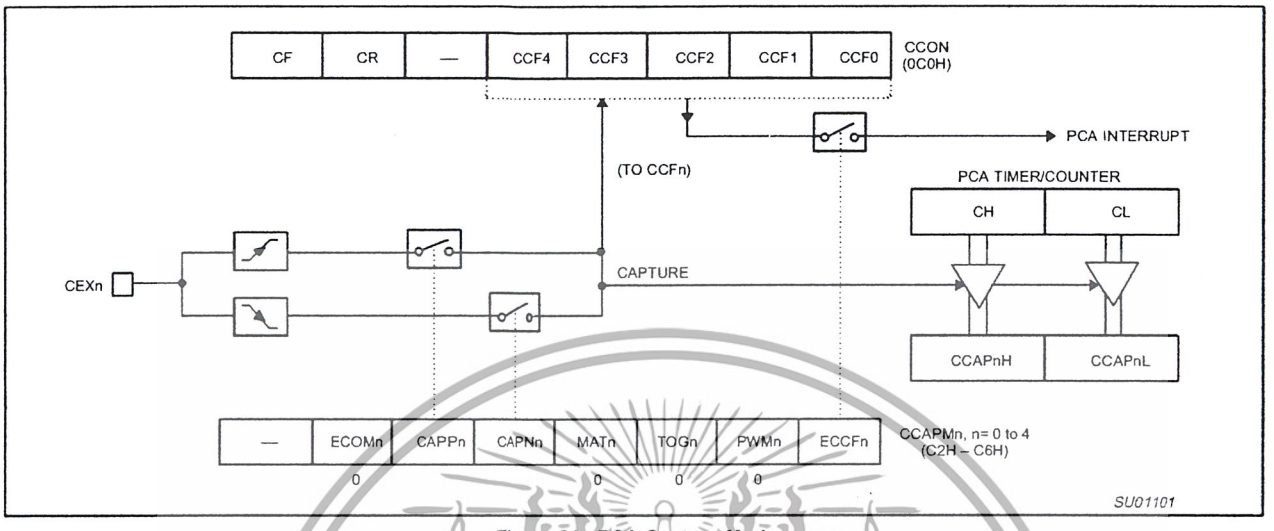


Figure 21. PCA Capture Mode

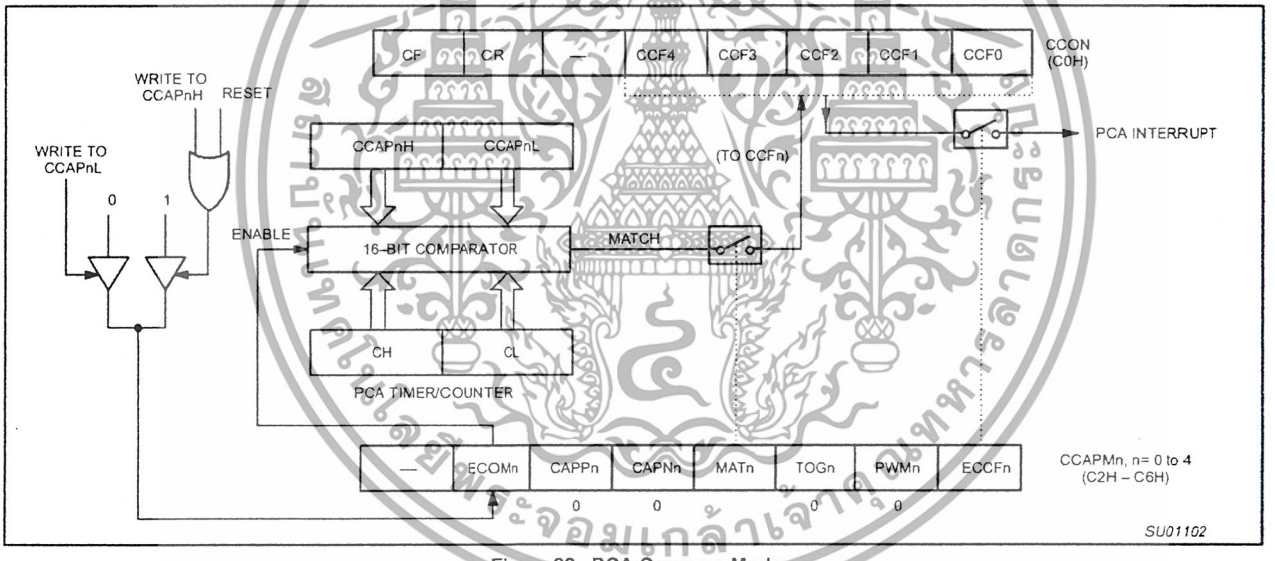


Figure 22. PCA Compare Mode

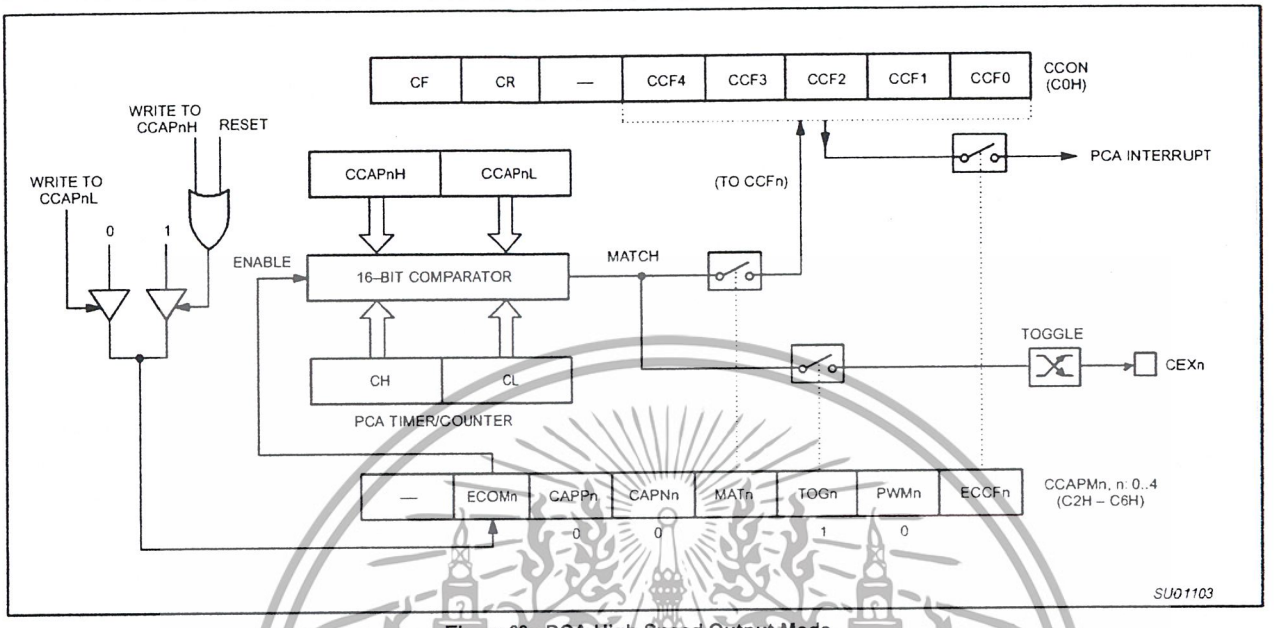


Figure 23. PCA High Speed Output Mode

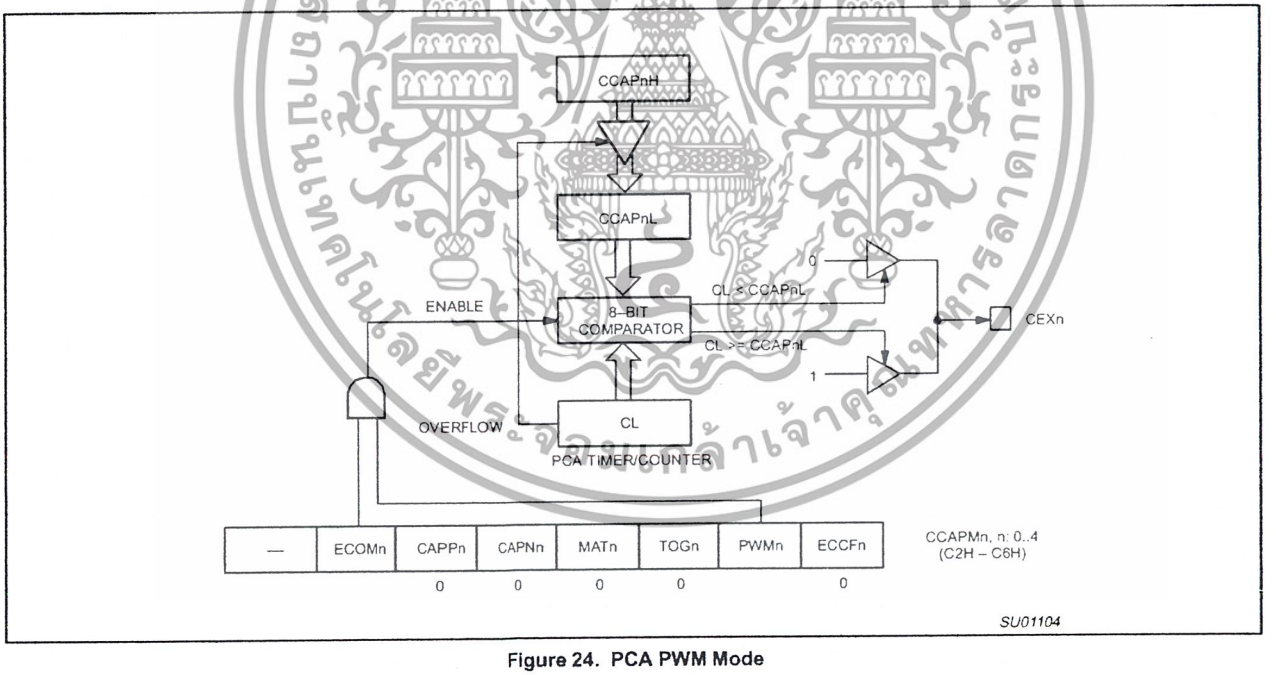


Figure 24. PCA PWM Mode

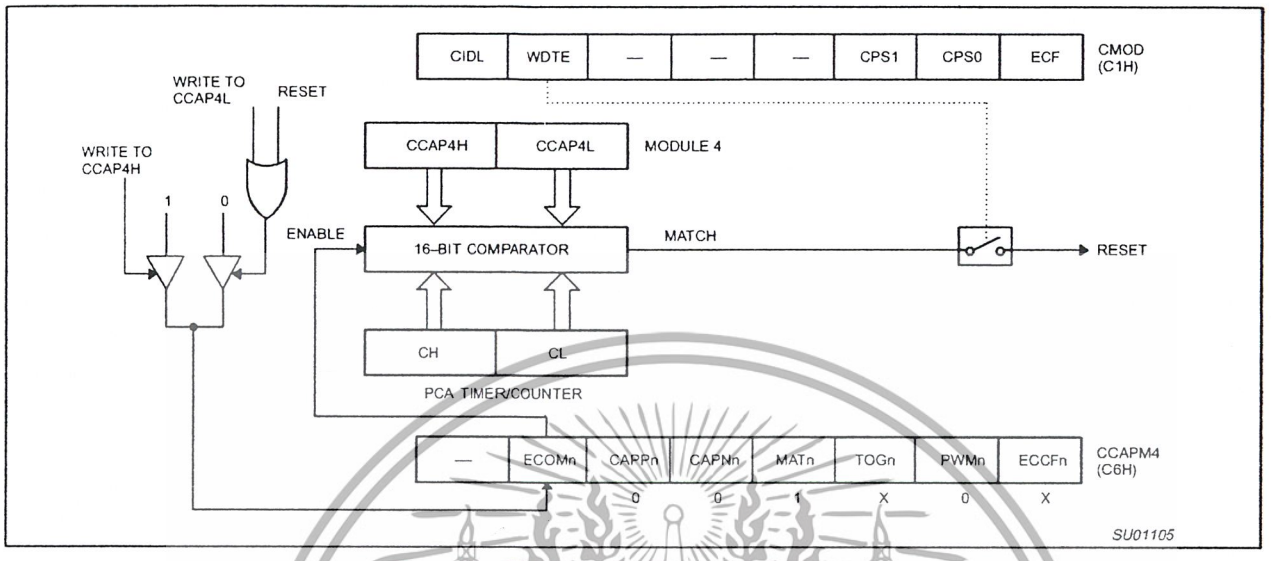


Figure 25. PCA Watchdog Timer m(Module 4 only)

PCA Watchdog Timer

An on-board watchdog timer is available with the PCA to improve the reliability of the system without increasing chip count. Watchdog timers are useful for systems that are susceptible to noise, power glitches, or electrostatic discharge. Module 4 is the only PCA module that can be programmed as a watchdog. However, this module can still be used for other modes if the watchdog is not needed.

Figure 25 shows a diagram of how the watchdog works. The user pre-loads a 16-bit value in the compare registers. Just like the other compare modes, this 16-bit value is compared to the PCA timer value. If a match is allowed to occur, an internal reset will be generated. This will not cause the RST pin to be driven high.

In order to hold off the reset, the user has three options:

1. periodically change the compare value so it will never match the PCA timer,
2. periodically change the PCA timer value so it will never match the compare values, or
3. disable the watchdog by clearing the WDTE bit before a match occurs and then re-enable it.

The first two options are more reliable because the watchdog timer is never disabled as in option #3. If the program counter ever goes astray, a match will eventually occur and cause an internal reset. The second option is also not recommended if other PCA modules are being used. Remember, the PCA timer is the time base for all modules; changing the time base for other modules would not be a good idea. Thus, in most applications the first solution is the best option.

Figure 26 shows the code for initializing the watchdog timer. Module 4 can be configured in either compare mode, and the WDTE bit in CMOD must also be set. The user's software then must periodically change (CCAP4H,CCAP4L) to keep a match from occurring with the PCA timer (CH,CL). This code is given in the WATCHDOG routine in Figure 26.

This routine should not be part of an interrupt service routine, because if the program counter goes astray and gets stuck in an infinite loop, interrupts will still be serviced and the watchdog will keep getting reset. Thus, the purpose of the watchdog would be defeated. Instead, call this subroutine from the main program within 2^{16} count of the PCA timer.

```
INIT_WATCHDOG:
   MOV CCAPM4, #4CH          ; Module 4 in compare mode
   MOV CCAP4L, #0FFH         ; Write to low byte first
   MOV CCAP4H, #0FFH         ; Before PCA timer counts up to
                             ; FFFF Hex, these compare values
                             ; must be changed
   ORL CMOD, #40H            ; Set the WDTE bit to enable the
                             ; watchdog timer without changing
                             ; the other bits in CMOD
;
;*******************************************************************
;
; Main program goes here, but CALL WATCHDOG periodically.
;
;*******************************************************************
;
WATCHDOG:
   CLR EA                    ; Hold off interrupts
   MOV CCAP4L, #00           ; Next compare value is within
   MOV CCAP4H, CH            ; 255 counts of the current PCA
   SETB EA                   ; timer value
   RET
```

Figure 26. PCA Watchdog Timer Initialization Code

Expanded Data RAM Addressing

The P89C51RB2/RC2/RD2 has internal data memory that is mapped into four separate segments: the lower 128 bytes of RAM, upper 128 bytes of RAM, 128 bytes Special Function Register (SFR), and 256 bytes expanded RAM (ERAM) (768 bytes for the RD2).

The four segments are:

1. The Lower 128 bytes of RAM (addresses 00H to 7FH) are directly and indirectly addressable.
2. The Upper 128 bytes of RAM (addresses 80H to FFH) are indirectly addressable only.
3. The Special Function Registers, SFRs, (addresses 80H to FFH) are directly addressable only.
4. The 256/768-bytes expanded RAM (ERAM, 00H – 1FFH/2FFH) are indirectly accessed by move external instruction, MOVX, and with the EXTRAM bit cleared, see Figure 27.

The Lower 128 bytes can be accessed by either direct or indirect addressing. The Upper 128 bytes can be accessed by indirect addressing only. The Upper 128 bytes occupy the same address space as the SFR. That means they have the same address, but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the CPU knows whether the access is to the upper 128 bytes of data RAM or to SFR space by the addressing mode used in the instruction. Instructions that use direct addressing access SFR space. For example:

MOV 0A0H,#data

accesses the SFR at location 0A0H (which is P2). Instructions that use indirect addressing access the Upper 128 bytes of data RAM.

For example:

MOV @R0,#data

where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

The ERAM can be accessed by indirect addressing, with EXTRAM bit cleared and MOVX instructions. This part of memory is physically located on-chip, logically occupies the first 7936-bytes of external data memory.

With EXTRAM = 0, the ERAM is indirectly addressed, using the MOVX instruction in combination with any of the registers R0, R1 of the selected bank or DPTR. An access to ERAM will not affect ports P0, P3.6 (WR#) and P3.7 (RD#). P2 SFR is output during external addressing. For example, with EXTRAM = 0,

MOVX @R0,#data

where R0 contains 0A0H, access the ERAM at address 0A0H rather than external memory. An access to external data memory locations higher than the ERAM will be performed with the MOVX DPTR instructions in the same way as in the standard 80C51, so with P0 and P2 as data/address bus, and P3.6 and P3.7 as write and read timing signals. Refer to Figure 28.

With EXTRAM = 1, MOVX @Ri and MOVX @DPTR will be similar to the standard 80C51. MOVX @ Ri will provide an 8-bit address multiplexed with data on Port 0 and any output port pins can be used to output higher order address bits. This is to provide the external paging capability. MOVX @DPTR will generate a 16-bit address. Port 2 outputs the high-order eight address bits (the contents of DPH) while Port 0 multiplexes the low-order eight address bits (DPL) with data. MOVX @Ri and MOVX @DPTR will generate either read or write signals on P3.6 ($\overline{WR}$) and P3.7 ($\overline{RD}$).

The stack pointer (SP) may be located anywhere in the 256 bytes RAM (lower and upper RAM) internal data memory. The stack may not be located in the ERAM.

AUXR Address = 8EH — Reset Value = xxxx xx00B

Not Bit Addressable

Bit:	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	EXTRAM	AO

Symbol	Function	
AO	Disable/Enable ALE	
	AO	**Operating Mode**
0		ALE is emitted at a constant rate of $^1/_3$ the oscillator frequency (6 clock mode; $^1/_6$ f_{OSC} in 12 clock mode).
	1	ALE is active only during a MOVX or MOVC instruction.
EXTRAM	Internal/External RAM access using MOVX @Ri/@DPTR	
	EXTRAM	**Operating Mode**
	0	Internal ERAM access using MOVX @Ri/@DPTR
	1	External data memory access.
—	Not implemented, reserved for future use*.	

NOTE:
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

SU01258

Figure 27. AUXR: Auxiliary Register

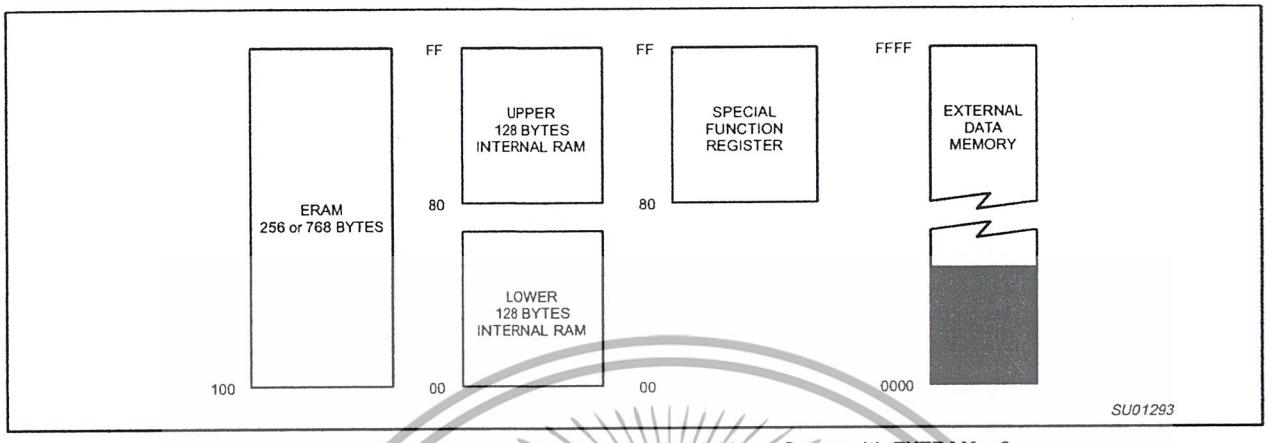


Figure 28. Internal and External Data Memory Address Space with EXTRAM = 0

HARDWARE WATCHDOG TIMER (ONE-TIME ENABLED WITH RESET-OUT FOR P89C51RB2/RC2/RD2)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upset. The WDT consists of a 14-bit counter and the WatchDog Timer reset (WDTRST) SFR. The WDT is disabled at reset. To enable the WDT, user must write 01EH and 0E1H in sequence to the WDTRST, SFR location 0A6H. When WDT is enabled, it will increment every machine cycle while the oscillator is running and there is no way to disable the WDT except through reset (either hardware reset or WDT overflow reset). When WDT overflows, it will drive an output reset HIGH pulse at the RST-pin (see the note below).

Using the WDT

To enable the WDT, user must write 01EH and 0E1H in sequence to the WDTRST, SFR location 0A6H. When WDT is enabled, the user needs to service it by writing to 01EH and 0E1H to WDTRST to avoid WDT overflow. The 14-bit counter overflows when it reaches 16383 (3FFFH) and this will reset the device. When WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 16383 machine cycles. To reset the WDT, the user must write 01EH and 0E1H to WDTRST. WDTRST is a write only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the reset pin (see note below). The RESET pulse duration is $98 \times T_{OSC}$ (6 clock mode; 196 in 12 clock mode), where $T_{OSC} = 1/f_{OSC}$. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

ABSOLUTE MAXIMUM RATINGS[1, 2, 3]

PARAMETER	RATING	UNIT
Operating temperature under bias	0 to +70 or –40 to +85	°C
Storage temperature range	–65 to +150	°C
Voltage on $\overline{EA}/V_{PP}$ pin to V_{SS}	0 to +13.0	V
Voltage on any other pin to V_{SS}	–0.5 to +6.5	V
Maximum I_{OL} per I/O pin	15	mA
Power dissipation (based on package heat transfer limitations, not device power consumption)	1.5	W

NOTES:

1. Stresses above 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 conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maximum.
3. Parameters are valid over operating temperature range unless otherwise specified. All voltages are with respect to V_{SS} unless otherwise noted.

DC ELECTRICAL CHARACTERISTICS

T_{amb} = 0°C to +70°C; V_{CC} = 5 V ± 10% or –40°C to +85°C; 5 V ±5%; V_{SS} = 0 V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			MIN	TYP[1]	MAX	
V_{IL}	Input low voltage	4.5 V < V_{CC} < 5.5 V	–0.5		$0.2V_{CC}$–0.1	V
V_{IH}	Input high voltage (ports 0, 1, 2, 3, $\overline{EA}$)		$0.2V_{CC}$+0.9		V_{CC}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{CC}$		V_{CC}+0.5	V
V_{OL}	Output low voltage, ports 1, 2, 3[8]	V_{CC} = 4.5 V I_{OL} = 1.6 mA[2]			0.4	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$ [7, 8]	V_{CC} = 4.5 V I_{OL} = 3.2 mA[2]			0.45	V
V_{OH}	Output high voltage, ports 1, 2, 3 [3]	V_{CC} = 4.5 V I_{OH} = –30 µA	V_{CC} – 0.7			V
V_{OH1}	Output high voltage (port 0 in external bus mode), ALE[9], $\overline{PSEN}$[3]	V_{CC} = 4.5 V I_{OH} = –3.2 mA	V_{CC} – 0.7			V
I_{IL}	Logical 0 input current, ports 1, 2, 3	V_{IN} = 0.4 V	–1		–75	µA
I_{TL}	Logical 1-to-0 transition current, ports 1, 2, 3[6]	V_{IN} = 2.0 V See Note 4			–650	µA
I_{LI}	Input leakage current, port 0	0.45 < V_{IN} < V_{CC} – 0.3			±10	µA
I_{CC}	Power supply current (see Figure 36): Active mode (see Note 5) Idle mode (see Note 5) Power-down mode or clock stopped (see Figure 42 for conditions) Programming and erase mode	See Note 5 T_{amb} = 0°C to 70°C T_{amb} = –40°C to +85°C f_{osc} = 20 MHz		 < 1 60	 40 50 	 µA µA mA
R_{RST}	Internal reset pull-down resistor		40		225	kΩ
C_{IO}	Pin capacitance[10] (except $\overline{EA}$)				15	pF

NOTES:

1. Typical ratings are not guaranteed. The values listed are at room temperature, 5 V.
2. Capacitive loading on ports 0 and 2 may cause spurious noise to be superimposed on the V_{OL}s of ALE and ports 1 and 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE pin may exceed 0.8 V. In such cases, it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input. I_{OL} can exceed these conditions provided that no single output sinks more than 5 mA and no more than two outputs exceed the test conditions.
3. Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the V_{CC}–0.7 specification when the address bits are stabilizing.
4. Pins of ports 1, 2 and 3 source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2 V.
5. See Figures 39 through 42 for I_{CC} test conditions and Figure 36 for I_{CC} vs Freq.
   Active mode: $I_{CC(MAX)}$ = (2.8 × FREQ. + 8)mA for all devices, in 6 clock mode; (1.4 × FREQ. + 8)mA in 12 clock mode.
   Idle mode: $I_{CC(MAX)}$ = (1.2 × FREQ. +1.0)mA in 6 clock mode; (0.6 × FREQ. +1.0)mA in 12 clock mode.
6. This value applies to T_{amb} = 0°C to +70°C.
7. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100 pF, load capacitance for all other outputs = 80 pF.
8. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 15 mA (*NOTE: This is 85°C specification.)
   Maximum I_{OL} per 8-bit port: 26 mA
   Maximum total I_{OL} for all outputs: 71 mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
9. ALE is tested to V_{OH1}, except when ALE is off then V_{OH} is the voltage specification.
10. Pin capacitance is characterized but not tested. Pin capacitance is less than 25 pF. Pin capacitance of ceramic package is less than 15 pF (except $\overline{EA}$ is 25 pF).

AC ELECTRICAL CHARACTERISTICS (6 CLOCK MODE)

T_{amb} = 0°C to +70°C; V_{CC} = 5 V ± 10% or –40°C to +85°C, V_{CC} = 5 V ±5%, V_{SS} = 0V[1, 2, 3]

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK[4]		20 MHz CLOCK[4]		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	29	Oscillator frequency	0	20	0	20	MHz
t_{LHLL}	29	ALE pulse width	t_{CLCL}–40		10		ns
t_{AVLL}	29	Address valid to ALE low	$0.5t_{CLCL}$–20		5		ns
t_{LLAX}	29	Address hold after ALE low	$0.5t_{CLCL}$–20		5		ns
t_{LLIV}	29	ALE low to valid instruction in		$2t_{CLCL}$–65		35	ns
t_{LLPL}	29	ALE low to $\overline{PSEN}$ low	$0.5t_{CLCL}$–20		5		ns
t_{PLPH}	29	$\overline{PSEN}$ pulse width	$1.5t_{CLCL}$–45		30		ns
t_{PLIV}	29	$\overline{PSEN}$ low to valid instruction in		$1.5t_{CLCL}$–60		15	ns
t_{PXIX}	29	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	29	Input instruction float after $\overline{PSEN}$		$0.5t_{CLCL}$–20		5	ns
t_{AVIV}	29	Address to valid instruction in		$2.5t_{CLCL}$–80		45	ns
t_{PLAZ}	29	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	30, 31	$\overline{RD}$ pulse width	$3t_{CLCL}$–100		50		ns
t_{WLWH}	30, 31	$\overline{WR}$ pulse width	$3t_{CLCL}$–100		50		ns
t_{RLDV}	30, 31	$\overline{RD}$ low to valid data in		$2.5t_{CLCL}$–90		35	ns
t_{RHDX}	30, 31	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	30, 31	Data float after $\overline{RD}$		t_{CLCL}–20		5	ns
t_{LLDV}	30, 31	ALE low to valid data in		$4t_{CLCL}$–150		50	ns
t_{AVDV}	30, 31	Address to valid data in		$4.5t_{CLCL}$–165		60	ns
t_{LLWL}	30, 31	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$1.5t_{CLCL}$–50	$1.5t_{CLCL}$+50	25	125	ns
t_{AVWL}	30, 31	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$2t_{CLCL}$–75		25		ns
t_{QVWX}	30, 31	Data valid to $\overline{WR}$ transition	$0.5t_{CLCL}$–25		0		ns
t_{WHQX}	30, 31	Data hold after $\overline{WR}$	$0.5t_{CLCL}$–20		5		ns
t_{QVWH}	31	Data valid to $\overline{WR}$ high	$3.5t_{CLCL}$–130		45		ns
t_{RLAZ}	30, 31	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	30, 31	$\overline{RD}$ or $\overline{WR}$ high to ALE high	$0.5t_{CLCL}$–20	$0.5t_{CLCL}$+20	5	45	ns
External Clock							
t_{CHCX}	33	High time	20	t_{CLCL}–t_{CLCX}			ns
t_{CLCX}	33	Low time	20	t_{CLCL}–t_{CHCX}			ns
t_{CLCH}	33	Rise time		5			ns
t_{CHCL}	33	Fall time		5			ns
Shift Register							
t_{XLXL}	32	Serial port clock cycle time	$6t_{CLCL}$		300		ns
t_{QVXH}	32	Output data setup to clock rising edge	$5t_{CLCL}$–133		117		ns
t_{XHQX}	32	Output data hold after clock rising edge	t_{CLCL}–30		20		ns
t_{XHDX}	32	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	32	Clock rising edge to input data valid		$5t_{CLCL}$–133		117	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100 pF, load capacitance for all other outputs = 80 pF.
3. Interfacing the microcontroller to devices with float times up to 45 ns is permitted. This limited bus contention will not cause damage to Port 0 drivers.
4. Parts are tested to 2 MHz, but are guaranteed to operate down to 0 Hz.

AC ELECTRICAL CHARACTERISTICS (12 CLOCK MODE)

T_{amb} = 0°C to +70°C; VCC = 5 V ± 10% or –40°C to +85°C, V_{CC} = 5 V ±5%, V_{SS} = 0 V[1, 2, 3]

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK[4]		33 MHz CLOCK[4]		UNIT
			MIN	MAX	MIN	MAX	
$1/t_{CLCL}$	29	Oscillator frequency	0	33	0	33	MHz
t_{LHLL}	29	ALE pulse width	$2t_{CLCL}-40$		21		ns
t_{AVLL}	29	Address valid to ALE low	$t_{CLCL}-25$		5		ns
t_{LLAX}	29	Address hold after ALE low	$t_{CLCL}-25$		5		ns
t_{LLIV}	29	ALE low to valid instruction in		$4t_{CLCL}-65$		55	ns
t_{LLPL}	29	ALE low to $\overline{PSEN}$ low	$t_{CLCL}-25$		5		ns
t_{PLPH}	29	$\overline{PSEN}$ pulse width	$3t_{CLCL}-45$		45		ns
t_{PLIV}	29	$\overline{PSEN}$ low to valid instruction in		$3t_{CLCL}-60$		30	ns
t_{PXIX}	29	Input instruction hold after $\overline{PSEN}$	0		0		ns
t_{PXIZ}	29	Input instruction float after $\overline{PSEN}$		$t_{CLCL}-25$		5	ns
t_{AVIV}	29	Address to valid instruction in		$5t_{CLCL}-80$		70	ns
t_{PLAZ}	29	$\overline{PSEN}$ low to address float		10		10	ns
Data Memory							
t_{RLRH}	30, 31	$\overline{RD}$ pulse width	$6t_{CLCL}-100$		82		ns
t_{WLWH}	30, 31	$\overline{WR}$ pulse width	$6t_{CLCL}-100$		82		ns
t_{RLDV}	30, 31	$\overline{RD}$ low to valid data in		$5t_{CLCL}-90$		60	ns
t_{RHDX}	30, 31	Data hold after $\overline{RD}$	0		0		ns
t_{RHDZ}	30, 31	Data float after $\overline{RD}$		$2t_{CLCL}-28$		32	ns
t_{LLDV}	30, 31	ALE low to valid data in		$8t_{CLCL}-150$		90	ns
t_{AVDV}	30, 31	Address to valid data in		$9t_{CLCL}-165$		105	ns
t_{LLWL}	30, 31	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLCL}-50$	$3t_{CLCL}+50$	40	140	ns
t_{AVWL}	30, 31	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLCL}-75$		45		ns
t_{QVWX}	30, 31	Data valid to $\overline{WR}$ transition	$t_{CLCL}-30$		0		ns
t_{WHQX}	30, 31	Data hold after $\overline{WR}$	$t_{CLCL}-25$		5		ns
t_{QVWH}	31	Data valid to $\overline{WR}$ high	$7t_{CLCL}-130$		80		ns
t_{RLAZ}	30, 31	$\overline{RD}$ low to address float		0		0	ns
t_{WHLH}	30, 31	$\overline{RD}$ or $\overline{WR}$ high to ALE high	$t_{CLCL}-25$	$t_{CLCL}+25$	5	55	ns
External Clock							
t_{CHCX}	33	High time	17	$t_{CLCL}-t_{CLCX}$			ns
t_{CLCX}	33	Low time	17	$t_{CLCL}-t_{CHCX}$			ns
t_{CLCH}	33	Rise time		5			ns
t_{CHCL}	33	Fall time		5			ns
Shift Register							
t_{XLXL}	32	Serial port clock cycle time	$12t_{CLCL}$		360		ns
t_{QVXH}	32	Output data setup to clock rising edge	$10t_{CLCL}-133$		167		ns
t_{XHQX}	32	Output data hold after clock rising edge	$2t_{CLCL}-80$		50		ns
t_{XHDX}	32	Input data hold after clock rising edge	0		0		ns
t_{XHDV}	32	Clock rising edge to input data valid		$10t_{CLCL}-133$		167	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100 pF, load capacitance for all other outputs = 80 pF.
3. Interfacing the microcontroller to devices with float times up to 45 ns is permitted. This limited bus contention will not cause damage to Port 0 drivers.
4. Parts are tested to 3.5 MHz, but guaranteed to operate down to 0 Hz.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{PSEN}$
Q – Output data
R – $\overline{RD}$ signal
t – Time
V – Valid
W – $\overline{WR}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{PSEN}$ low.

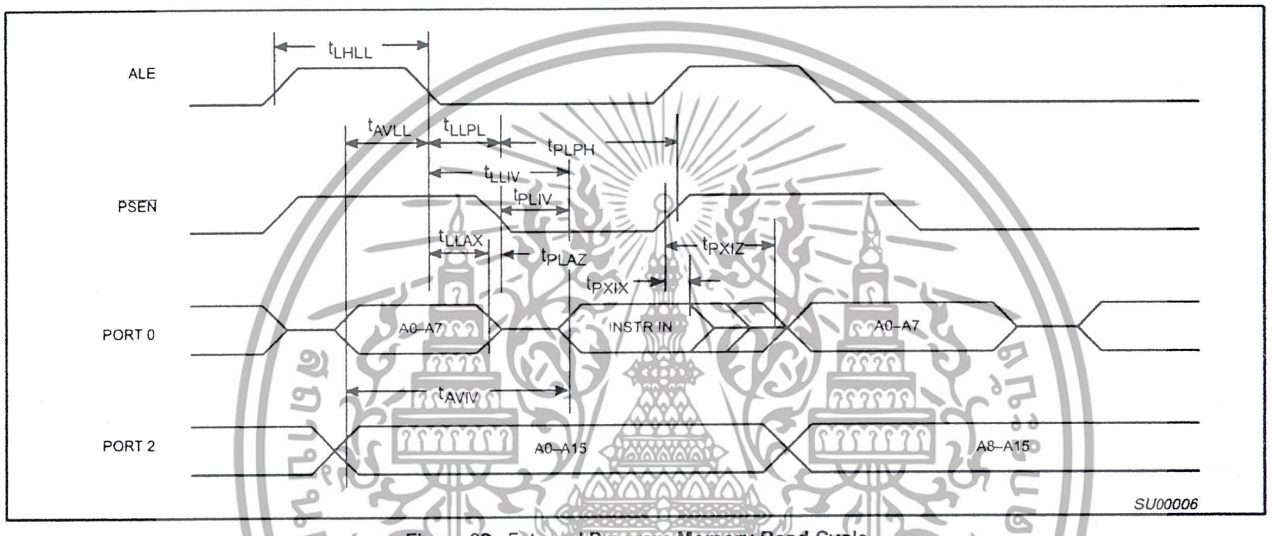


Figure 29. External Program Memory Read Cycle

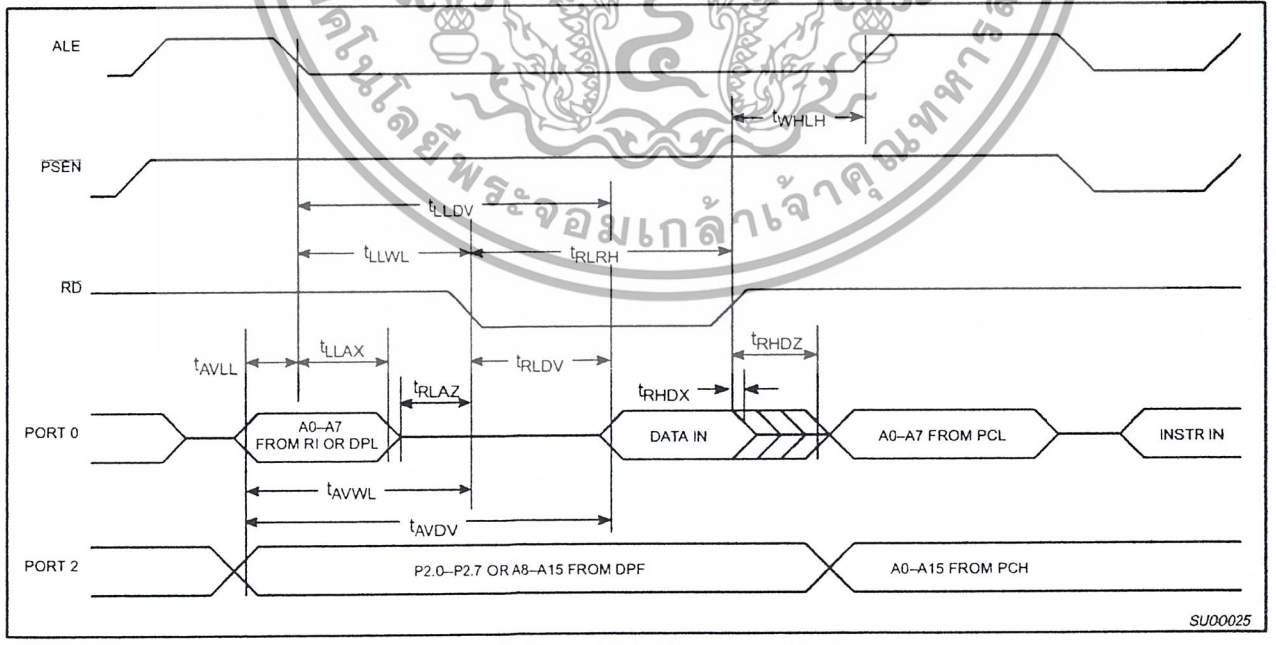


Figure 30. External Data Memory Read Cycle

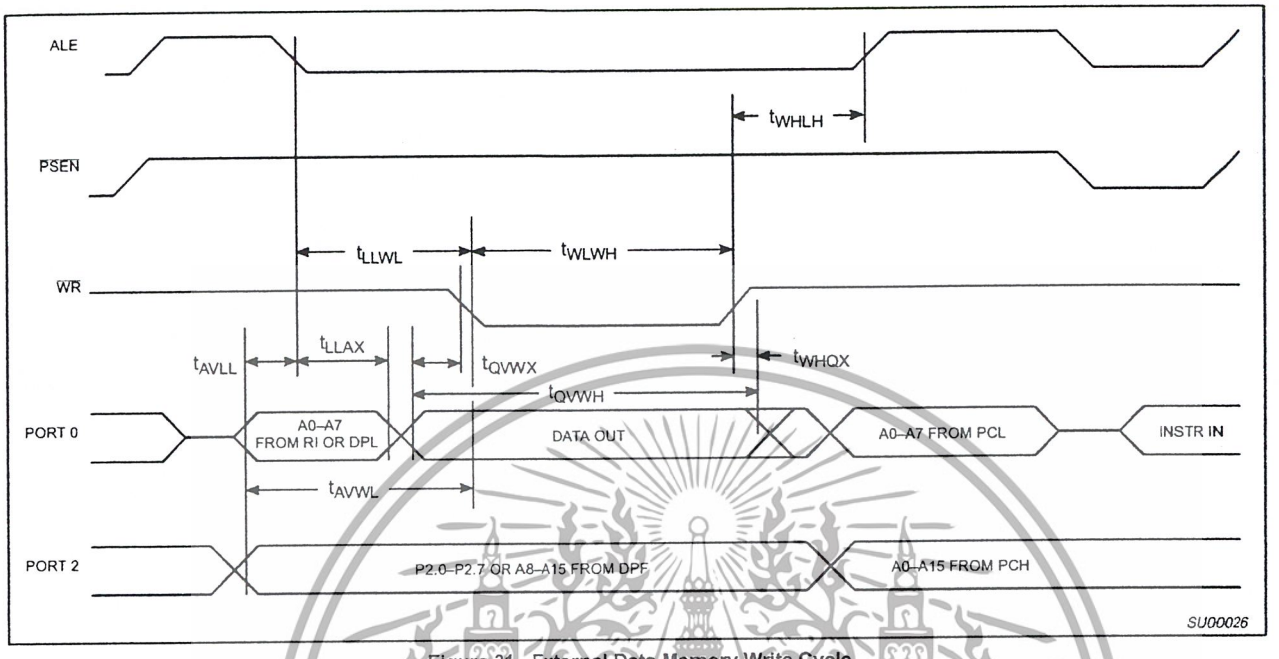


Figure 31. External Data Memory Write Cycle

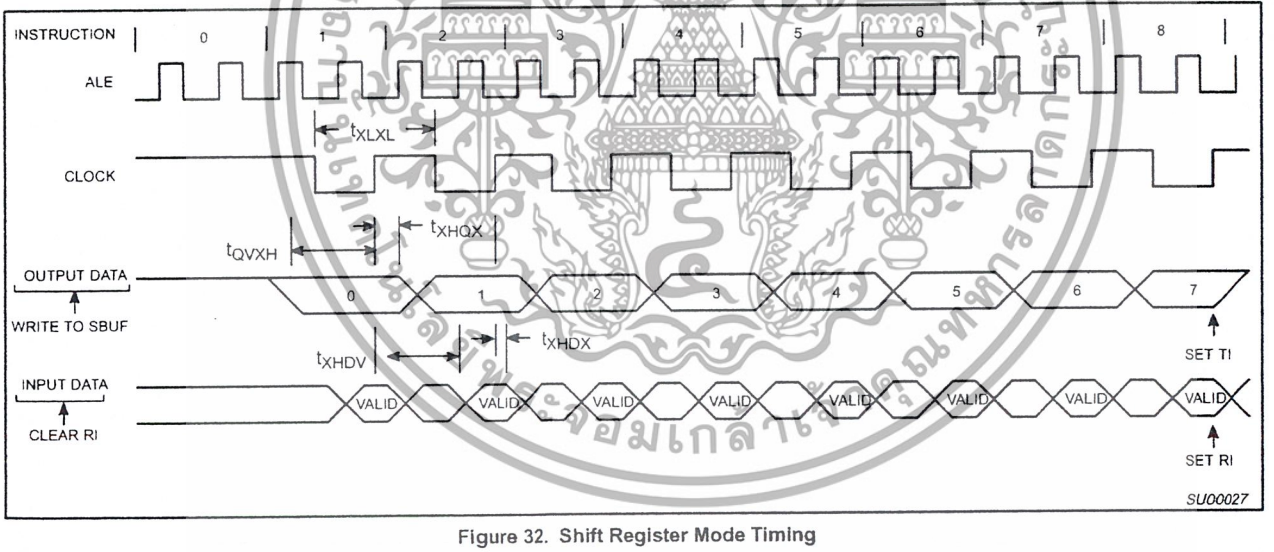


Figure 32. Shift Register Mode Timing

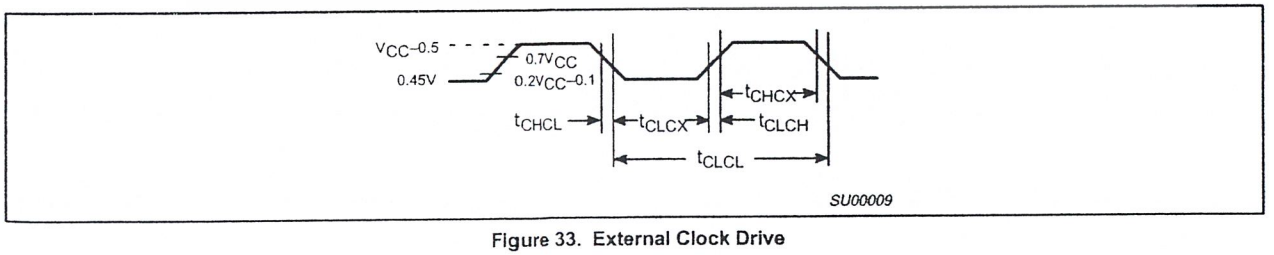


Figure 33. External Clock Drive

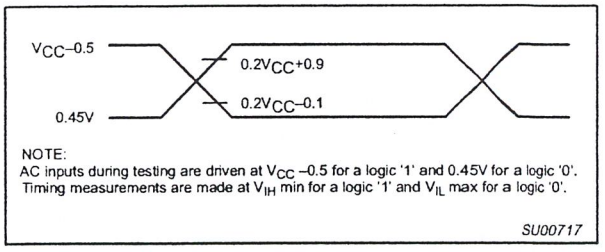


Figure 34. AC Testing Input/Output

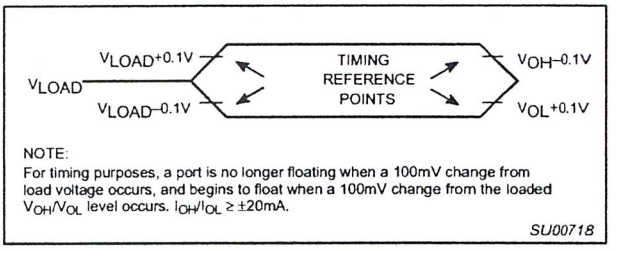


Figure 35. Float Waveform

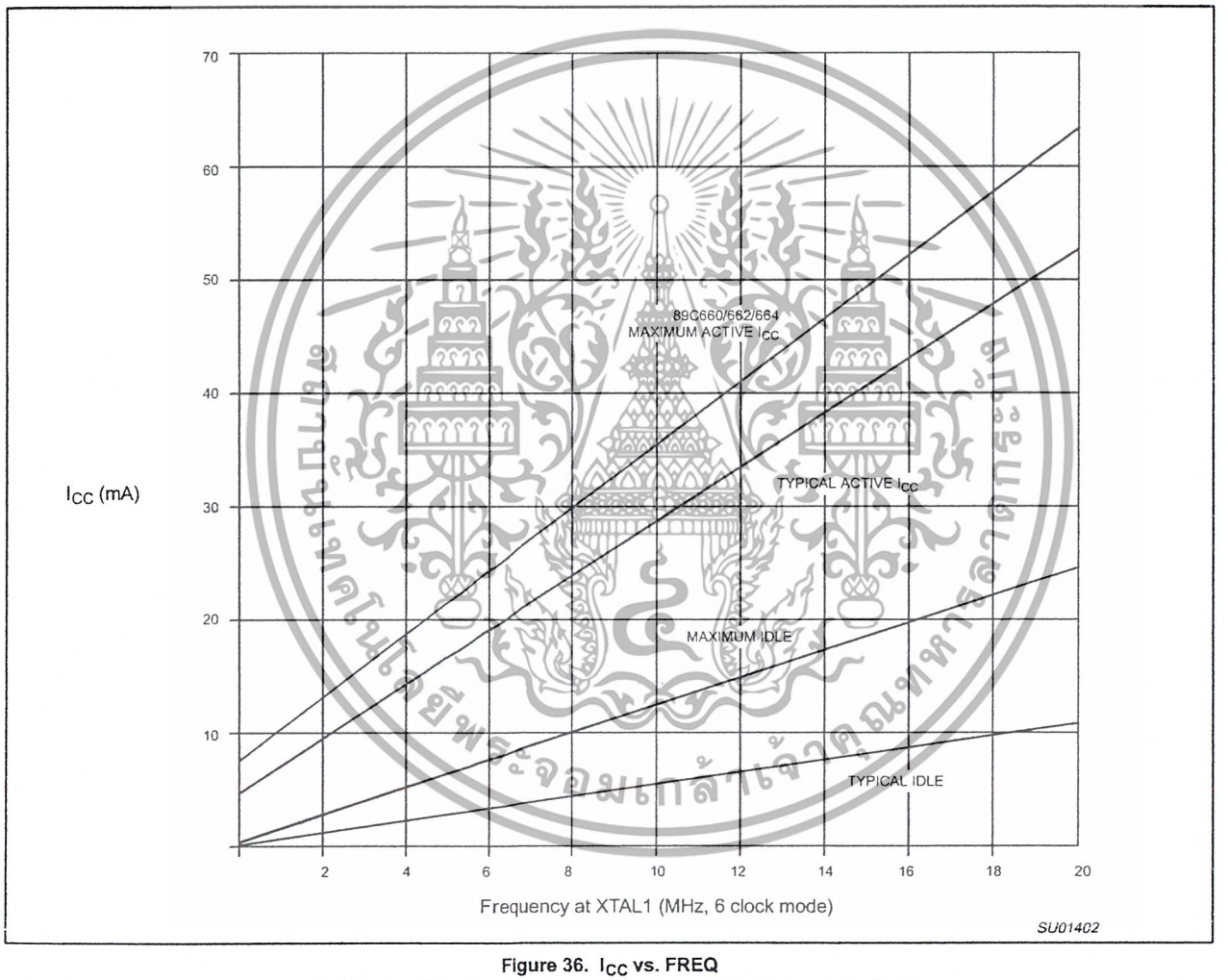
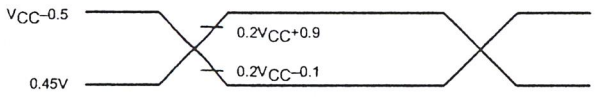


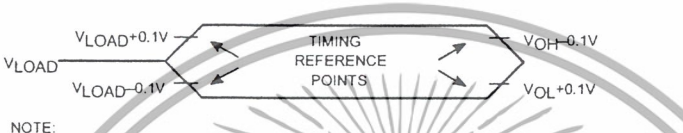
Figure 36. I_{CC} vs. FREQ
Valid only within frequency specifications of the device under test



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

SU00010

Figure 37. AC Testing Input/Output



NOTE:
For timing purposes, a port is no longer floating when a 100mV change from load voltage occurs, and begins to float when a 100mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OH}/I_{OL} \geq \pm 20$mA.

SU00011

Figure 38. Float Waveform

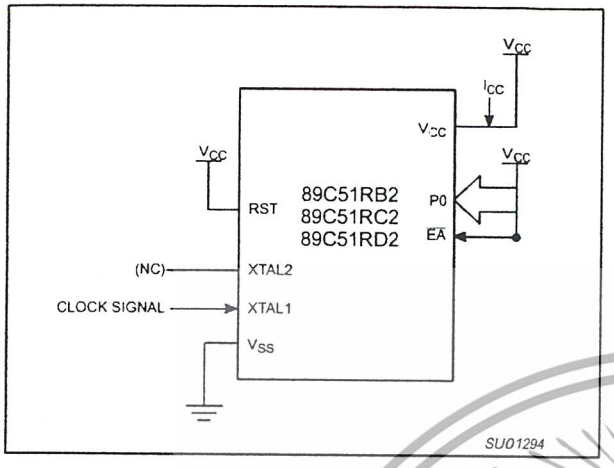


Figure 39. I_{CC} Test Condition, Active Mode.
All other pins are disconnected

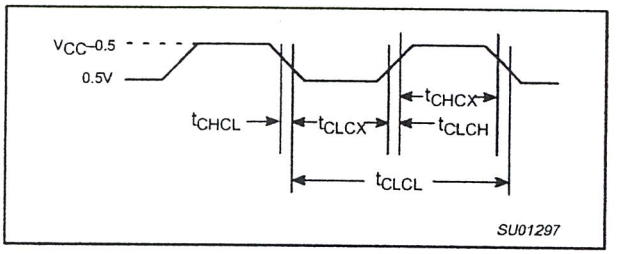


Figure 41. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes.
$t_{CLCL} = t_{CHCL} = 10$ ns

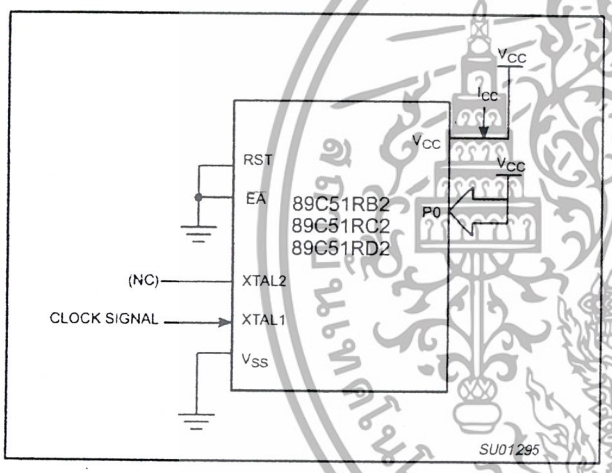


Figure 40. I_{CC} Test Condition, Idle Mode.
All other pins are disconnected

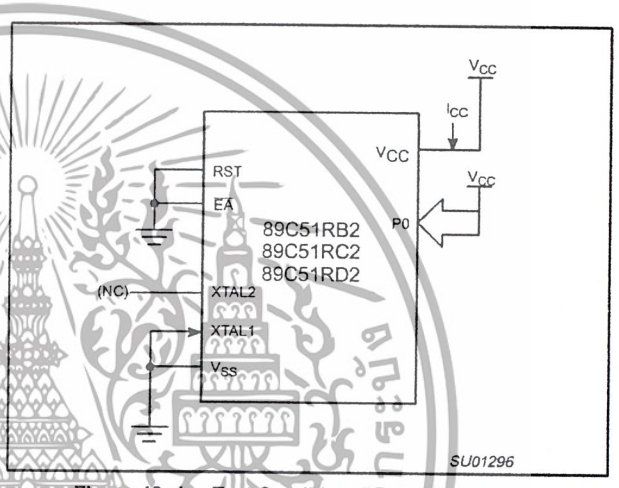


Figure 42. I_{CC} Test Condition, Power Down Mode.
All other pins are disconnected; V_{CC} = 2V to 5.5V

FLASH EPROM MEMORY

GENERAL DESCRIPTION

The P89C51RB2/RC2/RD2 Flash memory augments EPROM functionality with in-circuit electrical erasure and programming. The Flash can be read and written as bytes. The Chip Erase operation will erase the entire program memory. The Block Erase function can erase any Flash block. In-system programming and standard parallel programming are both available. On-chip erase and write timing generation contribute to a user friendly programming interface.

The P89C51RB2/RC2/RD2 Flash reliably stores memory contents even after 10,000 erase and program cycles. The cell is designed to optimize the erase and programming mechanisms. In addition, the combination of advanced tunnel oxide processing and low internal electric fields for erase and programming operations produces reliable cycling. The P89C51RB2/RC2/RD2 uses a +5 V V_{PP} supply to perform the Program/Erase algorithms.

FEATURES

- Flash EPROM internal program memory with Block Erase.
- Internal 1 kB fixed boot ROM, containing low-level in-system programming routines and a default serial loader. User program can call these routines to perform In-Application Programming (IAP). The Boot ROM can be turned off to provide access to the full 64 kB Flash memory.
- Boot vector allows user provided Flash loader code to reside anywhere in the Flash memory space. This configuration provides flexibility to the user.
- Default loader in Boot ROM allows programming via the serial port without the need for a user provided loader.
- Up to 64 kB external program memory if the internal program memory is disabled ($\overline{EA}$ = 0).
- Programming and erase voltage +5 V (+12 V tolerant).
- Read/Programming/Erase:
  - Byte-wise read (100 ns access time).
  - Byte Programming (20 µs).
  - Typical erase times:
    Block Erase (8 kB or 16 kB) in 3 seconds.
    Full Erase (64 kB) in 3 seconds.
- Parallel programming with 87C51 compatible hardware interface to programmer.
- In-system programming.
- Programmable security for the code in the Flash.
- 10,000 minimum erase/program cycles for each byte.
- 10-year minimum data retention.

CAPABILITIES OF THE PHILIPS 89C51 FLASH-BASED MICROCONTROLLERS

Flash organization

The P89C51RB2/RC2/RD2 contains 16KB/32KB/64K bytes of Flash program memory. This memory is organized as 5 separate blocks. The first two blocks are 8 kB in size, filling the program memory space from address 0 through 3FFF hex. The final three blocks are 16 kB in size and occupy addresses from 4000 through FFFF hex.

Figure 43 depicts the Flash memory configurations.

Flash Programming and Erasure

There are three methods of erasing or programming of the Flash memory that may be used. First, the Flash may be programmed or erased in the end-user application by calling low-level routines through a common entry point in the Boot ROM. The end-user application, though, must be executing code from a different block than the block that is being erased or programmed. Second, the on-chip ISP boot loader may be invoked. This ISP boot loader will, in turn, call low-level routines through the same common entry point in the Boot ROM that can be used by the end-user application. Third, the Flash may be programmed or erased using the parallel method by using a commercially available EPROM programmer. The parallel programming method used by these devices is similar to that used by EPROM 87C51, but it is not identical, and the commercially available programmer will need to have support for these devices.

Boot ROM

When the microcontroller programs its own Flash memory, all of the low level details are handled by code that is permanently contained in a 1 kB Boot ROM that is separate from the Flash memory. A user program simply calls the common entry point with appropriate parameters in the Boot ROM to accomplish the desired operation. Boot ROM operations include things like: erase block, program byte, verify byte, program security lock bit, etc. The Boot ROM overlays the program memory space at the top of the address space from FC00 to FFFF hex, when it is enabled. The Boot ROM may be turned off so that the upper 1 kB of Flash program memory are accessible for execution.

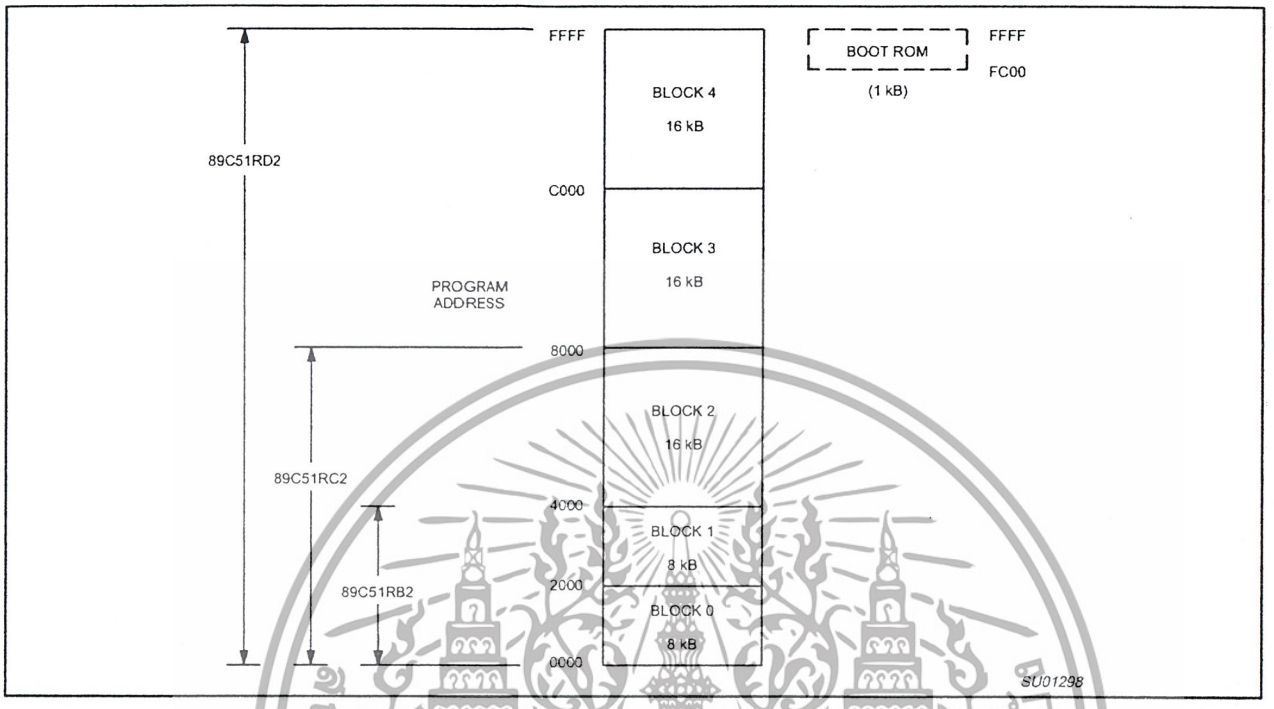


Figure 43. Flash Memory Configurations

Power-On Reset Code Execution

The P89C51RB2/RC2/RD2 contains two special Flash registers; the BOOT VECTOR and the STATUS BYTE. At the falling edge of reset, the P89C51RB2/RC2/RD2 examines the contents of the Status Byte. If the Status Byte is set to zero, power-up execution starts at location 0000H, which is the normal start address of the user's application code. When the Status Byte is set to a value other than zero, the contents of the Boot Vector is used as the high byte of the execution address and the low byte is set to 00H. The factory default setting is 0FCH, corresponds to the address 0FC00H for the factory masked-ROM ISP boot loader. A custom boot loader can be written with the Boot Vector set to the custom boot loader.

NOTE: When erasing the Status Byte or Boot Vector, both bytes are erased at the same time. It is necessary to reprogram the Boot Vector after erasing and updating the Status Byte.

Hardware Activation of the Boot Loader

The boot loader can also be executed by holding PSEN LOW, P2.7 high, $\overline{EA}$ greater than V_{IH} (such as +5 V), and ALE HIGH (or not connected) at the falling edge of RESET. This is the same effect as having a non-zero status byte. This allows an application to be built that will normally execute the end user's code but can be manually forced into ISP operation.

If the factory default setting for the Boot Vector (0FCH) is changed, it will no longer point to the ISP masked-ROM boot loader code. If this happens, the only way it is possible to change the contents of the Boot Vector is through the parallel programming method, provided that the end user application does not contain a customized loader that provides for erasing and reprogramming of the Boot Vector and Status Byte.

After programming the Flash, the status byte should be programmed to zero in order to allow execution of the user's application code beginning at address 0000H.

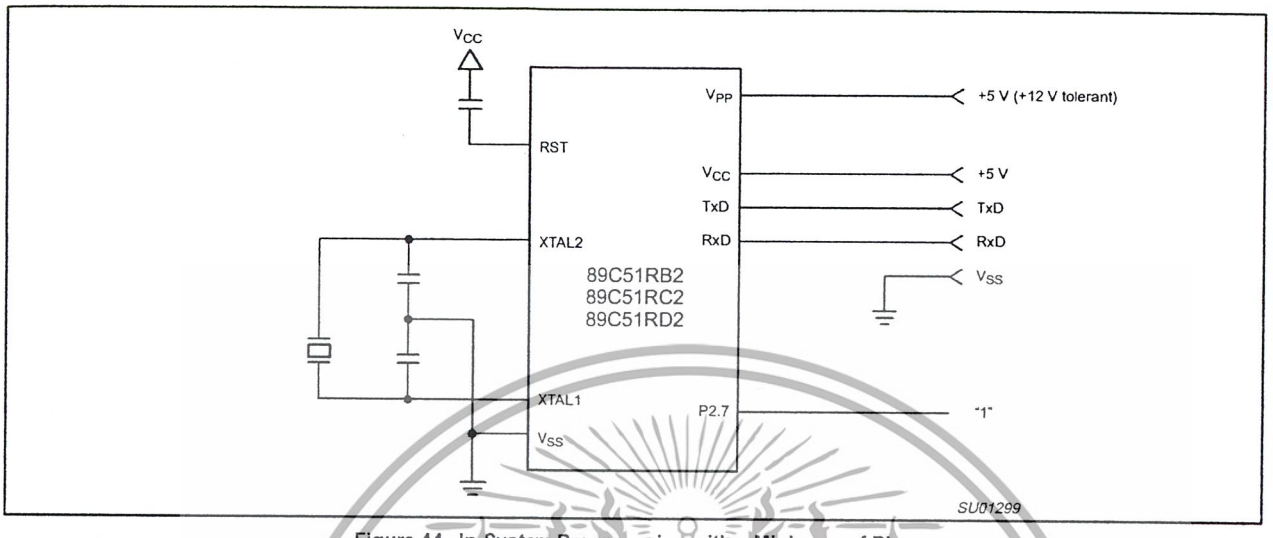


Figure 44. In-System Programming with a Minimum of Pins

In-System Programming (ISP)

The In-System Programming (ISP) is performed without removing the microcontroller from the system. The In-System Programming (ISP) facility consists of a series of internal hardware resources coupled with internal firmware to facilitate remote programming of the P89C51RB2/RC2/RD2 through the serial port. This firmware is provided by Philips and embedded within each P89C51RB2/RC2/RD2 device.

The Philips In-System Programming (ISP) facility has made in-circuit programming in an embedded application possible with a minimum of additional expense in components and circuit board area.

The ISP function uses five pins: TxD, RxD, V_{SS}, V_{CC}, and V_{PP} (see Figure 44). Only a small connector needs to be available to interface your application to an external circuit in order to use this feature. The V_{PP} supply should be adequately decoupled and V_{PP} not allowed to exceed datasheet limits.

Using the In-System Programming (ISP)

The ISP feature allows for a wide range of baud rates to be used in your application, independent of the oscillator frequency. It is also adaptable to a wide range of oscillator frequencies. This is accomplished by measuring the bit-time of a single bit in a received character. This information is then used to program the baud rate in terms of timer counts based on the oscillator frequency. The ISP feature requires that an initial character (an uppercase U) be sent to the P89C51RB2/RC2/RD2 to establish the baud rate. The ISP firmware provides auto-echo of received characters.

Once baud rate initialization has been performed, the ISP firmware will only accept Intel Hex-type records. Intel Hex records consist of ASCII characters used to represent hexadecimal values and are summarized below:

:NNAAAARRDD..DDCC<crlf>

In the Intel Hex record, the "NN" represents the number of data bytes in the record. The P89C51RB2/RC2/RD2 will accept up to 16 (10H) data bytes. The "AAAA" string represents the address of the first byte in the record. If there are zero bytes in the record, this field is often set to 0000. The "RR" string indicates the record type. A record type of "00" is a data record. A record type of "01" indicates the end-of-file mark. In this application, additional record types will be added to indicate either commands or data for the ISP facility. The maximum number of data bytes in a record is limited to 16 (decimal). ISP commands are summarized in Table 8.

As a record is received by the P89C51RB2/RC2/RD2, the information in the record is stored internally and a checksum calculation is performed. The operation indicated by the record type is not performed until the entire record has been received. Should an error occur in the checksum, the P89C51RB2/RC2/RD2 will send an "X" out the serial port indicating a checksum error. If the checksum calculation is found to match the checksum in the record, then the command will be executed. In most cases, successful reception of the record will be indicated by transmitting a "." character out the serial port (displaying the contents of the internal program memory is an exception).

In the case of a Data Record (record type 00), an additional check is made. A "." character will NOT be sent unless the record checksum matched the calculated checksum and all of the bytes in the record were successfully programmed. For a data record, an "X" indicates that the checksum failed to match, and an "R" character indicates that one of the bytes did not properly program. It is necessary to send a type 02 record (specify oscillator frequency) to the P89C51RB2/RC2/RD2 before programming data.

The ISP facility was designed to that specific crystal frequencies were not required in order to generate baud rates or time the programming pulses. The user thus needs to provide the P89C51RB2/RC2/RD2 with information required to generate the proper timing. Record type 02 is provided for this purpose.

WinISP, a software utility to implement ISP programming with a PC, is available on Philips Semiconductors' website. In addition, at the website is a listing of third party commercially available serial and parallel programmers.

Table 8. Intel-Hex Records Used by In-System Programming

RECORD TYPE	COMMAND/DATA FUNCTION
00	Program Data :nnaaaa00dd....ddcc Where: Nn = number of bytes (hex) in record Aaaa = memory address of first byte in record dd....dd = data bytes cc = checksum Example: :10008000AF5F67F0602703E0322CFA92007780C3FD
01	End of File (EOF), no operation :xxxxxx01cc Where: xxxxxx = required field, but value is a "don't care" cc = checksum Example: :00000001FF
02	Specify Oscillator Frequency :01xxxx02ddcc Where: xxxx = required field, but value is a "don't care" dd = integer oscillator frequency rounded down to nearest MHz cc = checksum Example: :0100000210ED (dd = 10h = 16, used for 16.0-16.9 MHz)

RECORD TYPE	COMMAND/DATA FUNCTION
03	Miscellaneous Write Functions :nnxxxx03ffssddcc Where: nn = number of bytes (hex) in record xxxx = required field, but value is a "don't care" 03 = Write Function ff = subfunction code ss = selection code dd = data input (as needed) cc = checksum Subfunction Code = 01 (Erase Blocks) ff = 01 ss = block code as shown below: block 0, 0k to 8k, 00H block 1, 8k to 16k, 20H block 2, 16k to 32k, 40H block 3, 32k to 48k, 80H block 4, 48k to 64k, C0H Example: :0200000301C03A erase block 4 Subfunction Code = 04 (Erase Boot Vector and Status Byte) ff = 04 ss = don't care Example: :020000030400F7 erase boot vector and status byte Subfunction Code = 05 (Program Security Bits) ff = 05 ss = 00 program security bit 1 (inhibit writing to Flash) 01 program security bit 2 (inhibit Flash verify) 02 program security bit 3 (disable external memory) Example: :020000030501F5 program security bit 2 Subfunction Code = 06 (Program Status Byte or Boot Vector) ff = 06 ss = 00 program status byte 01 program boot vector Example: :030000030601FCF7 program boot vector with 0FCH Subfunction Code = 07 (Full Chip Erase) Erases all blocks, security bits, and sets status and boot vector to default values ff = 07 ss = don't care dd = don't care Example: :0100000307F5 full chip erase
04	Display Device Data or Blank Check – Record type 04 causes the contents of the entire Flash array to be sent out the serial port in a formatted display. This display consists of an address and the contents of 16 bytes starting with that address. No display of the device contents will occur if security bit 2 has been programmed. Data to the serial port is initiated by the reception of any character and terminated by the reception of any character. General Format of Function 04 :05xxxx04sssseeeeffcc Where: 05 = number of bytes (hex) in record xxxx = required field, but value is a "don't care" 04 = "Display Device Data or Blank Check" function code ssss = starting address eeee = ending address ff = subfunction 00 = display data 01 = blank check cc = checksum Example: :0500000440004FFF0069 display 4000-4FFF

RECORD TYPE	COMMAND/DATA FUNCTION
05	Miscellaneous Read Functions General Format of Function 05 :02xxxx05ffsscc Where: 02 = number of bytes (hex) in record xxxx = required field, but value is a "don't care" 05 = "Miscellaneous Read" function code ffss = subfunction and selection code 0000 = read signature byte - manufacturer id (15H) 0001 = read signature byte - device id # 1 (C2H) 0002 = read signature byte - device id # 2 0700 = read security bits 0701 = read status byte 0702 = read boot vector cc = checksum Example: :0200000500001F8 read signature byte - device id # 1
06	Direct Load of Baud Rate General Format of Function 06 :02xxxx06hhllcc Where: 02 = number of bytes (hex) in record xxxx = required field, but value is a "don't care" 06 = "Direct Load of Baud Rate" function code hh = high byte of Timer 2 ll = low byte of Timer 2 cc = checksum Example: :02000006F500F3

In Application Programming Method

Several In Application Programming (IAP) calls are available for use by an application program to permit selective erasing and programming of Flash sectors. All calls are made through a common interface, PGM_MTP. The programming functions are selected by setting up the microcontroller's registers before making a call to PGM_MTP at FFF0H. The oscillator frequency is an integer number rounded down to the nearest megahertz. For example, set R0 to 11 for 11.0592 MHz. Results are returned in the registers. The IAP calls are shown in Table 9.

Using the Watchdog Timer (WDT)

The 89C51Rx2 devices support the use of the WDT in IAP. The user specifies that the WDT is to be fed by setting the most significant bit of the function parameter passed in R1 prior to calling PGM_MTP. The WDT function is only supported for Block Erase when using Quick Block Erase. The Quick Block Erase is specified by performing a Block Erase with register R0 = 0. Requesting a WDT feed during IAP should only be performed in applications that use the WDT since the process of feeding the WDT will start the WDT if the WDT was not running.

Table 9. IAP calls

IAP CALL	PARAMETER
PROGRAM DATA BYTE	Input Parameters: R0 = osc freq (integer) R1 = 02h R1 = 82h (WDT feed) DPTR = address of byte to program ACC = byte to program Return Parameter ACC = 00 if pass, !00 if fail
ERASE BLOCK	Input Parameters: R0 = osc freq (integer) R0 = 0 (Quick Erase) R1 = 01h R1 = 81h (WDT feed) DPH = block code as shown below: block 0, 0k to 8k, 00H block 1, 8k to 16k, 20H block 2, 16k to 32k, 40H block 3, 32k to 48k, 80H block 4, 48k to 64k, C0H DPL = 00h Return Parameter none
ERASE BOOT VECTOR	Input Parameters: R0 = osc freq (integer) R1 = 04h R1 = 84h (WDT feed) DPH = 00h DPL = don't care Return Parameter none
PROGRAM SECURITY BIT	Input Parameters: R0 = osc freq (integer) R1 = 05h R1 = 85h (WDT feed) DPH = 00h DPL = 00h - security bit # 1 (inhibit writing to Flash) 01h - security bit # 2 (inhibit Flash verify) 02h - security bit # 3 (disable external memory) Return Parameter none
PROGRAM STATUS BYTE	Input Parameters: R0 = osc freq (integer) R1 = 06h R1 = 86h (WDT feed) DPH = 00h DPL = 00h - program status byte ACC = status byte Return Parameter ACC = status byte

IAP CALL	PARAMETER
PROGRAM BOOT VECTOR	Input Parameters: R0 = osc freq (integer) R1 = 06h R1 = 86h (WDT feed) DPH = 00h DPL = 01h - program boot vector ACC = boot vector Return Parameter ACC = boot vector
READ DEVICE DATA	Input Parameters: R1 = 03h R1 = 83h (WDT feed) DPTR = address of byte to read Return Parameter ACC = value of byte read
READ MANUFACTURER ID	Input Parameters: R0 = osc freq (integer) R1 = 00h R1 = 80h (WDT feed) DPH = 00h DPL = 00h (manufacturer ID) Return Parameter ACC = value of byte read
READ DEVICE ID # 1	Input Parameters: R0 = osc freq (integer) R1 = 00h R1 = 80h (WDT feed) DPH = 00h DPL = 01h (device ID # 1) Return Parameter ACC = value of byte read
READ DEVICE ID # 2	Input Parameters: R0 = osc freq (integer) R1 = 00h R1 = 80h (WDT feed) DPH = 00h DPL = 02h (device ID # 2) Return Parameter ACC = value of byte read
READ SECURITY BITS	Input Parameters: R0 = osc freq (integer) R1 = 07h R1 = 87h (WDT feed) DPH = 00h DPL = 00h (security bits) Return Parameter ACC = value of byte read
READ STATUS BYTE	Input Parameters: R0 = osc freq (integer) R1 = 07h R1 = 87h (WDT feed) DPH = 00h DPL = 01h (status byte) Return Parameter ACC = value of byte read
READ BOOT VECTOR	Input Parameters: R0 = osc freq (integer) R1 = 07h R1 = 87h (WDT feed) DPH = 00h DPL = 02h (boot vector) Return Parameter ACC = value of byte read

Security

The security feature protects against software piracy and prevents the contents of the Flash from being read. The Security Lock bits are located in Flash. The P89C51RB2/RC2/RD2 has three programmable security lock bits that will provide different levels of protection for the on-chip code and data (see Table 10).

Table 10.

	SECURITY LOCK BITS[1]			PROTECTION DESCRIPTION
LEVEL	LB1	LB2	LB3	
1	0	0	0	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory.
2	1	0	0	Block erase is disabled. Erase or programming of the status byte or boot vector is disabled.
3	1	1	0	Verify of code memory is disabled.
4	1	1	1	External execution is disabled.

NOTE:
1. Security bits are independent of each other. Full-chip erase may be performed regardless of the state of the security bits.
2. Any other combination of lock bits is undefined.
3. Setting LBx doesn't prevent programming of unprogrammed bits.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำรวจลุล่วงตามวัตถุประสงค์มาได้ด้วยดีนั้น ผู้จัดทำต้องขอขอบคุณ อาจารย์เกียรติวรรณ ทรงสัตย์ อาจารย์ที่ปรึกษาที่น่ารักของเราที่ให้คำปรึกษาและให้ความช่วยเหลือที่ดีตลอดมา

ขอขอบคุณ อาจารย์ทุกท่านภายในสถาบันนี้ ที่ให้ความรู้และความเข้าใจในวิชาที่เรียน และสามารถนำไปใช้ได้

ขอขอบคุณ คุณพ่อคุณแม่ ญาติๆ ทุกคนที่คอยให้กำลังใจมาโดยตลอดตั้งแต่ยังเด็ก

ขอขอบคุณ พี่ชาย ที่มาให้ช่วยเหลือแม้ว่าโครงงานตัวเองจะยังไม่เสร็จก็ตาม

ขอขอบคุณ เพื่อนๆทุกคนกับความสนุกสนานร่าเริง และให้สิ่งดีๆตลอดช่วงเวลาที่อยู่ด้วยกัน

ขอขอบคุณ น้องๆและเพื่อนๆ ชมนุมประชาสัมพันธ์ ที่ทำคอยให้กำลังใจ

ขอขอบคุณ ชุมนุมวิชาการ ที่ทำให้เรามี play station ได้เล่นแก้เครียดหลังการทำโครงงาน

ขอขอบคุณ ร้านพี่เพ็ญ และ ร้านพี่จี๋ ที่ทำให้เราอิ่มมื้อเย็นตลอดช่วงเวลาทำโครงงาน

ขอขอบคุณร้าน 7-ELEVEN ที่ทำให้เราอิ่มมื้อดึกตลอดการทำงาน

ขอขอบคุณ บริษัท ศิลารีเสิร์ช จำกัด ที่ให้คำปรึกษาเมื่อเกิดปัญหาในการทำงาน

ขอขอบคุณ บริษัท ฟิลิปส์ อิเล็คทรอนิกส์ (ประเทศไทย) จำกัด ที่เอื้อเฟื้อ ไอซี P89C51RD2 แก่โครงงานนี้

สุดท้ายต้องขอขอบคุณคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่หล่อหลอมผู้ชายธรรมดา 2 คน ให้เป็นวิศวกรที่ดีของสังคม

เอกสารอ้างอิง

1. ธาริน สิทธิธรรมชารี , "คู่มือการเขียนโปรแกรม Microsoft Visual Basic Version 6.0" , บริษัท ซัคเซส มีเดีย จำกัด , 392 หน้า
2. สมศักดิ์ ศรีขจรเกียรติ , "Visual Basic 6. Teach Yourself" , บิบลิโอไฟล์ , 472 หน้า , 2542
3. กิตติ ภักดีวัฒนะกุล , "Visual Basic 6 ฉบับโปรแกรมเมอร์" , 621 หน้า , 2543
4. กฤษดา ใจเย็น , "เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ทอนุกรม" , บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด , 163 หน้า
5. ธีรวัฒน์ ประกอบผล , "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์" ,สำนักพิมพ์ ส.ส.ท., 234 หน้า , 2543
6. รศ.สมยศ จุณณะปิยะ , "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51" , ภาควิชาวิศวกรรมโทรคมนาคม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 358 หน้า , 2543