โปรแกรมสำหรับออกแบบ BOOLEAN ลงบน MCS-51

SOFTWARE FOR DESIGNING BOOLEAN BASED ON MCS-51

นายสุพีเรศ สุนิธิ

นายโสภณ แก้วรอด

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

SOFTWARE FOR DESIGNING BOOLEAN BASED ON MCS-51

SUPEERES SUNITI

SOPHON KEAWROD

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTUTITE OF TECHNOLOGY LADKRABANG

2003

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	โปรแกรมสำหรับการออกแบบ BOOLEAN ลงบน MCS-51		
	SOFTWARE FOR DISIGNING BOOLEAN BASED ON MCS-51		
นักศึกษาผู้จัดทำ	นายสุพีเรศ	สุนิธิ	ระหัสประจำตัว 44015450
	นายโสภณ	แก้วรอด	ระหัสประจำตัว 44015453
ปริญญา	วิศวกรรมศาสตรบัณฑิต		
สาขาวิชา	วิศวกรรมการวัดคุม		
ปีการศึกษา	2546		

อาจารย์ผู้ควบคุมปริญญานิพนธ์	**ลายมือชื่อ**
ผศ. ทวีพล ซื่อสัตย์	

วัน/เดือน/ปี ที่สอบ วันอังคารที่ 20 เมษายน พ.ศ. 2547

สถานที่สอบ ณ ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(รศ. ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์ โปรกแรมสำหรับออกแบบ BOOLEAN ลงบน MCS-51
SOFTWARE FOR DESIGNING BOOLEAN BASED ON MCS-51

นักศึกษาผู้จัดทำ นายโสภณ แก้วรอด
นายสุพีเรศ สุนิธิ

อาจารย์ที่ปรึกษา ผศ. ทวีพล ซื่อสัตย์

ปีการศึกษา 2546

บทคัดย่อ

ตัวไมโครคอนโทรลเลอร์ ปัจจุบันได้เข้ามามีบทบาทในชีวิตประจำวันของเรามากขึ้น โดยจะเป็นส่วนประกอบของอุปกรณ์ไฟฟ้าระบบควบคุมอัตโนมัติต่างๆ และยังนำมาใช้งานแพร่หลาย อย่างไรก็ตามด้วยสถาปัตยกรรมและชุดคำสั่งที่มีอยู่มากมาย อาจทำให้ผู้ใช้งานมือใหม่และผู้ที่สนใจต้องการนำไมโครคอนโทรลเลอร์ตระกลู MCS-51 นี้ไปใช้งานเกิดข้อติดขัดและไม่แน่ใจว่าตัวเองสามารถใช้งานได้หรือไม่ ปริญญานิพนธ์ฉบับนี้จึงได้มีการพัฒนาโปรแกรมการใช้งานสำหรับผู้เริ่มต้นและผูที่ต้องการเรียนรู้ โดยการใช้คำสั่งทางลอจิกที่อยู่ในรูปของ BOOLEAN แบบที่ใช้กับเครื่องควบคุมที่โปรแกรมได้ เพราะเป็นคำสั่งที่สามารถเข้าใจได้ง่าย สำหรับผู้ที่มีความรู้พื้นฐานทางไฟฟ้าอยู่บ้าง โปรแกรมนี้จะช่วยประหยัดเวลาการเขียนคำสั่งและศึกษาคำสั่งต่างๆ ของตัวควบคุมไมโครคอนโทรลเลอร์ตระกลู MCS-51 ลงได้มาก จากผลการทดลองเมื่อป้อนคำสั่งทางลอจิกที่อยู่ในรูป BOOLEAN ให้กับโปรแกรม ผลที่ได้ก็คือ เอาท์พุตที่ได้จะอยู่ในรูปของ HEX FILE ซึ่งจะนำไปอัดโปรแกรมลงตัว MCS-51 ขั้นตอนการอัดโปรแกรมเหมือนกับที่เขียนด้วยภาษา ASSEMBLY จะช่วยให้การทำงานสะดวกขึ้น

Thesis Title	Software For Designing Boolean Based On MCS-51
Authors	Mr. Supeerat Suniti
	Mr. Sophon Keawrod
Thesis Advisor	Asst.Prof. Tweepol suasat
Year	2003

ABSTRACT

Recently , The micro controller MCS-51 is widely used in automatic control system for industrial evironment and other applications. It has a varity instructions for applying the complicate works. However the assembly programming is diffical for beginning users , therefore this complicate project designs the software to convert from boolean instruction assembly. Typical , The boolean instruction is one of the programming language for programmable Logic Controller , It's easy to program basic logic instruction including timer and counter and special functions. This software can be able to burn program to amony number of flashchip suchas 89C1051

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเพราะได้รับความเมตตาจาก ผู้ช่วยศาสตราจารย์ ทวีพล ซื่อสัตย์ ที่ได้ให้คำแนะนำแก่ผู้วิจัยตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์และเครื่องมือต่างๆ ในการทำปริญญานิพนธ์นี้ ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ อาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่าน ที่ให้คำแนะนำอันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

และที่ลืมเสียไม่ได้คือ ขอกราบขอบพระคุณคุณแม่ อันเป็นที่รักยิ่ง ที่สนับสนุนและเป็นแรงบันดาลใจในการทำปริญญานิพนธ์ฉบับนี้

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

เนื่องจากปัจจุบันการศึกษาเรื่องไมโครคอนโทรลเลอร์ MCS-51 เป็นเรื่องที่ใกล้ตัวมากขึ้น จากหลายปีที่ผ่านมาคนที่จะศึกษาไมโครคอนโทรลเลอร์ได้นั้น จะต้องศึกษาในระดับปริญญาตรีเท่านั้น แต่ที่สำคัญจะต้องมีทุนพอที่จะศึกษา ทดลอง และสร้างโปรเจคงานขึ้นมาสักชิ้น ในสมัยก่อนการที่จะสร้างการควบคุมขึ้นมาสักชิ้นต้องพึ่งพาเครื่องมือหลายอย่าง อีกทั้งมีขั้นตอนมากมายในการพัฒนา ดังนั้นทางผู้จัดทำจึงมีความคิดว่าถ้าสามารถนำโปรแกรมที่เขียนด้วยภาษาบูลลีนก่อนลงบนเครื่องคอมพิวเตอร์ก่อนแล้วจึงทำการถ่ายโอนโปรแกรมลงเครื่องไมโครคอนโทรลเลอร์ MCS-51 อีกทีหนึ่งก็จะเป็นการสะดวกแก่ผู้ใช้เพราะถ้าทำงานบนคอมพิวเตอร์จะสามารถดูโปรแกรมและคำสั่งต่างๆได้โดยง่าย

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์นี้จะเป็น การศึกษาและพัฒนาการออกแบบโปรแกรมด้วยบูลลีน เพื่อนำไปใช้กับไมโครคอนโทรลเลอร์ MCS-51 ซึ่งช่วยให้ผู้เริ่มต้นใช้งานไมโครคอนโทรลเลอร์ MCS-51 สามารถเขียนโปรแกรมควบคุมได้สะดวกยิ่งขึ้น และสามารถเข้าใจวิธีการทำงานของโปรแกรมโดยอาศัยภาษาบูลลีนนอกจากนี้ยังช่วยประหยัดเวลา การเขียนโปรแกรมคำสั่งบนเครื่องของตัวควบคุมไมโครคอนโทรลเลอร์ MCS-51

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์นี้จะเป็น การศึกษาและพัฒนาขั้นตอนการออกแบบโปรแกรม สำหรับเพื่อที่ต้องการนำไปใช้กับไมโครคอนโทรลเลอร์ ซึ่งขั้นตอนการทำงานที่สำคัญของตัวโปรแกรมนี้จะเป็นขั้นการแปลงคำสั่งที่เขียนด้วยภาษาบูลลีนพื้นฐานให้เป็นภาษาแอสเซมบลีแล้วแปลงแอสเซมบลีเป็น Hex.File นำไปโหลดลงไมโครคอนโทรลเลอร์

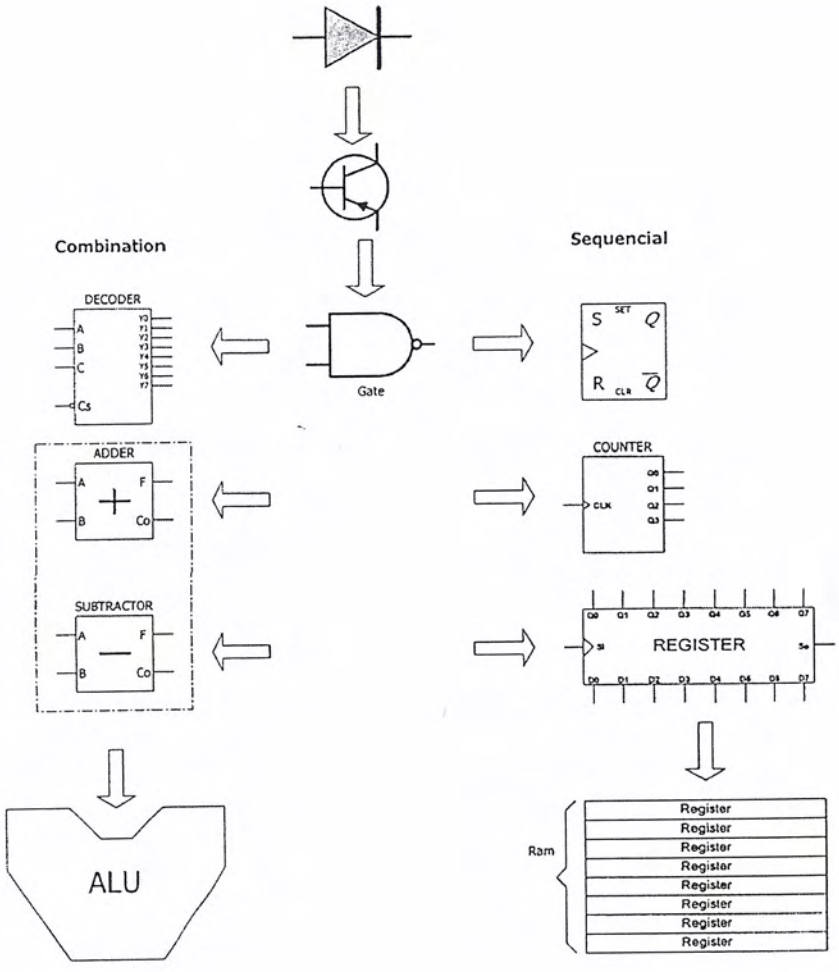
1.4 ขั้นตอนการศึกษา

การทำโครงงานวิจัยในปริญญานิพนธ์ฉบับนี้จะมีขั้นตอนการศึกษาเริ่มต้นจากการศึกษาในส่วนของตัวไมโครคอนโทรเลอร์ MCS-51 รายละเอียดโครงสร้างการทำงานของพอร์ตต่างๆภายในตัวไมโครคอนโทรลเลอร์รวมทั้งคำสั่งพื้นฐานใน PLC และการเขียนโปรแกรมด้วย Visual Basic

บทที่ 2
ทฤษฎีและหลักการ

2.1 ที่มาและวิวัฒนาการของไมโครคอนโทรลเลอร์

เริ่มแรกจะใช้หลอดสุญญากาศ หลังจากการเปลี่ยนแปลงครั้งยิ่งใหญ่ นั้นคือมนุษย์สามารถสร้างสารกึ่งตัวนำ สิ่งนี้ทำให้โลกพัฒนาไปไกล เริ่มจากสารกึ่งตัวนำตัวแรก นั้นคือที่เรียกว่า ไดโอด



ภาพที่ 2.1 แสดงวิวัฒนาการของไมโครคอนโทรลเลอร์

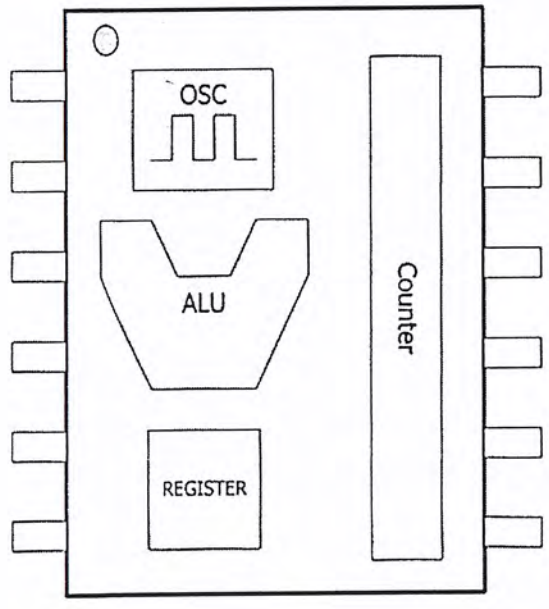
จากไดโอดเราสามารถสร้างเป็นทรานซิสเตอร์ และเมื่อเราเอาตัวทรานซิสเตอร์กับตัวไดโอด มารวมกันก็จะได้เป็นเกต(Gate ถูกพัฒนาออกเป็น 2 สายในวงจรดิจิตอล นั้นคือ

1. วงจร Combination

คือ วงจรที่ประกอบไปด้วยเกตต่างๆ ต่อเข้าด้วยกัน โดยไม่จำเป็นต้องมีสัญญาณพัลส์มากระตุ้น วงจรประเภทนี้ได้แก่ วงจร Decoder Adder Subtractor Muntiplex Demultiplex วงจรเหล่านี้ถ้านำมารวมกันแล้ว ก็จะได้กลายเป็นตัวที่เราเรียกว่า ALU (Arithmatic Logic Unit) ที่อยู่ในไมโครคอลโทรลเลอร์ทั่วไป ทำหน้าที่คำนวณทางด้านคณิตศาสตร์

2. วงจร Sequencial

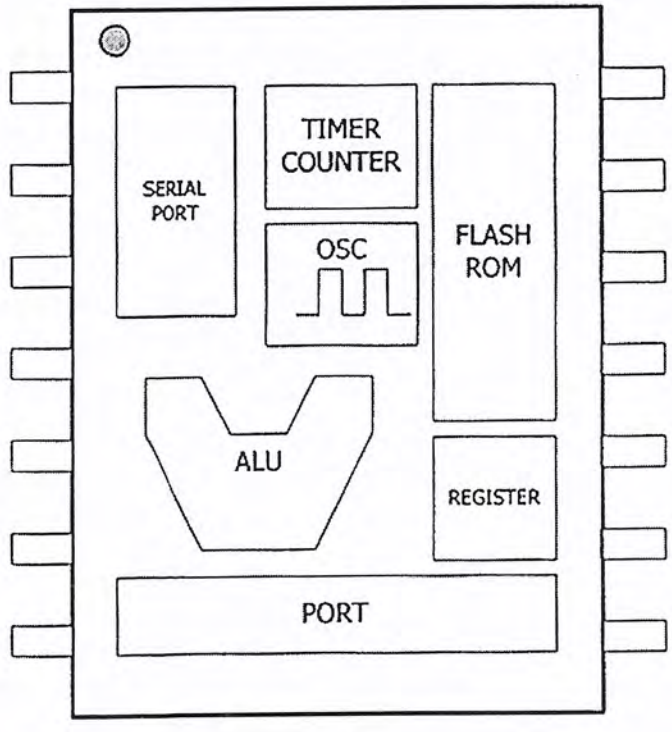
คือวงจรที่ประกอบจากพวกอุปกรณ์ Flip – Flop ต่อเข้าด้วยกันซึ่งต้องมีสัญญาณมากระตุ้นให้ทำงาน(Synchronous) หรือสัญญาณนาฬิกา เช่น วงจรCounter, Register, Latch วงจร Register เหล่านี้เมื่อนำมาต่อพ่วงกันหลายตัวก็จะกลายเป็น หน่วยความจำของไมโครคอนโทรลเลอร์ เพราะฉะนั้น Register นั้นจึงเป็นส่วนหนึ่งของหน่วยความจำ (RAM) นั้นเองถ้าหากเราเอาวงจรทั้งหมดนี้มาจัดเรียงเชื่อมกันและกันให้มีความสามารถในการคำนวณทางคณิตศาสตร์ทางลอจิกและการโอนย้ายข้อมูล เราก็จะได้อุปกรณ์ตัวใหม่ที่เรียกว่า



ภาพที่ 2.2 หน่วยประมวลผลกลาง CPU

2.1.1 หน่วยประมวลผลกลาง CPU (Central Processing Unit)

CPU เกิดจากวงจรต่างๆ มารวมกันให้สามารถทำงานได้หลายอย่าง สรุปแล้ว CPU นั้นจะประกอบด้วยวงจรต่างๆดังนี้ DECODER REGISTER ADDER SUBTRACTOR BUFFER OSCILLATOR นอกจากนี้แล้ววงจรลอจิกเกต ยังสามารถดัดแปลงเป็นวงจรต่างๆได้อีกมากมายตามวัตถุประสงค์ของ ผู้ออกแบบไม่ว่าจะทำเป็นวงจรพอร์ตขนาน(ParallelPort)วงจรพอร์ตอนุกรม (Serial Port) วงจร Timer Counter วงจร A/D (Analog to Digital) EPROM และอื่นๆ อีกมากมาย วงจรเหล่านี้ก็จะเป็นวงจรที่เชื่อมต่อกับ CPUให้ CPU สามารถเลือกใช้ได้ตามต้องการ เพราะฉะนั้นการที่เราจะสร้างวงจรเพื่อมาควบคุมอะไรสักอย่างนั้น ในบอร์ดควบคุมนั้นจะมีเพียงตัว CPU อย่างเดียวไม่ได้จะต้องมีการต่อ EPROM ภายนอกซึ่งทำหน้าที่เก็บคำสั่งต่างๆ ที่เราโปรแกรมไว้และต้องต่อพอร์ตขนานเพื่อควบคุมอุปกรณ์ภายนอก หรือถ้ามีการใช้งานเกี่ยวกับเวลาก็อาจจะต้องใช้วงจร Timer/Counter เข้ามาด้วย จะเห็นว่าอุปกรณ์ที่ใช้นั้นจะเริ่มมากขึ้นทำให้บอร์ดควบคุมนั้นใหญ่ขึ้นตามไปด้วย เหตุนี้จึงทำให้ต้นทุนที่ใช้ในการผลิตนั้นสูงขึ้นไปด้วย อีกทั้งยังมีขนาดใหญ่ไม่สะดวกต่อการพกพา ยังไม่รวมถึงพลังงานที่จ่ายให้กับบอร์ด ทำให้ต้องใช้แบตเตอรี่ขนาดใหญ่ขึ้น หรือไมก็ใช้งานได้ไม่นานแบตเตอรี่ก็หมด จึงได้ทำการออกแบบ Chip โดยการทำการรวมวงจรต่างๆ ที่กล่าวมา กับ CPU ไว้ใน Chip ตัวเดียวกัน เพื่อลดต้นทุน พลังงาน แล้วตั้งชื่อขึ้นมา ใหม่



ภาพที่ 2.3 ไมโครคอนโทรลเลอร์ MCU

2.1.2 ไมโครคอนโทรลเลอร์ MCU (Micro Controller Unit)

ในปัจจุบัน ไมโครคอนโทรลเลอร์ มีมากมายหลายเบอร์ที่อยู่ในท้องตลาดแต่ละเบอร์ก็จะมีหลายรุ่นให้เลือกใช้ตามต้องการ ตามความเหมาะสมทั้งในด้านความสามารถและราคา ไม่ว่าจะเป็นของบริษัทไหนโครงสร้างหรือหลักการทำงานที่ไม่ค่อยจะแตกต่างกันมากนัก และ เบอร์ที่นิยมใช้กันมากในปัจจุบันมีอยู่ 3-4 บริษัท

ตารางที่ 2.1 เบอร์ของไมโครคอนโทรลเลอร์ที่ได้รับความนิยมในปัจจุบัน

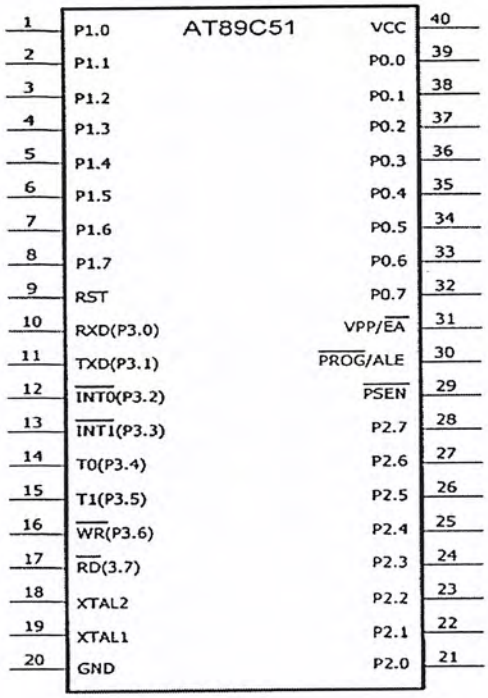
บริษัท	เบอร์	ข้อเด่น
Microchip	PIC 16Fxx,16F7x,16F8xx	Risc,เสถียรภาพดี
Atmel	AT89Cxx,AT89Sxx	เรียนรู้ง่าย คำสั่งมาก
Zilog	Z80Cxx,Z84Cxx	อ้างข้อมูลได้มาก
Motolola	68HCxxx	เสถียรภาพดี

เราใช้เบอร์ตระกูล MCS-51 พวก AT89xx ของบริษัท Atmel เหมาะสำหรับผู้ที่เริ่มต้นต้องการที่จะเรียนรู้ไมโครคอนโทรลเลอร์ ซึ่งเป็นเบอร์ที่นิยมในท้องตลาดมีการเครื่องพัฒนามากมาย ซึ่งมีคุณสมบัติทางเทคนิคดังนี้

1. เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
2. ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง
3. ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาต์พุต
4. ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
5. สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท
6. มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
7. มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ในภายชิป
8. มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
9. หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม
10. สามารถขยายหน่วยความจำภายนอกได้สูงสุด 64 กิโลไบต์
11. มีวอตช์ด็อกไทเมอร์ในตัว สำหรับในอนุกรม AT89Sxx

ซึ่งมีโครงสร้างพื้นฐานเหมือนกับไมโครคอนโทรลเลอร์ตระกลู MCS-51

วงจรพื้นฐาน MCS-51 และขาต่างๆของ MCS-51



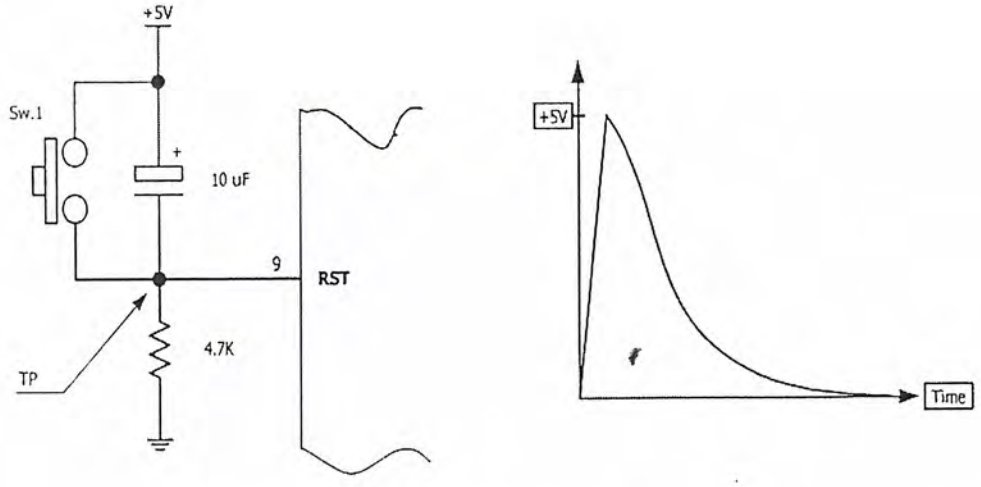
ภาพที่ 2.4 แสดงรูปร่างและขาต่างๆของ MCS-51

จากรูปข้างบนนั้นเป็นรูปร่างและขาต่างๆของตระกูล MCS-51

P0.0 - P0.7 (ขา39 - 32) เป็นพอร์ตใช้งานทั่วไปและพอร์ต Data และ Address มีขาทั้งหมด 8 ขาแต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต "0" ขาใดขาหนึ่งเป็นอินพุตเราก็ สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตที่ต้องการติดต่อ P1.0 - P1.7 (ขา1 - 8) เป็นพอร์ตใช้งานทั่วไป เป็นได้ทั้งพอร์ตอินพุตและเอาต์พุต นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าไทเมอร์

P2.0 - P2.7 (ขา 21 - 28) เป็นพอร์ตใช้งานทั่วไป

Reset (ขา9) ขานี้มีไว้สำหรับรีเซตการทำงานของ Mcs-51 ให้เริ่มต้นการทำงานขานี้จะทำงานที่ลอจิก "1" จะเห็นว่าไม่มีเครื่องหมาย - บนชื่อของขา ถ้ามีแสดงว่าทำงานที่ลอจิก "0" สิ่งที่ต้องออกแบบนั้นก็คือต้อง สร้างสัญญาณพัลส์บวก ให้ที่ขา RST ในช่วงเวลาที่จ่ายไฟให้กับ MCS-51 ตอนเปิดเครื่อง จะทำการรีเซตค่าตัวเองโดยอัตโนมัติซึ่ง วงจรเราสามารถ สร้างโดยค่า R และ C เพื่อสร้างพัลส์บวกได้ดังวงจรข้างล่าง



ภาพที่ 2.5 วงจร Reset ของ Mcs-51 และสัญญาณที่จุด TP

การทำงานของวงจร เมื่อเริ่มจ่ายไฟเข้าไปในวงจร C จะเริ่มชาร์จตัวเอง ตอนนี้ C จะเปรียบเสมือนช๊อต แรงดันตกคร่อมที่ C (VC) =0 แรงดันที่ตกคร่อมความต้านทาน (Vr) = 5V จากนั้นเมื่อเวลาผ่านไป C ก็จะเริ่มชาร์จไฟมากขึ้น ทำให้มีแรงดัน (VC) ตกครอมที่ตัวมันมากขึ้นเป็นผลทำให้มีแรงดันที่ความต้าน (Vr) ก็จะน้อยลง จนในที่สุด C ชาร์จไฟจนเต็มแรงดัน Vc = 5V แรงดันที่มีค่าความต้านทาน (Vr) =0 ตามรูปขวามือเวลาของสัญญาณพัลส์หรือกว้างนั้น ขึ้นอยู่กับค่า R และ C ค่าที่เหมาะสมของ R = 1K-10K , C = 1uF — 10 uF สวิทช์ที่ต่อคร่อม C นั้นทำหน้าที่เป็นตัว Discharge C และ Reset ให้กับตัว MCS-51 จะทำให้ Vr (ที่จุด TP) = VCC — Vc และ P3.0 —3.7 (ขา10 —17) นอกจากจะเป็นพอร์ตที่ใช้งานทั่วไปแล้วยังสามารถทำหน้าที่ได้ดังนี้

P3.0 , RXD สามารถเป็นขารับสัญญาณ Serial Port

P3.1 , TXD สามารถเป็นขาส่งสัญญาณ Serial Port

P3.2 , INTO สามารถเป็นขารับสัญญาณ Interrupt หมายเลข 0

P3.3 , INT1 สามารถเป็นขารับสัญญาณ Interrupt หมายเลข 1

P3.4 , TO สามารถเป็นขาที่รับสัญญาณพัลส์ หมายเลข 0 เพื่อเข้าวงจร Counter

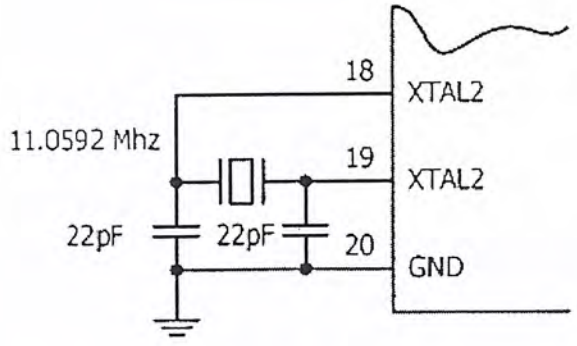
P3.5 , T1 สามารถเป็นขารับสัญญาณพัลส์ หมายเลข 1 เพื่อเข้าวงจร Counter

P3.6 , WR สามารถเป็นขาสัญญาณเขียนข้อมูลไปยังอุปกรณ์ภายนอก

P3.7 , RD สามารถเป็นขาสัญญาณอ่านข้อมูลจากอุปกรณ์ภายนอก

XTAL2 , 1 (ขา 18,19) ไว้สำหรับต่อ X – TAL เพื่อสร้างสัญญาณนาฬิกาให้กับ MCS-51 ให้สามารถทำงานได้ เพราะถ้าไม่มีการสร้างสัญญาณนาฬิกา MCS-51 ก็ไม่สามารถทำงานได้ ค่าของ X –TAL ที่เราใช้จะประมาณ 11.0592 Mhz เพราะเวลาที่ส่งข้อมูลผ่าน Serial Port นั้นจำเป็นที่ต้อง

สร้างความถี่ในการส่งข้อมูลให้ตรงกับ มาตราฐานค่า X – TAL ที่เราใช้นี้จะได้ความถี่ที่ตรงตาม มาตราฐานพอดีจึงเลือกใช้ค่านี้



ภาพที่ 2.6 แสดงการต่อวงจรของของ X – TAL

C ค่า 22 pF เป็นตัวป้องกันสัญญาณฮาร์โมนิก ที่เกิดจากวงจร Oscillator ที่อยู่ภายในตัว MCS-51 ค่าที่เหมาะสมของ C= 6pF – 30pF

PSEN (Program Status Enable)(ขา30) เป็นขาที่ทำหน้าที่บอกว่าตัวไมโครคอนโทรลเลอร์หรือ MCS-51 ติดต่อกับ Program Memory ขานี้จะทำงานที่ลอจิก "0" Program Memory ส่วนใหญ่คือ EPROM ที่ต่ออยู่ภายนอก หรือ FLASH ที่อยู่ภายในเบอร์ของ AT89Cxx ในกรนี้ที่ไม่ต่อ EPROM ภายนอก ก็จะปล่อยลอยไว้อย่างนั้น

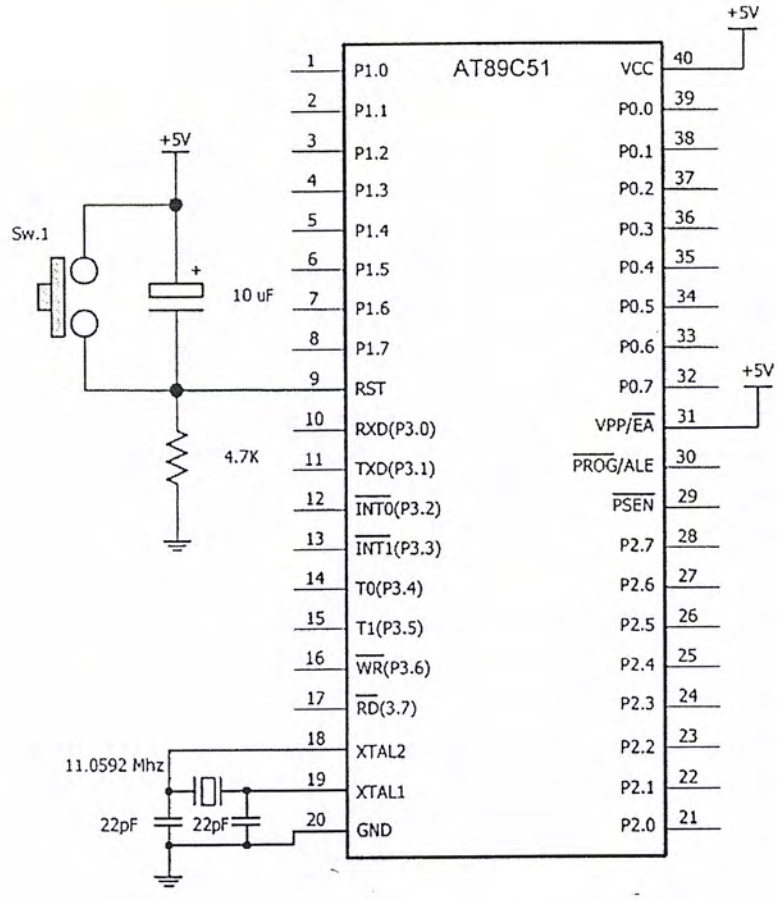
ALE (Address Latch Enable)(ขา30) ขานี้ใช้ในกรณีที่มีการต่อของตัว EPROM ,RAM, Chip Support ภายนอก ขานี้จะบอกให้รู้ว่าข้อมูที่พอร์ต "0" เป็น Address เพราะว่าพอร์ต 0 ของ MCS–51 นั้นออกแบบมาให้สามารถทำหน้าที่ได้ 2 อย่าง คือ เป็นทั้ง Data และ Address ในขณะที่พอร์ต 0 ก็จะมีข้อมูลเป็น Address ขา ALE จะเป็น "1" เพื่อส่งสัญญาณให้กับไอซีพวก 44373,74374ซึ่งจะต่ออยู่กับ พอร์ต 0 ซึ่งเอาท์พุทของไอซีตัวนี้มีค่าเป็น Address (A0 – A7)

EA (External Access)(ขา31) MCS-51 สามารถติดต่อกับหน่วยความจำได้ 3 แบบคือ

1. หน่วยความจำภายใน (Internal Memory) เป็นหน่วยความจำภายในที่อยู่ใน ภายในของตัวของ MCS – 51 ซึ่งจะเป็นพวกรีจีสเตอร์ต่างๆ

2. หน่ายความจำข้อมูลภายนอก (Data Memory) ซึ่งสามารถอ้างข้อมูลได้ 64 Kbyte เป็นส่วนที่ทำหน้าที่เก็บข้อมูล จะเป็นหน่วยความจำแบบ RAM

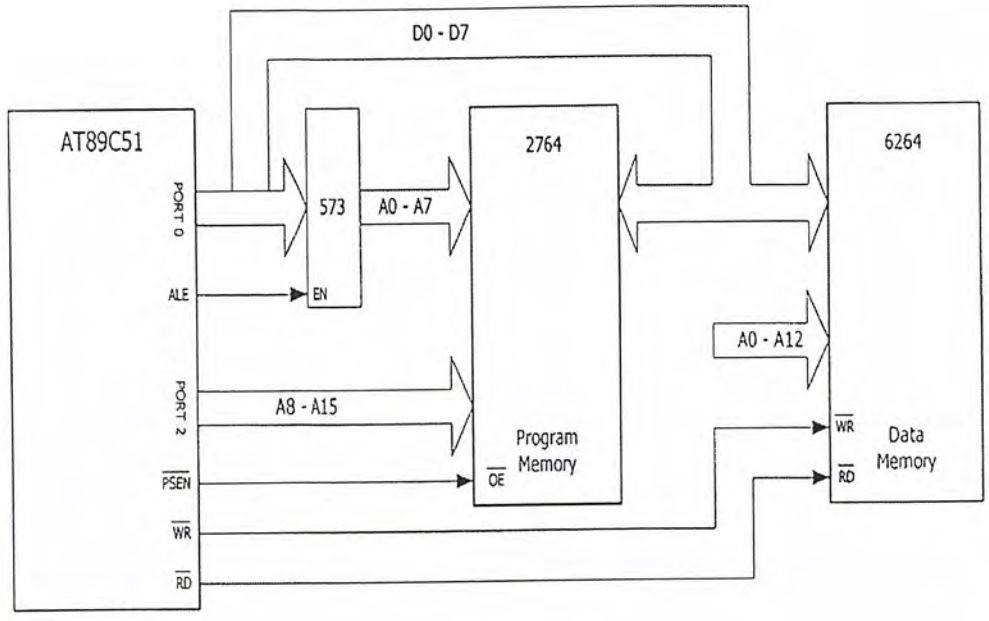
3. หน่วยความจำโปรแกรมภายนอก (Program Memory) สามารถอ้างข้อมูลได้ 64 Kbyte เป็นส่วนที่ทำหน้าที่เก็บคำสั่งที่เราเขียน ส่วนใหญ่เก็บไว้ที่ EPROM



ภาพที่ 2.7 แสดงวงจรการใช้งานของ MCS-51

2.2 รีจีสเตอร์และหน่วยความจำ

หน่วยความจำ (Memory) ในตัวของไมโครคอนโทรลเลอร์ MCS-51 ซึ่งเป็นแบบแฟลชจะมีหน่วยความจำภายในหลักๆอยู่ 2 ส่วนคือหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ซึ่งมีขนาด และการจัดสรรแตกต่างกันไปในแต่ละเบอร์ หน่วยความจำโปรแกรมใช้เก็บข้อมูลของโปรแกรมที่ใช้ควบคุมการทำงานของไมโครคอนโทรเลอร์ หรือที่เรียกว่าโปรแกรมมอนิเตอร์หากใช้หน่วยความจำภายนอกมักจะบรรจุอยู่ในหน่วยความจำของชนิดอีพรอม MCS-51 สามารถติดต่อได้กับหน่วยความจำได้ทั้งหมด 3 ประเภท



ภาพที่ 2.7 หน่วยความจำ (Memory)

2.2.1 หน่วยความจำภายนอก (Program Memory)

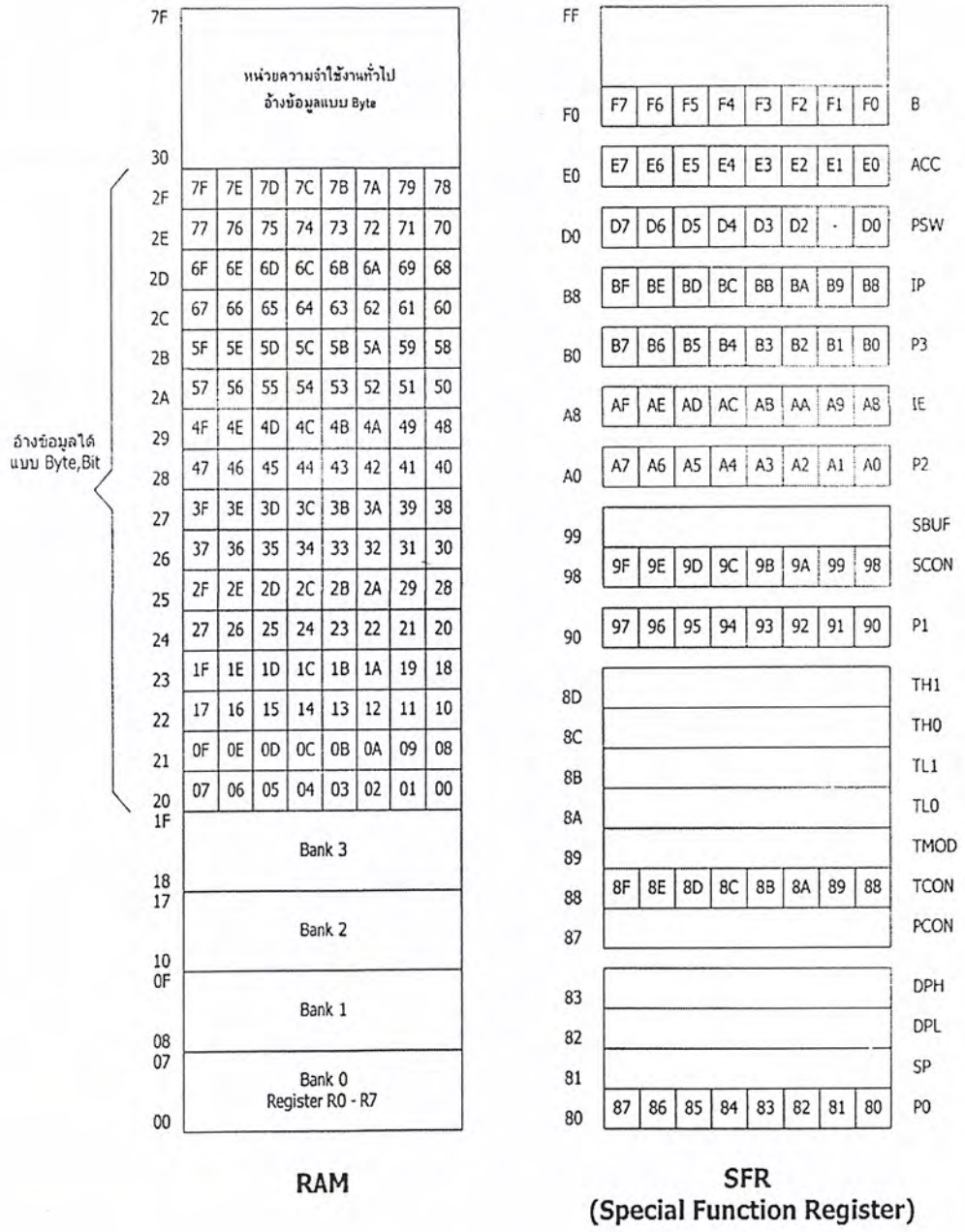
หน่วยความจำชนิดทำหน้าที่เก็บคำสั่งของผู้เขียนโปรแกรม ส่วนใหญ่จะใช้ EPROM เป็นตัวเก็บ โดยจะมีสัญญาณ PSEN เป็นขาสัญญาณเพื่อใช้ติดต่อกับตัว EPROM ที่สามารถอ้างถึงหน่วยความจำได้ ถึง64 Kbyte ส่วนใหญ่ออกแบบให้ Program Memory เริ่มต้นที่ Address 0000H เพราะเวลาที่ MCS-51 ถูกรีเซต MCS-51 จะกระโดดไปที่ Address 0000H ของ Program Memory EPROM นั้นส่วนใหญ่จะขึ้นต้นด้วย 27 แล้วตามด้วยขนาดของหน่วยความจำ เช่น 2764 หมายเลข 64 คือขนาดหน่วยความจำ โดยเอาค่า 64/8 = 8 Kbyte เราก็จะได้ขนาดของ EPROM ตัวนี้

2.2.2 หน่วยความจำข้อมูลภายนอก (Data Memory)

หน่วยความจำชนิดนี้ทำหน้าที่ เก็บข้อมูลต่างๆที่เราต้องการ ซึ่งก็คือ RAM ที่ต่ออยู่ภายนอก สามารถเข้าถึงข้อมูลโดยใช้ ขา RD,WR ในการเขียนและอ่าน Data จาก RAM MCS-51 สามารถอ้างหน่วยความจำข้อมูลภายนอกได้ 64 Kbyte เช่นกัน ส่วนใหญ่แล้วหน่วยความจำนี้จะถูกจัด Address ให้ถัดจาก Program Memory (EPROM) RAM ที่ใช้ส่วนใหญ่จะเป็น Static RAM เบอร์จะขึ้นด้วยหมายเลข 61,62,64,68 แล้วตามด้วยขนาดของหน่วยความจำเช่น 6264 หมายเลข 64 คือขนาดของหน่วยความจำ โดยเอาค่า 64/8 = 8 kbyte เราก็จะได้ขนาดของ RAM ตัวนี้ คำสั่งที่ใช้ในการเอาข้อมูลจาก Data Memory นั้นคือคำสั่งพวก MOVX

2.2.3 หน่วยความจำภายใน (Interal RAM)

เป็นหน่วยความจำภายใน ซึ่งเป็น RAM ในหน่วยความจำนั้น ก็จะประกอบด้วย Register ซึ่งจะมี Address ประจำตัว ทำหน้าที่ใช้งานแตกต่างกันไปตามการใช้งาน ซึ่งจะเป็นรูปข้างล่างนี้แสดงรีจีสเตอร์ของหน่วยความจำภายใน



ภาพที่ 2.9 โครงสร้างของหน่วยความจำข้อมูลภายในของไมโครคอนโทรลเลอร์

จากรูปข้างบนจะแสดงให้เห็นว่า Address ของหน่วยความจำเป็น Address ที่ทำหน้าที่เป็นรีจีสเตอร์ ได้แบ่งประเภทของหน่วยความจำได้ออกเป็น 4 กลุ่มดังนี้

1. กลุ่ม Register พื้นฐาน

เป็นกลุ่มที่สามารถนำไปใช้งานได้โดยตรง นั้นคือก็ Register R0-R7 ซึ่งจะแบ่ง Address ของหน่วยความจำออกเป็นช่วงๆหรือที่เรียกว่า Bank ตั้งแต่ Bank0 – Bank7 ดังนี้

Bank0 จะได้ R0-R7 ที่อยู่ Address 00H-07H

Bank1 จะได้ R0-R7 ที่อยู่ Address 08H-0FH

Bank2 จะได้ R0-R7 ที่อยู่ Address 10H-17H

Bank3 จะได้ R0-R7 ที่อยู่ Address 18H-1FH

เมื่อเริ่มจ่ายไฟให้ MCS-51 รีจีสเตอร์ R0-R7 จะถูกกำหนดให้ Address อยู่ใน Bank0 จากนั้นสามารถเซตได้ว่าจะให้รีจีสเตอร์ R0-R7 อยู่ที่ Bank ไหน โดยเซตที่รีจีสเตอร์ที่ชื่อว่า PSW ซึ่งอยู่ในส่วนของกลุ่มของ SFR(Special Function Registers)

2. กลุ่มหน่วยความจำที่สามารถอ้างเป็น Byte และ Bit

กลุ่มนี้จะเริ่มต้นที่ Address 20H-2FE ทั้งหมดมี 16 Byte= 128 Bit จะมีตำแหน่งกำกับบิตซึ่งอยู่ข้างบน ซึ่งในส่วนนี้จะสามารถเซตหรือเคลียร์บิตนั้นได้ อันนี้เป็นความสามารถเฉพาะตัวของ MCS-51 ยกตัวอย่างเช่น SETB 2BH คำสั่งนี้จะทำการเซตบิตที่ 2BH ให้เป็นลอจิก "1" นั้นก็คือเป็นบิตค่าที่ 3 ของ Address 25H นั้นเอง ส่วนคำสั่งเคลียร์ก็คือ CLR 2BH เป็นการเคลียร์บิตนั้นให้กับ "0" รีจีสเตอร์บางตัวที่อยู่ในกลุ่มของ SFR ก็สามารถเซตเป็นบิตได้เหมือนกัน

3. กลุ่มหน่วยความจำที่สามารถอ้างเป็น Byte ได้อย่างเดียว

กลุ่มนี้จะอยู่ Address 30H-7FH จะเอาไว้ใช้งานทั่วไป เช่นจะเอาไว้ค่าต่างๆ เวลาเราเขียนโปรแกรม ซึ่งสามารถทำคำสั่งที่เป็นไบต์ ได้อย่างเดียว ไม่สามารถทำเป็นบิตได้

4. กลุ่มรีจีสเตอร์พิเศษ SFR (Special Function Registers)

จะประกอบไปด้วยรีจีสเตอร์ต่างๆ เช่น รีจีสเตอร์คำนวณ และรีจีสเตอร์สำหรับการ Intrerupt รีจีสเตอร์สำหรับวงจร Timer/Counter รีจีสเตอร์สำหรับตัว Serial port ซึ่งรีจีสเตอร์แต่ละตัวนั้นก็จะมีรายละเอียดดังนี้

รีจีสเตอร์ P1–P3 จะเป็นรีจีสเตอร์ที่ใช้เชื่อมต่อตรงจากพอร์ต P1-P3 ก็คือข้อมูลที่รีจีสเตอร์ P1-P3 เป็นข้อมูลอะไรที่พอร์ต P1-P3 ก็จะเป็นอย่างนั้น รีจีสเตอร์ทั้ง 4 ตัวนี้ซึ่งสามารถทำเป็นให้

Bit Addressing ได้คือสามารถเซตและเคลียร์เป็นบิตนั้นได้นั้นเองรีจีสเตอร์ทั้ง 4 ตัวนี้อยู่ที่ Address 80H , 90H , A0H , B0H ตามลำดับ

รีจีสเตอร์ ACC เป็นรีจีสเตอร์ที่ใช้ในการคำนวณทางคณิตศาสตร์ และกระทำทางลอจิกในการโอนถ่ายข้อมูลต่างๆเป็นรีจีสเตอร์ที่ใช้บ่อยมากๆสามารถอ้างเป็นบิตได้รีจีสเตอร์นี้อยู่ที่ Address E0H

รีจีสเตอร์ B เป็นรีจีสเตอร์ใช้งานทั่วไปและเป็นรีจีสเตอร์ที่ทำหน้าที่เป็นตัวหารและคูณกับรีจีสเตอร์ A เท่านั้น

รีจีสเตอร์ PSW (Program StatusWord) เป็นรีจีสเตอร์ที่แสดงสถานะของรีจีสเตอร์ A ว่าเป็นอย่างไร

ตารางที่ 2.2 บิตต่างๆของรีจีสเตอร์ PSW

CY	AC	F0	RS1	RS0	OV	-	P

CY (Carry Flag) บิตนี้เป็นบิตที่7 ของรีจีสเตอร์ PSW.7 ซึ่งบิตนี้จะถูกเซตให้เป็น "1" เมื่อรีจีสเตอร์ A เป็นบวกกับค่าใดกับค่าหนึ่งและแล้วได้ผลลัพธ์เกิน FFH และเมื่อต้องลบกับค่าใดค่าหนึ่งโดยที่ค่าของรีจีสเตอร์ A มีค่าน้อยกว่า ผลลัพธ์ที่ได้ก็จะมีค่าเป็นติดลบ

AC (Auxiliary Carry Flag) บิตนี้จะเป็น "1" เมื่อมีการบวกแบบค่าที่ได้เข้าด้วยกัน BCD(Binary Code Deciman) คือเป็นการบวกแบบเลขฐานสิบ คือนำค่ามาบวกจะมีแค่ 0-9 ไม่มีค่า A-F เมื่อบวกกันแล้วมีการทดเลขจากบิตที่ 3 ไปบิตที่ 4 ของรีจีสเตอร์ A

ตารางที่ 2.3 ตำแหน่ง Address ของรีจีสเตอร์ R0-R7

RS1	RS0	BANK	ADDRESS
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

OV (Over Flow Flag) บิตนี้จะถูกเคลียร์ให้เป็น "0"ทันที เมื่อมีการกระทำค่ากันระหว่างกันทางคณิตศาสตร์ของรีจีส เตอร์ A แล้วค่าเกินเป็นบวก 0 – 127 (00H-7F) และบิตนี้จะเป็น "1" ทันที เมื่อผลที่ได้มีค่าเป็น (-1)-(-128)
(80H-FFH) นั้นก็คือเราจะกำหนดให้บิตที่ 7 ของรีจีสเตอร์ เป็นเครื่องหมายบวกลบ คือถ้าบิตที่ 7 เป็น "0" แสดงว่าเป็นบวกและถ้าเป็น "1" ก็แสดงว่าเป็นลบ

P (Parity Bit) บิตนี้จะเป็น "1" ก็ต่อเมื่อจำนวนบิตที่เป็นลอจิก "1" ของรีจีสเตอร์ A มีค่าเป็นคี่ (Odd Parity) และบิตนี้จะเป็น "0" ก็ต่อเมื่อจำนวณบิตเป็นลอจิก "1" ของรีจีสเตอร์ A เป็นคู่ (Even Parity) ส่วนใหญ่บิตนี้จะใช้ในกรณีสื่อสารข้อมูล Serial Port

DPTR (Data Pointer) เป็นรีจีสเตอร์ที่ทำหน้าที่ชี้ Address ของหน่วยความจำภายนอก จึงมีขนาด 16 บิต รีจีสเตอร์ตัวนี้จะประกอบไปด้วยค่ารีจีสเตอร์ DPH,DPL ขนาด 8 บิต โดย DPH เป็นรีจีสเตอร์ไบต์สูง อยู่ Address 83H และ DPL เป็นรีจีสเตอร์ไบต์ต่ำ อยู่ที่ Address 82H

SP (Stack Pointer) เป็นรีจีสเตอร์ที่ทำหน้าที่ชี้ตำแหน่งของหน่วยความจำภายใน ตำแหน่งมันชี้เราจะเรียกว่า Stack ทำหน้าที่เก็บขิอมูลต่างๆที่ MCS-51 ต้องการเก็บโดยมันจะมีการเก็บค่าของข้อมูลของ MCS-51 จากคำสั่งพวก Call,Push,Pop จากการอินเตอร์รัพท์

2.3 ชุดคำสั่งของไมโครคอนโทรลเลอร์ MCS-51

สามารถแบ่งออกได้ 4 กลุ่ม

1. กลุ่มคำสั่งโอนย้ายข้อมูล (data transfer instructions)
2. กลุ่มคำสั่งทางคณิตศาสตร์ (arithmatic instructions)
3. กลุ่มคำสั่งทางลอจิก (logical instructions)
4. กลุ่มคำสั่งจัดการข้อมูลระดับบิต (bit manipulated instructions)

การทำงานในแต่ละคำสั่งของตัวไมโครคอนโทรลเลอร์ MCS-51 ที่เป็นแบบแฟลชซึ่งจะนำไปใช้เวลาในการประมวลผลที่แตกต่างกัน โดยจะนับเป็นหน่วยของรอบการทำงานหรือที่เราเรียกว่าแมชชีนไซเกิล (machine cycle) โดยในคำสั่งใดมีค่าแมชีนไซเคิลมาก ซีพียูต้องใช้เวลามากในการประมวลผลด้วยเช่นกัน ในทุกคำสั่งของตัวไมโครคอนโทรเลอร์ MCS-51 จะมีรูปแบบมาตรฐานเดียวกัน ประกอบด้วยส่วนสำคัญ 2 ส่วนคือส่วน ที่เป็นรหัสคำสั่งหรือเรียกว่านีโมนิก และส่วนของตัวแปรที่ใช้ดำเนินการหรือโอเปอร์แรนด์โดยในส่วนรหัสคำสั่งนั้น มักเป็นคำของชุดคำสั่งของตัวไมโครคอนโทรลเลอร์ MCS-51 จะมีรูปแบบคือ MOV A,R0 จากคำสั่งข้างต้น รหัสคำสั่งหรือค่าของนีโมนิกคือ MOV ส่วนโอเปอแรนด์ คือA,R0 ส่วนกลุ่มคำสั่ง โอนย้ายข้อมูล เป็นกลุ่มคำสั่งที่ใช้ในการโอนย้ายข้อมูลระหว่างรีจีสเตอร์ ระหว่างหน่วยความจำด้วยกันและ การโอนย้ายข้อมูลกับหน่วยความจำภายนอก การสลับข้อมูลและคำสั่งที่ใช้ในการเก็บหรือเรียกข้อมูลออกจากสแต็ก

2.3.1 กลุ่มคำสั่งการโอนย้ายข้อมูลกับหน่วยความจำข้อมูลแรมภายใน

สามารถแบ่งย่อยได้อีกเป็น 5 กลุ่มดังนี้

1. คำสั่งสำหรับโอนย้ายข้อมูลมาเก็บไว้ที่แอคคิวมูเลเตอร์

MOV A,Rn การทำงานจะนำข้อมูลของรีจีสเตอร์ R0-R7 มาเก็บไว้ในแอกคิวมูลเลเตอร์ จำนวน 1 ไบต์และจำนวน 1 ไซเกิล เช่น MOV A,R0

MOV A,direct การทำงานจะนำข้อมูลของหน่วยความจำข้อมูลภายในของคอลโทรลเลอร์ มาเก็บในแอกคิวมูลเลเตอร์ จำนวน 2 ไบต์และจำนวน 1 ไซเกิล เช่น MOV A,20H (ค่า20H เป็นแอดเดรสของหน่วยความจำข้อมูลภายใน

MOV A,#data การทำงานจะนำข้อมูลมาเก็บไว้ที่แอกคิวมูลเลเตอร์ จำนวน 2 ไบต์และจำนวน 1 ไซเกิล เช่น MOV A,#40H (นำค่า 40H เป็นข้อมูลหรือค่าตัวเลข)

MOV A,@Rn การทำงานจะ นำข้อมูลจากแอดเดรสของหน่วยความจำข้อมูลภายใน ที่กำหนดไว้ในรีจีสเตอร์R0 หรือ R1 มาเก็บไว้ในแอกคิวมูลเลเตอร์จำนวน 1ไบต์และจำนวน 1ไซเกิล เช่น MOV R0,20H กำหนดค่าแอดเดรสให้รีจีสเตอร์ R0 และ MOV A,@R0 นำข้อมูลในแอดเดรส 20H มาเก็บในแอคคิวมูลเลเตอร์

2. คำสั่งสำหรับโอนย้ายข้อมูลมาเก็บไว้ในแอกคิวมูลเลเตอร์แบงค์ R0-R7

MOV Rn,A การทำงานนำข้อมูลจากแอกคิวมูลเลเตอร์ไปเก็บไว้ในรีจีสเตอร์ R0-R7 จำนวน 1ไบต์และ จำนวน 1 ไซเกิล เช่น MOV R0,A

MOV Rn,direct การทำงานจะนำข้อมูลจากแอดเดรสของค่าในหน่วยความจำข้อมูลภายในที่กำหนดมาเก็บไว้ใน รีจีสเตอร์ R0-R7 จำนวน 2 ไบต์และจำนวนไซเกิล 2 ไซเกิล เช่น MOV R0,40H(ค่า 40Hจะเป็นแอดเดรสของหน่วยความจำภายใน) หรือ MOV R7,20H (ค่า 20H เป็นแอดเดรสของหน่วยความจำภายใน)

MOV Rn,#data การทำงานจะนำข้อมูลมาเก็บไว้ที่รีจีสเตอร์ R0-R7 จำนวน 2 ไบต์และจำนวน 1 ไซเกิล เช่น MOV R0,#40H (40H เป็นข้อมูลหรือตัวเลข) หรือ MOV R7,20H (ค่า 20H จะเป็นข้อมูลหรือตัวเลข)

3. คำสั่งในการโอนย้ายข้อมูลมาเก็บไว้ในหน่วยความจำข้อมูลภายในโดยตรง

MOV direct,A การทำงานจะเป็นการนำเอาข้อมูลจากแอกคิวมูลเลเตอร์เพื่อมาเก็บไว้ที่หน่วยความจำข้อมูลภายในจำนวน 2 ไบต์และจำนวน 1 ไซเกิล เช่นคำสั่ง MOV 40H,A (40Hเป็นแอดเดรสของหน่วยความจำข้อมูลภายใน)

MOV direct,Rn การทำงานจะเป็นการนำข้อมูลที่อยู่ภายในรีจีสเตอร์ R0-R7 เพื่อมายังเก็บไว้ที่หน่วยความจำข้อมูลภายใน จำนวน 2ไบต์และจำนวน 2 ไซเกิล เช่น MOV 40H,R0 (40Hเป็นค่าของแอดเดรสของหน่วยความจำข้อมูลภายใน)

MOV direct,direct การทำงานจะเป็นการนำข้อมูลที่ได้จากหน่วยความจำของข้อมูลภายในแอดเดรสหนึ่งเพื่อนำมาเก็บไว้ที่หน่วยความจำข้อมูลภายในอีกแอดเดรสหนึ่ง จำนวน 3 ไบต์และจำนวน 2 ไซเกิล เช่น MOV 40H,50H(ค่า 50Hเป็นแอดเดรสของหน่วยความจำข้อมูลภายในต้นทาง ส่วนค่า 40H เป็นแอดเดรสของข้อมูลภายในปลายทาง)

MOV direct,#data การทำงานเป็นการ นำข้อมูลที่ได้จากแอกคิวมูเลเตอร์เพื่อนำมาเก็บไว้ที่หน่วยความจำข้อมูลภายใน จำนวน 3 ไบต์และจำนวน 2 ไซเกิล

4. คำสั่งสำหรับโอนย้ายข้อมูลมาเก็บไว้ในหน่วยความจำข้อมูลภายในโดยอ้อมผ่านทางรีจีสเตอร์ R0 หรือ R1

MOV @Rn,A การทำงานเป็นการนำข้อมูล จากแอกคิวมูลเลเตอร์เพื่อนำค่าที่ได้มาเก็บไว้ที่หน่วยความจำข้อมูลภายในที่กำหนด โดยค่าของรีจีสเตอร์ R0 หรือ R1 จำนวน 1 ไบต์ และ 1 ไซเกิล

MOV @Rn,direct การทำงานเป็นการนำข้อมูล จากหน่วยความจำข้อมูลภายในแอดเดรสหนึ่งมาเก็บไว้ที่หน่วยความจำข้อมูลภายในอีกแอดเดรสหนึ่ง ที่กำหนดโดยค่าของ R0 หรือ R1 จำนวน 2 ไบต์ และ 2 ไซเกิล

MOV @Rn,#data การทำงานเป็นการนำข้อมูลที่ได้ไปเก็บไว้ที่หน่วยความจำข้อมูลที่กำหนดโดยค่า R0 หรือ R1 จำนวน 3 ไบต์ และ 1 ไซเกิล

5. คำสั่งโอนย้ายข้อมูลระดับบิต

MOV C,bit การทำงานเป็นการนำข้อมูลระดับบิต จากหน่วยความจำภายใน มาเก็บในแฟลกทด ซึ่งอยู่ในรีจีสเตอร์ PSW จำนวน 2 ไบต์และจำนวน 1 ไซเกิล

MOV bit,C การทำงานเป็นนำข้อมูลในแฟลกทดไปเก็บไว้ที่หน่วยความจำข้อมูลภายในที่สามารถเข้าถึงระดับบิตได้ จำนวน 2 ไบต์และจำนวน 2 ไซเกิล

2.3.2 กลุ่มคำสั่งในการแลกเปลี่ยนข้อมูล

การแลกเปลี่ยนข้อมูลเป็นการโอนย้ายข้อมูล รูปแบบหนึ่งที่ทำการสลับค่าของข้อมูลระหว่างต้นทางกับข้อมูลปลายทาง คำสั่งในกลุ่มนี้มีอยู่ 4 คำสั่ง

XCH A,Rn การทำงานจะเป็นการแลกเปลี่ยนข้อมูลระหว่างรีจีสเตอร์A กับข้อมูลที่อยู่ภายในรีจีสเตอร์ R0-R7 จำนวน 1 ไบต์ และ 1 ไซเกิล

XCH A,direct การทำงานเป็นการแลกเปลี่ยนข้อมูลระหว่างตัวของค่ารีจีสเตอร์ A กับหน่วยความจำข้อมูลภายใน จำนวน 2 ไบต์ และ 1 ไซเกิล

XCH A,@Rn การทำงานเป็นการแลกเปลี่ยนข้อมูลระหว่างตัวของ A กับข้อมูลภายในแอดเดรส ที่ถูกชี้โดย R0 หรือ R1 จำนวน 1 ไบต์ และ 1 ไซเกิล

XCHD A,@Rn การทำงานเป็นการแลกเปลี่ยนข้อมูลบิต 0- 3 ของรีจีสเตอร์ A กับข้อมูลบิต 3- 0 ภายในแอดเดรสของ หน่วยความจำที่ชี้โดยรีจีสเตอร์ R0 หรือ R1 จำนวน 1 ไบต์และ 1 ไซเกิล

2.3.3 กลุ่มคำสั่งทางคณิตศาตร์

เป็นกลุ่มคำสั่งที่ใช้ในการคำนวณค่าทางคณิตศาสตร์ไม่ว่าจะเป็นการบวก ลบ คูณ หรือหาร ในไมโครคอนโทรลเลอร์MCS-51 การกระทำทางคณิตศาสตร์จำเป็นต้องกระทำกับค่ารีจีสเตอร์ A หรือค่าแอกคิวมูลเลเตอร์เป็นหลักและผลลัพธ์ที่ได้จากการคำนวก็จะถูกเก็บไว้ในแอกคิวมูลเลเตอร์เสมอ ซึ่งกลุ่มคำสั่งทางคณิตศาสตร์ที่ใช้ในไมโครคอนโทรลเลอร์มีอยู่ 4 กลุ่ม

1. กลุ่มคำสั่งการบวก

กลุ่มคำสั่งทางบวกแบ่งออกเป็น 2 ลักษณะคือคำสั่งการบวกแบบไม่คิดตัวทด และคำสั่งการบวกที่คิดตัวทด กลุ่มคำสั่งการบวกประกอบด้วยคำสั่งและมีรายละเอียดดังต่อไปนี้

ADD A,#data การทำงานเป็นการบวกค่าในแอกคิวมูลเลเตอร์เข้ากับข้อมูล data ขนาด 8 บิตแล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูลเลเตอร์

ADD A,direct การทำงานจะทำการบวกค่าในแอกคิวมูเลเตอร์ เข้ากับข้อมูลในหน่วยความจำข้อมูลภายในแล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

ADD A,Rn การทำงานจะทำการบวกค่าในแอกคิวมูลเลเตอร์ เข้ากับข้อมูลในรีจีสเตอร์ R0-R7 ขนาด 8 บิตแล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูลเลเตอร์

ADD A,@Rn การทำงานเป็น การบวกค่าของแอกคิวมูลเลเตอร์เข้ากับข้อมูล 8 บิตในแอดเดรสของหน่วยความจำที่ถูกชี้โดย R0 หรือ R1แล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูลเลเตอร์

ADDC A,#data การทำงานเป็นการบวกค่าในแอกคิวมูลเลเตอร์เข้ากับแฟลกทด แล้วบวกกับข้อมูล data ขนาด 8 บิต แล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูลเลเตอร์

ADDC A,Rn การทำงานเป็นการบวกค่าในแอกคิวมูลเลเตอร์เข้ากับค่าของแฟลกทด แล้วบวกกับข้อมูลในรีจีสเตอร์ R0-R7 ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในแอกคิวมูลเลเตอร์

2. กลุ่มคำสั่งในการลบ

SUBB A,#data การทำงาน ทำการลบค่าในแอกคิวมูลเลเตอร์ด้วยค่าของแฟลกทด แล้วลบด้วย ข้อมูล data ขนาด 8 บิต นำผลลัพธ์ไปเก็บไว้ในแอกคิวมูลเลเตอร์

SUBB A,direct การทำงาน ทำการลบค่าในแอกคิวมูลเลเตอร์ด้วยค่าของแฟลกทด แล้วลบด้วยข้อมูลในหน่วยความจำข้อมูลภายในนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

SUBB A,Rn การทำงาน ทำการลบค่าในแอกคิวมูลเลเตอร์ด้วยค่าของแฟลกทด แล้วลบด้วยค่าข้อมูลในรีจีสเตอร์ R0- R7 ขนาด 8 บิตนำผลลัพธ์ไปเก็บในแอกคิวมูลเลเตอร์

SUBB A,@Rn การทำงาน ทำการลบค่าในแอกคิวมูลเลเตอร์ด้วยค่าของแฟลกทด แล้วลบด้วยข้อมูลในหน่วยความจำที่ถูกชี้โดย R0 หรือ R1 นำผลลัพธ์เก็บในแอกคิวมูลเลเตอร์

3. กลุ่มคำสั่งการคูณและการหาร

ในไมโครคอนโทรลเลอร์ MCS-51 มีคำสั่งที่เป็นการคูณและการหารข้อมูล โดยเราต้องกระทำผ่านรีจีสเตอร์ A หรือแอกคิวมูลเลเตอร์ B เท่านั้น เมื่อกระทำคำสั่งการคูณซึ่ง ผลลัพธ์ที่ได้จะเก็บไว้ในค่ารีจีสเตอร์ A และรีจีสเตอร์ B กรณีที่กระทำคำสั่งการหารต้องกำหนดให้ค่าในรีจีสเตอร์ A หรือค่าแอกคิวมูลเลเตอร์เป็นตัวตั้ง รีจีสเตอร์ B เป็นตัวหาร

MUL AB การทำงาน ทำการคูณค่าในอิกคิวมูลเลเตอร์ด้วยค่าในรีจีสเตอร์ B นำผลลคูณไบต์ล่างเก็บไว้ในแอกคิวมูลเลเตอร์ และผลคูณไบต์บนเก็บไว้ในรีจีสเตอร์ B

DIV AB การทำงาน ทำการหารค่าในแอกคิวมูลเลเตอร์ ด้วยค่าในรีจีสเตอร์ B นำผลหารซึ่งก็คือข้อมูลไบต์บนไปเก็บไว้ในแอกคิวมูลเลเตอร์ เศษของการหารคือข้อมูลไบต์ล่าง ซึ่งจะนำไปเก็บไว้ในรีจีสเตอร์ B

4. กลุ่มคำสั่งเพิ่ม และลดค่า

เป็นกลุ่มคำสั่งที่ใช้ในการเพิ่ม และลดค่าของข้อมูล ค่าของรีจีสเตอร์ และค่าของแอดเดรสของหน่วยความจำ

INC A การทำงาน ทำการเพิ่มค่าในแอกคิวมูลเลเตอร์ขึ้นหนึ่งค่า แล้วนำค่าที่เพิ่มขึ้นนั้นไปเก็บไว้ในแอกคิวมูลเลเตอร์

INC direct การทำงาน ทำการเพิ่มค่าของข้อมูลในหน่วยความจำข้อมูลภายในขึ้นหนึ่งค่า

DEC A การทำงาน ทำการลดค่าในแอกคิวมูลเลเตอร์ลงหนึ่งค่า แล้วนำค่าที่ลดลงนี้ไปเก็บไว้ในรีจีสเตอร์A

DEC direct การทำงานจะทำการลดค่าของข้อมูลในหน่วยความจำข้อมูลภายในลงหนึ่งค่าเรียงต่อๆกันไป

DEC Rn การทำงานจะทำการลดค่าข้อมูลในรีจีสเตอร์ R0-R7 ลงหนึ่งค่า

2.3.4 กลุ่มคำสั่งทางลอจิก

เป็นกลุ่มคำสั่งที่ใช้ในการประมวลผลทางตรรกหรือลอจิก ไม่ว่าจะเป็นคำสั่ง AND , OR และ EX-OR operation

1. กลุ่มคำสั่งแอนด์ (AND operation)

เป็นกลุ่มคำสั่งที่ใช้ ทำการแอนด์ค่าข้อมูลระหว่าง รีจีสเตอร์กับรีจีสเตอร์ และค่ารีจีสเตอร์กับหน่วยความจำ และรีจีสเตอร์กับข้อมูลตัวเลข เช่น

ANL A,#data แอนด์ค่าในแอกคิวมูลเลเตอร์กับข้อมูล data ขนาด 8 บิตนำผลลัพธ์ไปเก็บในแอกคิวมูลเลเตอร์

ANL A,direct แอนด์ค่าของข้อมูลในหน่วยความจำข้อมูลในหน่วย ความจำข้อมูลภายในกับค่ารีจีสเตอร์ A นำผลลัพธ์ไปเก็บในรีจีสเตอร์ A

2. กลุ่มคำสั่งออร์ (OR operation)

เป็นกลุ่มคำสั่งที่ทำการออร์ข้อมูลระหว่างรีจีสเตอร์กับรีจีสเตอร์ รีจีสเตอร์กับหน่วยความจำ และรีจีสเตอร์กับข้อมูลตัวเลข เช่นคำสั่ง

ORL A,#data ทำการออร์ค่าในแอกคิวมูลเลเตอร์ กับข้อมูล data ขนาด 8 บิต นำผลลัพธ์เก็บไว้ในแอกคิวมูเลเตอร์

ORL A,direct ทำการออร์ค่าของข้อมูลในหน่วยความจำข้อมูลภายในกับค่าของรีจีสเตอร์ A นำผลลัพธ์ไปเก็บไว้ในรีจีสเตอร์ A

3. กลุ่มคำสั่งเอ็กคลูซีฟ-ออร์ (EX-OR operration)

เป็นกลุ่มคำสั่งที่ทำการเอ็กคลูซีฟ-ออร์ข้อมูลระหว่างรีจีสเตอร์กับรีจีสเตอร์ หรือรีจีสเตอร์กับหน่วยความจำ รีจีสเตอร์กับข้อมูลตัวเลข เช่น

XRL A,#data ทำการเอ็กคลูซีฟ-ออร์ค่าในแอกคิวมูเลเตอร์กับข้อมูล data ขนาด 8 บิตแล้วนำผลลัพธ์ไปเก็บไว้ในแอกคิวมูเลเตอร์

XRL A,direct ทำการเอ็กคลูซีฟ-ออร์ ค่าของข้อมูล ในหน่วยความจำของข้อมูลซึ่งจะอยู่ภายในกับค่าของข้อมูลในรีจีสเตอร์ A

4. กลุ่มคำสั่งเปลี่ยนแปลงค่าของแอกคิวมูลเลเตอร์

ซึ่งจะประกอบด้วย คำสั่งเคลียร์ค่าของแอกคิวมูลเลเตอร์ รวมทั้งคำสั่งสำหรับกลับข้อมูลของแอกคิวมูลเลเตอร์ คำสั่งหมุนหรือเลื่อนบิตของข้อมูลในอแกคิวมูลเลเตอร์

2.3.5 กลุ่มคำสั่งกระโดด

เป็นกลุ่มคำสั่งที่ใช้ในการเปลี่ยนแปลงตำแหน่งของ ค่าแอดเดรสการทำงานของซีพียู คำสั่งในกลุ่มนี้จึงมีประโยชน์อย่างมก ซึ่งมีหลายรูปแบบแต่จะกล่าวถึงแค่ กลุ่มคำสั่งการกระโดดแบบมีเงื่อนไข กับ แบบมีเงื่อนไข

1. กลุ่มคำสั่งการกะโดดแบบไม่มีเงื่อนไข

SJMP rel (Short Jump) กำหนดให้ซีพียูมาทำงานยังแอดเดรสที่กำหนดด้วยค่าสัมพัทธ์

AJMP addr11(Absoluute Jump) ให้ซีพียูมีทำงานยังแอดเดรสที่ระบุไว้ใน address11 มีขอบเขต 2 กิโลไบต์

LJMP addr16 (Long Jump) กำหนดให้ซีพียูมาทำงานตามที่ระบุไวว้ใน address16 มีขอบเขต 64 กิโลไบต์

JMP @A+DPTR (Jump Indirect) กำหนดให้ซีพียูกระโดดไปยังแอดเดรสของหน่วยความจำโปรแกรมในตำแหน่งที่ได้รับการกำหนดด้วยค่าของรีจีสเตอร์ A รวมกันกหับค่าใน DPTR

NOP (No Operation) เป็นคำสั่งที่ทำให้เกิดการเลื่อนแอดเดรสไปหนึ่งแอดเดรส

2. กลุ่มคำสั่งการกะโดดแบบมีเงื่อนไข

JB bit,rel (Jump if Bit set) กำหนดให้ตัวซีพียู กระโดดไปยังแอดเดรสปลายทางตามค่าความสัมพัสธ์(rel) เมื่อบิตของรีจีสเตอร์ที่ทำการตรวจสอบเกิดการเซต ใช้ได้กับรีจีสเตอร์ที่สามารถเข้าถึงได้ในระดับบิต

JNZ rel (JUMP if Accumulator Not Zero) กำหนดให้ซีพียูกระโดดไปยังค่าแอดเดรสปลายทาง ตามค่าสัมพัทธ์

2.3.6 ไทเมอร์/เคาน์เตอร์ (Timer/Counter)

ไทเมอร์/เคาน์เตอร์(Timer/Counter) เป็นส่วนประกอบที่สำคัญของ ไมโครคอนโทรลเลอร์ เนื่องจากการทำงานของไมโครคอนโทรลเลอร์ จะต้องมีการเก็บและตรวจสอบค่าของเวลาและจำนวนของสัญญาณนาฬิกาอยู่ตลอดเวลา เพื่อประโยชในการสร้างฐานเวลา สร้างสัญญาณพัลส์ เปรียบเทียบค่าเวลา หรือเปรียบเทียบค่าของการนับ รวมไปถึง การกำหนดอัตราเร็วในการสื่อสารของข้อมูลของพอร์ตอนุกรมด้วยรีจีสเตอร์ไทเมอร์/เคาน์เตอร์ สามารถกำหนดให้ทำงานเป็นตัวตั้งเวลาหรือเป็นไทเมอร์และตัวนับเคาน์เตอร์ได้อย่างอิสระต่อกัน การนับค่าของไทเมอร์/เคาเตอร์ในไมโครคอนโทรลเลอร์MCS-51 จะเป็นการนับขึ้นหรือเพิ่มค่าเพียงทางเดียว การทำงานเป็นตัวของไทเมอร์กำหนดให้ทำงานเป็นตัวตั้งเวลาหรือไทเมอร์ ค่าของรีจีสเตอร์จะเพิ่มขึ้นในทุกๆค่าของตัวแมชีนไซเกิล ส่วนการทำงานเป็นเคาน์เตอร์ เมื่อทำงานเป็นตัวนับเคาน์เตอร์ค่าของรีจีสเตอร์จะมีค่าที่เพิ่มขึ้นก็ต่อเมื่อมีการเปลี่ยนแปลงของระดับลอจิก " 1" เป็น "0" เกิดขึ้นที่ขาของอินพุตทาง

อาร์ดแวร์ของวงจรไทเมอร์/เคาน์เตอร์ ซึ่งในที่นี้จะมีรีจีสเตอร์ที่เกี่ยวข้องหรือสัมพันธ์กับการทำงานของ ตัวที่เรียกไทเมอร์/เคาน์เตอร์ 0 และ 1ดังนี้

1. รีจีสเตอร์ไทเมอร์

มีอยู่ด้วยกัน 4 ตัวคือ TL0 มีแอดเดรสอยู่ที่ 8AH,TH0 มีแอดเดรสอยู่ที่ 8CH, TL1และ มีแอดเดรสอยู่ที่ 8BH และ TH1 มีแอดเดรสอยู่ที่ 8DH รีจีสเตอร์ทั้ง 4 ตัวจะอยู่ในพื้นที่ของรีจีสเตอร์ฟังก์ชั่นพิเศษหรือ SFR รีจีสเตอร์แต่ละตัวมีขนาด 8 บิต แต่การใช้งานมักมีการใช้งานร่วมกันโดยจัดเป็นคู่คือ TL0 กับ TH0 รวมกันเป็นรีจีสเตอร์ Timer 0 ขนาด 16 บิต และ TL1 กับ TH1 รวมกันเป็นรีจีเตอร์ Timer1 ขนาด 16 บิต โดยใน TL0 และ TL1 จะเก็บข้อมูล 8 บิตล่าง ส่วน TH0 และ TH1 เก็บข้อมูล 8 บิตบนรีจีสเตอร์ไทเมอร์ทั้งคู่เมื่อนำมาใช้ร่วมกันจะสามารถเก็บค่าของการนับได้สูงสุด 65536 ค่า หรือ FFFH เมื่อนับถึงค่านี้แล้วก็จะวนไปเริ่มนับ 000H ใหม่และเมื่อเกิดการนับรอบใหม่ จะมีการเซตบิต TF0 หรือ TF1 ในรีจีสเตอร์ TCON ที่ใช้ควบคุมการทำงานของไทเมอร์เพื่อแจ้งให้ทราบว่าเกิดการนับสูงสุดแล้ว การเซตบิต TF0 หรือ TF1 ขึ้นอยู่กับว่าการเลือกใช้งานของรีจีสเตอร์ไทเมอร์ตัวใด รีจีสเตอร์ควบคุมการทำงานของไทเมอร์/เคาน์เตอร์หรือ ตัวของ TCON (Timer/Counter Register)และ เป็นรีจีสเตอร์ที่มีขนาด 8 บิต ซึ่งมีแอดเดรสอยู่ที่ 88 H ในพื้นที่ของตัวรีจีสเตอร์ SFR สามารถเข้าถึงข้อมูลได้ในระดับบิต

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 (Timer 1 overflow flag) เซตด้วยกระบวนการทางอาร์ดแวร์เมื่อค่าของรีจีสเตอร์ Timer 1 เกิดการนับเกินหรือโอเวอร์โฟลว์ การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางอาร์ดแวร์เช่นกันโดยบิตนี้จะเคลียร์เมื่อมีรการอินเตอร์รัปต์เกิดขึ้น

TR1 (Timer 1 run control bit) ใช้ในการเปิดปิดการทำงานของไทเมอร์ 1 ทำการเซตและเคลียร์ทางกระบวนการทางอาร์ดแวร์ ถ้าต้องการให้ไทเมอร์ 1 ทำงานต้องเซตบิตนี้ให้เป็น 1

TF0 (Timer 0 overflow flag) เซตด้วยกระบวนการทางอาร์ดแวร์เมื่อค่าของรีจีสเตอร์ Timer 0 เกิดการนับเกินหรือโอเวอร์โฟลว์ การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางอาร์ดแวร์เช่นกันโดยบิตนี้จะเคลียร์เมื่อมีรการอินเตอร์รัปต์เกิดขึ้น

IE1 (External Interrupt 1 edge flag) บิตนี้จะใช้ในกระบวนการทางอินเตอร์รัปต์ สามารถเซตได้ทั้งกระบวนการทางอาร์แวร์เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์ข้อมูลจากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 1 (INT1) ได้และจะทำการเคลียร์เมื่อมีการบริหารการอินเตอร์รัปต์

IT1 (Interrupt 1 type control bit) บิตนี้ใช้ในกระบวนการทางการอินเตอร์รัปต์โดยทำการเลือก

ลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอก ที่ต้องการให้ทำการตอบสนองสำหรับตัวขาอินพุต อินเตอร์รัปต์1 (INT1) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์

"0" เลือกกขอบขาลงของสัญญาณ (falling edge)

"1" เลือกระดับลอจิกต่ำ (low level triggered)

IE0 (External Interrupt 0 edge flag) บิตนี้ใช้ในกระบวนการทางอินเตอร์รัปต์ เวตด้วยกระบวนการทางอาร์ดแวร์ เมื่อสามารถตรวจจับขาขอบของสัญญาณอินเตอร์รัปต์ที่ได้จากภายนอกที่ขาอินเตอร์รัปต์ 0 (INT0) ได้ และจะทำการเคลียร์ค่าเมื่อมีการอินเตอร์รัปต์เกิดขึ้น

IT0 (Interrupt 0 type control bit) บิตนี้ใช้ในกระบวนการทางการอินเตอร์รัปต์โดยทำการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปต์0 (INT0) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์

"0" เลือกกขอบขาลงของสัญญาณ (falling edge)

"1" เลือกระดับลอจิกต่ำ (low level triggered)

2.4 วงจรตรรก

วงจรตรรก หมายถึง วงจรไฟฟ้าที่ประกอบไปด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือระบบรีเลย์ที่มี สัญญาณเพียง 2 ระดับหรือ 2 สภาวะเท่านั้น วงจรตรรกมี 2 ชนิด คือ แบบบวก กับ แบบลบและลอจิกแบบบวก จะใช้สัญญาณไฟฟ้าสูงแทนลอจิก 1 ไฟฟ้าระดับต่ำแทนสภาวะลอจิก 0 ส่วนด้านวงจรลอจิแบบลบจะใช้สัญญาณไฟฟ้าต่ำแทนสภาวะลอจิก 1 และใช้สัญญาณไฟฟ้าระดับสูงแทนสภาวะทางลอจิก 0 สภาวะทางลอจิก คือ สภาวะ 1 หรือสภาวะ 0 ใช้แทนการทำงานของอุปกรณ์ที่เปลี่ยนแปลง 2 สภาวะปฏิบัติการทางลอจิกประกอบด้วย AND OR NOT

2.5 ลักษณะรูปแบบของคำสั่งที่ใช้ในการโปรแกรม

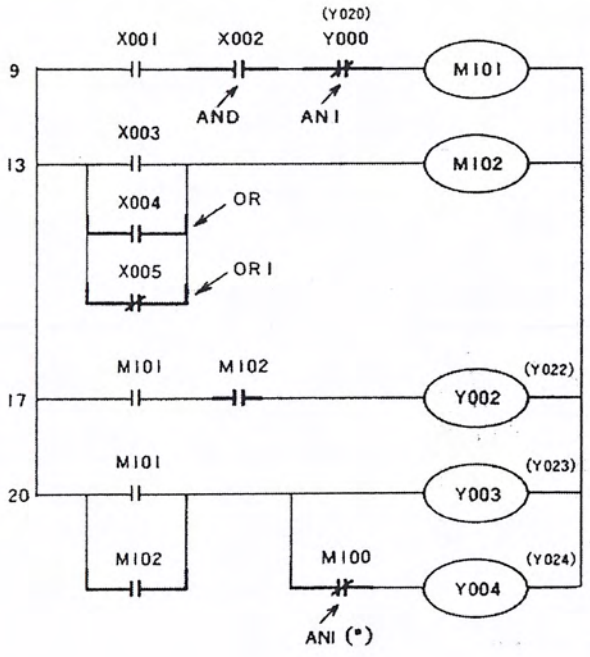
2.5.1 คำสั่ง LD , LDI , OUT

LD ต่อหน้าสัมผัส a กับสายไฟ

LDI ต่อหน้าสัมผัส b กับสายจ่ายไฟ

OUT คำสั่งขับคอยล์

ตัวอย่างการเขียนแลดเดอร์ไดอะแกรมด้วย AND ANI OR ORI



0	LD X 000
1	OUT Y 000
2	OUT T 0 SP K 100
5	LD T 0
6	OUT M 100
7	LDI M 100
8	OUT Y 001

ภาพที่2.10 ตัวอย่างการเขียนโปรแกรมโดยใช้คำสั่ง LD LDI OUT

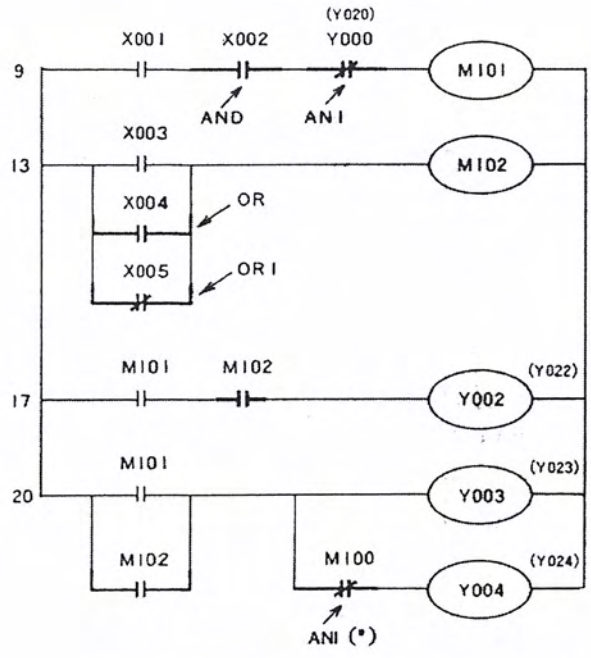
2.5.2 คำสั่ง AND , ANI , OR

AND คำสั่งต่ออนุกรมหน้าสัมผัส

ANI (And Inverse) คำสั่งต่ออนุกรมหน้าสัมผัส b

OR คำสั่งต่อขนานหน้าสัมผัส a

ตัวอย่างการเขียนแลดเดอร์ไดอะแกรมด้วย AND ANI OR ORI



9	LD	X001
10	AND	X002
11	ANI	Y000(20)
12	OUT	M101
13	LD	X003
14	OR	X004
15	ORI	X005
16	OUT	M102
17	LD	M101
18	AND	M102
19	OUT	Y002(22)
20	LD	M101
21	OR	M102
22	OUT	Y003(23)
23	ANI	M100
24	OUT	Y004(24)

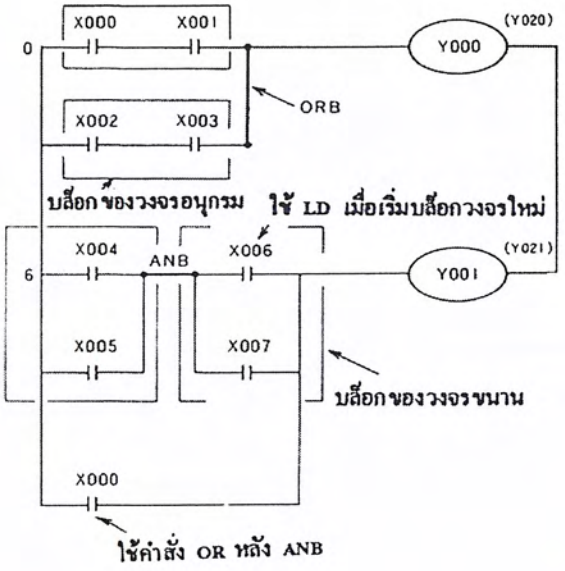
ภาพที่2.11 ตัวอย่างการเขียนโปรแกรมโดยใช้คำสั่ง AND ANDI OR ORI

2.5.3 คำสั่ง ORB และ ANB

ORB (OR Block) คำสั่งต่อขนานบล๊อกของวงจร

ANB (AND Block) คำสั่งต่ออนุกรมบล๊อกของวงจร

ตัวอย่างการเขียนแลดเดอร์ไดอะแกรม ORB ANB



0	LD	X 000	
1	AND	X 001	
2	LD	X 002	
3	AND	X 003	
4	ORB		
5	OUT	Y 000(20)	
6	LD	X 004	
7	OR	X 005	
8	LD	X 006	
9	OR	X 007	
10	ANB		
11	OR	X 000	
12	OUT	Y 001(21)	

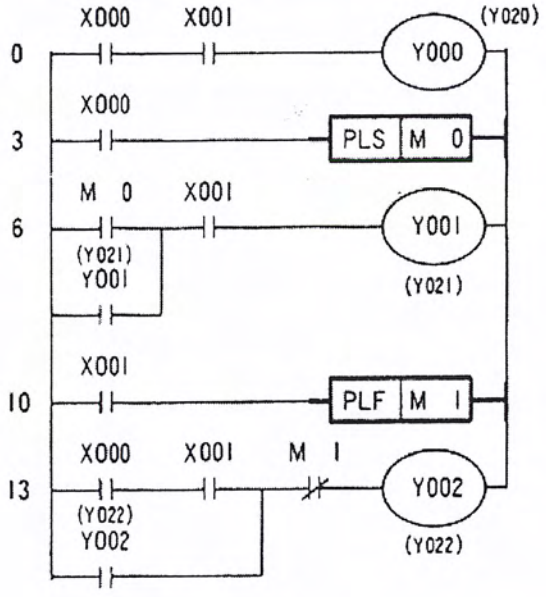
ภาพที่2.12 ตัวอย่างการเขียนโปรแกรมโดยใช้คำสั่ง ORB ANB

2.5.4 คำสั่ง PLS และ PLF

PLS (Pulse) คำสั่งกำเนิดพัลส์ที่ขอบของสัญญาณ

PLF (Pulse Fall) คำสั่งกำเนิดพัลส์ที่ขอบขาลงของสัญญาณ

ตัวอย่างการเขียนแลดเดอร์ไดอะแกรมด้วย PLS PLF



0	LD X 000
1	AND X 001
2	OUT Y 000 (20)
3	LD X 000
4	PLS M 0
6	LD M 0
7	OR Y 001 (21)
8	AND X 001
9	OUT Y 001 (21)
10	LD X 001
11	PLF M 1
13	LD X 000
14	AND X 001
15	OR Y 002 (22)
16	ANI M 1
17	OUT Y 002 (22)

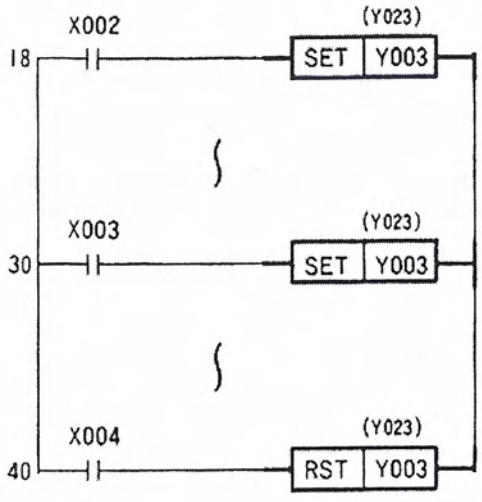
ภาพที่2.13 ตัวอย่างการเขียนโปรแกรมโดยใช้คำสั่ง PLS PLF

2.5.5 คำสั่ง SET และ RESET

SET (Set) คำสั่งให้ทำงานค้าง

RST (Reset) คำสั่งให้หยุดทำงานค้าง

ตัวอย่างการเขียนแลดเดอร์ไดอะแกรมด้วย SET RESET



18	LD X 002
19	SET Y003(23)
	⌇
30	LD X 003
31	SET Y003(23)
	⌇
40	LD X 004
41	RST Y003(23)
43	END

ภาพที่2.14 ตัวอย่างการเขียนโปรแกรมด้วยคำสั่ง SET RESET

การทำงาน เมื่อ X 002 ON จะทำให้ Y 003 ทำงาน และทำงานค้าง แม้ X 002 จะ OFF แล้ว ก็ตาม เมื่อ X 004 ON จะทำให้ Y 003 ทำงาน และทำงานค้าง แม้ X 004 จะ OFF ก็ตาม X 003 ก็จะทำงานเหมือนหน้าสัมผัสทั้งสองด้วย

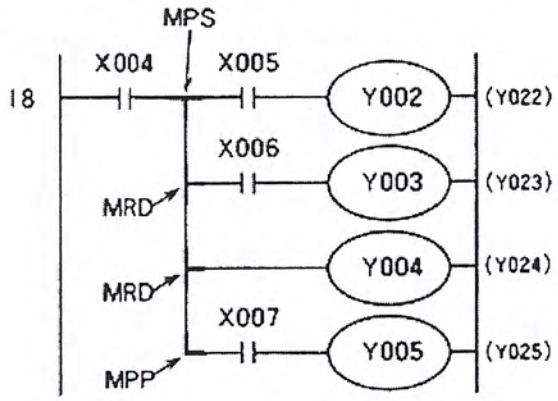
2.5.6 คำสั่ง MPS , MRD และ MPP

MPS (Memory Push) บันทึกค่าในหน่วยความจำชั่วคราว

MRD (Memory Read) อ่านค่าจากหน่วยความจำ

MPP อ่านค่าจากหน่วยความจำและรีเซท

ตัวอย่างการเขียนแลดเดอร์ไดอะแกรด้วย MPS MRD MPP



18	LD X 004
19	MPS
20	AND X 005
21	OUT Y 002 (22)
22	MRD
23	AND X 006
24	OUT Y 003 (23)
25	MRD
26	OUT Y 004 (24)
27	MPP
28	AND X 007
29	OUT Y 005 (25)
30	END

ภาพที่2.15 ตัวอย่างการเขียนโปรแกรมโดยใช้คำสั่ง MPS MRD MPP

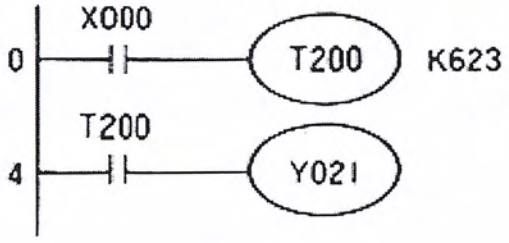
2.5.7 คำสั่ง TIMER

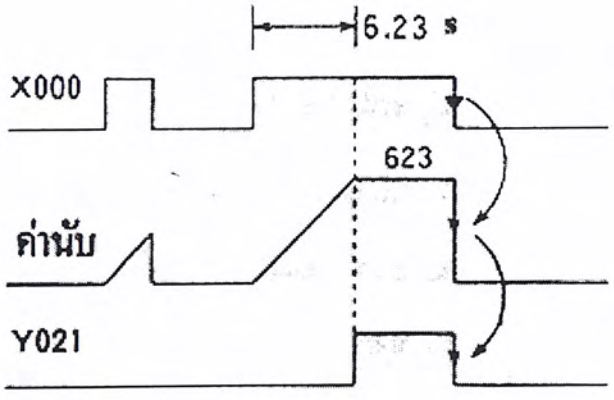
TIMER คำสั่งในการควบคุมเวลา Timer เป็นอุปกรณ์ที่นับพัลส์สัญญาณนาฬิกา 1ms.และ 10 ms หรือ 100 ms. เมื่อค่านับเท่ากับค่าที่ตั้งไว้ก็จะให้หน้าสัมผัสเอาท์พุต ทำงานของการตั้งค่าของไทเมอร์จะใช้วิธีตั้งค่า K ในการเขียนโปรแกรม หรือจะตั้งค่าจากรีจีสเตอร์ข้อมูลก็ได้

1. Timer ธรรมดา

ไทเมอร์ 100 ms. T0 – T199 (200 ตัว) ตั้งค่า 0.1 – 3276.7 วินาที

ไทเมอร์ 10 ms. T200 – T245 (46 ตัว) ตั้งค่า 0.01 – 327.67 วินาที





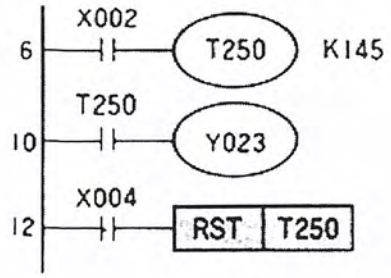
ภาพที่2.16 ตัวอย่างการทำงานของไทเมอร์ธรรมดา

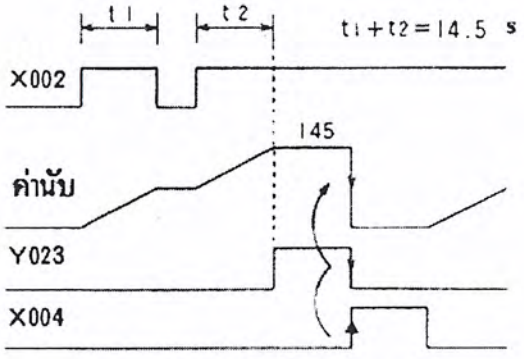
เมื่อหน้าสัมผัส X000 ON คอยของ T200 จะถูกขับ ไทเมอร์จะเริ่มนับพัลส์ 10 ms. เมื่อค่านับเท่ากับที่ตั้ง 623 จะให้เอาท์พุต นั้นคือหน้าสัมผัส เอาท์พุตจะ ON หลังจากที่คอยล์ถูกขับให้ทำงาน 6023 วินาที เมื่อ X000 OFF หรือไฟดับ ค่านับในไทเมอร์จะถูกรีเซท

2. Timer นับสะสม

ไทเมอร์นับสะสม 1ms. T246 – T248 (4 ตัว) ค่าตั้ง 0.001 – 32.767 วินาที

ไทเมอร์นับสะสม 100 ms. T250 – T255 (6 ตัว) ค่าตั้ง 0.1 – 3276.7 วินาที





ภาพที่2.17 ตัวอย่างการทำงานของไทเมอร์นับสะสม

X002 ON คอยล์ของ T205 จะถูกขับ ไทเมอร์จะเริ่มนับพัลส์ 100 ms. เมื่อค่านับเท่ากับค่าที่ตั้ง 145 จะให้เอาท์พุต ในระหว่างที่นับ ถ้า X002 OFF หรือไฟดับ ไทเมอร์จะรักษาค่านับไว้ เมื่อไฟมาจะนับต่อไปจนกว่าถึง 145 หรือรวมเวลาที่นับเป็น 1405 วินาที ถ้าหน้าสัมผัส X004 ON ไทเมอร์T250จะถูกรีเซทเป็น "0"

2.5.8 คำสั่ง COUNTER

COUNTER เป็นอุปกรณ์ที่สำหรับใช้ในการนับสัญญาณ ค่านับจะเพิ่มค่าหรือ เพื่อลดค่าตัวเลขการนับแล้วแต่ชนิดของ เคาน์เตอร์ซึ่งการทำงานของ COUNTERแบ่งออกได้เป็น

1. 16 บิตเคาน์เตอร์

เคาน์เตอร์ธรรมดา CO – C99 (100 ตัว)

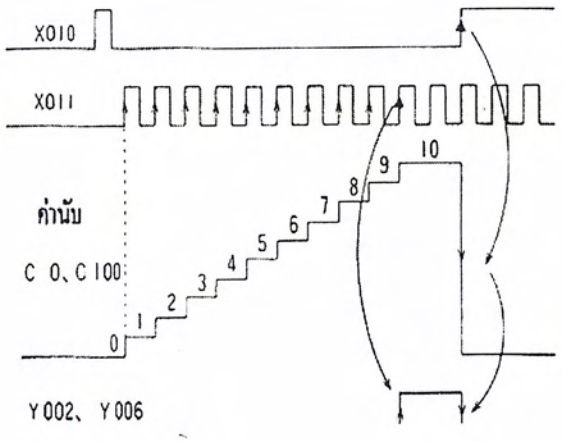
เคาน์เตอร์สำรองไฟดับ C100 – C199 (100 ตัว) 16 บิตเคาเตอร์ชนิดนับขึ้น (up count)ค่านับเป็นเลขฐานสองค่าตั้ง มีตั้งแต่ K1 – K32765 ค่าตั้ง K0 เท่ากับ K1 คือจะให้หน้าสัมผัสเอาท์พุตทันที เมื่อนับครั้งแรกค่าตั้งเป็นลบไม่ได้ เคาเตอร์จะไม่นับและเกิดการผิดพลาด ค่า CO และ C100 จะนับสัญญาณจากหน้าสัมผัส X011 ค่านับจะเพิ่มขึ้นเรื่อยๆ เมื่อนับได้ 10 เคาน์เตอร์ จะให้สัญญาณ

ที่ได้ออก หลังจากนั้นแม้จะมีสัญญาณเข้ามาอีก ค่านับในเคาเตอร์จะไม่เปลี่ยนแปลง เมื่อ X010 ON คำสั่ง RST จะทำรีเซตเคาน์เตอร์ทำให้ค่านับเป็น "0" และหน้าสัมผัสเอาท์พุตหยุดทำงาน

2. 32 บิต เคาน์เตอร์

เคาน์เตอร์ธรรมดา C200 – C219 (20 ตัว)

เคาเตอร์สำรองไฟ C220 – C234 (15 ตัว) 32 บิตเคาน์เตอร์ สามารถ นับขึ้นและลงได้ ค่านับเป็นฐานสอง ค่าตั้งอยู่ระหว่าง -2147483648 ถึง +2147483647



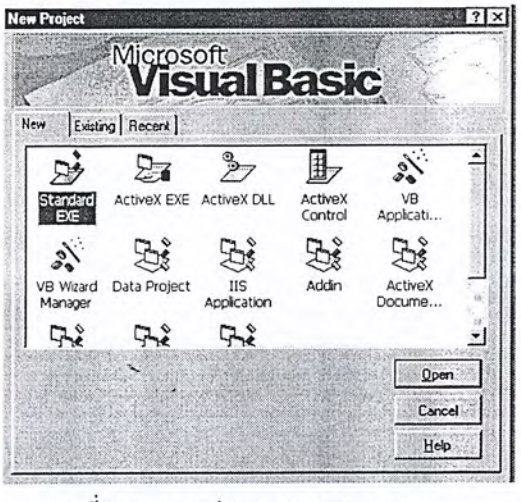
ภาพที่2.18 ตัวอย่างการทำงานของเคาน์เตอร์

บทที่ 3
การเขียนโปรแกรมด้วย Visual Basic

3.1 การเขียนโปรแกรมด้วย Visual Basic ขั้นพื้นฐาน

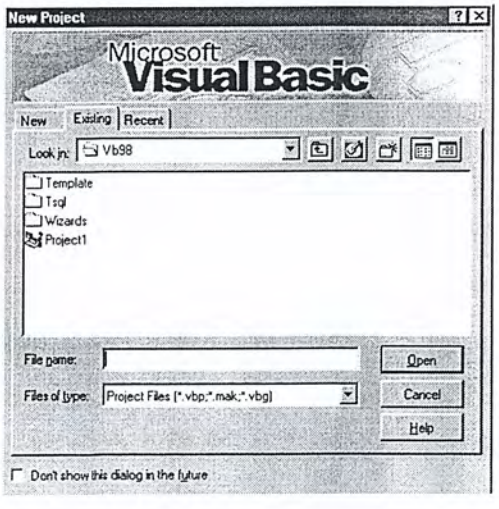
สิ่งแรกที่จะพบเมื่อเข้าสู่โปรแกรม Microsoft Visual Basic 6.0 ได้แก่ จอภาพ ที่ใช้สำหรับเปิด Project จะเป็นชื่อที่ใช้เรียกแทนระบบงานที่พัฒนาขึ้นด้วยการเขียนโปรแกรม Visual Basic ประกอบไปด้วย 3 Tab ดังนี้

1. Tab "New" เป็นจอภาพที่ประกอบไปด้วย Icon ต่างๆที่ใช้สำหรับเรียกใช้โปรเจคใหม่ขึ้นมาเพื่อใช้งาน



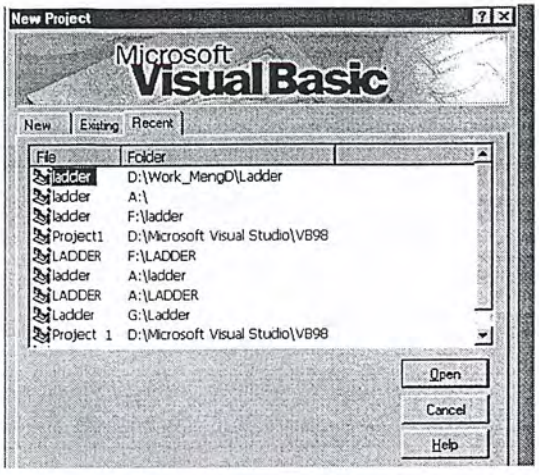
ภาพที่ 3.1 แสดงส่วนของ Tab"New"

2. Tab "Existing" เป็นจอภาพสำหรับเรียกใช้ Project เดิมที่พัฒนาขึ้นแล้วมาเก็บไว้ใน Directory ต่างๆขึ้นมาใช้งาน



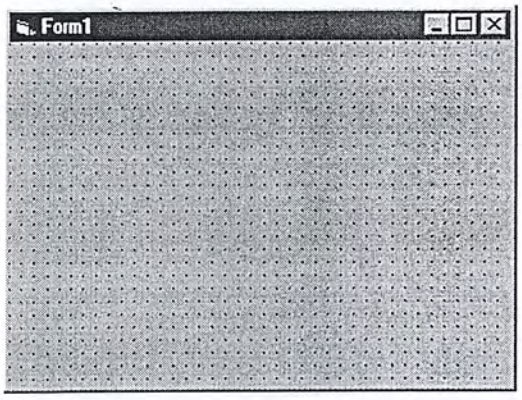
ภาพที่ 3.2 แสดงส่วนของ Tab "Existing"

3. Tab "Recent" เป็นจอภาพที่แสดงประวัติของ Project ต่างๆที่เคยถูกเรียกขึ้นมาพัฒนาส่วนประกอบของจอภาพ Visual Basic 6.0 มีดังนี้



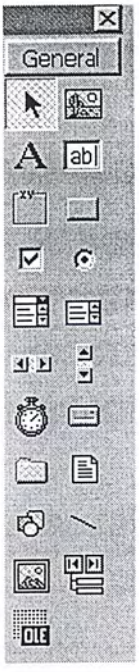
ภาพที่ 3.3 แสดง ส่วนของ Tab "Recent"

-Form เป็นส่วนที่ใช้สำหรับสร้างจอภาพของโปรแกรม โดยจะทำหน้าที่เป็นพื้น (Background) ของจอภาพ ทุกครั้งของการเปิด Project ใหม่ จะได้ฟอร์มเปล่า



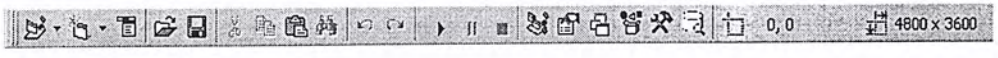
ภาพที่ 3.4 แสดง Form

-Toolbox เป็นแถบเครื่องมือที่ประกอบไปด้วย Icon ต่างๆ ซึ่งเรียกว่า "Control" ซึ่งจะใช้ร่วมกับ Form เพื่อสร้างจอภาพ Project แต่ละ Control จะใช้เป็นเครื่องมือที่ใช้สำหรับสร้างส่วนที่ใช้ติดต่อกับผู้ใช้ หรือที่เรียกว่า "User Interface" เช่นข้อความต่างๆ ช่องว่างสำหรับรับข้อมูลจากคีย์บอร์ดปุ่มต่างๆ ฯลฯ เป็นต้น และจะถูกนำไปใช้งานด้วย โดยการนำ Control ที่จะต้องการไปวางลงบน Form Control แต่ละจะมีชื่อและหน้าที่ที่แตกต่างกันไป เมื่อต้องการดูชื่อของ Control ใดก็เพียงแต่เลื่อนเม้าท์ไปชี้ที่ Control นั้นชื่อของ Control จะปรากฏขึ้นให้เห็น



ภาพที่ 3.5 แสดง Control บน Toolbox

-Toolbar เป็นแถบเครื่องมือที่ประกอบด้วย Icon ต่างๆดังรูป ซึ่ง Toolbar จะทำหน้าที่เป็นผู้ช่วยในการพัฒนาโปรแกรม ซึ่งเมื่อเลื่อนเม้าท์ไปชี้ยัง Icon ใด ก็จะปรากฏชื่ออยู่ใต้ Icon นั้นแต่ละ Icon จะมีหน้าที่ต่างกันไป



ภาพที่ 3.6 แสดง Toolbar

-Project Explorer Window เป็นส่วนสำหรับเรียก Form ต่างๆขึ้นมาแก้ไขในกรณีที่ Project ประกอบด้วย Form มากกว่า 1 Form

-Properties Window เป็นจอภาพที่ใช้สำหรับกำหนดคุณสมบัติ (Property) ให้กับ Form และ Object ต่างๆ ที่ปรากฏอยู่บน Form

- Form Layout Window ใช้สำหรับดูตำแหน่งของ Form บนจอภาพ ทำให้จัดตำแหน่งของ Form ได้สะดวกขึ้น

-Project โดยทั่วไปในงานหนึ่งๆมักประกอบไปด้วยหลายๆ จอภาพ ดังนั้นการพัฒนาโปรแกรม จึงนิยมแยกแต่ละจอภาพออกเป็นโปรแกรมเพื่อสะดวกต่อการแก้ไขและตามหลักการ

เขียนโปรแกรม แล้วจึงนำแต่ละโปรแกรมย่อยมาประกอบกันขึ้นเป็นระบบโดยการ Compile ไฟล์เหล่านั้นร่วมกันเป็น Executed Program (ไฟล์นามสกุล EXE) เพื่อนำไปใช้งาน

3.2 การ Run และเลิกงาน Project

ในการ Run Project ที่พัฒนาขึ้นด้วย Visual Basic สามารถ Run ทั้งโดยใช้ Interpreter และการใช้ Compiler กล่าวคือเราสามารถทดลอง Run สิ่งต่างๆ ที่เราจัดทำขึ้นไปพร้อม ๆ กับการแก้ไขโปรแกรมจนกระทั่งเสร็จสมบูรณ์ แล้วจึง Compile โปรแกรมให้อยู่ในรูปของ Executed Program เพื่อนำไปใช้ในงานได้เช่นเดียวกัน แต่ในเบื้องต้นนี้ จะขอกล่าวถึงส่วนที่เป็น Interpreter เป็นลำดับ

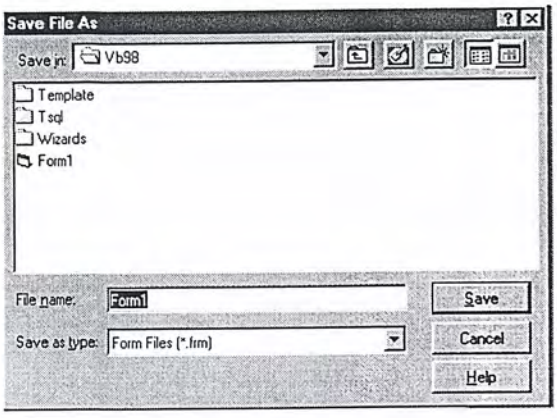
ตารางที่ 3.1 การ Run และเลิกงาน Project

ในการ Run Project ที่เราจัดทำขึ้นนั้นทำได้ 3 วิธี	สำหรับวิธีเลิกงาน Project ทำได้ 2 วิธี
วิธีที่ 1 Run โดยกด F5 วิธีที่ 2 คลิกที่ Icon "Run"ใน Toolbar วิธีที่ 3 เลือกจากเมนู Run และ Start ตามลำดับ	วิธีที่ 1 คลิกที่ Icon"End"ในToolbar วิธีที่ 2 เลือกจากเมนู Run และ End ตามลำดับ

3.3 การบันทึก Form และ Project

ในการบันทึก Project จะต้องบันทึกทั้งส่วนของ Form และ Project โดย Form จะถูกบันทึกลงในไฟล์นามสกุล FRM ในขณะที่ Project จะถูกบันทึกลงในไฟล์นามสกุล VBPในการบันทึก Form ให้ทำดังนี้

1. คลิกที่ Icon "Save" ใน Toolbar จะปรากฏจอภาพดังรูป
2. ใส่ชื่อ File ที่จะใช้เก็บ Form ซึ่ง Visual Basic จะใช้ชื่อเริ่มต้นเป็นชื่อเดียวกับ Form เสมอดังนั้นจึงปรากฏชื่อ Form 1 มาให้ผู้อ่านสามารถเปลี่ยนแปลงชื่อได้ตามต้องการ จากนั้นให้คลิกที่ปุ่ม Save
3. จะปรากฏจอภาพสำหรับบันทึก Project ซึ่งก็เช่นเดียวกับ Form เป็นชื่อ Project 1 ให้เปลี่ยนชื่อตามต้องการแล้วคลิกที่ปุ่ม Save



ภาพที่ 3.7 แสดงการบันทึก Form และ Project

3.4 ขั้นตอนในการพัฒนาโปรแกรมของ Visual Basic ประกอบไปด้วยขั้นตอนหลัก 2 ขั้นตอน

ขั้นตอนที่ 1 การสร้างจอภาพของโปรแกรม

ในขั้นตอนนี้จะนำ Form มาออกแบบเพื่อใช้ในการติดต่อกับผู้ใช้ หรือที่เรียกว่า การออกแบบ " User Interface" ในการพัฒนาโปรแกรมแบบเดิม แต่ในส่วนนี้ซึ่ง Visual Basic สามารถทำได้อย่างง่ายดายจากการพัฒนาโปรแกรมแล้ว เพียงแต่นำเอา Control ต่างๆ ในToolbox ที่ต้องการใช้งานมาวาดไว้บน Form

ขั้นตอนที่ 2 การเขียนโปรแกรม

เมื่อวาง Control ต่างๆลงบน Form เป็นที่เรียบร้อยแล้ว (Control ต่างๆเมื่อถูกนำมาวางไว้บน Form จะเรียกว่า "Object") ขั้นตอนต่อมาได้แก่ การเขียนโปรแกรมเพื่อกำหนดการทำงานให้แต่ละ Object ภายในเหตุการณ์ต่างๆ (Even) ที่จะเกิดขึ้นกับจอภาพนั้นๆ

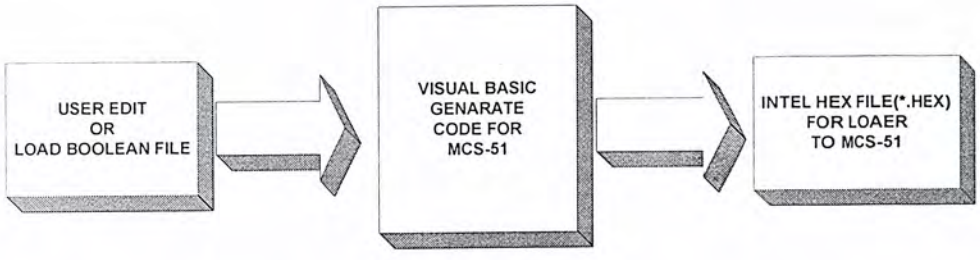
-Event-Driven Program กับ Visual Basic

แนวความคิดในกรเขียนโปรแกรมแบบ Event-Driven จะเปลี่ยนมาสนใจกับเหตุการณ์ (Event) ที่จะเกิดขึ้นในโปรแกรมมากกว่าการกำหนดขั้นตอนการทำงานของโปรแกรมในแบบเดิม เช่น ถ้ามีการ "เลื่อนเม้าท์" เกิดขึ้น จะให้โปรแกรมทำอย่างไร หรือมีการกดปุ่มที่ 1 ขึ้นจะให้โปรแกรมทำอย่างไรเป็นต้น อย่างไรก็ตาม ก็ยังต้องอาศัยแนวความคิดในการที่จะเขียนโปรแกรมแบบเดิมอยู่บ้าง เนื่องจากการกำหนดการทำงานให้กับแต่ละ Event ยังต้องกำหนดอย่างเป็นขั้นตอนอยู่ดี การเขียนโปรแกรมแบบ Event-Driven ใน Visual Basic จะเป็นการเขียนโปรแกรมให้กับ Object ต่างๆ ที่ปรากฏอยู่บน Form โดยจะพิจารณาว่าแต่ละ Object จะมี Event อะไรเกิดขึ้นบ้างแล้วจึงเลือกเขียนโปรแกรมเฉพาะ Event นั้น

บทที่ 4

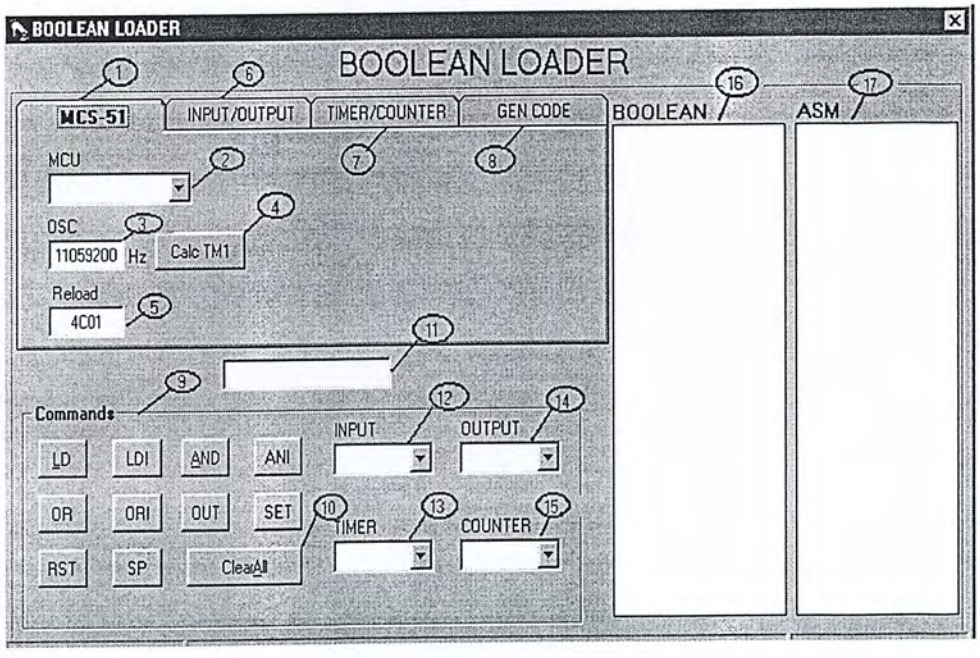
โครงสร้างโปรแกรม Boolean Software Support

SOFTWARE



ข้อมูลที่ได้รับเข้ามา เป็นBoolean File ใช้ Visual Basic เป็นตัวGenerate code ให้เป็น File.Asm และ Hex.File(.Hex) แล้วนำ(.Hex)ไปโหลดลงไมโครคอนโทรลเลอร์ MCS-51

4.1 โปรแกรมส่วน Boolean



ภาพที่ 4.1 ภาพส่วนของ Boolean

หมายเลข 1 – ปุ่ม MCS-51 เพื่อใช้ในการกำหนดเกี่ยวกับรายละเอียดต่างๆที่เกี่ยวข้องกับตัวไมโครคอนโทรลเลอร์ ซึ่งรายละเอียดต่างๆเหล่านี้จะเป็นส่วนสำคัญอย่างมาก เช่น MCU หรือ Microcontroller Unit, Osc, Cale TM1, Reload

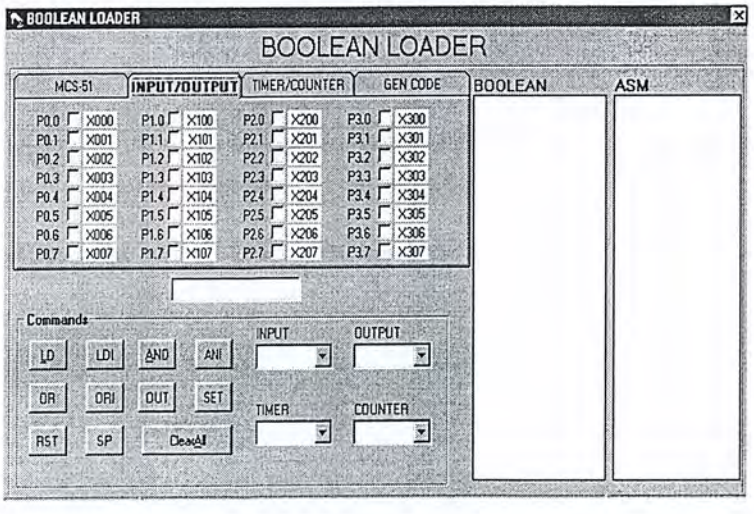
หมายเลข 2 – ปุ่ม Microcontroller Unit เพื่อใช้เป็นตัวกำหนดเมื่อต้องการที่เลือกเบอร์ของไมโครคอนโทรลเลอร์ว่าจะเลือกใช้เบอร์ใด ซึ่งแต่ละเบอร์จะมีพอร์ตอินพุตและเอาท์พุตจำนวนขาแตกต่างกันออกไป ขึ้นอยู่กับเบอร์ของ MCS-51 ที่เลือกใช้ได้แก่เบอร์ 8051, 8052, 89S53, 89S8252, 89C1051, 89C2051, 89C4051

หมายเลข 3 – เป็นความถี่ของคริสตอลที่ใช้ในการกำหนดความถี่ของสัญญาณนาฬิกา ซึ่งค่าความถี่ที่ใช้ทั่วไป ปกติใช้ Crytal ความถี่ 11.059200MHz หรือบางที่ใช้ 12 MHz เพราะว่าเป็นค่าที่สามารถคำนวณค่าบอร์ดเลทได้ลงตัวจะง่ายต่อการติดต่อกับภายนอก

หมายเลข 4 – ปุ่ม CaleTM1เพื่อใช้เป็นตัวในการคำนวณเวลาหรือเพื่อใช้เป็นฐานเวลาของตัวไมโครคอนโทรลเลอร์ MCS-51 เพื่อนำไปเป็นตัวอ้างอิงใน PLC หากใช้ฐานเวลา 12 MHz และหนึ่งแมชชินไซเคิลนั้นต้องใช้ สัญญาณนาฬิกา 12 ลูกจะทำให้หนึ่งรอบคำสั่งต้องใช้เวลาทั้งหมดเป็น 1 ไมโครวินาทีหรือความถี่ในการทำงาน 1 MHz

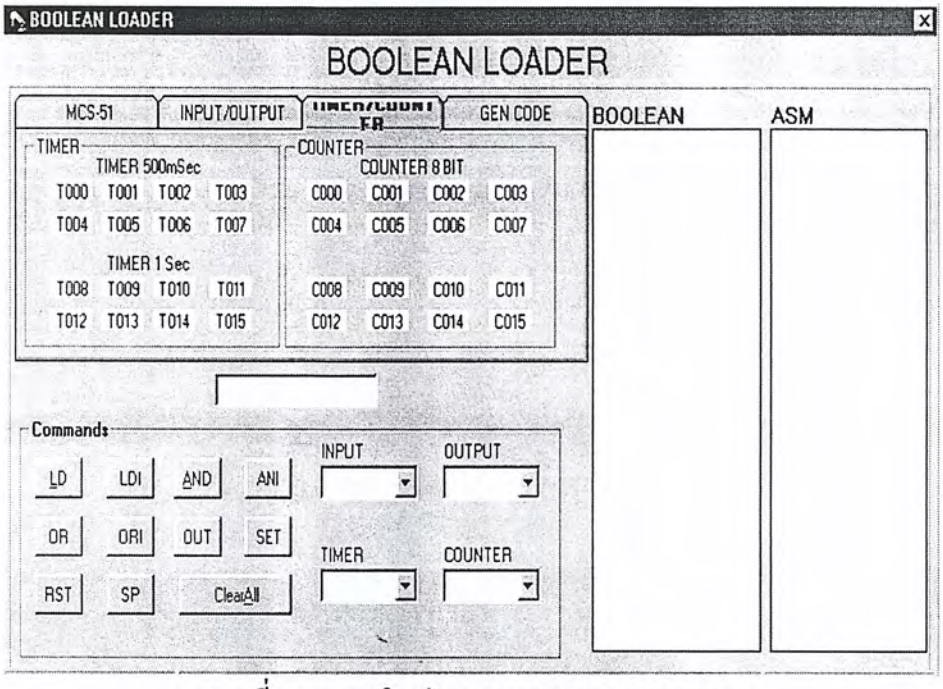
หมายเลข 5 – เป็นส่วนที่แสดงค่าที่ได้จากการคำนวณของ Cale TM1 ซึ่งค่าที่ได้จะเป็นค่าเลขฐานสิบหก

หมายเลข 6 – ปุ่ม INPUT/OUTPUT ใช้แสดงพอร์ตต่างๆของไมโครคอนโทรลเลอร์เพื่อต้องการเลือกและกำหนดพอร์ตเหล่านั้นใช้เป็นอินพุต/เอาท์พุตโดยจำนวนพอร์ตต่างๆที่สามารถเลือกได้นั้นจะต้องขึ้นอยู่กับเบอร์ไมโครคอนโทรลเลอร์ ซึ่งแต่ละเบอร์อาจมีจำนวนพอร์ตที่สามารถใช้งานได้มีไม่เท่ากัน



ภาพที่ 4.2 แสดงพอร์ต INPUT/OUTPUT

หมายเลข 7- ปุ่ม TIME/COUNTER แสดงเบอร์ของ Timer/Counter ว่าจะเลือกใช้ Timer Counter ชนิดไหนซึ่งในส่วนของ Timer มีชนิด 500 mSec กับ 1 Sec ในส่วนของ Counterมีขนาด 8 บิท



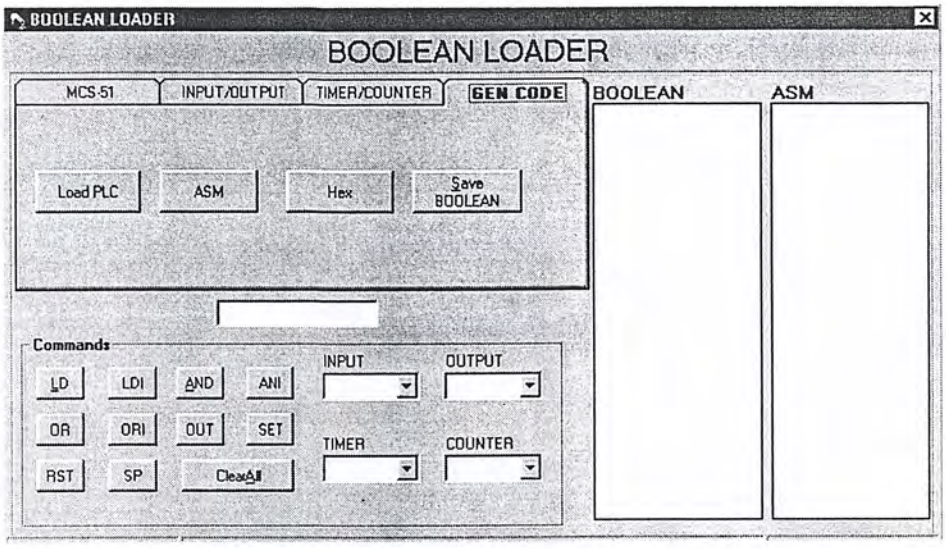
ภาพที่ 4.3 แสดงในส่วนของ TIMER/COUNTER

หมายเลข 8 – ปุ่ม GEN COD เพื่อใช้ในการ Generat code ของ Booleanหรือแปลงไฟล์ของบูลลีน ซึ่งมีอยู่ด้วยกันคือ

Load Boolean เป็นปุ่มที่ใช้สำหรับโหลดค่าหรือคำสั่ง Boolean ที่เขียนคำสั่งบูลลีนหรือ Save ไว้จากภายนอกมาใช้

ASM เป็นปุ่มที่ใช้ในการแสดงค่า Boolean ที่เขียนขึ้นมาหรือโหลดมาจากภายนอกแปลงให้เป็นภาษาแอสเซมบลี

Hex เป็นปุ่มที่ใช้สำหรับแปลงค่าจากแอสเซมบลี (.Asm) ให้เป็น Hex File(.Asm) เพื่อนำไปโหลดลงไมโครคอนโทรลเลอร์



ภาพที่ 4.4 แสดงในส่วนของ GEN CODE

หมายเลข 9 – แสดงคำสั่งพื้นฐานแบบลอจิก เช่น LD LDI AND ANI OR ORI OUT และยังมีคำสั่งพิเศษอีกดังนี้

ปุ่ม SET เป็นปุ่มที่ใช้สำหรับ Set ค่าให้เอาท์พุตทำงานค้างตลอด ถึงแม้ว่าจะ Off Input แล้วก็ตาม

ปุ่ม RST เป็นปุ่มที่หยุดการทำงานค้างของเอาท์พุต

ปุ่ม SP เป็นปุ่มที่ใช้ในการกำหนดค่าตัวเลขจำนวนเต็มให้กับตัว Timer เพื่อใช้ในการคำนวณการตั้งเวลาของตัว Timer

หมายเลข 10 – เป็นปุ่ม ClearAll มีไว้สำหรับลบสถานะการแสดงผลเดิมของค่าข้อมูลแบบลอจิกทั้งหมดในหน้าจอนั้นๆ

หมายเลข 11 – ใช้ในส่วนการแสดงผล ของคำสั่งก่อนที่จะทำการ Gen Code ตัวอย่างคำสั่งพื้นฐานแบบลอจิก เช่น LD LDI AND ANI OR ORI OUT SET RST SP TIMER COUNT

หมายเลข 12 – เป็นปุ่ม INPUT มีไว้สำหรับเลือกขาอินพุต โดยในที่นี้เราเซตไว้ให้พอร์ตทุกพอร์ตเป็นอินพุตซึ่งมี PO-P3 โดยจะขึ้นอยู่กับเบอร์ไมโครคอนโทรลเลอร์ที่เลือกใช้

หมายเลข 13 – เป็นปุ่ม TIMER เราจะกำหนดเองว่าจะเลือกใช้ Timer ชนิดไหนและเบอร์อะไร โดยมีทั้งหมด 16 ตัวตั้งแต่เบอร์ T000-T015 ซึ่งตัว Timer 500 mSec ใช้กับTimer เบอร์ T000-T007 และ Timer 1 Sec ใช้กับTimer เบอร์ T008-T015 ซึ่งเบอร์ของTimer สามารถเปลี่ยนแปลงได้ตามความเหมาะสม

หมายเลข 14 – ปุ่ม OUTPUT ใช้สำหรับเลือกเอาท์พุตว่าต้องการใช้พอร์ตไหนเป็นเอาท์พุต โดยเราจะต้องเป็นคนกำหนดให้มัน ซึ่งถ้าเราเลือกพอร์ตไหนเป็นเอาท์พุตแล้วในส่วนของอินพุตพอร์ตนั้นไม่สามารถเลือกใช้ได้

หมายเลข 15 – ปุ่ม COUNTER ขนาด 8 บิท มีอยู่ด้วยกันให้เลือกใช้ทั้งหมด 16 ตัวตั้งแต่เบอร์ C000-C015 สามารถเปลี่ยนเบอร์ได้ตามความต้องการ

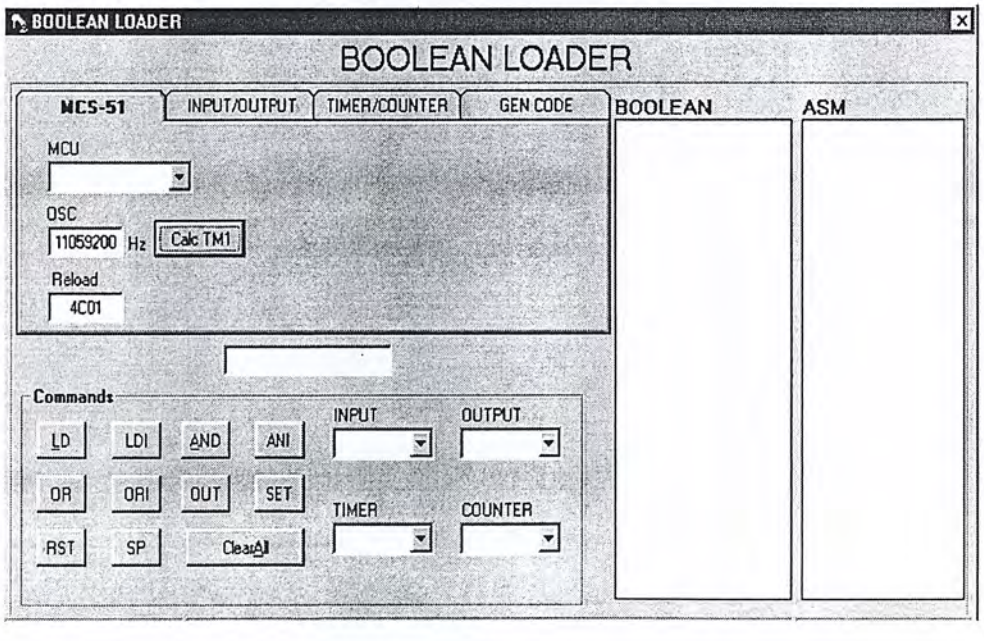
หมายเลข 16 – เป็นหน้าจอแสดงผลค่าสภาวะทางลอจิกและคำสั่งพื้นฐานรวมทั้งอินพุต/เอาท์พุตและค่าที่ได้จากการแปลงเป็น Hex File

หมายเลข 17 – เป็นส่วนที่แสดงผลของค่าแอสเซมบลีที่ได้จากการแปลงของบูลลีน

4.2 การใช้โปรแกรม

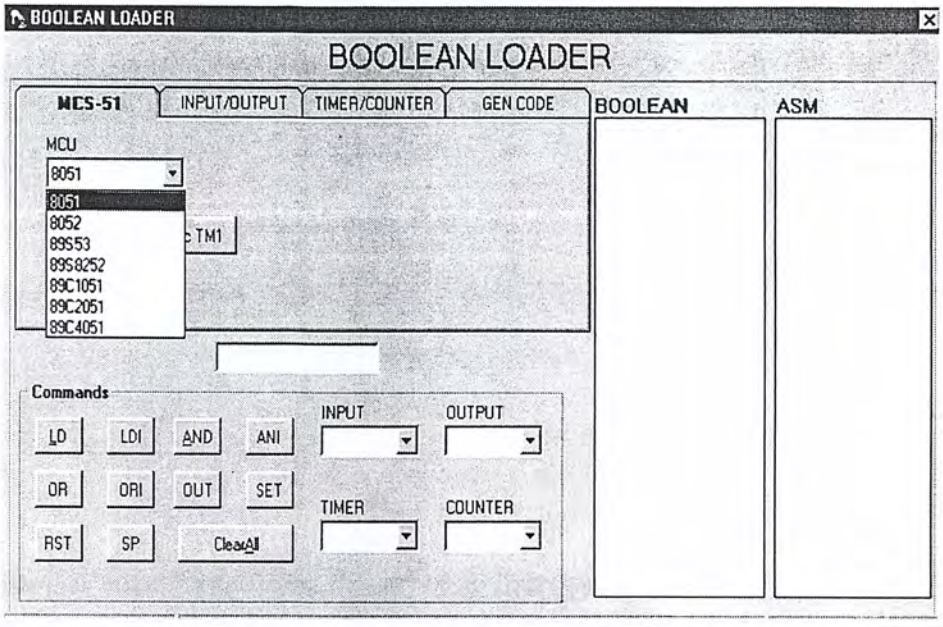
ขั้นตอนการใช้โปรแกรมสามารถอธิบายได้ตามขั้นตอนข้างล่างนี้

1. ทำการกดปุ่ม MCS-51 เพื่อที่จะต้องการกำหนดรายละเอียดเกี่ยวกับตัวไมโครคอนโทรลเลอร์ซึ่งจะแสดงยังภาพข้างล่าง



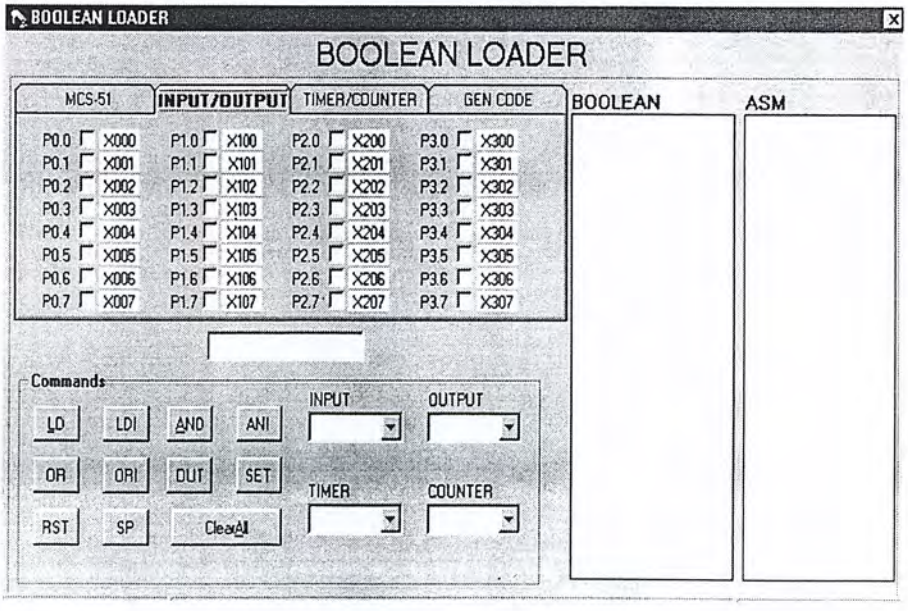
ภาพที่ 4.5 แสดงรายในส่วนของ MCS-51

2. กดเลือกเบอร์ไมโครคอนโทรลเลอร์ที่จะใช้โดยกดที่ปุ่ม MCU (Microcontroller Unit)แสดงดังรูปข้างล่าง



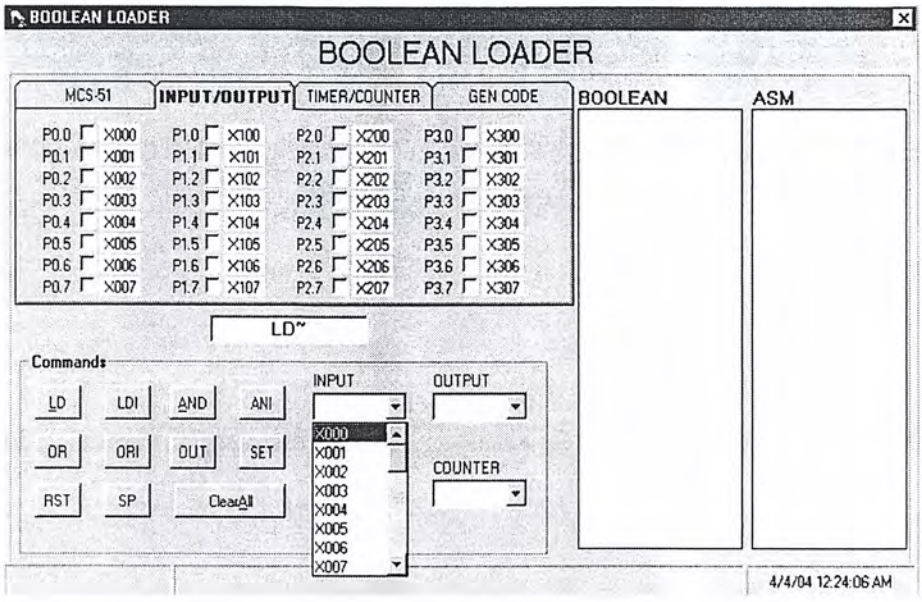
ภาพที่ 4.6 แสดงการกดเลือกเบอร์ไมโครคอนโทรลเลอร์

3. ทำการกำหนดอินพุตและเอาท์พุตโดยการกดปุ่ม INPUT/OUTPUT ซึ่งจะแสดงพอร์ตที่สามารถเลือกใช้เป็นอินพุตและเอาท์พุตได้



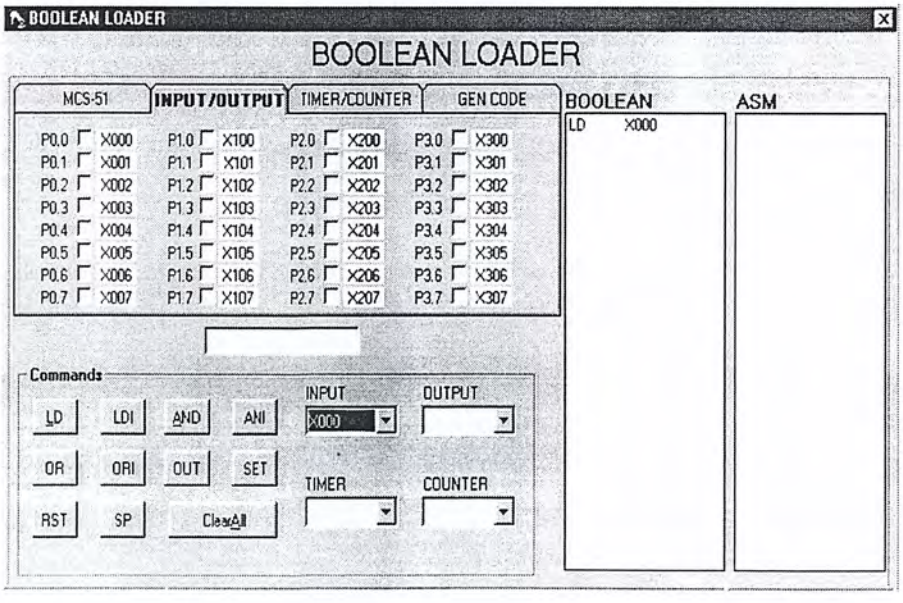
ภาพที่ 4.7 แสดงพอร์ตการใช้งานของ INPUT/OUTPUT

4. ทำการป้อนคำสั่งทางลอจิกโดยการกดปุ่มเลือกที่คำสั่งซึ่งคสั่งที่ถูกเลือกจะแสดงในส่วนของหน้าจอแสดงผล หลังจากนั้นกดปุ่ม INPUT เพื่อที่จะเลือกว่าใช้พอร์ตไหน



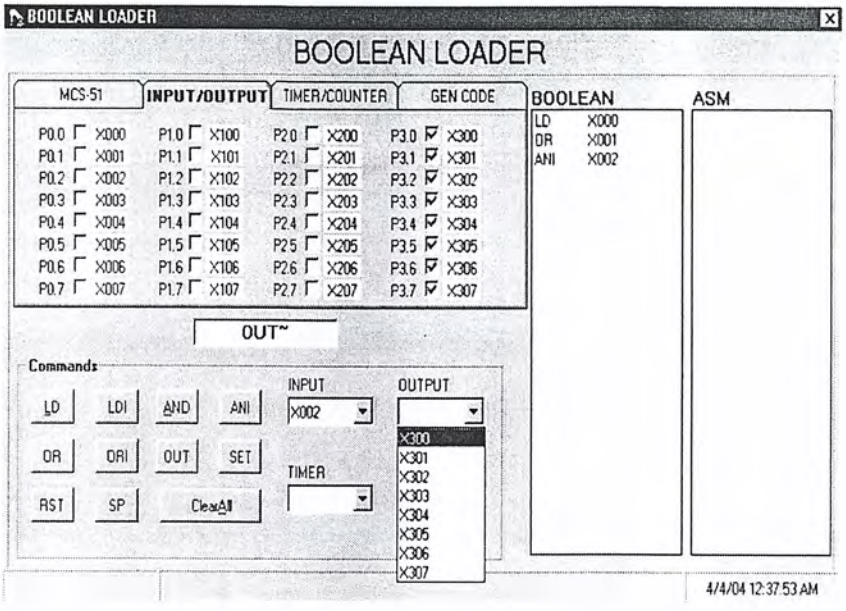
ภาพที่ 4.8 แสดงการเลือกพอร์ตอินพุต

5. หลังจากกดเลือกพอร์ตอินพุตโดยการคลิกที่เบอร์อินพุต ค่าคำสั่งที่เราป้อนเข้าไปจะปรากฏยังจอแสดงผลทางด้านในส่วนคำสั่งทางลอจิกบูลลีน



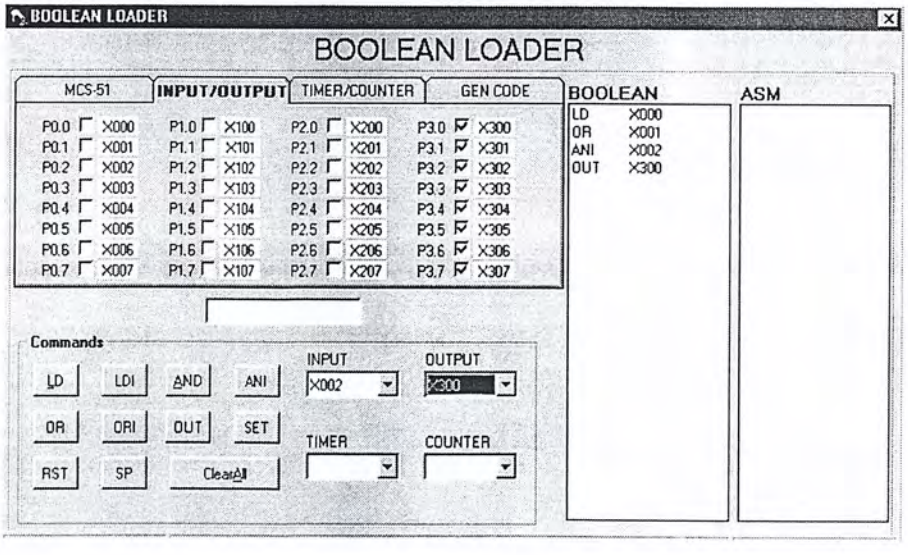
ภาพที่ 4.9 แสดงคำสั่งทางลอจิกที่ใช้ยังจอแสดงผล

6. เมื่อต้องการป้อนคำสั่งลอจิกทางเอาท์พุต ให้ทำการเลือกพอร์ตที่ต้องการใช้งานให้เอาท์พุตก่อน โดยทำเครื่องหมายหน้าพอร์ตที่ต้องการใช้ หลังจากนั้นกดปุ่ม OUTPUT โดยจะแสดงเบอร์เอาต์พุตตามจำนวนที่เราเลือกไว้



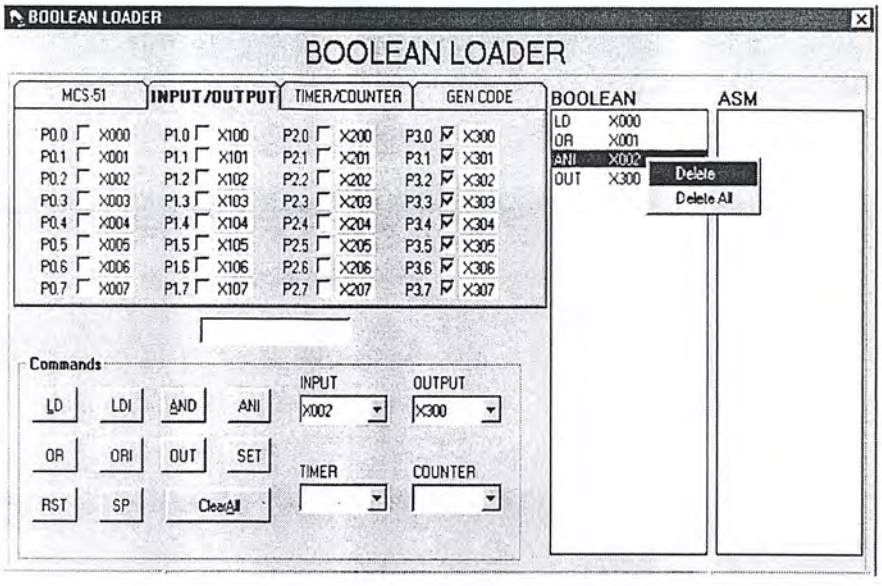
ภาพที่ 4.10 แสดงการกดเลือกใช้พอร์ต OUT

หลังจากนั้นป้อนคำสั่ง OUT แล้วกดปุ่มเลือกว่าจะให้ออกเอาต์พุตไหน คำสั่ง OUT ก็จะไปแสดงผลยังส่วนแสดงผลทางลอจิก



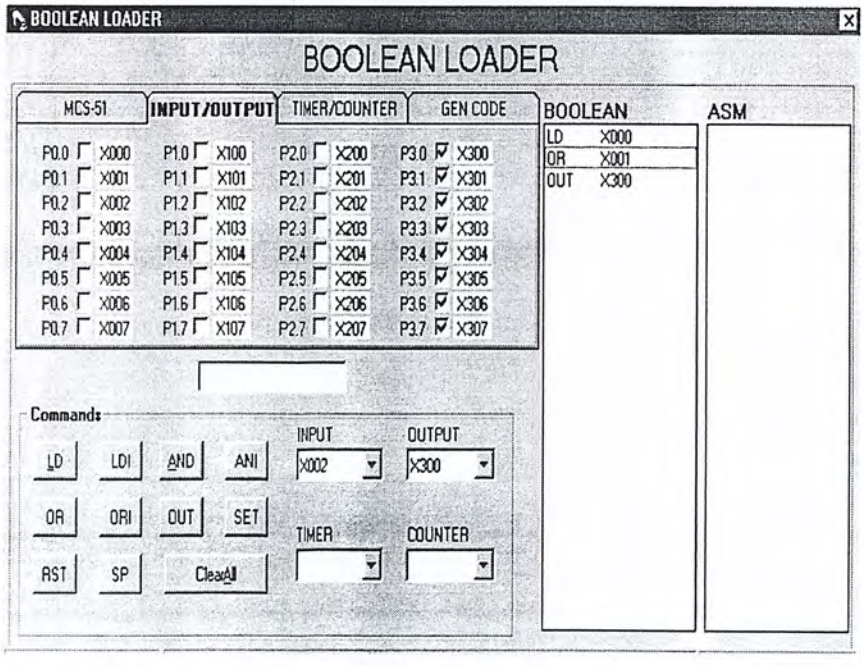
ภาพที่ 4.11 แสดงการใช้คำสั่ง OUTPUT

7. หากกรณีป้อนคำสั่งผิดต้องการลบคำสั่งนั้น ให้ทำการคลิกที่ตัวคำสั่งนั้นให้เกิดสีทึบ แล้วทำการคลิกขวาจะเกิดคำสั่งให้เลือก 2 คำสั่งว่าจะทำการลบคำสั่งที่เลือกไว้โดยการกดปุ่ม Delete หรือจะเลือกปุ่ม DeleteAll เพื่อลบคำสั่งทั้งหมดทิ้งไป



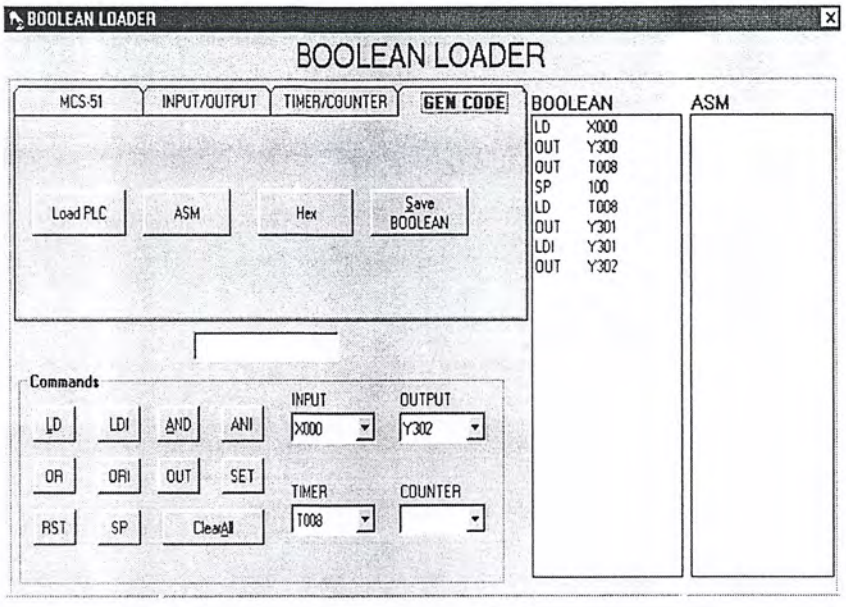
ภาพที่ 4.12 แสดงการลบคำสั่งโดยใช้ Delete

หลังจากนั้นคำสั่งที่ถูก Delete ก็จะหายไป



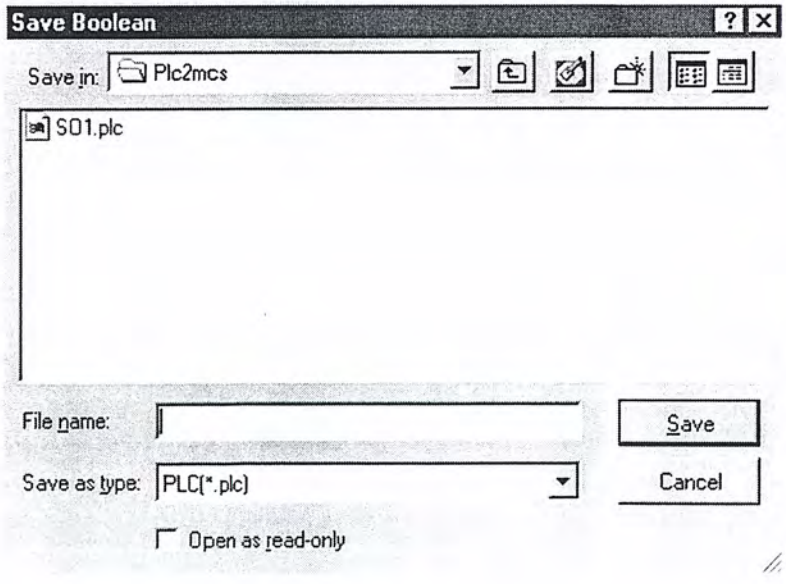
ภาพที่ 4.13 แสดงภาพหลังจากกด Delete

8. หลังจากป้อนคำสั่งทางลอจิกเสร็จเรียบร้อยแล้ว ให้ทำการกดปุ่ม Gen Code เพื่อต้องทำการ Save ค่าคำสั่งเหล่านั้นไว้ก่อน โดยการกดปุ่ม Save Boolean



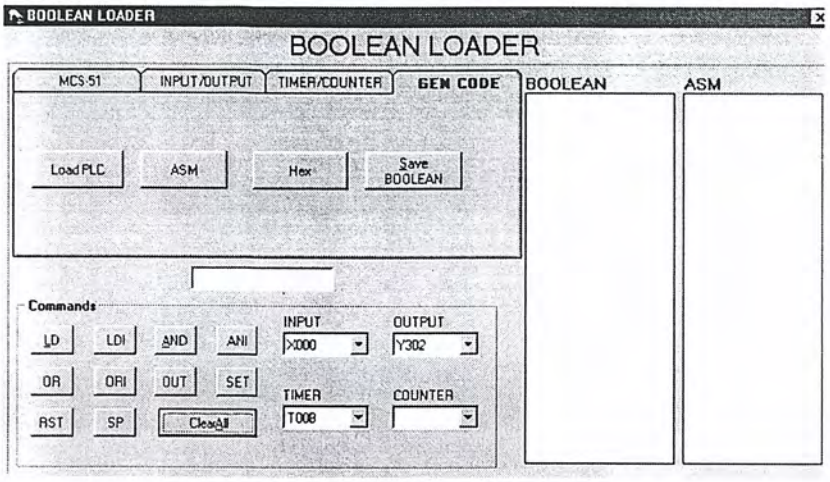
ภาพที่ 4.14 แสดงคำสั่งทางลอจิกก่อนการ Save

หลังจากนั้นคำสั่งก็จะแสดงยังหน้าจอเพื่อให้ต้องการ Save คำสั่งโดยการตั้งชื่อไฟล์ ที่จะ Save ให้ทำการ Save



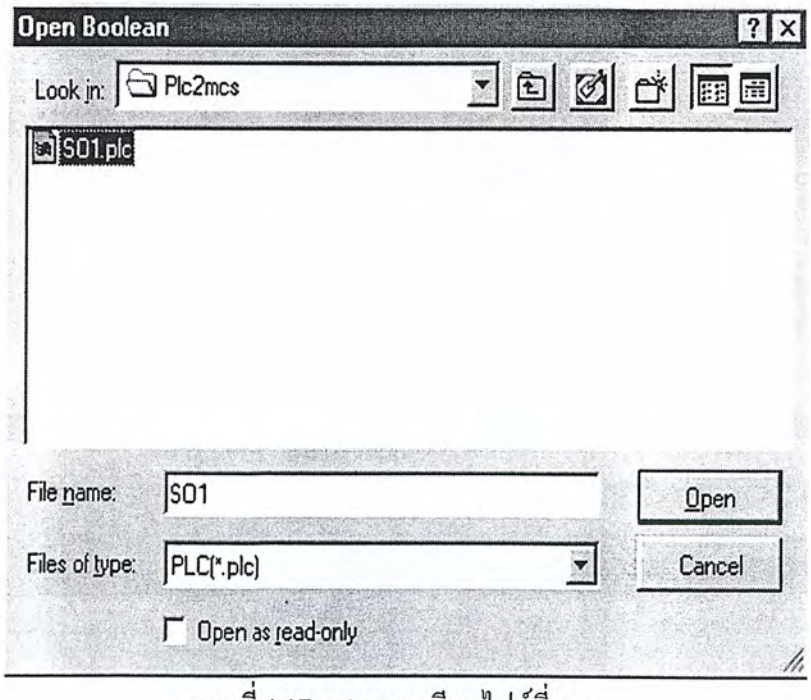
ภาพที่ 4.15 แสดงการตั้งชื่อและ Save ไฟล์

9. หลังจากการ Save ค่าคำสั่งทางลอจิกเรียนร้อยแล้วอาจลบคำสั่งทั้งหมดทิ้งไปเพื่อที่จะเรียกคำสั่งจากที่อื่นที่เรา Saveไว้ โดยการกดปุ่ม ClearAll

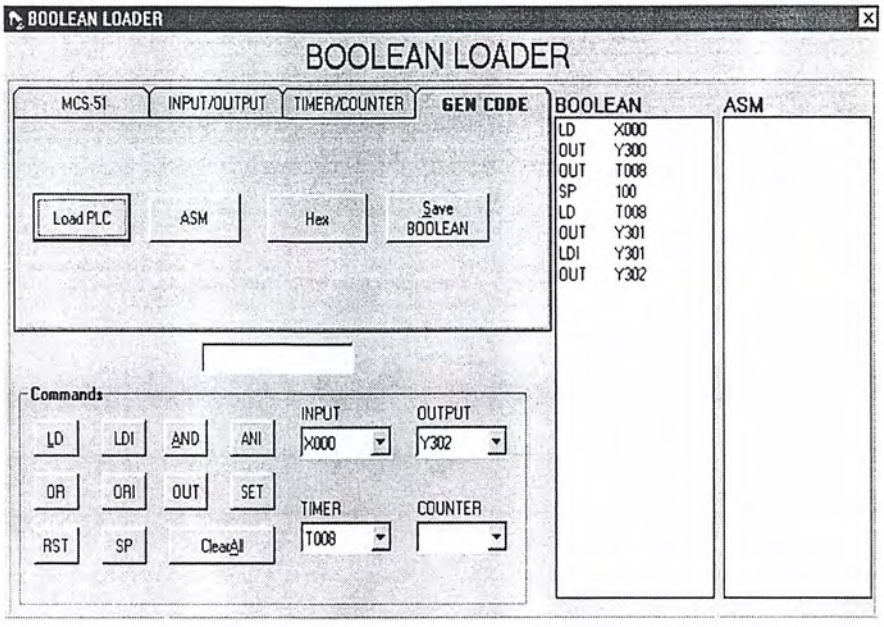


ภาพที่ 4.16 แสดงการใช้คำสั่ง ClearAll

เวลาต้องการเรียกใช้ให้กดปุ่ม LoodBoolean เรียกชื่อไฟล์ที่ Save ไว้ตอนแรก คำสั่งที่เราClearทิ้งไปก็จะคืนมา

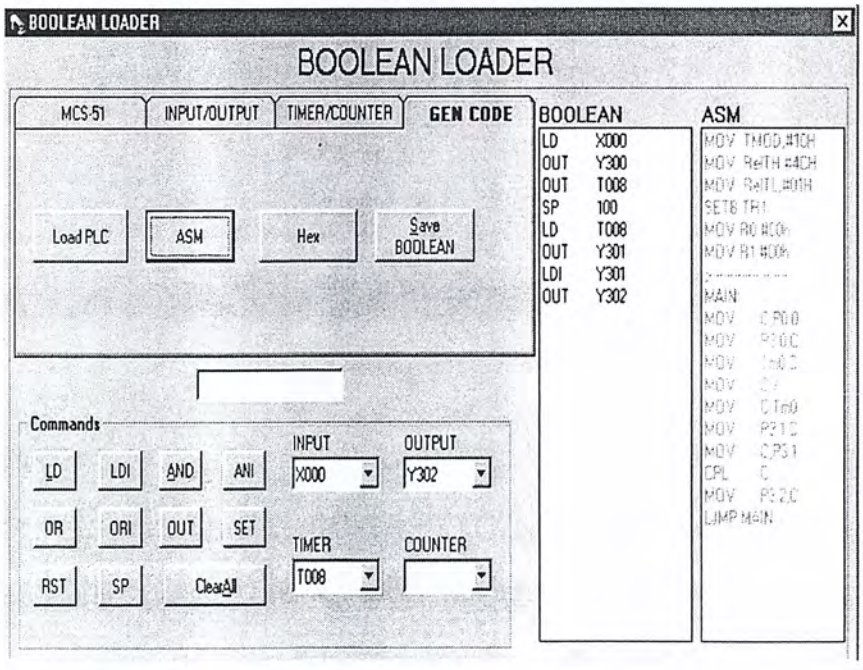


ภาพที่ 4.17 แสดงการเรียกไฟล์ที่ Save



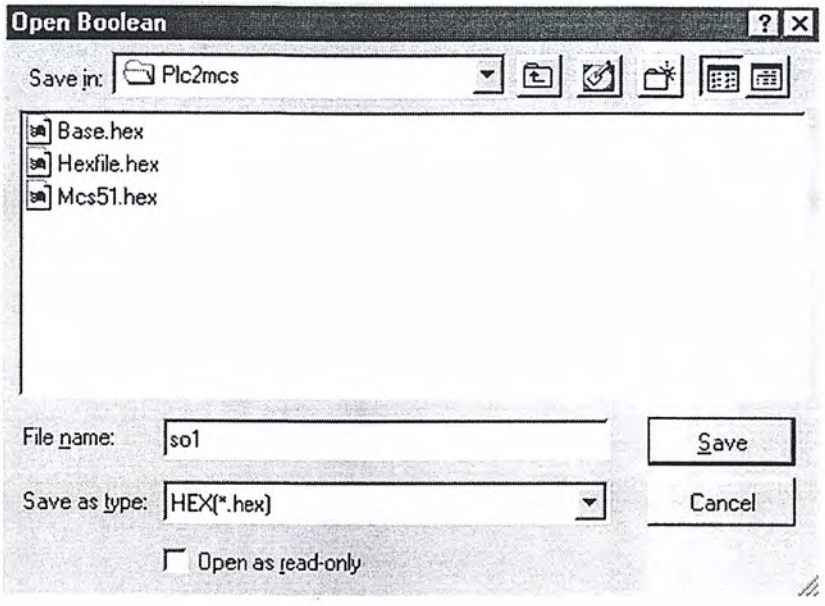
ภาพที่ 4.18 แสดงคำสั่งทางลอจิกที่ถูกเรียกกลับมา

10. ทำการแปลงคำสั่งทางลอจิกที่อยู่ในรูปของบูลลีน ให้เป็นภาษาแอสเซมบลี(.ASM) โดยการกดเลือกปุ่ม ASM คำสั่งทั้งหมดจะแปลงเป็นภาษาแอสเซมบลีแสดงยังจอแสดงผล



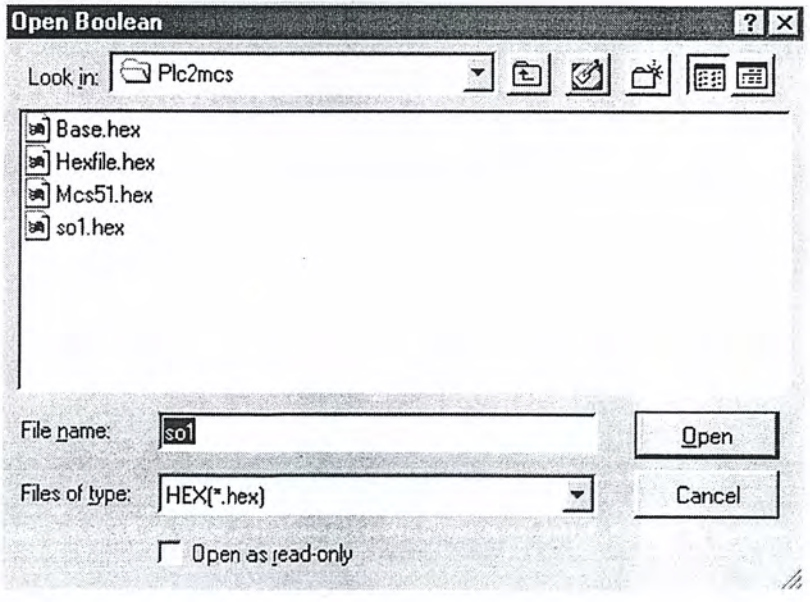
ภาพที่ 4.19 แสดงการแปลงคำสั่งทางลอจิกเป็นแอสเซมบลี

11. ต้องการแปลงคำสั่งที่เป็นแอสเซมบลีให้เป็น Hex File(.Hex) โดยการกดปุ่ม HEX คำสั่งจะสั่งให้ปรากฏหน้าจอสำหรับ Save เป็นไฟล์ (.Hex) ให้ตั้งชื่อไฟล์แล้วการ Save



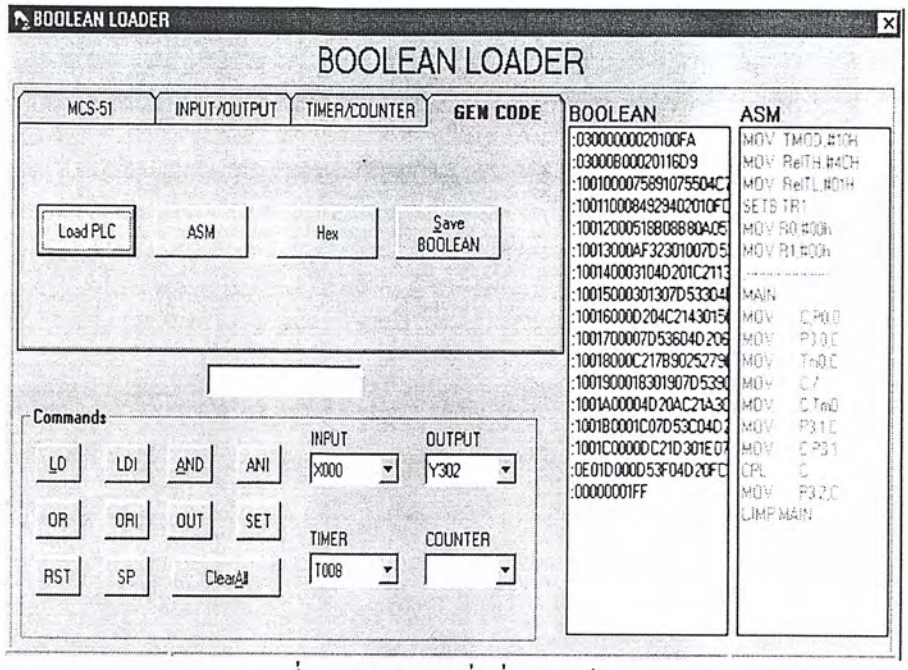
ภาพที่ 4.20 การตั้งชื่อและ Save ไฟล์ที่เป็น Hex File(.Hex)

12. ทำการโหลดไฟล์ที่ Save เก็บไว้โดยการกดปุ่ม Load Boolean คำสั่งจะแสดงหน้าจอให้เลือกไฟล์ดังรูป



ภาพที่ 4.21 แสดงการเรียกไฟล์

12. เลือกไฟล์ที่ Save แล้วทำการ Open Flie โดยการกดปุ่ม Open คำสั่งทั้งหมดจะถูกแปลงเป็น Hex File(.Hex)



ภาพที่ 4.22 แสดงคำสั่งที่แปลงเป็นHex File

หลังจากนั้นนำคำสั่งที่ถูกแปลงให้อยู่ในรูปของ Hex File ไปทำการอัดโปรแกรมลงตัว MCS-51 ขั้นตอนการอัดโปรแกรมเหมือนกับการเขียนคำสั่งด้วยแอสเซมบลี

บรรณานุกรม

ธีรวัฒน์ ประกอบผล , " การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ " , บริษัทดวงกมลสมัยจำกัด , 2546

ชัยวัฒน์ ลิ้มพรจิตรวิไล , " การเรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ " , บริษัทอินโนเวตีฟเอ็กเพอริเมนต์จำกัด, 2539

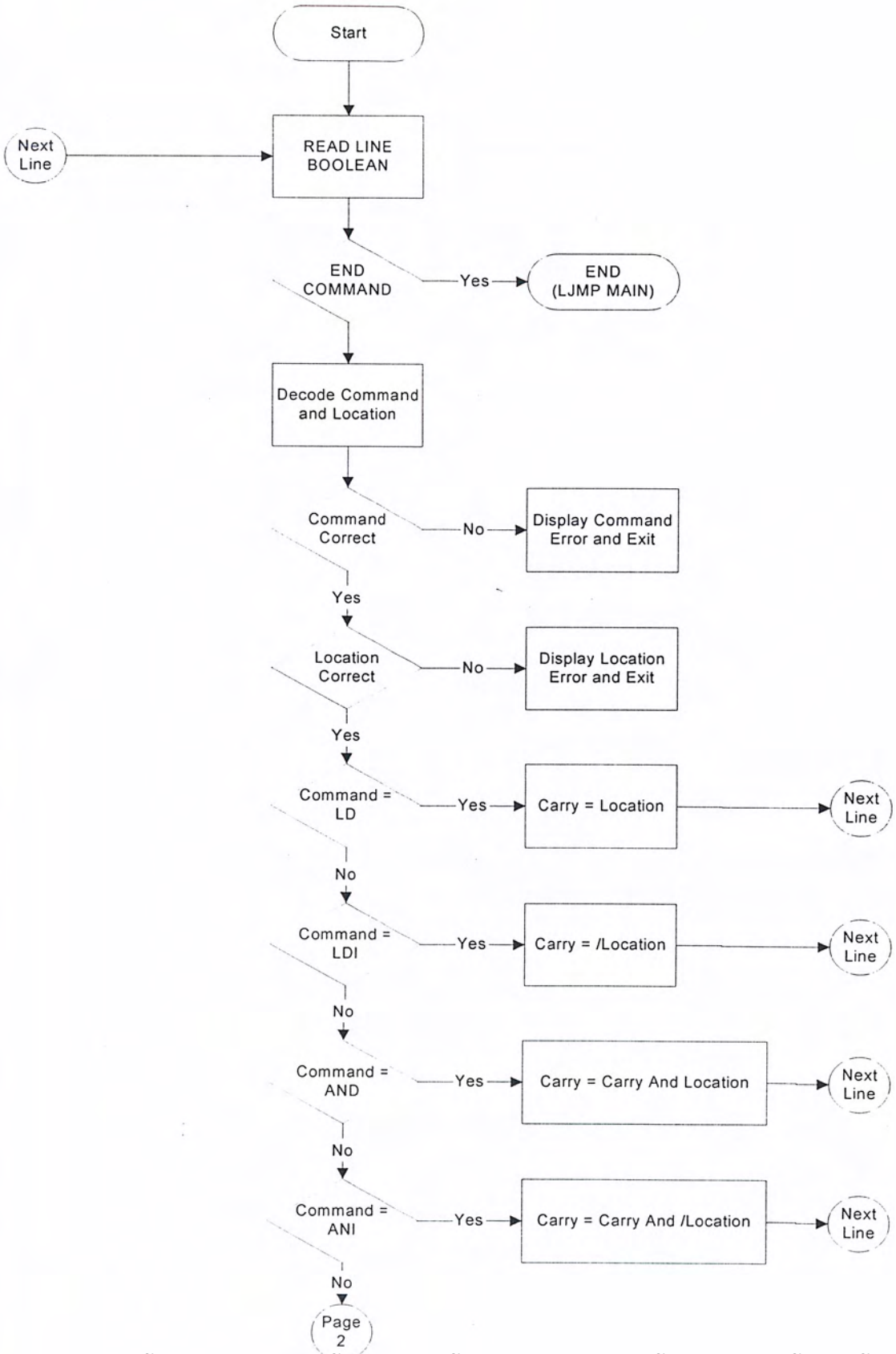
รศ. ธีรศิลป์ ทุมวิภาต, สุภาพร จำปาทอง, " เรียนรู้ PLC เบื้องต้นด้วยตนเอง " , บริษัทซีเอ็ดยูเคชั่นจำกัด (มหาชน), 2545

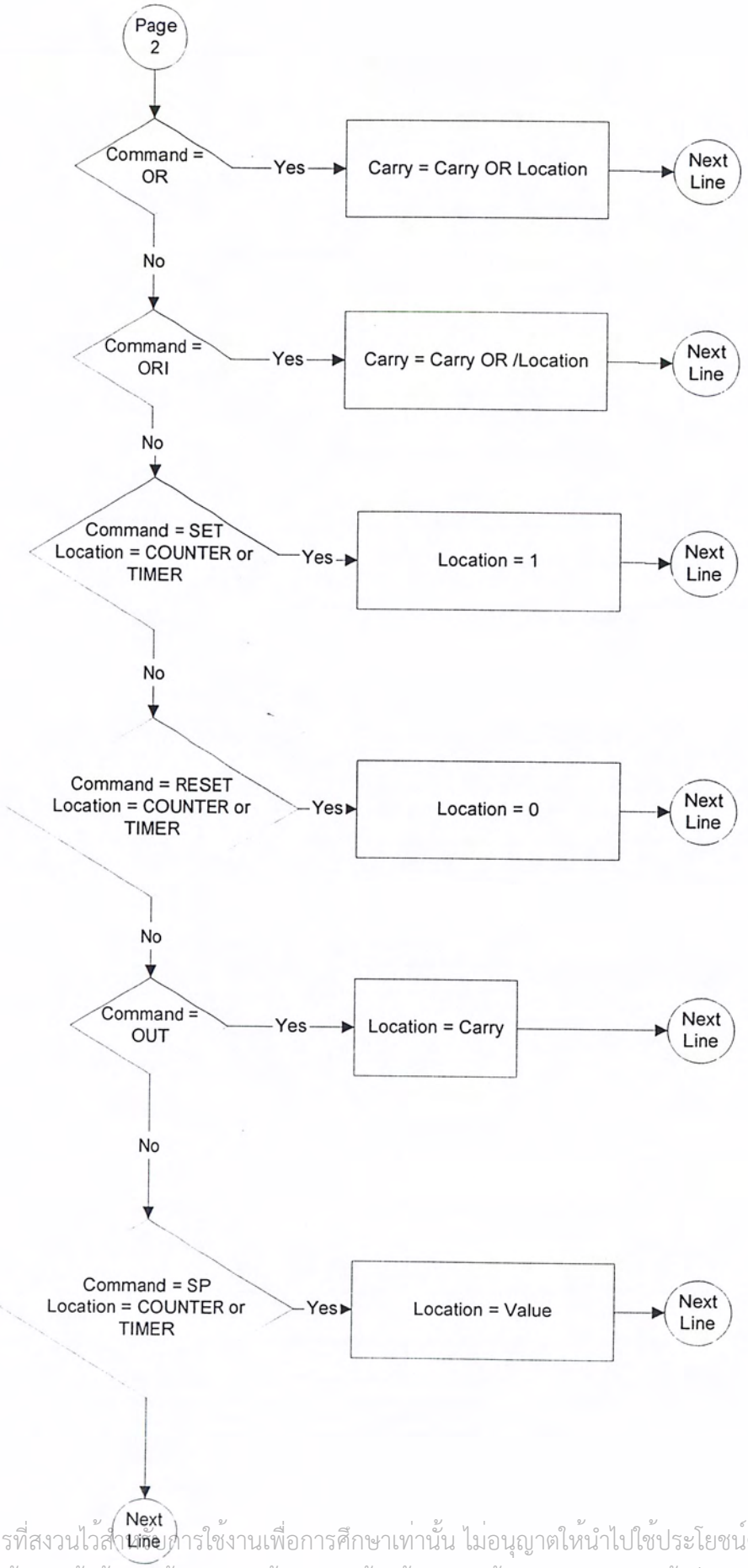
รศ. กฤษดา วิศวธีรานนท์, " PC ตัวควบคุมซีเคว้นหลักการทำงานและประยุกต์ " , บริษัทบุญชัยวิศวกรรมจำกัด, 2534

ชัยวุฒิ จันมา, " การเขียนโปรแกรมด้วย Visual Basic 6.0 " , บริษัทซีเอ็ดยูเวชั่นจำกัด (มหาชน), 2540

ภาคผนวก

Softwere Deccode Boolean





intel

PRELIMINARY

87C51/80C51BH/80C31BH
CHMOS SINGLE-CHIP 8-BIT MICROCONTROLLER

Commercial/Express

87C51/80C51BH/80C51BHP/80C31BH
***See Table 1 for Proliferation Options**

- High Performance CHMOS EPROM
- 24 MHz Operation
- Improved Quick-Pulse Programming Algorithm
- 3-Level Program Memory Lock
- Boolean Processor
- 128-Byte Data RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Extended Temperature Range (−40°C to +85°C)
- 5 Interrupt Sources
- Programmable Serial Port
- TTL- and CMOS-Compatible Logic Levels
- 64K External Program Memory Space
- 64K External Data Memory Space
- ONCE Mode Facilitates System Testing
- Power Control Modes
  — Idle
  — Power Down

MEMORY ORGANIZATION

PROGRAM MEMORY: Up to 4 Kbytes of the program memory can reside on-chip (except 80C31BH). In addition the device can address up to 64K of program memory external to the chip.

DATA MEMORY: This microcontroller has a 128 x 8 on-chip RAM. In addition it can address up to 64 Kbytes of external data memory.

The Intel 87C51/80C51BH/80C31BH is a single-chip control-oriented microcontroller which is fabricated on Intel's reliable CHMOS III-E technology. Being a member of the MCS® 51 controller family, the 87C51/80C51BH/80C31BH uses the same powerful instruction set, has the same architecture, and is pin-for-pin compatible with the existing MCS 51 controller family of products.

The 80C51BHP is identical to the 80C51BH. When ordering the 80C51BHP, customers must submit the 64 byte encryption table together with the ROM code. Lock bit 1 will be set to enable the internal ROM code protection and at the same time allows code verification.

The extremely low operating power, along with the two reduced power modes, Idle and Power Down, make this part very suitable for low power applications. The Idle mode freezes the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down mode saves the RAM contents but freezes the oscillator, causing all other chip functions to be inoperative.

For the remainder of this document, the 87C51, 80C51BH, and 80C31BH will be referred to as the 87C51/BH, unless information applies to a specific device.

October 1995
Order Number: 272335-003

Table 1. Proliferation Options

	*Standard	-1	-2	-24
80C31BH	X	X	X	X
80C51BH	X	X	X	X
80C51BHP	X	X	X	X
87C51	X	X	X	X

NOTES:
* 3.5 MHz to 12 MHz; V_{CC} = 5V ±20%
-1 3.5 MHz to 16 MHz; V_{CC} = 5V ±20%
-2 0.5 MHz to 12 MHz; V_{CC} = 5V ±20%
-24 3.5 MHz to 24 MHz; V_{CC} = 5V ±20%

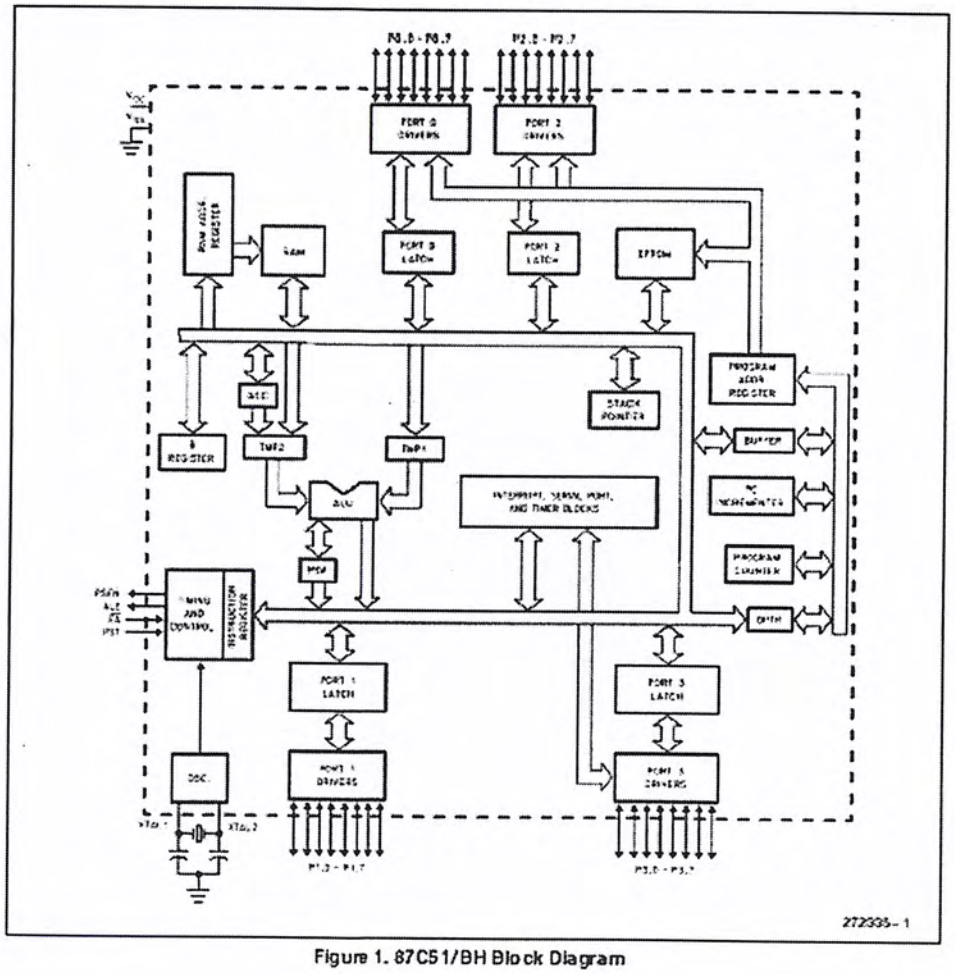


Figure 1. 87C51/BH Block Diagram

PIN DESCRIPTION

V_{CC}: Supply voltage during normal, Idle and Power Down operations.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink several LS TTL inputs. Port 0 pins that have 1's written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pullups when emitting 1's.

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

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can drive LS TTL inputs. Port 1 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

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

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. Port 2 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally pulled low will source current (I_{IL}, on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program memory and during accesses to external Data Memory that use 16-bit address (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1's.

During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

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

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can drive LS TTL inputs. Port 3 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally pulled low will source current (I_{IL}, on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Pin	Name	Alternate Function
P3.0	RXD	Serial input line
P3.1	TXD	Serial output line
P3.2	$\overline{INT0}$	External Interrupt 0
P3.3	$\overline{INT1}$	External Interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 1 external input
P3.6	$\overline{WR}$	External Data Memory Write strobe
P3.7	$\overline{RD}$	External Data Memory Read strobe

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

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. The port pins will be driven to their reset condition when a minimum V_{IH1} voltage is applied whether the oscillator is running or not. An internal pulldown resistor permits a power-on reset with only a capacitor connected to V_{CC}.

ALE/$\overline{PROG}$: Address Latch Enable output signal for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during EPROM programming for the 87C51.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With this bit set, the pin is weakly pulled high. However, the ALE disable feature will be suspended during a MOVX or MOVC instruction, idle mode, power down mode and ICE mode. The ALE disable feature will be terminated by reset. When the ALE disable feature is suspended or terminated, the ALE pin will no longer be pulled up weakly. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

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

$\overline{PSEN}$: Program Store Enable is the Read strobe to External Program Memory. When the 87C51/BH is executing from Internal Program Memory, $\overline{PSEN}$ is inactive (high). When the device is executing code from External Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to External Data Memory.

$\overline{EA}$/V_{PP}: External Access enable. $\overline{EA}$ must be strapped to V_{SS} in order to enable the 87C51/BH to fetch code from External Program Memory locations starting at 0000H up to FFFFH. Note, however, that if either of the Lock Bits is programmed, the logic level at $\overline{EA}$ is internally latched during reset.

$\overline{EA}$ must be strapped to V_{CC} for internal program execution.

This pin also receives the programming supply voltage (V_{PP}) during EPROM programming.

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

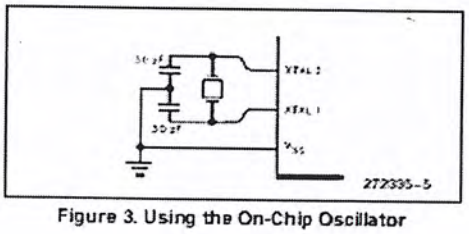


Figure 3. Using the On-Chip Oscillator

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3.

To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the data sheet must be observed.

An external oscillator may encounter as much as a 100 pF load at XTAL1 when it starts up. This is due to interaction between the amplifier and its feedback capacitance. Once the external signal meets the V_{IL} and V_{IH} specifications the capacitance will not exceed 20 pF.

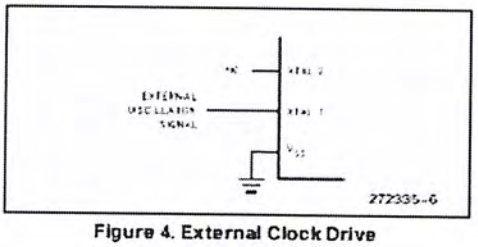


Figure 4. External Clock Drive

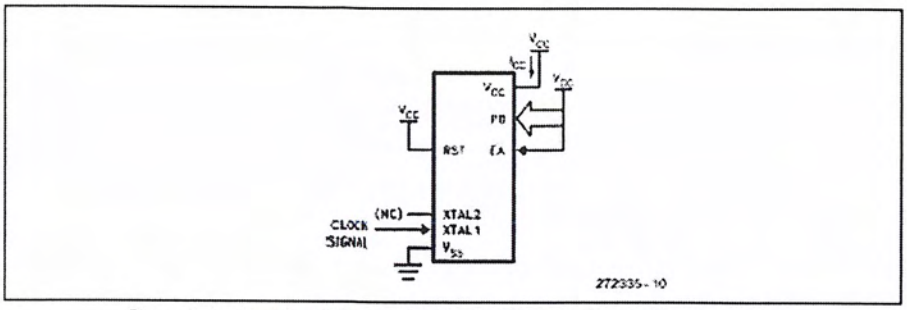


Figure 6. I_{CC} Test Condition, Active Mode. All other pins are disconnected.

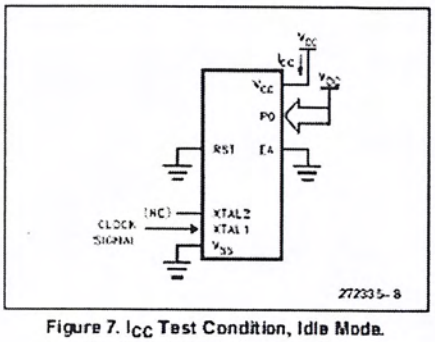


Figure 7. I_{CC} Test Condition, Idle Mode. All other pins are disconnected.

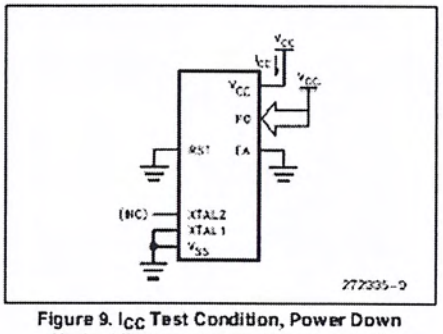


Figure 9. I_{CC} Test Condition, Power Down Mode. All other pins are disconnected. V_{CC} = 2V to 5.5V.

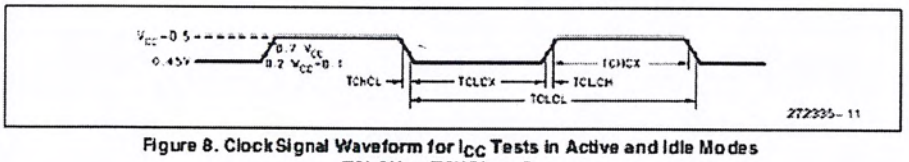


Figure 8. Clock Signal Waveform for I_{CC} Tests in Active and Idle Modes
TCLCH = TCHCL = 5 ns

EXTERNAL MEMORY CHARACTERISTICS

All parameter values apply to all devices unless otherwise indicated. In this table, 87C51/BH refers to 87C51/BH, 87C51-1/BH-1 and 87C51-2/BH-2. (Continued)

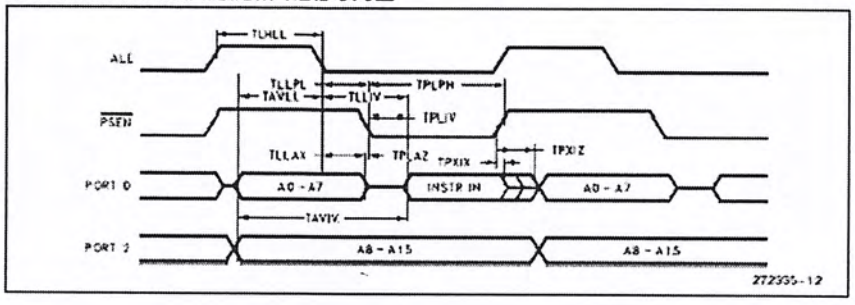
Symbol	Parameter	Oscillator						Units
		12 MHz		24 MHz		Variable		
		Min	Max	Min	Max	Min	Max	
TPXIX	Input Instr Hold After $\overline{PSEN}$	0		0		0		ns
TPXIZ	Input Instr Float After $\overline{PSEN}$ 87C51/BH 87C51-24/BH-24		59		 21		TCLCL − 25 TCLCL − 20	 ns ns
TAVIV	Address to Valid Instr In		312		103		5TCLCL − 105	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		10		10		10	ns
TRLRH	$\overline{RD}$ Pulse Width	400		150		6TCLCL − 100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		150		6TCLCL − 100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In 87C51/BH 87C51-24/BH-24		252		 113		5TCLCL − 165 5TCLCL − 95	 ns ns
TRHDX	Data Hold After $\overline{RD}$	0		0		0		ns
TRHDZ	Data Float After $\overline{RD}$		107		23		2TCLCL − 60	ns
TLLDV	ALE Low to Valid Data In 87C51/BH 87C51-24/BH-24		517		 243		8TCLCL − 150 8TCLCL − 90	 ns ns
TAVDV	Address to Valid Data In 87C51/BH 87C51-24/BH-24		585		 285		9TCLCL − 165 9TCLCL − 90	 ns ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	75	175	3TCLCL − 50	3TCLCL + 50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low 87C51/BH 87C51-24/BH-24	203		 77		4TCLCL − 130 4TCLCL − 90		ns ns
TQVWX	Data Valid to $\overline{WR}$ Transition 87C51/BH 80C51-24/BH-24	33		 12		TCLCL − 50 TCLCL − 30		ns ns

EXTERNAL MEMORY CHARACTERISTICS

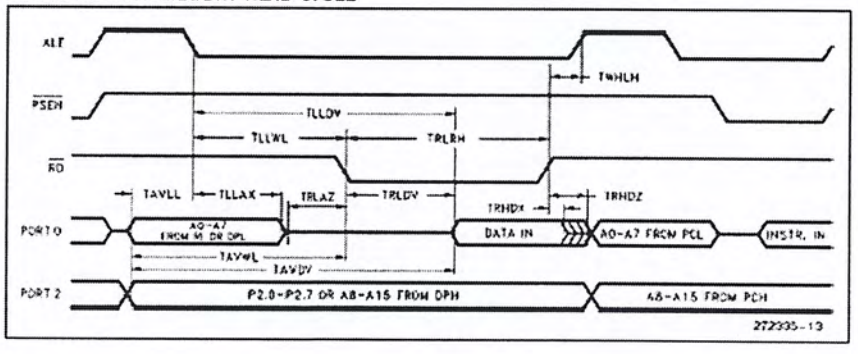
All parameter values apply to all devices unless otherwise indicated. In this table, 87C51/BH refers to 87C51/BH, 87C51-1/BH-1 and 87C51-2/BH-2. (Continued)

Symbol	Parameter	Oscillator						Units
		12 MHz		24 MHz		Variable		
		Min	Max	Min	Max	Min	Max	
TWHQX	Data Hold After $\overline{WR}$ 87C51/BH 87C51-24/BH-24	33		 7		 TCLCL − 50 TCLCL − 35		 ns ns
TQVWH	Data Valid to $\overline{WR}$ High 87C51/BH 87C51-24/BH-24	433		 222		 7TCLCL − 150 7TCLCL − 70		 ns ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High 87C51/BH 87C51-24/BH-24	43	123	 12	 71	 TCLCL − 40 TCLCL − 30	 TCLCL + 40 TCLCL + 30	 ns ns

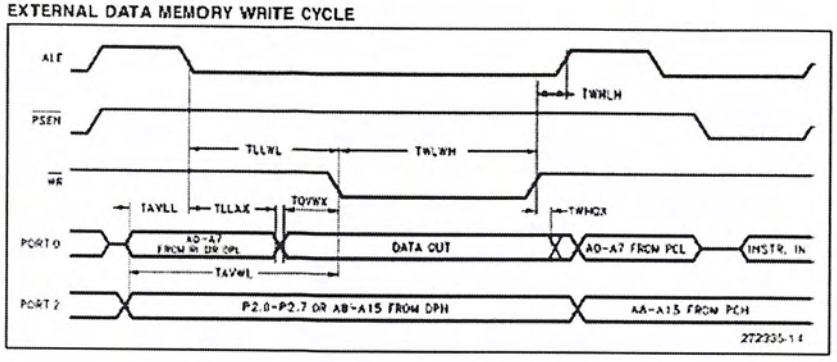
EXTERNAL PROGRAM MEMORY READ CYCLE



EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE

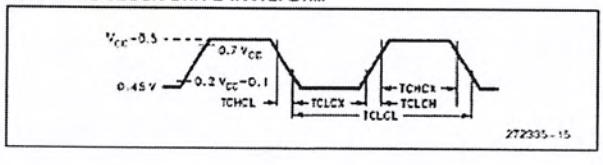


EXTERNAL CLOCK DRIVE

All parameter values apply to all devices unless otherwise indicated. In this table, 87C51/BH refers to 87C51/BH, 87C51-1/BH-1 and 87C51-2/BH-2.

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency			
	87C51/BH	3.5	12	MHz
	87C51-1/BH-1	3.5	16	MHz
	87C51-2/BH-2	0.5	12	MHz
	87C51-24/BH-24	3.5	24	MHz
TCHCX	High Time			
	87C51/BH	20		ns
	8751-24/BH-24	0.35TCLCL	0.65TCLCL	ns
TCLCX	Low Time			
	87C51/BH	20		ns
	87C51-24/BH-24	0.35TCLCL	0.65TCLCL	ns
TCLCH	Rise Time			
	87C51/BH		20	ns
	87C51-24/BH-24		10	ns
TCHCL	Fall Time			
	87C51/BH		20	ns
	87C51-24/BH-24		10	ns

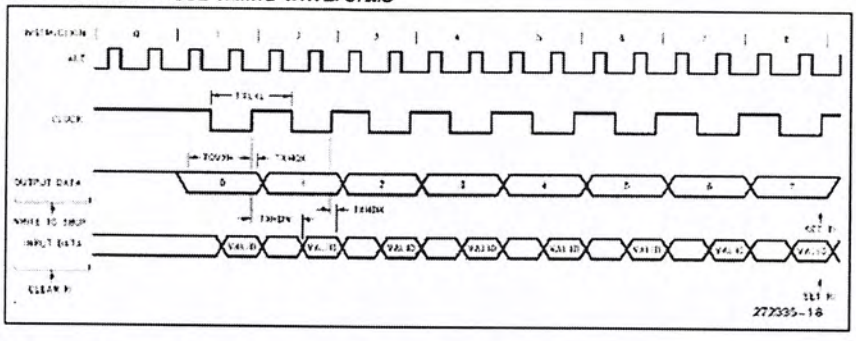
EXTERNAL CLOCK DRIVE WAVEFORM



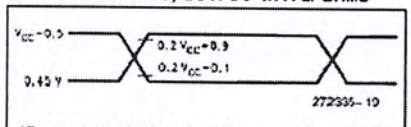
SERIAL PORT TIMING—SHIFT REGISTER MODE

Symbol	Parameter	12 MHz Oscillator		24 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		0.500		12TCLCL		µs
TQVXH	Output Data Setup to Clock Rising Edge	700		284		10TCLCL − 133		ns
TXHQX	Output Data Hold After Clock Rising Edge 87C51/BH 87C51-24/BH-24	50		34		2TCLCL − 117 2TCLCL − 34		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		283		10TCLCL − 133	ns

SHIFT REGISTER MODE TIMING WAVEFORMS

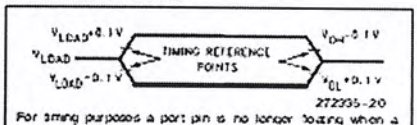


AC TESTING INPUT, OUTPUT WAVEFORMS



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".

FLOAT WAVEFORMS



For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs, and begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs. I_{OL}/I_{OH} ≥ ±20 mA.

PROGRAMMING THE 87C51

The part must be running with a 4 MHz to 6 MHz oscillator. The address of an EPROM location to be programmed is applied to address lines while the code byte to be programmed in that location is applied to data lines. Control and program signals must be held at the levels indicated in Table 4. Normally $\overline{EA}/V_{PP}$ is held at logic high until just before ALE/$\overline{PROG}$ is to be pulsed. The $\overline{EA}/V_{PP}$ is raised to V_{PP}, ALE/$\overline{PROG}$ is pulsed low and then $\overline{EA}/V_{PP}$ is returned to a high (also refer to timing diagrams).

NOTE:

- Exceeding the V_{PP} maximum for any amount of time could damage the device permanently. The V_{PP} source must be well regulated and free of glitches.

DEFINITION OF TERMS

ADDRESS LINES: P1.0–P1.7, P2.0–P2.5, P3.4 respectively for A0–A14.

DATA LINES: P0.0–P0.7 for D0–D7.

CONTROL SIGNALS: RST, $\overline{PSEN}$, P2.6, P2.7, P3.3, P3.6, P3.7.

PROGRAM SIGNALS: ALE/$\overline{PROG}$, $\overline{EA}/V_{PP}$.

Table 4. EPROM Programming Modes

Mode		RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.6	P2.7	P3.3	P3.6	P3.7
Program Code Data		H	L	⎍	12.75V	L	H	H	H	H
Verify Code Data		H	L	H	H	L	L	L	H	H
Program Encryption Array Address 0–3F		H	L	⎍	12.75V	L	H	H	L	H
Program Lock Bits	Bit 1	H	L	⎍	12.75V	H	H	H	H	H
	Bit 2	H	L	⎍	12.75V	H	H	H	L	L
	Bit 3	H	L	⎍	12.75V	H	L	H	H	L
Read Signature Byte		H	L	H	H	L	L	L	L	L

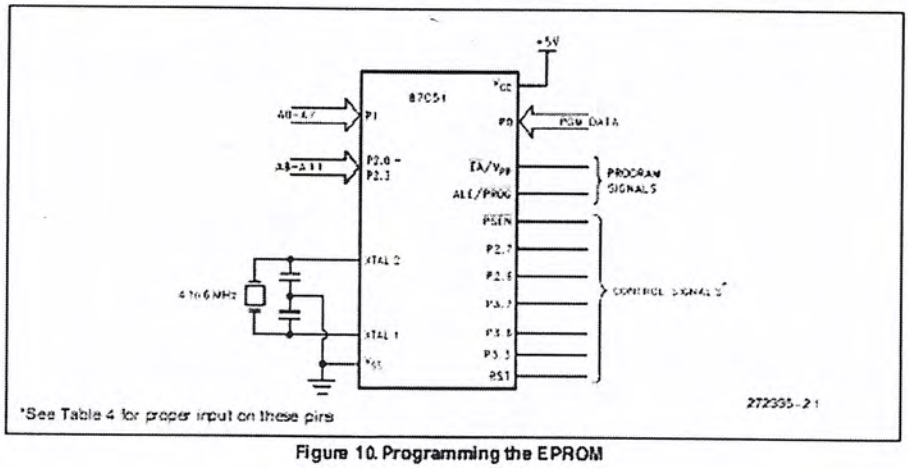


Figure 10. Programming the EPROM

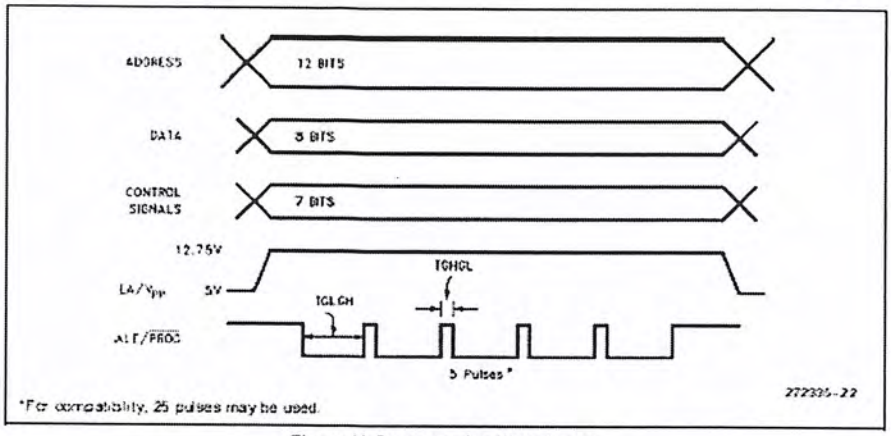


Figure 11. Programming Waveforms

PROGRAMMING ALGORITHM

Refer to Table 4 and Figures 10 and 11 for address, data, and control signals set up. To program the 87C51 the following sequence must be exercised.

1. Input the valid address on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}$/V_{PP} from V_{CC} to 12.75V ±0.25V.
5. Pulse ALE/$\overline{PROG}$ 5 times* for the EPROM array, and 25 times for the encryption table and the lock bits.

Repeat 1 through 5 changing the address and data for the entire array or until the end of the object file is reached.

Program Verify

Verification may be done after programming either one byte or a block of bytes. In either case a complete verify of the array will ensure reliable programming of the 87C51.

The lock bits cannot be directly verified. Verification of the lock bits is done by observing that their features are enabled.

ROM and EPROM Lock System

The program lock system, when programmed, protects the onboard program against software piracy.

The 80C51BH has a one level program lock system and a 64-byte encryption table. If program protection is desired, the user submits the encryption table with their code and both the lock bit and encryption array are programmed by the factory. The encryption array is not available without the lock bit. For the lock bit to be programmed, the user must submit an encryption table. The 87C51 has a 3-level program lock system and a 64-byte encryption array. Since this is an EPROM device, all locations are user-programmable. See Table 5.

Encryption Array

Within the EPROM array are 64 bytes of Encryption Array that are initially unprogrammed (all 1's). Every time that a byte is addressed during a verify, 6 address lines are used to select a byte of the Encryption Array. This byte is then exclusive-NOR'ed (XNOR) with the code byte, creating an Encryption Verify byte. The algorithm, with the array in the unprogrammed state (all 1's), will return the code in its original, unmodified form. For programming the Encryption Array, refer to Table 4 (Programming the EPROM).

When using the encryption array, one important factor needs to be considered. If a code byte has the value 0FFH, verifying the byte will produce the encryption byte value. If a large block (>64 bytes) of code is left unprogrammed, a verification routine will display the contents of the encryption array. For this reason all unused code bytes should be programmed with some value other than 0FFH, and not all of them the same value. This will ensure maximum program protection.

Program Lock Bits

The 87C51 has 3 programmable lock bits that when programmed according to Table 5 will provide different levels of protection for the on-chip code and data.

Erasing the EPROM also erases the encryption array and the program lock bits, returning the part to full functionality.

Reading the Signature Bytes

The 87C51 and 80C51BH have 3 signature bytes in locations 30H, 31H, and 60H. To read these bytes follow the procedure for EPROM verify, but activate the control lines provided in Table 4 for Read Signature Byte.

Location	Device	Contents
30H	All	89H
31H	All	58H
60H	87C51	51H
	80C51BH	11H

Erasure Characteristics (Windowed Devices Only)

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm^2 rating for 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1's state.

Table 5. Program Lock Bits and the Features

Program Lock Bits				Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features enabled. (Code verify will still be encrypted by the encryption array if programmed.)
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled.
3	P	P	U	Same as 2, also verify is disabled.
4	P	P	P	Same as 3, also external execution is disabled.

Thermal Impedance

All thermal impedance data is approximate for static air conditions at 1W of power dissipation. Values will change depending on operating conditions and applications. See the Intel Packaging Handbook (Order No. 240800) for a description of Intel's thermal impedance test methodology.

Package	θ_{JA}	θ_{JC}	Device
P	45°C/W	16°C/W	87C51
	75°C/W	23°C/W	BH
D	45°C/W	15°C/W	87C51
	35°C/W	13°C/W	BH
N	46°C/W	16°C/W	All
S	96°C/W	24°C/W	All

DATA SHEET REVISION HISTORY

Data sheets are changed as new device information becomes available. Verify with your local Intel sales office that you have the latest version before finalizing a design or ordering devices.

The following differences exist between this datasheet (272335-003) and the previous version (272335-002):

1. Removed -20 and -3 spec, replaced with -24 spec.
2. Added -24 spec.
3. 80C51BHP is replaced by 80C51BH with 64-byte encryption table submitted and lock bit 1 set.
4. 80C51BH/80C31BH are now having some additional features as 87C51.
5. Revised PRST value and I_{CC} idle values.
6. Added P3.3 control pin to programming and verification.
7. Added 80C51BH signature byte.

The following differences exist between the "-002" and the "-001" version of the 87C51/80C51BH/80C31BH datasheet.

1. Removed -L, I_{OL} = ±10 mA from Float Waveforms figure.

2. Removed QP, QD and QN (commercial with extended burn-in) from Table 3. Prefix Identification.

This data sheet (272335-001) replaces the following:

80C51BH/80C31BH Express	270216-003
80C51BHP	270603-004
87C51/80C51BH/80C31BH	270147-006
87C51 Express	270430-002
87C51-20/-3	272082-002

Features

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

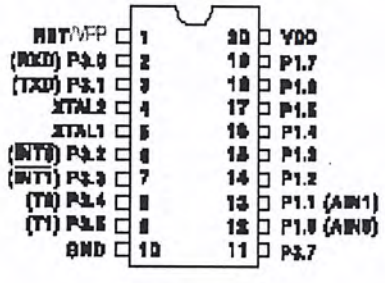
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2K Bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2K Bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

PDIP/SOIC

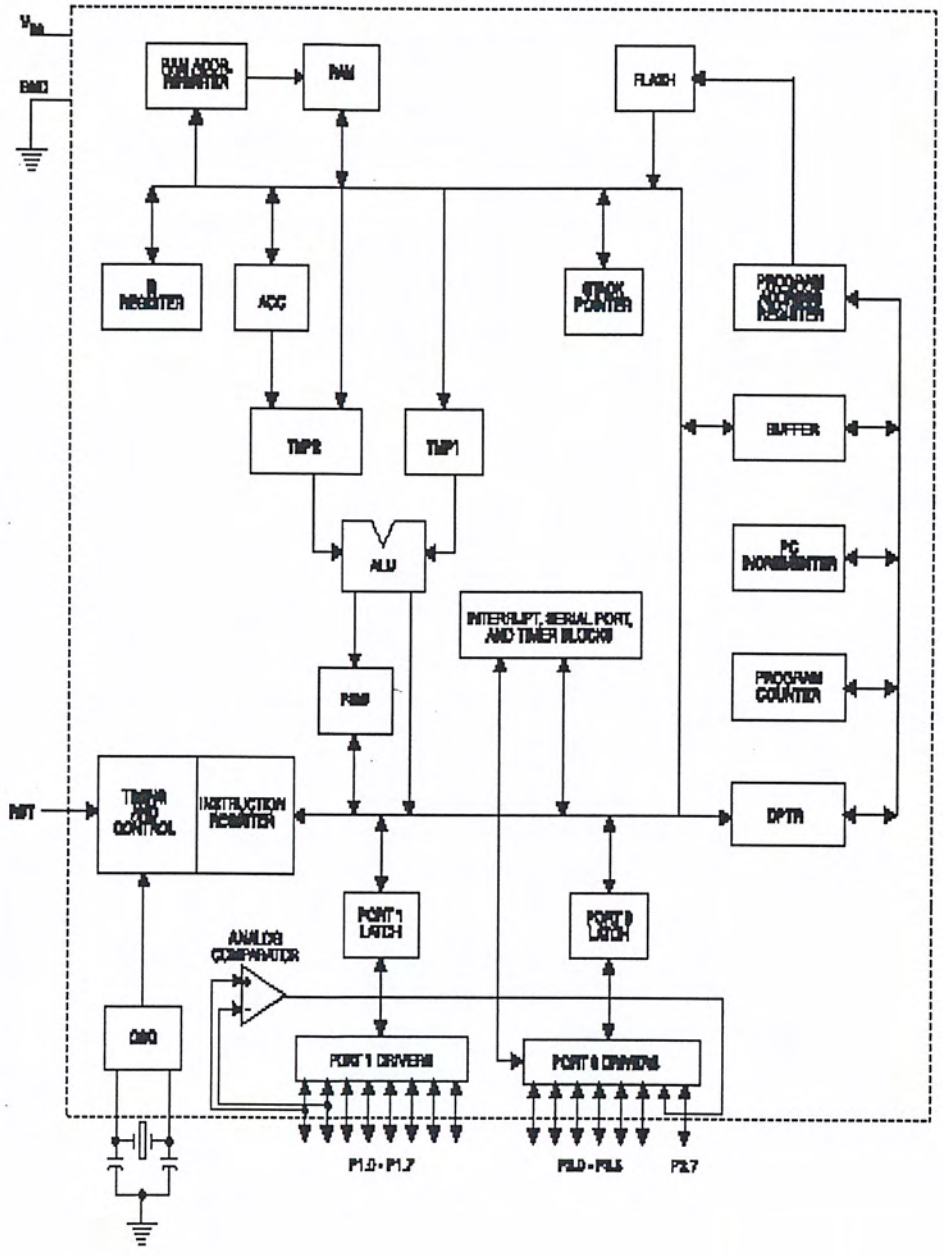


8-Bit Microcontroller with 2K Bytes Flash

AT89C2051

0368D-B-12/97

Block Diagram



Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 1

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

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

Port 3

Port 3 pins P3.0 to P3.5, P3.7 are seven bidirectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C2051 as listed below:

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

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

RST

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

Each machine cycle takes 12 oscillator or clock cycles.

XTAL1

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

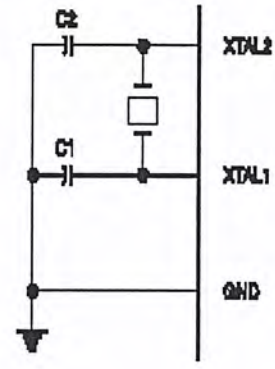
XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

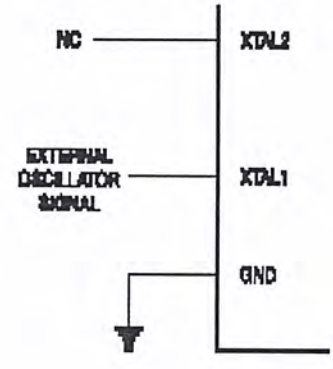
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected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

Figure 1. Oscillator Connections



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

Figure 2. External Clock Drive Configuration



Special Function Registers

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

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

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

Table 1. AT89C2051 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H									0CFH
0C0H									0C7H
0B8H	IP XXX00000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0XX00000								0AFH
0A0H									0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H		SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000	87H

Flash Programming Modes

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

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

Figure 3. Programming the Flash Memory

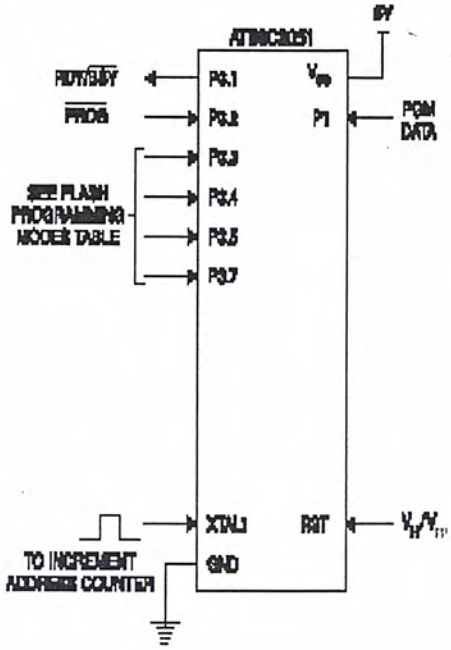
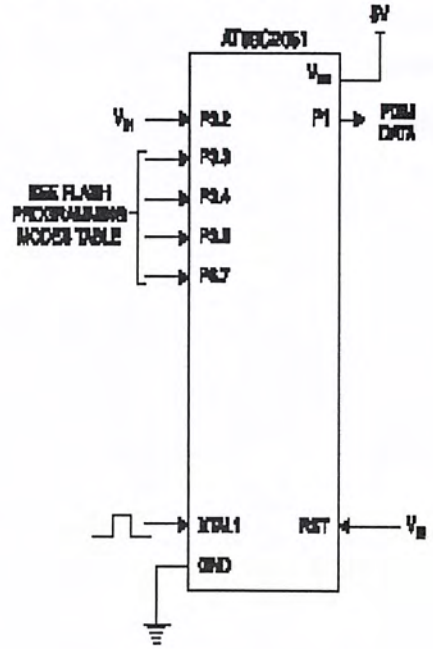


Figure 4. Verifying the Flash Memory



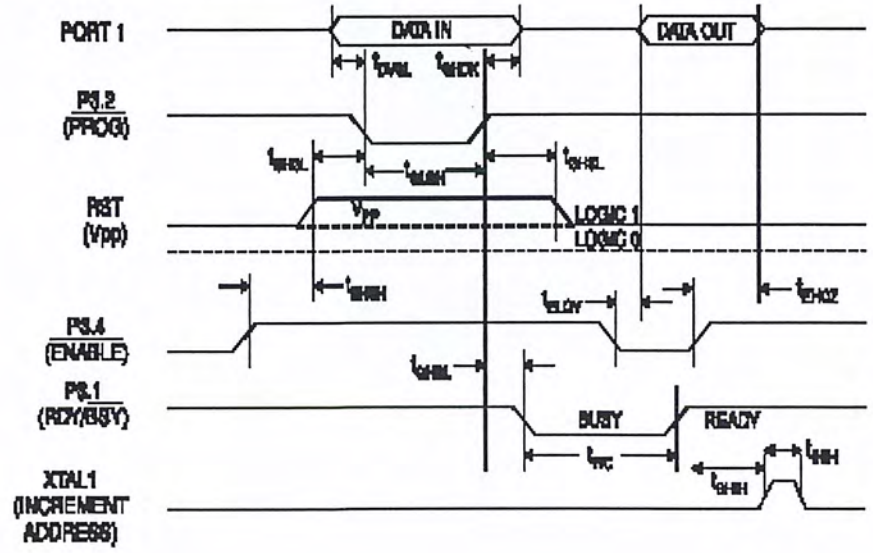
Flash Programming and Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ± 10%

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

Note: 1. Only used in 12-volt programming mode.

Flash Programming and Verification Waveforms



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	25.0 mA

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

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.0V to 6.0V (unless otherwise noted)

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

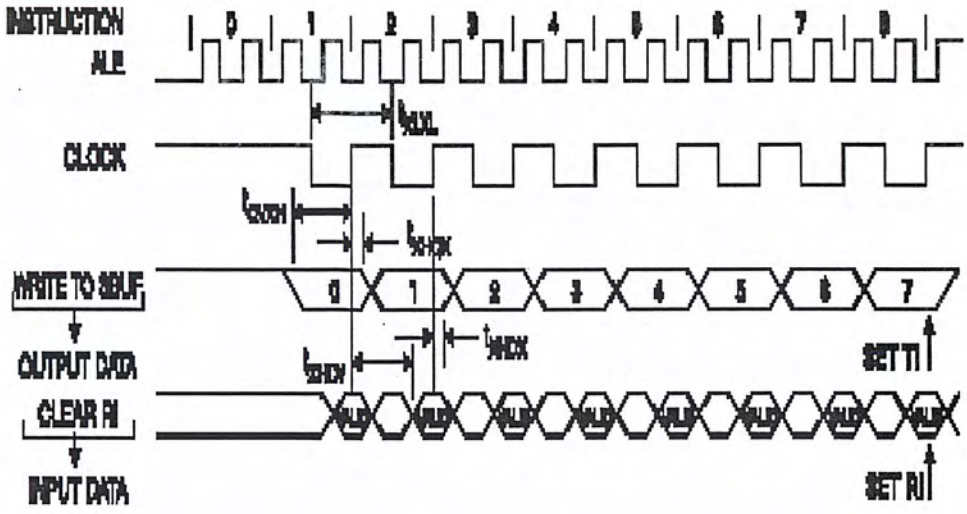
Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 20 mA
Maximum total I_{OL} for all output pins: 80 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

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

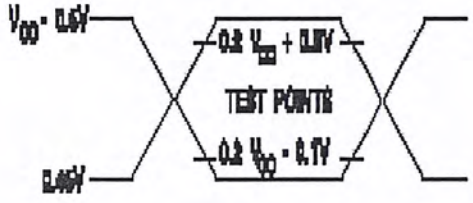
External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	12	0	24	MHz
t_{CLCL}	Clock Period	83.3		41.6		ns
t_{CHCX}	High Time	30		15		ns
t_{CLCX}	Low Time	30		15		ns
t_{CLCH}	Rise Time		20		20	ns
t_{CHCL}	Fall Time		20		20	ns

Shift Register Mode Timing Waveforms

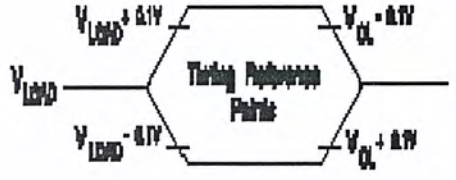


AC Testing Input/Output Waveforms[1]



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

Float Waveforms[1]



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

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	2.7V to 6.0V	AT89C2051-12PC AT89C2051-12SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-12PI AT89C2051-12SI	20P3 20S	Industrial (-40°C to 85°C)
		AT89C2051-12PA AT89C2051-12SA	20P3 20S	Automotive (-40°C to 105°C)
24	4.0V to 6.0V	AT89C2051-24PC AT89C2051-24SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C2051-24PI AT89C2051-24SI	20P3 20S	Industrial (-40°C to 85°C)

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