ปริญญานิพนธ์

เครื่องป้อนอาหาร(ปลา)อัตโนมัติ
AUTOMATIC FEEDING CONTROLLER

นาย สำเนียง พระภักดี 38013343
นาย สุรพงษ์ ทั่งทอง 38013347

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาอุตสาหกรรมศาสตร์บัณฑิต
สาขาเทคโนโลยีอิเล็คทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2540

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY

LADKRABANG

PROJECT

AUTOMATIC FEEDING CONTROLLER

PROJECT REPORT SUBMITTED IN PARTIAL FULFULIMENT

OF THE REQUIREMENTS FOR THE BACHELOR'S DEGREE

DEPARTMENT OF INDUSTIAL TECHNOLOGY

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

1997

หัวข้อปริญญานิพนธ์ เครื่องป้อนอาหาร(ปลา)อัตโนมัติ

AUTOMATIC FEEDING CONTROLLER

ชื่อนักศึกษา นาย สำเนียง พระภักดี

นาย สุรพงษ์ ทั่งทอง

อาจารย์ที่ปรึกษา อาจารย์ไพศาล สิทธิโยภาสกุล

อาจารย์มนชนก ศรีเสือขาม

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2540

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ ฉบับนี้เป็นส่วนหนึ่งของ การศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

.....

คณะกรรมการสอบปริญญานิพนธ์

..... อาจารย์ที่ปรึกษา

( )

..... กรรมการ

( )

..... กรรมการ

( )

..... กรรมการ

( )

..... กรรมการ

( )

เครื่องป้อนอาหารปลาอัตโนมัติ

AUTOMATIC FEEDING CONTROLLER

นาย สำเนียง พระภักดี รหัส 38013343
นาย สุรพงษ์ ทั่งทอง รหัส 38013347

อาจารย์ที่ปรึกษา
อาจารย์ไพศาล สุทธิโยภาสกุล
อาจารย์มนชนก ศรีเสือขาม

ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้นำเสนอระบบการป้อนอาหารปลาอัตโนมัติ โดยอาศัยการควบคุมการทำงานโดยซีพียูไมโครคอนโทรลเลอร์ตระกูล MCS51 ในการตั้งเวลาเปิดปิดกลไกป้อนอาหาร และสามารถทำการเปลี่ยนแปลงเวลาการเปิดปิดได้โดยผู้ใช้ ซึ่งการใช้ภาษาแอสเซมบลีเป็นตัวสร้างโปรแกรมขึ้นมา ซึ่งสามารถที่จะนำไปประยุกต์ใช้ในการควบคุมการป้อนอาหารสัตว์ชนิดอื่นๆได้อย่างกว้างขวาง

AUTOMATIC FEEDING CONTROLLER

MR. SAMNIEANG PRAPAKDEE 38013343

MR. SURAPONG THUNGTHONG 38013347

ADVISER

MR. PAISARN SITIYOPHASAKUL

MISS MONCHANOK SRISUAKHAM

ABSTRACT

Decoding Automatic Feeding Controller have been presented in thesis by being controlled by Microcontroller CPU series MCS51 for feeding open-closed time,and time can be changed by user. Which assembly language has been used for building programm in CPU, which can be applied to build feeding controller for various animals.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้มิอาจสำเร็จลุล่วงไปได้หากขาดความช่วยเหลือจากบุคคลต่างๆ ในหลายๆฝ่าย ดังนั้นทางกลุ่มผู้จัดทำจึงใคร่ขอขอบพระคุณบุคคลต่างๆดังต่อไปนี้

ขอบพระคุณ อาจารย์ไพศาล สิทธิโยภาสกุล และ อาจารย์มนชนก ศรีเสือขาม ที่กรุณาให้คำชี้แนะและเป็นที่ปรึกษาในคอนเซ็ปต์ และรายละเอียดต่างๆ ขอบพระคุณคณะอาจารย์โรงเรียนนายเรืออากาศหลายๆท่าน ที่กรุณาให้คำชี้แนะนำหนังสือและข้อมูลบางอย่าง ขอบคุณพี่ๆน้องๆที่กองปรับเทียบมาตรฐานเครื่องวัด กรมสื่อสารทหารอากาศที่คอยให้กำลังใจ และให้คำปรึกษาหลายอย่าง

สุดท้าย ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ และคุณครูบาอาจารย์ตั้งแต่เล็กจนโต ที่เลี้ยงดู สั่งสอนให้ข้าพเจ้าเติบโตขึ้นมา จนสำเร็จในสิ่งที่ดังหวังในชีวิต ฉนั้น คุณความดีใดๆก็ตามที่เกิดจากปริญญานิพนธ์ฉบับนี้ ขอมอบแด่ทุกๆท่านที่เอ่ยมาในข้างต้นนี้ เทอญ

คำนำ

โครงงานเรื่อง เครื่องป้อนอาหารปลาอัตโนมัตินี้ จัดทำขึ้นเพื่อประโยชน์ในด้านความสะดวกสบายในการเลี้ยงปลา โดยเฉพาะปลาที่เลี้ยงไว้ดูเล่นซึ่งปัจจุบันเป็นที่นิยมกันมาก แต่ปัญหาส่วนหนึ่งของผู้คิดจะเลี้ยงปลาก็คือ มักจะเกรงกันว่าตัวเองจะไม่มีเวลาคอยให้อาหารตามเวลา เมื่อต้องออกธุระไปนานๆ และสำหรับร้านจำหน่ายพันธุ์ปลาที่มีปลาจำนวนมากๆ ที่ต้องดูแล ซึ่งในบางครั้งการดูแลให้อาหารครบถ้วนทั้งปริมาณและเวลาที่เหมาะสมเป็นปัญหาที่ต้องเสียเวลามาก หรือบางครั้งหลงลืมกันบ่อยๆ โครงงานนี้จึงสามารถช่วยแก้ปัญหาตรงนี้ได้

หัวใจของโครงงานนี้อยู่ที่ตัวควบคุมระบบคือ ไมโครคอนโทรลเลอร์ตระกูล MSC-51 ซึ่งบันทึกโปรแกรมไว้ทำให้ผู้ใช้สามารถที่จะเวลาที่เหมาะสม สำหรับการกินอาหารของปลาแต่ละชนิดได้ รวมทั้งมีระบบเตือนเมื่ออาหารใกล้หมดแล้ว และภายในเครื่องยังออกแบบสวิทช์ต่างๆ ไว้ เพื่อตรวจเช็คการทำงานของวงจร กรณีสงสัยว่ากลไกจะขัดข้องด้วย รายละเอียดต่างๆ นั้นมีอธิบายอยู่ภายในเอกสารปริญญานิพนธ์นี้แล้ว

ขอแสดงความนับถือ

นาย สำเนียง พระภักดี

นาย สุรพงษ์ ทั่งทอง

สารบัญ

บทที่ 1
บทนำ

เครื่องป้อนอาหาร(ปลา)อัตโนมัตินี้ เหตุที่วงเล็บ"ปลา"ไว้ ก็เนื่องจากเครื่องมือนี้สามารถดัดแปลงเพื่อใช้ป้อนอาหารสัตว์ชนิดอื่นที่ต้องการอาหารใกล้เคียงกัน เช่น นก,สัตว์น้ำบางจำพวก เป็นต้น ปัจจุบันนี้การเลี้ยงสัตว์เกิดขึ้นมากในหมู่คนทั่วไป แต่เครื่องอำนวยความรับผิดชอบต่อการให้อาหารยังไม่มีการผลิตออกมาเป็นเชิงพาณิชย์เลย จึงเป็นแรงบันดาลใจให้ผู้จัดทำสร้างขึ้นมา

หัวใจสำคัญของโครงงานนี้คือ ไมโครคอนโทรลเลอร์ตระกูล MCS51 ซึ่งง่ายต่อการพัฒนาโปรแกรมแต่เข้าถึงพอร์ตอินพุต/เอาท์พุต ทั้งยังเป็นที่แพร่หลายในหมู่นักประดิษฐ์รุ่นใหม่ โดยใช้ภาษาแอสเซมบลีเป็นตัวเขียนโปรแกรม อีกทั้งเพื่อสะดวกในการใช้ของผู้ใช้ ผู้จัดทำจึงได้ออกแบบโครงงานเพื่อให้ผู้ใช้ทั่วไปสามารถทำการเปลี่ยนแปลงการตั้งเวลาได้ง่ายๆตามที่ต้องการ การออกแบบจึงเน้นหนักออกเป็น 2 ส่วน คือ ส่วนของอิเล็คทรอนิกส์ซึ่งเป็นตัวควบคุมและเก็บข้อมูลในหน่วยความจำ กับอีกในส่วนของภาคกลไกเพื่อนำคำสั่งมาปฏิบัติในเชิงกล

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

แม่เหล็กไฟฟ้าและการนำไปใช้งาน

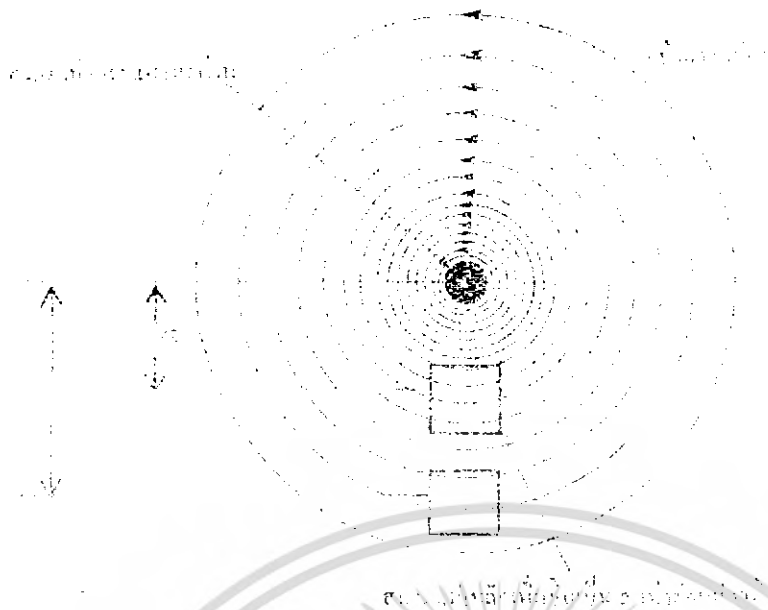
บทนำ

ความเกี่ยวข้องระหว่างไฟฟ้ากับแม่เหล็กถูกค้นพบเมื่อ พ.ศ.2367 โดยนักวิทยาศาสตร์ที่ชื่อ เออร์สเตด เขาได้ทำการทดลองและพบว่ากระแสที่ไหลในเส้นลวดสามารถทำให้เข็มทิศแม่เหล็กเบี่ยงเบนได้ อีก 2-3 ปีต่อมาผลที่เกิดขึ้นตรงกันข้ามก็ถูกค้นพบ การเปลี่ยนแปลงของสนามแม่เหล็กจะทำให้อิเล็คตรอนเคลื่อนที่ซึ่งจะทำให้เกิดการไหลของกระแสไฟฟ้า ผู้ที่ได้ทำการศึกษาผลของอันนี้ได้แก่ ฟาราเดย์, เฮนรี และ เลนซ์ ฉะนั้นจึงกล่าวได้ว่าแม่เหล็กไฟฟ้าจะรวมถึงอำนาจแม่เหล็กที่เกิดจากการไหลของกระแสไฟฟ้าด้วย ผลที่เกิดจากแม่เหล็กไฟฟ้าได้ถูกนำไปใช้งานอย่างกว้างขวาง อาทิ เช่น มอเตอร์ เจนเนอเรเตอร์ และตัวเหนี่ยวนำในวงจรอิเล็คทรอนิกส์

การเกิดสนามแม่เหล็กรอบตัวเหนี่ยวนำที่มีกระแสไหลผ่าน

ในรูปที่ 1 แสดงการวางตัวของผงคะไบเหล็กเป็นวงรอบตัวนำที่มีกระแสไฟฟ้าไหลผ่าน ผงคะไบเหล็กจะเรียงตัวกันอย่างเหนียวแน่นในบริเวณที่อยู่ใกล้ลวดตัวนำ ซึ่งแสดงว่าตำแหน่งนี้สนามแม่เหล็กมีความเข้มมากที่สุด ความเข้มของสนามแม่เหล็กจะลดลงเป็นสัดส่วนกำลังสองของระยะทางที่ห่างจากตัวนำ กล่าวคือความเข้มที่ระยะหนึ่งนิ้ววัดจากตัวนำจะมีความเข้มเป็นครึ่งหนึ่งของความเข้มที่ระยะครึ่งนิ้วเป็นต้น

รูปที่ 1 การเรียงตัวของผงคะไบเหล็กรอบตัวนำที่มีกระแสไหลผ่าน



รูปที่ 2 แสดงความเข้มของสนามแม่เหล็กรอบตัวนำทีนำกระแส

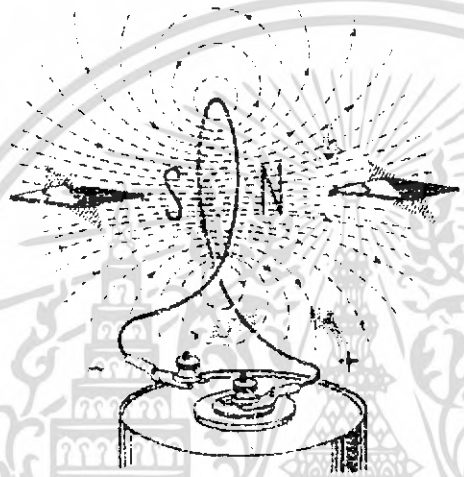
สนามแม่เหล็กที่มีเส้นแรงทิศตามและทวนเข็มนาฬิกา

เนื่องจากเส้นแรงแม่เหล็กเป็นวงกลม เมื่อนำแม่เหล็กมาวางในสนามแม่เหล็กๆ จะพยายามบังคับให้แท่งแม่เหล็กเบนขั้วที่เป็นขั้วเหนือให้อยู่ในแนวของเส้นรอบวง ซึ่งจะทวนเข็มหรือตามเข็ม เราสามารถทดลองได้ดังรูปที่ 3 การหาทิศทางทำได้โดยการใช้เข็มทิศ เมื่อเข็มทิศอยู่ทางด้านหน้าของของเส้นลวดขั้วเหนือจะชี้ขึ้น และในด้านตรงข้ามจะชี้ลง ถ้าหากวางเข็มทิศไว้ที่ด้านบนของเส้นลวดขั้วเหนือจะชี้ตรงไปที่ด้านหลังของเส้นลวด(พุ่งตั้งฉากเข้าหาหน้ากระดาษ) และถ้าวางเข็มทิศใต้เส้นลวดขั้วเหนือจะชี้มาทางด้านหน้า เมื่อรวบรวมทิศทางการวางตัวของเข็มทิศรอบเส้นลวดจะได้เส้นแรงแม่เหล็กเป็นวงกลม มีทิศทางทวนเข็มนาฬิกาตามรูป ถ้าหากกระแสไหลกลับทิศทางเส้นแรงแม่เหล็กจะมีทิศตามเข็มนาฬิกา

รูปที่ 3 วิธีหาทิศทางของสนามแม่เหล็กรอบตัวนำโดยใช้เข็มทิศ

ขั้วแม่เหล็กของขดลวด

เมื่อนำเส้นลวดตัวนำมาดัดให้เป็นวง วงลวดนี้เราเรียกว่าขดลวด(คอยน์) เมื่อนำมาต่อกับแบตเตอรี่ ดังรูปที่ 4 เส้นแรงแม่เหล็กที่อยู่ภายในวงของขดลวดจะกระจุกกันอย่างหนาแน่น โดยที่จำนวนเส้นแรงทั้งหมดจะยังคงเท่ากับในกรณีที่ยืดออกเป็นเส้นตรง เพียงแต่ในกรณีที่ขดลวดให้เป็นวงเส้นแรงแม่เหล็กจะกระจุกอยู่ในที่ว่างที่เล็กกว่าเท่านั้น นอกจากนั้นเส้นแรงแม่เหล็กในวงจะเสริมกันในทิศทางเดียวกัน ลักษณะของสนามแม่เหล็กเช่นนี้จะเหมือนกับสนามแม่เหล็กของแท่งแม่เหล็ก โดยที่ขั้วแม่เหล็กต่างชนิดกันจะอยู่ที่คนละด้านของวงขดลวด



รูปที่ 4 แสดงการเกิดขั้วแม่เหล็กจากกระแสที่ไหลผ่านวงขดลวด

ขดลวดโซลินอยด์

ถ้าหากขดลวดมีมากกว่า 1 รอบ ชุดขดลวดนี้เราเรียกว่าโซลินอยด์ ในทางทฤษฎีนั้นจะกำหนดว่า ความยาวของโซลินอยด์จะต้องยาวกว่าเส้นผ่าศูนย์กลางมากๆ เส้นแรงแม่เหล็กหนาแน่นภายในแนวง และปลายทั้งสองจะเกิดขั้วแม่เหล็กต่างชนิดกันโดยที่จะมากเป็นทวีคูณตามจำนวนรอบของขดลวดของโซลินอยด์ เส้นแรงแม่เหล็กของสนามแม่เหล็กไฟฟ้าจะเหมือนกับเส้นแรงแม่เหล็กของแท่งแม่เหล็กของแท่งแม่เหล็กที่เริ่มจากขั้วเหนือไปยังขั้วใต้ดังแสดงในรูปที่ ๕

รูปที่ 5 ขั้วแม่เหล็กของโซลินอยด

การเหนี่ยวนำกระแส

เมื่อให้เส้นแรงแม่เหล็กเคลื่อนที่ตัดลวดตัวนำแล้วจะทำให้อิเล็คตรอนในลวดตัวนำเคลื่อนที่ กรรมวิธีนี้เราเรียกว่า การเหนี่ยวนำ (Induction) เหตุที่เรียกดังนี้เพราะไม่มีสิ่งเชื่อมโยงเป็นตัวตนระหว่างแท่งแม่เหล็กกับตัวนำ กระแสเหนี่ยวนำที่เกิดขึ้นเป็นผลจากการใส่พลังงานกลเข้าไปโดยการทำให้สนามแม่เหล็กเคลื่อนที่ พลังงานกลนี้จะถูกเปลี่ยนเป็นพลังงานไฟฟ้าอีกทีหนึ่งในรูปของการไหลของกระแส

จากรูปที่ 6 เราให้ลวดตัวนำ กข วางตั้งฉากกับเส้นแรงแม่เหล็กที่อยู่ระหว่างขั้วแม่เหล็กของแม่เหล็กเกือกม้า เมื่อแม่เหล็กเคลื่อนที่ขึ้นหรือลง เส้นแรงแม่เหล็กจะตัดลวดตัวนำ ผลจากการที่เส้นแรงแม่เหล็กจะตัดลวดตัวนำจะทำให้เกิดกระแสไหล ซึ่งเราเรียกว่าการเกิดปรากฏการณ์เช่นนี้จริงโดยดูจากการกระดิกของเข็มของไมโครมิเตอร์ที่นำไปต่อคร่อมลวดตัวนำ เมื่อให้แม่เหล็กเคลื่อนที่ลงกระแสจะไหลในทิศทางดังรูป แต่ถ้าให้แม่เหล็กเคลื่อนที่ขึ้น กระแสจะไหลในทิศทางตรงกันข้าม ถ้าไม่มีการเคลื่อนที่จะไม่มีกระแสไหลผิดขนาด

กฎของเลนซ์

กฎของเลนซ์นี้ตั้งขึ้นเพื่อใช้หาทิศทางของแรงดันหรือกระแสเหนี่ยวนำโดยถือหลักว่า พลังงานจะไม่มีการสูญหายไปไหน ซึ่งกฎนี้กล่าวเป็นใจความว่า ทิศทางของกระแสเหนี่ยวนำที่เกิดขึ้นจะอยู่ในลักษณะที่สนามแม่เหล็กไฟฟ้า ซึ่งเกิดจากกระแสเหนี่ยวนำพยายามต่อต้านการกระทำที่ทำให้เกิดกระแสเหนี่ยวนำ ดังรูปที่ 7 ประกอบด้วยขดลวดมีไมโครมิเตอร์ต่อคร่อมที่ปลายทั้งสอง เมื่อพุ่งขั้วเหนือของแท่งแม่เหล็กเข้าหาปลายขดลวดทางด้านซ้าย กระแสเหนี่ยวนำที่เกิดขึ้นจะไหลในทิศทางที่ทำให้เกิดขั้วเหนือที่ปลายขดลวดทางด้านซ้าย นี่เป็นเหตุผลว่าทำไมเราจึงต้องออกแรงพุ่งแท่งแม่เหล็กเข้าหาขดลวด และพลังงานกลที่เราใช้ไปนี้จะถูกเปลี่ยนไปเป็นพลังงานไฟฟ้าในรูปของการเหนี่ยวนำให้เกิดกระแสไหลในขดลวด

โดยการใช้กฎเรารู้ว่าปลายขดลวดด้านซ้ายในรูปที่ 7 ต้องเป็นขั้วเหนือเพื่อต่อต้านการพุ่งขั้วเหนือของแท่งแม่เหล็กเข้าหา ดังนั้นเราสามารถหาทิศทางของกระแสเหนี่ยวนำที่ไหลในขดลวดโดยใช้กฎมือขวา ในทางกลับกันถ้าหากดึงขั้วเหนือของแท่งแม่เหล็กเคลื่อนที่ออกไป การที่จะเกิดขั้วใต้ที่ปลายขดลวดด้านซ้านนั้นแสดงว่ากระแสเหนี่ยวนำจะต้องไหลกลับทิศทาง ลักษณะของการเคลื่อนที่เข้าออกของแท่งแม่เหล็กแล้วทำให้กระแสเหนี่ยวนำที่เกิดขึ้นไหลกลับไปกลับมา วิธีการอย่างนี้เป็นหลักการผลิตกระแสไฟฟ้าสลับที่ใช้กันตามบ้านเรือนทั่วๆไป

กฎการเคลื่อนที่ของฟาราเดย์

แรงดันเหนี่ยวนำที่เนื่องจากเส้นแรงแม่เหล็กเคลื่อนที่ตัดขดลวดนั้น จะมากหรือน้อยขึ้นอยู่กับหัวข้อดังต่อไปนี้

1. ความแรงของสนามแม่เหล็ก สนามแม่เหล็กที่ตัดลวดตัวนำแรงขึ้น ค่าแรงดันเหนี่ยวนำที่เกิดขึ้นมีค่ามากขึ้น

2. จำนวนรอบของขดลวด ลวดมากรอบแรงดันเหนี่ยวนำจะมากเนื่องจากแรงดันเหนี่ยวนำทั้งหมด

3. อัตราเร็วที่เส้นแรงแม่เหล็กตัดขดลวด ถ้าเส้นแรงแม่เหล็กตัดลวดตัวนำเร็วขึ้น แรงดันเหนี่ยวนำก็จะสูงขึ้นหรือพูดได้อีกทางว่า ในช่วงเวลาหนึ่งจำนวนเส้นแรงแม่เหล็กที่ตัดขดลวดมีมากขึ้น แรงดันเหนี่ยวนำก็จะมากขึ้นตามไปด้วย

จากหลักการทั้ง 3 ข้อนี้ สามารถเขียนเป็นสมการเป็นกฎของฟาราเดย์ได้ดังนี้

แรงดันเหนี่ยวนำ = จำนวนรอบ x อัตราเร็วเส้นแรงแม่เหล็กตัดลวดตัวนำในหนึ่งหน่วยเวลา เมื่อแรงดันมีหน่วยเป็นโวลท์

จำนวนรอบมีหน่วยเป็นรอบ

อัตราเร็วที่เส้นแรงตัดขดลวดเป็นเวเบอร์ต่อวินาที

การนำไปใช้งาน

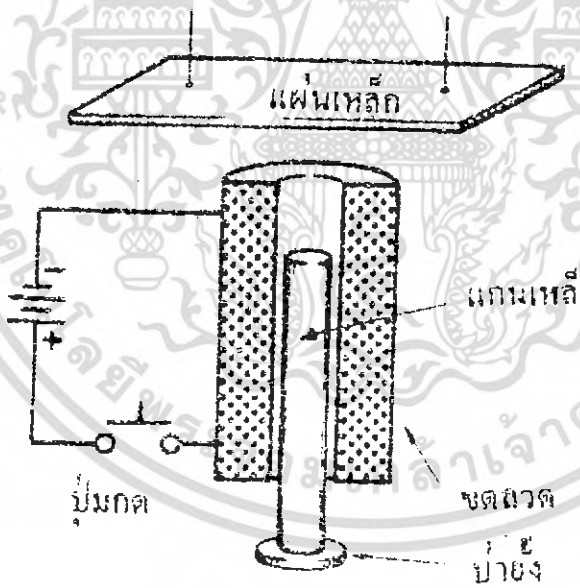
ในที่นี้จะกล่าวถึงหลักการทางแม่เหล็กไฟฟ้าไปใช้สร้างอุปกรณ์ทางไฟฟ้าพอสังเขป ที่ผู้จัดทำได้นำอุกรณ์ไฟฟ้าดังกล่าวมาเป็นส่วนประกอบของโครงงาน คือ โซลินอยด์และรีเลย์

โซลินอยด์

รูปที่ 8 เส้นแรงแม่เหล็กดูดแกนเหล็กเข้าไปในขดลวด (รูป ก.) และแกนเหล็กหยุดนิ่งที่ตรงกลางขดลวด(รูป ข.)

โซลินอยด์(Solinoild) ประกอบด้วยขดลวดและแกนเหล็กซึ่งเคลื่อนที่ได้ สอดอยู่ในรูตรงกลางของขดลวด ดังในรูปที่ 8 เมื่อต่อแรงดันไฟให้กับขดลวดของโซลินอยด์ จะเกิดเส้นแรงแม่เหล็กขึ้น ดังแสดงด้วยเส้นประ เส้นแรงแม่เหล็กจะพยายามวางแนวเส้นทางเดินระหว่างขั้วแม่เหล็กที่สั้นที่สุดเท่าที่จะเป็นไปได้ เส้นแรงแม่เหล็กจะทำให้เกิดแรงดึงดูดบนแกนแม่เหล็กที่เคลื่อนที่ได้ และดึงมันเข้าไปอยู่ตรงกลางขดลวดทำให้สปริงที่ต่ออยู่ทางอีกด้านหนึ่งจะถูกยืดออก

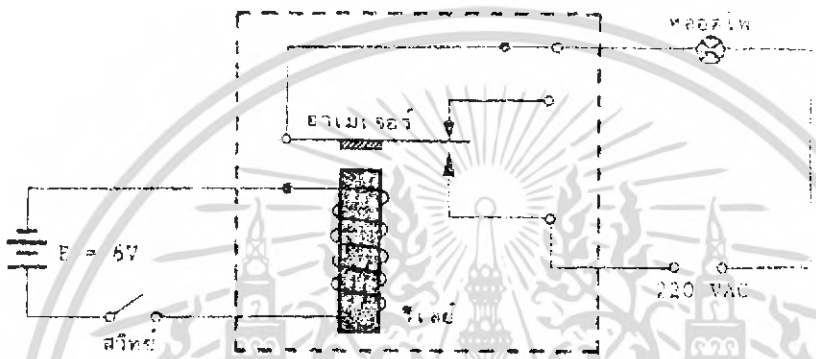
เมื่อหยุดป้อนกระแสเข้าขดลวด แรงดึงดูดเนื่องจากอำนาจแม่เหล็กจะหมดไป สปริงจะดึงแกนเหล็กให้กลับมาอยู่ตำแหน่งเดิมดังรูป 8 ก. จากตัวอย่างอันนี้เป็นวิธีการที่นิยมกันมากในการเปลี่ยนพลังงานไฟฟ้าเป็นพลังงานกล(มีการเคลื่อนที่) โดยผ่านทางแม่เหล็กไฟฟ้า ในการใช้งานด้านอื่นๆอาจมีก้านหรือคานมาต่อแกนเหล็กที่เคลื่อนที่ได้ โซลินอยด์มีที่ใช้งาน เช่น ใช้เป็นตัวบังคับการปิดเปิดของท่อน้ำในเครื่องซักผ้า หรือใช้ทำกริ่งไฟฟ้าดังรูปที่ 9 ปุ่มกดจะอยู่ที่ประตู และตัวเคาะซึ่งประกอบด้วยโซลินอยด์กับแผ่นเหล็กจะอยู่ในบ้าน เมื่อกดปุ่มแกนเหล็กจะถูกดูดและเคาะแผ่นเหล็กทำให้เกิดเสียง



รูปที่ 9 รูปแสดงโครงสร้างของกริ่งเคาะเรียกประตู

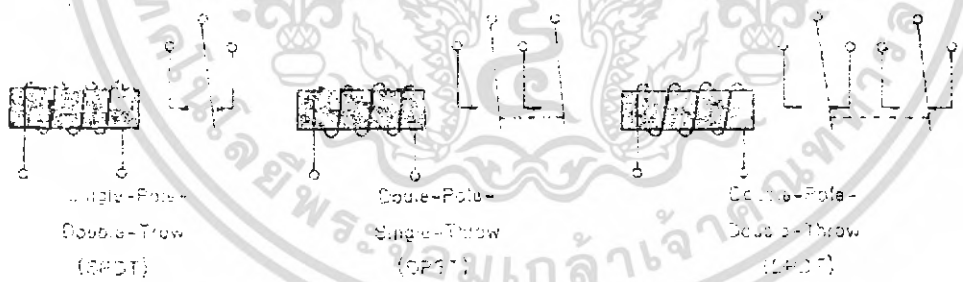
รีเลย์

รีเลย์ (Relay) คือ สวิทช์แม่เหล็กดังแสดงในรูปที่ 10 ประกอบด้วยอาร์เมเจอร์ลักษณะเป็นแผ่นสปริง ปลายข้างหนึ่งตรึงอยู่กับที่อยู่เหนือแกนเหล็กของแม่เหล็กไฟฟ้า เมื่อปิดสวิทช์มีกระแสไหลเข้าขดลวดแม่เหล็กไฟฟ้าจะดูดอาร์เมเจอร์ หน้าสัมผัสจะเปิดหรือปิด ขึ้นอยู่กับลักษณะการจัดหน้าสัมผัสที่อยู่ในตำแหน่ง ปกติปิด (NC) จะเปิดก็ต่อเมื่อแม่เหล็กไฟฟ้าดูดอาร์เมเจอร์ หน้าสัมผัสจะอยู่ในตำแหน่ง ปกติเปิด (NO) จะปิดเมื่อแม่เหล็กไฟฟ้าดูดอาร์เมเจอร์ จากรูป เมื่อเปิดสวิทช์แม่เหล็กไฟฟ้าจะปล่อยอาร์เมเจอร์จะดีดตัวเองด้วยแรงสปริงกลับไปอยู่ที่ตำแหน่งเดิม



รูปที่ 10

รีเลย์ทั่วๆไปจะมีการจัดหน้าสัมผัสและลักษณะการปิดเปิดหน้าสัมผัสดังรูป 11 มีแบบ SPDT, DPST และ DPDT



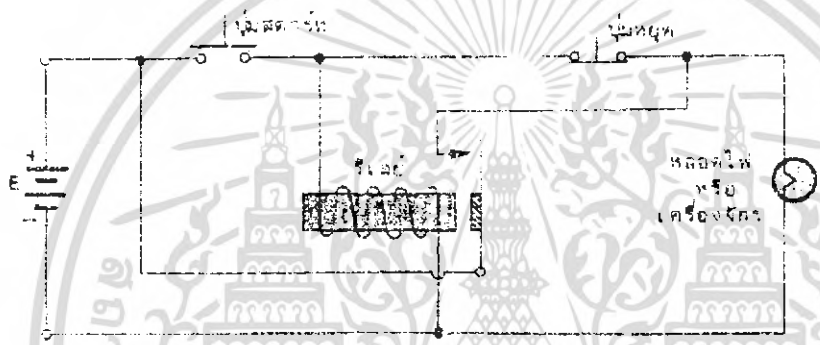
รูปที่ 11

รีเลย์ที่ใช้งานในปัจจุบันมีทั้งแบบใช้ไฟตรงและไฟสลับ รีเลย์แบบไฟสลับมีอุปกรณ์เพิ่มเติมมาบางชิ้นจะไม่ขอกล่าวในที่นี้ เพราะจำเป็นจะต้องกล่าวถึงทฤษฎีทางกระสลับเสียก่อน สำหรับลักษณะการจัดหน้าสัมผัสก็ยังคงเหมือนเดิม

ข้อดีของรีเลย์นั้น คือ เราสามารถใช้กระแสเพียงเล็กน้อยควบคุมวงจรที่มีกระแสไหลมากๆได้ หรือใช้แรงดันต่ำๆควบคุมวงจรที่ใช้แรงดันไฟฟ้าสูงซึ่งอาจเป็นอัตรายต่อชีวิตได้ง่าย ผู้ใช้งานสามารถควบคุมการทำงานในระยะไกลได้

ในการเลือกใช้รีเลย์ ควรพิจารณาข้อกำหนดของขดลวด เช่น ความต้านทานทางตรงของขดลวด ซึ่งหมายถึงความต้านทานของลวดที่นำมาใช้พันขดลวด กระแสในขณะที่ทำงานหรือแรงดันที่ป้อนให้กับขดลวด เมื่อแม่เหล็กดูดอาร์เมเจอร์ ข้อกำหนดเหล่านี้จะบอกว่ารีเลย์มีความไวขนาดไหน ผู้ใช้ต้องเลือกให้เหมาะสม และต้องพิจารณาให้หน้าสัมผัสสามารถทนกระแสที่ต้องการให้ไหลได้

การต่อรีเลย์วิธีหนึ่งที่เป็นที่นิยมมาก คือ รีเลย์สามารถยึดตัวมันเองได้โดยการใช้ควบคู่กับสวิทช์ ชนิดเมื่อไม่ได้ออกแรงกดมันจะคืนสู่สภาพเดิม สวิทช์ที่กล่าวถึงนี้มีทั้งแบบ NO(Normal Open) ซึ่งมักใช้เป็นปุ่มกดเริ่มทำงาน และปุ่ม NC(Normal Closed) ซึ่งนิยมใช้เป็นปุ่มหยุด ดังแสดงในรูปที่ 12และ13

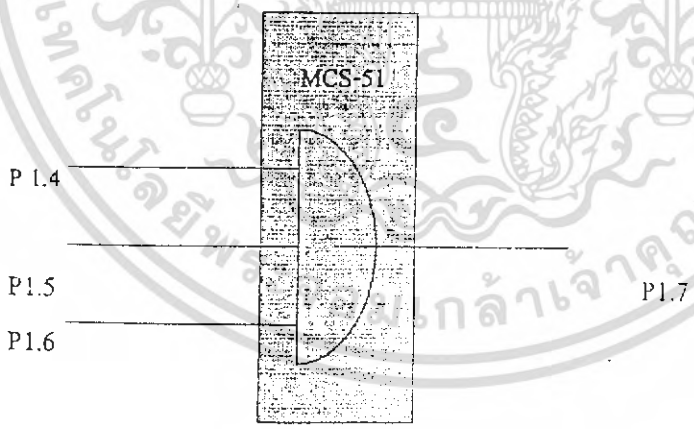


รูปที่ 12 รีเลย์ที่สามารถยึดตัวเองได้ ขณะอยู่ตำแหน่ง Off หรืออาร์เมเจอร์ไม่ถูกดูดให้หน้าสัมผัสปิด

รูปที่ 13 ขณะอยู่ในตำแหน่ง ON อาร์เมเจอร์ถูกดูดหน้าสัมผัสปิด

ไมโครคอนโทรลเลอร์ MCS-51

ถ้านำ CPU เบอร์ Z-80 มาประกอบเป็นคอมพิวเตอร์จะต้องนำไอซี หน่วยความจำ, พอร์ตมาประกอบเป็นระบบ แต่ถ้าเป็นไมโครคอนโทรลเลอร์ภายในชิพไอซีจะมีหน่วยความจำ, พอร์ตประกอบอยู่ในไอซีเพียงตัวเดียว ซึ่งอาจเรียกได้ว่าเป็นคอมพิวเตอร์เดี่ยวหรืออาจมองง่ายๆว่า ถ้ามีไอซีไมโครคอนโทรลเลอร์เราสามารถสร้างเป็นระบบคอมพิวเตอร์ได้เลย เพียงแต่ป้อนแหล่งจ่ายไฟและสัญญาณนาฬิกาเข้าไปเท่านั้น ถ้าพิจารณาไมโครคอนโทรลเลอร์ 8051 เราสามารถลองใช้งานง่ายๆ ได้เช่น ถ้าจะนำไมโครคอนโทรลเลอร์เบอร์ตระกูล MCS-51 สร้างเป็น NAND Gate 3 อินพุต ดังรูปที่ 1 โดยอ่านค่าจากพอร์ตอินพุตเข้ามา 3 บิต ถ้าทุกบิตเป็น "1" หมดให้เอาท์พุตบิต P1.7 เป็นลอจิก "0" เราสามารถใช้ MCS-51 เป็นพอร์ตได้เลย เพราะมีพอร์ตในตัวอยู่แล้ว จากรูปที่ 1 สามารถเขียนไดอะแกรมการทำงานได้ดังรูปที่ 2



รูปที่ 1 ตัวอย่างการใช้ MCS-51

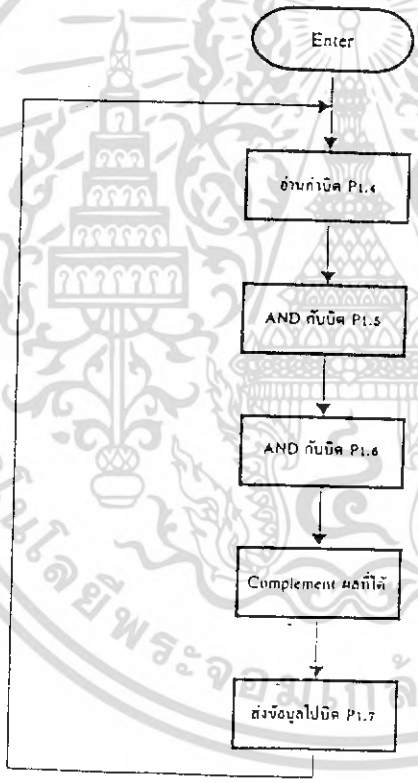
จากโปรแกรมและ Flowchart จะเห็นว่าถ้าจะใช้งาน 8051 เป็นแนนเกต จะทำได้โดยอ่านค่าจากบิตต่างๆ และทำการแอนด์กัน จากนั้นกลับค่าพร้อมกับส่งออกทางบิต P1.7 ซึ่งที่ขาบิตต่างๆ สามารถใช้งานเป็นอินพุตและเอาท์พุตได้เลย แต่ถ้าหากใช้ Z-80 จะต้องสร้างพอร์ตอินพุตและเอาท์พุตขึ้นมาประกอบด้วย

1. โครงสร้างของ MCS-51

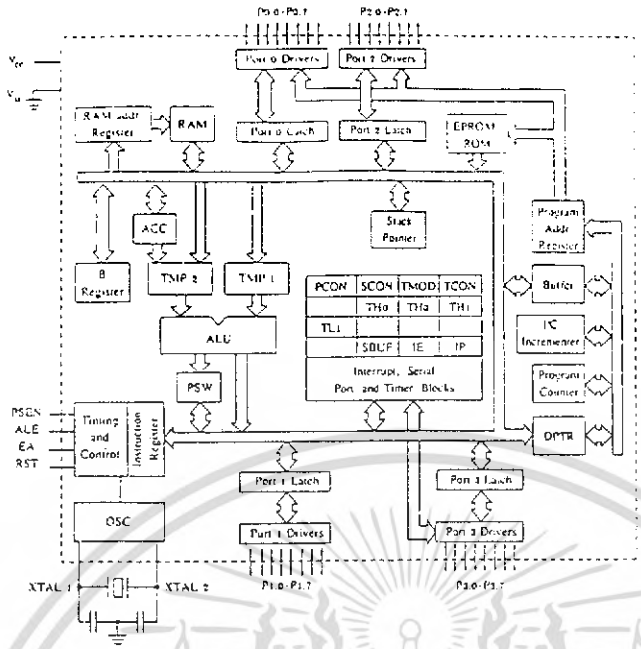
ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีด้วยกันหลายเบอร์ขึ้นอยู่กับโครงสร้างภายในของมัน บางเบอร์จะมีหน่วยความจำภายในแบบรอม บางเบอร์เป็นแบบอีพรอม บางเบอร์มีแรมภายใน 128 ไบต์ บางเบอร์มี 256 ไบต์ เป็นต้น ซึ่งรายละเอียดจะศึกษาได้จากคู่มือของมันโดยตรง และลักษณะของขาต่างๆจะเหมือนกัน คุณสมบัติที่สำคัญของ MCS-51 มีดังนี้

- มีหน่วยความจำ ROM 4K bytes
- มีหน่วยความจำ RAM 128K bytes
- มีพอร์ต I/O ขนาด 8 บิต 4 พอร์ต
- มีไทม์เมอร์ 16 บิต 2 ตัว
- สามารถอินเตอร์รัพได้ 5 แหล่ง
- มีวงจรออสซิเลเตอร์และวงจรนาฬิกาบนชิพ
- มีพอร์ตอนุกรมที่สมารถรับส่งข้อมูลแบบ Full Duplex ความเร็วสูง
- อ้างหน่วยความจำโปรแกรมภายนอกได้ 64K
- สามารถประมวลผลทีละบิตได้
- สามารถอ้างหน่วยความจำแบบบิตได้ 210 ตำแหน่ง
- หนึ่งวัฏจักรคำสั่งกินเวลาประมาณ 1 ไมโครวินาที ขณะทำงานด้วย Clock 12 MHz

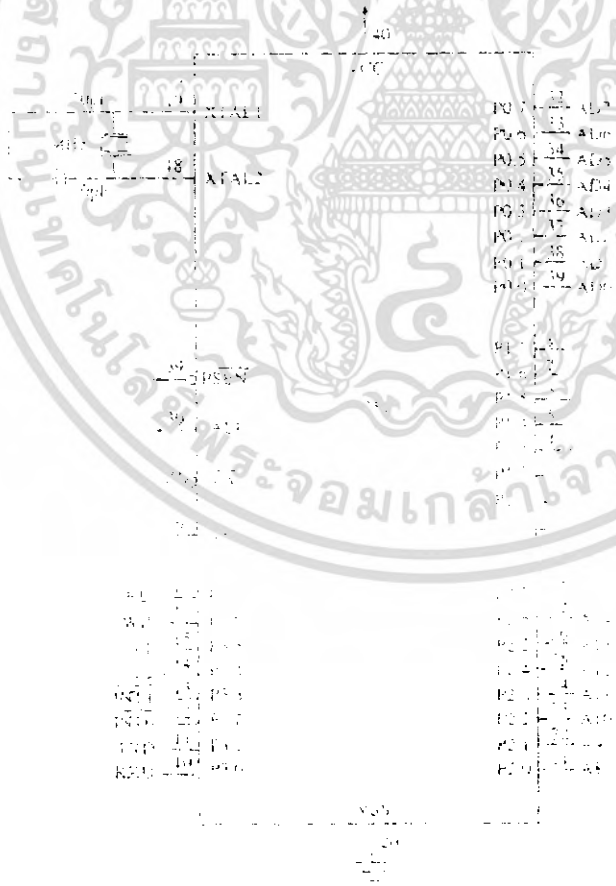
```
LOOP: MOV  C,P1.4  :        อ่านค่าบิต P1.4 เก็บในแฟลก C
      ANL  C,P1.5  :        AND ค่าใน C กับบิต P1.5
      ANL  C,P1.6  :        AND ค่าใน C กับบิต P1.6
      CPL  C       :              สลับค่า C
      MOV  P1.7,C  :        นำค่า C ออกไปที่ P1.7
      SJMP LOOP    :กระโดดกลับ
```



รูปที่ 2 การเขียนโปรแกรมให้ MCS-51 ทำงานเป็น NAND เกต



รูป 3 แสดงโครงสร้างภายในของ MCS51



รูปที่ 4 แสดงขาต่างของไอซี

ความหมายของขาต่างๆ มีดังนี้

1. พอร์ต 0 ได้แก่ขาที่ ได้แก่ขาที่ 32-39 ของ MCS-51 สามารถใช้เป็นอินพุตเอาท์พุตได้นอกจากนี้ ในการติดต่อกับหน่วยความจำภายนอกยังเป็นขาแอดเดรสบัส และบัสข้อมูลอีกด้วย

2. พอร์ต 1 ได้แก่ขาที่ 1-8 เป็นพอร์ท 8 บิต สามารถอ้างทีละบิตได้คือ P1.0,P1.1,...etc

3. พอร์ต 2 ได้แก่ขาที่ 21-28 จะใช้งาน 2 หน้าที่ คือ ใช้เป็นพอร์ต 8 บิตกับใช้เป็นขาแอดเดรส 8 บิต ในการอ้างหน่วยความจำภายนอก

4. พอร์ต 3 ได้แกขาที่ 10-17 จะใช้งาน 2 หน้าทีคือ เป็นพอร์ตอินพุตและเอาท์พุตและ ใช้เป็นขาควบคุมต่างๆดังตารางที่ 2

ตารางที่ 2 แสดงบิตและหน้าที่ต่างๆของพอร์ต 3

บิต	ชื่อ	หน้าที่พิเศษ
P3.0	RXD	ใช้รับข้อมูลทางพอร์ตอนุกรม
P3.1	TXD	ใช้ส่งข้อมูลทางพอร์ตอนุกรม
P3.2	INT0	อินเตอร์รัพภายนอกหมายเลข 0
P3.3	INT1	อินเตอร์รัพภายนอกหมายเลข 1
P3.4	T0	ตัวจับเวลา/ตัวนับตัวที่ 0
P3.5	T1	ตัวจับเวลา/ตัวนับตัวที่ 1
P3.6	WR	สัญญาณเขียนข้อมูลหน่วยความจำภายนอก
P3.7	RD	สัญญาณอ่านข้อมูลหน่วยความจำภายนอก

5. PSEN (Programn Store Enable) ขา PSEN เป็นขาที่ส่งสัญญาณออก คือ ขา 29 ขานี้จะแอคติฟเมื่อ MCS-51 ต้องการอ่าน Code โปรแกรมภายนอก โดยปกติถ้าหน่วยความจำภายนอกเป็นอีพรอม ขา PSEN จะต่อกับขา Output Enable (OE) ของ EPROM

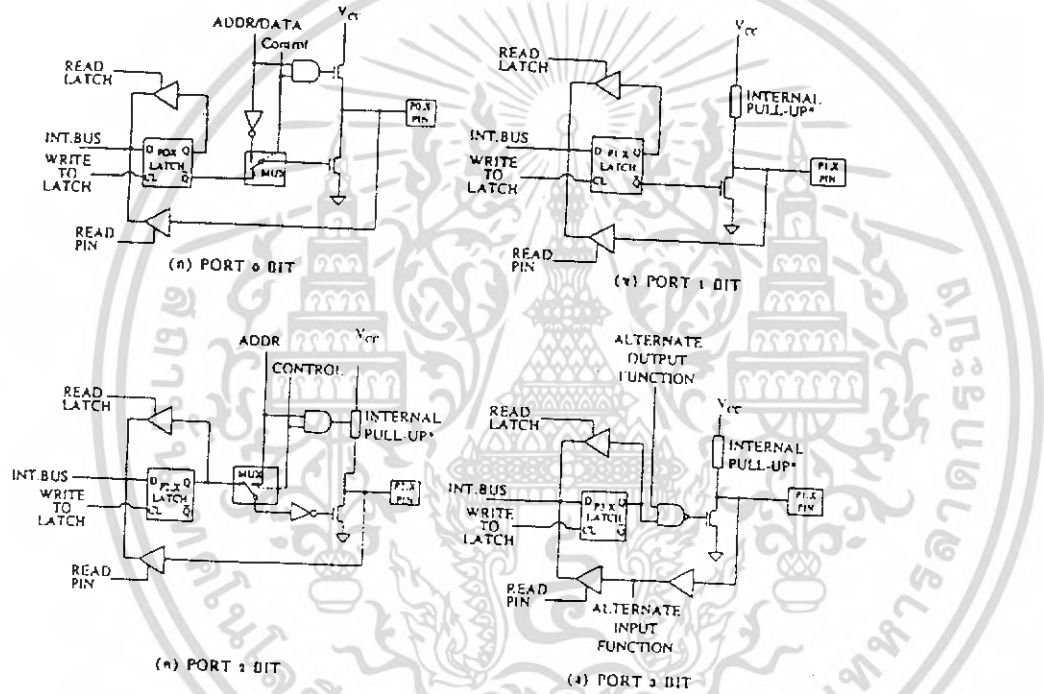
6. ALE (Address Latch Enable) เนื่องจากพอร์ต 0 สามารถใช้เป็นขาอ้างคำแหน่ง และขาข้อมูล MCS-51 จะมีขา ALE ได้แก่ขา 30 ขานี้จะใช้มัลติเพล็กซ์สัญญาณแอดเดรสบัส ของพอร์ต 0 ในการใช้งานระบบ MCS-51 นั้น จะต้องมีอุปกรณ์มาต่อกับพอร์ต 0 ที่ทำหน้าที่แลทช์สัญญาณแอดเดรสบัสออกมาก่อนทางพอร์ต 0 จากนั้นจะส่งสัญญาณ ALE มาแลทช์ อุปกรณ์ภายนอก ให้เก็บค่าแอดเดรสบัสของพอร์ต 0 ไว้เพื่อใช้เป็นบัสข้อมูลต่อไป

7. EA (External Access) ขา EA ได้แก่ขาที่ 31 ถ้าขานี้เป็นลอจิก "1" จะบอกว่าให้อ่านโปรแกรมภายใน แต่ถ้าเป็นลอจิก "0" จะบอกว่าให้ MCS-51 ทำโปรแกรมโดยอ่านจากหน่วยความจำโปรแกรมภายนอก

8. RST (Reset) ได้แก่ขา 9 จะใช้ในการรีเซต MCS-51 โดยจะให้ขานี้เป็นลอจิก"1" อย่างน้อย 2 แมทชีนไซเคิลจึงจะรีเซตระบบได้

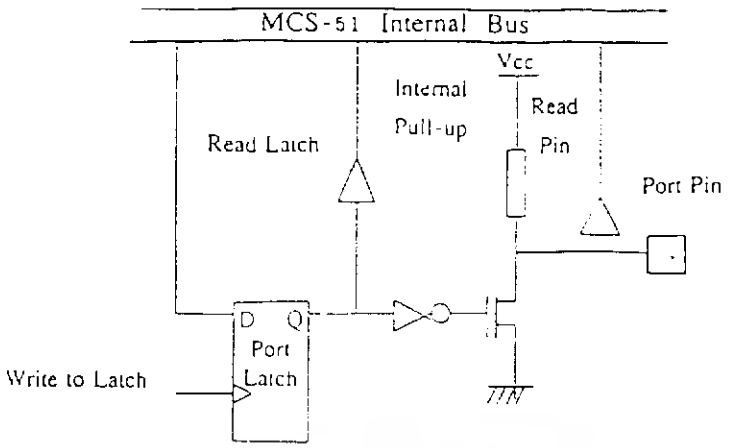
3. โครงสร้างของพอร์ตอินพุตเอาท์พุต(I/O Port Structure)

ขาของพอร์ตจะแสดงโครงสร้างภายในได้ดังรูปที่ 5 โดยจะมีโดครงสร้างเป็น Field-effect Transistor ต่ออยู่กับขาภายนอก และมีความต้านทานต่อ Pull-up ภายใน เพราะว่าต้องใช้เป็นขา Address Bus และ Data Bus



รูปที่ 5 โครงสร้างพอร์ตทั้ง 4 ของ MCS-51

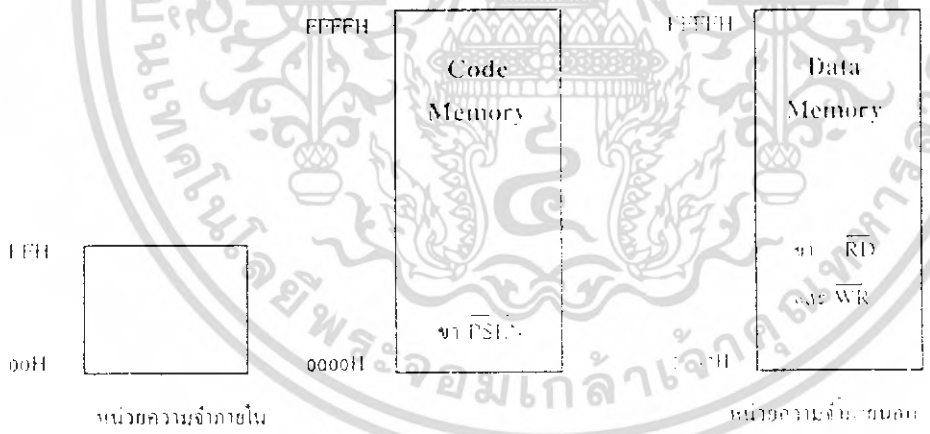
พอร์ตนี้สามารถใช้เป็นอินพุตเอาท์พุตกับอุปกรณ์ภายนอกได้ ในการอ่านข้อมูลจากพอร์ตจะอ่านได้สองแบบคือ Read Lacth และ Read Pin โดย Read Lacth หมายถึง การอ่านข้อมูลที่ถูก Lacth เอาไว้เข้าสู่บัสภายในของ MCS-51 เช่นการทำคำสั่ง CPL P1.5 แต่ถ้าเป็นการ Read Pin จะเป็นการใช้พอร์ตเป็นอินพุต โดยจะอ่านค่าจากขาของไอซีเข้าสู่บัสภายในโดยการอ่านแบบ Read Latch และ Read Pin จะมีสัญญาณมาควบคุมที่บัฟเฟอร์ดังรูปที่ 6



รูปที่ 6 การต่อพอร์ตเข้ากับระบบบัสภายในของ MSC-51

4. โครงสร้างหน่วยความจำ

หน่วยความจำสำหรับ MSC-51 จะมี 2 ชนิดคือ หน่วยความจำที่ใช้เก็บโปรแกรม (ROM) กับหน่วยความจำที่ใช้เก็บข้อมูลในการประมวลผล (RAM) MSC-51 บางเบอร์เช่น 8051,8052 จะมีหน่วยความจำภายในชิพ และ MSC-51 ทุกเบอร์สามารถอ้างตำแหน่งหน่วยความจำภายนอกได้มากที่สุด 64K และสำหรับหน่วยความจำภายใน (RAM) จะประกอบไปด้วยพื้นที่ใช้งานทั่วไป, รีจิสเตอร์แบงค์, พื้นที่ใช้งานระดับบิท และรีจิสเตอร์ฟังก์ชั่นพิเศษ เราอาจเขียนไดแกรมของหน่วยความจำของ MSC-51 ได้ดังรูปที่ 7 โดยในรูปจะบอกว่าขาใดเป็นแอคทีฟ



รูปที่ 7 การจัดหน่วยความจำของ MSC-51

ใน MSC-51 จะมีหน่วยความจำภายในตั้งแต่ตำแหน่ง 00H ถึง FFH และสามารถอ้างหน่วยความจำภายนอกได้ 64K ถ้าอ่านข้อมูลจากหน่วยความจำโปรแกรมขา PSEN จะแอคทีฟ นอกจากนี้ยังสามารถหน่วยความจำข้อมูลภายนอกได้ 64K ตำแหน่ง โดยการติดต่อกับหน่วยความจำนี้ ขา RD และ WR จะแอคทีฟ สำหรับหน่วยความจำข้อมูลภายในนั้นจะแบ่งออกได้ดังนี้

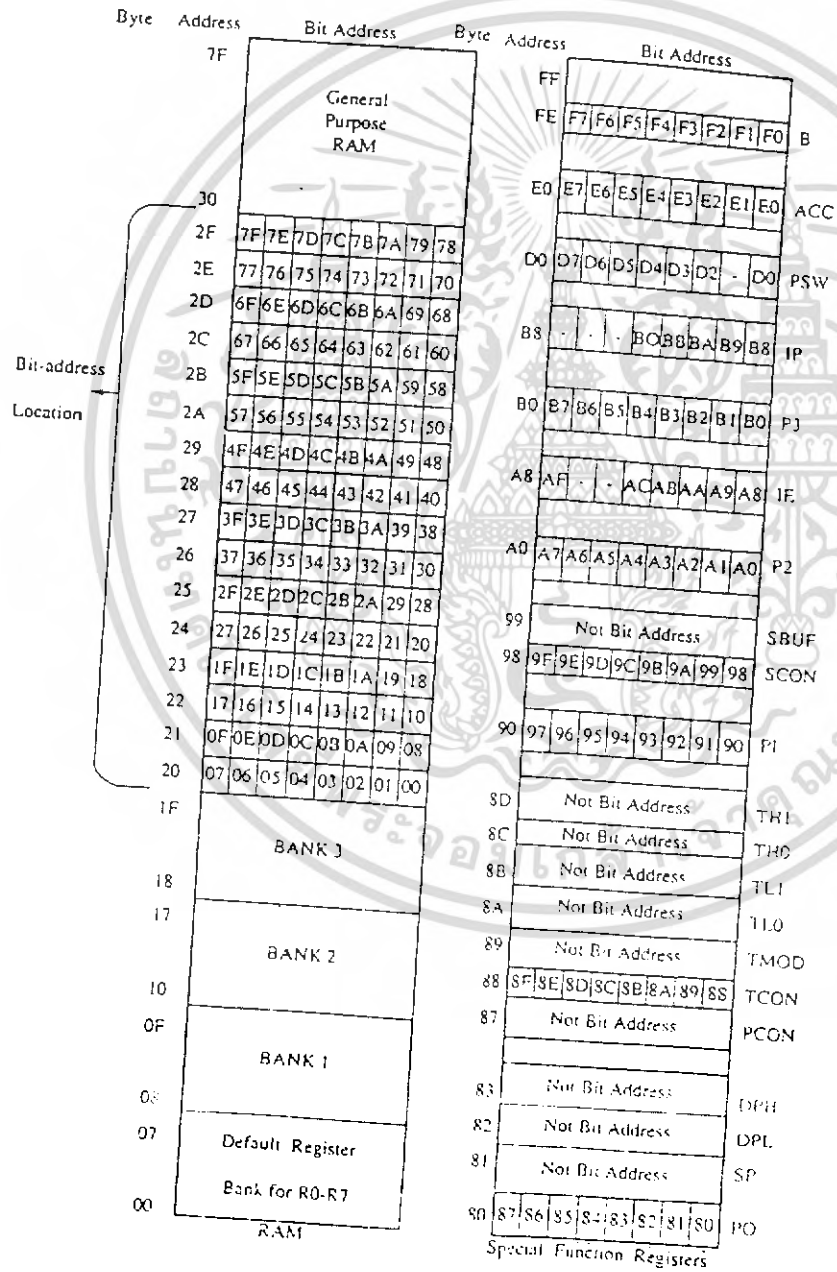
1) ชุดรีจิสเตอร์ 4 ชุด แต่ละเรียกว่า รีจิสเตอร์แบงค์ที่ตำแหน่ง 00H ถึง 1FH โดยแต่ละชุดประกอบด้วยรีจิสเตอร์ R0 ถึง R7

2) หน่วยความจำที่สามารถเข้าถึงข้อมูลระดับบิตได้ตำแหน่ง 20H ถึง 2FH

3) หน่วยความจำใช้งานทั่วไปตำแหน่ง 30H ถึง 7FH

4) รีจิสเตอร์ฟังก์ชันพิเศษ ตำแหน่ง 80H ถึง FFH

แผนผังการจัดหน่วยความจำข้อมูลภายในแสดงได้ดังรูปที่ 8 จากแผนผังจะเห็นได้ว่าการอ้างตำแหน่งหน่วยความจำภายในจะอ้างได้สองแบบ คือการอ้างไปที่ตำแหน่งของไบต์ (เขียนลนหมายเลขตำแหน่งด้านนอก) หรือการอ้างไปที่ตำแหน่งของบิต (เขียนหมายเลขตำแหน่งด้านใน) โดยที่ตำแหน่งของหน่วยความจำที่อ้างเป็นแบบบิตได้จะมีตำแหน่งที่แน่นอน



รูปที่ 8 ตำแหน่งของหน่วยความจำทั้งแบบไบต์และแบบบิต

4.1. หน่วยความจำใช้งานทั่วไป

จากรูปที่ 8 จะเห็นว่าใน 8031 จะมีหน่วยความจำ RAM สำหรับใช้งานทั่วไปจำนวน 80 ไบต์ ตั้งแต่ตำแหน่งที่ 30H ถึง 7FH ตำแหน่งเหล่านี้สามารถอ้างตำแหน่งแบบ Direct Addressing Mode หรือ Indirect Addressing Mode (รายละเอียดจะกล่าวในเรื่องชุดคำสั่ง) ได้ ตัวอย่างเช่น ถ้าต้องการอ่านข้อมูลที่อยู่ในตำแหน่ง 5FH มาเก็บไว้ในรีจิสเตอร์ A สามารถเขียนaนเป็นคำสั่งได้เป็น

```
MOV A,5FH
```

การย้ายตำแหน่งแบบนี้เป็นการย้ายข้อมูลจากตำแหน่งที่เก็บโดยตรง (ตำแหน่ง 5FH) เรียกว่าการอ้างตำแหน่งแบบ Direct Addressing Mode นอกจากนี้ยังสามารถอ่านข้อมูลโดยใช้รีจิสเตอร์ R0 หรือ R1 เป็นตัวชี้ตำแหน่งได้เรียกว่า การอ้างตำแหน่งแบบ Indirect Addressing Mode ตัวอย่างเช่น

```
MOV R0,#5FH
MOV A,@R0
```

การเขียนโปรแกรมด้านบน หมายความว่า เก็บค่า 5FH ไว้ใน R0 จากนั้นอ่านค่าที่ R0 จากนั้นอ่านค่าที่ R0 ชี้อยู่คือตำแหน่ง 5FH มาเก็บไว้ที่รีจิสเตอร์ A ถ้าในตำแหน่ง 5FH มี 80 อยู่ค่า 80 จะถูกเก็บใน A

รูปที่ 9 ขั้นตอนต่างๆ ในการอ่านข้อมูล

4.2 Bit Addressable RAM

ใน MCS51 จะมีหน่วยความจำที่สามารถอ้างข้อมูลในระดับบิตได้ตั้งแต่ตำแหน่ง 20H ถึง 2FH รวม 16 ไบต์ โดยสามารถ SET, CLEAR, AND, OR ทางลอจิกได้ จำนวนบิตที่ใช้งานได้ทั้งหมดมีจำนวน 128 บิต (8 บิต x 16 ไบต์) ถ้าต้องการเซตบิตที่ 67H สามารถเขียนaนคำสั่งได้ดังนี้

SET B,67H

จากรูปที่ 8 จะเห็นว่าบิตที่ 67H จะอยู่ในตำแหน่งไบต์ที่ 2CH

4.3 Register Banks

หน่วยความจำข้อมูลภายในที่ใช้เป็นรีจิสเตอร์ มีทั้งหมด 32 ตำแหน่ง โดยจะมี 4 ชุดแต่ละชุดมีรีจิสเตอร์ 8 ตัว คือ R0 ถึง R7 โดยชุดแรกจะอยู่ในตำแหน่ง 00H-07H ถ้าหากจะอ่านค่าจากตำแหน่ง 05H มาเก็บไว้ในรีจิสเตอร์ A จะเขียนโปรแกรมได้ดังนี้

MOV A,R5

การอ้างตำแหน่งจะใช้แบบ Register Addressing ซึ่งขนาดของรหัสคำสั่งจะมีขนาด 1 ไบต์ แต่ถ้าเขียนคำสั่งเป็น MOV A,05H ผลที่ได้จะเหมือนกันแต่การเขียนแบบนี้ ถ้าแปลงเป็นรหัสคำสั่งจะมีขนาด 2 ไบต์ ซึ่งจะทำให้โปรแกรมยาวกว่าแบบแรก ในการติดต่อกับ Register Bank นั้น เราสามารถเลือกให้ Bank ใดแอคทีฟได้โดยเขียนข้อมูลไปที่ Program Status Word ซึ่งอยู่ในส่วนของรีจิสเตอร์ฟังก์ชันพิเศษ เช่น ถ้าโปรแกรมให้ Bank 3 แอคทีฟ จะย้ายข้อมูลจากรีจิสเตอร์ A ไปที่ตำแหน่ง 18H ได้ดังนี้

MOV R0,A

ถ้าไม่มีการเลือก Bank จะเป็นการติดต่อกับรีจิสเตอร์ Bank แรกเสมอ

5 **รีจิสเตอร์ฟังก์ชันพิเศษ (Spesial Function Register)**

ใน MCS-51 รีจิสเตอร์จะใช้หน่วยความจำ RAM ภายในชิพ โดยส่วนหนึ่งเป็นรีจิสเตอร์พิเศษ (SFR) ซึ่งมีทั้งหมด 21 ตัว โดยรีจิสเตอร์พิเศษต่างๆ จะเริ่มที่หน่วยความจำตั้งแต่ 80H ถึง FFH ซึ่งมีทั้งหมด 128 ตำแหน่ง แต่จะเป็นรีจิสเตอร์ฟังก์ชันพิเศษเพียง 21 ตำแหน่ง แต่ถ้าเป็น 8032/8051 ซึ่งจะใช้ 26 ตำแหน่งหรือมี SFR 26 ตัว

จากรูปที่ 8 จะแสดงตำแหน่งหน่วยความจำของรีจิสเตอร์ บางตัวสามารถเข้าถึงข้อมูลแบบบิตได้อีกด้วย เช่น ถ้าเขียนโปรแกรมเป็น

SET B,0E0H

จะเป็นการเซตบิต 0 ของ Accumulator เนื่องจากตำแหน่ง E0H เนื่องจากตำแหน่งของรีจิสเตอร์ A และเป็นบิต Address บิตแรกของรีจิสเตอร์ A ด้วย คำสั่ง SETB (set bit) จะมีผลต่อบิตเท่านั้นจะไม่มีผลต่อไบต์ ถ้าหากต้องการติดต่อกับพอร์ต 1 อยู่ที่ตำแหน่ง 90H แต่ตำแหน่งของระดับบิตจะอยู่ที่ตำแหน่ง 90H ถึง 97H รีจิสเตอร์ในกลุ่ม Spasial Function Register มีดังนี้

SET B,67H

จากรูปที่ 8 จะเห็นว่าบิตที่ 67H จะอยู่ในตำแหน่งไบต์ที่ 2CH

4.3 Register Banks

หน่วยความจำข้อมูลภายในที่ใช้เป็นรีจิสเตอร์ มีทั้งหมด 32 ตำแหน่ง โดยจะมี 4 ชุดแต่ละชุดมีรีจิสเตอร์ 8 ตัว คือ R0 ถึง R7 โดยชุดแรกจะอยู่ในตำแหน่ง 00H-07H ถ้าหากจะอ่านค่าจากตำแหน่ง 05H มาเก็บไว้ในรีจิสเตอร์ A จะเขียนโปรแกรมได้ดังนี้

MOV A,R5

การอ้างตำแหน่งจะใช้แบบ Register Addressing ซึ่งขนาดของรหัสคำสั่งจะมีขนาด 1 ไบต์ แต่ถ้าเขียนคำสั่งเป็น MOV A,05H ผลที่ได้จะเหมือนกันแต่การเขียนแบบนี้ ถ้าแปลงเป็นรหัสคำสั่งจะมีขนาด 2 ไบต์ ซึ่งจะทำให้โปรแกรมยาวกว่าแบบแรก ในการติดต่อกับ Register Bank นั้น เราสามารถเลือกให้ Bank ใดแอคทีฟได้โดยเขียนข้อมูลไปที่ Program Status Word ซึ่งอยู่ในส่วนของรีจิสเตอร์ฟังก์ชันพิเศษ เช่น ถ้าโปรแกรมให้ Bank 3 แอคทีฟ จะย้ายข้อมูลจากรีจิสเตอร์ A ไปที่ตำแหน่ง 18H ได้ดังนี้

MOV R0,A

ถ้าไม่มีการเลือก Bank จะเป็นการติดต่อกับรีจิสเตอร์ Bank แรกเสมอ

5 รีจิสเตอร์ฟังก์ชันพิเศษ (Spesial Function Register)

ใน MCS-51 รีจิสเตอร์จะใช้หน่วยความจำ RAM ภายในชิพ โดยส่วนหนึ่งเป็นรีจิสเตอร์พิเศษ (SFR) ซึ่งมีทั้งหมด 21 ตัว โดยรีจิสเตอร์พิเศษต่างๆ จะเริ่มที่หน่วยความจำตั้งแต่ 80H ถึง FFH ซึ่งมีทั้งหมด 128 ตำแหน่ง แต่จะเป็นรีจิสเตอร์ฟังก์ชันพิเศษเพียง 21 ตำแหน่ง แต่ถ้าเป็น 8032/8051 ซึ่งจะใช้ 26 ตำแหน่งหรือมี SFR 26 ตัว

จากรูปที่ 8 จะแสดงตำแหน่งหน่วยความจำของรีจิสเตอร์ บางตัวสามารถเข้าถึงข้อมูลแบบบิตได้อีกด้วย เช่น ถ้าเขียนโปรแกรมเป็น

SET B,0E0H

จะเป็นการเซตบิต 0 ของ Accumulator เนื่องจากตำแหน่ง E0H เนื่องจากตำแหน่งของรีจิสเตอร์ A และเป็นบิต Address บิตแรกของรีจิสเตอร์ A ด้วย คำสั่ง SETB (set bit) จะมีผลต่อบิตเท่านั้นจะไม่มีผลต่อไบต์ ถ้าหากต้องการติดต่อกับพอร์ต 1 อยู่ที่ตำแหน่ง 90H แต่ตำแหน่งของระดับบิตจะอยู่ที่ตำแหน่ง 90H ถึง 97H รีจิสเตอร์ในกลุ่ม Spasial Function Register มีดังนี้

5.1 Program Status Word

รีจิสเตอร์ตัวนี้เรียกย่อๆว่า PSW จะอยู่ที่ตำแหน่ง D0H ซึ่งสามารถเข้าถึงข้อมูลระดับบิต ได้โดย รีจิสเตอร์ตัวนี้จะเป็นตัวบอกสภาวะต่างๆของไมโครคอนโทรลเลอร์ ความหมายของแต่ละบิต แสดงได้ดังในตารางที่ 2

CY	AC	F0	RS1	RS2	OV	-	P

ตารางที่ 3 แสดงบิตและหน้าที่ต่างๆใน PSW

บิต	ชื่อบิต	ตำแหน่ง	ความหมาย
PSW.7	CY	D7H	Carry Flag
PSW.6	AC	D6H	Auxilary Carry Flag
PSW.5	F0		Flag 0
PSW.4	RS1		บิตสำหรับเลือก Register Bank 1
PSW.3	RS0		บิตสำหรับเลือก Register Bank 1
			00=Bank 0;Address 00H-07H 01=Bank 1;Address 08H-0FH 10=Bank 2;Address 10H-17H 11=Bank 3;Address 18H-1FH
PSW.2	OV		Overflow Flag
PSW.1	-		Reserved
PSW.0	P		Even Parity Flag

1. แฟลกตัวทด Carry Flag (CF) บิตนี้เป็นบิตที่ 7 ของ PSW บิตนี้จะมีความสำคัญหากมีการกระทำทางคณิตศาสตร์ โดยบิตนี้จะ SET เมื่อเกิดการทดของบิตที่ 7 เมื่อเกิดการลบเลข ตัวอย่างเช่น ถ้าหากค่าใน Accumulator มีค่าเป็น FFH แล้วทำคำสั่งนี้

ADD A,#1

ค่าใน Accumulator จะเปลี่ยนเป็น 00H และบิต CY ใน PSW จะถูกเซตนอกจากนี้ บิต CY สามารถใช้เป็น "Boolean Accumulator" ได้ซึ่งอาจเรียกได้ว่าเป็น รีจิสเตอร์ขนาด 1 บิตได้ ตัวอย่าง เช่น ถ้าหากจะ AND บิตที่ 25H กับ CY ผลลัพธ์ที่ได้จะถูกเก็บใน CY ซึ่งเขียนลนคำสั่งได้ดังนี้

ANL C,25H

2. แฟลกตัวช่วยทด Auxilary Carry Flag (AC) เมื่อมีการบวกแบบ Binary-code-decimal (BCD) บิต AC จะถูกเซตเมื่อมีการทดจากบิตที่ 3 ไปบิตที่ 4 หรือถ้าใน Lower Nibble มีค่าระหว่าง 0AH-0FH เนื่องจากรหัส BCD นี้มีค่ามากที่สุดแค่ 9 ถ้าหากมีการบวกแบบ BCD จะต้องตามด้วยคำสั่ง DAA (Decimal Adjust Accumulator) เพื่อปรับค่าที่มีค่าเกิน 9 โดยบวกเลข 6 เข้าไป จะทำให้รหัส BCD ที่แทนเลขฐานสิบได้

3 แฟลกศูนย์ เป็นแฟลกที่ใช้งานได้ทั่วไป

4. บิตเลือกรีจิสเตอร์แบงก์ (Register Bank Bits) ตามที่ทราบมาแล้วว่าใน MCS-51 จะมีชุดรีจิสเตอร์อยู่ 4 ชุด ถ้าจะเลือกให้ชุดใดแอกทีฟจะกำหนดได้ในบิต RS1,RS2 ของ PSW และจะเคลียร์ตัวเองเมื่อระบบถูกรีเซต ถ้าหากต้องการติดต่อกับรีจิสเตอร์ Bank 3 โดยย้ายข้อมูลจาก R7 (ตำแหน่ง 1FH) มาเก็บในแอคคิวมูเลเตอร์ จะเขียนโปรแกรมได้ดังนี้

```
SETB RS1
SETB RS0
MOV A,R7
```

ในโปรแกรม Assembled สามารถรับรู้สัญญลักษณ์ RS1 และ RS2 ได้ เช่น SETB RS1 จะมีความหมายเท่ากับ SETB 0D4H หรือเซตบิตตำแหน่งที่ D4H

5. แฟลกโอเวอร์โฟลว์ (OV) จะถูกเซต หลังจากการกระทำทางคณิตศาสตร์แล้วเกิด Overflow คือจำนวนที่เกิดจากการบวกหรือลบมีค่าเกินกว่าที่จำนวนไบต์จะ เป็นไปได้มากกว่า +128 หรือน้อยกว่า -128 ตัวอย่างเช่น ถ้าเกิดการบวกเลขสองจำนวนนี้จะเกิดการเซตบิต OV ขึ้นใน PSW

```
Hex  :  0F        DEC  :  15
       +7E               +127
        8E               142
```

6. บิตพาริตี้ (Parity Bit) เป็นบิตที่บอกค่าพาริตี้ของรีจิสเตอร์แอคคิวมูเลเตอร์ ซึ่งอาจเป็นตัวตรวจสอบความถูกต้องของข้อมูลได้ โดยจะเซตหรือเคลียร์ขึ้นกับผลที่เกิดขึ้นกับแอคคิวมูเลเตอร์ เช่น ถ้าแอคคิวมูเลเตอร์ มีค่าเป็น10101101B บิต P จะเป็น 1

5.2 รีจิสเตอร์ B

รีจิสเตอร์ B จะอยู่ตำแหน่ง F0H ของหน่วยความจำข้อมูลภายใน เป็นรีจิสเตอร์ที่สามารถใช้งานทั่วไปได้ โดยทั่วไปรีจิสเตอร์นี้จะใช้คูณหรือหารกับรีจิสเตอร์แอกคิวมูเลเตอร์ เช่น การทำคำสั่ง MUL AB ซึ่งเป็นการคูณแบบ 8 บิต โดยผลลัพธ์ที่ได้จะมีขนาด 16 บิต ซึ่งรีจิสเตอร์ A จะเก็บค่า 8 บิตต่ำ และรีจิสเตอร์ B จะเก็บค่า 8 บิตสูง สำหรับการหารโดยการทำคำสั่ง DIV AB โดยค่าใน A จะ

กหารด้วย B ผลลัพธ์ที่ได้จะเก็บใน รีจิสเตอร์ AB โดย B จะเก็บค่า 8 บิตต่ำ และ A จะเก็บค่า 8 บิตสูง รีจิสเตอร์ B นี้สามารถเข้าถึงข้อมูลระดับบิตได้โดยตำแหน่งของบิตคือ ตำแหน่ง F0H-F7H

5.3 ตัวชี้สแตค (Stack Pointer,SP)

SP เป็นริจิสเตอร์ขนาด 8 บิต อยู่ที่ตำแหน่ง 81H การเขียนค่าเข้าไปในตำแหน่งที่ SP ชี้อยู่นี้เรียกว่า "Pushing" สำหรับการอ่านค่าที่ SP ชี้อยู่เรียกว่า "Popping" ค่าของ SP จะเพิ่มขึ้นหนึ่งก่อนที่จะเขียนข้อมูลลงไป และจะลดลงหนึ่งเมื่ออ่านข้อมูลออกมาแล้ว หากโปรแกรมทำคำสั่ง CALL จะใช้รีจิสเตอร์สแตคนี้เก็บค่าตำแหน่งเดิมของโปรแกรมย่อย เมื่อทำโปรแกรมย่อยเสร็จแล้วจะคืนค่าในสแตคให้กับ PC ตามเดิม โดยปกติค่า PC จะกำหนดให้อยู่ใน RAM ภายในถ้าต้องการให้มี SP เริ่มที่ตำแหน่ง 60H จะต้องเขียนคำสั่งดังนี้

```
MOV SP,#5FH
```

5.4 รีจิสเตอร์ Data Pointer (DPTR)

รีจิสเตอร์นี้ใช้สำหรับชี้ตำแหน่งรหัสโปรแกรมหรือข้อมูลในหน่วยความจำ โดยเป็นรีจิสเตอร์ขนาด 16 บิต ซึ่งประกอบด้วยรีจิสเตอร์ 2 ตัว คือ DPL ตำแหน่งที่ 82H โดยจะเก็บเป็น 8 บิตต่ำ และ DPH ตำแหน่งที่ 83H โดยจะเก็บค่า 8 บิตสูง รีจิสเตอร์ทั้งสองตัวนี้จะรวมกันกลายเป็นรีจิสเตอร์ 16 บิต ถ้าหากต้องการเก็บค่า 55H ไปยังตำแหน่งหน่วยความจำข้อมูลภายนอกตำแหน่งที่ 1000H จะเขียนโปรแกรมได้ดังนี้

```
MOV A,#55H
MOV DPTR,#1000H
MOVX @DPTR,A
```

ในบรรทัดแรกจะเป็นการอ้างตำแหน่งแบบ Immediate Address ซึ่งจะเก็บค่า 55H ลงในรีจิสเตอร์ A ต่อมาจะเก็บค่า 1000H ลงในรีจิสเตอร์ 16 บิต DPTR เพื่อชี้ไปที่ตำแหน่งหน่วยความจำ บรรทัดที่ 3 จะเป็นการอ้างตำแหน่งแบบ Indirect Addressing ซึ่งจะเก็บค่าใน A คือ 55H ลงในตำแหน่งที่ DPTR ชี้อยู่คือ ตำแหน่ง 1000H

5.5 รีจิสเตอร์พอร์ต (Port Register)

ใน MCS-51 ค่าของพอร์ตจะหมายถึงค่าของหน่วยความจำด้วย หากต้องการส่งข้อมูลออกไปที่พอร์ต ก็เพียงแต่เขียนข้อมูลไปที่หน่วยความจำตำแหน่งที่พอร์ตนั้นอยู่ใน MCS-51 พอร์ต 0 จะอยู่ที่ตำแหน่ง 80H ,พอร์ต 1 จะอยู่ที่ 90H ,พอร์ต 2 อยู่ที่ตำแหน่ง A0H และพอร์ต 3 อยู่ที่ตำแหน่ง

80H พอร์ต 0,2 และ3 โดยทั่วไปแล้วจะไม่ใช้ถ้าหากมีการติดต่อกับหน่วยความจำภายนอกหรือใช้เป็นพอร์ตพิเศษ (เช่น Interupt,Serial Portฯลฯ) โดยปกติแล้วจะใช้พอร์ต 1 ในการติดต่อกับอุปกรณ์ภายนอก พอร์ตทุกพอร์ตสามารถอ้างข้อมูลในระดับบิตได้

5.6 รีจิสเตอร์เวลา (Timer Register)

ใน MCS-51 เบอร์ 8051 จะมีรีจิสเตอร์ที่ใช้นับและจับเวลาขนาด 16 บิต 2 ตัว คือ Timer 0 อยู่ในตำแหน่งที่ 8AH และ 8CH โดยตำแหน่ง 8AH หมายถึง TL0 ซึ่งเป็น 8 ไบต์ต่ำและ 8CH หมายถึง 8 ไบต์สูง TH0 รีจิสเตอร์อีกตัวคือ Timer 1โดยแบ่งเป็น TL1 อยู่ที่ตำแหน่ง 8BH เป็นไบต์ต่ำ และ TH1 อยู่ที่ตำแหน่ง 8DH เป็นไบต์สูง การใช้ Timer จะต้องกำหนดการทำงานในรีจิสเตอร์ TMOD (Timer/Couter Mode Control Register) ซึ่งอยู่ที่ตำแหน่ง 88H เสียก่อน

5.7 รีจิสเตอร์พอร์ตอนุกรม Serial Port Register)

MCS-51 จะมีพอร์ตสื่อสารอนุกรมอยู่ภายในชิพ ซึ่งสสามารถจะรับหรือส่งข้อมูลได้โดยติดต่อผ่านรีจิสเตอร์ SBUF (Serial Data Buffer) ซึ่งอยู่ที่ตำแหน่ง 99H โดยถ้าต้องการส่งข้อมูลแบบอนุกรมให้เขียนข้อมูลไปที่รีจิสเตอร์นี้ ตัว Serial Port สามารถโปรแกรมให้ทำงานได้ 4 โหมด โดยโปรแกรมผ่านรีจิสเตอร์ SCON (Serial Port Control Register) ตำแหน่ง 98H

5.8 รีจิสเตอร์อินเตอร์รัพท์ (Interupt Port Register)

MCS-51 สามารถอินเตอร์รัพท์ได้ 5 ตำแหน่ง โดยมี 2-Priority ตัวอินเตอร์รัพท์นี้จะถูก Disable หลังจากระบบถูกรีเซตและจะ Enable หลังจากที่เขียนข้อมูลไปที่รีจิสเตอร์ IE หรือตำแหน่ง A8H ลำดับสำคัญสามารถเซตได้ที่รีจิสเตอร์ IP หรือตำแหน่ง B8H

5.9 Power Control Register (PCON)

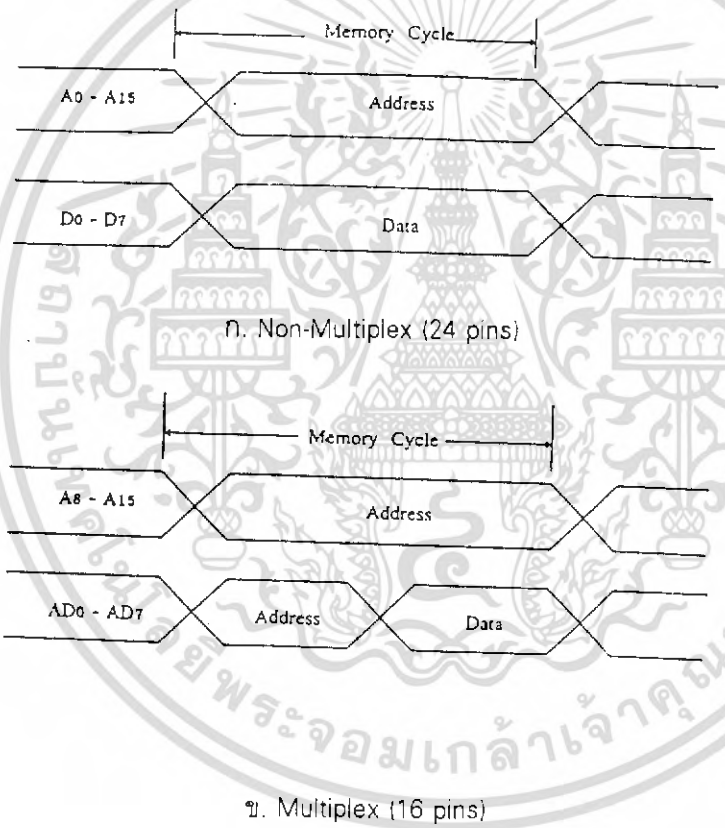
รีจิสเตอร์ PCON อยู่ที่ตำแหน่ง 87H ใช้หยุดการทำงานของ MCS-51 โดยจะหยุดจ่ายสัญญาณนาฬิกาให้ระบบ ทำให้ข้อมูลต่างๆภายใน MCS51 ไม่มีการเปลี่ยนแปลง นอกจากนี้ยังลดพลังงานไฟฟ้าที่จ่ายให้ MCS51 ด้วย

6. หน่วยความจำภายนอก (External Memery)

MCS-51 สามารถอ้างหน่วยความจำข้อมูลภายนอกได้ 64K และอ้างหน่วยความจำโปรแกรมภายนอกได้ 64K MCS-51 จะใช้พอร์ต 0 ในการอ้างตำแหน่งหน่วยความจำ 8 บิตล่างและใช้พอร์ต 0

เป็นพอร์ตข้อมูล (Data) ด้วย โดยใช้ขา ALE มาเป็น Latch ข้อมูลพอร์ต 0 และใช้พอร์ต 2 เป็นขาอ้างตำแหน่ง 8 บิตบน (รวมขาอ้างตำแหน่ง 16 เส้น ซึ่งอ้างได้ 64K)

เนื่องจากพอร์ต 0 จะใช้งาน 2 หน้าที่ในการติดต่อกับหน่วยความจำ จะใช้วิธี Multiplex ระหว่าง Address กับ Data พิจารณาจากรูป ถ้าต้องการติดต่อกับหน่วยความจำที่เก็บข้อมูล 8 บิต และเก็บได 64K จะต้องใช้สัญญาณ 24 เส้น คือเป็นขาแอดเดรส 16 เส้น และขาข้อมูล 8 เส้น ดังรูปที่ 11 ก. แต่ถ้าใช้วิธี Multiplex คือใช้ขา A0-A7 เป็นขาข้อมูลด้วยคือ D0-D7 จะใช้สายสัญญาณเพียง 16 เส้นเท่านั้น จากรูปที่ 11 ข. จะเห็นว่าเมื่อต้องการติดต่อกับหน่วยความจำจะส่งสัญญาณแอดเดรส A0-A15 ออกมาก่อน 16 เส้น และเวลาต่อมาขา A0-A7 จะถูกเปลี่ยนเป็น D0-D7 ในการติดต่อกับหน่วยความจำภายนอกของ MCS-51 จะใช้วิธีนี้



ก. Non-Multiplex (24 pins)

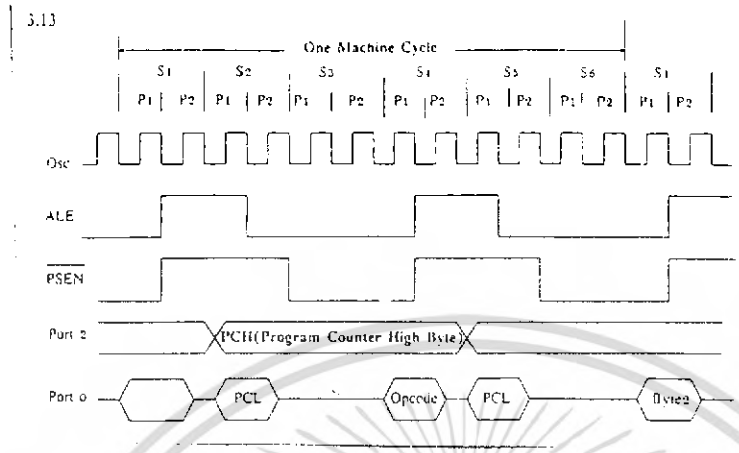
ข. Multiplex (16 pins)

รูปที่ 11 ไดอะแกรมกลุ่มสัญญาณที่ใช้อ่านข้อมูล

6.1 การติดต่อกับหน่วยความจำโปรแกรมภายนอก

ในการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก MCS-51 จะส่งค่าตำแหน่งของหน่วยความจำออกไปก่อน ซึ่งค่าตำแหน่งจะเก็บอยู่ใน PC โดยส่งออกไปทางพอร์ต 0 และพอร์ต 2 จากนั้นเวลาต่อมาจะส่งขา ALE ให้เป็นลอจิก "0" เพื่อแลทช์ขาแอดเดรสของ 8 บิตต่ำ คือพอร์ต 0 จากนั้น

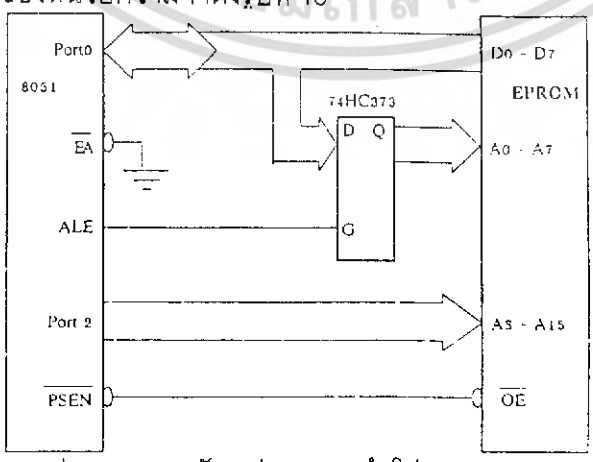
จะส่งสัญญาณทางขา PSEN ให้เป็นลอจิก "0" เพื่ออ่านข้อมูลซึ่งจะได้ Opcode เข้าไปทางขา Data Bus คือพอร์ต 0 ไดอะแกรมเวลาอ่านข้อมูลจากหน่วยความจำภายนอกแสดงได้ดังรูปที่ 12



รูปที่ 12 ไดอะแกรมเวลาการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

จากรูปช่วงเวลาการทำงานของ MCS-51 เรียกว่า State โดยแต่ละสเตทจะใช้สัญญาณนาฬิกาสองคาบ การทำงานของคำสั่งต่างๆจะใช้เวลาหกสเตท เรียกว่าแมทชีนไซเคิล จากรูปจะเห็นว่าใน S2 MCS-51 จะส่งค่าตำแหน่งของหน่วยความจำโปรแกรม (ค่า PC) ออกมาทางพอร์ต 0 และพอร์ต 2 เวลาต่อมาขา ALE จะเป็น "0" เพื่อ Latch อุปกรณ์ภายนอกให้คงค่าตำแหน่งไบต์ต่ำไว้(จากพอร์ต 0) ใช้พอร์ต 0 เป็นขาข้อมูลต่อไป เวลาต่อมาPSEN จะเป็น "0" เพื่ออ่านค่า Opcode เข้าทางพอร์ต 0

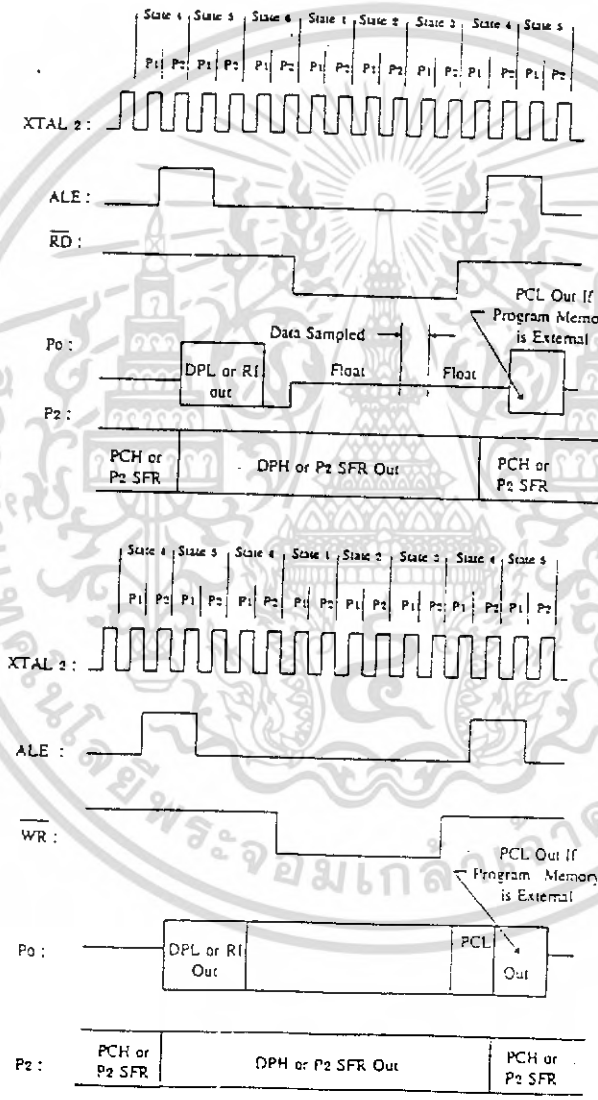
สำหรับการติดต่อกับหน่วยความจำกับ MCS-51 แสดงได้ดังรูปที่ 13 โดยขา EA จะต่อเป็น "0" เพื่อบอก MCS-51 ว่าให้อ่านหน่วยความจำโปรแกรมภายนอก สำหรับการ Multiplex จะใช้ฟลิปฟลอป 8 ตัว เก็บค่าตำแหน่ง 8 บิตต่ำเอาไว้ เมื่อ MCS-51 ส่งค่าตำแหน่งพอร์ตออกไปเวลาต่อมาจะส่งขา ALE ให้เป็น "0" ซึ่งจะใช้ขานี้ต่อกับฟลิปฟลอปเพื่อให้แลทช์ข้อมูลสำหรับขา PSEN จะต่อกับขา Output Enable (OE) ของหน่วยความจำดังรูปที่ 13



รูปที่ 13 การต่อ MCS-51 กับหน่วยความจำโปรแกรมภายนอก

หน่วยความจำข้อมูลภายนอก MCS-51 สามารถอ่านและเขียนได้ ในการติดต่อกับหน่วยความจำข้อมูลภายนอก MCS-51 จะส่งขา Address ออกไปทางพอร์ต 0 และพอร์ต 2 จากนั้นจะส่งขา ALE เพื่อไปแลทช์ Address 8 บิตต่ำ โดยการอ่านเขียนข้อมูลนั้นจะใช้ขา RD หรือ P3.7 และขา WR หรือ P3.6 ตามลำดับ

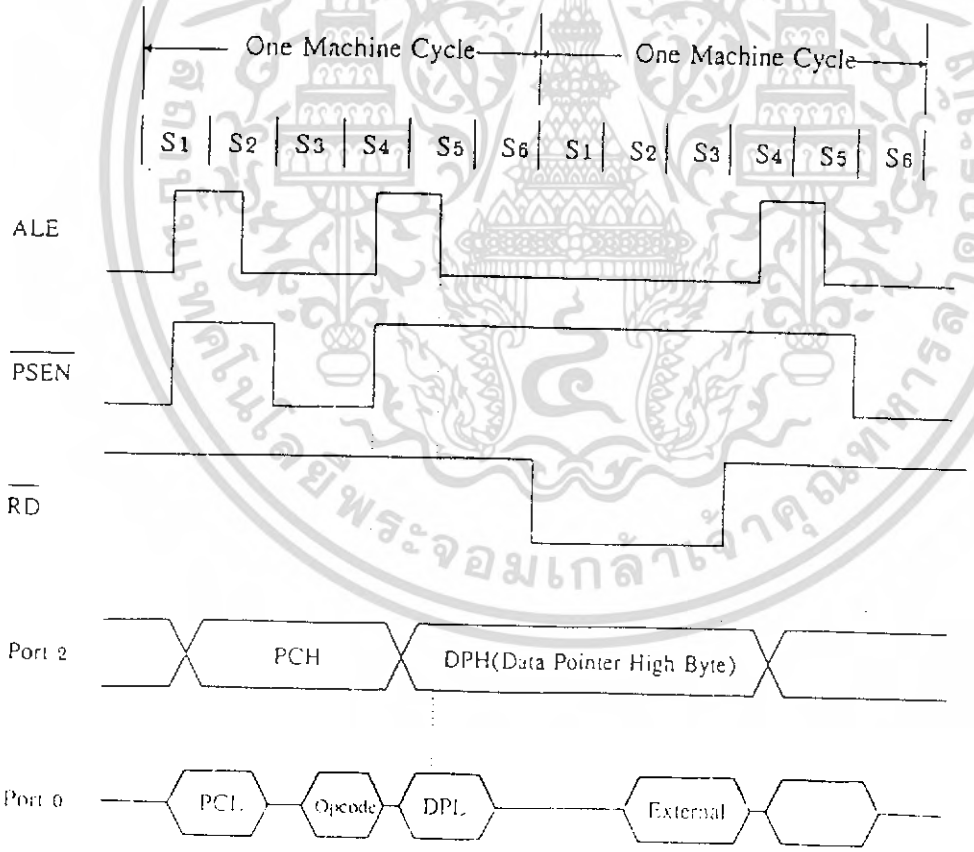
ไดอะแกรมเวลาการอ่านและเขียนข้อมูลกับหน่วยความจำภายนอกแสดงได้ดังรูปที่ 3.15



รูปที่ 14 ไดอะแกรมเวลาการอ่านและเขียนข้อมูลกับหน่วยความจำภายนอก

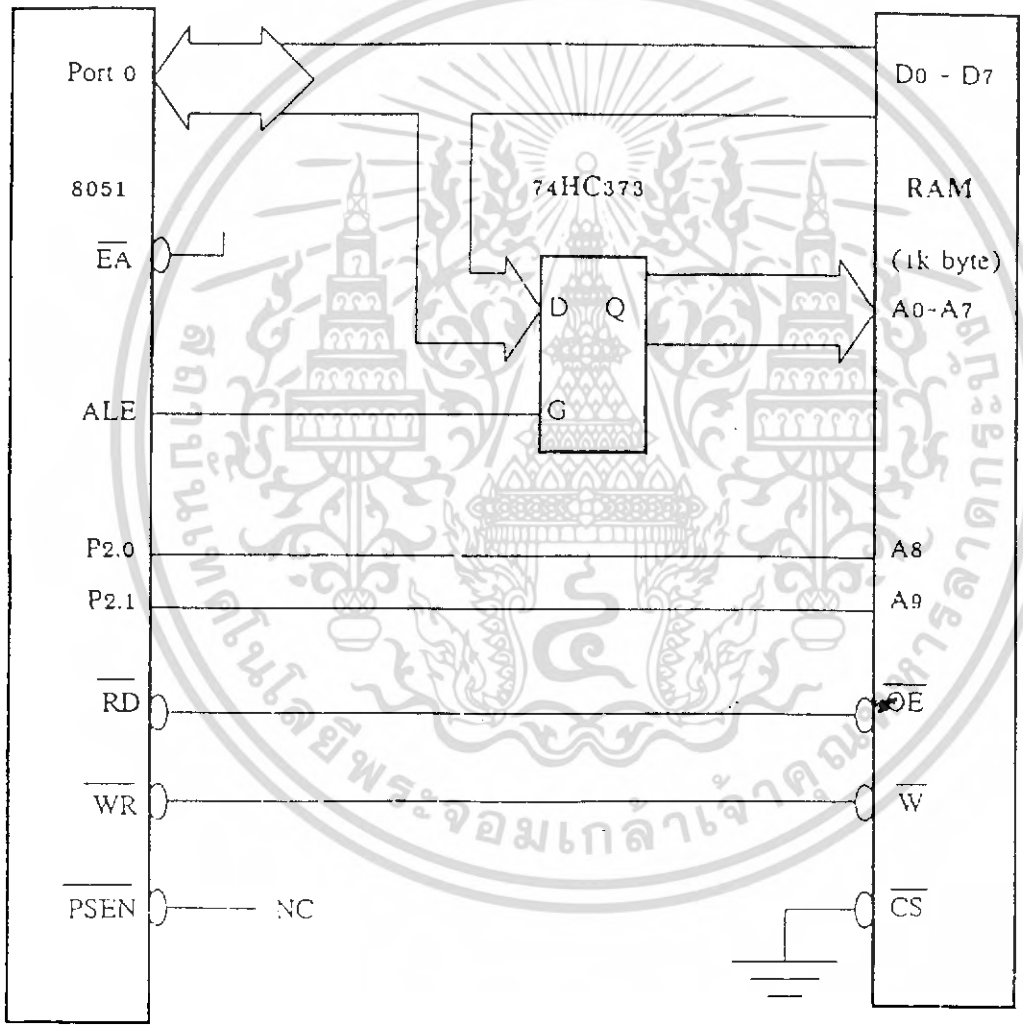
เนื่องจากตำแหน่งของหน่วยความจำภายนอกมีได้ถึง 64K รีจิสเตอร์ที่ใช้เก็บค่าตำแหน่งของหน่วยความจำภายนอกจะใช้รีจิสเตอร์ 16 บิต คือ DPTR นอกจากนี้ยังใช้รีจิสเตอร์ 8 บิต ได้ 2 ตัวคือ R0 และ R1 ในการติดต่อกับหน่วยความจำภายนอกจะใช้คำสั่ง MOVX

ถ้าหาก MCS-51 ทำคำสั่ง MOVX A,@DPTR ซึ่งหมายความว่าให้อ่านค่าตำแหน่งที่ DPTR ชี้มาเก็บในรีจิสเตอร์ A ไดอะแกรมเวลาจะเป็นดังรูปโดยแมทชีนไซเคิลแรก จะเป็นการอ่านค่า Opcode ของโปรแกรมให้รู้ว่าทำคำสั่ง MOVX A,@DPTR ซึ่งในการอ่านค่าจากโปรแกรมจะได้ Opcode เข้ามาและตีความ จากนั้น MCS-51 จะรู้ว่าจะต้องอ่านข้อมูลจากตำแหน่งที่ DPTR ชี้อยู่ในแมทชีนไซเคิลต่อไป ก็จะนำค่า DPTR ส่งออกเป็นค่าแอดเดรส โดย DPH จะส่งไปทางพอร์ต 2 และ DPL จะส่งไปทางพอร์ต 0 จากนั้นขา ALE จะเป็น "0" และข้อมูลจะอ่านออกมาทางบัสข้อมูล คือพอร์ต 0 ไดอะแกรมเวลาแสดงได้ดังรูปที่ 15



รูปที่ 15 สัญญาณต่างๆที่เกิดขึ้นขณะทำคำสั่ง MOVX

สำหรับการเชื่อมต่อหน่วยความจำกับ MCS-51 โดยให้ทำงานกับหน่วยความจำแสดงได้ดังรูปที่ 16 ซึ่งเป็นการเชื่อมต่อแรมขนาด 1K byte ซึ่งจะใช้ขาแอดเดรส เพียง 10 เส้น ดังนั้น A8 และ A9 จะต่อกับ P2.0 และ P2.1 ส่วนขา EA จะต่อกับลอจิก "1" เพื่อบอกว่าให้อ่านจากรอมภายใน และขา PSEN จไม่ใช้เพราะไม่ได้ต่อ รอมแสดงได้ดังรูป



รูปที่ 16 การต่อหน่วยความจำโปรแกรมกับ MCS-51

ข้อสังเกตขนาดที่ MCS-51 ติดต่อกับหน่วยความจำโปรแกรม หรือหน่วยความจำข้อมูลภายนอกจะใช้ขาแอดเดรสเหมือนกัน แต่จะต่างกันตรงที่ถ้าติดต่อกับหน่วยความจำโปรแกรมขา PSEN จะแอคทีฟ ถ้าจะติดต่อกับหน่วยความจำข้อมูลขา WR,RD จะแอคทีฟและ MCS-51 จะติดต่อกับหน่วยความจำโปรแกรมด้วยคำสั่ง MOVC และติดต่อกับหน่วยความจำข้อมูลด้วยคำสั่ง MOVX

7. Reset Operation

การรีเซตหรือเริ่มต้นทำงานใหม่ของ MCS-51 จะต้องให้ลอจิก "1" ที่ขา RST เป็นเวลา 2 แมทชีนไซเคิล(เท่ากับ 12 clock) จากนั้นให้กลับเป็นลอจิก "0" การรีเซตอาจทำได้โดยใช้สวิทช์กดดังรูปที่ 17(ก) หรือใช้ Power-up โดยใช้ตัว R-C ต่อเป็นวงจรดังรูป (ข)

รูปที่ 17 การรีเซต MCS-51

การอินเตอร์เฟสกับจอแสดงผล LCD

การอินเตอร์เฟสและการเขียนโปรแกรมควบคุมการทำงานของจอแสดงผล LCD กับ MCS-51 ซิงกิลบอร์ดคอมพิวเตอร์ในยุคปัจจุบัน จอแสดงผล LCD ถูกนำมาใช้งานมากยิ่งขึ้น อันเนื่องมาจากการควบคุมที่ง่าย, แสดงผลตัวอักษรได้สวยงามชัดเจนและราคาคุ้มค่ากับประสิทธิภาพที่ได้รับ

การอินเตอร์เฟสกับจอแสดงผล LCD โมดุล

ในระบบที่ใช้งานไมโครคอนโทรลเลอร์ทำงานแบบระบบเดี่ยว บ่อยครั้งที่จำเป็นต้องติดตั้งอุปกรณ์แสดงผลด้วย เพื่อแสดงข้อความ,ตัวเลข,ผลการวัดค่า หรือข้อมูลต่างๆในการสื่อสารกับผู้ใช้งานได้ และอุปกรณ์หนึ่งที่ถูกนำมาใช้งานกันอย่างกว้างขวางก็คือ จอแสดงผล LCD โมดูลตัวอย่างที่เห็นกันบ่อยก็คือนำมาใช้ในการแสดงผลแบบเมนู เพื่อให้ผู้ใช้งานกดคีย์บอร์ดในการเลือกรูปแบบการทำงานต่อไป เป็นต้น

1. การเชื่อมต่อและควบคุม

การเชื่อมต่อทางไฟฟ้าระหว่างบอร์ด MCS-51 และจอแสดงผล LCD ใช้ลวดจั๊มหรือสายไฟแพรขนาด 14 เส้น จอแลดงผล LCD กำหนดให้ใช้รุ่นใดยี่ห้อใดก็ได้ที่ใช้ไอซีคอนโทรลเลอร์อ้างอิงเป็นเบอร์ HD44780 หรือเบอร์อื่นที่เทียบเท่า ตัวอย่างเช่นรุ่น H2570, LM016L, และ LM1612A เป็นต้น ขาเชื่อมต่อของ LCD ของแต่ละยี่ห้อแต่ละแบบไม่มีมาตรฐานที่แน่นอนเหมือนกัน การเชื่อมต่อขาต่างๆจึงต้องอาศัยดาต้าชีทของ LCD รุ่นนั้นๆที่จัดซื้อมาอ้างอิงเป็นหลักตำแหน่งขาต่างๆที่เชื่อมต่อไปยัง LCD ที่คอนเน็กเตอร์ K10 บนบอร์ดแสดงผลดังรูปที่ 1 อาศัยรายละเอียดจากรูปนี้เพื่อต่อเชื่อมไปยัง LCD โมดูลและอ้างอิงในการตรวจสอบวงจรได้ การควบคุมการทำงานของ LCD อาศัยสัญญาณอีนาเบิลซึ่งจะถูกส่งไปยัง LCD หลังจากที่มีการกำหนดรูปแบบการทำงานว่าอยู่ในโหมดอ่านหรือเขียนไปยังตำแหน่งแอดเดรส 0C009H จากรูปที่ 1 จะเห็นว่าขาสัญญาณเลือกโหมดการอ่านหรือเขียน(R/W) ต่อวงจรโดยตรงกับขา RD ของ MCS-51 บอร์ด เมื่อระบุโหมดการอ่านหรือเขียนแล้ว ต้องกำหนดการทำงานของ LCD ด้วยว่าในโหมดการอ่านหรือการเขียนนั้นอยู่ในรูปของตัวอักขระหรือข้อมูล คำสั่งการกำหนดรูปแบบของข้อมูลทำได้ที่ขาสัญญาณ RS ของ LCD ซึ่งเชื่อมต่อโดยตรงกับขา IOA3 ของไมโครคอนโทรลเลอร์ โดยถ้า RS=IOA3=1 และติดต่อกับแอดเดรส 0C009H 0 จะเป็นโหมดการอ่านหรือเขียนข้อมูลอักขระ และถ้า RS=IOA3=0 และการติดต่อกับแอดเดรส 0C001H จะเป็นการอ่านหรือเขียนข้อมูลคำสั่งข้อมูลภายใน LCD

การส่งผ่านข้อมูลทั้งตัวอักขระและคำสั่งควบคุมถูกป้อนเข้ากับบัสข้อมูล 2 ทิศทางของระบบ จริงๆแล้วการใช้งานจอแสดงผล LCD สามารถใช้งานได้ทั้งในโหมด 4 บิต (D0-D3) หรือโหมด 8 บิต (D0-D7) ก็ได้แต่เนื่องจากในระบบที่เราศึกษาอยู่เป็นไมโครคอนโทรลเลอร์ที่ทำงานแบบ 8 บิต ดังนั้น

การเชื่อมต่อ LCD จึงต้องใช้งานแบบ 8 บิตกับบัสข้อมูลพอดี และสามารถเขียนโปรแกรมให้ทำงานในโหมดส่งผ่านข้อมูล 8 บิต กับจอ LCD ได้ทันที อย่างไรก็ตามในบางกรณีผู้อ่านจะต้องศึกษาข้อมูลเพิ่มเติมในการใช้งาน LCD ในโหมดการส่งผ่านข้อมูล 4 บิต ด้วย เช่น ถ้ามีพอร์ตเหลือใช้งานเพียงพอร์ตเดียว แต่จำเป็นต้องมีการต่อใช้งานพอร์ตนี้กับระบบส่วนอื่นๆ ด้วยการเลือกใช้งาน LCD ให้เป็นโหมดส่งผ่านข้อมูล 4 บิตนี้ จึงเป็นวิธีช่วยลดสายสัญญาณของพอร์ตที่มีจำกัดให้เหลือใช้งานได้มากขึ้น จากรูปที่ 1 ที่ตำแหน่งขา 3 ของ K10 ถูกต่อเข้ากับตัวต้านทานปรับค่าได้เพื่อปรับความเข้มได้เหมาะสมกับสภาพแวดล้อมในขณะนั้นพร้อมทั้งความเข้มของแสงจากภายนอกและมุมมองของผู้ใช้งาน ก่อนการเชื่อมต่อ LCD กับ K10 ควรแน่ใจว่าตำแหน่งขาต่างๆ ในการเชื่อมต่อทั้งหมดถูกต้องแล้วจึงทำการเชื่อมต่อจริง ถ้าไม่แน่ใจควรตรวจสอบจากดาต้าชีทให้แน่ใจอยู่เสมอ เพราะไม่เช่นนั้นอาจทำให้เกิดการเสียหายได้

2. คำสั่งควบคุมการทำงานของ LCD

คำสั่งควบคุมการทำงานของ LCD ที่ใช้ไมโครคอนโทรลเลอร์ของฮิตาชิตามดาต้าบุคนั้นมีความยาวประมาณ 30 หน้า สำหรับคำอธิบายครบทุกคุณสมบัติของ LCD ดังนั้นในที่นี้จะกล่าวถึงคำสั่งและการใช้งานทั้งหมดคงไม่ไหว หารนำเสนอคงทำได้เฉพาะสิ่งที่สำคัญ และควรทราบครอบคลุมการใช้งานทั่วไปเพียงเท่านั้น ถ้าท่านใดต้องการศึกษารายละเอียดของ LCD รุ่นใดก็หาได้จากดาต้าบุคจากผู้ผลิตหรือผู้จำหน่าย

ความสามารถหลักที่ LCD นิยมถูกหยิบยกมาใช้งานนั้น เพราะมันสามารถแสดงผลตัวอักขระแอสกีได้ ซึ่งมีอยู่ในไมโครคอนโทรลเลอร์ของ LCD ซึ่งแต่ละรุ่นก็มักใช้คอนโทรลเลอร์ตัวเดียวกันเสมอ นั่นคือคำสั่งควบคุมต่างๆจึงใช้เหมือนกันใน LCD แต่ละรุ่น

ก่อนที่จะเริ่มต้นศึกษารูปแบบคำสั่งควบคุมต่างๆ ผู้อ่านควรทราบถึงคำศัพท์บางคำเกี่ยวกับองค์ประกอบและการงานพื้นฐานของคอนโทรลเลอร์ใน LCD กันก่อน การทำงานภายใน LCD จะมีบัฟเฟอร์อยู่ภายใน ซึ่งมีความจุประมาณ 80 อักขระมีชื่อเรียกว่า DD-RAM (Display Data RAM) มีตำแหน่งแอดเดรสอยู่ระหว่าง 000H-04FH ยกตัวอย่างเช่น ถ้าเป็น LCD ขนาด 16 ตัวอักษร 1 บรรทัด DD-RAM ที่จัดเก็บข้อมูลของตัวอักขระที่จะแสดงผลทั้ง 16 ตัว จะถูกเก็บอยู่ที่ตำแหน่งแอดเดรส 00H-FH โดยเริ่มต้นจากด้านซ้ายของจอแสดงผล

การทำให้เกิดช่องว่างใน DD-RAM สามารถทำได้โดยการใช้คำสั่งชิฟตัวอักขระในการแสดงผล หรืออีกวิธีหนึ่งที่ง่ายมากขึ้นก็คือ ในขณะที่มีข้อมูลตัวอักขระเก็บไว้ในแอดเดรสค่าหนึ่ง การแสดงผลตัวอักขระต่อไปหรือทำให้เกิดช่องว่างอาจทำได้โดยการกำหนดจุดเริ่มต้นของแอดเดรสใหม่แตกต่างจากเดิมไป 1 แอดเดรสของ DD-RAM ซึ่งทำให้ได้ผลลัพธ์เช่นเดียวกัน

การเชื่อมต่อ LCD จึงต้องใช้งานแบบ 8 บิตกับบัสข้อมูลพอดี และสามารถเขียนโปรแกรมให้ทำงานในโหมดส่งผ่านข้อมูล 8 บิต กับจอ LCD ได้ทันที อย่างไรก็ตามในบางกรณีผู้อ่านจะต้องศึกษาข้อมูลเพิ่มเติมในการใช้งาน LCD ในโหมดการส่งผ่านข้อมูล 4 บิต ด้วย เช่น ถ้ามีพอร์ตเหลือใช้งานเพียงพอร์ตเดียว แต่จำเป็นต้องมีการต่อใช้งานพอร์ตนี้กับระบบส่วนอื่นๆ ด้วยการเลือกใช้งาน LCD ให้เป็นโหมดส่งผ่านข้อมูล 4 บิตนี้ จึงเป็นวิธีช่วยลดสายสัญญาณของพอร์ตที่มีจำกัดให้เหลือใช้งานได้มากขึ้น จากรูปที่ 1 ที่ตำแหน่งขา 3 ของ K10 ถูกต่อเข้ากับตัวต้านทานปรับค่าได้เพื่อปรับความเข้มได้เหมาะสมกับสภาพแวดล้อมในขณะนั้นพร้อมทั้งความเข้มของแสงจากภายนอกและมุมมองของผู้ใช้งาน ก่อนการเชื่อมต่อ LCD กับ K10 ควรแน่ใจว่าตำแหน่งขาต่างๆ ในการเชื่อมต่อทั้งหมดถูกต้องแล้วจึงทำการเชื่อมตอจริง ถ้าไม่แน่ใจควรตรวจสอบจากดาต้าชีทให้แน่ใจอยู่เสมอ เพราะไม่เช่นนั้นอาจทำให้เกิดการเสียหายได้

2. คำสั่งควบคุมการทำงานของ LCD

คำสั่งควบคุมการทำงานของ LCD ที่ใช้ไมโครคอนโทรลเลอร์ของฮิตาชิตามดาต้าบุคนั้นมีความยาวประมาณ 30 หน้า สำหรับคำอธิบายครบทุกคุณสมบัติของ LCD ดังนั้นในที่นี้จะกล่าวถึงคำสั่งและการใช้งานทั้งหมดคงไม่ไหว หารนำเสนอคงทำได้เฉพาะสิ่งที่สำคัญ และควรทราบครอบคลุมการใช้งานทั่วไปเพียงเท่านั้น ถ้าท่านใดต้องการศึกษารายละเอียดของ LCD รุ่นใดก็หาได้จากดาต้าบุคจากผู้ผลิตหรือผู้จำหน่าย

ความสามารถหลักที่ LCD นิยมถูกหยิบยกมาใช้งานนั้น เพราะมันสามารถแสดงผลตัวอักขระแอสกีได้ ซึ่งมีอยู่ในไมโครคอนโทรลเลอร์ของ LCD ซึ่งแต่ละรุ่นก็มักใช้คอนโทรลเลอร์ตัวเดียวกันเสมอ นั่นคือคำสั่งควบคุมต่างๆจึงใช้เหมือนกันใน LCD แต่ละรุ่น

ก่อนที่จะเริ่มต้นศึกษารูปแบบคำสั่งควบคุมต่างๆ ผู้อ่านควรทราบถึงคำศัพท์บางคำเกี่ยวกับองค์ประกอบและการงานพื้นฐานของคอนโทรลเลอร์ใน LCD กันก่อน การทำงานภายใน LCD จะมีบัฟเฟอร์อยู่ภายใน ซึ่งมีความจุประมาณ 80 อักขระมีชื่อเรียกว่า DD-RAM (Display Data RAM) มีตำแหน่งแอดเดรสอยู่ระหว่าง 000H-04FH ยกตัวอย่างเช่น ถ้าเป็น LCD ขนาด 16 ตัวอักษร 1 บรรทัด DD-RAM ที่จัดเก็บข้อมูลขตัวอักขระที่จะแสดงผลทั้ง 16 ตัว จะถูกเก็บอยู่ที่ตำแหน่งแอดเดรส 00H-FH โดยเริ่มต้นจากด้านซ้ายของจอแสดงผล

การทำให้เกิดช่องว่างใน DD-RAM สามารถทำได้โดยการใช้คำสั่งชิฟตัวอักขระในการแสดงผล หรืออีกวิธีหนึ่งที่ง่ายมากขึ้นก็คือ ในขณะที่มีข้อมูลตัวอักขระเก็บไว้ในแอดเดรสค่าหนึ่ง การแสดงผลตัวอักขระต่อไปหรือทำให้เกิดช่องว่างอาจทำได้โดยการกำหนดจุดเริ่มต้นของแอดเดรสใหม่แตกต่างจากเดิมไป 1 แอดเดรสของ DD-RAM ซึ่งทำให้ได้ผลลัพธ์เช่นเดียวกัน

ผู้เขียนโปรแกรมหรือผู้ใช้งานสามารถทราบถึง การแสดงผลตัวอักขระต่อไปที่จะแสดงผลด้วยเคอเซอร์ ซึ่งเป็นตัวชี้ตำแหน่งต่อไปของอักขระไปยัง LCD คือทำให้การเลื่อนเคอเซอร์ตามไปด้วยเพื่อชี้ตำแหน่งของตัวอักษรต่อไป การเขียนโปรแกรมให้แสดงผลแบบนี้ทำให้ง่ายต่อการอ่านและเข้าใจของผู้ใช้งานได้มากที่สุด

ใน LCD มีหน่วยความจำส่วนหนึ่งที่สำคัญคือ CG-RAM ซึ่งทำหน้าที่เก็บข้อมูลรายละเอียดโครงร่างของตัวอักษรที่สามารถนำมาแสดงผลได้ หน่วยความจำในส่วนนี้ผู้ใช้งานสามารถที่จะสร้างตัวอักขระใดๆ ได้ตามต้องการแล้วเก็บลงในหน่วยความจำส่วนนี้ สำหรับเรียกมาใช้งานได้ด้วยถ้าหากตัวอักขระที่มีอยู่แล้วนั้นไม่เพียงพอต่อการใช้งาน การใช้งานในส่วนของ CG-RAM นี้สามารถดูได้จากดาต้าชีทของ LCD รุ่นนั้นๆที่ซื้อมาใช้งานได้

การส่งคำสั่งควบคุมไปยัง LCD สามารถกำหนดเป็นรหัสคำสั่งต่างๆ ซึ่งจะกล่าวต่อไปนี้ ออกที่ตำแหน่งแอดเดรส 0C001H โดยกำหนดให้ขาสัญญาณ RS มีค่าเป็น "0" และเซตโหมดให้เป็นการอ่านหรือเขียนได้ด้วยสัญญาณ R/W ต่อไปจะกล่าวถึงคำสั่งพื้นฐาน ที่ใช้งานบ่อยครั้งในการควบคุมการทำงานของ LCD สำหรับเครื่องหมายดอกจันทร์(*) แทนสถานะของบิตที่จะเป็นสถานะใดก็ได้ไม่สนใจและไม่มีผลกับคำสั่งนั้น

คำสั่งเคลียร์จอแสดงผล (Clear Display)

D7							D0
0	0	0	0	0	0	0	1

เมื่อคอนโทรลเลอร์ประมวลผลคำสั่งนี้จะทำให้ข้อมูลใน DD-RAM ถูกแทนที่ด้วยค่า 20H =Space ในรหัสแอสกี้ ทำให้จอแสดงผลไม่ปรากฏตัวอักขระใดๆ บนจอภาพ เคอเซอร์จะถูกเซตให้อยู่ในตำแหน่งเริ่มต้นใหม่อีกครั้ง และยกเลิกผลจากการใช้คำสั่งเลื่อนข้อมูลที่ผ่านมาแล้ว

คำสั่งเลื่อนเคอเซอร์ไปยังตำแหน่งเริ่มต้น (Return Home)

D7							D0
0	0	0	0	0	0	1	*

คำสั่งนี้มีผลให้รีเซตเคอเซอร์ให้กลับไปอยู่ที่ตำแหน่งเริ่มต้นใหม่ และรีเซตคำสั่งเลื่อนข้อมูลที่ผ่านมาแล้วโดยที่ข้อมูลตัวอักขระใน DD-RAM ไม่เกิดการเปลี่ยนแปลง นั่นคือตัวอักขระบนจอแสดงผลไม่เกิดการเปลี่ยนแปลง

คำสั่งกำหนดโหมดป้อนข้อมูล (Entry Mode Set)

D7							D0
0	0	0	0	0	1	I/D	S

คำสั่งนี้ถูกใช้ให้เป็นประโยชน์ในการกำหนดการทำงานของ LCD หลังจากเกิดการส่งข้อมูลไปยังจอแสดงผลบิต I/D (Increase/Decrease bit) ทำหน้าที่ในการกำหนดเพิ่ม (I/D=1) หรือลด (I/D=0) ค่าตำแหน่งแอดเดรสใน DD RAM 1 แอดเดรสอัตโนมัติเมื่อเกิดการอ่านหรือเขียนตัวอักขระ ค่าของตำแหน่งแอดเดรสนี้ถูกเก็บอยู่ในแอดเดรสเคาเตอร์ (AC) บิต S (Shift Bit) เป็นบิตที่ใช้กำหนดลักษณะการเลื่อนข้อมูลตัวอักขระอย่างอัตโนมัติ โดยถ้าบิต S=1 เมื่อมีการส่งไบต์ข้อมูลใหม่เกิดขึ้นตัวเคอเซอร์จะอยู่กับที่ แต่ตัวอักขระข้อมูลเดิมจะถูกดันไปทางซ้ายแต่ถ้าหากบิต S=0 เมื่อเกิดข้อมูลใหม่ตัวเคอเซอร์จะเลื่อนไปทางขวา

คำสั่งควบคุมการแสดงผล (Display On/Off Control)

D7							D0
0	0	0	0	1	D	C	B

คำสั่งนี้ใช้สำหรับควบคุมการแสดงผลในการปิดหรือเปิดการแสดงผลทางหน้าจอและเคอเซอร์ โดยปราศจากการเปลี่ยนแปลงข้อมูลใน DD RAM ผู้เขียนโปรแกรมสามารถกำหนดหน้าจอแสดงผลให้ปิดหรือเปิดได้ด้วยบิต D โดยกำหนดให้บิต D=1 เป็นการเปิดหน้าจอแสดงผลและถ้า D=0 เป็นการปิดจอแสดงผล เช่นเดียวกันสำหรับบิต C=1 และบิต C=0 เป็นการควบคุมให้เคอเซอร์ให้เปิดหรือปิดตามลำดับ และบิต B ซึ่งเป็นบิตกำหนดว่าจะให้เคอเซอร์กระพริบหรือไม่

คำสั่งควบคุมการเลื่อนเคอเซอร์และตัวอักขระ (Cursor or Display Shift)

D7							D0
0	0	0	1	S/C	R/L	*	*

คำสั่งนี้ใช้สำหรับการเลื่อนของเคอเซอร์และตัวอักขระที่แสดงผล ซึ่งมีความสำคัญและใช้งานบ่อยเนื่องจากผู้เขียนโปรแกรมต้องแสดงผลข้อความบนจอแสดงผลตามแนวนอน ผลที่เกิดจากการกำหนดสถานะของบิต S/C และ R/L

ตารางแสดงการกำหนดสถานะของบิต S/C และ R/L และการทำงานที่เกิดขึ้น

S/C	R/L	การทำงาน
0	0	เลื่อนเคอเซอร์ไปทางซ้าย
0	1	เลื่อนเคอเซอร์ไปทางขวา
1	0	เลื่อนตัวอักขระไปทางซ้าย พร้อมทั้งเคอเซอร์ด้วย
1	1	เลื่อนตัวอักขระไปทางขวา พร้อมทั้งเคอเซอร์ด้วย

คำสั่งเซตฟังก์ชัน (Function Set)

D7							D0
0	0	1	DL	N	F	*	*

คำสั่งนี้ใช้สำหรับเซตโหมดการทำงานของ LCD หลังจากที่รีเซตการทำงานของหรือเริ่มต้นการทำงานระบบทุกครั้ง ในที่นี้เราจะใช้งาน LCD ในโหมด 8 บิตและใช้งานเพียง 1 บรรทัดเท่านั้น ความหมายของบิตต่างๆมีดังนี้

บิต DL=1 หมายถึงทำงานในโหมดอินเตอร์เฟสแบบ 8 บิต

บิต DL=0 หมายถึงทำงานในโหมดอินเตอร์เฟสแบบ 4 บิต

บิต N= 1 หมายถึงการทำงานแบบ 2 บรรทัดหรือมากกว่า

บิต N= 0 หมายถึงการทำงานแบบ 1 บรรทัด

บิต F= 0 หมายถึงทำงานในโหมดความละเอียด 5x7 จุด

บิต F= 1 หมายถึงทำงานในโหมดความละเอียด 5x7 จุด

คำสั่งเซตตำแหน่งแอดเดรสใน CG-RAM (Set CG-RAM Address)

D7							D0
0	1	a5	a4	a3	a2	a1	a0

คำสั่งนี้ใช้สำหรับการกำหนดค่าตำแหน่งแอดเดรสใน CG-RAM ภายใน LCD เพื่อทำการส่งผ่านข้อมูลโดยกำหนดตำแหน่งแอดเดรสใน CG-RAM แน่นอนค่าหนึ่งเพื่อถ่ายทอดไบต์ข้อมูลต่อไป บิต a0-a5 เป็นบิตที่กำหนดตำแหน่งแอดเดรสใน CG-RAM ซึ่งจะถูกโหลดไปเก็บไว้ในแอดเดรสเคาเตอร์ AC

คำสั่งเซตตำแหน่งแอดเดรสใน DD-RAM (Set DD-RAM Address)

D7							Do
1	a6	a5	a4	a3	a2	a1	a0

คำสั่งนี้ใช้สำหรับกำหนดค่าตำแหน่งแอดเดรสใน DD-RAM ภายใน LCD เพื่อทำการส่งผ่านข้อมูลโดยกำหนดตำแหน่งแอดเดรสใน DD-RAM แน่นอนค่าหนึ่งเพื่อถ่ายทอดไบต์ข้อมูลต่อไป บิต a0-a6 เป็นบิตกำหนดตำแหน่งแอดเดรสใน DD-RAM ซึ่งถูกโหลดไปเก็บไว้ในแอดเดรสเคาเตอร์ AC

คำสั่งอ่านแฟลกบิวซี (Read Busy Flag)

D7							Do
BF	a6	a5	a4	a3	a2	a1	a0

เมื่อต้องการอ่านสถานะของแฟลกบิวซี ต้องกำหนดขา R/W เป็น "1" เสมอ แฟลกบิวซีเป็นตัวบอกสถานะการทำงานของ LCD ว่าอยู่ในสถานะพร้อมรับข้อมูลหรือไม่ โดยถ้า BF=1 หมายถึงขณะนั้น LCD พร้อมที่จะรับข้อมูลต่อไป อันเนื่องจากกระบวนการประมวลผลคำสั่งหรือไบต์ข้อมูลที่ผ่านมายังไม่เสร็จสิ้นแต่ถ้า BF=0 หมายถึงขณะนี้ LCD พร้อมที่จะรับคำสั่งใหม่หรือข้อมูลใหม่ได้แล้ว นอกจากคำสั่งนี้ทำให้ทราบสถานะของแฟลกบิวซีแล้วในขณะเดียวกันที่บิต a0-a6 จะถูกอ่านด้วยซึ่งเป็นค่าที่เก็บอยู่ในแอดเดรสเคาเตอร์ AC

คำสั่งเขียนข้อมูลไปยัง CG หรือ DD-RAM

D7							Do
d7	d6	d5	d4	d3	d2	d1	d0

เมื่อต้องการใช้คำสั่งนี้ต้องกำหนดให้ขาสัญญาณ R/w =0 และ RS=1 เสมอ คำสั่งนี้ใช้สำหรับเขียนข้อมูลไปยัง CG หรือ DD-RAM ในตำแหน่งแอดเดรสที่ต้องการซึ่งขึ้นอยู่กับการกำหนดตำแหน่งแอดเดรสใน CG หรือ DD-RAM ก่อนหน้าที่จะส่งข้อมูลนี้แล้ว คำสั่งนี้จะมีผลเกี่ยวเนื่องถึงการกำหนดโหมดการทำงานก่อนหน้านี้ด้วยว่า จะให้ทำการเพิ่มหรือลดค่าใน AC หลังจากส่งผ่านไบต์ข้อมูลแล้ว ซึ่งเป็นผลมาจากคำสั่งกำหนดโหมดป้อนข้อมูล (Entry Data CG or DD-RAM)

คำสั่งอ่านข้อมูลจาก CG หรือ DD-RAM

D7							Do
d7	d6	d5	d4	d3	d2	d1	d0

เมื่อต้องการใช้คำสั่งนี้ต้องกำหนดให้ขาสัญญาณ R/w=1 เสมอ คำสั่งนี้ใช้สำหรับอ่านข้อมูลจาก CG หรือ DD-RAM ในตำแหน่งที่แอดเดรสต้องการขึ้นอยู่กับการกำหนดตำแหน่งแอดเดรสใน CG หรือ DD-RAM ก่อนหน้าที่จะใช้คำสั่งอ่านข้อมูลแล้ว

เข้าใจคำสั่งควบคุมการทำงานของ LCD โมดูลมาพอสังเขปแล้ว ทีนี้ถึงเวลาปฏิบัติจริงกันบางตัวอย่างด้วยโปรแกรมทดสอบจอแสดงผล LCD โมดูล ช่วยเสริมความรู้ ความเข้าใจ ในการใช้งานคำสั่งต่างๆอีกครั้ง

3. โปรแกรมทดสอบ LCD โมดูล โปรแกรมทดสอบ LCD โมดูลนี้เน้นการใช้งานคำสั่งควบคุมLCD โมดูลมาใช้ในการเขียนโปรแกรม โดยโปรแกรมนี้จะทำการเขียนตัวอักษรไปยัง LCD โมดูลให้แสดงผลบนหน้าจอหลังจากนั้นก็ควบคุมให้เลื่อนตัวอักษรเหล่านั้นไปทางซ้ายหรือทางขวาด้วยการกดปุ่ม ตัวอย่างโปรแกรมนี้คือไฟล์ XAMPLE12.A51 ที่มีอยู่ในแผ่นดิสก์แล้ว โดยการเขียนเป็นโฟลว์ชาร์ตการทำงาน จากไฟล์เอกสารของโปรแกรม XAMPLE12 จะเห็นได้ว่ามีการใช้ซับรูทีนจะกล่าวในหัวข้อต่อไป

4. การทำงานของซับรูทีน เริ่มต้นที่ซับรูทีน RCOM ทำหน้าที่อ่านค่าสถานะการทำงานของ LCD โมดูลไปเก็บไว้ในแอคคิวมูเลเตอร์ในซับรูทีนนี้จำเป็นต้องติดต่อกับอุปกรณ์อินพุตเอาท์พุตนั่นคือ LCD โมดูลซึ่งอยู่ตำแหน่งแอดเดรสของหน่วยความจำภายนอก ดังนั้นจึงต้องใช้การอ้างแอดเดรสทางอ้อม ในการเข้าถึงโดยการกำหนดไบต์แอดเดรสสูง ไปยังพอร์ต P2 ด้วยค่า 00H และการกำหนดค่าไบต์ต่ำ เก็บไว้ในรีจิสเตอร์ Ro หลังจากนั้นจึงใช้คำสั่ง MOVX ทำการอ่านแฟลกบิวซีของ LCD โมดูลปรากฏเป็นสถานะของของบิตที่ 7 ในแอคคิวมูเลเตอร์ ทั้งนี้เพื่อตรวจสอบความพร้อมในการรับคำสั่งและข้อมูลต่อไปของ LCD โมดูล

ซับรูทีน LCDRDY ทำหน้าที่ตรวจสอบแฟลกบิวซีที่อ่านสถานะได้จากซับรูทีน RCOM โดยมันจะคอยจนกระทั่งแฟลกบิวซีมีค่าเป็น "0" ก่อนที่จะทำงานต่อไปซับรูทีนนี้มีผลทำให้ไมโครคอนโทรลเลอร์และ LCD โมดูลทำงานอย่างควบคู่กันและมั่นใจได้เสมอว่า LCD โมดูลพร้อมรับคำสั่งได้อย่างสมบูรณ์

ซับรูทีน WT1 เป็นซับรูทีนหน่วงเวลาประมาณ 100 ไมโครวินาที เพราะว่าการประมวลผลคำสั่งของ LCD โมดูล เกือบทุกยี่ห้อมักมีค่าต่ำกว่า 100ไมโครวินาที ซับรูทีนนี้จะถูกเรียกใช้เพื่อคอยให้ LCD โมดูลประมวลผลคำสั่งได้เสร็จสิ้นก่อนที่จะทำงานต่อไป

ซับรูทีน LCDCOM ทำหน้าที่ส่งคำสั่งควบคุมไปยัง LCD โมดูล โดยกำหนดให้ขาสัญญาณ RS=0 การอ้างตำแหน่งแอดเดรสไปยังหน่วยความจำ LCD โมดูลใช้คำสั่ง MOVX เช่นเดียวกับที่มีใช้ในซับรูทีนแล้วโปรแกรมจะเข้าสู่ซับรูทีนหน่วงเวลา 100 ไมโครวินาทีต่อไป

ซับรูทีน LCDCHR ทำหน้าที่ส่งข้อมูลตัวอักขระไปยังหน่วยความจำ DD-RAM ภายใน LCD โมดูล โดยการกำหนดให้ขาสัญญาณ RS=1 การอ้างตำแหน่งแอดเดรสไปยัง LCD โมดูลยังคงใช้คำสั่ง MOVX เช่นเดียวกัน หลังจากนั้นจึงทำการหน่วงเวลา 100 ไมโครวินาที

ซับรูทีน LCDSET ทำหน้าที่กำหนดโหมดควบคุมการทำงานของ LCD โมดูล การทำงานในซับรูทีนนี้ขั้นแรก จะทำการเรียกซับรูทีน LCDRDY เพื่อตรวจสอบให้แน่ใจก่อนว่าคำสั่งครั้งก่อน ที่ส่งไปยัง LCD โมดูลถูกประมวลผลเสร็จสิ้นแล้ว ขั้นต่อไปโปรแกรมจะเซ็ตโหมดการทำงานของ LCD โมดูลให้เป็นแบบ 8 บิต, 1 บรรทัด และความละเอียด 5x7 จุด โดยการส่งคำสั่งควบคุมผ่านซับรูทีน LCDCOM หลังการกำหนดโหมดการทำงานพื้นฐานแล้ว โปรแกรมกำหนดให้เปิดจอแสดงและปิดการแสดงผลเคอเซอร์ในบรรทัดที่ 39 และ 40 และกำหนดโหมดการเลื่อนต่อไปในบรรทัดที่ 41 และ 42 ต่อจากนั้นโปรแกรมจะส่งให้เคลียร์หน้าจอแสดงผล ซึ่งคำสั่งนี้อาจต้องให้เวลาในการประมวนผลนานถึงประมาณ 1.6 มิลลิวินาที ดังนั้นหลังจากใช้คำสั่งควบคุมแล้วจึงต้องเรียกใช้ซับรูทีน LCDRDY มาใช้ในการตรวจสอบความพร้อมของ LCD โมดูล ก่อนที่จะส่งคำสั่งต่อไปด้วยจ

จากไฟล์เอกสารที่นำมาเรียนรู้กันในนี้ ถ้าสังเกตกันดีๆบางท่านอาจสงสัยว่า ทำไมบางซับรูทีนไม่เห็นมีคำสั่ง RET ปิดท้านเลยแล้วคำสั่งจะทำงานได้หรือ คำตอบก็คือมันเป็นเคล็ดลับ ในการเขียนโปรแกรมแบบหนึ่งซึ่งใช้ในกรณีที่มีการเรียกใช้ซับรูทีนต่อกันหลายซับรูทีน เช่นเมื่อโปรแกรมทำงานตามคำสั่งในซับรูทีนที่ 1 จนกระทั่งถึงบรรทัดสุดท้ายก่อนจบรูทีนนี้กลับมีการเรียกอีกซับรูทีนหนึ่ง ซึ่งเป็นซับรูทีนที่ 2 คำสั่งที่ใช้เขียนจะเป็นรูปแบบนี้คือ

```
:
LCALL subroutine 2
RET
:
```

จะเห็นได้ว่าต้องใช้คำสั่งจำนวน 2 บรรทัด ในส่วนนี้สามารถเขียนแทนได้โดยใช้คำสั่งเพียงบรรทัดเดียว คือ

```
LJMP subroutine 2
```

หลังจากกระโดดไปทำงานในซับรูทีนที่ 2 แล้ว คำสั่ง RET ซึ่งเกิดขึ้นจากการเรียกซับรูทีนที่ 1 ยืมคำสั่ง RET ของซับรูทีนที่ 2มาใช้ ถึงแม้ว่าเคล็ดลับการเขียนโปรแกรมนี้จะช่วยประหยัดคำสั่งและพื้นที่หน่วยความจำของโปรแกรมได้เพียงไม่กี่บรรทัด แต่ว่าการเขียนแบบนี้จะช่วยลดความสับสนอันเกิดจากการมีคำสั่ง RET มากเกินไป

บทที่ 3

การออกแบบและการสร้าง

จุดประสงค์

1. นำความรู้ที่ได้ศึกษามาจากทฤษฎีของไมโครคอนโทรลเลอร์มาประยุกต์ใช้กับงานปฏิบัติจริง
2. เพื่อศึกษาและทดลองเขียนโปรแกรมเพื่อควบคุมอุปกรณ์ที่เป็นแม็คคานิกส์ ให้ทำงานตามคำสั่งของโปรแกรม
3. เพื่อเป็นประโยชน์ต่อการนำไปใช้ในการเลี้ยงปลา เพื่อช่วยในด้านการให้อาหารอย่างเหมาะสมทั้งปริมาณและเวลา
4. เพื่อให้ผู้ใช้สามารถนำระบบนี้ไปใช้งานได้โดยไม่จำเป็นต้องมีความรู้ทางด้านนี้

แนวคิดในการสร้าง

เริ่มแรกในการคิดสร้างโครงงานวิจัยนี้ เดิมทีอยู่ในหัวข้อเครื่องป้อนอาหารสัตว์อัตโนมัตินั้น ทางผู้จัดทำได้ปรึกษากับอาจารย์ที่ปรึกษาถึงชนิดของสัตว์ ที่เราจะสร้างเครื่องมือตัวนี้ขึ้นมา เนื่องจากผู้จัดทำได้ทราบมาว่า มีนักศึกษาของสถานอุดมศึกษาบางแห่งได้ทำโครงงานวิจัย เครื่องป้อนอาหารสัตว์นี้ขึ้นมาสำหรับสุนัข โดยใช้วงจรดิจิตัลแบบธรรมดา และใช้ไอซีหน่วยความจำบันทึกคำสั่งในการสั่งการการทำงาน เป็นการถาวรแต่มิได้มีการออกแบบเพื่อให้ผู้ใช้สามารถเปลี่ยนแปลงการตั้งเวลาในการให้อาหาร หลังจากที่ได้ปรึกษากับอาจารย์ที่ปรึกษาแล้ว จึงได้รับความกรุณาจากอาจารย์ให้สร้างเครื่องป้อนอาหารสำหรับปลา และการนำไปใช้งานนั้นโดยให้สร้างเงื่อนไขที่ผู้ใช้สามารถเลือกกำหนดจำนวนครั้งและเวลาในการให้อาหารได้ โครงงานนี้จึงมิได้มีขนาดที่ใหญ่เนื่องจากเป็นการป้อนอาหารสำหรับปลาที่เลี้ยงไว้ดูเล่นในบ้าน จึงมีขนาดเล็กพอเหมาะกับขนาดของตู้ปลา แต่ปัญหาในการสร้างนั้นสิ่งที่เป็นอุปสรรคมากคือกลไก ที่นำมาใช้ในการประกอบเป็นแม็คคานิกส์ ที่มีขนาดที่เล็กและหาซื้อยาก ต้องทดลองและแก้ไขหลายครั้งในการที่จะให้แม็คคานิกส์ทำงานได้สัมพันธ์กับโปรแกรม

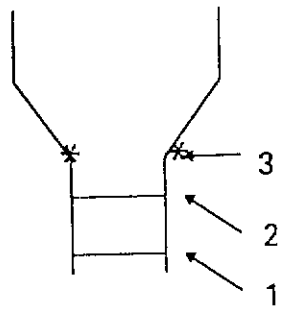
ตัวไอซี ซีพียูที่ใช้ในการบันทึกโปรแกรมนั้นได้ใช้ซีพียูตระกูล MCS51 ซึ่งเป็นที่นิยมใช้ในการควบคุมในยุคปัจจุบันมาก แต่ในช่วงการทดลองโปรแกรมนั้นได้ใช้ไอซี DS5000T เป็นตัวทดลองบันทึกโปรแกรมเนื่องจาก ความสะดวกในการลบข้อมูลด้วยไฟฟ้าและสามารถบันทึกซ้ำได้ถึง 1,000 ครั้ง เมื่อโปรแกรมทำงานได้ถูกต้องแล้วจึงบันทึกลงบนซีพียู MCS51 เป็นการถาวรซึ่งมีคุณสมบัติเป็น EPROM ซึ่งมีราคาที่ถูกกว่ามาก เพราะฉะนั้นถ้าหากนำไปใช้ในการสร้างขึ้นในทางการค้าจะสามารถสร้างได้จำนวนมากในราคาที่ไม่แพง

อุปกรณ์ที่ใช้ในการตรวจจับกรณีที่อาหารหมดนั้น เดิมทีผู้สร้างได้ใช้เป็นกลไกแม็คคานิกส์ แต่ประสบปัญหาเรื่องชิ้นส่วน จึงหันมาใช้อินฟราเรดและโฟโต้ไดโอดตรวจจับเพื่อสะดวกในการควบคุมที่แน่นอน

การตั้งเวลาในการให้อาหารนั้นได้สร้างโปรแกรมให้ผู้ใช้สามารถตั้งโปรแกรมได้สูงสุด 4 ครั้งต่อวัน เนื่องจากผู้สร้างได้หาความรู้จากผู้ที่สันทัดเรื่องการเลี้ยงปลาตู้ และผู้จำหน่ายพันธุ์ปลา ซึ่งโดยมากแล้ว มักจะกำหนดเวลาให้อาหารปลาวันละ 2-3 ครั้ง เพราะถ้าให้บ่อยเกินไปจะทำให้อาหารเหลือ น้ำเสียเร็วและปลาอาจจะกินอาหารมากเกินไปจนตายไว หรือมีสีสันที่ไม่สวยงาม เครื่องป้อนอาหารปลาอัตโนมัติชุดนี้เหมาะสำหรับเลี้ยงปลา จำนวน 5-30 ตัวต่อ 1 ชุดป้อน ทั้งนี้ขึ้นอยู่กับขนาดของตู้ ชนิดและขนาดของปลา และจำนวนครั้งในการให้อาหาร หากจำนวนปลามีจำนวนมาก ผู้ใช้ควรตั้งเวลาในการให้มากขึ้น ฉะนั้นผู้นำชุดเครื่องมือป้อนอาหารปลานี้ไปใช้ ควรเป็นผู้ที่มีความรู้เรื่องการเลี้ยงปลาด้วย เพื่อที่จะให้เกิดประโยชน์สูงสุดในการนำไปใช้

คุณสมบัติโดยรวม

1. สำหรับเครื่องป้อนอาหารปลาชุดนี้ออกแบบให้ใช้กับเครื่องป้อน 2 ตัว
2. มีช่องเปิดปิดอาหารโดยอาศัยกลไกของโซลินอยด์ที่ถูกควบคุมจาก CPU
3. มีเซ็นเซอร์อินฟราเรดสำหรับตรวจจับกรณีมีอาหารเหลือน้อย โดยจะมีไฟเตือนบอกในแต่ละชุดแยกกัน
4. มีสวิทช์คีย์แม็ทริกซ์ 4x3 เพื่อให้ผู้ใช้สามารถติดต่อและกำหนดเวลาในการป้อนอาหารและสามารถทำการเปลี่ยนแปลงได้ตลอด
5. มีจอแสดงผล LCD 16 ตัวอักษร 1 บรรทัด เพื่อบอกเวลาจริงและข้อมูลเวลาที่ผู้ใช้กำหนด
6. มีวงจรสร้างสัญญาณนาฬิกาเพื่อใช้กำหนดเวลาเงื่อนไข และส่งสัญญาณนาฬิกา เข้าไปควบคุมจังหวะการทำงานของ CPU
7. มีวงจร Back up กรณีที่ไฟฟ้าดับหรือ Vcc ตก ซึ่งใช้แบตเตอรีลิเทียม ขนาด 3.6 โวลท์ เป็นไฟเลี้ยงฉุกเฉิน



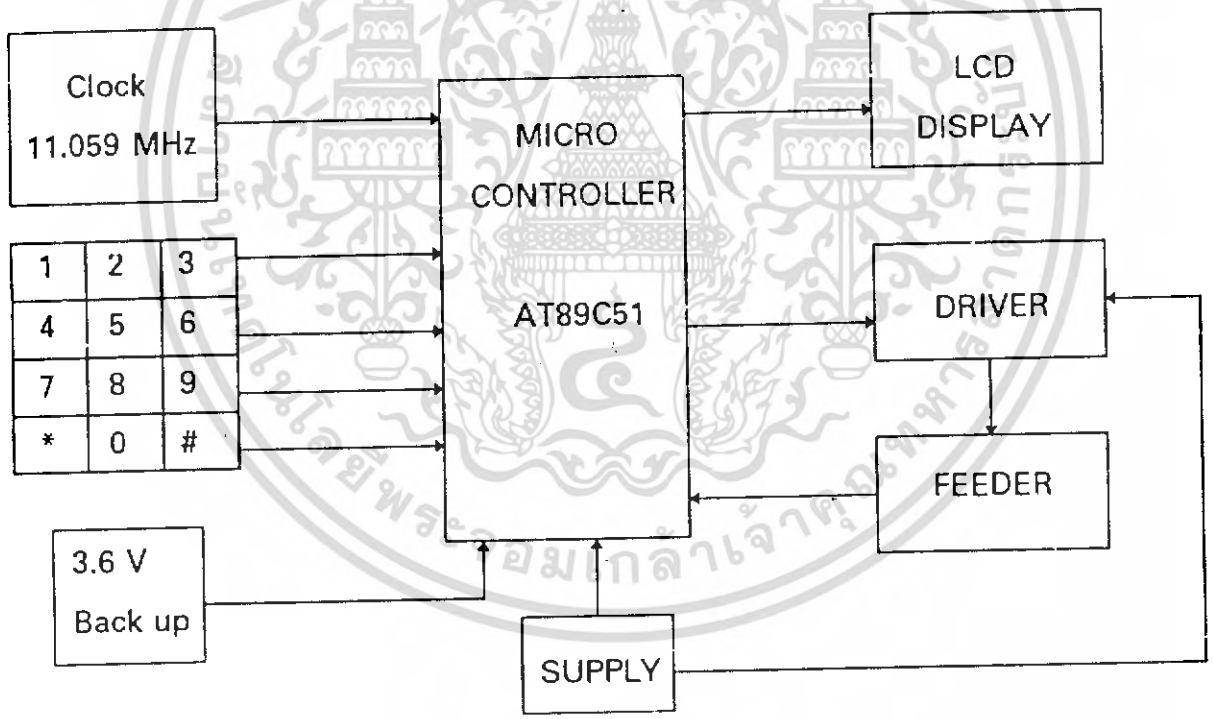
รูปแสดงชุดอุปกรณ์ป้อนอาหาร และส่วนประกอบที่สำคัญ

1 คือ ตำแหน่งของโซลินอยด์ตัวล่าง

2 คือ ตำแหน่งของโซลินอยด์ตัวบน

3 คือ ตำแหน่งของเซ็นเซอร์อินฟราเรดตรวจสอบอาหารหมด

ขอบเขตของโครงงาน



อุปกรณ์ที่ใช้

1) ไอซี AT89C51

2) บอร์ด ANT-C51 (พร้อมอุปกรณ์)

3) DS 5000T (ใช้ในการพัฒนาโปรแกรม)

4) ไอซี MAX691

5) ไอซี DS 1202

6) LCD 16 ตัวอักษร 1 บรรทัด

7) Switch PB

8) Crystal 11.059 MHz

9) Resister ปรับค่าได้ 10k โอห์ม

10) KEY MATRIX 4x3

11) LED สีเขียว สีแดง

12) แบตเตอรีลิเทียม 3.6 โวลท์

13) Resister 10k,1k,330 โอห์ม

14) คาปาซิเตอร์ 10 pF

15) เทอร์มินอล

16) คอนเน็คเตอร์

17) ทรานซิสเตอร์ไดรเวอร์

18) ไดโอดเร็กติฟรายเออร์

19) โซลินอยด์ 12โวลท์ ดีซี

20) หลอดอินฟราเรด

21) โฟโต้ไดโอด

22) ตัวต้านทาน 500 โอห์ม

23) ข้อต่อท่อเอสล่อน,กาว

<u>เครื่องมือที่ใช้</u>

1) Personal Computer

2) ไอซี Burner

3) มิเตอร์

4) หัวแร้ง,ที่ดูดตะกั่ว,ตะกั่วบัดกรี

5) คีมตัด,คีมจับ,เลื่อย

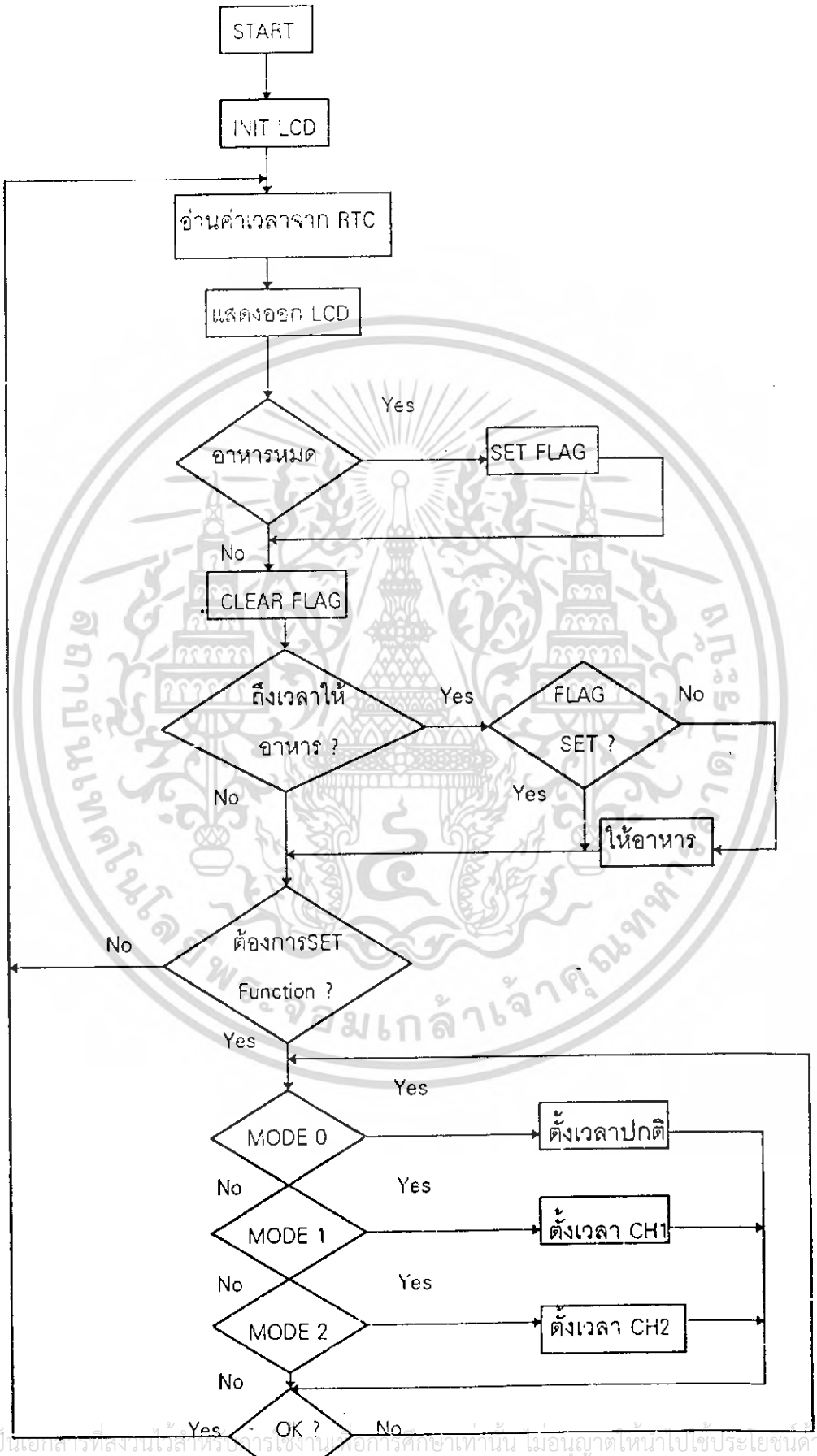
6) สายไวร์แร็บ

ขั้นตอนการสร้าง

แบ่งออกเป็น ๙ ขั้นตอน

1) **ส่วนของ** Central Process **ซึ่งใช้ไอซี** AT89C51 **เป็นซีพียูนั้น ในขั้นแรกเราจำเป็นต้องใช้ไอซี** DALLAS DS 5000T **เป็นตัวทดลองโปรแกรมที่เขียนขึ้นมาก่อน เนื่องจากเป็นไอซีที่มีโปรแกรมสำหรับการสื่อสารระหว่างคอมพิวเตอร์กับตัวไอซี ซึ่งสามารถทำการเขียนและอ่านได้** และสามารถลบข้อมูล (Burn) ได้ถึง 1,000 ครั้ง หลังจากทำการทดลองโปรแกรมที่เขียนขึ้นมาก่อน หลังจากหยุดพัฒนาโปรแกรมแล้วจึงทำการอ่านเพื่อลอกเลียนแบบได้ และมีราคาที่ไม่สูงมากนัก การติดต่อระหว่างซีพียูของระบบคอมพิวเตอร์นั้นเราใช้บอร์ด ANT-C51 ซึ่งมีไอซี MAX232 เป็นตัวอินเตอร์เฟสการรับส่งข้อมูลจาก PC ลงบน CPU
2) ต่อ LCD ขา 7-14 เข้ากับขา P1.0-1.7 และขา 4,5,6 เข้ากับขา P2.5-P2.7 ของซีพียู โดยต่อความต้านทานปรับค่าได้ 10K โอห์ม เข้าที่ขา 1,2,3 ของ LCD เพื่อปรับความเข้มของหน้าจอ
3) ต่อแมทริกซ์คีย์ขนานกับความต้านทาน 10K เข้ากับขา P3.0-P3.7 ของซีพียู ซึ่งใช้ในการต่อแบบ Active High ซึ่งได้เขียนโปรแกรมการตรวจสอบคีย์ไว้แล้ว
4) การต่อภาคไดรว์เวอร์เพื่อใช้ในการขับชุดป้อนอาหาร โดยทรานซิสเตอร์ไดรเวอร์ต่อในลักษณะคอมมอนอิมิตเตอร์ทำหน้าที่เป็นสวิทช์ปิด-เปิด โดยใช้สัญญาณขาออกของซีพียูเป็นตัวทริกที่ขาเบส เอาท์พุตที่ได้ทางขา C ซึ่งต่อไฟเลี้ยง 12 โวลท์ผ่าน R 500 โอห์ม
5) ต่อชุดอินฟราเรดและโฟโต้ไดโอดเพื่อเช็คปริมาณอาหารหมดเข้ากับซีพียู และต่อเข้ากับ LED เพื่อเป็นตัวแจ้งเตือน
6) ต่อวงจรสัญญาณนาฬิกาโดยใช้ไอซี DS1202 ซึ่งรับความถี่จาก CRYSTAL 11.059MHz เป็นตัวสร้างจังหวะการทำงาน
7) ต่อวงจรแบ็กอัพโดยใช้ไอซี MAX 691 และต่อถ่านลิเทียม 3.6 โวลท์ ที่ขา VB โดยให้ V0 ของ MAX691 ต่อเข้ากับขา Vcc ของ DS1202

FLOW CHART การทำงานของโปรแกรม



บทที่ 4 ผลการทำงานของแต่ละภาค

1. ภาคประมวลผลกลางหรือ CPU ทำหน้าที่ในการนำคำสั่งที่เก็บไว้ในรอมออกมาปฏิบัติตามเงื่อนไขของโปรแกรมภาษาแอสเซ็มบลี้ ซึ่งในการทดลองโปรแกรมนี้ใช้ชิพ DS 5000T ในการติดต่ออ่านและเขียนข้อมูลกับคอมพิวเตอร์ ใช้ในการติดต่อแบบ Full Duplex ซึ่งมีไอซีเบอร์ MAX232 เป็นมาตรฐานการสื่อสารแบบอนุกรม (RS232)
2. ภาคแสดงผลใช้ LCD ในการ Display ค่าต่างๆที่ต้องการแสดง และค่าที่ได้รับจากคีย์มี R ปรับค่าได้ทำหน่าที่ปรับความเข้มของหน้าจอ
3. ภาค คีย์แมทริกซ์ ใช้สวิทช์โทรศัพท์กดติดปล่อยดับ และมีโปรแกรมในการตรวจสอบการกดคีย์แต่ละตัว
4. ภาคสัญญาณนาฬิกาใช้ DS1202 มี Crystal 11.059 MHz เป็นความถี่มาตรฐานเพื่อกำเนิดสัญญาณพัลส์เข้าไปควบคุมจังหวะการทำงานของ CPU
5. ภาคแบ็คอัพใช้แบตเตอรีลิเทียม 3.6 โวลท์ ต่อเข้ากับ MAX691 ซึ่งใช้ Vcc 5V การทำงานคือเมื่อ Vcc ยังมีค่ามาก Vbatt Voutจะเท่ากับ Vcc แต่ถ้า Vcc ลดต่ำลงกว่า Vbatt Vout จะต่อเข้ากับ Vbatt วงจรนี้สร้างขึ้นเพื่อเป็นไฟเลี้ยงฉุกเฉินให้กับ CPU เพื่อไม่ให้การทำงานสะดุดผิดพลาด
6. ภาคไดรเวอร์ซึ่งทำหน้าที่เป็นสวิทช์ในการนำเอาสัญญาณขาออก ของซีพียู ซึ่งสัญญาณ"1" คือระดับไฟ 5 โวลท์แทนการเปิดสวิทช์ และสัญญาณ "0" (0 โวลท์) แทนการปิดสวิทช์ไปบังคับการทำงานของโซลินอยด์เพื่อเปิดช่องอาหาร เหตุที่ต้องมีภาคไดรเวอร์เป็นเพราะเอาท์พุตของซีพียูเพียงแค่ 5 โวลท์ ไม่พอสำหรับการขับโซลินอยด์ที่ใช้ไฟ 12 โวลท์
7. ภาค SUPPLY ซึ่งทำหน้าที่เป็นไฟเลี้ยงของทุกวงจร ในโครงงานชุดนี้จะมีชุดไฟเลี้ยง 2 ชุด ชุดแรกให้เอาท์พุต 5 โวลท์เพื่อเป็นซัพพลายให้แก่ซีพียูและวงจรประกอบ ส่วนอีกชุดหนึ่ง นำเป็นไฟเลี้ยงสำหรับขับโซลินอยด์ขนาด 12 โวลท์ โดยใช้ทรานฟอร์มเมอร์ 220/24V AC เป็นตัวแปลงไฟ ผ่านไดโอดแบบบริดจ์ มีซีเนอร์ไดโอดเป็นตัวควบคุมแรงดันเกิน

การใช้งาน

เมื่อต้องการใช้งาน ผู้ใช้ต้องทำการตั้งเวลาจริงและเวลาที่ต้องการป้อนอาหารของแต่ละชุด ซึ่งโปรแกรมนี้เขียนไว้ให้ตั้งได้ 4 เวลา ซึ่งเป็นจำนวนครั้งที่ปลาต้องการกินอาหาร ถ้าต้องการลดเวลาในการให้อาหารสามารถทำได้เลย โดย

ไม่ต้องแก้ไขใดๆ เพียงแต่โปรแกรมเวลาซ้ำลงไป จำนวนครั้งจะลดไป 1 ครั้ง ดังนั้นจึงสามารถลดจำนวนได้เองเป็น 3,2,1 ครั้ง

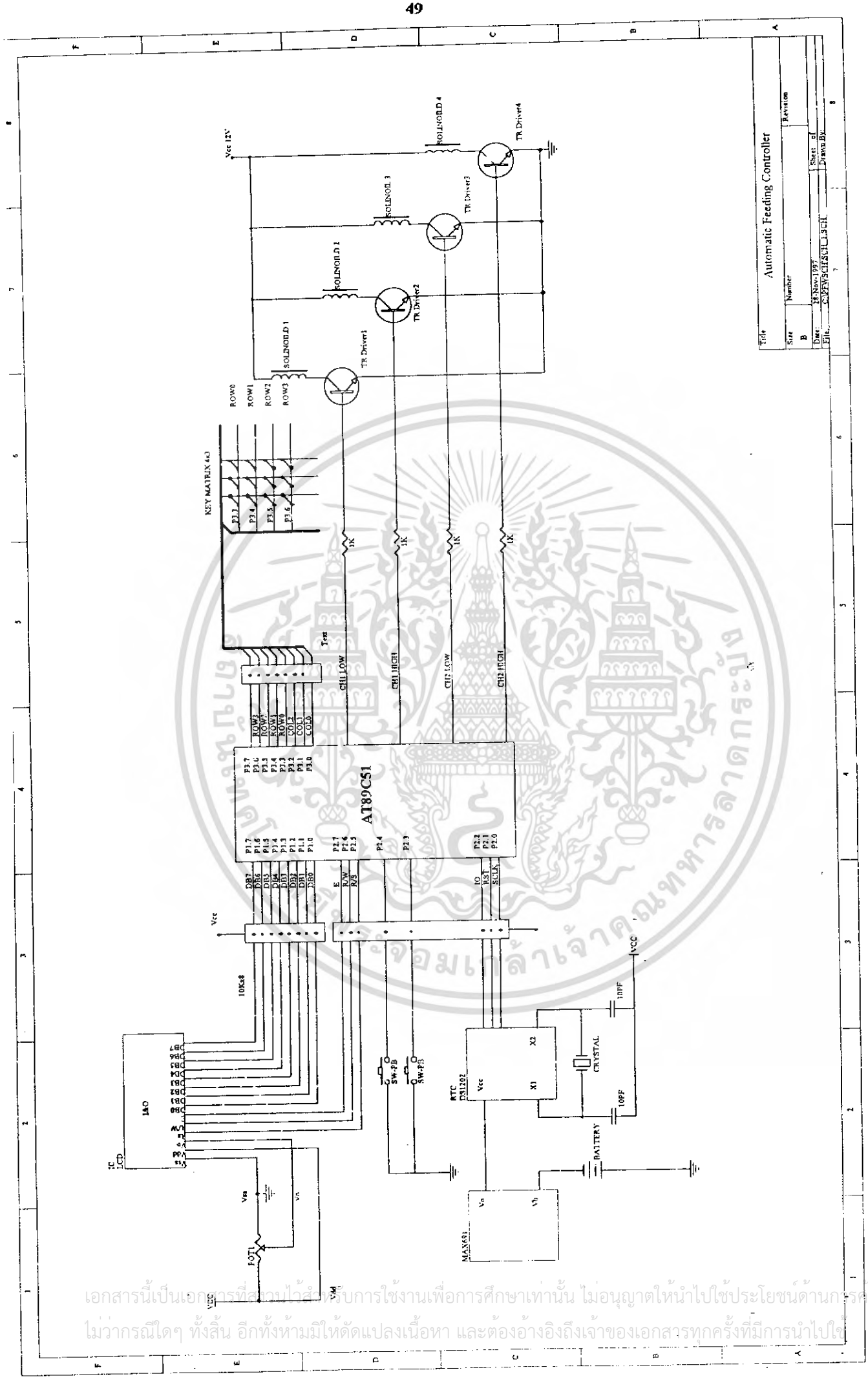
วิธีตั้งเวลา

1) เมื่อต้องการตั้งเวลาจริงให้กดปุ่มใดๆ จะขึ้นหน้าจอว่า SET MODE ให้กด ๐ หน้าจอจะแสดงว่า INPUT HOUR ให้กดเวลาเป็นชั่วโมงจริงลงไป แล้วยืนยันด้วย # หน้าจอจะแสดงว่า INPUT MINUTE ให้กดเลขนาทีแล้วยืนยันด้วย # การตั้งวินาทีทำเช่นเดียวกัน
2) เมื่อต้องการตั้งเวลา Chanel 1 กดปุ่มใดๆจะขึ้นหน้าจอว่า SET MODE ให้กด 1 จะว่า TIME1 CH1 สักครู่จะขึ้นแสดงว่า INPUT HOUR เมื่อตั้งเวลาของ TIME1 แล้ว หน้าจอจะขึ้น TIME2 CH1 ให้ทำต่อในลักษณะเดียวกัน เพื่อให้ได้จำนวนครั้งที่ต้องการ
3) ถ้าต้องการตั้งเวลา CH2 กดปุ่มใดๆแล้วกด 2 หน้าจอจะขึ้นว่า TIME1 CH2 ให้ตั้งเวลาเช่นเดิม ถ้าต้องการตั้งเวลาที่ 2 CH2 ให้กด 2 อีกครั้งหนึ่งแล้วตั้งเวลา

เมื่อถึงเวลาให้อาหารของ Chanel ใด เวลาใด CPU จะส่งสัญญาณไปที่ภาคไดรเวอร์ เพื่อทริกสัญญาณนำไปขับโซลินอยด์ตัวล่างให้ทำการเปิดช่องอาหารเป็นเวลา 3 วินาทีแล้วปิด จากนั้นโซลินอยด์ตัวบนจะเปิดเพื่อให้อาหารไหลลงมาที่ช่องกัก หากอาหารหมด อินฟราเรดจะตรวจเช็คแล้วให้สัญญาณเป็น "1" ทำให้ LED แสดงผลติด

สรุป

การสร้างโครงงานวิจัยนี้เป็นประโยชน์ในการนำไปใช้และสร้างจำนวนมาก เพื่อจำหน่าย เนื่องจากว่าเอื้อประโยชน์แก่ผู้ใช้ในการที่จะเปลี่ยนแปลงเวลา และจำนวนครั้งในการให้อาหารแก่ปลา โดยเฉพาะตัวโปรแกรมเมื่อพัฒนาได้เสร็จเรียบร้อยแล้ว ก็สามารถที่จะทำการ Burn ลงบนไอซีอีพรอมเพื่อสร้างจำนวนมากๆได้ ปัญหาอุปสรรคโดยส่วนใหญ่อยู่ที่ภาคแม็คคานิกส์ ซึ่งต้องสร้างให้มีขนาดเล็กและเครื่องมือต้องมีความพร้อมมากกว่านี้ แต่ถ้ามีการนำไปผลิตภาคอุตสาหกรรมแล้ว ปัญหานี้จะหมดไปทำให้ค่าใช้จ่ายต่อเครื่องต่ำ แต่เนื่องจากงานนี้เป็นโครงงานวิจัย ดังนั้นค่าใช้จ่ายสำหรับการพัฒนาโปรแกรมค่อนสูง โดยเฉพาะซีพียู DS5000T ซึ่งเป็นไอซี EEPROM ซึ่งใช้ในการพัฒนาโปรแกรมมีราคาแพง จึงเหมาะสำหรับงานวิจัยเท่านั้น



บทที่ 5
สรุปผลการดำเนินการและข้อเสนอแนะ

เครื่องควบคุมการป้อนอาหาร(ปลา)อัตโนมัติ ที่ได้สร้างขึ้นมานี้ จะเป็นประโยชน์ในแนวทางที่จะนำไปพัฒนาในการควบคุมป้อนอาหารสัตว์ชนิดอื่นๆ ได้อย่างกว้างขวางโดยเฉพาะการประยุกต์ใช้กับไมโครคอนโทรลเลอร์ ซึ่งมีคุณสมบัติที่เหนือกว่า Z80 มาก ด้วยการเข้าถึงพอร์ต I/O ที่ดีกว่า การพัฒนานำไปใช้กับสัตว์ชนิดอื่นเพียงแต่ ดัดแปลงในส่วนของกลไก และไดรเวอร์เท่านั้น

ปัญหาที่เกิดขึ้น

1. ส่วนใหญ่เป็นปัญหาที่เกี่ยวกับกลไก สำหรับเป็นตัวป้อน โดยเฉพาะตัววาล์วเปิดปิดซึ่งออกแบบได้ยากเนื่องจากมีขนาดเล็ก ซึ่งชุดสำเร็จที่ใช้กับท่อที่มีขายในท้องตลาดราคาชุดละหลายพันบาท

2. ปัญหาเกี่ยวกับไฟเลี้ยงที่ต้องมีการดัดแปลงหลายครั้งเนื่องจาก ชุดกลไกที่ต้องใช้โซลินอยด์ในการเปิด-ปิดใช้กระแสไฟมาก

3. ชุดเซ็นเซอร์ตรวจจับอาหารหมดครั้งแรกออกแบบเป็นกลไก แต่เนื่องจากชิ้นส่วนมีขนาดเล็กและอาหารปลามีผงฝุ่นด้วย จึงต้องเปลี่ยนเป็นใช้โฟโต้ทรานซิสเตอร์

แนวทางในการพัฒนา

ถ้าเกิดต้องการพัฒนาออกแบบเป็นเครื่องควบคุมเครื่องป้อนอาหารสัตว์ชนิดอื่น ถ้าเป็นสัตว์ใหญ่ ปัญหาในการออกแบบกลไกจะลดน้อยลง และสามารถหาอุปกรณ์ไดรวเวอร์ได้ง่าย และถ้าเลี่ยงไปใช้ไฟสลับในการทำงานของกลไกจะทำให้หาอุปกรณ์ง่ายกว่า ใช้ไฟตรงยิ่งขนาดเล็กยิ่งหาซื้อยาก

ถ้าหากต้องการที่จะพัฒนาเพิ่มไปอีก เพียงแต่เขียนโปรแกรมใหม่และออกแบบกลไกใหม่เท่านั้น อีกทั้งยังสามารถที่จะใช้ไมโครคอนโทรลเลอร์ในการควบคุม แสง อุณหภูมิควบคู่ในการเลี้ยงได้อย่างครบวงจรด้วย

บรรณานุกรม

1. ส่ง สุคานนท์มวัฒนา เชียงกูล, พงษ์ศักดิ์ ศิวภัทรกำพล,ไฟฟ้าเบื้องต้น 1 , สำนักพิมพ์ บริษัท ซีเอ็ดยูเคชั่นจำกัด, พ.ศ. 2528

2. ยืน ภู่วรวรรณ, ทฤษฎีและการใช้งานอิเล็คทรอนิกส์ เล่ม 2 , พิมพ์ครั้งที่ 1 ,สำนักพิมพ์ บริษัทซีเอ็ดยูเคชั่น จำกัด , พ.ศ. 2521

3. ธีรวัฒน์ ประกอบผล, การประยุกต์ใช้งานไมโครคอนโทรลเลอร์, พิมพ์ครั้งที่ 1 กรุงเทพ, สำนักพิมพ์ บริษัท ประชาชน จำกัด, 2540

4. สุนทร วิทูสุรพจน์, การโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล C51, พิมพ์ครั้งที่ 1 กรุงเทพ, สำนักพิมพ์ บริษัท ซีเอ็ดยูเคชั่น (มหาชน) จำกัด, 2538

โปรแกรมที่ใช้ในปริญญานิพนธ์

```
COMMANDS EQU 00H
RBD     EQU 01000000B
WRD     EQU 00100000B
DELAY_SCAN EQU 07FH
RTC_DATA EQU   P2.2
RTC_CLK  EQU   P2.0
RTC_RST  EQU   P2.1


    ORG 0000H

START:

    MOV P0,#00H
    MOV P2,#00H
    CALL INIT

MM:
    CALL TIME_TO_SCAN
    CALL CHECK_FOOD
    CALL CHECK_TIME
    CALL CKEY
    JMP MM


CHECK_TIME:

    MOV A,5CH
    CJNE A,#0AAH,H1
    JMP MH2

H1:  MOV  A,30H
    CJNE A,50H,H2
    MOV  A,31H
```

```
        CJNE A,51H,H2
        MOV  A,32H
        CJNE A,52H,H2
        CALL OPEN_CH1

H2:     MOV  A,30H
        CJNE A,53H,H3
        MOV  A,31H
        CJNE A,54H,H3
        MOV  A,32H
        CJNE A,55H,H3
        CALL OPEN_CH1

H3:     MOV  A,30H
        CJNE A,56H,H4
        MOV  A,31H
        CJNE A,57H,H4
        MOV  A,32H
        CJNE A,58H,H4
        CALL OPEN_CH1

H4:     MOV  A,30H
        CJNE A,59H,MH2
        MOV  A,31H
        CJNE A,5AH,MH2
        MOV  A,32H
        CJNE A,5BH,MH2
        CALL OPEN_CH1


MH2:
        MOV A,6CH
        CJNE A,#0AAH,H12
        JMP EXCH
```

```
H12:
    MOV  A,30H
    CJNE A,60H,H22
    MOV  A,31H
    CJNE A,61H,H22
    MOV  A,32H
    CJNE A,62H,H22
    CALL OPEN_CH2

H22:  MOV  A,30H
    CJNE A,63H,H32
    MOV  A,31H
    CJNE A,64H,H32
    MOV  A,32H
    CJNE A,65H,H32
    CALL OPEN_CH2

H32:  MOV  A,30H
    CJNE A,66H,H42
    MOV  A,31H
    CJNE A,67H,H42
    MOV  A,32H
    CJNE A,68H,H42
    CALL OPEN_CH2

H42:  MOV  A,30H
    CJNE A,69H,EXCH
    MOV  A,31H
    CJNE A,6AH,EXCH
    MOV  A,32H
    CJNE A,6BH,EXCH
    CALL OPEN_CH2

EXCH:
    RET
```

```
CHECK_FOOD:
    SETB P2.3
    SETB P2.4

    JB P2.3,NFL1
    SETB P0.4
    MOV 5CH,#0AAH,NFL1
    JMP NFL2
NFL1:
    MOV 5CH,#00
    CLR P0.4

    JB P2.4,NFL2
    SETB P0.5
    MOV 6CH,#0AAH  ;LOW SET AA
    JMP NFL3
NFL2:
    MOV 6CH,#00
    CLR P0.5
NFL3:
    RET


OPEN_CH1:
    MOV DPTR,#CH1_OPEN
    CALL IN_MEM
    CALL SCAN
    SETB P0.0
    MOV  A,#2
    CALL DELAY_SEC
    CLR  P0.0
    SETB P0.1
    MOV  A,#2
    CALL DELAY_SEC
```

```
    CLR  P0.1
    RET

OPEN_CH2:
    MOV DPTR,#CH2_OPEN
    CALL IN_MEM
    CALL SCAN

    SETB P0.2
    MOV  A,#2
    CALL DELAY_SEC
    CLR  P0.2
    SETB P0.3
    MOV  A,#2
    CALL DELAY_SEC
    CLR  P0.3
    RET

SET_HMS:
LRX:
    MOV 71H,#0
    MOV DPTR,#INHOUR
    CALL IN_MEM
    CALL SCAN
    MOV R0,#4DH
    MOV R4,#0

LR3:
    CALL TKEY
    MOV A,70H
    MOV @R0,A
    INC R0

    CJNE A,#2AH,S0
    JMP LRX
```

```
S0:  CJNE A,#23H,S1
     JMP XYY

S1:  SUBB A,#30H
X1:
     XCH A,71H
     MOV B,#10
     MUL AB
     MOV R4,B
     CJNE R4,#0,LRX

     ADD A,71H
     MOV 71H,A

     MOV B,#24
     DIV AB
     JNZ LRX

     MOV @R0,#20H
     CALL SCAN
;    INC R0
     JMP LR3
XYY:

LRX1:
     MOV 72H,#0
     MOV DPTR,#INMIN
     CALL IN_MEM
     CALL SCAN
     MOV R0,#4DH
     MOV R4,#0

LR31:
     CALL TKEY
```

```
        MOV A,70H
        MOV @R0,A
        INC R0

        CJNE A,#2AH,S01
        JMP LRX1

S01:  CJNE A,#23H,S11
        JMP XYY1

S11:  SUBB A,#30H
X11:
        XCH A,72H
        MOV B,#10
        MUL AB
        MOV R4,B
        CJNE R4,#0,LRX1

        ADD A,72H
        MOV 72H,A

        MOV B,#60
        DIV AB
        JNZ LRX1

        MOV @R0,#20H
        CALL SCAN
;       INC R0
        JMP LR31
XYY1:

LRX12:
        MOV 73H,#0
        MOV DPTR,#INSEC
        CALL IN_MEM
```

```
        CALL SCAN
        MOV R0,#4DH
        MOV R4,#0

LR312:
        CALL TKEY
        MOV A,70H
        MOV @R0,A
        INC R0

        CJNE A,#2AH,S02
        JMP LRX12

S02:  CJNE A,#23H,S12
        JMP XYY12

S12:  SUBB A,#30H
X12:
        XCH A,73H
        MOV B,#10
        MUL AB
        MOV R4,B
        CJNE R4,#0,LRX12

        ADD A,73H
        MOV 73H,A

        MOV B,#60
        DIV AB
        JNZ LRX12

        MOV @R0,#20H
        CALL SCAN
        INC R0
        JMP LR312
```

```
XYY12:


    RET

DTH:
    PUSH ACC
    PUSH B
    MOV A,74H
    MOV 74H,#0
    MOV 75H,#0
    MOV 76H,#0

    MOV B,#10
    DIV AB
    MOV 75H,B ;LSB

    ORL A,#00
    JZ EX_DTH

    MOV B,#10
  ; MOV A,75H
    DIV AB
    MOV 76H,B   ;MSB
    MOV A,76H
    SWAP A


EX_DTH:
    ORL 74H,A
    MOV A,75H
    ORL 74H,A
    POP B
    POP ACC
    RET
```

```
CKEY: MOV P3,#11100100B
      PUSH ACC
      MOV A,P3
      CJNE A,#11100100B,USE_KEY
      POP ACC
      RET

USE_KEY:
      POP ACC
USE_KEYX:
      MOV DPTR,#SETTIME
      CALL IN_MEM
      CALL SCAN

DFG:  CALL TKEY
      MOV R5,70H
      CJNE R5,#23H,SXM1
      JMP EX_UK

SXM1:

      CJNE R5,#30H,NCH1
     ; FUNCTION SET TIME
      CALL FSTIME
      JMP USE_KEYX

NCH1:
    CJNE R5,#31H,X01
X00: JMP CH1
```

```
X01: JMP NCH2

CH1:
   MOV DPTR,#SELECT
   CALL IN_MEM
   CALL SCAN
;-----------------------------
   CALL TKEY
   MOV R5,70H


   CJNE R5,#31H,T12
   MOV DPTR,#CH1_T1
   CALL IN_MEM
   CALL SCAN
   MOV A,#1
   CALL DELAY_SEC

   CALL SET_HMS
   MOV 74H,71H
   CALL DTH
   MOV 50H,74H

   MOV 74H,72H
   CALL DTH
   MOV 51H,74H


   MOV 74H,73H
   CALL DTH
   MOV 52H,74H
   JMP CH1
```

```
T12:
    CJNE R5,#32H,T13

    MOV DPTR,#CH1_T2
    CALL IN_MEM
    CALL SCAN
    MOV A,#1
    CALL DELAY_SEC

    CALL SET_HMS
    MOV 74H,71H
    CALL DTH
    MOV 53H,74H

    MOV 74H,72H
    CALL DTH
    MOV 54H,74H

    MOV 74H,73H
    CALL DTH
    MOV 55H,74H
    JMP CH1

T13:
   CJNE R5,#33H,T14

    MOV DPTR,#CH1_T3
    CALL IN_MEM
    CALL SCAN
    MOV A,#1
    CALL DELAY_SEC

    CALL SET_HMS
    MOV 74H,71H
```

```
    CALL DTH
    MOV 56H,74H

    MOV 74H,72H
    CALL DTH
    MOV 57H,74H


    MOV 74H,73H
    CALL DTH
    MOV 58H,74H
    JMP CH1


T14:
    CJNE R5,#34H,T15

    MOV DPTR,#CH1_T4
    CALL IN_MEM
    CALL SCAN
    MOV A,#1
    CALL DELAY_SEC

    CALL SET_HMS
    MOV 74H,71H
    CALL DTH
    MOV 59H,74H

    MOV 74H,72H
    CALL DTH
    MOV 5AH,74H


    MOV 74H,73H
```

```
    CALL DTH
    MOV 5BH,74H
    JMP CH1

T15: CJNE R5,#23H,A00
    JMP A01
A00: JMP CH1
A01:
    JMP USE_KEYX



NCH2:
    CJNE R5,#32H,B00
    JMP CH2
B00:  JMP DFG

CH2:
    MOV DPTR,#SELECT
    CALL IN_MEM
    CALL SCAN
;-------------------------
    CALL TKEY
    MOV R5,70H


    CJNE R5,#31H,T22
    MOV DPTR,#CH2_T1
    CALL IN_MEM
    CALL SCAN
    MOV A,#1
    CALL DELAY_SEC

    CALL SET_HMS
```

```
MOV 74H,71H
CALL DTH
MOV 60H,74H

MOV 74H,72H
CALL DTH
MOV 61H,74H

MOV 74H,73H
CALL DTH
MOV 62H,74H
JMP CH2

T22:
CJNE R5,#32H,T23

MOV DPTR,#CH2_T2
CALL IN_MEM
CALL SCAN
MOV A,#1
CALL DELAY_SEC

CALL SET_HMS
MOV 74H,71H
CALL DTH
MOV 63H,74H

MOV 74H,72H
CALL DTH
MOV 64H,74H
```

```
    MOV 74H,73H
    CALL DTH
    MOV 65H,74H
    JMP CH2

T23:
    CJNE R5,#33H,T24

    MOV DPTR,#CH2_T3
    CALL IN_MEM
    CALL SCAN
    MOV A,#1
    CALL DELAY_SEC

    CALL SET_HMS
    MOV 74H,71H
    CALL DTH
    MOV 66H,74H

    MOV 74H,72H
    CALL DTH
    MOV 67H,74H

    MOV 74H,73H
    CALL DTH
    MOV 68H,74H
    JMP CH2

T24:
    CJNE R5,#34H,T25

    MOV DPTR,#CH2_T4
```

```
    CALL IN_MEM
    CALL SCAN
    MOV A,#1
    CALL DELAY_SEC

    CALL SET_HMS
    MOV 74H,71H
    CALL DTH
    MOV 69H,74H

    MOV 74H,72H
    CALL DTH
    MOV 6AH,74H

    MOV 74H,73H
    CALL DTH
    MOV 6BH,74H
    JMP CH1

T25: CJNE R5,#23H,A002
    JMP A012
A002: JMP CH2
A012:
    JMP USE_KEYX


EX_UK:
    RET

FSTIME:
    CALL SET_HMS
    MOV 74H,71H
    CALL DTH
```

```
MOV 30H,74H

MOV 74H,72H
CALL DTH
MOV 31H,74H

MOV 74H,73H
CALL DTH
MOV 32H,74H

CALL WTIME

RET

TKEY:
MOV 70H,#0
MOV DPTR,#TABLE_KEY

TK1: MOV P3,#11101111B
CALL CKX
MOV P3,#11110111B
CALL CKX
MOV P3,#11111101B
CALL CKX
MOV P3,#11111110B
CALL CKX

MOV A,70H
ORL A,#00
JZ TKEY
RET
```

```
SET_DATA:
    MOV A,#00H
    MOVC A,@A+DPTR
    MOV 70H,A
NO_D: NOP
    JNB P3.6,NO_D
    NOP
    JNB P3.2,NO_D
    NOP
    JNB P3.5,NO_D
    RET

CKX:
    JNB P3.6,SET_DATA
    INC DPTR
    JNB P3.2,SET_DATA
    INC DPTR
    JNB P3.5,SET_DATA
    INC DPTR

    RET

TIME_TO_SCAN:
   MOV DPTR,#TTIME
   CALL IN_MEM
   CALL TIME_DISPLAY


   MOV A,30H
```

```
ANL A,#0FH
ADD A,#30H
MOV 46H,A
MOV A,30H
SWAP A
ANL A,#0FH
ADD A,#30H
MOV 45H,A

MOV A,31H
ANL A,#0FH
ADD A,#30H
MOV 49H,A
MOV A,31H
SWAP A
ANL A,#0FH
ADD A,#30H
MOV 48H,A

MOV A,32H
ANL A,#0FH
ADD A,#30H
MOV 4CH,A
MOV A,32H
SWAP A
ANL A,#0FH
ADD A,#30H
MOV 4BH,A
CALL SCAN


RET
```

```
RTC_WRITE_CH:

    CLR   RTC_CLK      ;low CLK line
    LCALL  DELAY
    SETB  RTC_RST      ;high RST line
    LCALL  DELAY
    MOV   A,R2         ;write COMMAND byte
    LCALL  RTC_WRITE_8BIT
    MOV   A,R3         ;and DATA byte
    LCALL  RTC_WRITE_8BIT
    CLR   RTC_RST      ;low RST line
    LCALL  DELAY
    RET
;
; Transfer 8-bits data to DS1202
; IN: A = data
;
RTC_WRITE_8BIT:
    MOV   R4,#08H      ;8 bits to transfer
WR8BIT1:
    RRC   A
    MOV   RTC_DATA,C
    SETB  RTC_CLK      ;Rising edge clock
    LCALL  DELAY
    CLR   RTC_CLK
    LCALL  DELAY
    DJNZ  R4,WR8BIT1
    RET
;
; Perform READ Command and read parameter byte from DS1202 RTC
; IN:  R2 = COMMAND
; OUT: R3 = DATA
;
RTC_READ_CH:
    CLR   RTC_CLK      ;low clk line
```

```
    LCALL  DELAY
    SETB   RTC_RST      ;high RST line
    LCALL  DELAY
    MOV    A,R2         ;write COMMAND byte
    LCALL  RTC_WRITE_8BIT
;
    MOV    R4,#8        ;then read DATA byte
    CLR    A
RD_CH1: CLR    RTC_CLK
    LCALL  DELAY
    MOV    C,RTC_DATA
    RRC    A
    SETB   RTC_CLK
    LCALL  DELAY
    DJNZ   R4,RD_CH1
    MOV    R3,A
;
    CLR    RTC_RST      ;low RST line
    LCALL  DELAY
    RET

DELAY:
    NOP    ;delay
    RET

TIME_DISPLAY:

    MOV R2,#81H
    LCALL RTC_READ_CH     ;READ SECOUND
    MOV 32H,R3

    MOV R2,#83H
    LCALL RTC_READ_CH
    MOV 31H,R3
```

```
    MOV R2,#85H
    LCALL RTC_READ_CH
    MOV 30H,R3
    MOV A,30H

    RET

WTIME:
    MOV R2,#8EH
    MOV R3,#00H
    LCALL RTC_WRITE_CH

    MOV R2,#80H
    MOV R3,32H
    LCALL RTC_WRITE_CH

    MOV R2,#82H
    MOV R3,31H
    LCALL RTC_WRITE_CH

    MOV R2,#84H
    MOV R3,30H
    LCALL RTC_WRITE_CH

    MOV R2,#86H
    MOV R3,33H
    LCALL RTC_WRITE_CH

    MOV R2,#88H
    MOV R3,34H
    LCALL RTC_WRITE_CH

    MOV R2,#8CH
```

```
        MOV R3,35H
        LCALL RTC_WRITE_CH

        MOV R2,#8EH
        MOV R3,#80H
        LCALL RTC_WRITE_CH
        RET

SCANL:

KHE:  MOV R4,#16
      MOV R0,#40H
      PUSH DPH
      PUSH DPL
IMEM:  MOV A,#0
      MOVC A,@A+DPTR
      CJNE A,#40H,JKL
      JMP XFX
JKL:  MOV  @R0,A
      INC R0
      INC DPTR
      DJNZ R4,IMEM
```

```
        CALL SCAN

        PUSH ACC
        MOV A,#DELAY_SCAN
        CALL DELAY_MS
        POP ACC

        POP DPL
        POP DPH
        INC DPTR
        MOV A,#0
        MOVC A,@A+DPTR
        JMP KHE
XFX:    POP DPL
        POP DPH

        RET



SCAN:
        PUSH DPH
        PUSH DPL
        PUSH 00
        MOV A,#80H
        CALL BLOCK2

        MOV B,#8
        MOV R0,#40H
L1:
        MOV  A,@R0
        CALL WRITE
```

```
        INC R0
        DJNZ B,L1



        MOV A,#0C0H
        CALL BLOCK2

        MOV B,#8
        MOV R0,#48H
LX:     MOV A,#0
        MOV A,@R0
        CALL WRITE
        INC R0
        DJNZ B,LX
        POP 00
        POP DPL
        POP DPH
        RET


IN_MEM:
        ;DPTR IS OFFSET DATA TO 40 ==> 4FH
;       PUSH DPH
;       PUSH DPL

        MOV R2,#16
        MOV R0,#40H

IN1:    MOV A,#0
        MOVC A,@A+DPTR
        MOV @R0,A
        INC DPTR
        INC R0
        DJNZ R2,IN1
```

```
;    POP DPL
;    POP DPH
     RET


BLOCK2:
;A SELECT BLOCK
     PUSH ACC
     MOV P2,#COMMANDS
     POP ACC
     MOV P1,A
     CALL EPLUSE
     CALL WAITBF
     RET


WRITE:
     PUSH ACC
     MOV P2,#WRD
     POP ACC
     MOV P1,A
     CALL EPLUSE
     CALL WAITBF
     RET


EPLUSE: SETB P2.7
```

```
    PUSH ACC
    MOV A,#01
    CALL DELAY_MS
    CLR  P2.7
    POP ACC
    RET


WAITBF:
    PUSH ACC
    MOV A,#4
    CALL DELAY_MS
    POP ACC
    RET

INIT:
    MOV P2,#COMMANDS

    MOV P1,#38H     ;DISPLAY 5*7 DOT 1 LINE
    CALL EPLUSE
    CALL WAITBF


    MOV P1,#0FH
    CALL EPLUSE
    CALL WAITBF

    MOV P1,#6
    CALL EPLUSE
    CALL WAITBF

    MOV P1,#1
    CALL EPLUSE
    CALL WAITBF
```

```
        MOV P1,#80H
        CALL EPLUSE
        CALL WAITBF
        RET

TABLE:
        DB "                "
        DB "Program demo [KEYBOARD and LCD DISPLAY] "
        DB "KMITL ENGINEERING  "
        DB " <==== 11 MARCH 1997. ====>."
        DB "        @"
TTIME:
        DB "TIME  :  :    "
CLE:    DB "                "
TABLE_KEY:
        DB "123456789*0#"
;       0123456789ABCDEF

SETTIME: DB "  SELET MODE    "
INSEC:  DB "INPUT SEC=       "
INMIN:  DB "INPUT MINUTE=    "
INHOUR: DB "INPUT HOUR=      "
SELECT: DB "  SELECT TIME    "

CH1_T1: DB "INPUT TIME1 CH1  "
CH1_T2: DB "INPUT TIME2 CH1  "
CH1_T3: DB "INPUT TIME3 CH1  "
CH1_T4: DB "INPUT TIME4 CH1  "

CH2_T1:  DB "INPUT TIME1 CH2  "
CH2_T2:  DB "INPUT TIME2 CH2  "
CH2_T3:  DB "INPUT TIME3 CH2  "
CH2_T4:  DB "INPUT TIME4 CH2  "
CH1_OPEN: DB "  CH1 OPEN      "
CH2_OPEN: DB "  CH2 OPEN      "
```

```
TM1:    DB "TIME 1  : : : "
TM2:    DB "TIME 2  : : : "
TM3:    DB "TIME 3  : : : "
TM4:    DB "TIME 4  : : : "


 $INCLUDE "DELAY.ASM"


 END
```

ผนวก ก.

ข้อมูลอ้างอิงของไอซีต่างๆ

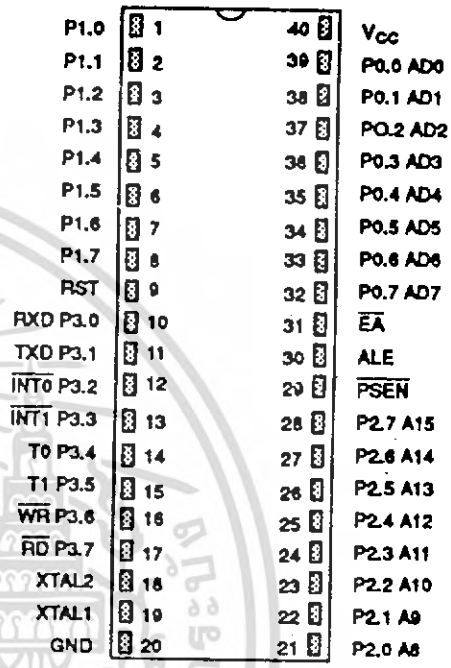
DALLAS SEMICONDUCTOR

DS5000(T
Soft Microcontrolle

FEATURES

- 8–bit 8051 compatible uC adapts to task–at–hand:
  - 8 or 32 Kbytes of nonvolatile RAM for program and/or data memory storage
  - Initial downloading of software in end system via on–chip serial port
  - Capable of modifying its own program and/or data memory in end use
- Crashproof operation:
  - Maintains all nonvolatile resources for 10 years in the absence of V_{CC}
  - Power–fail reset
  - Early warning power–fail interrupt
  - Watchdog timer
- Software Security Feature:
  - Executes encrypted software to prevent unauthorized disclosure
- On–chip, full–duplex serial I/O ports
- Two on–chip timer/event counters
- 32 parallel I/O lines
- Compatible with industry standard 8051 instruction set and pinout
- Optional Permanently Powered Real–Time Clock (DS5000T)

PIN ASSIGNMENT



40-Pin Encapsulated Package

DESCRIPTION

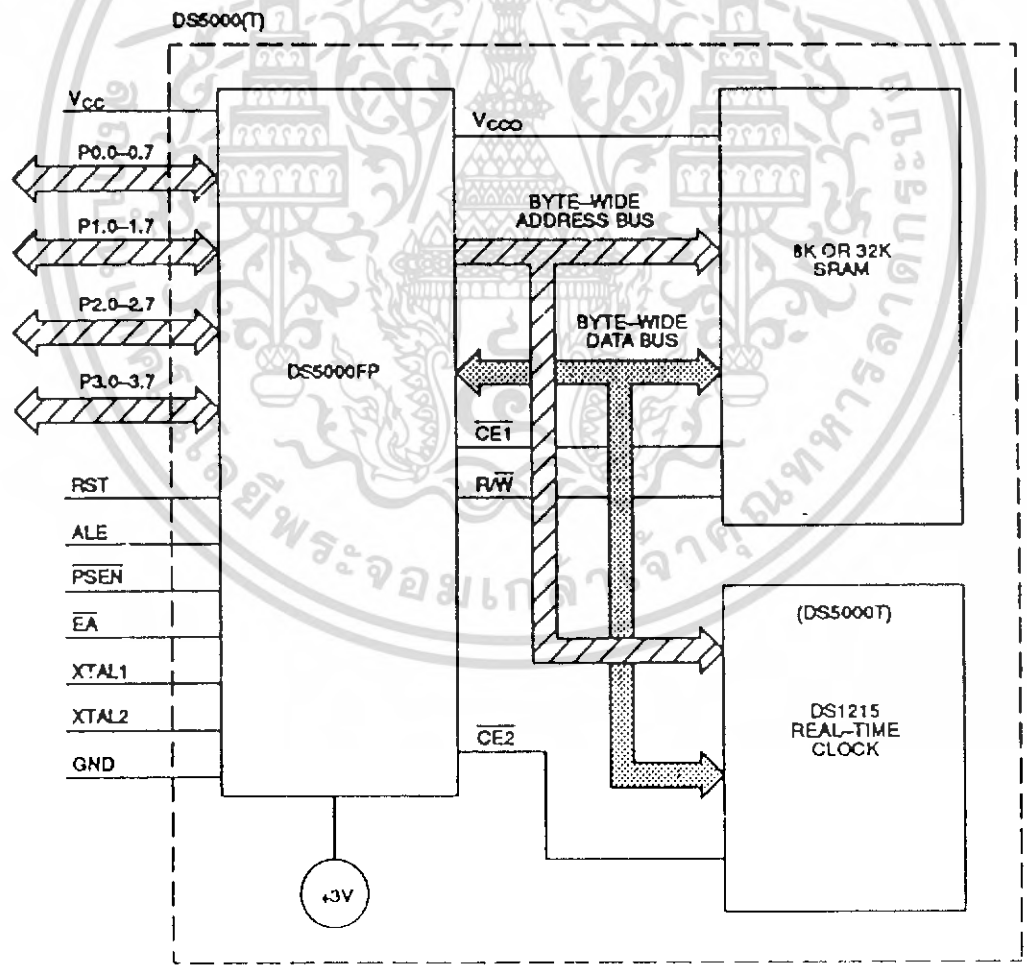
The DS5000(T) Soft Microcontroller is a fully 8051 compatible 8–bit CMOS microcontroller that offers "softness" in all aspects of its application. This is accomplished through the comprehensive use of nonvolatile technology to preserve all information in the absence of system V_{CC}. The internal program/data memory space is implemented using either 8K or 32K bytes of nonv tile CMOS SRAM. Furthermore, internal data regis and key configuration registers are also nonvolatile. optional real–time clock gives permanently powe timekeeping. The clock keeps time to a hundredth second using an on–board crystal.

ORDERING INFORMATION

PART NUMBER	RAM SIZE	MAX CRYSTAL SPEED	TIMEKEEPING?
DS5000–8–8	8K bytes	8 MHz	No
DS5000–8–12	8K bytes	12 MHz	No
DS5000–8–16	8K bytes	16 MHz	No
DS5000–32–8	32K bytes	8 MHz	No
DS5000–32–12	32K bytes	12 MHz	No
DS5000–32–16	32K bytes	16 MHz	No
DS5000T–8–8	8K bytes	8 MHz	Yes
DS5000T–8–12	8K bytes	12 MHz	Yes
DS5000T–8–16	8K bytes	16 MHz	Yes
DS5000T–32–8	32K bytes	8 MHz	Yes
DS5000T–32–12	32K bytes	12 MHz	Yes
DS5000T–32–16	32K bytes	16 MHz	Yes

Operating information is contained in the User's Guide section of the Soft Microcontroller Data Book. This data sheet provides ordering information, pinout, and electrical specification.

DS5000(T) BLOCK DIAGRAM Figure 1



PIN DESCRIPTION

PIN NUMBER	DESCRIPTION
1–8	P1.0 – P1.7 General purpose I/O Port 1
9	RST Active high reset input. A logic 1 applied to this pin will activate a reset state. This pin is pulled down internally so this pin can be left unconnected if not used.
10	P3.0 RXD General purpose I/O port pin 3.0. Also serves as the receive signal for the on board UART. This pin should not be connected directly to a PC COM port.
11	P3.1 TXD General purpose I/O port pin 3.1. Also serves as the transmit signal for the on board UART. This pin should not be connected directly to a PC COM port.
12	P3.2 $\overline{INT0}$ General purpose I/O port pin 3.2. Also serves as the active low External Interrupt 0.
13	P3.3 $\overline{INT1}$ General purpose I/O port pin 3.3. Also serves as the active low External Interrupt 1.
14	P3.4 T0 General purpose I/O port pin 3.4. Also serves as the Timer 0 input.
15	P3.5 T1 General purpose I/O port pin 3.5. Also serves as the Timer 1 input.
16	P3.6 $\overline{WR}$ General purpose I/O port pin. Also serves as the write strobe for Expanded bus operation.
17	P3.7 $\overline{RD}$ General purpose I/O port pin. Also serves as the read strobe for Expanded bus operation.
18, 19	XTAL2, XTAL1 Used to connect an external crystal to the internal oscillator. XTAL1 is the input to an inverting amplifier and XTAL2 is the output.
20	GND Logic ground.
21–28	P2.0–P2.7 General purpose I/O Port 2. Also serves as the MSB of the Expanded Address bus.

PIN NUMBER	DESCRIPTION
29	$\overline{PSEN}$ Program Store Enable. This active low signal is used to enable an external program memory when using the Expanded bus. It is normally an output and should be unconnected if not used. $\overline{PSEN}$ also is used to invoke the Bootstrap Loader. At this time, $\overline{PSEN}$ will be pulled down externally. This should only be done once the DS5000 is already in a reset state. The device that pulls down should be open drain since it must not interfere with $\overline{PSEN}$ under normal operation.
30	ALE Address Latch Enable. Used to de–multiplex the multiplexed Expanded Address/Data bus on Port 0. This pin is normally connected to the clock input on a '373 type transparent latch. When using a parallel programmer, this pin also assumes the $\overline{PROG}$ function for programming pulses.
31	$\overline{EA}$ External Access. This pin forces the DS5000 to behave like an 8031. No internal memory (or clock) will be available when this pin is at a logic low. Since this pin is pulled down internally, it should be connected to +5V to use NVRAM. In a parallel programmer, this pin also serves as V_{PP} for super voltage pulses.
32–39	P0.7–P0.0 General purpose I/O Port 0. This port is open–drain and can not drive a logic 1. It requires external pull–ups. Port 0 is also the multiplexed Expanded Address/Data bus. When used in this mode, it does not require pull–ups.
40	V_{CC} +5 volts.

INSTRUCTION SET

The DS5000 executes an instruction set which is object code compatible with the industry standard 8051 microcontroller. As a result, software development packages which have been written for the 8051 are compatible with the DS5000, including cross–assemblers, high–level language compilers, and debugging tools.

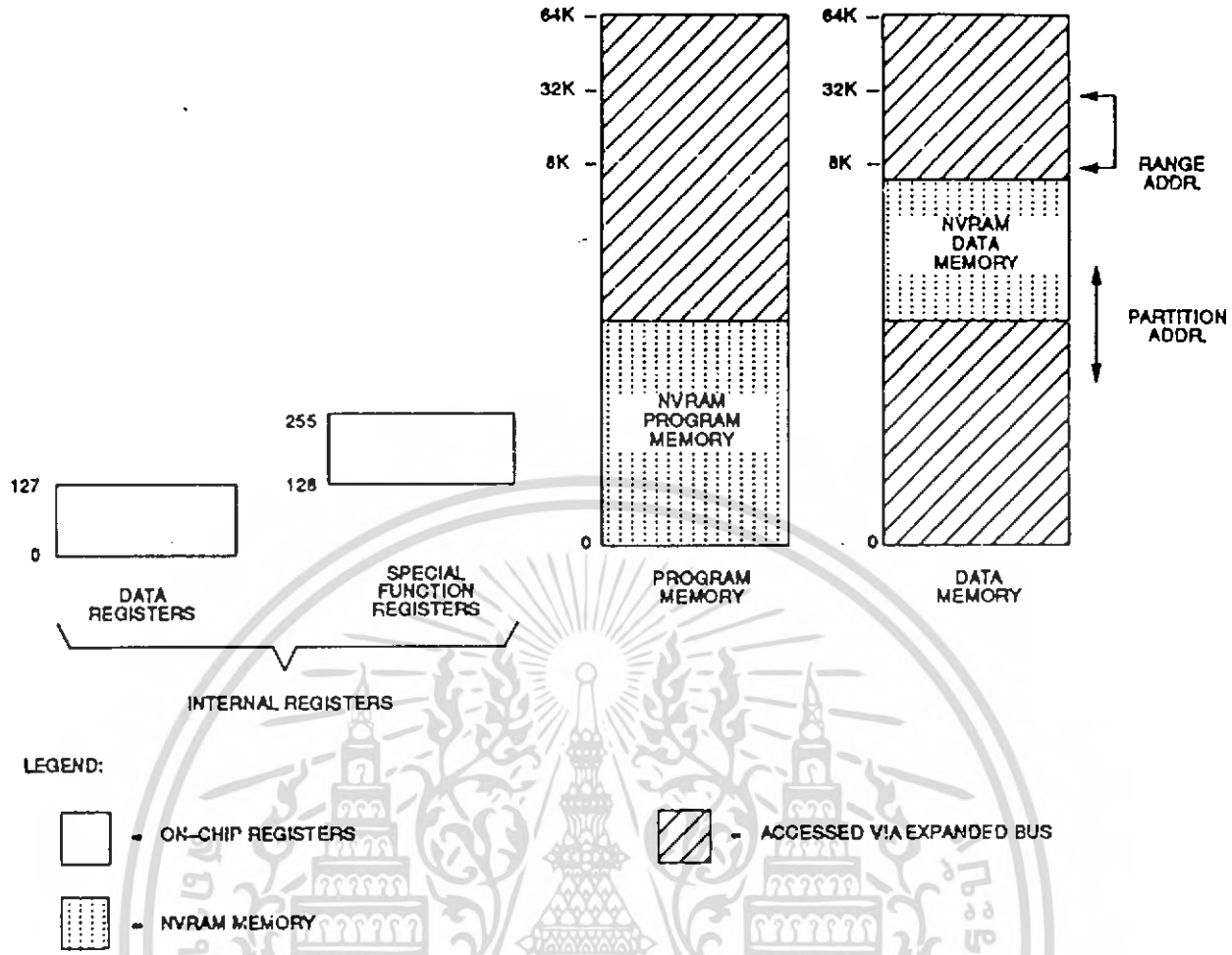
A complete description for the DS5000 instruction set is available in the User's Guide section of the Soft Microcontroller Data Bock.

MEMORY ORGANIZATION

Figure 2 illustrates the address spaces which are accessed by the DS5000. As illustrated in the figure, separate address spaces exist for program and data memory. Since the basic addressing capability of the machine is 16 bits, a maximum of 64 Kbytes of program memory and 64 Kbytes of data memory can be accessed by the DS5000 CPU. The 8K or 32K byte RAM area inside of the DS5000 can be used to contain both program and data memory.

The Real–time Clock (RTC) in the DS5000T is reached in the memory map by setting a SFR bit. The MCON.2 bit (ECE2) is used to select an alternate data memory map. While ECE2=1, all MOVXs will be routed to this alternate memory map. The real–time clock is a serial device that resides in this area. A full description of the RTC access and example software is given in the User's Guide section of the Soft Microcontroller Data Book. If the ECE2 bit is set on a DS5000 without a timekeeper, the MOVXs will simply go to a nonexistent memory. Software execution would not be affected otherwise.

DS5000 LOGICAL ADDRESS SPACES Figure 2



PROGRAM LOADING

The Program Load Modes allow initialization of the NVRAM Program/Data Memory. This initialization may be performed in one of two ways:

1. Serial Program Loading which is capable of performing Bootstrap Loading of the DS5000(T). This feature allows the loading of the application program to be delayed until the DS5000(T) is installed in the end system.
2. Parallel Program Load cycles which perform the initial loading from parallel address/data information presented on the I/O port pins. this mode is timing–set compatible with the 8751H microcontroller programming mode.

The DS5000 is placed in its Program Load configuration by simultaneously applying a logic 1 to the RST pin and forcing the $\overline{\text{PSEN}}$ line to a logic 0 level. Immediately following this action, the DS5000 will look for a parallel Program Load pulse, or a serial ASCII carriage return (0DH) character received at 9600, 2400, 1200, or 300 bps over the serial port.

The hardware configurations used to select these modes of operation are illustrated in Figure 3.

PROGRAM LOADING CONFIGURATIONS Figure 3

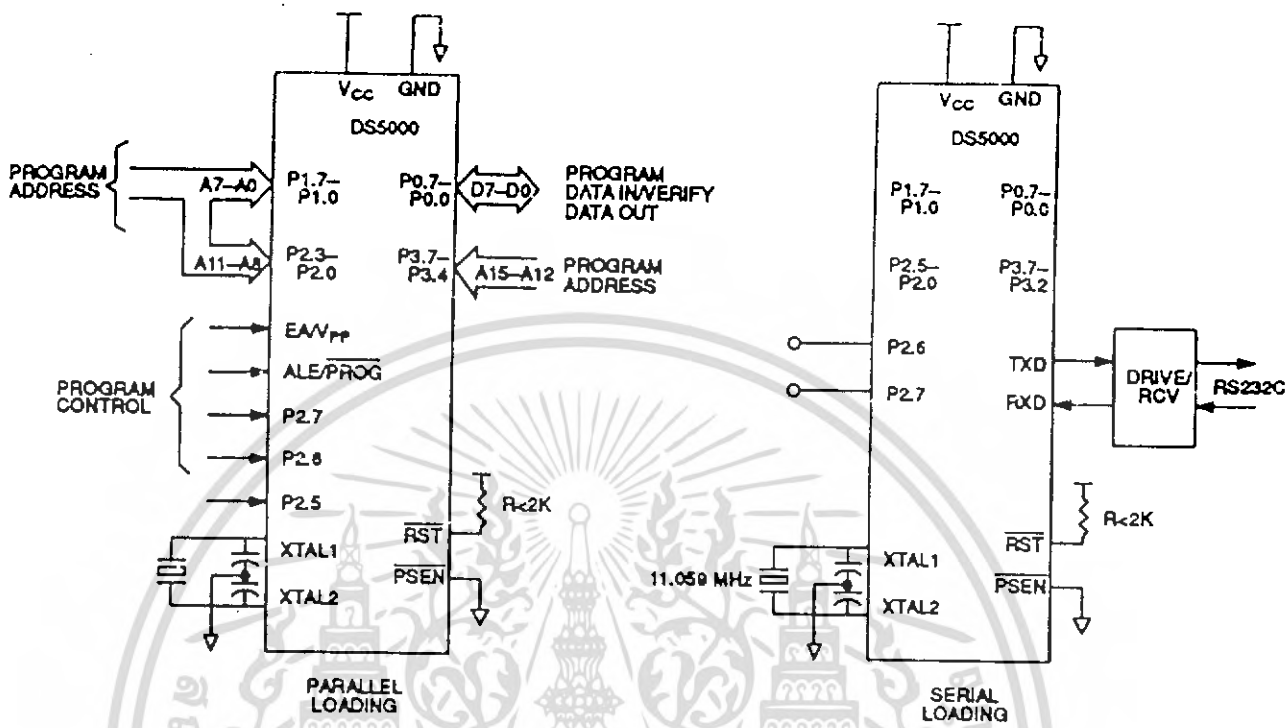


Table 1 summarizes the selection of the available Parallel Program Load cycles. The timing associated with these cycles is illustrated in the electrical specs.

SERIAL BOOTSTRAP LOADER

The Serial Program Load Mode is the easiest, fastest, most reliable, and most complete method of initially loading application software into the DS5000 nonvolatile RAM. Communication can be performed over a standard asynchronous serial communications port. A typical application would use a simple RS232C serial interface to program the DS5000 as a final production procedure. The hardware configuration which is required for the Serial Program Load mode is illustrated in Figure 3. Port pins 2.7 and 2.6 must be either open or pulled high to avoid placing the DS5000 in a parallel load cycle. Although an 11.0592 MHz crystal is shown in Figure 3, a variety of crystal frequencies and loader baud rates are supported, shown in Table 2. The serial loader is designed to operate across a three–wire interface from a standard UART. The receive, transmit, and ground wires are all that are necessary to establish communication with the DS5000.

The Serial Bootstrap Loader implements an easy–to–use command line interface which allows an application program in an Intel hex representation to be loaded into and read back from the device. Intel hex is the typical format which existing 8051 cross–assemblers output. The serial loader responds to single character commands which are summarized below:

COMMAND	FUNCTION
C	Return CRC–16 checksum of embedded RAM
D	Dump Intel Hex File
F	Fill embedded RAM block with constant
K	Load 40–bit Encryption Key
L	Load Intel Hex File
R	Read MCON register
T	Trace (Echo) incoming Intel Hex data
U	Clear Security Lock
V	Verify Embedded RAM with incoming Intel Hex
W	Write MCON register
Z	Set Security Lock
P	Put a value to a port.
G	Get a value from a port.

PARALLEL PROGRAM LOAD CYCLES Table 1

MODE	RST	$\overline{PSEN}$	$\overline{PROG}$	$\overline{EA}$	P2.7	P2.6	P2.5
Program	1	0	0	V_{PP}	1	0	X
Security Set	1	0	0	V_{PP}	1	1	X
Verify	1	X	X	1	0	0	X
Prog Expanded	1	0	0	V_{PP}	0	1	0
Verify Expanded	1	0	1	1	0	1	0
Prog MCON or Key registers	1	0	0	V_{PP}	0	1	1
Verify MCON registers	1	0	1	1	0	1	1

The Parallel Program Cycle is used to load a byte of data into a register or memory location within the DS5000. The Verify Cycle is used to read this byte back for comparison with the originally loaded value to verify proper loading. The Security Set Cycle may be used to enable and the Software Security feature of the DS5000. One may also enter bytes for the MCON register or for the five encryption registers using the Program MCON cycle. When using this cycle, the absolute register address must be presented at Ports 1 and 2 as in the normal program cycle (Port 2 should be 00H). The MCON contents can likewise be verified using the Verify MCON cycle.

When the DS5000 first detects a Parallel Program Strobe pulse or a Security Set Strobe pulse while in the Program Load Mode following a Power On Reset, the internal hardware of the DS5000 is initialized so that an existing 4 Kbyte program can be programmed into a DS5000 with little or no modification. This initialization automatically sets the Range Address for 8 Kbytes and maps the lowest 4 Kbyte bank of Embedded RAM as program memory. The next 4 Kbytes of Embedded RAM are mapped as Data Memory.

In order to program more than 4 Kbytes of program code, the Program/Verify Expanded cycles can be used. Up to 32 Kbytes of program code can be entered and verified. Note that the expanded 32 Kbyte Program/Verify cycles take much longer than the normal 4 Kbyte Program/Verify cycles.

A typical parallel loading session would follow this procedure. First, set the contents of the MCON register with the correct range and partition only if using expanded programming cycles. Next, the encryption registers can be loaded to enable encryption of the program/data memory (not required). Then, program the DS5000 using either normal or expanded program cycles and check the memory contents using Verify cycles. The last operation would be to turn on the security lock feature by either a Security Set cycle or by explicitly writing to the MCON register and setting MCON.0 to a 1.

SERIAL LOADER BAUD RATES FOR DIFFERENT CRYSTAL FREQUENCIES Table 2

CRYSTAL FREQ (MHz)	BAUD RATE					
	300	1200	2400	9600	19200	57600
14.7456		Y	Y	Y	Y	
11.0592	Y	Y	Y	Y	Y	Y
9.21600	Y	Y	Y	Y		
7.37280	Y	Y	Y	Y		
5.52960	Y	Y	Y	Y		
1.84320	Y	Y	Y	Y		

ADDITIONAL INFORMATION

A complete description for all operational aspects of the DS5000, is provided in the User's Guide section of the Soft Microcontroller Data Book.

DEVELOPMENT SUPPORT

Dallas Semiconductor offers a kit package for developing and testing user code. The DS5000TK Evaluation Kit allows the user to download Intel hex formatted code directly to the DS5000 from a PC–XT/AT or compatible computer. The kit consists of a DS5000T–32–12, an interface pod, demo software, and an RS232 connector that attaches to the COM1 or COM2 serial port of a PC. See the User's Guide for further details.

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.3V to 7.0V
Operating Temperature	0°C to +70°C
Storage Temperature	–40°C to 70°C
Soldering Temperature	260°C for 10 seconds

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC CHARACTERISTICS

(t_A = 0°C to70°C; V_{CC} = 5V ± 5%

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Low Voltage	V_{IL}	-0.3		0.8	V	1
Input High Voltage	V_{IH1}	2.0		V_{CC} +0.3	V	1
Input High Voltage RST, XTAL2	V_{IH2}	3.5		V_{CC} +0.3	V	1
Output Low Voltage @ I_{OL}=1.6mA (Ports 1, 2, 3)	V_{OL1}		.15	0.45	V	
Output Low Voltage @ I_{OL}=3.2mA (Ports 0, ALE, $\overline{PSEN}$)	V_{OL2}		.15	0.45	V	1
Output High Voltage @ I_{OH}=–80µA (Ports 1, 2, 3)	V_{OH1}	2.4	4.8		V	1
Output High Voltage @ I_{OH}=–400 µA (Ports 0, ALE, $\overline{PSEN}$)	V_{OH2}	2.4	4.8		V	1
Input Low Current V_{IN} = 0.45V (Ports 1, 2, 3)	I_{IL}			-50	µA	
Transition Current; 1 to 0 V_{IN} = 2.0V (Ports 1, 2, 3)	I_{TL}			-500	µA	
Input Leakage Current 0.45 < V_{IN} < V_{CC} (Port 0)	i_L			±10	µA	
RST, $\overline{EA}$ Pulldown Resistor	R_{RE}	40		125	KΩ	
Stop Mode Current	I_{SM}			80	µA	4
Power Fail Warning Voltage	V_{PFW}	4.15	4.6	4.75	V	1
Minimum Operating Voltage	V_{CCmin}	4.05	4.5	4.65	V	1
Programming Supply Voltage (Parallel Program Mode)	V_{PP}	12.5		13	V	1
Program Supply Current	I_{PP}		15	20	mA	
Operating Current DS5000–8K@8 MHz DS5000–32K @ 12 MHz DS5000T–32–16 @ 16 MHz	I_{CC}		25.2 35.7 45.6	43 48 54	mA	2
Idle Mode Current @ 12 MHz	I_{CC}		4.5	6.2	mA	3

AC CHARACTERISTICS
EXPANDED BUS MODE TIMING SPECIFICATIONS

(t_A = 0°C to70°C; V_{CC} = 5V ± 5%)

#	PARAMETER	SYMBOL	MIN	MAX	UNITS
1	Oscillator Frequency	$1/t_{CLK}$	1.0	8 (-8) 12 (-12) 16 (-16)	MHz
2	ALE Pulse Width	t_{ALPW}	$2*t_{CLK}$		ns
3	Address Valid to ALE Low	t_{AVALL}	t_{CLK} -40		ns
4	Address Hold After ALE Low	t_{AVAAV}	t_{CLK} -35		ns
5	ALE Low to Valid Instr. In @16 MHz	t_{ALLVI}		$4t_{CLK}$ -150 $4t_{CLK}$ -90	ns ns
6	ALE Low to $\overline{PSEN}$ Low	t_{ALLPSL}	t_{CLK} -25		ns
7	$\overline{PSEN}$ Pulse Width	t_{PSPW}	$3t_{CLK}$ -35		ns
8	$\overline{PSEN}$ Low to Valid Instr. In @16 MHz	t_{PSLVI}		$3t_{CLK}$ -150 $3t_{CLK}$ -90	ns ns
9	Input Instr. Hold after $\overline{PSEN}$ Going High	t_{PSIV}	0		ns
10	Input Instr. Flat after $\overline{PSEN}$ Going High	t_{PSIX}		t_{CLK} -20	ns
11	Address Hold after $\overline{PSEN}$ Going High	t_{PSAV}	t_{CLK} -8		ns
12	Address Valid to Valid Instr. In @16 MHz	t_{AVVI}		$5t_{CLK}$ -150 $5t_{CLK}$ -90	ns ns
13	$\overline{PSEN}$ Low to Address Float	t_{PSLAZ}	0		ns
14	$\overline{RD}$ Pulse Width	t_{RDPW}	$6t_{CLK}$ -100		ns
15	$\overline{WR}$ Pulse Width	t_{WRPW}	$6t_{CLK}$ -100		ns
16	$\overline{RD}$ Low to Valid Data In @16 MHz	t_{RDLDV}		$5t_{CLK}$ -165 $5t_{CLK}$ -105	ns ns
17	Data Hold after $\overline{RD}$ High	t_{RDHDV}	0		ns
18	Data Float after $\overline{RD}$ High	t_{RDHDZ}		$2t_{CLK}$ -70	ns
19	ALE Low to Valid Data In @16 MHz	t_{ALLVD}		8_{CLK} -150 $8t_{CLK}$ -90	ns ns
20	Valid Addr. to Valid Data In @16 MHz	t_{AVDV}		$9t_{CLK}$ -165 $9t_{CLK}$ -105	ns ns
21	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	t_{ALLRDL}	$3t_{CLK}$ -50	$3t_{CLK}$ +50	ns
22	Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	t_{AVRDL}	$4t_{CLK}$ -130		ns
23	Data Valid to $\overline{WR}$ Going Low	t_{DVWRL}	t_{CLK} -60		ns
24	Data Valid to $\overline{WR}$ High @16 MHz	t_{DVWRH}	$7t_{CLK}$ -150 $7t_{CLK}$ -90		ns ns
25	Data Valid after $\overline{WR}$ High	t_{WRHDV}	t_{CLK} -50		ns
26	$\overline{RD}$ Low to Address Float	t_{RDLAZ}		0	ns
27	$\overline{RD}$ or $\overline{WR}$ High to ALE High	t_{RDHALH}	t_{CLK} -40	t_{CLK} +50	ns

AC CHARACTERISTICS (cont'd)
PARALLEL PROGRAM LOAD TIMING

(t_A = 0°C to70°C; V_{CC} = 5V ± 5%)

#	PARAMETER	SYMBOL	MIN	MAX	UNITS
40	Oscillator Frequency	$1/t_{CLK}$	1.0	12.0	MHz
41	Address Setup to $\overline{PROG}$ Low	t_{AVPRL}	0		
42	Address Hold after $\overline{PROG}$ High	t_{PRHAV}	0		
43	Data Setup to $\overline{PROG}$ Low	t_{DVPRL}	0		
44	Data Hold after $\overline{PROG}$ High	t_{PRHDV}	0		
45	P2.7, 2.6, 2.5 Setup to V_{PP}	t_{P27HVP}	0		
46	V_{PP} Setup to $\overline{PROG}$ Low	t_{VPHPRL}	0		
47	V_{PP} Hold after $\overline{PROG}$ Low	t_{PRHVPL}	0		
48	$\overline{PROG}$ Width Low	t_{PRW}	2400		t_{CLK}
49	Data Output from Address Valid	t_{AVDV}		48 1800*	t_{CLK}
50	Data Output from P2.7 Low	t_{DVP27L}		48 1800*	t_{CLK}
51	Data Float after P2.7 High	t_{P27HDZ}	0	48 1800*	t_{CLK}
52	Delay to Reset/$\overline{PSEN}$ Active after Power On	t_{PORPV}	21504		t_{CLK}
53	Reset/$\overline{PSEN}$ Active (or Verify Inactive) to V_{PP} High	t_{RAVPH}	1200		t_{CLK}
54	V_{PP} Inactive (Between Program Cycles)	t_{VPPPC}	1200		t_{CLK}
55	Verify Active Time	t_{VFT}	48 2400 t_{CLK}		t_{CLK}

* Second set of numbers refers to expanded memory programming up to 32K bytes.

Features

- Compatible with MCS-51™ Products
- 4 Kbytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
  - Data Retention: 10 Years
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Five Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

Description

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

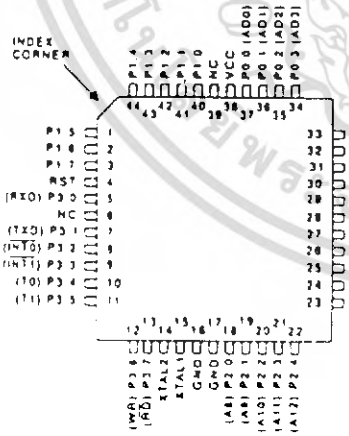
The AT89C51 provides the following standard features: 4 Kbytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five source two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 is

continued

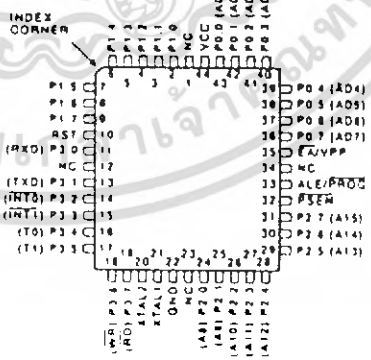
Pin Configurations

PDIP/Cerdip

PQFP/TQFP



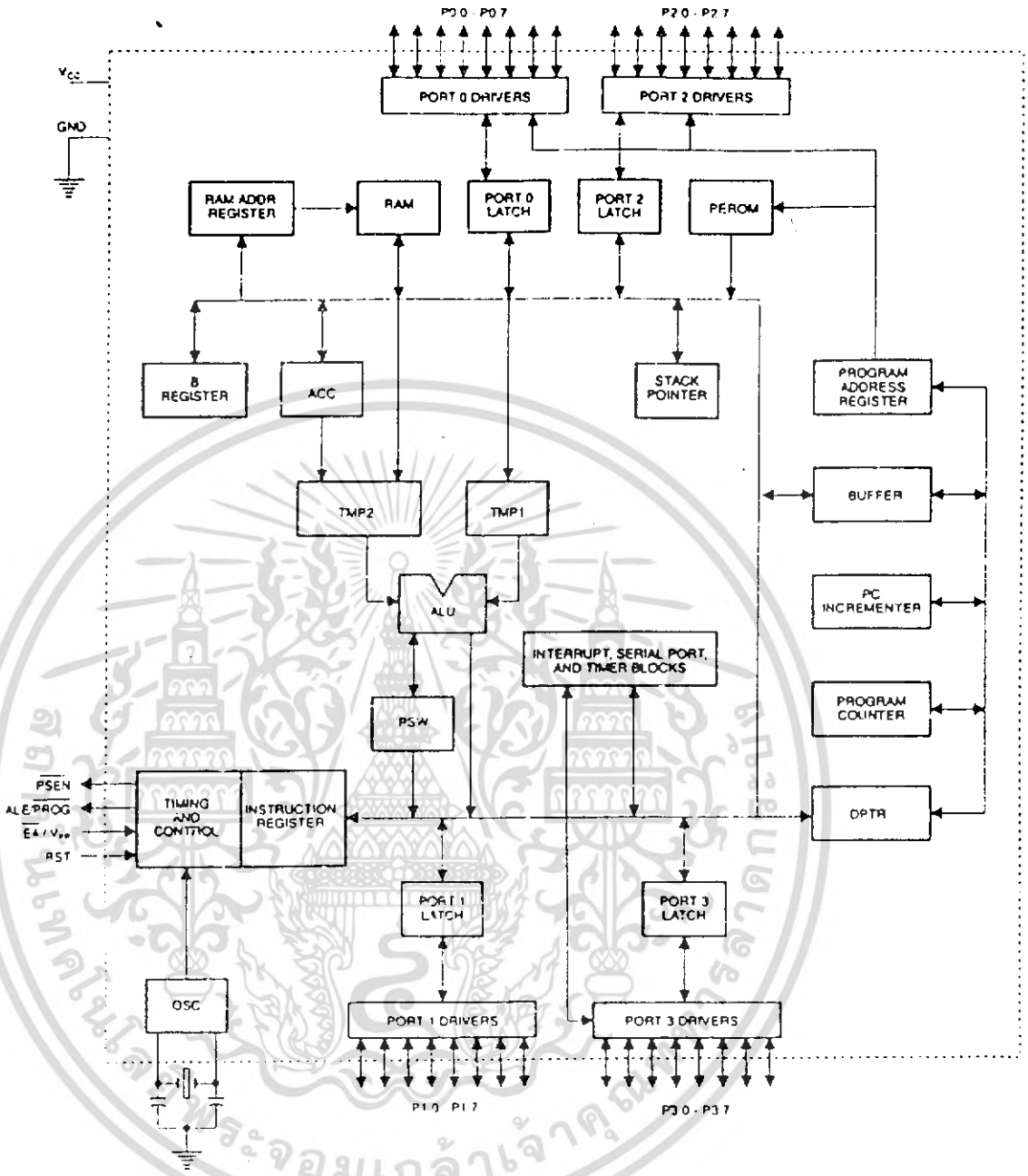
PLCC/LCC



8-Bit Microcontroller with 4 Kbytes Flash

AT89C51

Block Diagram



Description (Continued)

designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Description

V_{CC}

Supply voltage.

GND

Ground.

Port 0

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

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

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

Port 1

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

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

Port 2

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

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

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

Port 3

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

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

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

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

RST

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

ALE/$\overline{PROG}$

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

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

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

$\overline{PSEN}$

Program Store Enable is the read strobe to external program memory.

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

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

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

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

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

continued

Pin Description (Continued)

XTAL1

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

XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

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

Idle Mode

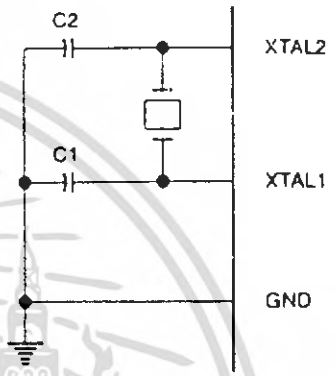
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware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

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

Power Down Mode

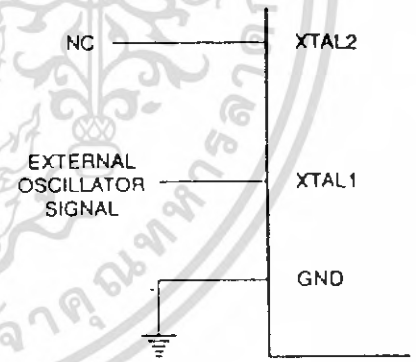
In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines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

Figure 1. Oscillator Connections



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

Figure 2. External Clock Drive Configuration



Status of External Pins During Idle and Power Down

Mode	Program Memory	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

Program Memory Lock Bits

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

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

Lock Bit Protection Modes

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

Programming the Flash

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

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

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

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

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

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

$\overline{Data}$ Polling: The AT89C51 features $\overline{Data}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on P0.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

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

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

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

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H,

031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

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

Programming Interface

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

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

Flash Programming Modes

Mode	RST	PSEN	ALE/PROG	EA/VPP	P2.6	P2.7	P3.6	P3.7
Write Code Data	H	L	⌇	H/12V(1)	L	H	H	H
Read Code Data	H	L	H	H	L	L	H	H
Write Lock Bit - 1	H	L	⌇	H/12V	H	H	H	H
Bit - 2	H	L	⌇	H/12V	H	H	L	L
Bit - 3	H	L	⌇	H/12V	H	L	H	L
Chip Erase	H	L	⌇(2)	H/12V	H	L	L	L
Read Signature Byte	H	L	H	H	L	L	L	L

Notes: 1. The signature byte at location 032H designates whether V_{PP} = 12 V or V_{PP} = 5 V should be used to enable programming.
2. Chip Erase requires a 10 ms $\overline{PROG}$ pulse.

Figure 3. Programming the Flash

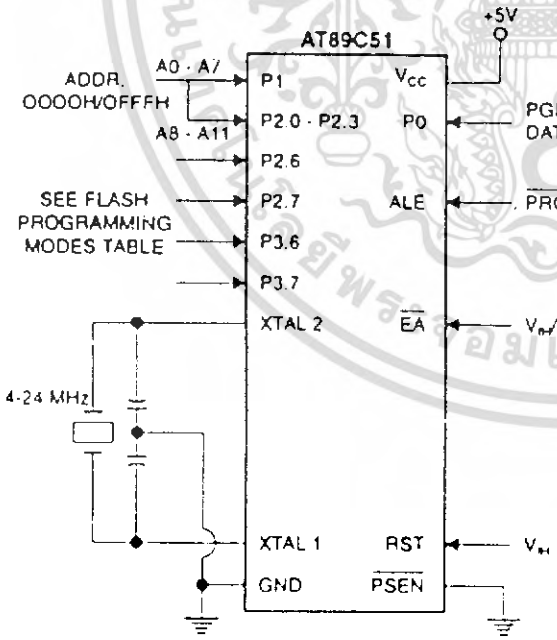
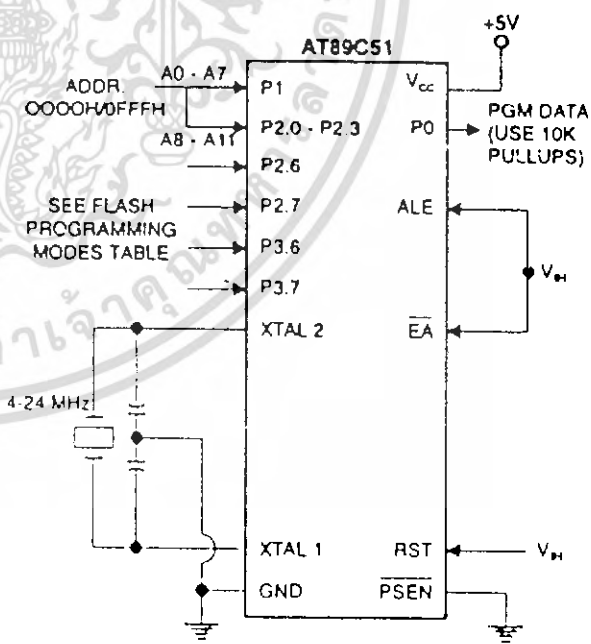


Figure 4. Verifying the Flash



Flash Programming and Verification Characteristics

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

Symbol	Parameter	Min	Max	Units
V_{PP}[1]	Programming Enable Voltage	11.5	12.5	V
I_{PP}[1]	Programming Enable Current		1.0	mA
$1/t_{CLCL}$	Oscillator Frequency	4	24	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	48t_{CLCL}		
t_{GHAX}	Address Hold After $\overline{PROG}$	48t_{CLCL}		
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	48t_{CLCL}		
t_{GHDX}	Data Hold After $\overline{PROG}$	48t_{CLCL}		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48t_{CLCL}		
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}[1]	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	1	110	µs
t_{AVQV}	Address to Data Valid		48t_{CLCL}	
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		48t_{CLCL}	
t_{EHQV}	Data Float After $\overline{ENABLE}$	0	48t_{CLCL}	
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		2.0	ms

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

DALLAS SEMICONDUCTOR

DS1202, DS1202S
Serial Timekeeping Chip

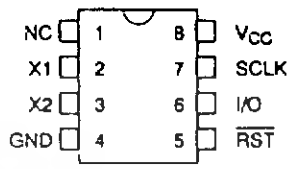
FEATURES

- Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation
- 24 x 8 RAM for scratchpad data storage
- Serial I/O for minimum pin count
- 2.0-5.5 volt full operation
- Uses less than 300 nA at 2 volts
- Single-byte or multiple-byte (burst mode) data transfer for read or write of clock or RAM data
- 8-pin DIP or optional 16-pin SOIC for surface mount
- Simple 3-wire interface
- TTL-compatible (V_{CC} = 5V)
- Optional industrial temperature range -40°C to +85°C

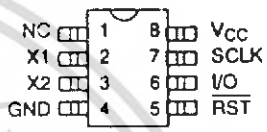
ORDERING INFORMATION

DS1202	8–pin DIP
DS1202S	16–pin SOIC
DS1202S8	8–pin SOIC

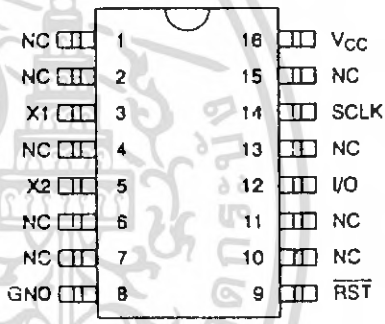
PIN ASSIGNMENT



8 PIN DIP



8–PIN SOIC
(208 mil)



16–PIN SOIC

PIN DESCRIPTION

NC	–	No Connection
X1, X2	–	32.768 KHz Crystal Input
GND	–	Ground
$\overline{RST}$	–	Reset
I/O	–	Data Input/Output
SCLK	–	Serial Clock
V_{CC}	–	Power Supply Pin

DESCRIPTION

The DS1202 Serial Timekeeping Chip contains a real time clock/calendar and 24 bytes of static RAM. It communicates with a microprocessor via a simple serial interface. The real time clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Interfacing the DS1202 with a microprocessor is simplified by using synchronous serial communication. Only three wires are required to communicate with the clock/RAM: (1) $\overline{RST}$ (Reset), (2) I/O (Data line), and (3) SCLK (Serial clock). Data can be transferred to and from the clock/

Patcharapak Pathueng

RAM one byte at a time or in a burst of up to 24 bytes. The DS1202 is designed to operate on very low power and retain data and clock information on less than 1 microwatt.

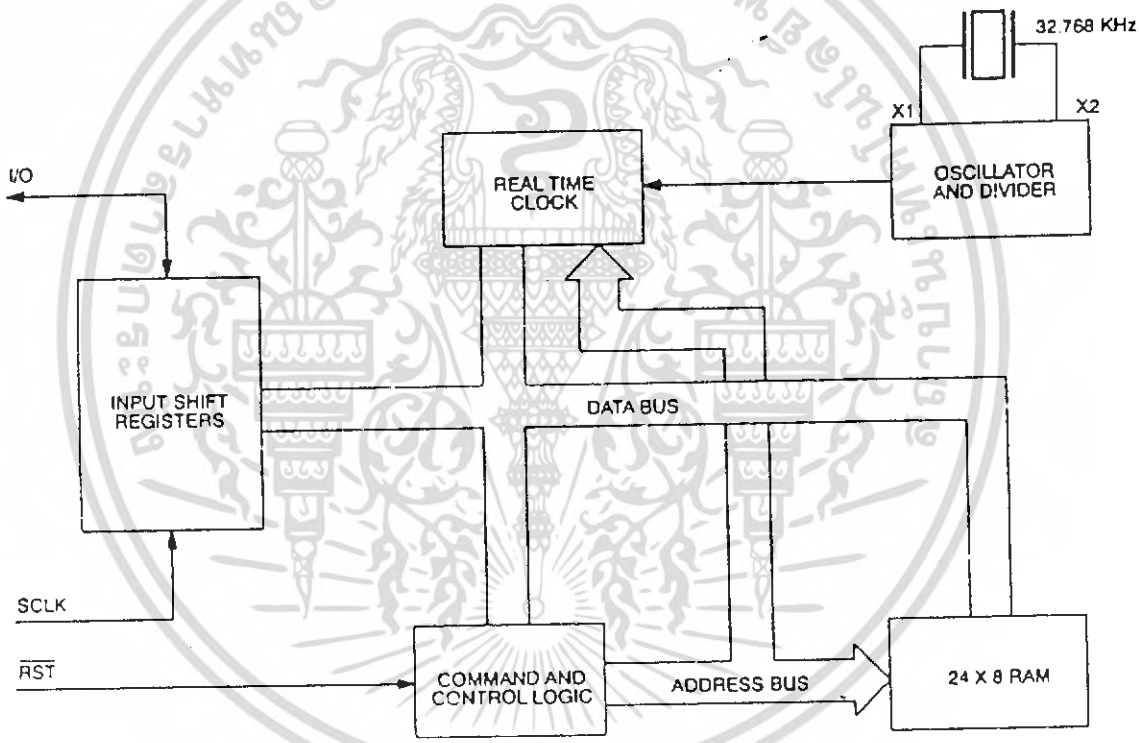
OPERATION

The main elements of the Serial Timekeeper are shown in Figure 1: shift register, control logic, oscillator, real time clock, and RAM. To initiate any transfer of data, $\overline{RST}$ is taken high and eight bits are loaded into the shift register providing both address and command information. Data is serially input on the rising edge of the SCLK. The first eight bits specify which of 32 bytes will be accessed, whether a read or write cycle will take place, and whether a byte or burst mode transfer is to occur. After the first eight clock cycles have occurred which load the command word into the shift register, additional clocks will output data for a read or input data for a write. The number of clock pulses equals eight plus eight for byte mode or eight plus up to 192 for burst mode.

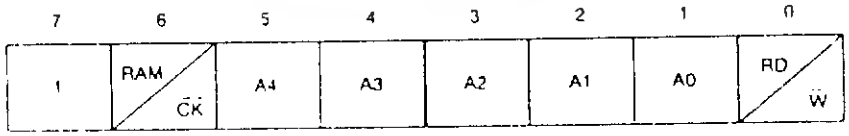
COMMAND BYTE

The command byte is shown in Figure 2. Each data transfer is initiated by a command byte. The MSB (Bit 7) must be a logic 1. If it is zero, further action will be terminated. Bit 6 specifies clock/calendar data if logic 0 or RAM data if logic 1. Bits one through five specify the designated registers to be input or output, and the LSB (Bit 0) specifies a write operation (input) if logic 0 or read operation (output) if logic 1. The command byte is always input starting with the LSB (bit 0).

DS1202 BLOCK DIAGRAM Figure 1



ADDRESS/COMMAND BYTE Figure 2



RESET AND CLOCK CONTROL

All data transfers are initiated by driving the $\overline{RST}$ input high. The $\overline{RST}$ input serves two functions. First, $\overline{RST}$ turns on the control logic which allows access to the shift register for the address/command sequence. Second, the $\overline{RST}$ signal provides a method of terminating either single byte or multiple byte data transfer. A clock cycle is a sequence of a falling edge followed by a rising edge. For data inputs, data must be valid during the rising edge of the clock and data bits are output on t he falling edge of clock. All data transfer terminates if the $\overline{RST}$ input is low and the I/O pin goes to a high impedance state. Data transfer is illustrated in Figure 3.

DATA INPUT

Following the eight SCLK cycles that input a write command byte, a deta byte is input on the rising edge of the next eight SCLK cycles. Additional SCLK cycles are ignored should they inadvertently occur. Data is input starting with bit 0.

DATA OUTPUT

Following the eight SCLK cycles that input a read command byte, a data byte is output on the falling edge of the next eight SCLK cycles. Note that the first data bit to be transmitted occurs on the first falling edge after the last bit of the command byte is written. Additional SCLK cycles retransmit the data bytes should they inadvertently occur so long as $\overline{RST}$ remains high. This operation permits continuous burst mode read capability. Data is output starting with bit 0.

BURST MODE

Burst mode may be specified for either the clock/calendar or the RAM registers by addressing location 31 decimal (address/command bits one through five = logical one). As before, bit six specified clock or RAM and bit 0 specifies read or write. There is no data storage capacity at locations 8 through 31 in the Clock/Calendar Registers or locations 24 through 31 in the RAM registers. When writing to the clock registers in the burst mode, the first eight registers must be written in order for the data to be transferred.

However, when writing to RAM in burst mode it is not necessary to write all 24 bytes for the data to transfer. Each byte that is written to will be transferred to RAM regardless of whether all 24 bytes are written or not.

CLOCK/CALENDAR

The clock/calendar is contained in eight write/read registers as shown in Figure 4. Data contained in the clock/calendar registers is in binary coded decimal format (BCD).

CLOCK HALT FLAG

Bit 7 of the seconds register is defined as the clock halt flag. When this bit is set to logic 1, the clock oscillator is stopped and the DS1202 is placed into a low-power standby mode with a current drain of not more than 100 nanoamps. When this bit is written to logic 0, the clock will start.

AM-PM/12-24 MODE

Bit 7 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode. bit 5 is the second 10 hour bit (20-23 hours).

WRITE PROTECT REGISTER

Bit 7 of write protect register is the write protect bit. The first seven bits (bits 0-6) are forced to zero and will always read a zero when read. Before any write operation to the clock or RAM, bit 7 must be zero. When high, the write protect bit prevents a write operation to any other register.

CLOCK/CALENDAR BURST MODE

The clock/calendar command byte specifies burst mode operation. In this mode the eight clock/calendar registers can be consecutively read or written (see Figure 4) starting with bit 0 of address 0.

RAM

The static RAM is 24 x 8 bytes addressed consecutively in the RAM address space.

RAM BURST MODE

The RAM command byte specifies burst mode operation. In this mode, the 24 RAM registers can be consecutively read or written (see Figure 4) starting with bit 0 of address 0.

19-0094; Rev 4; 3/94

MAXIM

Microprocessor Supervisory Circuits

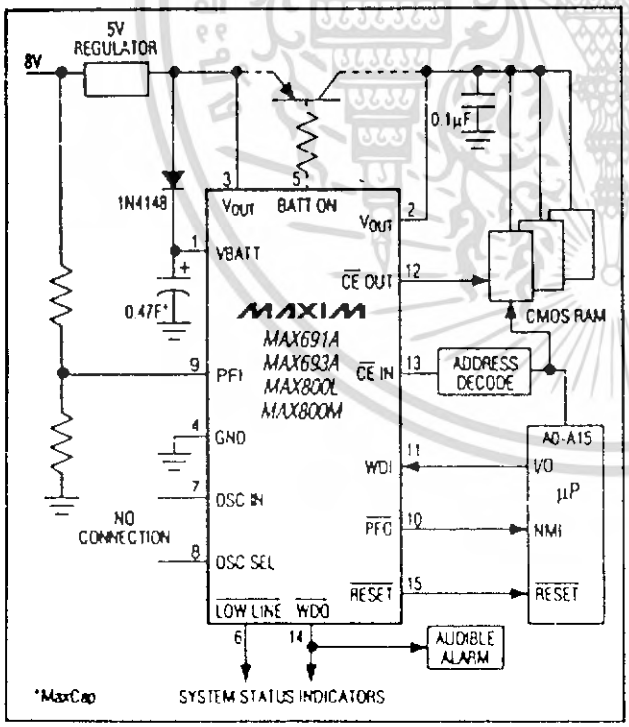
General Description

The MAX691A/MAX693A/MAX800L/MAX800M microprocessor (µP) supervisory circuits are pin-compatible upgrades to the MAX691, MAX693, and MAX695. They improve performance with 30µA supply current, 200ms typ reset active delay on power-up, and 6ns chip-enable propagation delay. Features include write protection of CMOS RAM or EEPROM, separate watchdog outputs, backup-battery switchover, and a RESET output that is valid with V_{CC} down to 1V. The MAX691A/MAX800L have a 4.65V typ reset-threshold voltage, and the MAX693A/MAX800M's reset threshold is 4.4V typ. The MAX800L/MAX800M guarantee power-fail accuracies to ±2%.

Applications

Computers
Controllers
Intelligent Instruments
Automotive Systems
Critical µP Power Monitoring

Typical Operating Circuit



Features

- ♦ 200ms Power OK/Reset Timeout Period
- ♦ 1µA Standby Current, 30µA Operating Current
- ♦ On-Board Gating of Chip-Enable Signals, 10ns Max Delay
- ♦ MaxCap® or SuperCap® Compatible
- ♦ Guaranteed $\overline{\text{RESET}}$ Assertion to V_{CC} = 1V
- ♦ Voltage Monitor for Power-Fail or Low-Battery Warning
- ♦ Power-Fail Accuracy Guaranteed to ±2% (MAX800L/M)
- ♦ Available in 16-Pin Narrow SO and Plastic DIP Packages

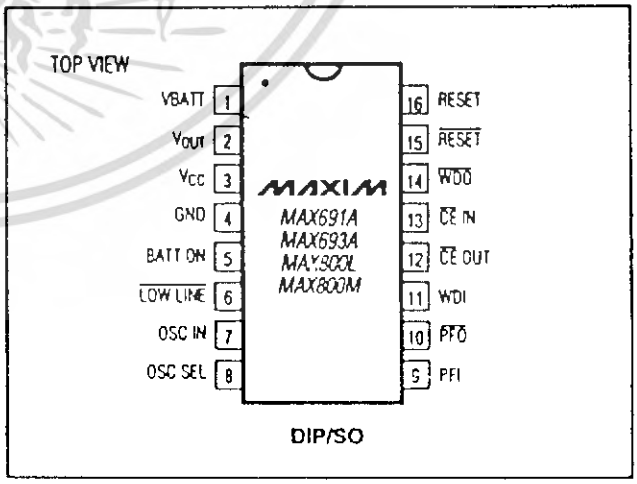
Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX691ACPE	0°C to +70°C	16 Plastic DIP
MAX691ACSE	0°C to +70°C	16 Narrow SO
MAX691ACWE	0°C to +70°C	16 Wide SO
MAX691AC/D	0°C to +70°C	Dice*
MAX691AEPE	-40°C to +85°C	16 Plastic SO
MAX691AESE	-40°C to +85°C	16 Narrow SO
MAX691AEWE	-40°C to +85°C	16 Wide SO
MAX691AEJE	-40°C to +85°C	16 CERDIP
MAX691AMJE	-55°C to +125°C	16 CERDIP

Ordering Information continued on last page.

** Dice are specified at T_A = +25°C.*

Pin Configuration



® SuperCap is a registered trademark of Baknor Industries

® MaxCap is a registered trademark of Carborundum Corp.

Pin Description

PIN	NAME	FUNCTION
1	VBATT	Battery-Backup Input. Connect to external battery or capacitor and charging circuit. If backup battery is not used, connect to GND.
2	V_{OUT}	Output Supply Voltage. When V_{CC} is greater than VBATT and above the reset threshold, V_{OUT} connects to V_{CC}. When V_{CC} falls below VBATT and is below the reset threshold, V_{OUT} connects to VBATT. Connect a 0.1µF capacitor from V_{OUT} to GND. Connect V_{OUT} to V_{CC} if no backup battery is used.
3	V_{CC}	Input Supply Voltage, 5V input.
4	GND	Ground. 0V reference for all signals.
5	BATT ON	Battery On Output. When V_{OUT} switches to VBATT, BATT ON goes high. When V_{OUT} switches to V_{CC}, BATT ON goes low. Connect the base of a PNP through a current-limiting resistor to BATT ON for V_{OUT} current requirements greater than 250mA.
6	$\overline{\text{LOW LINE}}$	$\overline{\text{LOW LINE}}$ output goes low when V_{CC} falls below the reset threshold. It returns high as soon as V_{CC} rises above the reset threshold.
7	OSC IN	External Oscillator Input. When OSC SEL is unconnected or driven high, the internal oscillator sets the reset delay and watchdog timeout period. The timing can also be adjusted by connecting an external capacitor to this pin. (see Figure 3). When OSC SEL is high or floating, OSC IN selects between fast and slow watchdog timeout periods.
8	OSC SEL	Oscillator Select. When OSC SEL is unconnected or driven high, the internal oscillator sets the reset delay and watchdog timeout period.When OSC SEL is low, the external oscillator input, OSC IN, is enabled (see Table 1). OSC SEL has a 10µA internal pull-up.
9	PFI	Power-Fail Input. This is the non-inverting input to the power-fail comparator. When PFI is less than 1.25V, $\overline{\text{PFO}}$ goes low. When PFI is not used, connect PFI to GND or V_{OUT}.
10	$\overline{\text{PFO}}$	Power-Fail Output. This is the output of the power-fail comparator. $\overline{\text{PFO}}$ goes low when PFI is less than 1.25V. This is an uncommited comparator, and has no effect on any other internal circuitry.
11	WDI	Watchdog Input. WDI is a three-level input. If WDI remains either high or low for longer than the watchdog time-out period, $\overline{\text{WDO}}$ goes low and reset is asserted for the reset timeout period. $\overline{\text{WDO}}$ remains low until the next transition at WDI. Leaving WDI unconnected disables the watchdog function. WDI connects to an internal voltage divider between V_{OUT} and GND, which sets it to mid-supply when left unconnected.
12	$\overline{\text{CE}}$ OUT	Chip-Enable Output. $\overline{\text{CE}}$ OUT goes low only when $\overline{\text{CE}}$ IN is low and V_{CC} is above the reset threshold. If CE IN is low when reset is asserted, $\overline{\text{CE}}$ OUT will stay low for 15µs or until $\overline{\text{CE}}$ IN goes high, whichever occurs first.
13	$\overline{\text{CE}}$ IN	Chip-Enable Input. The input to chip-enable gating circuit. If $\overline{\text{CE}}$ IN is not used, connect $\overline{\text{CE}}$ IN to GND or V_{OUT}.
14	$\overline{\text{WDO}}$	Watchdog Output. If WDI remains high or low longer than the watchdog timeout period, $\overline{\text{WDO}}$ goes low and reset is asserted for the reset timeout period. $\overline{\text{WDO}}$ returns high on the next transition at WDI. $\overline{\text{WDO}}$ remains high if WDI is unconnected.
15	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$ Output goes low whenever V_{CC} falls below the reset threshold. $\overline{\text{RESET}}$ will remain low typically for 200ms after V_{CC} crosses the reset threshold on power-up.
16	RESET	RESET is an active-high output. It is open drain, and the inverse of $\overline{\text{RESET}}$.

Detailed Description

$\overline{\text{RESET}}$ and RESET Outputs

TheMAX691A/MAX693A/MAX800L/MAX800M's $\overline{\text{RESET}}$ and RESET outputs ensure that the µP (with reset inputs asserted either high or low) powers up in a known state, and prevents code-execution errors during power-down or brownout conditions.

The $\overline{\text{RESET}}$ output is active low, and typically sinks 3.2mA at 0.1V saturation voltage in its active state. When deasserted, $\overline{\text{RESET}}$ sources 1.6mA at typically V_{OUT} - 0.5V. RESET output is open drain, active high, and typically sinks 3.2mA with a saturation voltage of 0.1V. When no backup battery is used, $\overline{\text{RESET}}$ output is guaranteed to be valid down to V_{CC} = 1V, and an external 10kΩ pull-down resistor on $\overline{\text{RESET}}$ insures that it will be valid with V_{CC} down to GND (Figure 1). As V_{CC} goes below 1V, the gate drive to the $\overline{\text{RESET}}$ output switch reduces accordingly, increasing the $r_{DS(ON)}$ and the saturation voltage. The 10kΩ pull-down resistor insures the parallel combination of switch plus resistor is around 10kΩ and the output saturation voltage is below 0.4V while sinking 40µA. When using a 10kΩ external pull-down resistor, the high state for $\overline{\text{RESET}}$ output with V_{CC} = 4.75V will be 4.5V typ. For battery voltages ≥ 2V connected to VBATT, RESET and $\overline{\text{RESET}}$ remain valid for V_{CC} from 0V to 5.5V.

Table 1. Reset Pulse Width and Watchdog Timeout Selections

OSC SEL	OSC IN	Watchdog Timeout Period		Reset Timeout Period
		Normal	Immediately After Reset	
Low	External Clock Input	1024 clks	4096 clks	2048 clks
Low	External Capacitor	(600/47pF x C)ms	(2.4/47pF x C)sec	(800/47pF x C)ms
Floating	Low	100ms	1.6sec	200ms
Floating	Floating	1.6sec	1.6sec	200ms

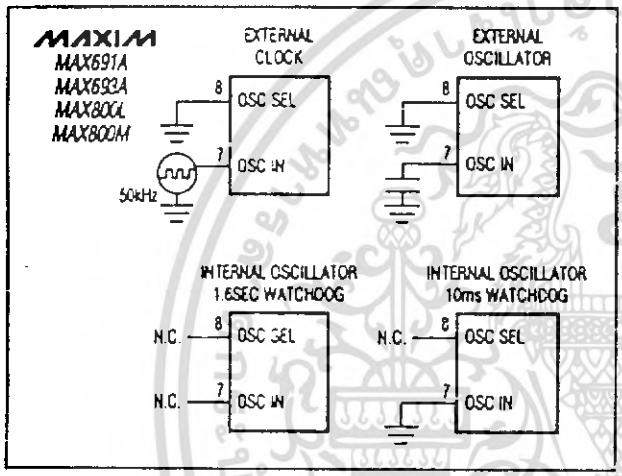


Figure 3. Oscillator Circuits

CE gate is enabled and passes all CE transitions. When reset is asserted, this path becomes disabled, preventing erroneous data from corrupting the CMOS RAM. All these parts use a series transmission gate from $\overline{CE}$ IN to $\overline{CE}$ OUT (Figure 4).

The 10ns max CE propagation delay from $\overline{CE}$ IN to $\overline{CE}$ OUT enables the parts to be used with most µPs.

Chip-Enable Input

The Chip-Enable Input ($\overline{CE}$ IN) is high impedance (disabled mode) while RESET and $\overline{RESET}$ are asserted.

During a power-down sequence where V_{CC} falls below the reset threshold or a watchdog fault, $\overline{CE}$ IN assumes a high-impedance state when the voltage at $\overline{CE}$ IN goes high or 15µs after reset is asserted, whichever occurs first (Figure 5).

During a power-up sequence, $\overline{CE}$ IN remains high impedance, regardless of $\overline{CE}$ IN activity, until reset is deasserted following the reset timeout period.

In the high-impedance mode, the leakage currents into this terminal are ±1µA max over temperature. In the low-impedance mode, the impedance of $\overline{CE}$ IN appears as a 75Ω resistor in series with the load at $\overline{CE}$ OUT.

The propagation delay through the CE transmission gate depends on both the source impedance of the drive to $\overline{CE}$ IN and the capacitive loading on the Chip-Enable Output ($\overline{CE}$ OUT) (see Chip-Enable Propagation Delay vs. $\overline{CE}$ OUT Load Capacitance in the *Typical Operating Characteristics*). The CE propagation delay is production tested from the 50% point of $\overline{CE}$ IN to the 50% point of $\overline{CE}$ OUT using a 50Ω driver and 50pF of load capacitance (Figure 6). For minimum propagation delay, minimize the capacitive load at $\overline{CE}$ OUT, and use a low output-impedance driver.

Chip-Enable Output

In the enabled mode, the impedance of $\overline{CE}$ OUT is equivalent to 75Ω in series with the source driving $\overline{CE}$ IN. In the disabled mode, the 75Ω transmission gate is off and $\overline{CE}$ OUT is actively pulled to V_{OUT}. This source turns off when the transmission gate is enabled.

$\overline{LOW\ LINE}$ Output

$\overline{LOW\ LINE}$ is the buffered output of the reset threshold comparator. $\overline{LOW\ LINE}$ typically sinks 3.2mA at 0.1V. For normal operation (V_{CC} above the $\overline{LOW\ LINE}$ threshold), $\overline{LOW\ LINE}$ is pulled to V_{OUT}.

Power-Fail Comparator

The power-fail comparator is an uncommitted comparator that has no effect on the other functions of the IC. Common uses include low-battery indication (Figure 7), and early power-fail warning (see *Typical Operating Circuit*).

Power-Fail Input

PFI is the input to the power-fail comparator. PFI has a guaranteed input leakage of ±25nA max over temperature. The typical comparator delay is 25µs from V_{IL} to V_{OL} (power failing), and 60µs from V_{IH} to V_{OH} (power being restored). If unused, connect this input to ground.

Power-Fail Output

The Power-Fail Output ($\overline{PFO}$) goes low when PFI goes below 1.25V. It typically sinks 3.2mA with a saturation voltage of 0.1V. With PFI above 1.25V, $\overline{PFO}$ is actively pulled to V_{OUT}.

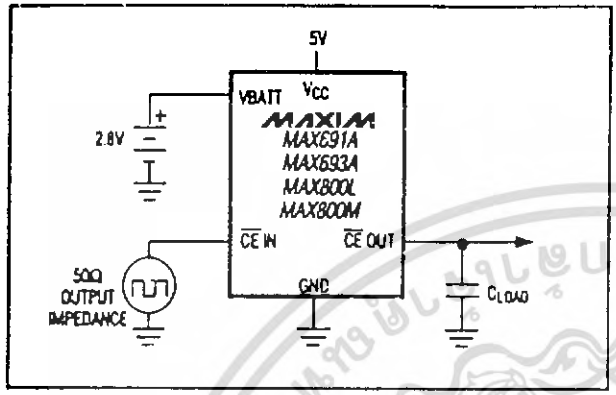


Figure 6. CE Propagation Delay Test Circuit

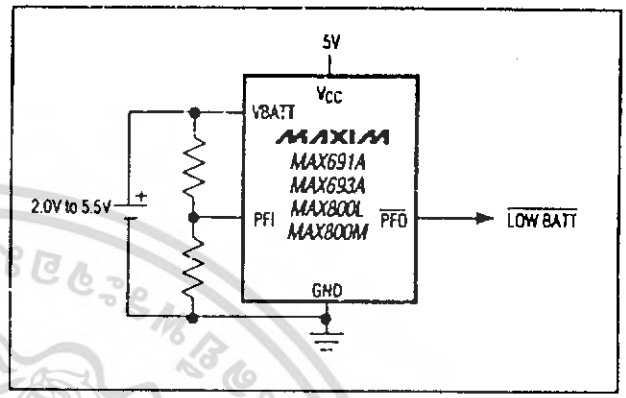


Figure 7. Low-Battery Indicator

Table 2. Input and Output Status In Battery-Backup Mode

PIN	NAME	STATUS
1	VBATT	Supply Current is 1µA max.
2	V_{OUT}	V_{OUT} is connected to VBATT through an internal PMOS switch.
3	V_{CC}	Battery switchover comparator monitors V_{CC} for active switchover.
4	GND	GND 0V, 0V reference for all signals.
5	BATT ON	Logic high. The open-circuit output is equal to V_{OUT}.
6	$\overline{LOWLINE}$	Logic Low*
7	OSC IN	OSC IN is ignored.
8	OSC SEL	OSC SEL is ignored.
9	PFI	The power-fail comparator remains active in the battery-backup mode for $V_{CC} \geq$ VBATT - 1.2V typ.
10	$\overline{PFO}$	The power-fail comparator remains active in the battery-backup mode for $V_{CC} \geq$ VBATT - 1.2V typ. Below this voltage, PFO is forced low.
11	WDI	Watchdog is ignored.
12	$\overline{CE}$ OUT	Logic high. The open-circuit voltage is equal to V_{OUT}.
13	$\overline{CE}$ IN	High Impedance
14	$\overline{WDO}$	Logic high. The open-circuit voltage is equal to V_{OUT}
15	$\overline{RESET}$	Logic low*
16	RESET	High Impedance*

** V_{CC} must be below the reset threshold to enter battery-backup mode.*

Battery-Backup Mode

Two conditions are required to switch to battery-backup mode: 1) V_{CC} must be below the reset threshold, and 2) V_{CC} must be below VBATT. Table 2 lists the status of the inputs and outputs in battery-backup mode.

Battery On Output

The Battery On (BATT ON) output indicates the status of the internal V_{CC}/battery-switchover comparator, which controls the internal V_{CC} and VBATT switches. For V_{CC} greater that VBATT (ignoring the small hysteresis effect), BATT ON typically sinks 3.2mA at 0.1V saturation voltage. In battery-backup mode, this terminal sources approximately 10µA from V_{OUT}. Use BATT ON to indicate battery-switchover status or to supply base drive to an external pass transistor for higher-current applications (see *Typical Operating Circuit*).

Input Supply Voltage

The Input Supply Voltage (V_{CC}) should be a regulated 5V. V_{CC} connects to V_{OUT} via a parallel diode and a large PMOS switch. The switch carries the entire current load for currents less than 250mA. The parallel diode carries any current in excess of 250mA. Both the switch and the diode have impedances less than 1Ω each. The maximum continuous current is 250mA, but power-on transients may reach a maximum of 1A.

Backup-Battery Input

The Backup-Battery Input (VBATT) is similar to the V_{CC} input except the PMOS switch and parallel diode are much smaller. Accordingly, the on resistances of the diode and the switch are each approximately 10Ω. Continuous current should be limited to 25mA and peak currents (only during power-up) limited to 250mA. The reverse leakage of this input is less than 1µA over temperature and supply voltage (Figure 8).

Output Supply Voltage

The Output Supply Voltage (V_{OUT}) pin is internally connected to the substrate of the IC and supplies current to the external system and internal circuitry. All open-circuit outputs will, for example, assume the V_{OUT} voltage in their high states rather than the V_{CC} voltage. At the maximum source current of 250mA, V_{OUT} will typically be 200mV below V_{CC}. Decouple this terminal with a 0.1µF capacitor.

Applications Information

The MAX691A/MAX693A/MAX800L/MAX800M are not short-circuit protected. Shorting V_{OUT} to ground, other than power-up transients such as charging a decoupling capacitor, destroys the device.

All open-circuit outputs swing between V_{OUT} and GND rather than V_{CC} and GND.

If long leads connect to the chip inputs, insure that these leads are free from ringing and other conditions that would forward bias the chip's protection diodes.

There are three distinct modes of operation:

1) Normal operating mode with all circuitry powered up. Typical supply current from V_{CC} is 35µA while only leakage currents flow from the battery.
2) Battery-backup mode where V_{CC} is typically within 0.7V below VBATT. All circuitry is powered up and the supply current from the battery is typically less than 60µA.
3) Battery-backup mode where V_{CC} is less than VBATT by at least 0.7V. VBATT supply current is 1µA max.

Using SuperCaps or MaxCaps with t
MAX691A/MAX693A/MAX800L/MAX80(

VBATT has the same operating voltage range as V_{C} and the battery switchover threshold voltages are ty cally ±30mV centered at VBATT, allowing use o SuperCap and a simple charging circuit as a back source (Figure 9).

If V_{CC} is above the reset threshold and VBATT is 0. above V_{CC}, current flows to V_{OUT} and V_{CC} from VBA until the voltage at VBATT is less than 0.5V above V_{C} For example, with a SuperCap connected to VBA and through a diode to V_{CC}, if V_{CC} quickly chang from 5.4V to 4.9V, the capacitor discharges throu V_{OUT} and V_{CC} until VBATT reaches 5.1V typ. Leaka current through the SuperCap charging diode and t internal power diode eventually discharges t SuperCap to V_{CC}. Also, if V_{CC} and VBATT start fr 0.1V above the reset threshold and power is lost V_{CC}, the SuperCap on VBATT discharges through V until VBATT reaches the reset threshold; then the b tery-backup mode is initiated and the current throu V_{CC} goes to zero.

Using Separate Power Suppli
for VBATT and V

If using separate power supplies for V_{CC} and VBAT VBATT must be less than 0.3V above V_{CC} when V_{CC} above the reset threshold. As described in the pre ous section, if VBATT exceeds this limit and power lost at V_{CC}, current flows continuously from VBATT V_{CC} via the VBATT-to-V_{OUT} diode and the V_{OUT}-to-V switch until the circuit is broken (Figure 8).

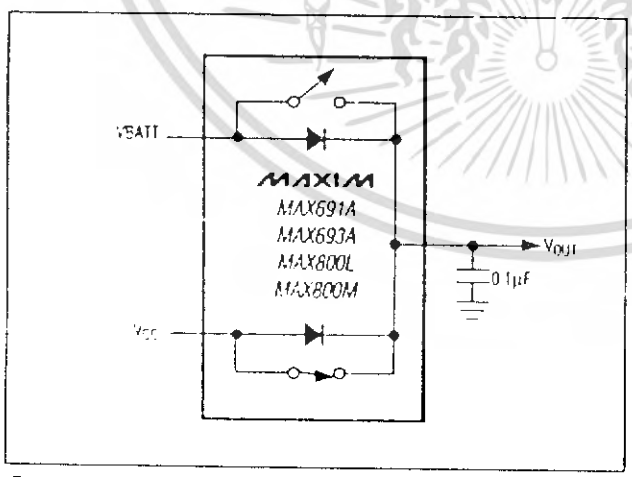


Figure 8. V_{CC} and VBATT to V_{OUT} Switch

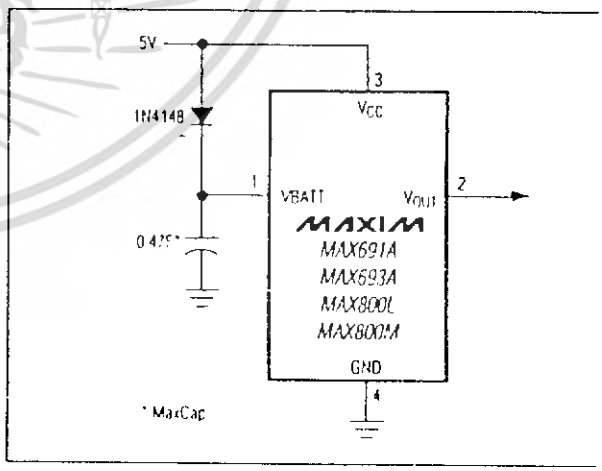


Figure 9. SuperCap or MaxCap on VBATT