แขนกล

Robot Arm

โดย

นาย เจตนา ปรักมาตร์

นาย ณัฐพล เทพสถิตย์ศิลป์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

แขนกล

Robot arm

โดย

นาย เจตนา ปรักมาตร์ 48015244

นาย ณัฐพล เทพสถิตย์ศิลป์ 48015248

อาจารย์ที่ปรึกษา

รศ.ดร. ชูชาติ ปิณฑวิรุจน์

ปริญญานิพนธ์นี้สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

ปริญญานิพนธ์ ปีการศึกษา 2550

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง แขนกล

ผู้จัดทำ

1. นาย เจตนา ปรักมาตร์ รหัส 48015244
2. นาย ณัฐพล เทพสถิตย์ศิลป์ รหัส 48015248

.....อาจารย์ที่ปรึกษา

(รศ.ดร. ชูชาติ ปิณฑวิรุจน์)

แขนกล

นาย เจตนา ปรักมาตร์ รหัส 48015244
นาย ณัฐพล เทพสถิตย์ศิลป์ รหัส 48015248
รศ.ดร. ชูชาติ ปิณฑวิรุจน์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2550

บทคัดย่อ

ปริญญานิพนธ์ นี้เป็นส่วนหนึ่งของการศึกษาในหลักสูตรวิศวกรรมศาสตร์บัณฑิต (วิศวกรรมอิเล็กทรอนิกส์) ในปัจจุบันมีการใช้อุปกรณ์ที่สามารถทำงานแทนมนุษย์มากขึ้นเรื่อยๆ แขนกลเป็นอุปกรณ์ประเภทหนึ่งที่ได้รับความนิยม โครงงานนี้จึงได้นำเสนอแขนกลที่ทำงานโดยใช้เซอร์โวมอเตอร์เป็นตัวขับเคลื่อน ให้แขนกลเคลื่อนที่ไปยังตำแหน่งที่ต้องการ โดยมีไมโครคอนโทรลเลอร์ (MCS-51) เป็นตัวควบคุมการหมุนของแขนกล และทำงานโดยใช้ภาษาซีเพื่อไปควบคุมการทำงานของเซอร์โวมอเตอร์

Robot Arm

Mr. Jatetana Prakmart ID48015244

Mr.Nuttapon Thepsatitsin ID48015248

Assoc. Prof. Dr. Chuchat Pintavirooj Advisor

Education Year 2007

Abstract

Nowadays the substitutions of machines to human work are increasing rapidly. The Robot Arm is the most popular device. Thus this thesis presents a construction of robot arm driven by Servo motor. This will control the movement of robot arm to the required position. Au MCS-51 controller will control the rotating of the robot arm . C language is used to control Servo motor

กิตติกรรมประกาศ

ปริญญานิพนธ์ ฉบับนี้กว่าจะสำเร็จลุล่วงไปได้ด้วยดีนั้นต้องพบกับปัญหาต่างๆมากมายในการงานแต่ก็ผ่านมาได้เพราะได้รับคำแนะนำและคำปรึกษาจาก รศ.ดร.ชูชาติ ปิณฑวิรุจน์ (อาจารย์ที่ปรึกษา) จนงานนี้ประสบความสำเร็จ รวมทั้งอาจารย์ทุกท่านในภาควิชาอิเล็กทรอนิกส์ที่อบรมสั่งสอนและแนะในเรื่องต่างๆและขอขอบคุณภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ได้สนับสนุนเครื่องมือต่างๆ ในการทำโครงงานให้สำเร็จ

สุดท้ายขอกราบขอบพระคุณ บิดา มารดา ที่คอยเป็นกำลังใจและให้การสนับสนุนในทุกๆด้านจนประสบความสำเร็จ

จัดทำโดย

นาย เจตนา ปรักมาตร์ 48015244

นาย ณัฐพล เทพสถิตย์ศิลป์ 48015248

สารบัญ

สารบัญ(ต่อ)

สารบัญรูป

หน้า

สารบัญรูป(ต่อ)

หน้า

บทที่ 1
บทนำ

ในปัจจุบันเทคโนโลยีด้านหุ่นยนต์ (Robot) ได้พัฒนาขึ้นอย่างรวดเร็วอันเป็นผลสืบเนื่องมาจากความต้องการความสะดวกสบายของมนุษย์ ทั้งเพื่อความสะดวกรวดเร็ว และแก้ปัญหาการขาดแคลนแรงงาน อีกทั้งการใช้คนเพื่อผลิตงานในโรงงานอุตสาหกรรม ก็มักก่อให้เกิดปัญหามาก และประสิทธิภาพการทำงานมีก็อยู่จำกัด จึงเริ่มหันมาใช้หุ่นยนต์แทนคนมากขึ้น

มนุษย์ได้พยายามที่จะคิดค้นสร้างสิ่งต่าง ๆ ขึ้นมาใช้ และวิทยาการสมัยใหม่ก็อำนวยให้การสร้างหุ่นยนต์ไม่ใช่เรื่องยากอีกต่อไป จากเหตุผลดังกล่าวข้างต้น ก็ทำให้พอที่จะมองเห็นได้ว่าในอนาคตอันใกล้หุ่นยนต์จะเข้ามามีบทบาทกับชีวิตมนุษย์อย่างแน่นอน ดังนั้นจึงมีความจำเป็นไม่น้อยที่จะต้องศึกษาและพัฒนารูปแบบของหุ่นยนต์ ให้สามารถใช้งานได้จริงในอุตสาหกรรม เพื่อรองรับความเจริญของการใช้หุ่นยนต์ที่จะเกิดขึ้นในอนาคต

ในโครงงานนี้ จะเป็นการออกแบบและสร้างแขนกล โดยมีวัตถุประสงค์ของงานคือ

1. ศึกษาระบบการอินเตอร์เฟส และการใช้โปรแกรมภาษาซีในการควบคุมแขนกล
2. ทำให้ได้ต้นแบบของแขนกล และพัฒนารูปแบบให้ดีขึ้นต่อไป
3. เพื่อให้เกิดความรู้ความเข้าใจในส่วนระบบเกียร์ แรงบิด และการเคลื่อนที่ของมอเตอร์
4. เพื่อควบคุมให้แขนกลสามารถเคลื่อนที่ไปยังตำแหน่งที่ต้องการได้

ในการออกแบบเลือกระบบขับเคลื่อนโดยใช้ เซอร์โวมอเตอร์ เริ่มติดตั้งชุดเฟืองส่งกำลัง และโครงสร้าง หลังจากนั้นทำการสั่งงานโดยใช้โปรแกรมภาษาซีผ่านวงจรอินเตอร์เฟส เพื่อไปควบคุมการทำงานเซอร์โวมอเตอร์

ประโยชน์ที่คาดว่าจะได้รับ

1. ทำให้ได้ต้นแบบในการพัฒนาแขนกล
2. รู้ปัญหาและแนวทางในการพัฒนาแขนกล
3. เข้าใจในการเขียนโปรแกรม แก้ไขคำสั่ง ซึ่งอาจมีการกำหนดขอบเขตงานของแขนกลมากยิ่งขึ้น
4. ได้ใช้ความคิดและแนวคิดสร้างสรรค์ในการทำงานโปรเจ็คเป็นการฝึกให้มีการทำงานเป็นระบบและมีความรับผิดชอบยิ่งขึ้น

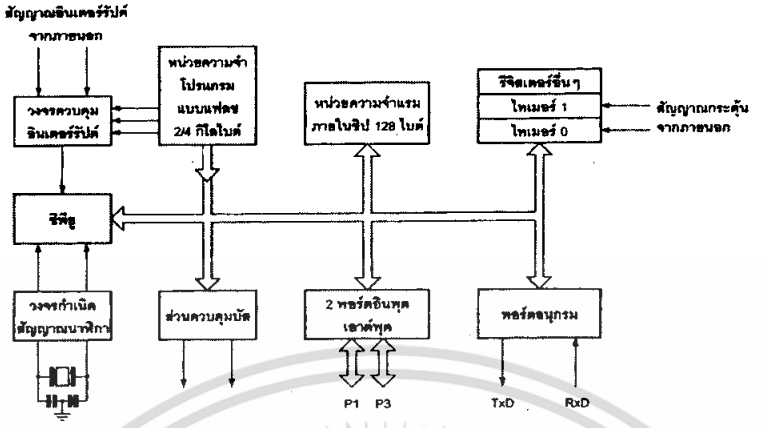
บทที่ 2
AT89C4051 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่ใช้ในโครงงานนี้นี้จะอ้างอิงถึงไมโครคอนโทรลเลอร์ในอนุกรม AT89C×051 ซึ่งมีหน่วยความจำภายในเป็นแบบแฟลช (flash memory) ของ Atmel Corporation มีเบอร์หลักๆคือ AT89C2051 และ AT89C4051 ซึ่งทั้งสองเบอร์มีความแตกต่างกันที่ความจุของหน่วยความจำโปรแกรม โดย AT89C2051 มีความจุ 2 กิโลไบต์ ในขณะที่ AT89C4051 มีความจุ 4 กิโลไบต์

2.1 คุณสมบัติของไมโครคอนโทรลเลอร์ MCS-51 อนุกรม AT89C×051

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลช สามารถลบและเขียนใหม่ได้พันครั้ง ความจุ 2 ถึง 4 กิโลไบต์ ขึ้นอยู่กับเบอร์ของไมโครคอนโทรลเลอร์
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาต์พุต
- มีขาพอร์ตสำหรับต่อใช้งาน 15 ขา โดยแบ่งเป็นพอร์ต 1 (8 บิต : P1.0-P1.7) และพอร์ต3 (7 บิต : P3.0-P3.5 และ P3.7)
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิต 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท

ในรูปที่ 2-1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C×051 จะเห็นได้ว่า เหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแต่แตกต่างกันที่หน่วยความจำโปรแกรมแบบแฟลชที่เพิ่มเติมเข้ามา และจำนวนขาพอร์ตที่น้อยกว่า



รูป 2-1 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89C×051

เบอร์ของไมโครคอนโทรลเลอร์	หน่วยความจำโปรแกรม	หน่วยความจำข้อมูล	จำนวนไทเมอร์/เคาน์เตอร์ 16 บิต
AT89C2051	แบบแฟลช ขนาด 2 กิโลไบต์	แรม 128 ไบต์	2
ATB9C4051	แบบแฟลช ขนาด 4 กิโลไบต์	แรม 128 ไบต์	2

ตารางที่2-1 รายละเอียดโดยสรุปบางส่วนของไมโครคอนโทรลเลอร์ AT89C×051 แบบแฟลชที่ Atmel ผลิตขึ้น

ในตารางที่ 2-1 แสดงรายละเอียดบางส่วนของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C×051 ที่ Atmel ผลิตขึ้น และมีใช้งานอยู่ในปัจจุบัน

2.2 การจัดขาของไมโครคอนโทรลเลอร์ AT89C×051

ไมโครคอนโทรลเลอร์ MCS-51ในอนุกรมAT89C×051 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังแสดงในรูปที่2-2 โดยมีรายละเอียดขึ้นต้น ดังนี้

ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5V

ขา GND เป็นขากราวด์ สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต 1(P1.0-P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย

ขาพอร์ต 3 (P3.0-P3.5,P3.7) มี 7 ขา สามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล “1” ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย

(float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดขั้นต้นต่อไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา R×D

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา T×D

P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 0 หรือขา INT0

P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา INT1

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาไทเมอร์จากภายนอกช่อง 0 หรือขา T0

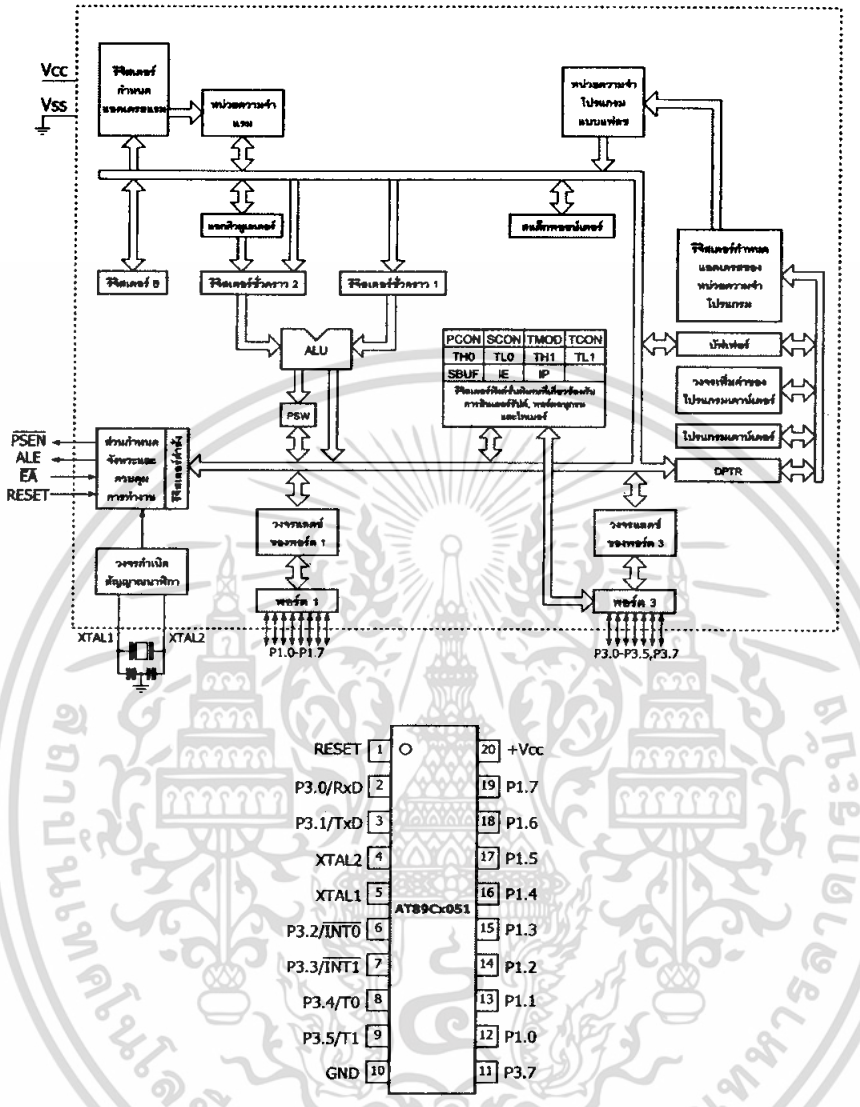
P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาอินเตอน์รัปต์จากภายนอกช่อง 1 หรือขา T1

P3.7 เป็นขาพอร์ตใช้งานทั่วไป

สำหรับ P3.6 โดยแท้จริงแล้วใน AT89C×051 มีให้ใช้งาน เพราะ P3.6 เป็นเอาต์พุตของโมดูลเปรียบเทียบแรงดันอะนาลอก (analog comparator) ที่มีอยู่ภายในตัว AT89C×051 ทุกตัว แต่ต้องอ่านค่าด้วยซอฟต์แวร์ผ่านทางรีจิสเตอร์ ไม่มีการต่อขาออกมาให้ใช้งานภายนอก

ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซต สถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องอย่างปกติ

ขา XTAL1 และ **XTAL2** เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

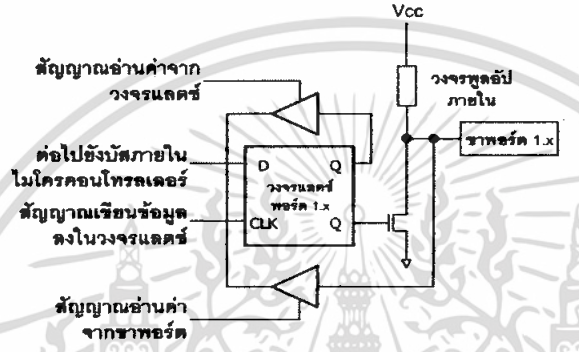


รูป 2-2 รายละเอียดโครงสร้างและหลักการจัดขาของไมโครคอนโทรลเลอร์ AT89C×051 ของ

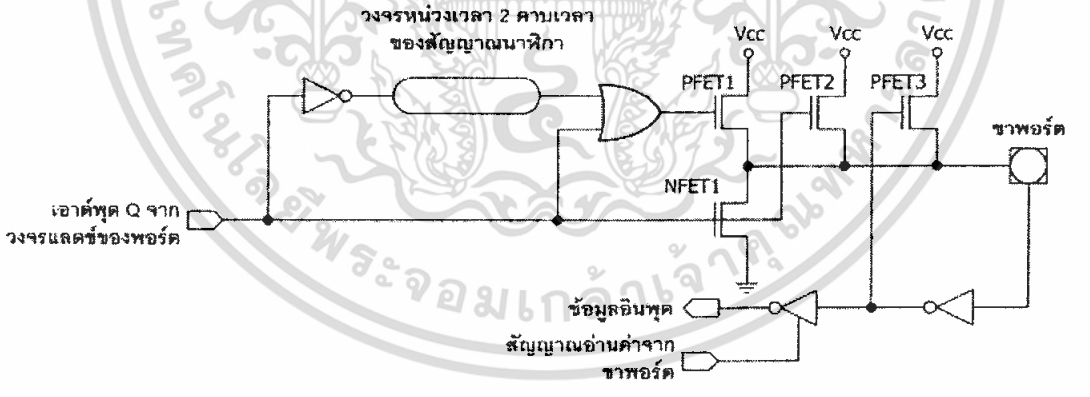
2.3 โครงสร้างและการทำงานของพอร์ท

ไมโครคอนโทรลเลอร์ AT89C×051 มีพอร์ตให้ใช้งาน 2 พอร์ตคือ พอร์ต 1 และพอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต แต่สำหรับพอร์ต 3 มีต่อให้ใช้งานทางฮาร์ดแวร์จริง 7 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้าและเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ AT89C×051 มีวงจรแลตช์และวงจรขับตลอดจนบัฟเฟอร์อินพุต ดังแสดงให้เห็นในสถาปัตยกรรมรูปที่ 2-2

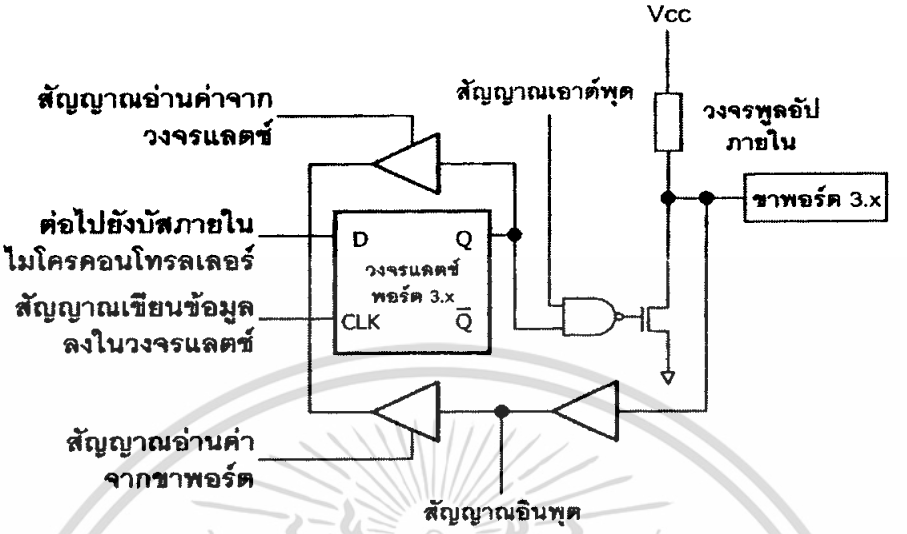
ในรูปที่ 2-3 เป็นวงจรของพอร์ต 1 วงจรแลตช์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิฟลอป การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระด้วยสัญญาณที่แยกจากกัน นั่นคือ สัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิปฟลอปในขณะที่ข้อมูลจะผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิฟลอป มีวงจรพูลอัพภายในที่แต่ละบิตของพอร์ต สำหรับรายละเอียดของวงจรพูลอัพแสดงรูปที่ 2-4



รูป 2-3 วงจรภายในของพอร์ต 1 ในไมโครคอนโทรลเลอร์ AT89C×051



รูป 2-4 วงจรพูลอัพภายในพอร์ต 1 และ 3 ของไมโครคอนโทรลเลอร์ AT89C×051



รูป 2-5 วงจรภายในของพอร์ต 3 ในไมโครคอนโทรลเลอร์ AT89C×051

ในรูปที่ 2-5 เป็นวงจรภายในของพอร์ต 3 จะเห็นได้ว่าคล้ายกับพอร์ต 1 มีการเพิ่มเติมวงจรบัฟเฟอร์ และวงจรอินพุตเอาต์พุตเมื่อทำงานในฟังก์ชั่นพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานในหน้าที่พิเศษได้เกือบทุกขา (เว้น P3.7)

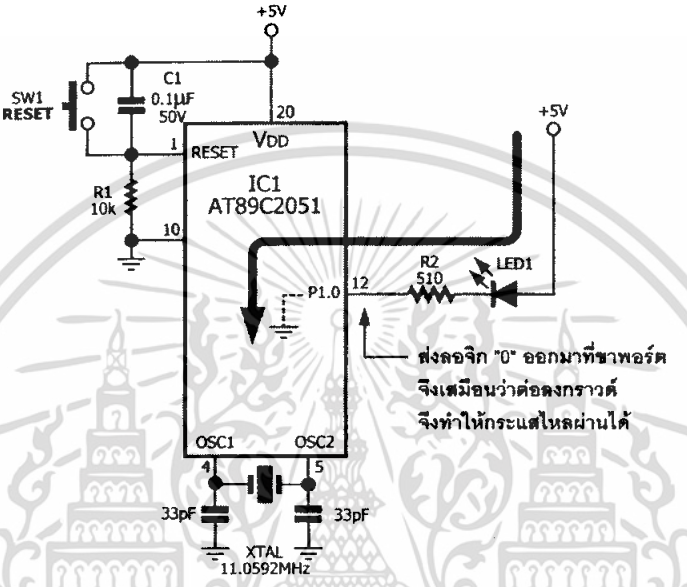
2.4 การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ AT89C×051 สามารถเป็นได้ทั้งอินพุตและเอาต์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ต

ในการกำหนดให้เป็นพอร์ตอินพุต ต้องเริ่มต้นด้วยการเขียนข้อมูล “1” มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดการทำงานของเฟตที่ใช้ในการขับสัญญาณเอาต์พุตของบิตนั้นๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัปภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น “1” สามารถรับสัญญาณลอจิก “0” จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไป เมื่อเป็นเช่นนี้ อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ AT89C×051 ควรกำหนดให้ทำงานในสภาวะลอจิก “0” จะดีและสะดวกที่สุด (ซึ่งในปัจจุบันอุปกรณ์อินพุตที่เชื่อมต่อไมโครคอลโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก “0” แล้ว)

2.5 การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาต์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งอมูลออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล “0” ออกไปทางเอาต์พุต ก็ให้เขียนข้อมูล “0” ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปขับเฟต ทำให้เฟตทำงาน ที่ขาพอร์ต



รูป 2-6 แสดงการขับโหลดในลักษณะกระแสซิงก์ของขาพอร์ตของไมโครคอนโทรลเลอร์

ที่กำหนดให้ทำงานก็จะเกิดลอจิก “0” ขึ้น ในทางตรงกันข้างหากต้องการส่งข้อมูล “1” ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก “1” ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

เมื่อใช้งานเป็นพอร์ตเอาต์พุต แต่ละขา (หรือแต่ละบิต) ของแต่ละพอร์ตมีความสามารถในการจ่ายกระแสหรือที่เรียกว่า กระแสซอร์ส (source current) ได้น้อยมาก (ในหน่วย µA) แต่จะจ่ายกระแสในลักษณะ กระแสซิงก์ (sink current) ซึ่งทำงานด้วยลอจิก “0” ได้สูงถึง 20 mA ต่อขาพอร์ต ในกรณีที่ใช้งานทุกพอร์ตเอาต์พุตจะสามารถจ่ายกระแสได้รวมกันสูงสุด 80 mA ดังนั้นในการใช้งานเป็นพอร์ตเอาต์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาตุพุตเพื่อช่วยในการขับกระแสอีกทางหนึ่ง หรือ ขับกระแสในลัษณะซิงก์ นั่นคือ ส่งลอจิก “0” ออกไปทางเอาต์พุตเพื่อขับโหลด ดังแสดงวงจรตัวอย่างในรูปที่ 2-6

2.6 การอ่านค่าลอจิกจากพอร์ต

ในไมโครคอนโทรลเลอร์ AT89C×051 สามารถอ่านค่าลอจิกจากพอร์ตได้ 2 ลักษณะ คือ อ่านจากขาพอร์ตโดยตรง และ อ่านจากวงจรแลตช์ ของแต่ละพอร์ต เหมือนกับในไมโครคอนโทรลเลอร์ MCS-51 มาตรฐาน

ในกรณีที่พอร์ตต่อกับขาเบสทรานซิสเตอร์ชนิด NPN และขาอิมิตเตอร์ของทรานซิสเตอร์ตัวนั้นต่อลงกราวด์ หากมีการส่งข้อมูล "1" ไปยังทรานซิสเตอร์ จะทำให้ทรานซิสเตอร์ทำงาน สถานะลอจิกที่ขาพอร์ตจะเป็น "0" เนื่องจากเมื่อทรานซิสเตอร์ทำงานจะเสมือนว่าขาพอร์ตนั้นถูกต่อลงกราวด์ ทำให้หากอ่านค่าลอจิกที่ขาพอร์ตจะได้ผลตรงข้ามกับที่ส่งออกมา แต่ถ้าหากอ่านค่าลอจิกที่วงจรแลตช์ จะได้ค่าที่ตรงกับค่าที่ต้องการส่งจริง ดังนั้น ในการอ่านค่าลอจิกจากพอร์ตจึงต้องเลือกวิธีการให้เหมาะสมกับอุปกรณ์ที่นำมาต่อด้วย

2.7 จังหวะการทำงานของไมโครคอนโทรลเลอร์ AT89C×051

เหมือนกับไมโครคอนโทรลเลอร์ MCS-51 มาตรฐาน นั่นคือ ใน 1 รอบการทำงานหรือ 1 แมชีนไซเกิล ซีพียูในไมโครคอนโทรลเลอร์ใช้เวลา 12 คาบเวลาของสัญญาณนาฬิกา นั่นคือ เวลาในการทำงาน 1 ไซเกิลมีค่าเท่ากับ 1 ms หรือมีความเร็วในการทำงานภายใน 1 MHz ในกรณีที่ใช้ความถี่สัญญาณนาฬิกา 12 MHz ดังนั้นถ้าต้องการทราบความเร็วของการทำงานภายในของไมโครคอนโทรลเลอร์ AT89C×051 สามารถหาได้จากค่าความถี่สัญญาณนาฬิกา หารด้วย 12 และถ้าต้องการหาค่าเวลาของ 1 รอบการทำงานหรือ 1 แมชีนไซเกิล สามารถทำได้โดยการหารส่วนกลับของความเร็ว ในการทำงานภายในของไมโครคอนโทรลเลอร์ AT89C×051 สามารถสรุปเป็นสูตรทางคณิตศาสตร์ได้ดังนี้

ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์เท่ากับ

ความถี่ของสัญญาณนาฬิกา (ค่าของคริสตอลที่ต่ออยู่ที่ขา XTAL1 และ XTAL2)/12

เวลา 1 แมชีนไซเกิล = 1/ ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์

อย่างไรก็ตาม ไมโครคอนโทรลเลอร์ AT89C×051 สามารถทำงานกับสัญญาณนาฬิกาความถี่สูงสุด 24MHz

2.8 การอินเตอร์รัปต์จากสัญญาณภายนอกของไมโครคอนโทรลเลอร์ AT89Cx051

สำหรับการทดลองนี้ เป็นตัวอย่างการใช้งานขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกของไมโครคอนโทรลเลอร์ AT89Cx051ซึ่งตรงกับขา P3.2 (INTO) และ P3.3 (INTI) นี้จะเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์ AT89Cx051 ตอบสนองการอินเตอร์รัปต์ จากสัญญาณ

ภายนอก โดยใช้สวิตช์กดติดปล่อยดับเป็นอุปกรณ์ในการกำเนิดสัญญาณ อินเตอร์รัปต์ เมื่อเกิดการอินเตอร์รัปต์ขึ้น ไมโครคอนโทรลเลอร์จะตอบสนองการอินต์รัปต์ ด้วยการขับLED

2.8.1 ความรู้พื้นฐานที่ควรทราบ

2.8.1.1 การจัดอินเตอร์รัปต์ในไมโครคอสโทรลเลอร์ MCS-51

เมื่อมีการอินเตอร์รัปต์เกิดขึ้น และมีการเอ็นเอเบิลการตอบสนองการอินต์รัปต์ไว้ ซีพียูจะกระโดดไปยังแอดเดรสที่เรียกว่า **แอดเรสอินเตอร์รัปต์ (interrupt vector address)** ดังนั้นจะต้องมีการเขียบโปรแกรมย่อยบริการอินเตอร์รัปต์ไว้ที่แอดเดรสอินเตอร์รัปต์เวอเตอร์นี้ โดยค่าของแอดเดรสรัปต์เวอกเอตร์จะแตกต่างกันไปในการอินเตอร์รัปต์แบบต่างๆดังนี้

การอินเตอร์รัปต์ภายนอกที่ขา INTO มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 0x03

การอินเตอร์รัปต์จากไทเมอร์ 0 มีค่าแอดเดรสอินเตอร์รัปเวกเตอร์อยู่ที่ 0x0B

การอินเตอร์รัปต์ภายนอกที่ขา INT1 มีค่าแอดเดรสอินเตอร์รัปเวกเตอร์อยู่ที่ 0x13H

การอินเตอร์รัปต์จากไทเมอร์ 1 มีค่าแอดเดรสอินเตอร์รัปเวกเตอร์อยู่ที่ 0x1BH

การอินเตอร์รัปต์จากพอร์ตอนุกรม NT1 มีค่าแอดเดรสอินเตอร์รัปเวกเตอร์อยู่ที่ 0x23H

การอินเตอร์รัปตไทเมอร์2 มีค่าแอดเดรสอินเตอร์รัปเวกเตอร์อยู่ที่ 0x2BH

2.8.2 รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51

2.8.2.1 รีจิสเตอร์เอ็นเอเบิลการอินเตอร์รัปหรือ IE (Interrupt Enable register)

มีแอดเดรสอยู่ที่ 0xA8 ในพื้นที่ของริจิสเตอร์ฟังก์ชั้นพิเศาหรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต ใช้เอ้นเอเบิลการตอลสนองการอินเตอร์รัปต์แบบต่างๆมีรายละเอียดดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต1	บิต 0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA (Global enable/disable interrupt) ใช้เอ็นเอเบิลและดิสเอเบิลการตอลสนองอินเตอร์รัปต์ทั้งหมด

" 0" ดิสเอเบิลการอินเตอร์รัปต์ นั่นคือ การกำหนดให้ไม่ตอบสนองการอินเตอร์รัปต์

"1 " เอ็นเอเบิลการอินเตอร์รัปต์ นั่นคือ กำหนดให้ตอบสนองการอินเตอร์รัปต์

นั่นคือ ถ้าต้องการให้ไมโครคอนโทรลเลอร์ตอบสนองการอินเตอร์รัปต์ไม่ว่าจะแหล่งกำเนิดใดจะต้องเซตบิตนี้ก่อนเสมอ สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ET2 (Timer 2 interrupt enable) ใช้เอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมากจากการโอเวอร์โลวหรือการแคปเจอร์ไนไทเมอรี/เคาวน์เตอร์ 2 มรเฉพาะในเบอร์ AT89C52 และ ในอนุกรม AT89xx เท่านั้นบิตนี้สามารถเซตและเคลียรีด้ววกระบวนการทางซอฟต์แวร์

ES (Serial port interrupt enable bit) ใช้เอ็นเอเบิลการอินเตอร์รัปต์เนื่องจากการรับหรือส่งข้อมูลบพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ET1 (Timer interrupt enable) ใช้เอ็นเอเบิลการอินเตอร์รัปต์จากการโอเวอร์โฟลวไทเมอร์ 1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

EX1 (External interrupt 1 enable bit) ใช้เอ็นเอเบิลการอินเตอร์รัปต์ภายนอกที่ป้อนเข้ามายังขา INT1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์

ET0 (Timer 0 interrupt enable) ใช้เอ็นเอเบิลการอินเตอร์รัปต์เนื่องจากสัญญาณภายนอกที่ป้อน 0 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

EX0 (External interrupt 1 enable bit) ใช้เอ็นเอเบิลการอินเตอร์รัปต์จากสัญญาณภายนอกที่ป้อนเข้ามายังขา INTO บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

สำหรับบิต 6 ของรีจิสเตอ์ IE ไม่มีการใช้งาน ต้องกำหนดให้เป็น 0 เสมอ

2.8.2.2 รีจิสเตอร์จัดลำดับความสำคัญการตอบสนองการอินเตอร์รัปตืหรือ IP

มีแอดเดรสอยู่ที่ 0xB8 ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต เลือกลำดับความสำคัญของการตอบสนองการ อินเตอร์รัปต์ได้ว่า ต้องการให้ตอบสนองสัญญาณอินเอตร์รัปต์จากแหล่งกำเนิดใดเป็นลำดับก่อนหลัง ถ้าต้องการให้อินเตอร์รัปต์จากแหล่งกำเนิดใดมีวามสำคัญสูงสุด ให้กำหนดที่บิตนั้นเป็น 1 มีรายละเอียดของรีจิสเอตร์ IP ดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต1	บิต 0
-	-	PT2	PS	PT1	PX1	PT0	PX0

PT2 (Timer 2 Interrupt priority bit) ใช้กำหนดระดับความสำคัญของการอินเตอร์รัปต์เนื่อจากการโอเวอร์โฟลวหรือการแคปเจอร์ไนไทเอมร์ 2 มีเฉพาะเบอร์ AT89C52 และ ในอนุกรม AT89Sxx เท่านั้น สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

PS (Serial port Interrupt priority bit) ใช้กำหนดความสำคัญของการอินเตอร์รัปต์เนื่องจากการรับหรือส่งข้อมูชทางพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตและเคลียร์ได้ทางซอฟต์แวร์

PT1 (Timer 1 Interrupt priority bit) ใช้กำหนดระดับความสำคัญของการอินเตอร์รัปต์เนื่องจากการโอเวอร์โฟลวในไทเมอร์ 1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

PX1 (External Interrupt1 priority bit) ใช้กำหนดระดับความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INT1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซปฟต์แวร์

PT0 (Timer 0 Interrupt priority bit) ใช้กำหนดระดับความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทเมอร์ 0 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

PX0 (External Interrupt 0 priority bit) ใช้กำหนดระดับความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INTO บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

สำหรับบิต 6 และ 7 ของรีจิสเตอร์ IP ไม่มีการใช้งาน ต้องกำหนดให้เป็น 0 เสมอ

2.8.3 การเข้าถึงรีจิสเตอร์เพื่อกำหนดการตอบสนองอินเตอร์รัปต์

สำหรับรีจิสเตอร์ของ MCS-51 ที่มีแอดเดรสที่ลงท้ายด้วย 8 หรือ 0 จะเป็นรีจิสเตอร์ที่สามารถเข้าถึงได้ในระดับบิต กล่าวคือในการเขียนโปรรแกรมควบคุมนั้นสามารถอ้างอิงประจำบิตได้เลย เช่น เดียวกับรีจิสเตอร์ IE ที่มีตำแหน่งแอสเดรสเป็น 0xA8ก็มีตำแหน่งแอดเดรสที่ลงท้ายด้วย 8 ดังนั้นเมื่อต้องการกำหนดค่าข้อมูลใกล้กับบิตใดๆภายในรีจิสเตอร์นี้ก็สามารถอ้างถึงได้ทันที เช่น ต้องการให้ค่าของบิต EA และ EX0 เป็น 1 ทั้งคู่ สามารถเขียนโปรแกรมภาษา C เพื่อกำหนดค่าได้ 2 แบบ

(ก) **แบบที่ 1** : เข้าถึงในระดับบิต

```
EA = 1;          // เปิดการตอบสนองของอินเตอร์รัปต์ทั้งหมด
EX0 = 1;         // เปิดการตอบสนองของอินเตอร์รัปต์ที่ขา P3.2 (INTO)
```

(ข) **แบบที่ 2** : กำหนดค่าข้อมูลให้กับรีจิสเตอร์โดยตรง

```
IE = 0 x 81 ;    // กำหนดให้ EA (บิต 7) และบิต EX0 (บิต 0) ของริจิสเตอร์ IE
                 // เป็น 1 บิตอื่นเป็น 0
```

จากการอ้างอิงทั้ง 2 แบบจะให้ผลลัพธ์เหมือนกัน ในแบบที่ 1 ค่อนข้างจะสะดวกตรงที่ถ้าจำชื่อบิตควบคุมได้ก็อ้างชื่อได้ทันที สำหรับในแบบที่ 2 จะต้องทราบทั้งชื่อริจิสเตอร์และตำแหน่งของบิตควบคุมด้วยจึงจะกำหนดค่าได้ถูกต้อง

2.9 การเขียนโปรแกรมย่อยบริการอินเตอร์รัปต์

มีหลักการโดยทั่วๆไป ดังนี้

1. เริ่มต้นด้วยแอดเดรสอินเอตร์รัปต์เวกเตอร์เสมอ เพื่อให้การตรวจสอบทำได้ง่าย และแยกส่วนของโปรแกรมย่อยนี้ออกจากโปรแกรมหบักหรือโปรแกรมย่อยๆอื่นๆอย่างชัดเจน

2. เมื่อเข้าสู่โปรแกรมย่อย ควรเก็บค่าของรีจิสเตอร์หรือแฟลกที่ใช้แสดงสถานนะต่างๆซึ่งต้องมีการใช้งานในโปรแกรมย่อยบริการอินเตอร์รัปต์นี้ไว้ในสแตกเสียก่อน เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นการทำงานของทั้งโปรแกรมบริการอินเตอร์รัปต์นี้และโปรแกรมหลัก

3. เมื่อเขียนโปรแกรมบริการอินเตอร์รัปต์เรียบร้อยแล้วให้การทำการคืนค่าของรีจิสเตอร์ที่นำมาใช้ในโปรแกรมบริการอินเตอร์รัปต์ ยกเว้นรีจิสเตอร์ที่ต้องการนำผลการกระทำในโปรแกรมบริการอินเตอร์รัปต์นี้ไปใช้งาน ในทางปฏิบัติจริง ไม่พบมากนัก และไม่นำให้เขียนโปรแกรมในลักษณะนี้

2.9.1 การประกาศฟังก์ชั่นตอบสนองอินเตอร์รัปต์

เมื่อเกิดการอินเตอร์รัปต์เนื่องจากแปล่งกำเนิดใดๆขึ้นและมีการเอ็นเอเบิลการตอบสนองอินเตอร์รัปต์นั้นๆไว้ล่วงหน้า จะทำให้เกิดการกระโดดของโปรแกรมเข้าไปกระทำคำสั่งภายในฟังก์ชั่นตอบสนองอินเตอร์รัปต์ของเหตุการณ์นั้นๆ โดยรูปแบบในการเขียนฟังก์ชั่นตอบสนองการอินเตอร์รัปต์ของภาษา C มีรูปแบบดังนี้

```
void function_name interrupt num
{
        คำสั่งที่ 1;
        คำสั่งที่ 2 ;
        ……..
                ……..
                คำสั่งที่ n;
}
```

โดยที่ function_name คือ ชื่อของฟังก์ชั่น สามารถตั้งชื่อได้เหมือนการประกาศฟังก์ชั่นทั่วๆไป

num คือ เลขลำดับอินเตอร์รัปต์ชั่นคือค่าลำดับของตัวเลขที่จะต้องระบุต่อท้ายคีย์เวิร์ด interrupt ซึ่งจะเขียนกำกับไว้ในส่วนหัวของฟังก์ชั่นตอบสนองอินเตอร์รัปต์ ซึ่งจะเป็นตัวบ่งบอกว่าฟังก์ชั่นดังกล่าวจะถูกเรียกตอบสนองเมื่อเกิดการอินเตอร์รัปจากเหตุการณ์ใด ซึ่งระบุได้จาก

สาเหตุการอินเตอร์รัปต์	**ลำดับที่ (num)**
อินเตอร์รัปต์จากภายนอกที่ขา P3.2 (INT0)	0
อินเตอร์รัปต์จากไทเมอร์ 0	1
อินเตอร์รัปต์จากภายนอกที่ขา P3.3 (INT1)	2
อินเตอร์รัปต์จากไทเมอร์ 1	3
อินเตอร์รัปต์จากพอร์ตอนุกรม	4

ตัวอย่าง

```
void external0 (void) interrupt 0          // ฟังก์ชั่นตอบสนองอินเตอร์รัปต์จากภายนกที่ขา P3.2
{
      /*      คำสั่งต่างๆที่กำหนด */
}
void timer0 (void) interrupt 1             // ฟังก์ชั่นตอบสนองอินเตอร์รัปต์จากไทเมอร์ 0
{
      /*      คำสั่งต่างๆที่กำหนด */
}
void serial (void) interrupt 4             // ฟังก์ชั่นตอบสนองอินเตอร์รัปต์จากพอร์ตอนุกรม
{
      /*      คำสั่งต่างๆที่กำหนด */
}
```

2.10 การใช้งานไทเมอร์ภายในไมโครคอนโทรลเลอร์ AT89Cx051

2.10.1 การทำงานเป็นไทเมอร์

เมื่อการทำงานเป็นตัวตั้งเวลาหรือไทเมอร์ **ค่าของรีจิสเตอร์จะเพิ่มขึ้นในทุกๆแมชีนไซเกิล** ดังนี้นเมื่อทำงานเป็นไทเมอร์รีจิสแตร์จะนับค่าของแมนชีนไซเกิลนั่นเอง และเนื่องจากแมชีไซเกิล

ประกอบด้วยคาบเวลาของวงจรกำเนิดสัญญาณนาฬิการ 12 คาบเวลา ดังนั้นอัตราในการนับของรีจิสเอตร์จึงเท่ากับ 1/12 ของความถี่สัญญานาฬิกา

2.10.2 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของไทเมอร์ /เคาน์เตอร์ 0และ 1

ไทเมอร์/เคาน์ตอร์ 0 และ 1 ในไมโครคอนโทรลเลอร์ AT89Cx051 มีริสจิสเตอร์เกี่ยวข้องเป็นพื้นฐานอยู่ 6ตัว ดังมีรายละเอียดต่อไปนี้

2.10.2.1 รีจิสเตอร์ไทเมอร์

มี 4 ตัวคือ

TL0 มีแอดเดรสอยู่ที่ 0x8A

TH0 มีแอดเดรสอยู่ที่ 0x8C

TL1 มีแอดเดรสอยู่ที่ 0x8B

TH1 มีแอดเดรสอยู่ที่ 0x8D

รีจิสเตอร์แต่ละตัวมีขนาด 8 บิต แต่ในการใช้งานโดยทั่วไปมักใช้ร่วมกันโดยจัดเป็นคู่ คือ

TL0 กับ THO รวมเป็นรีจิสเตอร์ Timer 0 ขนาด 16 บิต

TL1 กับTH1 รวมเป็นรีจิสเตอร์ Timer 1 ขนาด 16 บิต

โดย TL0และ TL1 เก็บข้อมูล 8 บิตล่าง ส่วน TH0และ TH1 เก็บข้อมูล 8 บิตบน รีจิสเตอร์ทั้ง 2 คู่เมื่อนำมาใช้ร่วมกันจะสามารถหาค่าของการนับได้สูงสุด 65,536 หรือ 0xFFFF เมื่อนับถึงค่านี้แล้วจะวนไปเริ่มนับ 0000 ใหม่ และเมื่อเกิดการนับรอบใหม่ บิต TF0 หรือ TF1 ในรีจิสเตอร์ TCON ที่ใช้ควบคุมการทำงานของไทเมอร์เกิดการเซต เพื่อแจ้งให้ทราบว่า นับเกินค่าสูสุดแล้ว การเซตบิต TF0 หรือ TF1 ขึ้นอยู่กับว่าเลือกใช้งานรีจิสเตอร์ตัวใด

2.10.2.2 รีจิสเตอร์ควบคุมการทำงานของไทเมอร์ /เคาน์เตอร์หรือ TCON

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 0x88 ในพื้นที่ของรีจิสเตอร์ SFRสามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต1	บิต 0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 (Timer 1 overflow flag) เซตด้วยกระวนการทางฮาร์ดแวร์เมื่อค่าของรีจิสเตอร์ Timer 1 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางฮาร์ดแวร์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น

TR1 (Timer 1 run control bit) ใช้เปิดการทำงานของไทเมอร์ 1 (เอ็นเอเบิลหรือดิสเอบเบิล) ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทเมอร์ 1 ทำงานต้องเซตบิตนี้ให้เป็น 1

TF0 (Timer 0 overflow flag) เซตด้วยกระวนการทางฮาร์ดแวร์เมื่อค่าของรีจิสเตอร์ Timer 0 เกิดการนับเกินหรือเกิดโอเวร์โฟลว การเคลียร์บิตนี้ทำได้ด้วยกระบวนการ ฮาร์ดแวร์เช่นกัน โดยบิตนี้จะเคลียร์เมื่อมีการอินเตรอร์รัปต์เกิดขึ้น

TR0 (Timer 0 run control bit) ใช้เปิดการทำงานของไทเมอร์ 1 (เอ็นเอเบิลหรือดิสเอบเบิล) ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทเมอร์ 0 ทำงานต้องเซตบิตนี้ให้เป็น 1

IE1 (External Interrupt 1 edge flag) บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 1 (INT1) ได้ และจะทำการเคลียร์เมื่อมีการบริการอินเตอร์รัปต์เกิดขึ้น

IT1 (Interrupt 1 type control bit) บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปต์ 1 (INT1) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์

"0" เลือกขอบขาลงของสัญญาณ (falling edge)

"1" เลือกระดับลอจิกต่ำ (low level triggered)

IE0 (External Interrupt 0 edge flag) บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 0 (INT0) ได้ และจะทำการเคลียร์เมื่อมีการบริการ อินเตอร์รัปต์เกิดขึ้น

IT0 (Interrupt 0 type control bit)) บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปต์ 0 (INT0) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์

"0" เลือกขอบขาลงของสัญญาณ (falling edge)

"1" เลือกระดับลอจิกต่ำ (low level triggered)

2.10.2.3 ริจิสเตอร์เลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์ หรือ TMOD

เป็นรีจิสเตอร์ขนาด 8 บิตมีแอดเดรสอยู่ 0x89 ในพื้นที่ของรีจิสเตอร์SFR ไม่สามารถเข้าถึงได้ในระดับบิต แบ่งการทำงานเป็น 2 ส่วนคือ 4 บิตล่างใช้เลือกโหมดการทำงานของไทเมอร์ 0

และ 4 บิตบน ใช้เลือกโหมดการทำงานของไทเมอร์ 1 ดังนั้นในการอธิบายการทำงานขออธิบายเพียงส่วนเดียวดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต1	บิต 0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
ไทเมอร์ 1				ไทเมอร์ 0			

GATE : ใช้เลือกลักษณะการควบคุมการทำงานของไทเมอร์/เคาน์เตอร์

"0" ไทเมอร์/เคาน์เตอร์ จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น 1 เรียกการควบคุมแบบนี้ว่า การควบุมกทางซอฟต์แวร์

"1" ไทเมอร์ / เคาน์เอตร์จะทำงานเมือบิต TRx ในรีจิสเตอร์ TCON เป็น 1 และสถานะลอจิกที่ขาอินพุตอินเตอร์รัปต์ INTO และ INT1 เป็น1 เรียกการควบคุมแบบนี้ว่า การควบุมกทางฮาร์ดแวร์

C/T (Timer or Counter selector) : ใช้เลือกลักษณะการทำงานของไทเมอร์/เคาน์เตอร์

"0" เลือกเป็นไทเมอร์ ใช้สัญญาณอินพุตจากสัญญาณนาฬิการภายในไมโครคอนโทรลเลอร์

"1" เลือกเป็นเคาน์เตอร์ โดยรับสัญญาณอินพุตทางขา T0 หรือ T1

M1 , M0 (Mode selector bit) : ใช้เลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์

"00" เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์ 13 บิต

"01" เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์ 16 บิต

"10" เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์ขนาด 8 บิต แบบตั้งค่าอัตโนมัติ

"11" สำหรับไทเมอร์ 0 เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์แยกส่วน โดยแยกออกเป็นไทเมอร์/เคาน์เตอร์ 8 บิต 2 ตัว รีจิสเอตร์ TL0 จะได้รับการควบคุมจากบิต TR0 ในรีจิสเตอร์ %ธฉฯ และรีจิสเตอร์ TH0 ซึ่งเป็นไทเมอร์ / เคาน์เตอร์ 8 บิตอีกตัวหนึ่ง จะได้รับการควบคุมจากบิต TR1 ในรีจิสเตอร์ TCON

ในกรณีของไทเมอร์ 1 เป็นการสั่งให้ไทเมอร์/เคาน์เตอร์ 1 หยุดทำงาน (ดิสเอเบิล)

2.10.3 โหมดการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1

ไทเมอร์ 0 และไทเมอร์ 1 สามารถเลือกโหมดการทำงานได้ 4 โหมด คือ

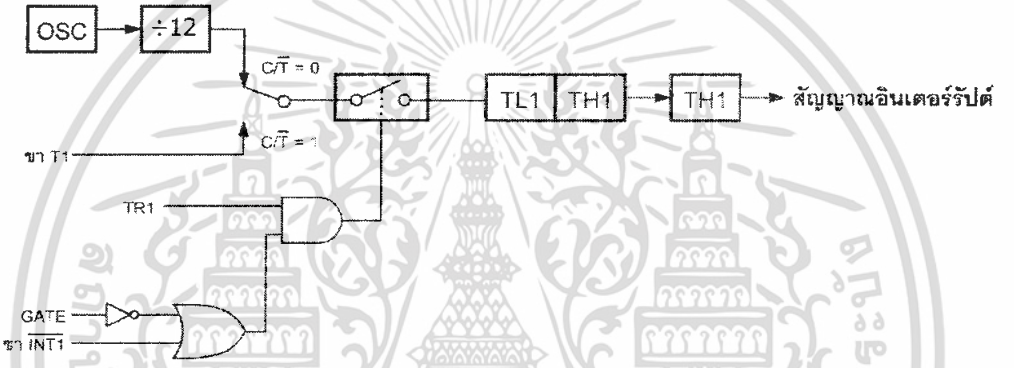
โหมด 0 : ไทเมอร์/เคาน์เตอร์ 13 บิต (13 bit timer/counter)

โหมด 1 : ไทเมอร์/เคาน์เตอร์ 16 บิต (16 bit timer/counter)

โหมด 2 : ตั้งค่าอัตโนมัติ บิต (8 bit auto-reload timer/counter)

โหมด 3 : ไทเมอร์ / เคาน์เตอร์แยกส่วน (split timer/counter) หรือไทเมอร์/เคาน์เตอร์ 8 บิต

การเลือกโหมดการทำงานของไทเมอร์ 0 และ 1 ทำได้โดยการกำหนดค่าของรีจิสเตอร์ TCON และ TMOD ร่วมกัน โดยTCON ใช้เอ็นเอเบิลไทเมอร์ ส่วนTMOD ใช้เลือกโหมดและลักษณะการทำงาน



รูป 2-7 ไดอะแกรมการทำงานในโหมด 0 และ ของไทเมอร์ 1

2.10.3.1 โหมด 0 : ไทเมอร์/เคาน์เตอร์ 13 บิต

มีไดอะแกรมการทำงานแสดงในรูปที่ 2-7 ในที่นี้จะใช้ไทเมอร์ 1 ในการอธิบาย โหมดนี้จะเป็นการกำหนดให้ใช้งานรีจิสเตอร์ TL1 เพียง 5 บิต และ TH1 ครบ 8 บิต โดย TL0 จะทำหน้าที่คล้ายกับเป็นปรีสเกลเลอร์หาร 32 สัญญาณอินพุตสำหรับการนับจะเลือกจากสัญญาณนาฬิกาภายในหรือภายนอกผ่านทางขา T1 ขึ้นอยู่กับการควบคุมของบิต C/T และGATE ในรีจิสเตอร์ TMOD ,บิต TR1ในรีจิสเตอร์ TCON และสถานะของลอจิกที่ขาอินพุต INT1 เมื่อ TL1 นับครบ 32 คือจาก 0-31 ก็จะส่งสัญญาณไปยัง TH1เพื่อทำการเพิ่มค่า ดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 13 บิต เมื่อทำการนับครบรอบ ก็จะทำการเซตบิต TF1ในรีจิสเตอร์ TCON

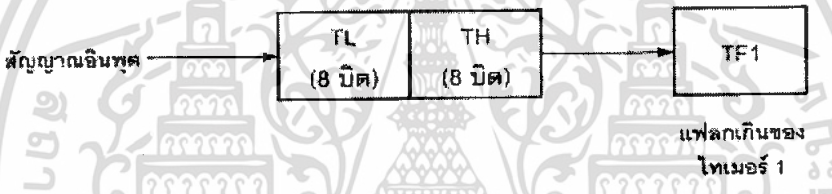
ส่วนการทำงานในโหมดนี้ของไทเมอร์/เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการเพียงแต่เปลี่ยนรีจิเตอร์และขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์/เคาน์เตอร์ 0

2.10.3.2 โหมด 1 : ไทเมอร์/เคาน์เตอร์ 16 บิต

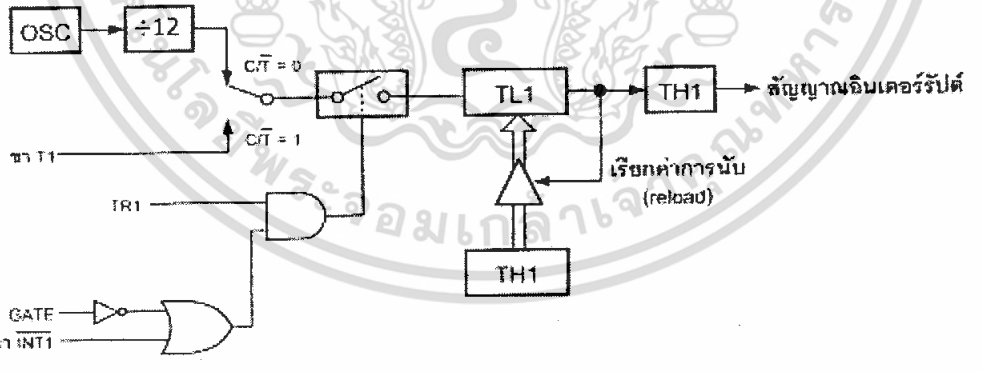
มีไดอะแกรมการทำงานแสดงในรูปที่ 2-8 ในที่นี้จะใช้ไทเมอร์ 1 ในการอธิบาย การทำงานในโหมดนี้จะคล้ายกับโหมด 0 แต่จะใช้งานรีจิสเตอร์ TL1 และ TH1 ครบ 8 บิต ดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 1 6บิต คือ 0x0000-0xFFFF เมื่อนับครบรอบ ค่าของการนับเปลี่ยนจาก 0xFFFF เป็น 0x0000 ก็จะเซตบิต TF1 ในรีจิสเตอร์ TCON ส่วนการทำงานในโหมดนี้ของไทเมอร์ 0 มีลักษณะเหมือนกันทุกประการ เพียงแต่เปลี่ยนริจิสเตอร์และสัญญาณที่เกี่ยวข้องให้เป็นขอบไทเมอร์ 0

2.10.3.3 โหมด 2 : ไทเมอร์/เคาน์เตอร์ 8 บิตแบบตั้งค่าอัตโนมัติ

มีไดอะแกรมการทำงานแสดงในรูปที่ 2-9 ในที่นี้จะใช้ไทเมอร์ 1 ในการอธิบายการทำงานในโหมดนี้จะแยกรีจิสเอตร์ไทเมอร์ออกเปน 2 ตัว ตัวละ 8 บิต โดยรีจิสเตอร์ TL1 ทำหน้าที่เป็นตัวนับค่า ส่วน TH1



รูป 2-8 ไดอะแกรมการทำงานในโหมด 1 ไทเมอร์ 1



รูป 2-9 ไดอะแกรมการทำงานในโหมด 2 ของไทเมอร์ 1

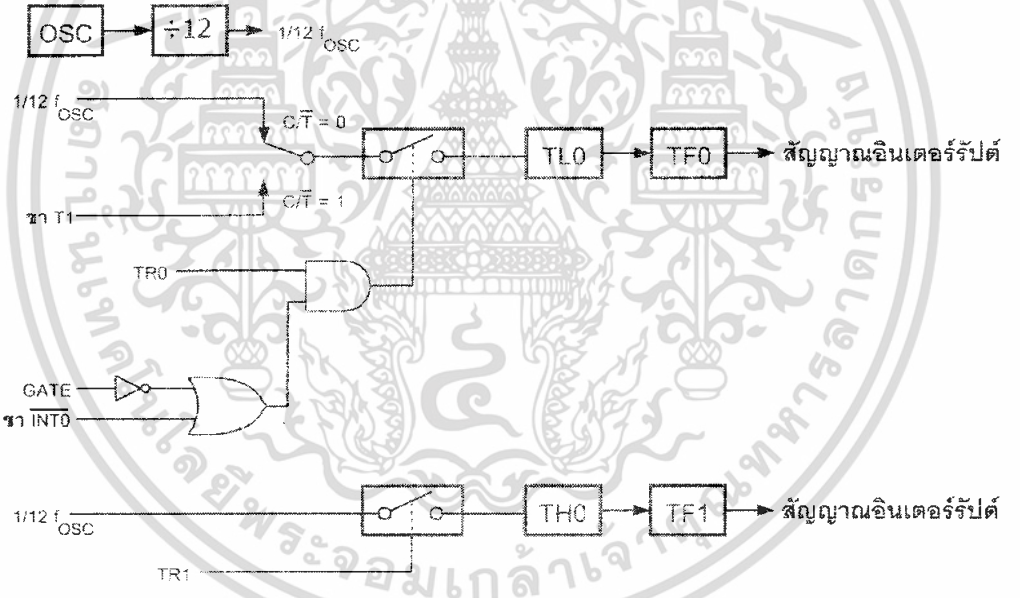
ใช้ในการเก็บค่าเริ่มต้นของการนับ เมื่อเริ่มต้นการทำงาน ค่าของรีจจิสเอตร์ TH1 จะถูกส่งไปยังรีจิสเตอร์ TL1 ทำให้เมื่อเริ่มต้นการทำงาน ค่าของรีจิสเตอร์ TL1 และ TK1 จะเหมือนกัน เมื่อ TL1 นับถึง 0xFF และจะเริ่มต้นการนับรอบใหม่ จะทำการเซตบิต TF1 พร้อมๆกับการับค่าการนับเริ่มต้นจาก TH1 ใหม่ดดยอัตโนมัติ หรือเรียกกระบวนการนี้ว่า รีโหลด (reload) แม้ว่าจะมี

การส่งค่าเริ่มต้นไปยัง TL1 แล้วก็ตาม ค่าของข้อมูลในรีจิสเตอร์ TH1 ก็ยังคงเป็นค่าเดิม ไม่มีการเปลี่ยนแปลง จนกว่าจะมีการกำหนดค่าใหม่ด้วยกระบวนการทางซอฟต์แวร์

ส่วนการทำงานในโหมดนี้ของไทเมอร์/เคาน์เตอร์ 0 มีลักษณะเหมือนกันทุกประการ **เพียงแต่เปลี่ยนรีจิสเตอร์และขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์/เคาน์เตอร์ 0**

2.10.3.4 โหมด 3 : ไทเมอร์/เคาน์เตอร์แยกส่วนหรือไทเมอร์เคาน์เตอร์ 8 บิต

ในโหมดนี้เป็นฌหมดเดียยที่การทำงานของไทเมอร์ 0 และไทเมอร์ 1 ไม่เหมือนกัน ขออธิบายในส่วนของไทเอมร์ 1 เมื่อเข้าสู่โหมดนี้ จะเป็น การสั่งให้ไทเมอร์หยุดนับ ค่าของการนับก่อนหน้านี้จะถูกเก็บไว้ในรีจิสเตอร์ไทเมอร์ 1 มีลักษณะการทำงานเหมือนกับการดิสเอเบิลไทเมอร์ 1 ด้วยการเคลียร์บิต TR1 ในรีจิสเตอร์ TCON



รูป 2-10ไดอะแกรมการทำงานในโหมด 3 ของไทเมอร์ 0

ส่วนการทำงานของไทเมอร์ 0 ในโหมดนี้มีไดอะแกรมการทำงานแสดงในรูปที่2-10 การทำงานในโหมดนี้จะแยกรีจิสเตอร์ไทเมอร์ 0 ออกเป็น 2 ตัว ตัวละ 8 บิต คือรีจิสเตอร์ TL0 และ TH1 โดยแยกการทำงานออกจากกัน รีจิสเตอร์ TL0 สามารถเลือกการทำงานได้เหมือนกับไทเมอร์/เคาน์เตอร์ตามปกติ ส่วนรีจิสเตอร์ TH0 สามารถทำงานในโหมดไทเมอร์ได้เพียงอย่างเดียว กล่าวคือสามารถรับสัญญาณอินพุตจากสัญญาณนาฬิการภายในเพียงทางเดียวเท่านั้น แต่การแจ้งการนับเกินยังคงเหมือนเดิม หากแต่ TL0 แจ้งผ่านบิต TF0 ในขณะที่TH0 จะแจ้งผ่านทางบิต TF1

2.10.3.5 **ข้อมูลที่ใช้ในการเลือกโหมดการทำงานของไทเมอร์ 0 และ 1**

เนื่องจากมีตัวแปรอยู่หลายตัวที่ใช้ในการควบคุมและเลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1 เพื่อให้เกิดความสะดวกในการใช้งานจึงได้ทำการสรุปข้อมูลที่ใช้ในการกำหนดข้อมูลที่ใช้ในการกำหนดรูปแบบการทำงานของไทเมอร์ 0 และ 1 ไว้ในตารางที่ 2-2 ถึง 2-5

อย่างไรก็ตาม ข้อมูลที่นำมายกเป็นตัวอย่างขั้นต้นนี้ สามรถเปลี่ยนแปลงได้ตามความต้องการของผู้ใช้งาน ไม่จำเป็นต้องยึดค่าเหล่านี้ไว้เสมอไป แต่สำหรับผู้เริ่มต้นใช้งานควรใช้ค่าตัวอย่างที่ให้วในการเขียนโปรแกรมควบคุมก่อน จนกว่ามีความชำนาญจึงค่อยปรับเปลี่ยนต่อไป

ตาราง2-2 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ ไทเมอร์/เคาน์เตอร์0ทำงานเป็นไทเมอร์

โหมด	ฟังก์ชั่นของไทเมอร์ 0	TMOD	
		การควบคุมจากภายใน	การควบคุมจากภายนอก
0	ไทเมอร์/เคาน์เตอร์ 13 บิต	0x00	0x08
1	ไทเมอร์/เคาน์เตอร์ 16 บิต	0x01	0x09
2	8 บิตตั้งค่าอัตโนมัติ	0x02	0x0A
3	ไทเมอร์/เคาน์เตอร์ แยกส่วน	0x03	0x0B

หมายเหตุ

การควบคุมจากภายใน : ควบคุมให้ไทเมอร์เปิดปิดด้วยการเซตและเคลียร์บิต TR0 โดยกระบวนการทางซอฟต์แวร์

การควบคุมจากภายนอก : ควบคุมให้ไทเมอร์เปิดปิดด้วยการตรวจสอบการเปลี่ยนแปลง จาก “1” เป็น “0” ที่ขา INT0 (P3.2) เมื่อ TR0 = “1”

ตาราง2-3แสดงข้อมูลของรีจิสเตอร์ TMODเพื่อกำหนดให้ไทเมอร์/เคาน์เตอร์0ทำงานเป็นคาน์เตอร์

โหมด	ฟังก์ชั่นของ ไทเมอร์ 0	TMOD	
		การควบคุมจาก ภายใน	การควบคุมจาก ภายนอก
0	ไทเมอร์/เคาน์เตอร์ 13 บิต	0x04	0x0C
1	ไทเมอร์/เคาน์เตอร์ 16 บิต	0x05	0x0D
2	8 บิตตั้งค่าอัตโนมัติ	0x06	0x0E
3	ไทเมอร์/เคาน์เตอร์ แยกส่วน	0x07	0x0F

หมายเหตุ

การควบคุมจากภายใน : ควบคุมให้ไทเมอร์เปิดปิดด้วยการเซตและเคลียร์บิต TR0 โดยกระบวนการทางซอฟต์แวร์

การควบคุมจากภายนอก : ควบคุมให้ไทเมอร์เปิดปิดด้วยการตรวจสอบการเปลี่ยนแปลง จาก “1” เป็น “0” ที่ขา INT0 (P3.2) เมื่อ TR0 = “1”

ตาราง2-4 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ไทเมอร์/เคาน์เตอร์ 1ทำงานเป็นไทเมอร์

โหมด	ฟังก์ชั่นของ ไทเมอร์ 0	TMOD	
		การควบคุมจาก ภายใน	การควบคุมจาก ภายนอก
0	ไทเมอร์/เคาน์เตอร์ 13 บิต	0x00	0x80
1	ไทเมอร์/เคาน์เตอร์ 16 บิต	0x10	0x90
2	8 บิตตั้งค่าอัตโนมัติ	0x20	0xA0
3	หยุดทำงาน	0x30	0xB0

หมายเหตุ

การควบคุมจากภายใน : ควบคุมให้ไทเมอร์เปิดปิดด้วยการเซตและเคลียร์บิต TR1 โดยกระบวนการทางซอฟต์แวร์

การควบคุมจากภายนอก : ควบคุมให้ไทเมอร์เปิดปิดด้วยการตรวจสอบการเปลี่ยนแปลง จาก “1” เป็น “0” ที่ขา INT1 (P3.3) เมื่อ TR1 = “1”

ตาราง2-5 แสดงข้อมูลของรีจิสเตอร์ TMOD เพื่อกำหนดให้ไทเมอร์/เคาน์เตอร์ 1ทำงานเป็นไทเมอร์

โหมด	ฟังก์ชั่นของ ไทเมอร์ 0	TMOD	
		การควบคุมจาก ภายใน	การควบคุมจาก ภายนอก
0	ไทเมอร์/เคาน์เตอร์ 13 บิต	0x40	0xC0
1	ไทเมอร์/เคาน์เตอร์ 16 บิต	0x50	0xD0
2	8 บิตตั้งค่าอัตโนมัติ	0x60	0xE0
3	-	-	-

หมายเหตุ

การควบคุมจากภายใน : ควบคุมให้ไทเมอร์เปิดปิดด้วยการเซตและเคลียร์บิต TR1 โดยกระบวนการทางซอฟต์แวร์

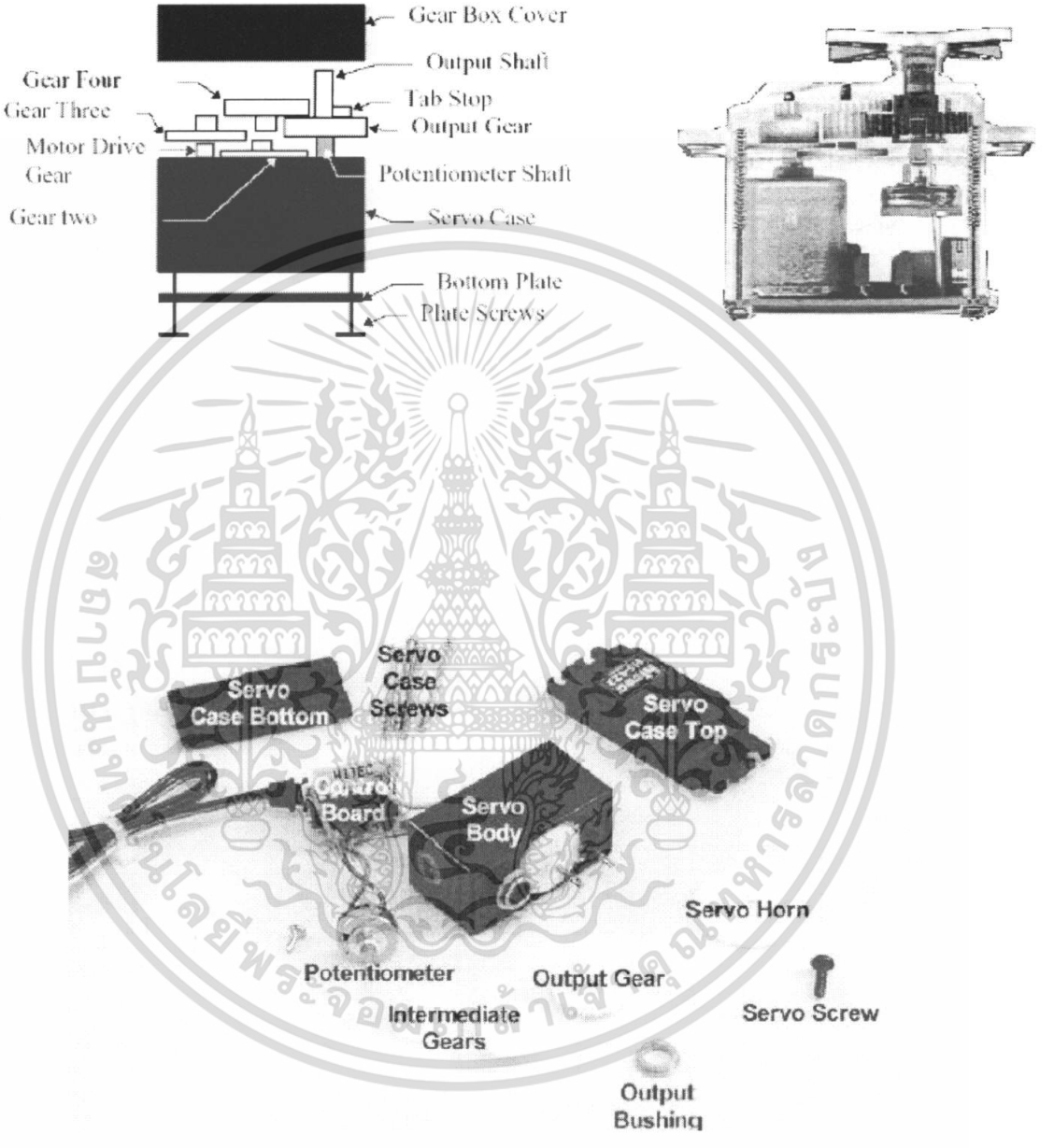
การควบคุมจากภายนอก : ควบคุมให้ไทเมอร์เปิดปิดด้วยการตรวจสอบการเปลี่ยนแปลง จาก “1” เป็น “0” ที่ขา INT1 (P3.3) เมื่อ TR1 = “1”

บทที่ 3

Servo motor

3.1 Servo motor คืออะไร

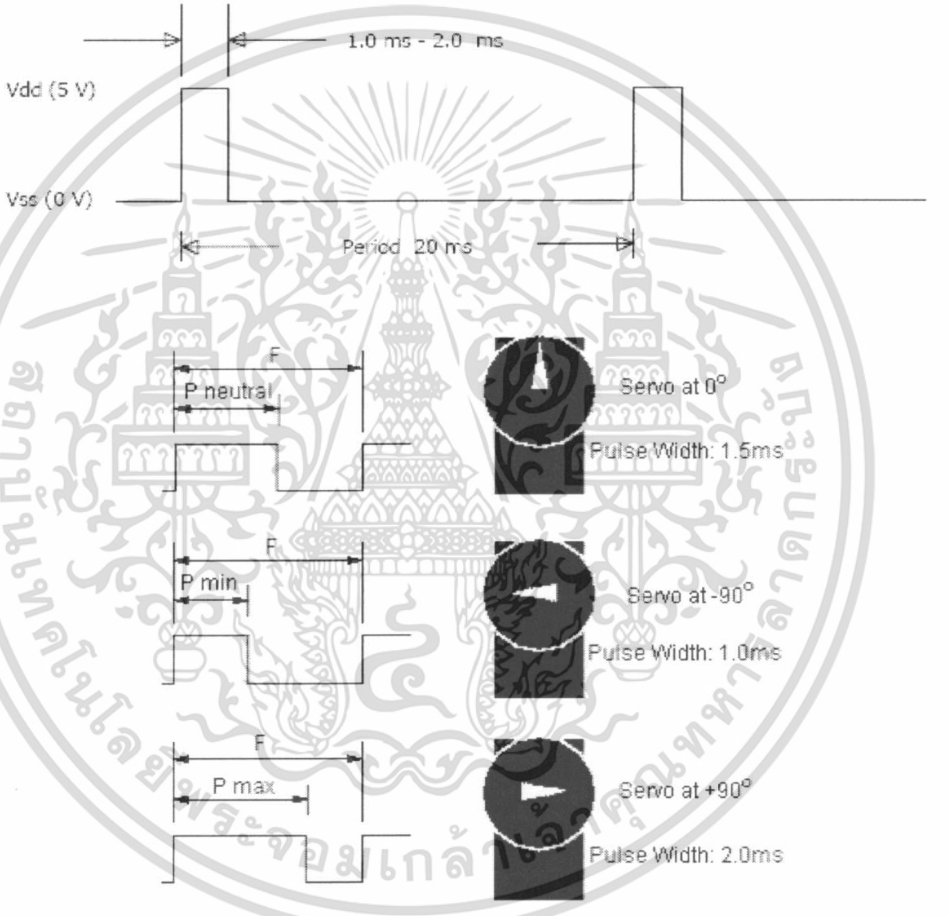
Servo motor คือ มอเตอร์ไฟฟ้ากระแสตรง (DC motor) ที่ถูกประกอบรวมกับ ชุดเกียร์ และส่วนควบคุม ต่างๆไว้ในโมดูลเดียวกัน หรือภายในกล้องพลาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อให้งานเพียง 3 เส้นเท่านั้น คือ VCC , GNDและ สายสัญญาณควบคุม (Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้าย หรือขวาได้จากสายสัญญาณเพียงเส้นเดียว โดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณ พัลส์วิตมอด(PWM) แบบ TTL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 โวลต์ ขึ้นอยู่กับบุณสมบัติของ มอเอตร์แต่ละตัว ข้อดีของมอเตอร์ชนิดนี้ก็คือ จะมีขนาดเล็กน้ำหนักเบาให้แรงบิตสูงเกินพลังงานน้อย และสามารถควบคุม ด้วยแรงดันลอจิกที่เป็น TTL ได้โดยตรงไม่จำเป็นต้องต่อวลจรขับ (Driver) อื่นๆ เพราะมอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้ว ซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่งหรือทิศทางองศาที่ต้องการได้ โดยอาศัยสัญญาณความกว้างพัลส์ ที่ป้อนให้มอเตอร์ แต่เซอร์โวมอเตอร์นี้จะหมุนได้แต่เพียงในช่วงประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้น หรือ บางรุ่นอาจหมุนได้ถึง 210 องศา แต่จะไม่สามารถหมุนเป็นวงรอบได้เนื่องจากโครงสร้างภายในจะประกอบด้วย ตัวต้านทานชนิดปรับค่าด้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และตัวต้านทานนี้จะพูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้นเซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้เพียงแต่ประมาณ 180 องศา หรือ ครึ่งรอบเท่าเท่านั้น เพื่อป้องกันความเสียหายที่จะเกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360 องศา) นั้นก็สามารถทำได้ โดยจะต้องทำการปรับแต่ง (Modify)ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์ ซึ่งวิธีการต่างๆจะได้กล่าวไว้ในภายหลัง



รูป 3-1 ส่วนประกอบต่างๆของ Servo motor

3.2 หลักการทำงานของ Servo motor

การควบคุมการทำงานของ เซอร์โวมอเตอร์ ทำได้โดย การป้อนสัญญาณความกว้างพลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดความกว้างของพัลส์นั้นๆโดยทั่วไปแล้วความกว้างของสัญญาณพัลส์จะมีจุดให้อ้างอิง 3 จุด ดังรูป คือ



รูป 3-2 ขนาดความกว้างของพัลส์ที่ใช้ควบคุมเซอร์โวมอเตอร์

- สัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศาหรือจุด กึ่งกลางของมอเตอร์

- สัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม -90 องศาหรือในทิศทางทวนเข็มนาฬิกา
- สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม +90 องศาหรือในทิศทางตามเข็มนาฬิกา

ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆนั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่างๆโดยอ้างอิงจากจุดทั้ง 3 จุดที่กล่าวมาแล้วนี้ ตัวอย่างเช่น ถ้าต้องการให้มอเตอร์หมุนไปที่มุม -45 องศา เราก็จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 ms เป็นต้น และ สัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุกๆ 20 ms (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้

โดยหลักการก็คือ จะอาศัยการเปรียบเทียบช่วงเวลาของความกว่างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์ ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงการหมุนของมอเตอร์ เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่า (VR)เปลี่ยนแปลงไป เป็นผลทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงตามไปด้วย โดยในขณะที่เราป้อนสัญญาณความกว่างพัลส์ให้กับมอเตอร์ทางขาสัญญาณควบคุม สัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้ง 2 ไม่เท่ากันมอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั่งค่าเวลาความกว้างพัลส์ของ วงจร RC เปลี่ยนแปลงจนเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

3.3 การทำงานของเซอร์โวมอเตอร์

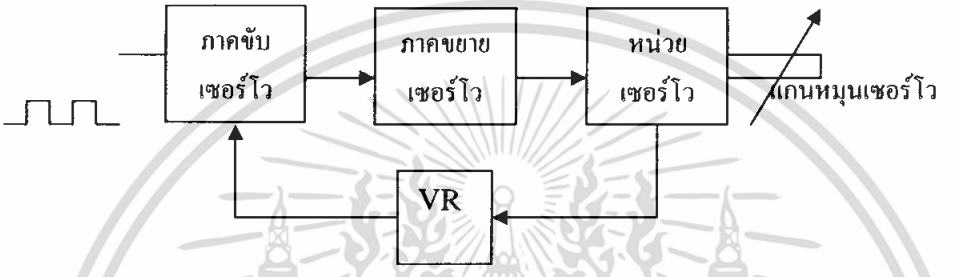
ในเซอร์โวมอเตอร์หนึ่งตัวจะประกอบไปด้วย 3 ภาคการทำงานแต่ละภาคมีหนาที่และการทำงานดังนี้

ภาคขับเซอร์โว ประกอบด้วย วงจรสร้างสัญญาณพัลซ์ และวงจรเปรียบเทียบสัยญาณพัลซ์ที่สร้างขึ้น กับสัญญาณพัลซ์ I/P ที่รับเข้ามา

ภาคขยายเซอร์โซ ประกอบด้วย วงจร RC Network ที่ช่วยหน่วงสัญญาณให้เซอร์โวสามารถทำงานได้ตอลดช่วงคาบเวลา จนกระทั่มีสัญญาณลูกต่อไปมา รวมถึงวงจรกลับขั้วแรงดันไฟฟ้าควบคุมทิศทางการหมุนของมอเตอร์

หน่วยเซอร์โว ประกอบด้วย มอเตอร์ความเร็วสูง เฟืองทดรอบ แกนหมุน อุปกรณ์ต่างๆ และVR (ตัวต้านทานปรับค่าได้) ทำหน้าที่ป้อนกลับตำแหน่ง (Position Feedback)

ซึ่งในขณะที่มอเตอร์หมุน VR จะถูกปรับค่า Feedback กลับมาปรับและเปรียบเทียบค่าความกว้างของพัลซ์ที่ภาคขับเซอร์โว เมื่อขนาดความกว่างของพัลซ์ มีค่าเฉลี่ยของค่าแรงดันเท่ากับมอเตอร์จะหยุดหมุนทันที ซึ่งรูปที่ 3-3 ได้แสดงภาคการทำงานของเซอร์โวมอเตอร์ตามที่ได้กล่าวไว้ข้างต้น และได้แสดงไว้ดังรูปต่อไปนี้



รูป 3-3 แสดงภาคการทำงานของเซอร์โวมอเตอร์

บทที่ 4

การสื่อสารผ่านพอร์ตอนุกรม กับโมโครคอนโทรลเลอร์ MCS-51

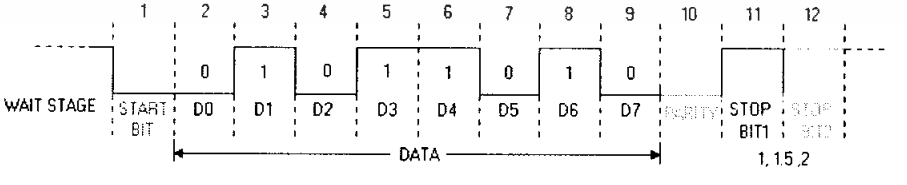
ไมโครคอนโทรลเลอร์ตระกูล MXS-51 มีวงจรสื่สารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด (วงจรสื่อสารแบบฟลูดูเพล็กซ์ หมายถึง วงจรสื่อสารที่สามารถทำการรับและส่งข้อมูลในลักษณะที่ 2 ทิศทางได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P 3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบเฟรชเป็นแบบอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 แต่ในปัจจุบันสามารถติดต่อกัน ในมาตรฐาน RS-422หรือRS-485 โดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

4.1 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารแบบอะซิงโครนัส คือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอดหรือ บอดเรต (baud rate) มีหน่วยเป็นบิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกัน คือ

1. บิตเริ่มต้น (start bit) มีขนาด1 บิต
2. บิตข้อมูลแบบอนุกรมมีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity bit) มีขนาด1 บิตหรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด(stop bit) มีขนาด 1 บิต



รูป 4-1 แบบข้อมูลอนุกรมแบบอะซิงโครนัส

รูปที่ 4-1 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูล DATA จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting stag) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น (start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งมีจำนวน 8 บิตจากนั้นตามด้วยบิตพาริตี้ (parity bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูลบิตสุดท้ายที่จะส่งออกไปคือ บิตปิดท้ายหรือบิตหยุด (stop bit) โดยจะเป็นการทำให้ขา DATA มีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิตก 1.5 บิต หรือ 2. บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่า ได้แก่ 100 150 300 600 1,200 2,400 4,800 9,600 และ 19,200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์เนื่องจากอัตราบอดคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมติว่าข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และปิดท้าย 1 บิต ความยาวของข้อมูล 1 ไบต์จะมีความยาวเท่ากับ 10บิตถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9,600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd) แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์รวมบิตพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีท่าเท่ากับ 99H หรือ1001 1001B จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัว ซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของบิตพาริตี้จะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดพาริตี้เป็นคี่ ค่าของบิตพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART (Universal Asymchronous Receiver Transmitter : เป็นอุปกรณ์ที่ใช้ในการรับส่งข้อมูลอนุกรม) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพพาริรี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริรี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก 1 ทั้งหมดรวมทั้งบิตจะทำการตรวจสอบค่าพิริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมา ให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตที่

ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้ จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

4.2 รีจิสเตอร์ที่เกี่ยวข้องกับการทานของพอร์ตอนุกรมใน MCS-51

ในการทำงานของวงจรพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51มีรีจิสเตอร์ที่ต้องเกี่ยวข้องอยู่ 2 ตัว ดังมีรายละเอียดต่อไปนี้

4.2.1 รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial data buffer register)

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของริจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต การแบ่งเป็น 2 ส่วน คือ ริจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และริจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P 3.1 ในกรณีที่มีการอ่านข้อมูลจากริจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลช

4.2.2 ริจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial port Control Register)

เป็นริจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 98 H ในพื้นที่ของรีจิสเตอร์ SFRสามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต1	บิต 0
SM0	SM1	SM2	REN	TBS	RBS	TI	RI

SMO –SM1 (Serial port mode bit0-1) : ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายใน ไมโครคอนโทรลเลอร์ MCS-51

SM2 : ใช้ในการอีนาเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ (multiprocessor) ในการทำงานของโหมด 2 และ3 ของพอร์ตอนุกรมในไมโครคอนโทลเลอร์ MCS-51 ในโหมด 2 และ3 ถ้าบิตนี้เป็น “1” บิต R1 จะไม่มีแอกตีฟถ้าบิตที่ 9 ที่รับเข้ามาเป็น “0” (ข้อมูลบิตที่ 9 เก็บไว้ที่บิต RB8)

ในการทำงานโหมด 1 ถ้าบิตนี้เซต บิต R1 จะไม่แกตีฟถ้ายังไม่ได้รับบิตหยุด ส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน

REN (Enable serial reception) : **ใช้ในการอีนาเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้มีการรับข้อมูลต้องเซตบิตนี้ให้เป็น** "1"

TBS : ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไปในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 เซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

RB 8 : ใช้สำหรับรับข้อมูลบิตที่ 9 ที่เข้ามาในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แต่ถ้าพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น"0" ข้อมูลที่บิต RB8 คือข้อมูลของบิตหยุด (Stop bit) สำหรับในการทำงานโหมด 0 บิตนี้จะไม่ใช้งาน บิต RB 8 นี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์เท่านั้น

T1 (Transmit Interrupt flage): ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการส่งข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงนของโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟแวร์เท่านั้น

4.3 โหมดการทำงานของพอร์ตอนุกรมใน MCS-51

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลชสามารถเลือกโหมดการทำงานได้ถึง 4 โหมด คือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์
2. โหมด 1 เป็นการทำงานให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้
3. โหมด 2 เป็นการทำงานให้เป็น UART ขนาด 9 บิตโดยมีอัตราบอดคงที่
4. โหมด 3 เป็นการทำงานให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดการทำงานของวงจรพอร์ตในไมโครคอนโทรลเลอร์ MCS-51 กระทำได้โดยการกำหนดข้อมูลให้แก่บิต SM0และ SM1 ในริจิสเตอร์ SCON

4.3.1 การทำงานในโหมด 0 ของวงจรพอร์ตอนุกรม

ข้อมูลอนุกรมเวลาจะผ่านเข้าและออกทางขา RxD ส่วนขาTxD ทำหน้าที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล(shift clock) ในโหมดนี้มีจำนวนข้อมูล 8 บิต โดยทำการรับและส่งข้อมูลในบิต LSB ก่อน อัตราในการับส่งข้อมูลหรืออัตราบอดถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่สัญญาณนาฬิกา

เริ่มต้นการส่งข้อมูลด้วยการเขียนข้อมูลที่ต้องการส่งมายังรีจิสเตอร์ SBUF สัญญาณเขียนข้อมูลSUBF แอกตีฟเป็น "1" ที่สเตต 6 เฟส 2 (S6P2) ของแมชีน ไซเกิล ส่งมายังวงจรควบคุมกานส่ง (TX control) ทำให้วงจรควบคุมการส่งเริ่มต้นการส่งข้อมูลสัญญาณ SEND จะแอกตีฟเป็น "1" ตลอดกระบวนการส่งข้อมูล

ข้อมูลจากรีจิสเตอร์ SUBF จะถูกเลื่อนออกจากขาที่ P3.0 หรือขา RxD ครั้งละบิต ตามจังหวะของสัญญาณนาฬิกาที่ส่งออกมาทางขา P3.1 หรือ TxD โดยสัญญาณนาฬิการของการเลื่อนข้อมูลจะมีขอบขาที่สัญญาณที่สเตต 3 เฟส และมีขอบขาขึ้นของสัญญาณที่สเตต 6 เฟส 1 ของแต่ละแมชชีนไซเกิลในกระบวนการส่งข้อมูล จนกระทั่งเมื่อส่งข้อมูลครบ 8 บิต แล้ว แล้วบิต T1 ในรีจิสเตอร์ SCON จะเกิดการเซตเป็นการแจ้งใกห้ทราบว่าส่งข้อมูลครบแล้ว หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการอีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ SEND จะกลายเป็นฯ "0" จนกว่ากระบวนการรับข้อมูลใหม่

ในกระบวนการรับข้อมูล เริ่มต้นด้วยการเซต REN ให้เป็น "1" และเคลียร์บิต R1 ในริจิสเตอร์ SCON ก่อน ที่สเตต 6 เฟส 2 ของแมชชีนไซเกิลถัดไป วงจรควบคุมการับ (RX control) จะทำการเขียนข้อมูล 11111110 ไปยังชิฟต์รีจิสเตอร์ สำหรับรับข้อมูลและทำการแอกตีฟสัญญาณ RECEIVE ให้เป็น "1" ในสัญญาณนาฬิการลูกถัดไป

เมื่อสัญญาณRECEIVEแอกตีฟ ก็จะเกิดการส่งสัญญาณนาฬิกาของการเลื่อนข้อมูล (Shift clock) ขึ้นผ่านทางขา P3.1 หรือ TxD เพื่อทำการกำหนดจังหวะการรับข้อมูลครั้งละบิต โดยสัญญาณนาฬิกาจะเกิดขึ้นในช่วงสเตต 3 เฟส 1 ถึง สเตต 6 เฟส 21 ของแต่ละแมชีนไซเกิล การรับข้อมูลเข้ามาทางขา P3.0หรือ RxD จะเกิดขึ้นที่สเตต 5 เฟส 2 ในแมชีนไซเกิลเดียวกับสัญญาณนาฬิกาของการเลื่อนข้อมูล จนกระทั่งรับข้อมูลครบทั้ง 8 บิต R1 จะได้รับการเซตเพื่อแจ้งการเสร็จสิ้นกระบวนกการรับข้อมูลหากการอินเอตร์รัปต์จากพอร์อนุกรมได้รับการอีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นการรับข้อมูล สัญญาณ RECEIVE จะกลายเป็นฯ "0" จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

การทำงานในโหมดนี้ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 จะใช้ประโยชน์ในการเชื่อมต่อกับไอซีริจิสเตอร์ภายนอกเพื่อทำการขยายจำนวนพอร์ตอินพุตหรือเอาท์

พุต แต่ไม่เป็นที่นิยมใช้งานมากนัก เนื่องจากไมโครคอนโทรลเลอร์ MCS-51 เอง มีพอร์ตอยู่ค่อนข้างมาก และสามารถติดต่อกับเหล่านั้นได้ง่ายและเร็วกว่ามาก

4.3.2 การทำงานในโหมด 1 ของวงจรพอร์ตอนุกรม

ในโหมดนี้จะใช้ในการรับส่งข้อมูลรวม 10 บิตโดยส่งข้อมูลออกทางขา P3.1 หรือ TxD และรับข้อมูลเข้าทางขา P3.0 หรือ RxD ข้อมูลทั้ง 10 บิตประกอบด้วย บิตเริ่มต้น (มีค่าเป็น "0") 1 บิตบิตข้อมูล 8 บิต โดยรับหรือส่งข้อมูลในบิต LSB ก่อน และบิตหยุดหรือปิดท้าย (มีค่าเป็น"1") ในการรับข้อมูล บิตหยุดจะถูกเก็บไว้ในบิต RB8 ในริจิสเตอร์ SCON อัตราบอดในโหมดนี้ได้รับการกำหนดโดยอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 ในAT89C51 ส่วนนไมโครคอนโทรลเลอร์เบอร์ AT89C52 และในอนุกรม AT89Sxx สามารถเลือกใช้อัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 หรือไทเมอร์ 2 ในการกำหนดเป็นอัตราบอดได้

กระบวนการส่งข้อมูลเริ่มต้นด้วยการแอกตีฟสัญญาณเขียนข้อมูลมายังรีจิสเอตร์ SBUF ส่งมายังวงจรควบคุมการส่ง (TX control) จากนั้นวงจรควบคุมการส่งจะทำการแอกตีฟสัญญาณ SEND ที่สเตต 1 เฟส 1 ของแมชชีนไซเกิลโดยสัญญาณ SEND จะเป็น "0" ตลอดกระบวนการส่งข้อมูลเมื่อสัญญาณ SEND แอกตีฟ จะทำการส่งบิตเร่อม้นก่อนเป็นบิตแรก โดยมีคาบเวลาของบิตเริ่มต้นเท่ากับ 1 แมชชีน"วเกิล จากนั้นตามด้วยการส่งบิตข้อมูล 8 บิตเรียงลำดับจากบิต LSB โดยข้อมูลที่ทำการส่งนี้จะถูกเรียกออกมาจากรีจิสเตอร์บัฟเฟอร์สำหรับการส่งข้อมูล ในทุกๆบิตข้อมูลที่ทำการส่งออกไปจะเกิดสัญญาณพัลส์ SHIFT ขึ้นเพื่อให้เรียกข้อมูลในแต่ละบิตจากรีจิสเตอร์บัฟเฟอร์ การกำหนดจังหวะการส่งข้อมูลจะใช้สัญญาณนาฬิกาการส่ง (TX clock)เป็นตัวกำหนดโดยสัญญาณนาฬิกานี้ได้มาจากการหารสัญญาณ TCLK จากไทเมอร์ : ด้วย 16 หลังจากการส่งบิตข้อมูลก็จะทำการส่งบิตหยุดหรือบิตปิดท้ายขนาด 1 บิต ดังนั้นการส่งข้อมูลจะใช้สัญญาณนาฬิกาทั้งหมด 10 ลูก เมื่อทำการ ส่งข้อมูลได้รับการอีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ

หลังจากที่ทำการบิรการอินเตอร์รัปต์หรือส่งข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต T1 ก่อนเป็นอันดับแรกเพื่อให้สามารถเริ่มต้นกระวนการรับส่งข้อูลทางพอร์ตอนุกรมดำเนินต่อไปได้

ด้านการรับข้อมูล จะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจาก "1" เป็น "0" ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ 1/16 เท่าของอัตราบอด เมื่อตรวจจับพบ ไทเมอร์/เคาน์เตอร์ที่ใช้ในการกำหนดอัตราบอดรีเซต และทำการเขียนข้อมูล1FFH ไปยังชิฟต์ รีจิสเตอร์อินพุต

ข้อมูลจะเริ่มเดินทางเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ผ่านทางขา RxD ในการตีความว่าบิตที่เข้ามาเป็น "0" หรือ "1" จะให้ผลการสุ่มค่อนข้างมาก ดดยบิตของข้อมูลที่เข้ามาได้รับการแบ่งออกเป็น 16 สเตต การสุ่มข้อมูลจะทำการสุ่มสเตตที่ 7 ,8 และ9 หาก 2 ใน 3 ของการ

สุ่มพบว่าข้อมูลเป็นลอจิกใด จะตีความข้อมูลในบิตนั้นเป็นตามเสียงข้างมาก ยกตัวอย่างสุ่มพบลอจิก "1" 2 ใน 3 ครั้ง จะตีความว่าบิตของข้อมูลที่ได้รับนั้นเป็น "1"

ลำดับของการรับข้อมูลมีลักษณะเดียวกับการส่งข้อมูล คือ เริ่มด้วยบิตเริ่มต้นก่อนตามด้วยบิตข้อมูล และบิตปิดท้าย ในทุกๆการรับข้อมูลได้ 1 บิต มีพัลส์ SHIFT เกิดขึ้น เพื่อทำการเลื่อนข้อมูลเข้าสู่รีจิสเตอร์บัฟเฟอร์การรับข้อมูล การกำหนดจังหวะการรับข้อมูล ใช้สัญญาณนาฬิกาการรับข้อมูล(RX clock) หลังจากสัญญาณนาฬิกาลูกสุดท้าย อันหมายถึงสามารถรับข้อมูลได้ครบแล้ว วงจรควบคุมการรับข้อมูลจะทำการส่งข้อูลจกากรีจิสเตอร์บัฟเฟอร์ไปยังรีจิสเตอร์ SBUF และบิต RB 8 ในรีจิสเตอร์SCON โดยข้อมูลในบิต RB8 ก็คือข้อมูลของบิตหยุดนั่นเองพร้อมกันนั้นยังทำการเซตบิต R1 ในรีจิสเตอร์ SCON ด้วย หากการอินเตอร์รัปต์จากพอร์ตอนุกรมมาได้รับการอีนาเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ

หลังจากที่ทำการบริการอินเตอร์รัปต์หรือรับข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต R1 ก่อน เป็นอันดับแรก เพื่อให้สามารถเริ่มต้นกระบวนการรับส่งข้อมูลทางพอร์ตอนุกรม มาดำเนินต่อไปได้

การทำงานในโหมดนี้ได้รับความนิยมสูงสุด เนื่องจากมีกระบวนการที่ซับซ้อนและสามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

4.3.3 การทำงานในโหมด2 และ 3 ของวงจรพอร์ตอนุกรม

ในทั้งสองโหมดนี้จะใช้รูปแบบการรับข้อมูลรวม 11 บิต ประกอบด้วยบิตเริ่มต้นมีค่าเป็น "0" จำนวน 1 บิต บิตข้อมูล 8 บิต โดยทำการรับและส่งบิต LSB ก่อน บิตข้อมูล บิตที่ 9 และบิตปิดท้ายมีค่าเป็น "1" จำนวน 1 บิต ในการส่งข้อมูล ข้อมูลบิตที่ 9 จะได้รับ การเก็บไว้ในบิต TB 8 ในรีจิสเตอร์ SCON และในการรับข้อมูล ข้อมูลบิตที่ 9 จะนำไป เก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำหรับอัตราบอดในโหมด 2 จะคงที่โดยเลือกได้ 2 ค่าคือ 1/32 หรือ 1/64 ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 อัตราบอดสามารถปรับได้เหมือนกับในโหมด 1

การทำงานโดยรวมจะคล้ายกับการทำงานในโหมด 1 ส่วน ที่แตกต่างกันคือจำนวนบิตของข้อมูลที่ในโหมด 2 แบะ 3 จะมีเพิ่มมาอีก 1 ยิค โดยส่วนใหญ่จะเป็นบิตตรวจสอบพาริตี้

4.4 อัตราการบอดของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51

4.4.1โหมด 0

อัตราบอดของโหมดนี้มีค่าคงที่ โดยสามารถคำนวณได้จากสูตร

อัตราบอดในโหมด 0 = ความถี่ของสัญญานาฬิกา / 12หน่วยเป็นบิตต่อวินาที

4.4.2 โหมด 1 และ 3

เนื่องจากทั้งสองโหมดนี้สามารถเลือกแหล่งกำเนิดอัตราบอดได้ 2 แหล่ง คือ จากอัตราโอเวอร์โฟลวของไทเมอร์ 1 และ 2 สำหรับอัตราบอดเมื่อใช้การโอเวอร์โฟลวของไทเมอร์ 1 จะต้องใช้ค่าองบิต SMOD ในรีจิสเตอร์ PCON (จะกล่าวถึงรายละเอียดของรีจิสเตอร์ตัวนี้ในบทที่ว่าด้วยการทำงานในโหมดประหยัดพลังงาน) มาพิจารณาประกอบด้วย โดยสามารถคำนวณหาค่าอัตราบอดได้จาก

$$\text{อัตราบอด} = (12^{\text{ค่าในรีจิสเตอร์ SMOD}} /32) \text{ X อัตราโอเวอร์โฟลวของไทเมอร์ 1}$$

ตารางที่ 4-1 การเลือกอัตราบอดของวงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

อัตราบอด (บิตต่อวินาที : bps)	ความถี่ สัญญาณนาฬิกา	SMOD	ไทเมอร์1		
			C/T	โหมด	ค่ารีโหลด
โหมด 0: สูงสุด 1 MHz	12 MHz	×	×	×	×
โหมด 2 : สูงสุด 375K	12 MHz	1	×	×	×
โหมด 1,3:62.5 K	12 MHz	1	0	2	FFH
19.2 K (19,200)	11.0592 MHz	1	0	2	FDH
9.6 K (9,600)	11.0592 MHz	0	0	2	FDH
4.8K (4,800)	11.0592 MHz	0	0	2	FAH
2.4K (2,400)	11.0592 MHz	0	0	2	F4H
1.2K(1,200)	11.0592 MHz	0	0	2	E8H
137.5	11.0592 MHz	0	0	2	1DH
110	6Hz	0	0	2	72H
110	12Hz	0	0	1	FEE8H

ถ้าหากในไทเมอร์ 1 ไม่ได้อีนาเบิลการอินเตอร์รัปต์ไว้ สามารถคำนวณหาค่าอัตราบอดได้จาก

$$\text{อัตราบอด} = (2^{\text{ค่าในรีจิสเตอร์ SMOD}} /32) \text{ X ความถี่สัญญาณนาฬิกา}/\{12 \text{ x}[256-(TH1)]\}$$

ในตารางที่ 4-1 แสดงการกำหนดอัตราบอดโดยใช้ไทเมอร์ 1

ในกรณีที่ใช้ไทเมอร์ 2 ในการกำหนดอัตราบอด โดยกำหนดให้ไทเมอร์ 2 ทำงานในโหมดกำเนิดอัตราบอด (baud rate generator) สามารถคำนวฯหาอัตราบอดได้จาก

อัตราบอด = อัตราโอเวอร์โฟลวของไทเมอร์ 2/16 หน่วยเป็น บิตต่อวินาที

ถ้าหากกำหนดให้ไทเมอร์ 2 ทำงานในโหมดไทเมอร์หรือเคาน์เตอร์ตามปกติ สามารถคำนวฯหาค่าอัตราบอดได้จาก

อัตราบอด = ความถี่ของสัญญาณนาฬิกา / (32 x (65536-(RCAP2H , RCAP2L))

โดยที่ (RCAP2H , RCAP2L) เป็นค่าของรีจิสเตอร์ RCAP2H และ RCAP2L มีขนาด 16 บิตไม่คิดเครื่องหมาย

4.4.3 โหมด 2

ในโหมดนี้การกำหนดอัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON ถ้า SMOD เป็น “0” อัตราบอดจะเท่ากับ 1/64 ของความถี่สัญญาณนาฬิการ ในกรณีที่ SMOD เป็น “1” อัตราบอดจะเท่ากับ 1/32 ของความถี่สัญญาณนาฬิกา สามารถแสดงเป็นสูตรคำนวณทางคณิตศาสตร์ได้ดังนี้

$$\text{อัตราบอด} = (2^{\text{ค่าในรีจิสเตอร์ SMOD}} /64) \text{ X ความถี่สัญญาณนาฬิกา}$$

4.5 การกำหนดค่าของไทเมอร์เพื่อเลือกอัตราบอด

ในการใช้งานพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สิ่งที่ต้องให้ความสนใจมากที่สุดประการหนึ่งคืออัตราการถ่ายทอดหรืออัตราบอด ซึ่งการกำหนดอัตราบอดนั้นจะขึ้นอยู่กับค่าความถี่ของสัญญาณนาฬิกาเป็นหลัก สำหรับโหมดการทำงานของพอร์ตอนุกรมที่สามารถเลือกอัตราบอดได้อย่างอิสระคือในโหมด 1 และ 3 โดยกำหนดได้จากอัตราการเกิดโอเวอร์โฟลวของไท

เมอร์ 1 ถ้าหากไทเมอร์1มีการเกิดโอเวอร์โฟลวในอัตราที่สูงมาก สามารถถ่ายทอดข้อมูลได้อย่างรวดเร็ว ดังนั้นการกำหนดค่าและโหมดการทำงานของไทเมอร์ 1 จึงเป็นสิ่งที่มีผลต่อการเปลี่ยนแปลงของอัตราบอดด้วย

ในการใช้ไทเมอร์ 1 เพื่อกำหนดอัตราบอดในโหมด 1 และ 3 ของพอร์ตอนุกรมจะต้องกำหนดให้ไทเมอร์ 1 ทำงานโหมด 2 หรือ โหมด 8 บิตแบบตั้งค่าการนับอัตโนมัติ และ การกำหนดค่ารีโหลดให้แก่รีจิสเตอร์ TH1 จึงเป็นตัวแปรหลักที่ใช้ในการกำหนดอัตราบอดให้แก่พอร์ตอนุกรมไมโครคอนโทรลเลอร์ MCS-51

เริ่มต้นด้วยการเคลียร์บิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON ให้เป็น “0” ค่าของการรีโหลดให้แก่ TH1 สามารถคำนวณได้จาก

TH1 = 256-((ค่าความถี่ของคริสตอล/384)/อัตราบอด)

แต่ถ้าบิตSMODเกิดการรีเซต จะเป็นการเกิดอีนาเบิลทวีคูณของอัตราบอด ดังนั้นการกำหนดค่าให้แก่ TH1 จึงต้องคำนวณจาก

TH1 = 256-((ค่าความถี่ของคริสตอล/192)/อัตราบอด)

ยกตัวอย่างถ้าหากในไมโครคอนโทรลเลอร์ At89C4051 ใช้คริสตอล 11.0592MHz ต้องการกำหนดอัตราบอดของพอร์ตอนุกรมของไมโครคอนโทรเลอร์ไว้ที่ 19,200บิต ต่อวินาที ในกรณีที่ไม่อีนาเบิบการทีวีคูณของอัตราบอดค่ารีโหลดของไมโครคอนโทรเลอร์ จะเท่ากับ

TH1 = 256-((ค่าความถี่ของคริสตอล/384)/อัตราบอด)
= 256-((11059200/238)/19200)
= 256-(28800/19200)
=256-1.5
= 254.5

เนื่องจากผลลัพธ์ที่ได้เป็นค่าที่ไม่ใช่จำนวนเต็ม ถ้าหากกำหนดค่าของ TH1 เป็น 254 เมื่อทำการแทนค่าเพื่อคำนวณหาค่าอัตราบอดจะได้อัตราบอดเท่ากับ 14,400 บิตต่อวินาที และถ้าหากกำหนดค่าของ TH1 เป็น 255 อัตราบอดจะมีค่าเท่ากับ 28,800 บิตต่อวินาที ดังนั้น จะเห็นได้ค่าของ TH1 ที่ไม่เป็นจำนวนเต็มจะไม่สามารถทำให้เกิดอัตราบอดตาที่ต้องการได้

ทางแก้ไข คือ ทำการอีนาเบิลทวีคูณอัตราบอด โดยการเซตบิตSMOD ในรีจิสเตอร์ PCON ให้เป็น “1” จากนั้นทำการแทนค่าลงในสมการหาค่า TH1 เมื่อมีการเซตบิต SMOD ได้ผลดังนี้

TH1 = 256-((ค่าความถี่ของคริสตอล/192)/อัตราบอด)

= 256-((11059200/192)/19200)

= 256-(51600/19200)

=256-3

= 253

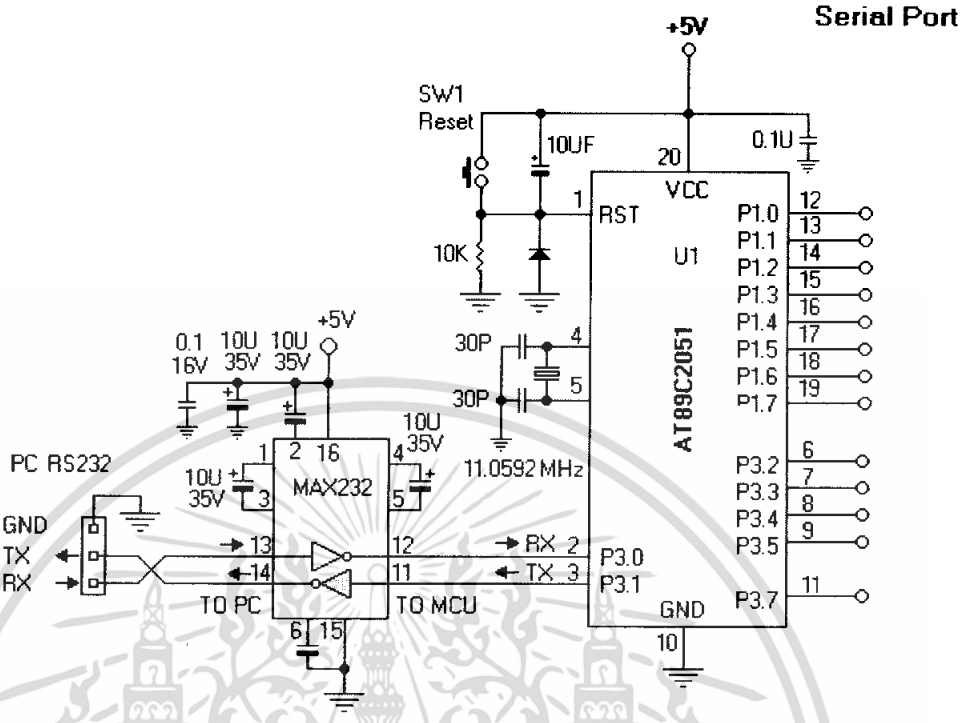
นำค่าของ TH1 ที่ได้ทำการแทนค่าหาอัตราบอดจะได้ท่ากับ 19,200 บิตต่อวินาที สามารถสรุปขั้นตอนในการเลือกอัตราบอดโดยการกำหนดค่าของไทเมอร์ 1 ได้ดังนี้

1. กำหนดให้พอร์ตอนุกรมของไมโครคอนโทรเลอร์ MCS-51 ทำงานในโหมด 1 หรือ3
2. กำหนดให้ไทเมอร์ 1 ทำงานในโหมด 2 หรือ โหมด8 บิตตั้งค่าอัตโนมัติ
3. กำหนดข้อมูลให้ TH1 เท่ากับ 253 เพื่อให้สามารถกำเนิดอัตราบอดได้ 19,200 บิตต่อวินาทีตามที่ต้องการ
4. ทำการเซตบิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON เพื่ออีนาเบิลการทวีคูณอัตราบอด

4.6 การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมพื้นฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ ±3 ถึง ±12 V ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้น จึง ไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ ในการแปลงระดับสัญญาณ

ไอซีที่ทำหน้าที่ในการแปลงระดับสัญญาณนี้ต้องทำการแปลงข้อมูลส่งไปไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่าวมีอยู่ด้วยกันหลายเบอร์จากหลายผู้ผลิต อาทิ MAX232 จากMAXIM หรือ ICI.232 จากHARRIS เป็นต้น

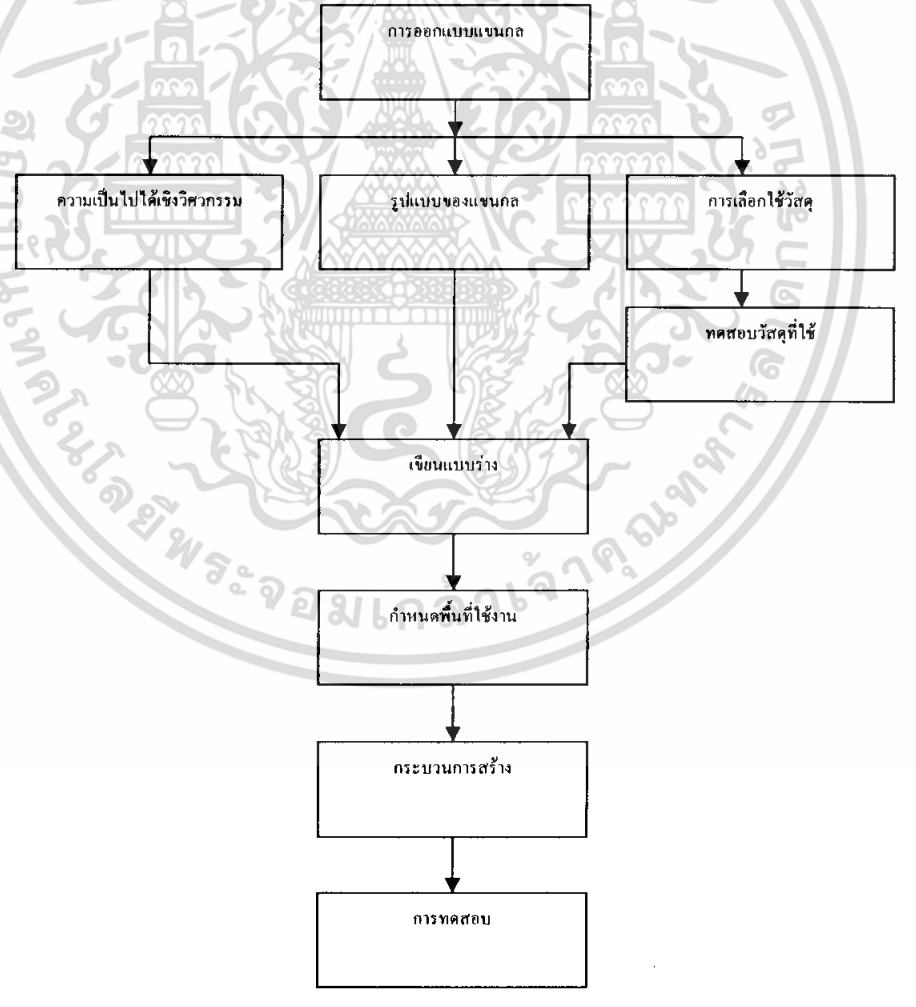


รูป 4-2 วงจรการเชื่อมต่อพอร์ตอนุกรม

บทที่ 5
การออกแบบโครงสร้างแขนกล

5.1 Project Flowchart

ลักษณะการออกแบบโครงสร้างแขนกล สิ่งที่ต้องคำนึงถึง คือ ลักษณะของงานที่จะนำไปใช้ความแข็งแรงทนทาน ความสวยงาม กลไกการเคลื่อนที่ของแขนกล และวัสดุที่นำมาใช้ โดยหลักการออกแบบ จะนำลักษณะทางกายภาพและลักษณะการเคลื่อนที่ของแขนมนุษย์มาเป็นแบบ



5.1.1 ความเป็นไปได้เชิงวิศวกรรม ปัจจัยในการพิจารณา

1. สามารถได้จริงหรือไม่
2. ใช้ร่วมกับอุปกรณ์อะไร
3. ความแข็งแรงทนทาน
4. ความปลอดภัยในการใช้งาน
5. ข้อดีข้อเสียที่เกิดขึ้น

5.1.2 รูปร่างของแขนกล ปัจจัยในการออกแบบ

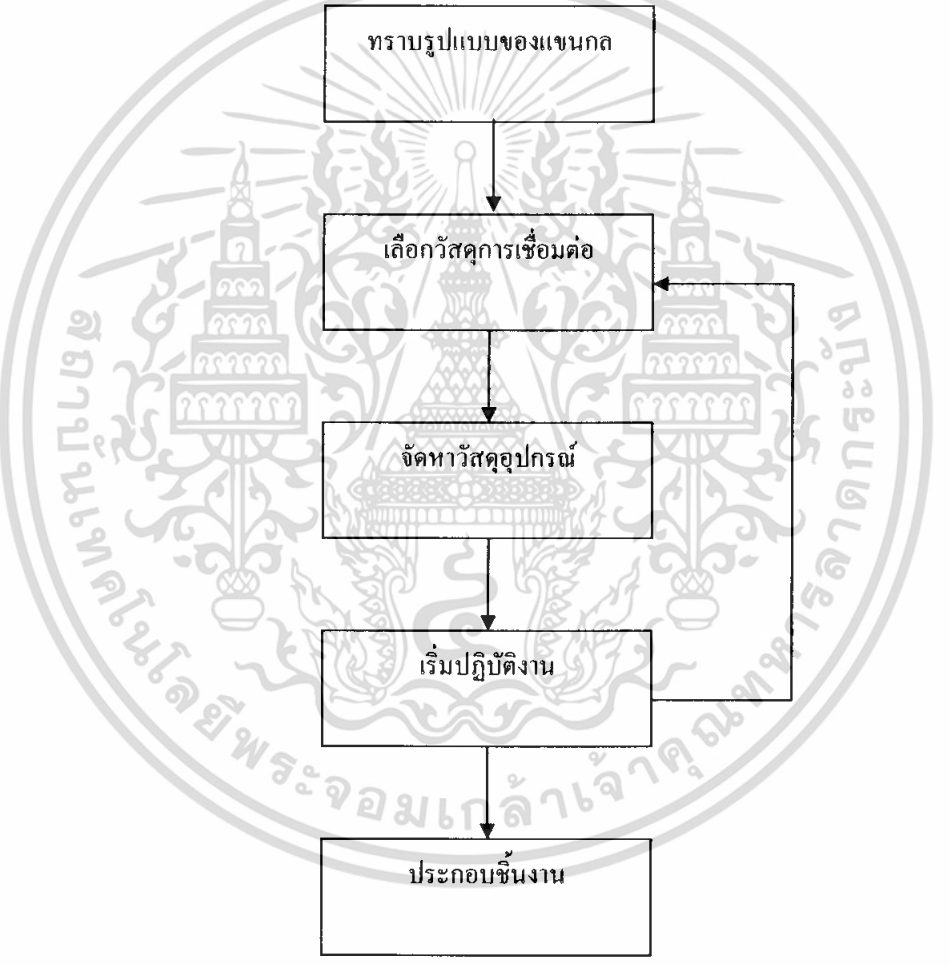
1. สภาพแวดล้อมการทำงานของแขนกล
2. ลักษณะของงาน
3. ความสะดวกและง่ายในการใช้งาน
4. ความสวยงาม
5. กลไกการเคลื่อนที่ของแขนกล

5.1.3 การเลือกใช้วัสดุ ปัจจัยในการเลือกใช้

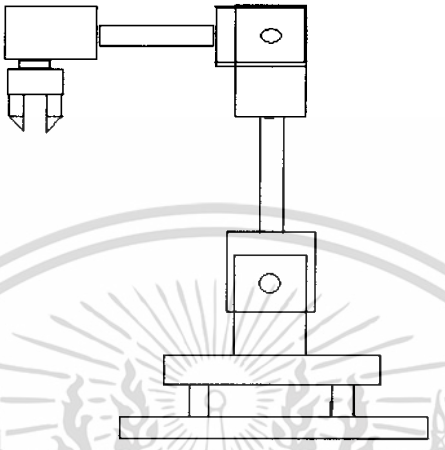
1. งบประมาณในการสร้าง
2. ความเหมาะสมกับงาน
3. การใช้วัสดุทดแทน

5.2 Hardware Flowchart

เมื่อได้รูปแบบของแขนกลที่แน่นอน จะทำการเลือกวัสดุที่ใช้ ทำการตรวจสอบความแข็งแรง และเริ่มสร้างแขนกลตามโครงร่างที่ได้ออกแบบไว้

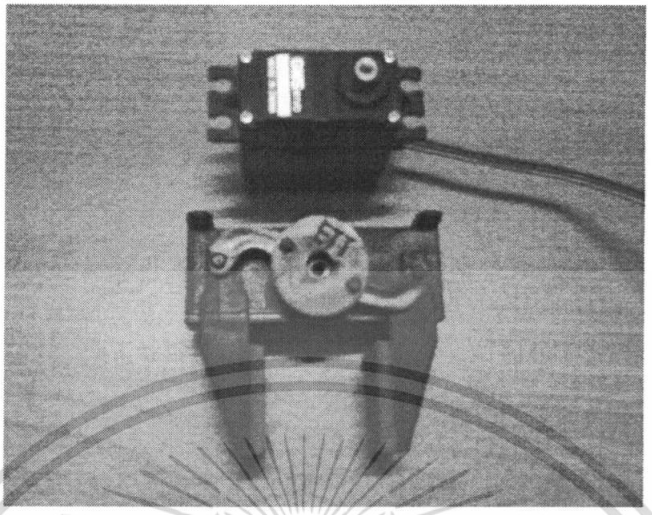


5.3 การสร้างและการประกอบชิ้นงาน



รูป 5-1 แบบร่างของแขนกล

การสร้างและการประกอบชิ้นส่วนนั้น เริ่มสร้างส่วนที่เป็นองค์ประกอบหลักๆ ก่อน และส่วนประกอบอื่นๆ ให้สร้างตามลำดับ ก่อน-หลัง เมื่อได้ตามแบบที่วางไว้แล้ว จึงเป็นการนำชิ้นส่วนทั้งหมดที่เตรียมไว้มาประกอบเข้าด้วยกัน ตามแบบร่างที่ได้เขียนเอาไว้ตอนแรก ซึ่งชิ้นส่วนแต่ละชิ้นจะสัมพันธ์กัน แต่สิ่งที่ต้องคำนึงถึงก็คือ ต้องประกอบชิ้นส่วนนั้นให้แข็งแรง และชิ้นส่วนแต่ละชิ้นจะต้องสามารถเข้ากันพอดี เมื่อนำมาประกอบกันโดยชิ้นส่วนแต่ละชิ้นนั้นจะประกอบไปด้วยส่วนต่างๆ ดังรูปที่แสดงต่อไปนี้คือ

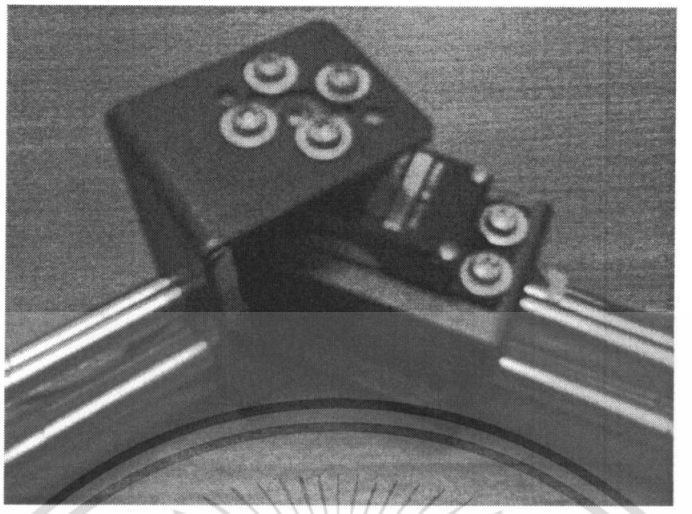


รูป5-2 ชุด GRIP จับสิ่งของ

ชุด GRIP จับสิ่งของ จะประกอบไปด้วย ตัว GRIP จับสิ่งของ และ sorvo motor และในส่วนนี้จะทำการต่อ sorvo motor เข้ากับ ตัว GRIP จับสิ่งของ ซึ่ง sorvo motor จะมีหน้าที่บังคับให้ตัวจับสิ่งของ จับหรือปล่อยสิ่งของ โดยตัว GRIP จับสิ่งของจะเลือกใช้เป็นพลาสติกที่มีความแข็งแรงทนทาน และมีน้ำหนักเบา ที่ตัวจับสิ่งของนั้นจะทำการติดฟองน้ำไว้เพื่อลดแรงบีบของตัว GRIP จับสิ่งของ เพื่อไม่ให้วัตถุที่ถูกจับเกิดความเสียหายจากแรงบีบ

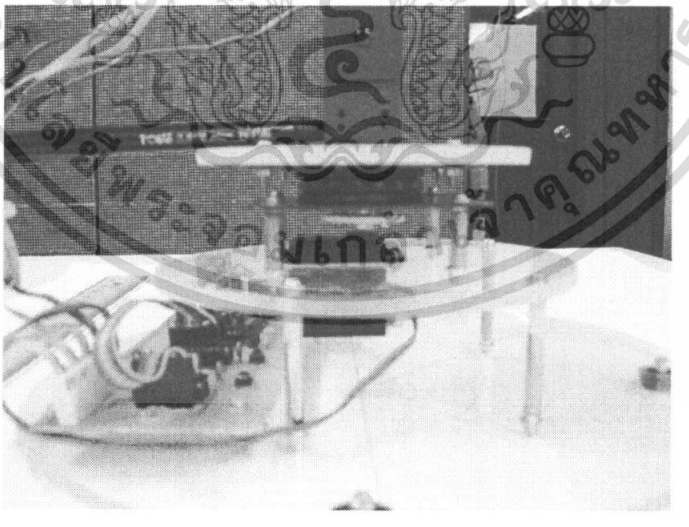
รูป5-3 ชุดแกนเหล็ก

รูปที่ 5-3 เป็นชุดแกนเหล็ก เป็นส่วนที่ใช้สำหรับเชื่อมต่อระหว่างข้อต่อทั้งสองของแขนกล ซึ่งในส่วนนี้จะใช้จะเป็นแกนเหล็กเพราะเป็นส่วนที่รับน้ำหนักของแขนกลในขณะที่แขนกลทำการยกสิ่งของ จึงต้องใช้วัสดุที่มีความแข็งแรงทนทานและรับน้ำหนักได้ดี โดยจะใช้แกนเหล็กทั้งหมดสี่แกนด้วยกัน



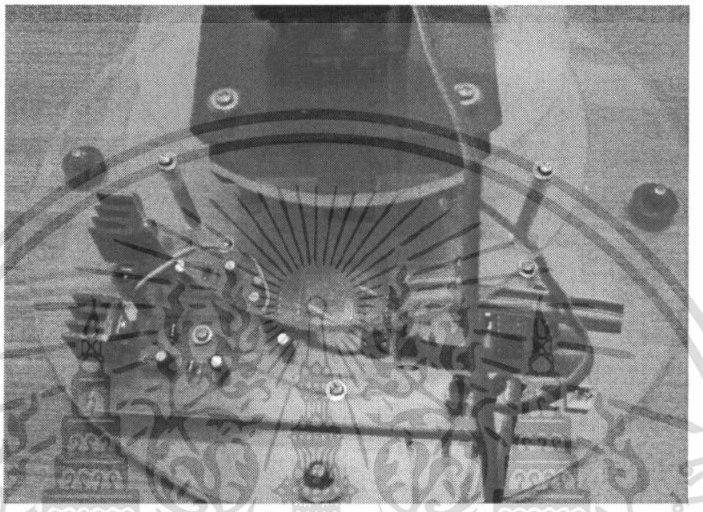
รูป5-4 ชุดข้อผับ

ชุดข้อพับ ซึ่งจะประกอบไปด้วย อลูมิเนียมที่เป็นข้อพับสองตัว และ sorvo motor ประกอบติดกันอยู่ ซึ่ง sorvo motor จะเป็นตัวบังคับให้อลูมิเนียมเคลื่อนที่ได้ตั้งแต่ 0 องศา ถึง 180 องศา ในส่วนนี้จะเป็นการหมุนแบบขึ้นกับลง และในส่วนของข้อพับเราจะเลือกใช้อลูมิเนียม เพราะมีน้ำหนักเบา ทำให้ลดแรง



รูป5-5 ชุดฐานหมุนลูกปืน

ชุดฐานหมุนลูกปืน โดยจะมีชุดลูกปืนยึดติดกับฐานของแขนกล 2 ส่วน คือ ฐานส่วนล่าง กับ ฐานส่วนบน โดย sorvo motor จะยึดติดกับฐานทั้งสองส่วน เพื่อใช้บังคับการหมุนของแขนกล ให้หมุนทางซ้ายและขวา โดยที่แกนหมุนของ sorvo motor จะยึดติดกับฐานส่วนบนของแขนกล และ ในส่วนของตัวถังของ sovor motor จะยึดติดกับฐานส่วนล่างของแขนกล



รูป 5-6 แผงวงจรควบคุม

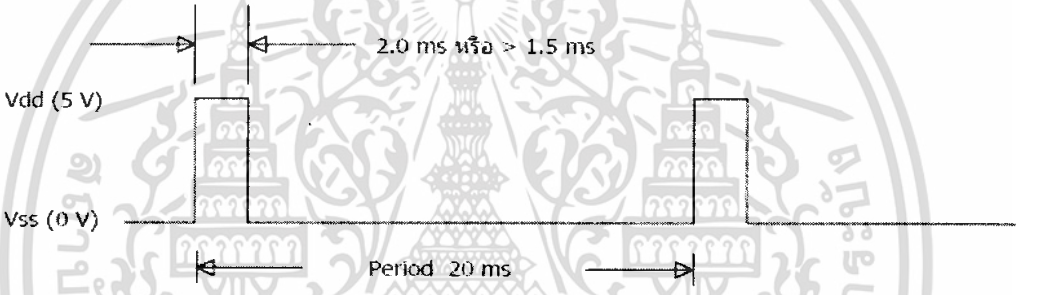
แผงวงจรควบคุม เป็นวงจรที่ใช้ในการควบคุมแขนกล โดยจะมีในส่วนของ IC MAX 232 ซึ่งจะใช้ในติดต่อสื่อสารผ่านพอร์ตอนุกรม กับ คอมพิวเตอร์ และจะมี Microcontrollor อีก1 ตัว เพื่อที่จะนำข้อมูล ส่งไปยังตัวถัดไป และ Microcontrollor ตัวที่สอง จะใช้เป็นตัว control ให้กับ sorvo motor ทั้ง 5 ตัวหมุนตามที่ต้องการ

บทที่ 6

การทดลองและผลการทดลอง

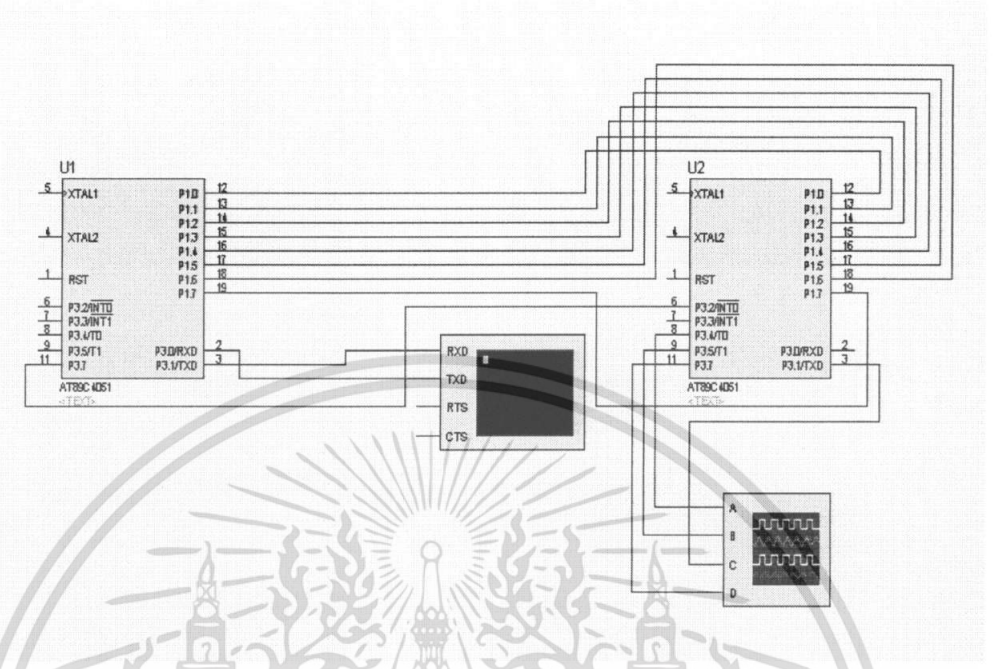
6.1 การจำลองการทำงานโดยใช้โปรแกรม Proteus 7 Professional และการทดลอง

การทดสอบการหมุนของเซอร์โวมอเตอร์ จากการป้อนพัลส์ด้วย ไมโครคอนโทรลเลอร์ MCS-51 (AT89C4051) โดยการเขียนโปรแกรมด้วย ภาษาซี เพื่อกำเนิดพัลส์ เพื่อที่จะขับเซอร์โวมอเตอร์ ให้หมุนไปยังตำแหน่งที่ต้องการ เช่น ถ้าต้องการให้เซอร์โวมอเตอร์ หมุนไปที่ 90 องศา จะได้คาบสัญญาณพัลส์ ซีกบวก เท่ากับ 2 mS โดยจะมีความกว้างของพัลส์ เท่ากับ 20 mS

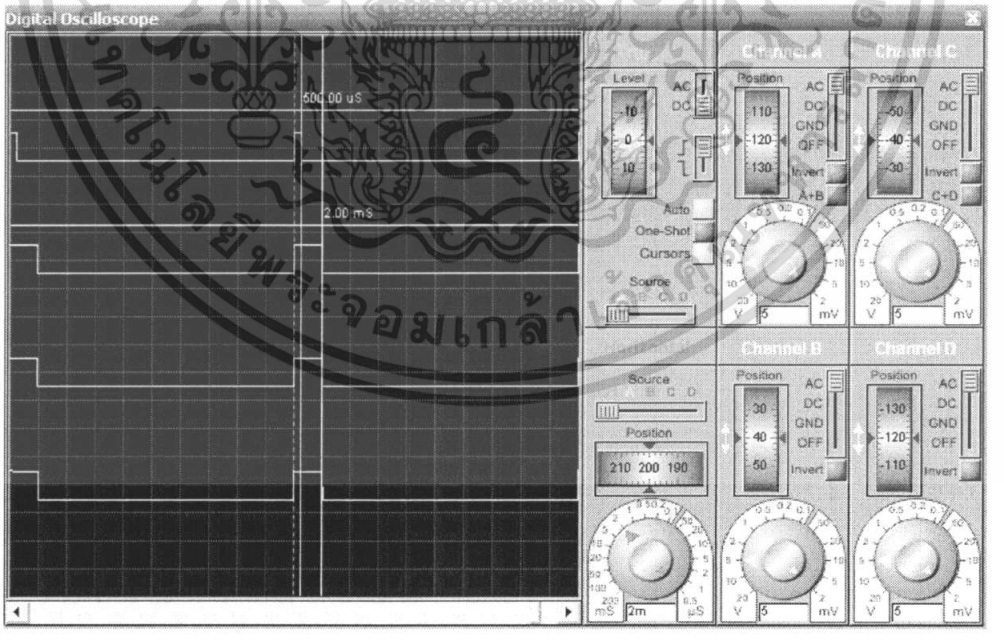


รูป6-1 ภาพสัญญาณพัลส์ที่ใช้ขับเซอร์โวมอเตอร์ ให้เคลื่อนที่ไป ที่มุม 90 องศา

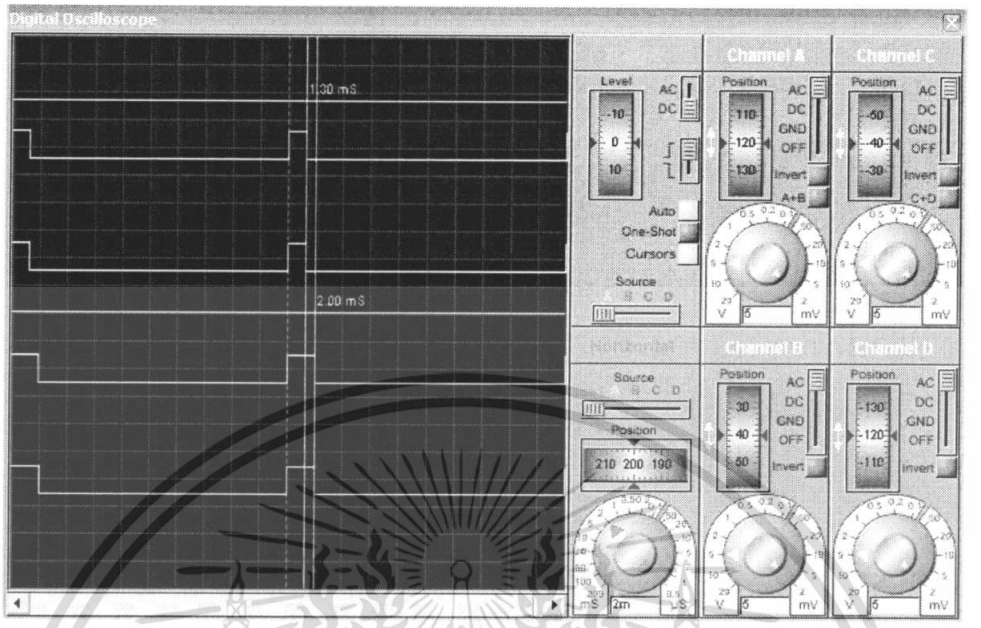
การทดลองโดยใช้โปรแกรม Proteus 7 Professional ในการ จำลองการทำงาน โดยการเขียนโปรแกรม ภาษาซี และทำการโปรแกรมลงใน ไมโครคอนโทรลเลอร์ (AT89C4051) จากนั้นทำการควบคุมตำแหน่ง ที่ต้องการ โดยการคีย์ตัวอักษร ซึ่งจะมีค่าตามในรหัส Ascii คีย์ลงใน Virtual Terminal จากนั้นสังเกตการเปลี่ยนแปลงของ เซอร์โวมอเตอร์ และรูปสัญญาณที่ออกมาจากไมโครคอนโทรลเลอร์



รูป 6-2 วงจรที่ใช้ทดลองในโปรแกรม Proteus 7 Professional

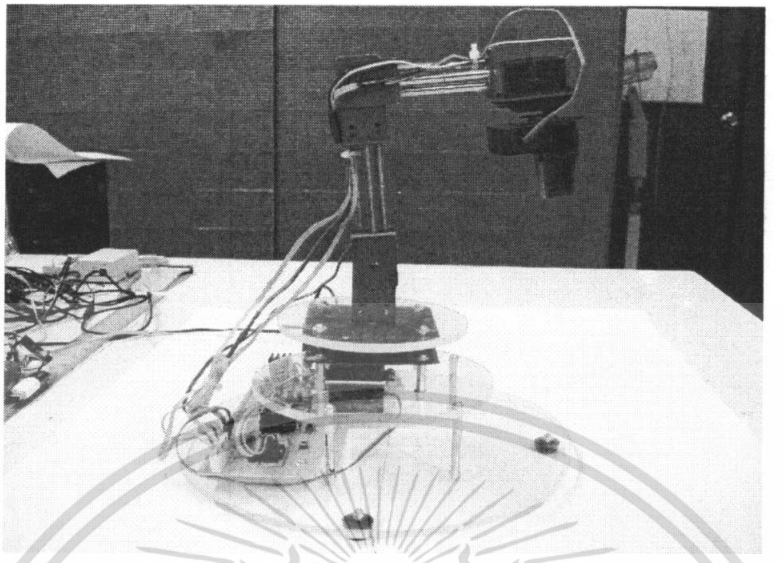


รูป6-3 สัญญาณที่จ่ายให้ SERVO MOTOR ต่ำสุด และ สูงสุด



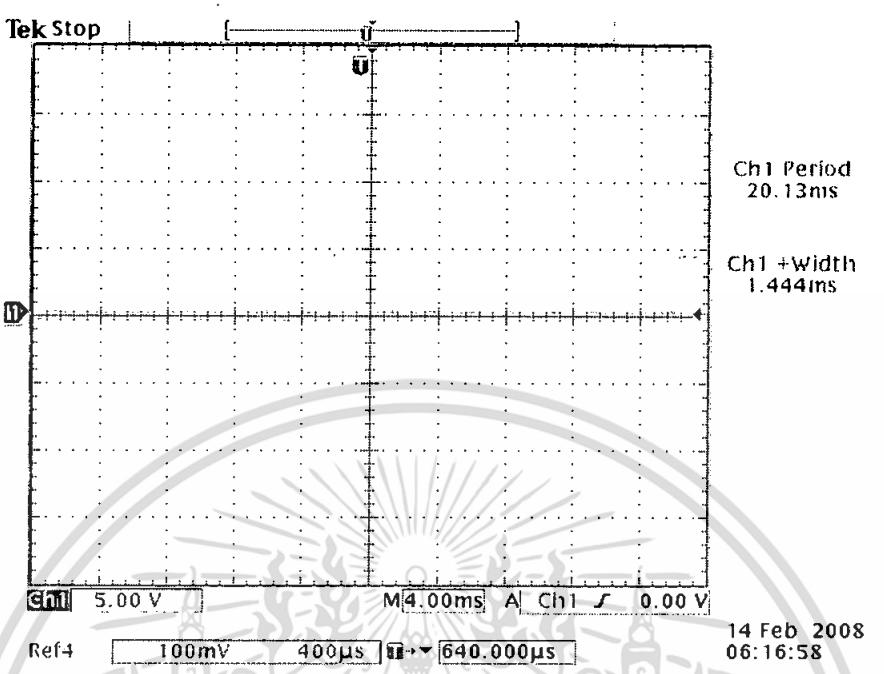
รูป 6-4 สัญญาณที่เริ่มการทำงานของแขนกล

จากรูปที่ 6-3 เป็นรูปสัญญาณที่เกิดจากการจำลองการทำงาน โดยใช้โปรแกรม Proteus 7 Professional ซึ่งรูปสัญญาณนี้เกิดจากการเขียนโปรแกรมด้วยภาษาซี โดยรูปนี้จะมี คาบสัญญาณพัลส์ ที่มากที่สุด เท่ากับ 2 mS มีความกว้างของพัลส์ เท่ากับ 20 mS เพื่อนำไปควบคุมเซอร์โวมอเตอร์ ให้หมุนไปที่ 90 องศา และมีคาบสัญญาณพัลส์ ที่น้อยที่สุดเท่ากับ 0.5 mS เพื่อนำไปควบคุมเซอร์โวมอเตอร์ ให้หมุนไปที่ -90 รูปที่ 6-4 เป็นรูปสัญญาณที่เกิดจากการจำลองการทำงาน โดยเป็นการเริ่มการทำงานซึ่งพัลส์ที่เกิดโดยไมโครคอนเลอร์ เป็นพัลส์ที่ทำให้ แขนกลอยู่ในสภาพพร้อมใช้งาน

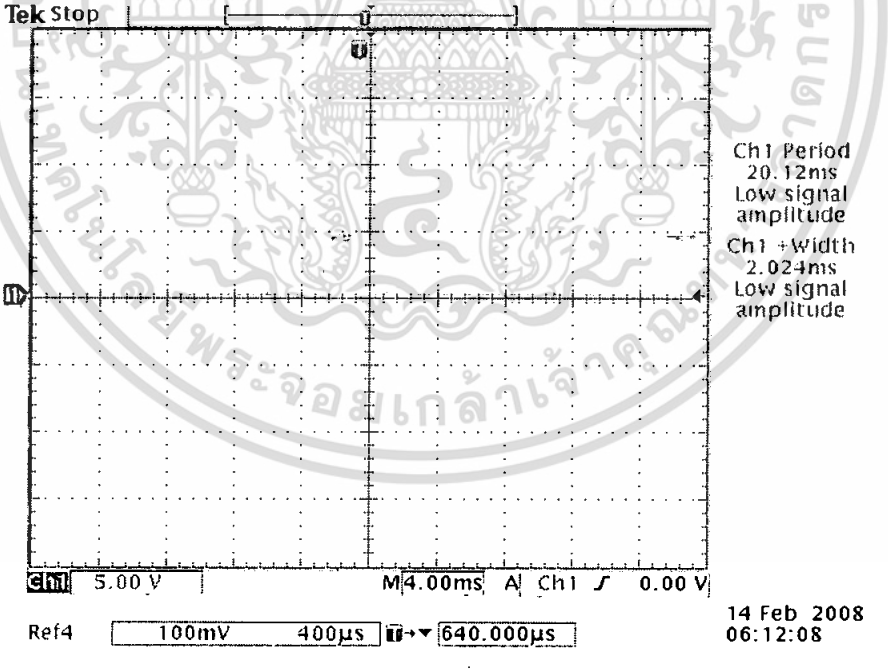


รูป 6-5 แขนกลอยู่ในสภาพพร้อมใช้งาน

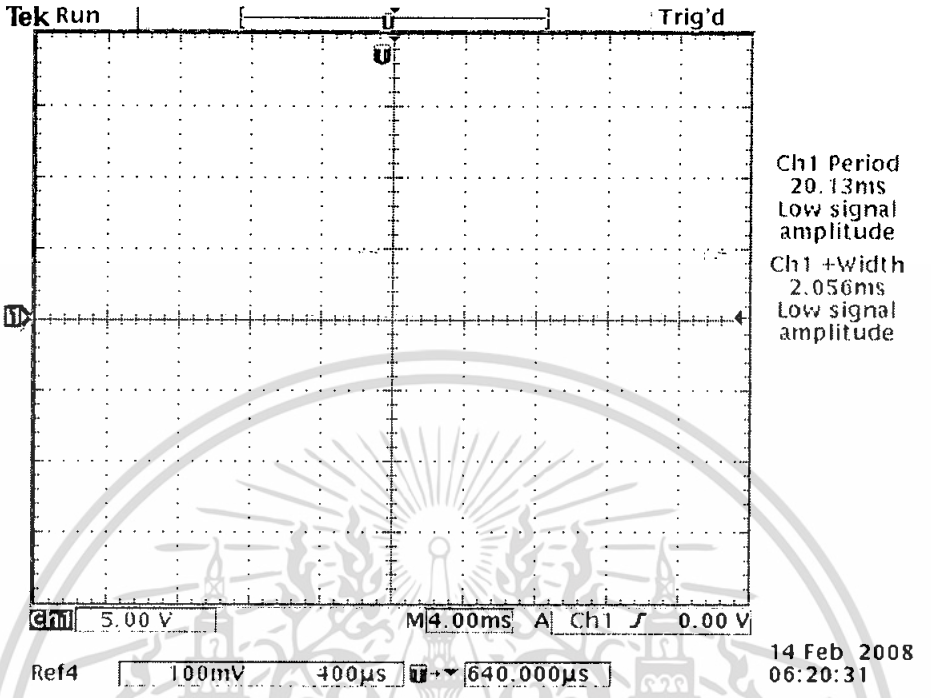
รูป 6-6 สัญญาณที่จ่ายให้ SERVO MOTOR ตัวที่ 1



รูป 6-7 สัญญาณที่จ่ายให้ SERVO MOTOR ตัวที่ 2



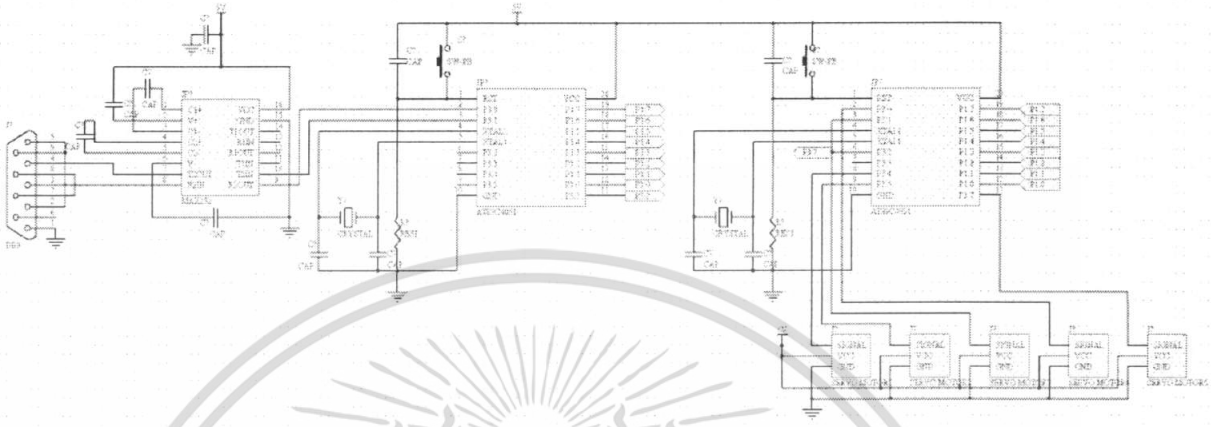
รูป 6-8 สัญญาณที่จ่ายให้ SERVO MOTOR ตัวที่ 3



รูป 6-9 สัญญาณที่จ่ายให้ SERVO MOTOR ตัวที่ 5

จากภาพ รูปที่ 6-6 ถึง รูปที่ 6-9 นั้นเป็นสัญญาณที่เกิดจากไมโครคอนโทรลเลอร์ ที่จ่ายให้กับ มอเตอร์ในส่วนต่างๆ ของแขนกล ซึ่งมีทั้งหมด 4 ภาพโดย จากเซอร์โวมอเตอร์ทั้งหมด 5 ตัว เนื่องจากขณะใช้เป็นแบบ sequence เซอร์โวมอเตอร์ตัวที่ 4 นั้นไม่มีการเปลี่ยน(เนื่องจากผู้จัดทำได้กำหนดขึ้น)จึงไม่ได้แสดงสัญญาณให้ดู ซึ่งสัญญาณที่ได้แสดงทั้งหมดนี้ได้ใช้ออสซิลโลสโคป เป็นอุปกรณ์ในการวัด ซึ่งพัลส์ดังกล่าว จะสามารถ ทำให้แขนกล มีสภาพพร้อมใช้งานดังใน รูปที่ 6-5

6.2 การทดลองการทำงานของแขนกล



รูป 6-10 วงจรที่ใช้ในการทดลอง

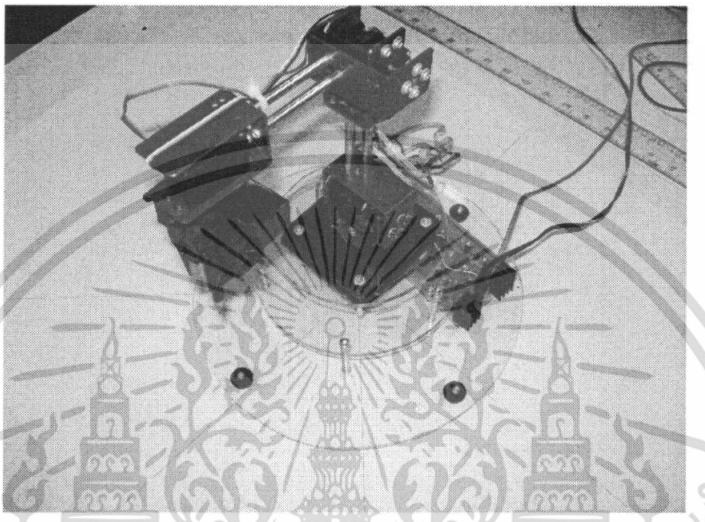
ในการทดลองการทำงานของแขนกลนี้ ได้แบ่งการทดลอง ออกเป็น 2 การทดลอง คือ

6.2.1 การวัดความแม่นยำของแขนกล

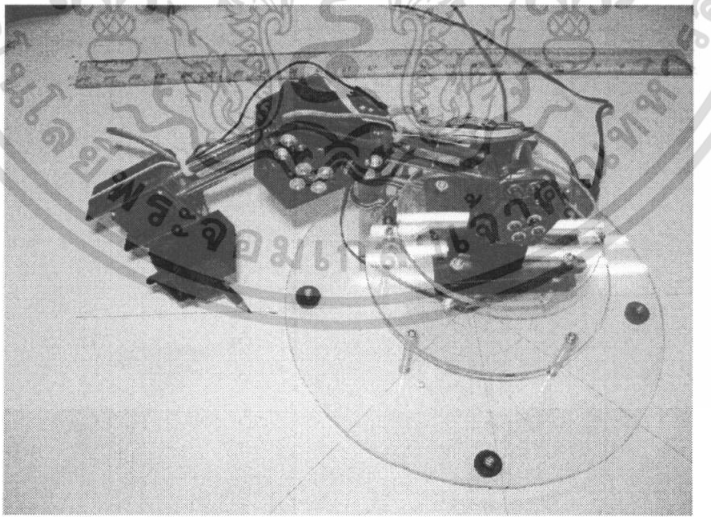
การทดลองนี้ เป็นการวัดความแม่นยำของแขนกล ซึ่งจะทำการโดยให้แขนกล จับดินสอแล้วนำไปจุดตรงที่ เราต้องการ โดยการสั่งงานทาง โปรแกรม Hyper Terminal โดยโปรแกรม จะส่งรหัส Ascii ออกมาเพื่อไปใช้ในการคอลโทรล ของไมโครคอนโทรลเลอร์ ซึ่งในการทดลองนี้แขนกลจะทำงานเป็นแบบ sequence

รูป 6-11 แบบที่ใช้ในการทดลอง

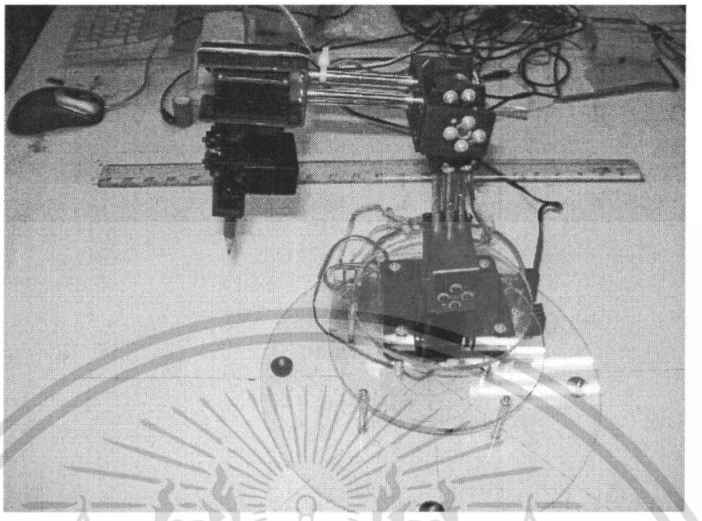
รูปที่ 6-10 นี้เป็นแบบวัด ที่ใช้ในการทดลอง ซึ่งทางผู้จัดทำได้จำลองขึ้นมาเพื่อวัดค่าความแม่นยำ ของแขนกล



รูป 6-12 แขนกลพร้อมที่จะทำการทดลอง



รูป 6-13 แขนกลทำการทดลอง 1



รูป 6-14 แขนกลทำการทดลอง 2

รูป 6-15 แขนกลทำการทดลอง 3

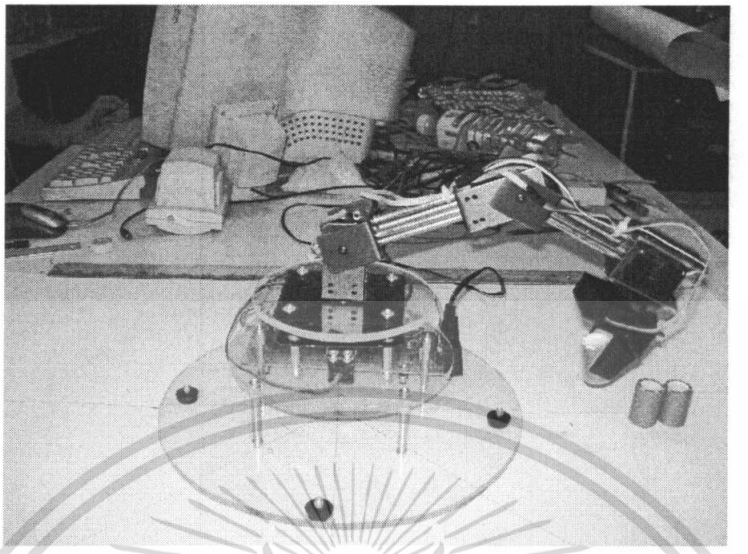
ซึ่งจากการทดลอง ทั้งหมด 10 ครั้งสามารถ สรุปได้ดังนี้

ครั้งที่	ระยะจากจุดศูนย์กลาง(cm)	มุม(องศา)
1	15.0	179
2	15.0	179
3	15.1	178
4	15.1	178
5	15.1	178
6	15.1	178
7	15.1	178
8	15.1	178
9	15.1	177
10	15.1	177

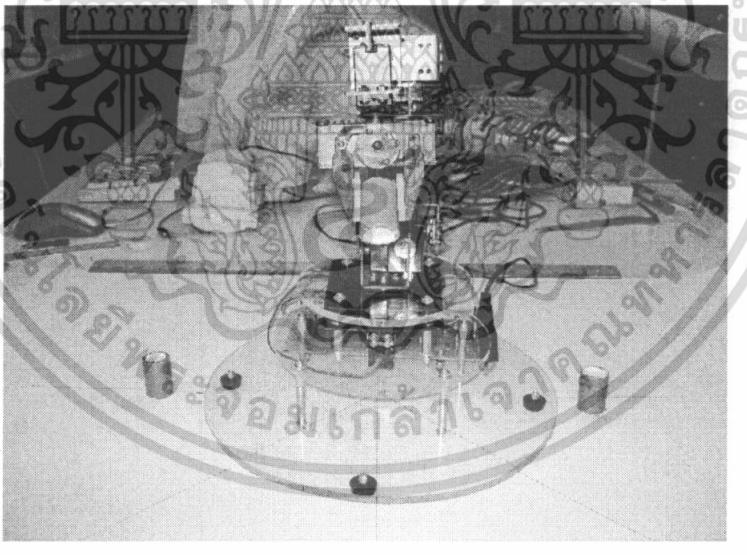
จากการทดลอง สังเกตุ ได้ว่าค่าที่ได้มีค่าใกล้เคียงกันมาก ไม่ว่าจะเป็นค่าของระยะ หรือ ค่าของมุมก็ตาม แต่จะมีค่าคลาดเคลื่อนเล็กน้อย ซึ่งค่าคลาดเคลื่อนดังกล่าว อาจเกิดขึ้น ได้จาก การกระทบ ของแขนกล กับ พื้น ทำให้ดินสอที่ grip ของแขนกลหนีบอยู่นั่น ทำให้เกิดความคลาดเคลื่อนได้

6.2.2 การทำงานของแขนกลแบบ sequence

การทดลองนี้เป็นการทดลอง การทำงาน ของแขนกล ซึ่งจะเป็นการทดลองแบบ sequence โดยจะสั่งคำสั่ง ผ่าน โปรแกรม Hyper Terminal เพื่อไปควบคุมการทำงานของแขนกล โดยการทดลอง จะให้แขนกลหยิบวัตถุ ทั้งหมด 10



รูป 6-16 แขนกลทำการทดลอง 4



รูป 6-17 แขนกลทำการทดลอง 5

รูป 6-18 แขนกลทำการทดลอง 6

จากการทดลองนี้ ซึ่งเป็นการทดลอง การทำงานของแขนกลแบบ sequence หยิบวัตถุ ทั้งหมด 10 ครั้ง สามารถหยิบได้ถึง 9 ครั้ง ซึ่งมีเพียงครั้งเดียวเท่านั้นที่ไม่สามารถ หยิบได้ ความคลาดเคลื่อนนี้เกิดจาก การ เคลื่อนที่ของแขนกล ในช่วงแรกที่ เคลื่อนที่ ไม่สม่ำเสมอ ซึ่งอาจเกิดจาก พัลส์ที่เกิดจากไมโครคอนโทรลเลอร์ จ่ายมาให้กับ เซอร์โวมอเตอร์ ไม่นิ่งเท่าที่ควร จึงทำให้ การเคลื่อนที่ของแขนกล เกิดการสั่นขึ้นได้ จึงทำให้เกิด ค่า ERROR ขึ้นได้

ซึ่งจากการทดลอง การทำงานของแขนกลนี้ ซึ่งผ่าน พอร์ตอนุกรม โดยใช้โปรแกรม Hyper Terminal ในการสั่งงาน เพื่อไปควบคุมการทำงาน ของแขนกล จากการทดลอง การทำงานนี้สรุปได้ว่า แขนกลมีความ แม่นยำ ค่อนข้างมาก แต่ยังมีค่า คลาดเคลื่อนเล็กน้อย ที่เกิดจากปัญหาต่างๆ ที่ได้พูดถึงไปในการ ทดลอง 6.2.1 และการทดลอง 6.2.1

บทที่ 7
สรุปและวิจารณ์ผลการทดลอง

โครงงานแขนกลควบคุมการทำงานด้วย MCS-51 ใช้ขับ Servo Motor การทำงานเบื้องต้นของแขนกล คือ สามารถขยับแขนตามมุมองศาที่กำหนด การกำหนดมุมองศาให้แขนกลทำได้โดยการควบคุมมอเตอร์ที่ใช้ขับเคลื่อนแขนมอเตอร์ตัวนี้เรียกว่า เซอร์โวมอเตอร์ ถ้าเราต้องการให้มอเตอร์ตัวนี้ทำงานต้องจ่ายสัญญาณพัลส์ความกว้าง 20mS และการควบคุมมุมของมอเตอร์ทำได้โดยจ่ายช่วงพัลส์บวกที่มีขนาดต่างๆตัวอย่างเช่นถ้าต้องการให้มอเตอร์หมุนไปที่ 90 องศาเราจะต้องจ่ายพัลส์บวกที่มีความกว้างเท่ากับ 0.5mS และถ้าต้องการให้มอเตอร์หมุนไปที่ -90 องศาเราต้องจ่ายพัลส์ซีกบวกที่มีความกว้างเท่ากับ 2mS เป็นต้น เมื่อเราสามารถควบคุมมุมของมอเตอร์ได้แล้วเราก็จะสามารถควบคุมแขนกลได้จากหลักการเบื้องต้นนี้ จึงได้จัดการทำงานของมอเตอร์ โดยมีการจำลองการทำงานดูก่อนแล้วค่อยลงมือทดลองจริง ดังนั้นจึงมีการทดลองดังนี้

7.1 การจำลองการทำงาน

ก่อนจะทดลองการทำงานจะมีการจำลองการทำงานก่อน เพื่อดูว่ามีโอกาสมากน้อยเพียงใดที่จะทำให้ มอเตอร์หมุน จากการจำลองการทำงานจึงได้นำโปรแกรมที่สามารถจำลองการทำงานได้เสมือนจริง โปรแกรมนี้เรียกว่า Proteus ซึ่งโปรแกรมนี้สามารถจำลองการทำงานของวงจรทางอิเล็กทรอนิกส์ได้ เมื่อนำมาใช้กับโครงงานนี้ จึงสามารถใช้ได้อย่างเหมาะสมโดยสามารถเขียนโปรแกรมแล้วทำการโปรแกรมลงใน MCS-51 ที่มีโปรแกรมในตัวนี้ได้เลย ซึ่งผลจากการทำงานนั้นก็เป็นที่น่าพอใจมากเพราะในการจำลองการทำงานได้เขียนโปรแกรมการทำงานโดยใช้ Keil 3 แล้วทำการโปรแกรมลง MCS-51 เบอร์ AT89C4051 ปรากฏว่าการจำลองการทำงานสามารถทำงานได้จริง ซึ่งพัลส์ที่เกิดจากการเขียนโปรแกรมสามารถขับมอเตอร์ให้หมุนได้ ตามความต้องการของผู้ทดลอง ดังนั้นเมื่อสามารถทำงานได้ แล้วจึงทำการทดลองวงจรจริงต่อไป

7.2 การทดลองการทำงานของวงจร

จากที่เราได้ทำการทดลองการทำงานของวงจรแล้วทำให้เราทราบว่าขับมอเตอร์ได้จริง จากนั้นถึงขั้นตอนทดลองต่อวงจรจริง ก็พบกับปัญหาเล็กน้อย เช่นสัญญาณพัลส์ไม่ออกมาจาก MCS-51 ไม่เหมือนกับที่ได้จำลองการทำงานไว้ เมื่อแก้ไขจนมีสัญญาณพัลส์ออกมาก็ปรากฏว่า

ขนาดของสัญญาณมีการคลาดเคลื่อนบ้างเล็กน้อย แนวทางในการแก้ไข คือการศึกษาโครงสร้างภายในของ MCS-51 เบอร์ AT89C4051 ปรากฏว่าบางขาต้องต่อตัวต้านทานภายนอกด้วย ดังนั้นจึงได้เขียนโปรแกรมขึ้นมาใหม่เพื่อให้สัมพันธ์กับการทำงานและต่ออุปกรณ์ภายนอกเมื่อแก้ปัญหาได้แล้วปรากฏว่ามีสัญญาณพัลส์ออกมาจาก MCS-51 ตามที่ได้ออกแบบไว้และสัญญาณพัลส์นี้ยังสามารถควบคุมมอเตอร์ให้หมุนไปในมุมที่ต้องการได้ดังนั้นการทดลองการทำงานจึงถือว่าได้ประสบความสำเร็จแล้ว

บรรณานุกรม

1. **ชัยวัฒน์ ลิ้มพรจิตรวิไล , นคร ภักดีชาติ , "ทดลองและใช้งานไมโครคอนโทรลเลอร์ MCS-51 ด้วยโปรแกรมภาษาซี สำหรับ AT89Cx051", บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด,** 192 หน้า
2. "คู่มือการใช้งาน SERVO MOTOR " , บริษัท อีทีที จำกัด ETT CO.,LTD.

ภาคผนวก

โปรแกรมไมโครคอนโทรลเลอร์ตัวที่ 1

```
#include<reg51.h>
sbit output1=P1;
sbit output2=P3^7;
int a=0;
void main(void){
TMOD=0x22;
SCON=0x50;
EA=1;
ET0=1;
ES=1;
TF1=0;
TH0=26;
TL0=26;
TH1=0xFD;
TL1=0xFD;
TR1=1;
while(1){
        output2=1;
        if(a==400){
                output2=0;
                TR0=0;
                a=0;
                }
        }
}
```

```
void rs232(void) interrupt 4{
RI=0;
P1=SBUF;
TR0=1;
}
void time0(void) interrupt 1{
TF0=0;
a++;
}
```

โปรแกรมไมโครคอนโทรลเลอร์ตัวที่ 2

```
#include<reg51.h>
sbit input=P1;
sbit output1=P3^4;
sbit output2=P3^5;
sbit output3=P3^1;
sbit output4=P3^0;
sbit output5=P3^7;
unsigned int num=0;t1=29;t2=32;t3=47;t4=32;t5=39;count=0;z;
char a;
bit b=0,c=0,d=0,e=0,f=0,g=0,h=0,i=0,j=0,k=0,l=0,m=0,n=0;
void init_port(void){
output1=1;
output2=1;
output3=1;
output4=1;
output5=1;
}
void  delay(unsigned int num){
while(num<250){
        num++;
        }
if(num==250)num=0;
}
```

```
void main(void){
TMOD=0x22;
EA=1;
ET0=1;
EX0=1;
TF0=0;
TH0=196;
TL0=196;
IT0=1;
init_port();
TR0=1;
while(1){
        if(b==0){
                if(t1>47) t1=47;
                if(t2>49) t2=49;
                if(t3>47) t3=47;
                if(t4>47) t4=47;
                if(t5>47) t5=47;
                if(t1<12) t1=12;
                if(t2<27) t2=27;
                if(t3<27) t3=27;
                if(t4<12) t4=12;
                if(t5<12) t5=12;
                }
        if(b==1){
                if(c==1){
                        if(t1<47)t1++;
                        delay(100);
                        if(t1==47){c=0;d=1;}
                        }
```

```
if(d==1){
        if(t3>40)t3--;
        delay(100);
        if(t3==40){d=0;e=1;}
        }
if(e==1){
        if(t2<49)t2++;
        delay(100);
        if(t2==49){e=0;f=1;}
        }
if(f==1){
        if(t5>z)t5--;
        delay(100);
        if(t5==z){f=0;g=1;}
        }
if(g==1){
        if(t2>32)t2--;
        delay(100);
        if(t2==32){g=0;h=1;}
        }
if(h==1){
        if(t3<47)t3++;
        delay(100);
        if(t3==47){h=0;i=1;}
        }
if(i==1){
        if(t1>12)t1--;
        delay(100);
        if(t1==12){i=0;j=1;}
        }
```

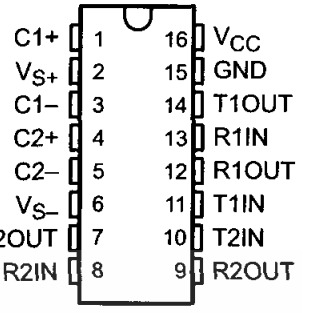
```
            if(j==1){
                if(t3>40)t3--;
                delay(100);
                if(t3==40){j=0;k=1;}
                }
            if(k==1){
                if(t2<49)t2++;
                delay(100);
                if(t2==49){k=0;l=1;}
                }
            if(l==1){
                if(t5<39)t5++;
                delay(100);
                if(t5==39){l=0;m=1;}
                }
            if(m==1){
                if(t2>32)t2--;
                delay(100);
                if(t2==32){m=0;n=1;}
                }
            if(n==1){
                if(t3<47)t3++;
                delay(100);
                if(t3==47){n=0;c=1;}
                }
            }
        }
}
```

```
void service(void) interrupt 1{
TF0=0;
count++;
if(count>=t1)output1=0;
if(count>=t2)output2=0;
if(count>=t3)output3=0;
if(count>=t4)output4=0;
if(count>=t5)output5=0;
if(count>=433){
                        count=0;
                        init_port();
                    }
}
void service_EX0(void) interrupt 0{
IE0=0;
a=P1;
switch (a){     case 0x36 :  {t1++;b=0;}  break;
                case 0x34 :  {t1--;b=0;}  break;
                case 0x38 :  {t2++;b=0;}  break;
                case 0x35 :  {t2--;b=0;}  break;
                case 0x77 :  {t3++;b=0;}  break;
                case 0x73 :  {t3--;b=0;}  break;
                case 0x64 :  {t4++;b=0;}  break;
                case 0x61 :  {t4--;b=0;}  break;
                case 0x65 :  {t5++;b=0;}  break;
                case 0x71 :  {t5--;b=0;}  break;
                case 0x43 :  {num=0;b=1;c=1;z=20;}  break;
                case 0x4B :  {num=0;b=1;c=1;z=25;}  break;
        }
}
```

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

- **Meet or Exceed TIA/EIA-232-F and ITU Recommendation V.28**
- **Operate With Single 5-V Power Supply**
- **Operate Up to 120 kbit/s**
- **Two Drivers and Two Receivers**
- **±30-V Input Levels**
- **Low Supply Current . . . 8 mA Typical**
- **Designed to be Interchangeable With Maxim MAX232**
- **ESD Protection Exceeds JESD 22 – 2000-V Human-Body Model (A114-A)**
- **Applications**
  - **TIA/EIA-232-F**
  - **Battery-Powered Systems**
  - **Terminals**
  - **Modems**
  - **Computers**

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube	MAX232N	MAX232N
	SOIC (D)	Tube	MAX232D	MAX232
		Tape and reel	MAX232DR	
	SOIC (DW)	Tube	MAX232DW	MAX232
		Tape and reel	MAX232DWR	
	SOP (NS)	Tape and reel	MAX232NSR	MAX232
–40°C to 85°C	PDIP (N)	Tube	MAX232IN	MAX232IN
	SOIC (D)	Tube	MAX232ID	MAX232I
		Tape and reel	MAX232IDR	
	SOIC (DW)	Tube	MAX232IDW	MAX232I
		Tape and reel	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

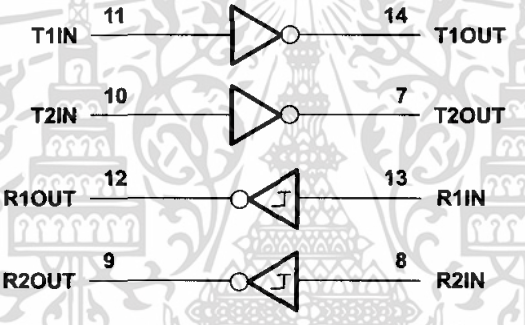
H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) . . . . . –0.3 V to 6 V
Positive output supply voltage range, V_{S+} . . . . . V_{CC} – 0.3 V to 15 V
Negative output supply voltage range, V_{S-} . . . . . –0.3 V to –15 V
Input voltage range, V_I: Driver . . . . . –0.3 V to V_{CC} + 0.3 V
Receiver . . . . . ±30 V
Output voltage range, V_O: T1OUT, T2OUT . . . . . V_{S-} – 0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT . . . . . –0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT . . . . . Unlimited
Package thermal impedance, θ_{JA} (see Note 2): D package . . . . . 73°C/W
DW package . . . . . 57°C/W
N package . . . . . 67°C/W
NS package . . . . . 64°C/W
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds . . . . . 260°C
Storage temperature range, T_{stg} . . . . . –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values are with respect to network ground terminal.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	–40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (see Note 3 and Figure 4)

PARAMETER		TEST CONDITIONS		MIN	TYP‡	MAX	UNIT
I_{CC}	Supply current	V_{CC} = 5.5 V, T_A = 25°C	All outputs open,		8	10	mA

‡ All typical values are at V_{CC} = 5 V and T_A = 25°C.

NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

DRIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 3)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	T1OUT, T2OUT	R_L = 3 kΩ to GND		5	7		V
V_{OL}	Low-level output voltage‡	T1OUT, T2OUT	R_L = 3 kΩ to GND			–7	–5	V
r_o	Output resistance	T1OUT, T2OUT	V_{S+} = V_{S-} = 0,	V_O = ±2 V	300			Ω
I_{OS}§	Short-circuit output current	T1OUT, T2OUT	V_{CC} = 5.5 V,	V_O = 0		±10		mA
I_{IS}	Short-circuit input current	T1IN, T2IN	V_I = 0				200	µA

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
§ Not more than one output should be shorted at a time.
NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 3)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
SR	Driver slew rate	R_L = 3 kΩ to 7 kΩ, See Figure 2			30	V/µs
SR(t)	Driver transition region slew rate	See Figure 3		3		V/µs
	Data rate	One TOUT switching		120		kbit/s

NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

RECEIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 3)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	R1OUT, R2OUT	I_{OH} = –1 mA		3.5			V
V_{OL}	Low-level output voltage‡	R1OUT, R2OUT	I_{OL} = 3.2 mA				0.4	V
V_{IT+}	Receiver positive-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C		1.7	2.4	V
V_{IT-}	Receiver negative-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C	0.8	1.2		V
V_{hys}	Input hysteresis voltage	R1IN, R2IN	V_{CC} = 5 V		0.2	0.5	1	V
r_i	Receiver input resistance	R1IN, R2IN	V_{CC} = 5,	T_A = 25°C	3	5	7	kΩ

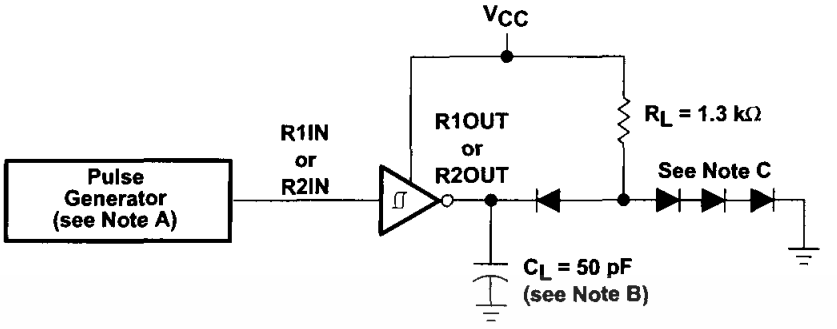
† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 3 and Figure 1)

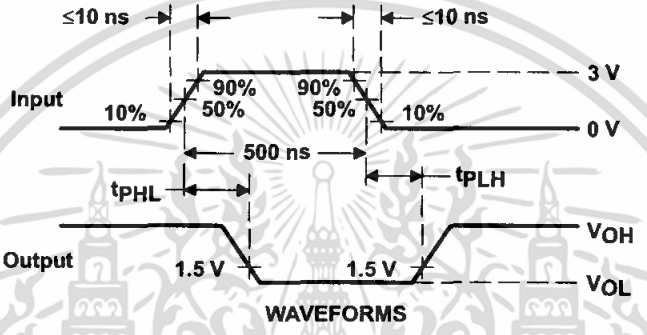
PARAMETER		TYP	UNIT
$t_{PLH(R)}$	Receiver propagation delay time, low- to high-level output	500	ns
$t_{PHL(R)}$	Receiver propagation delay time, high- to low-level output	500	ns

NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

PARAMETER MEASUREMENT INFORMATION



TEST CIRCUIT

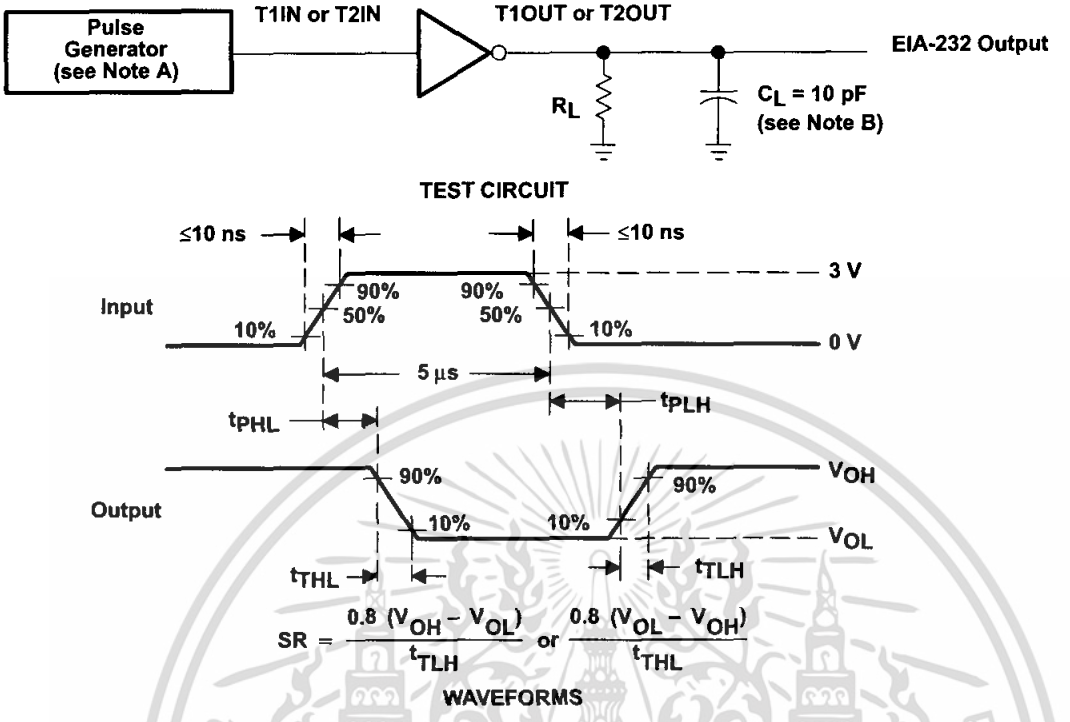


WAVEFORMS

NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.
C. All diodes are 1N3064 or equivalent.

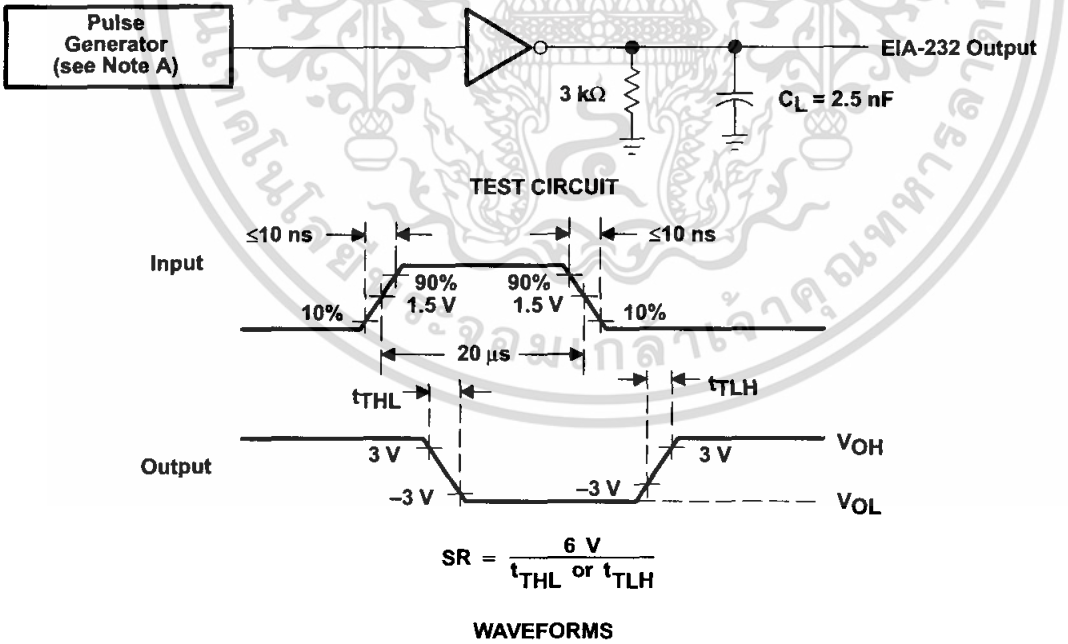
Figure 1. Receiver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements

PARAMETER MEASUREMENT INFORMATION



NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.

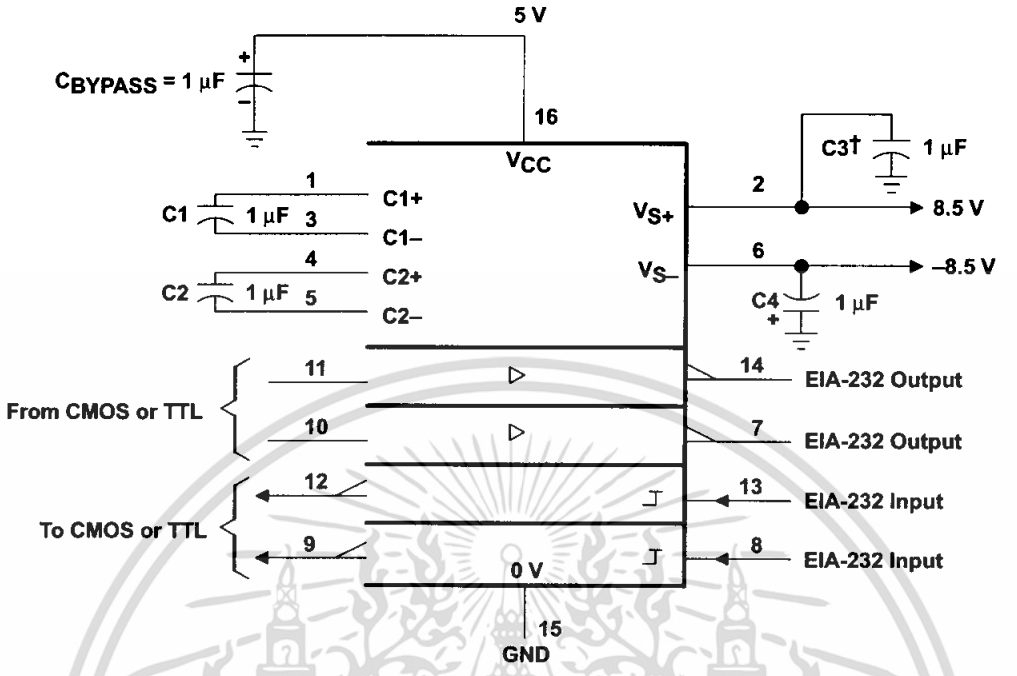
Figure 2. Driver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements (5-µs Input)



NOTE A: The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.

Figure 3. Test Circuit and Waveforms for t_{THL} and t_{TLH} Measurements (20-µs Input)

APPLICATION INFORMATION



† C3 can be connected to V_{CC} or GND.

Figure 4. Typical Operating Circuit

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Features

- Compatible with MCS®51 Products
- 4K Bytes of Reprogrammable Flash Memory
- Endurance: 1,000 Write/Erase Cycles
- 2.7V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Two-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 15 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial UART Channel
- Direct LED Drive Outputs
- On-chip Analog Comparator
- Low-power Idle and Power-down Modes
- Brown-out Detection

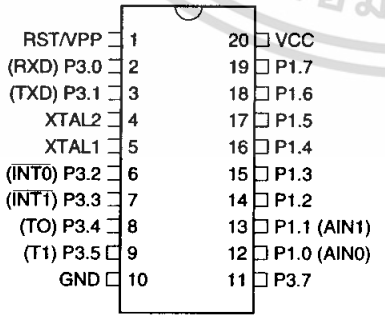
Description

The AT89C4051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read-only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C4051 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89C4051 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five-vector, two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C4051 is designed with static logic for operation down to zero frequency and supports two software-selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

PDIP/SOIC

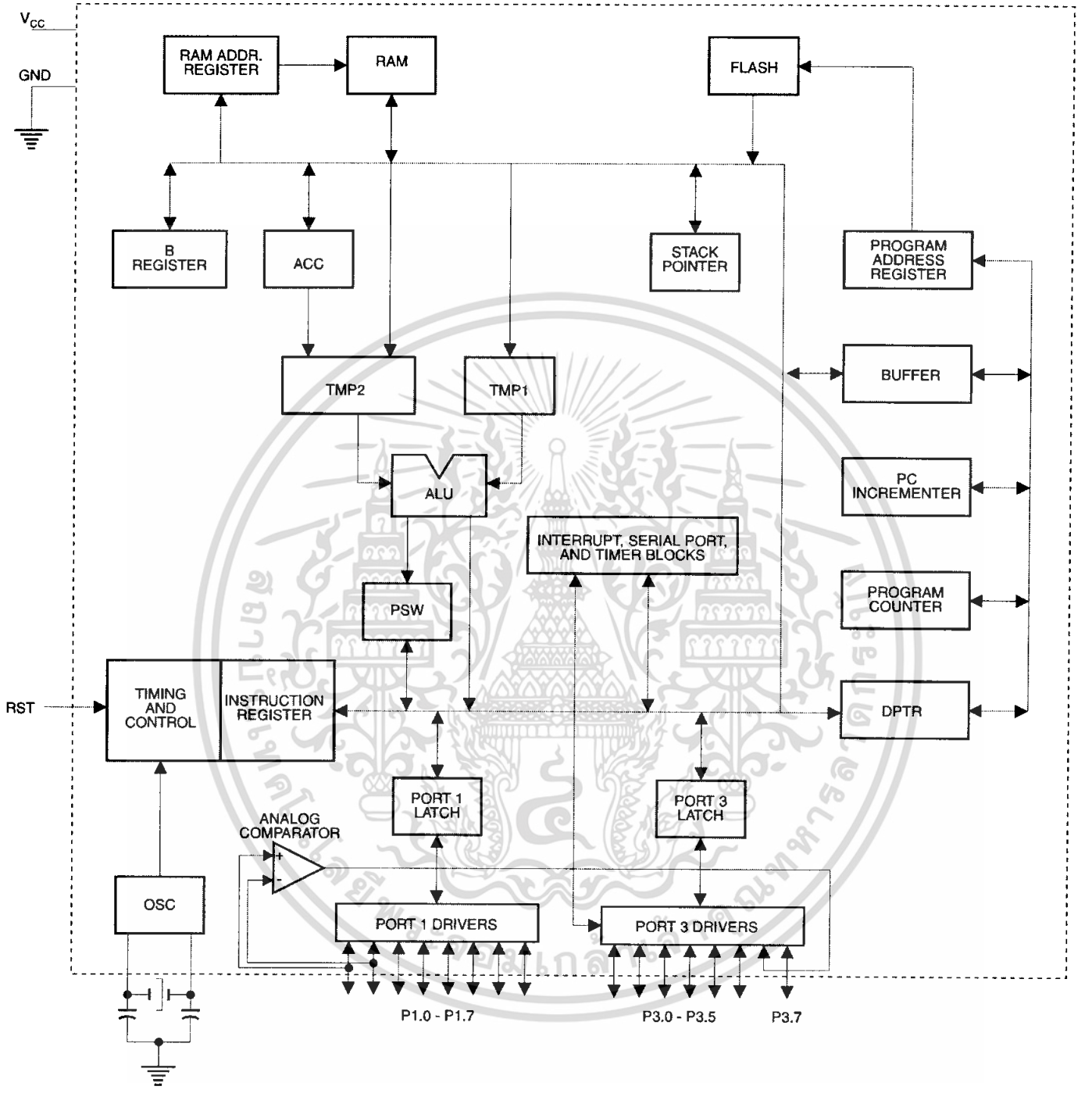


8-bit Microcontroller with 4K Bytes Flash

AT89C4051

Rev. 1001D–06/01

Block Diagram



Pin Description

VCC Supply voltage.

GND Ground.

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

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

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

Port 3 also serves the functions of various special features of the AT89C4051 as listed below:

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

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

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

Each machine cycle takes 12 oscillator or clock cycles.

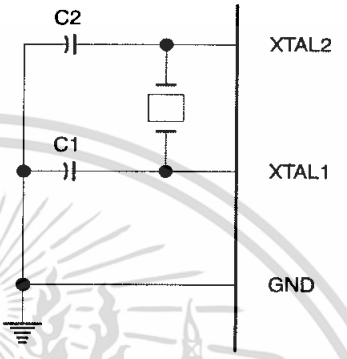
XTAL1 Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2 Output from the inverting oscillator amplifier.

Oscillator Characteristics

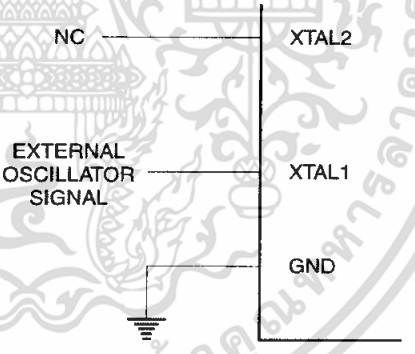
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected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

Figure 1. Oscillator Connections



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

Figure 2. External Clock Drive Configuration



Special Function Registers

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

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

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

Table 1. AT89C4051 SFR Map and Reset Values

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

Restrictions on Certain Instructions

The AT89C4051 is an economical and cost-effective member of Atmel's growing family of microcontrollers. It contains 4K bytes of Flash program memory. It is fully compatible with the MCS-51 architecture, and can be programmed using the MCS-51 instruction set. However, there are a few considerations one must keep in mind when utilizing certain instructions to program this device.

All the instructions related to jumping or branching should be restricted such that the destination address falls within the physical program memory space of the device, which is 4K for the AT89C4051. This should be the responsibility of the software programmer. For example, LJMP 0FE0H would be a valid instruction for the AT89C4051 (with 4K of memory), whereas LJMP 1000H would not.

Branching Instructions

LCALL, LJMP, ACALL, AJMP, SJMP, JMP @A+DPTR. These unconditional branching instructions will execute correctly as long as the programmer keeps in mind that the destination branching address must fall within the physical boundaries of the program memory size (locations 00H to FFFH for the 89C4051). Violating the physical space limits may cause unknown program behavior.

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

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

MOVX-related Instructions, Data Memory

The AT89C4051 contains 128 bytes of internal data memory. Thus, in the AT89C4051 the stack depth is limited to 128 bytes, the amount of available RAM. External DATA memory access is not supported in this device, nor is external PROGRAM memory execution. Therefore, no MOVX [...] instructions should be included in the program.

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

Program Memory Lock Bits

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

Lock Bit Protection Modes[1]

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

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

dle Mode

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

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

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

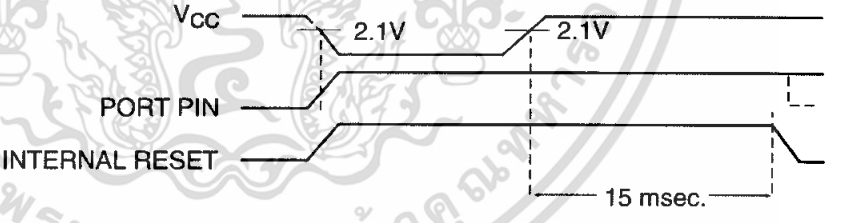
?ower-down Mode

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

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

3rown-out Detection

When V_{CC} drops below the detection threshold, all port pins (except P1.0 and P1.1) are weakly pulled high. When V_{CC} goes back up again, an internal Reset is automatically generated after a delay of typically 15 msec. The nominal brown-out detection threshold is 2.1V ± 10%.



Programming The Flash

The AT89C4051 is shipped with the 4K bytes of on-chip PEROM code memory array in the erased state (i.e., contents = FFH) and ready to be programmed. The code memory array is programmed one byte at a time. *Once the array is programmed, to re-program any non-blank byte, the entire memory array needs to be erased electrically.*

Internal Address Counter: The AT89C4051 contains an internal PEROM address counter which is always reset to 000H on the rising edge of RST and is advanced by applying a positive going pulse to pin XTAL1.

Programming Algorithm: To program the AT89C4051, the following sequence is recommended.

1. Power-up sequence:
   Apply power between VCC and GND pins
   Set RST and XTAL1 to GND
2. Set pin RST to "H"
   Set pin P3.2 to "H"
3. Apply the appropriate combination of "H" or "L" logic
   levels to pins P3.3, P3.4, P3.5, P3.7 to select one of the programming operations shown in the PEROM Programming Modes table.

To Program and Verify the Array:

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

$\overline{\text{Data}}$ Polling: The AT89C4051 features $\overline{\text{Data}}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P1.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{\text{Data}}$ Polling may begin any time after a write cycle has been initiated.

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

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

1. Reset the internal address counter to 000H by bringing RST from "L" to "H".
2. Apply the appropriate control signals for Read Code data and read the output data at the port P1 pins.

3. Pulse pin XTAL1 once to advance the internal address counter.
4. Read the next code data byte at the port P1 pins.
5. Repeat steps 3 and 4 until the entire array is read.

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

Chip Erase: The entire PEROM array (4K bytes) and the two Lock Bits are erased electrically by using the proper combination of control signals and by holding P3.2 low for 10 ms. The code array is written with all "1"s in the Chip Erase operation and must be executed before any non-blank memory byte can be re-programmed.

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

(000H) = 1EH indicates manufactured by Atmel

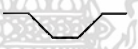
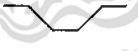
(001H) = 41H indicates 89C4051

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

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

Figure 3. Programming the Flash Memory

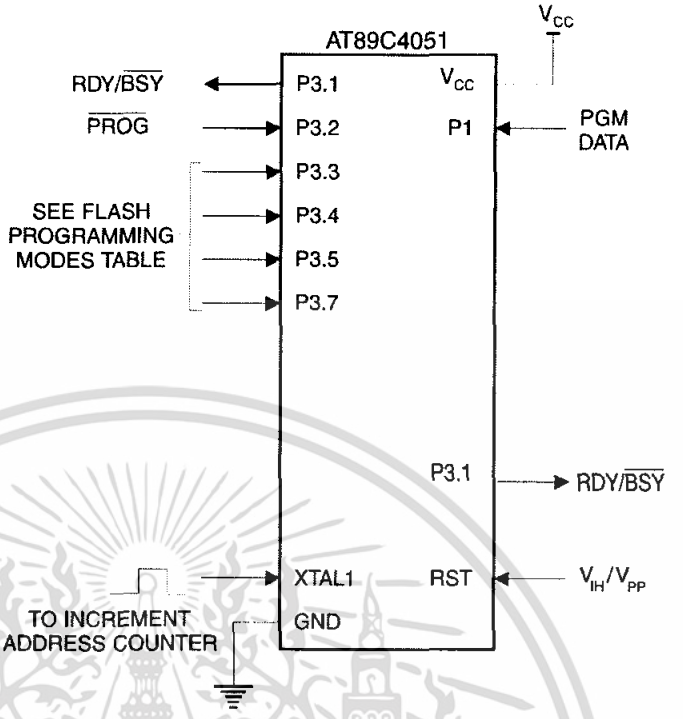
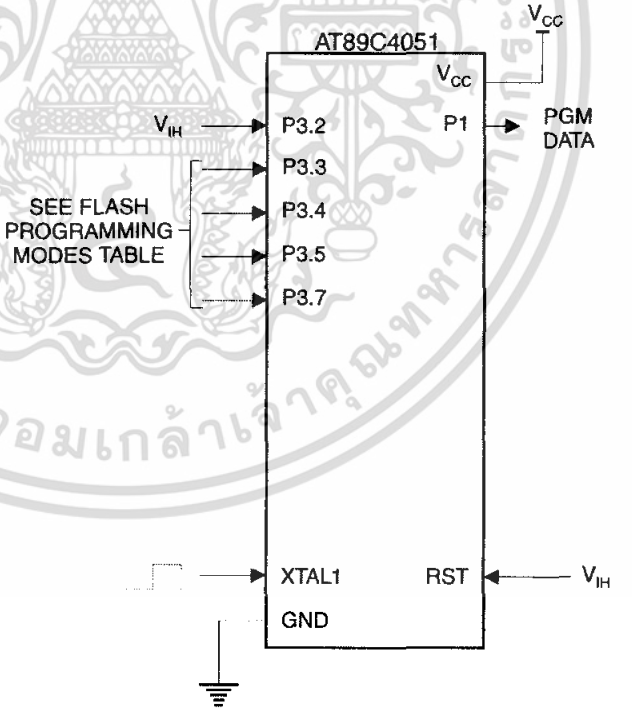


Figure 4. Verifying the Flash Memory



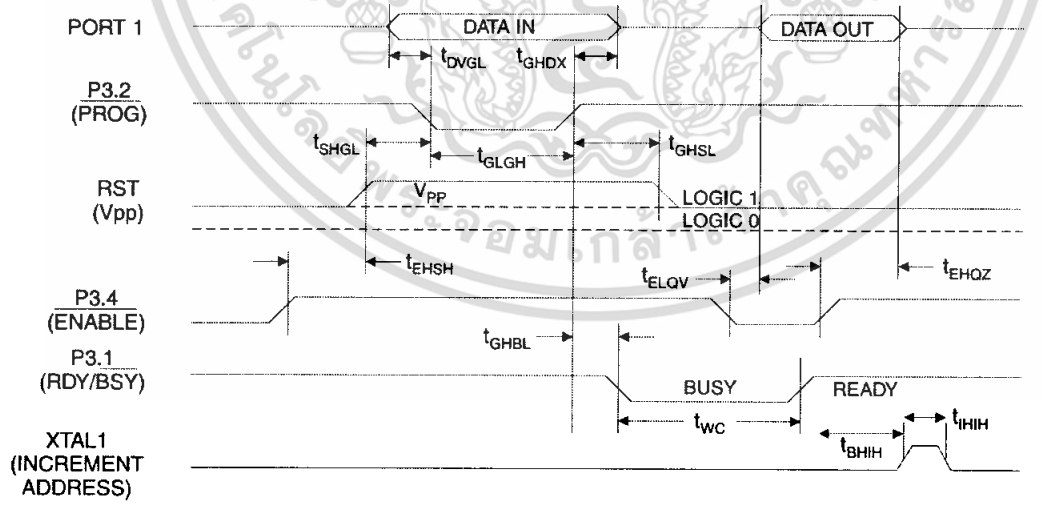
Flash Programming and Verification Characteristics

T_A = 20°C to 30°C, V_{CC} = 5.0 ± 10%

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

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

Flash Programming and Verification Waveforms



Absolute Maximum Ratings*

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

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

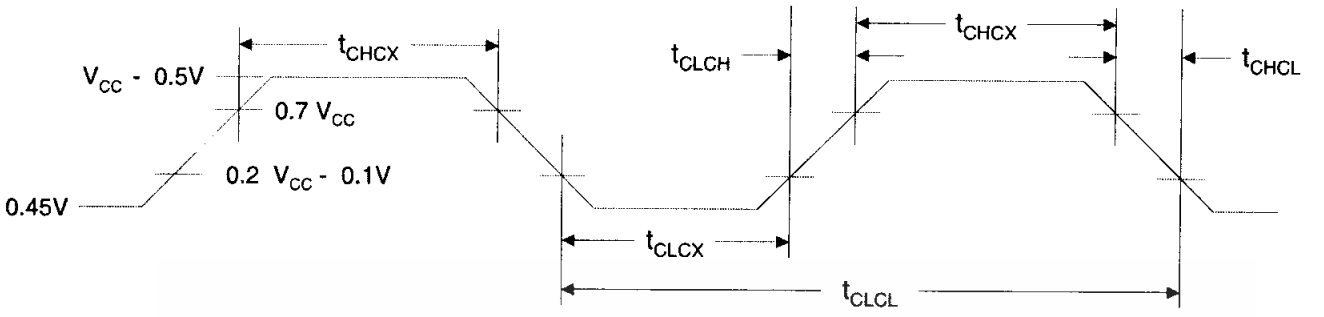
DC Characteristics

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

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

Notes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 20 mA
   Maximum total I_{OL} for all output pins: 80 mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power-down is 2V.

External Clock Drive Waveforms



External Clock Drive

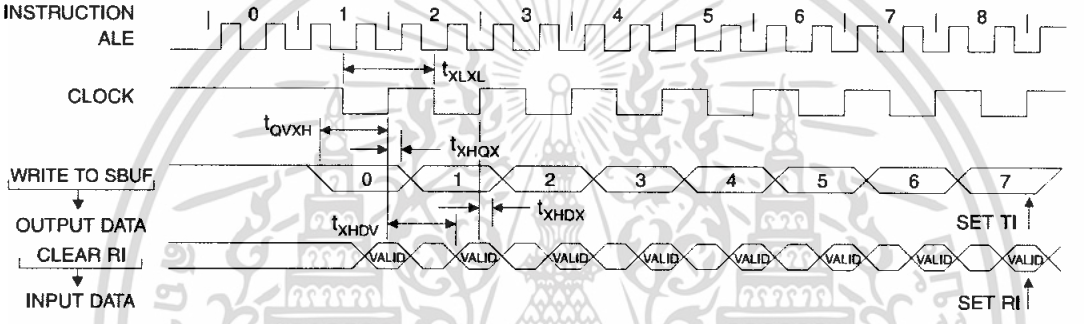
Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	12	0	24	MHz
t_{CLCL}	Clock Period	83.3		41.6		ns
t_{CHCX}	High Time	30		15		ns
t_{CLCX}	Low Time	30		15		ns
t_{CLCH}	Rise Time		20		20	ns
t_{CHCL}	Fall Time		20		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

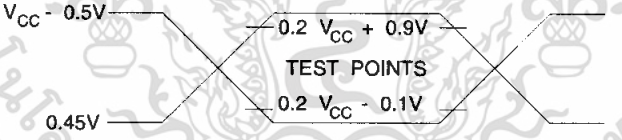
V_{CC} = 5.0V ± 20%; Load Capacitance = 80 pF

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

Shift Register Mode Timing Waveforms

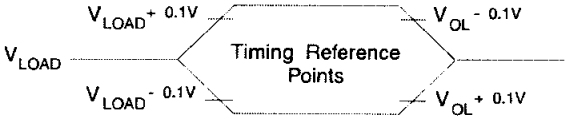


AC Testing Input/Output Waveforms(1)



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

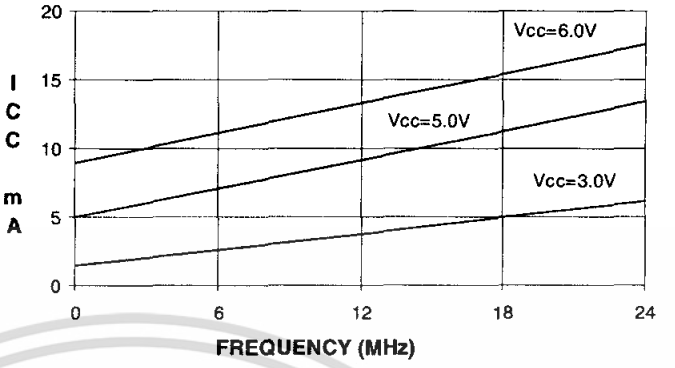
Float Waveforms(1)



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

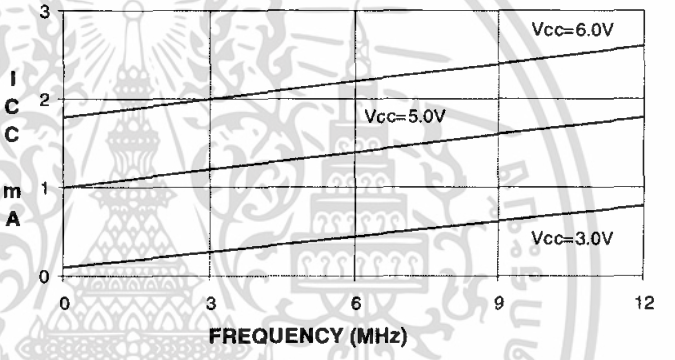
AT89C4051

TYPICAL ICC - ACTIVE (85°C)



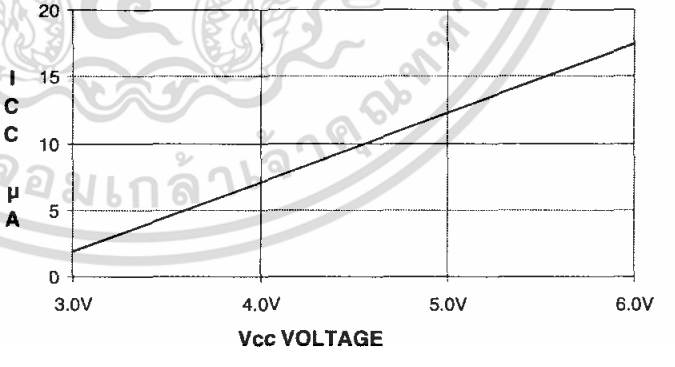
AT89C4051

TYPICAL ICC - IDLE (85°C)



AT89C4051

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



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

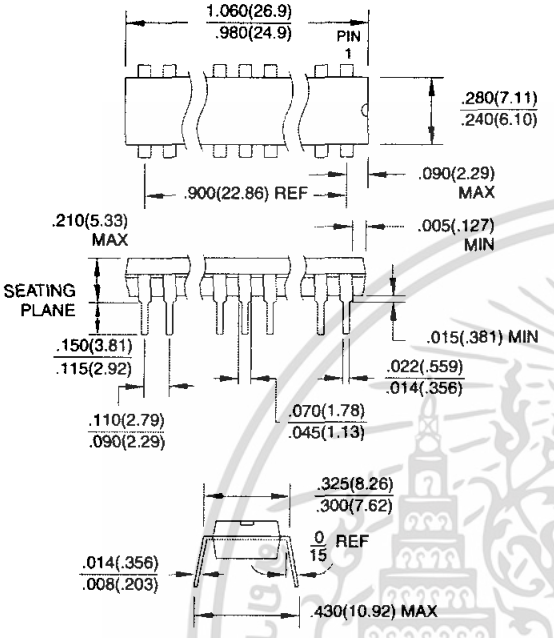
Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	2.7V to 6.0V	AT89C4051-12PC AT89C4051-12SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C4051-12PI AT89C4051-12SI	20P3 20S	Industrial (-40°C to 85°C)
24	4.0V to 6.0V	AT89C4051-24PC AT89C4051-24SC	20P3 20S	Commercial (0°C to 70°C)
		AT89C4051-24PI AT89C4051-24SI	20P3 20S	Industrial (-40°C to 85°C)

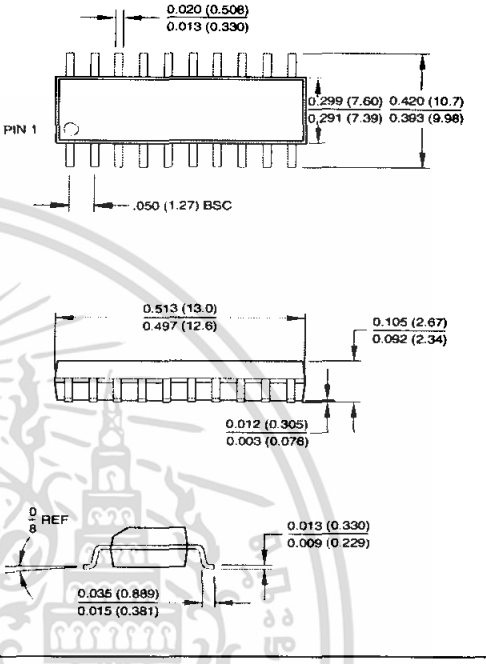
Package Type	
20P3	20-lead, 0.300" Wide, Plastic Dual In-line Package (PDIP)
20S	20-lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)

Packaging Information

20P3, 20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-001 AD



20S, 20-lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)
Dimensions in Inches and (Millimeters)



Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

Printed on recycled paper.

1001D–06/01/xM