เครื่องวัดรอบใบพัดมิเตอร์น้ำโดยการใช้มอเตอร์ไฟฟ้ากระแสตรง

Propeller blade round water meter by using direct electric current motor

นางสาวพัชมณฑ์ คเชนทร์ไพร รหัส 45370178

นายมนตรี วิฑูรทัศน์ รหัส 45370202

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2548

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน เครื่องวัดรอบใบพัดมิเตอร์น้ำโดยการใช้มอเตอร์ไฟฟ้ากระแสตรง

ผู้ดำเนินโครงงาน นางสาวพัชมณฑ์ คเชนทร์ไพร รหัส 45370178
นายมนตรี วิฑูรทัศน์ รหัส 45370202

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น

สาขา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2548

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น)

.....กรรมการ
(ดร.สมยศ เกียรติวนิชวิไล)

.....กรรมการ
(ดร.พนมขวัญ ริยะมงคล)

หัวข้อโครงงาน	เครื่องวัดรอบใบพัดมิเตอร์น้ำโดยการใช้มอเตอร์ไฟฟ้ากระแสตรง
ผู้ดำเนินโครงงาน	นางสาวพัชมณฑ์ คเชนทร์ไพร รหัส 45370178
	นายมนตรี วิฑูรทัศน์ รหัส 45370202
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2548

.....

บทคัดย่อ

โครงงานนี้เป็นการศึกษาเครื่องวัดรอบใบพัดมิเตอร์น้ำโดยการใช้มอเตอร์ไฟฟ้ากระแสตรง การดำเนินโครงงานได้แบ่งเป็นสองส่วนหลักคือ สร้างวงจรควบคุมและเขียนโปรแกรมควบคุม ในการสร้างวงจรควบคุมจะมีวงจร stepping มอเตอร์และวงจร sensor ซึ่งใช้ในเครื่องวัดรอบ สำหรับการเขียนโปรแกรมควบคุม ได้ใช้ภาษา C ในการเขียนโปรแกรม ผ่าน port RS232 เพื่อควบคุมมอเตอร์และเครื่องวัดรอบ

จากการทดสอบให้ใบพัดหมุนเป็นเวลา 5 นาที ทำซ้ำ 10 ครั้ง พบว่าค่าเฉลี่ยของรอบที่ได้จากเครื่องวัดรอบนั้นช้ากว่าค่าเฉลี่ยใบพัดอยู่ 4 เท่า และพบว่าค่าความผิดพลาด 25 เปอร์เซ็นต์ นี้เกิดจากหัวนับวัดสัญญาณที่นับรอบไม่ทัน

Project title	propeller blade round water meter by using direct electric current motor
Name	Miss.Patchamon Kachenphrai ID. 45370178
	Mr. Montree Vitoontus ID. 45370202
Project advisor	Assistant Professor Suchart Yammen, Ph.D.
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic year	2005

.....

Abstract

This project is to study a propeller blade round water meter by using a direct electric current motor and divided into two part : Hardware and Software.The first part is to build the control circuit consisting of step motor and sensor circuits for measuring the propeller blade. The lastpart is to write the C program through a RS232 port for controlling the motor and propeller blade revolution.

From using the propeller blade round meter test in five minutes and turning it in ten times,we found that the average value of the revolution of the round meter is still slower than that of the motor in four times and the error is twenty-five percentage.This is because the measured signal obtained from the motor cannot count immediately.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความช่วยเหลือจากหลายๆ ท่าน ผู้จัดทำจึงถือโอกาสนี้กราบขอพระคุณ

ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มแม่น ซึ่งเป็นอาจารย์ที่ปรึกษา ซึ่งได้ให้คำปรึกษาชี้แนะแนวทางและข้อคิดเห็นต่างๆ ในการแก้ปัญหาที่เป็นประโยชน์ อย่างสูงในการทำโครงงานนี้ให้สำเร็จลุล่วงด้วยดี

ขอขอบคุณอาจารย์ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ทุกท่าน ที่ได้ประสิทธิ์ประสาทความรู้จนสามารถนำมาประยุกต์ใช้งานได้

ขอขอบคุณพี่ๆ เพื่อนๆ และน้องๆ นิสิตภาควิชาวิศวกรรมไฟฟ้าทุกคนที่ได้ให้ความช่วยเหลือในหลายๆด้าน ทั้งเรื่องส่วนตัวและเรื่องเรียนมาด้วยดีเสมอมา

ท้ายนี้ผู้จัดทำโครงงาน ขอกราบขอบพระคุณบิดา มารดา และญาติพี่น้องของข้าพเจ้าที่คอยเลี้ยงดูและให้การสนับสนุนด้านการเงิน รวมทั้งเป็นกำลังใจให้ผู้จัดทำเสมอมาจนสำเร็จการศึกษา

คณะผู้จัดทำโครงงาน
นายมนตรี วิฑูรทัศน์
นางสาวพัชมณฑ์ คเชนทร์ไพร

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

น้ำ เป็นสารประกอบที่พบมากถึง 3 ใน 4 ส่วนของพื้นโลก โดยส่วนใหญ่เป็นน้ำเค็มในทะเลและมหาสมุทรประมาณ 97 เปอร์เซ็นต์ เป็นน้ำแข็งตามขั้วโลกประมาณ 2 เปอร์เซ็นต์ และเป็นน้ำจืดตามแม่น้ำลำคลองต่างๆ ประมาณ 1 เปอร์เซ็นต์ ถ้าโลกเราปราศจากน้ำสิ่งมีชีวิตต่างๆ บนโลกก็จะไม่สามารถดำรงชีวิตอยู่ได้เลย และในปัจจุบันน้ำส่วนใหญ่ที่ คือ น้ำจืดที่ผ่านกระบวนการต่างๆมาแล้วหลายขั้นตอนเราใช้ในบ้านเรือนเรียกว่า " น้ำประปา "

น้ำประปานั้นมาจากแม่น้ำลำคลอง จากนั้นใช้เครื่องสูบน้ำเข้าคลองประปาตามที่โรงกรองน้ำต้องการสำหรับโรงกรองน้ำนั้นอยู่ตามจุดต่างๆ และต้องขยายเป็นหลายโรงเพื่อให้เพียงพอต่อปริมาณผู้ใช้ขั้นตอนการทำน้ำจากธรรมชาติให้เป็นน้ำประปาที่สะอาดนั้น ต้องผ่านการตกตะกอนก่อน จากนั้นจึงนำไปกรองสิ่งเจือปนมากับน้ำหลายชั้น จนกระทั่งนำไปฆ่าเชื้อก่อนำไปใช้ตามบ้าน ซึ่งน้ำประปาที่ส่งไป ถือว่ามีมาตรฐานพอที่จะใช้นำไปต้มดื่มได้เลย และในแต่ละบ้านต้องมิเตอร์น้ำไว้คอยตรวจวัดปริมาณน้ำที่ใช้แต่ละเดือนเพื่อนำไปคำนวณค่าน้ำและเพื่อใช้ในการแนวโน้มการใช้น้ำว่าจะมีใช้เพิ่มขึ้นหรือลดลงได้ด้วย

มิเตอร์น้ำที่ใช้อยู่ในปัจจุบันนั้นเราใช้กันมานานหลายสิบทำให้ไม่ทันสมัย คือเทคโนโลยีสมัยนี้นั้นต้องเป็นดิจิตอลและมีการส่งข้อมูลแบบไร้สายเพิ่มมากขึ้น จึงต้องมีการสร้างและออกแบบมิเตอร์น้ำให้เป็นแบบดิจิตอลเพื่อในอนาคตจะพัฒนาให้สามารถส่งข้อมูลแบบไร้สายหรือเก็บข้อมูลได้เป็นระยะเวลานานแต่ทั้งหมดนี้ยังต้องมีมาตรฐานการนับรอบเท่ากับของเดิม

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 สร้างวงจรมอเตอร์ไฟฟ้ากระแสตรงหมุนใบพัดแล้วนับรอบ แสดงรอบออกที่จอคอมพิวเตอร์

1.2.2 เขียนโปรแกรมเพื่อควบคุมมอเตอร์ด้วยภาษา C

1.2.3 สร้างวงจรเครื่องวัดรอบแบบดิจิตอลให้แสดงการนับรอบออกที่จอ LCD

1.2.4 เขียนโปรแกรมเพื่อควบคุมเครื่องวัดรอบแบบดิจิตอลด้วยภาษา C

1.2.5 เพื่อให้เครื่องวัดรอบสามารถนับรอบได้อย่างถูกต้อง

1.3 ขอบข่ายของโครงงาน

1.3.1 สร้างวงจรมอเตอร์กระแสตรงที่มีใบพัดให้นับรอบได้และแสดงออกทางคอมพิวเตอร์

1.3.2 สร้างวงจรเครื่องวัดรอบให้นับรอบได้และแสดงผลออกทางจอ LCD

1.3.3 ใช้ภาษา C ในการเขียนโปรแกรมให้มอเตอร์กับตัววัดรอบทำงานร่วมกันได้

1.3.4 ทำให้เครื่องวัดรอบกับมอเตอร์ทดสอบนับรอบได้เท่ากับ

1.4 ขั้นตอนของการดำเนินงาน

1.4.1 ศึกษาการทำงานของบอร์ด stk 200 และ chip AT90S8515 ร่วมถึงศึกษาการทำงานเกี่ยวตัววัดรอบ และศึกษาการเขียนโปรแกรมให้บอร์ดติดต่อกับมอเตอร์และตัววัดรอบได้

1.4.2 สร้างมอเตอร์กระแสตรงที่มีใบพัดให้นับรอบที่มี Opto sensor และมี LED บอกว่าหมุน

1.4.3 เขียนโปรแกรมควบคุมมอเตอร์ให้หมุนและสามารถนับจำนวนรอบการหมุนด้วยใบพัดได้ รวมถึงนำค่าแต่ละรอบแสดงออกมาทาง จอคอมพิวเตอร์

1.4.4 สร้างเครื่องวัดรอบแบบดิจิตอล ให้แสดงจำนวนรอบออกทาง LCD

1.4.5 เขียนโปรแกรมควบคุมเครื่องวัดรอบให้สามารถนับจำนวนรอบการหมุนได้ ร่วมถึงนำค่าแต่ละรอบแสดงออกมาทาง LCD

1.4.6 ทดสอบเครื่องวัดรอบเทียบกับรอบของใบพัดว่าจำนวนรอบในการหมุนนั้นเท่ากันหรือไม่

1.4.7 สรุปผลการทดสอบและบันทึกการทดสอบตามความเป็นจริง

1.5 แผนการดำเนินงาน

กิจกรรม	ปี 2548			ปี 2549		
	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1.ศึกษาและค้นคว้าข้อมูลเกี่ยวกับโปรแกรมภาษา C	←	→				
2. ค้นคว้าและศึกษาเกี่ยวกับมอเตอร์และเครื่องวัดรอบ		←	→			
3.เขียนโปรแกรมเพื่อทดสอบเครื่องวัดรอบ			←	→		
4. ทดสอบการทำงาน					← →	
5.สรุปผลการทดลองและจัดทำรูปเล่มโครงงาน					←	→

1.6 ผลที่คาดว่าจะได้รับ

1.6.1 มีความรู้ความเข้าใจหลักการของเครื่องวัดรอบและนำไปสู่การประยุกต์เป็นมิเตอร์น้ำในอนาคตหรือใช้ในส่วนที่เกี่ยวข้องกับการสิ่งที่มีความสามารถใกล้เคียงกันได้

1.6.2 มีความรู้ความเข้าใจเกี่ยวกับคุณสมบัติของ Board stk 200 และ ชิพ AT90S8515 ว่าเหมาะกับงานชนิดใด

1.6.3 สามารถเขียนโปรแกรมให้บอร์ดติดต่อสื่อสารกับมอเตอร์และเครื่องวัดรอบได้โดยผ่าน port RS232

1.7 งบประมาณของโครงงาน

1.7.1 ค่าถ่ายเอกสารและค่าเข้าเล่มโครงงาน 1,000 บาท

1.7.2 ค่ากระดาษ 1,000 บาท

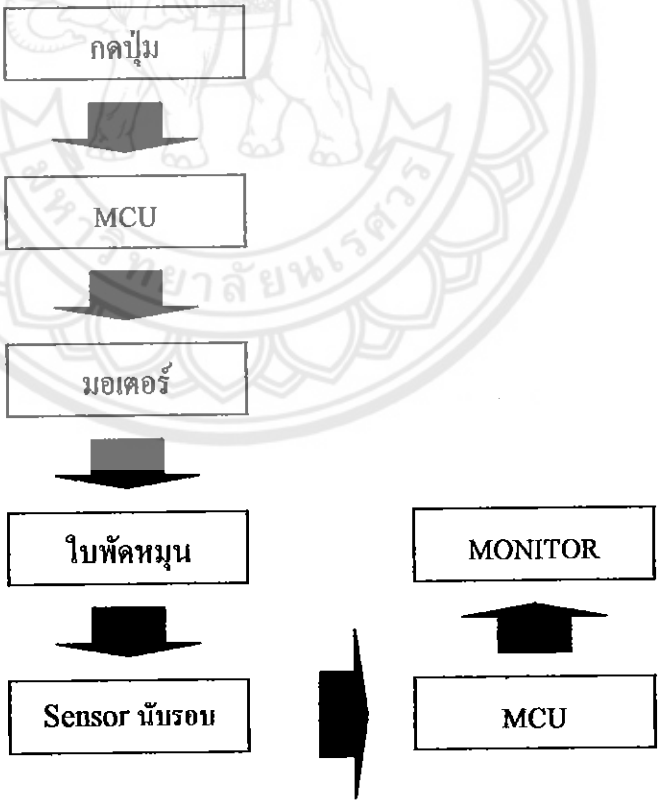
รวมเป็นเงิน 2,000 บาท (สองพันบาทถ้วน)

บทที่ 2
หลักการและทฤษฎีเบื้องต้น

บทนี้จะกล่าวถึงทฤษฎีและหลักการทำงานของ บอร์ด stk 200 ว่าบอร์ดนี้ทำงานอย่างไร มีคุณสมบัติอย่างไรซึ่งในบอร์ดนี้จะมีชิพ AT90S8515 ทำหน้าที่เป็น MCU ของบอร์ดรวมถึงอธิบายคุณสมบัติของชิพนี้ และจะอธิบายหลักการของมอเตอร์กับตัววัดรอบว่ามีการทำงานอย่างไรจะเขียนโปรแกรมอย่างไรให้ติดต่อสื่อสารกับบอร์ดได้ โดยการเขียนโปรมภาษา C

2.1 หลักการทำงานของมอเตอร์

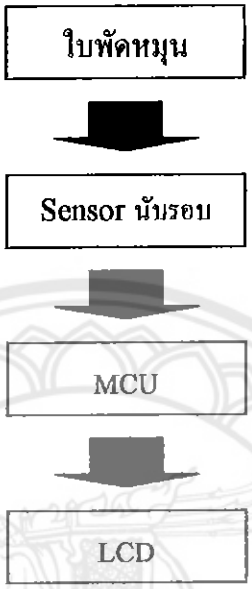
การควบคุมการทำงานของมอเตอร์นั้นจะแสดงไว้ที่รูปที่ 2.1 เป็นการทำงานโดยกดปุ่มสั่งให้มอเตอร์หมุน คำสั่งการหมุนจะถูกส่งไปที่ MCU ในบอร์ด stk 200 หรือ ชิพ AT90S8515 แล้ว MCU จะประมวลผลและสั่งการให้มอเตอร์หมุน ขณะที่มอเตอร์หมุน sensor จะเป็นตัวนับรอบคือใบพัดหมุนครบ 1 รอบจะทำการนับและเก็บค่าไว้บวกกันในรอบต่อๆ ไปและแสดงบวกออกมาที่จอคอมพิวเตอร์ทุกรอบที่ใบพัดหมุน



รูปที่ 2.1 แสดงหลักการทำงานของมอเตอร์

2.2 หลักการทำงานของตัววัดรอบ

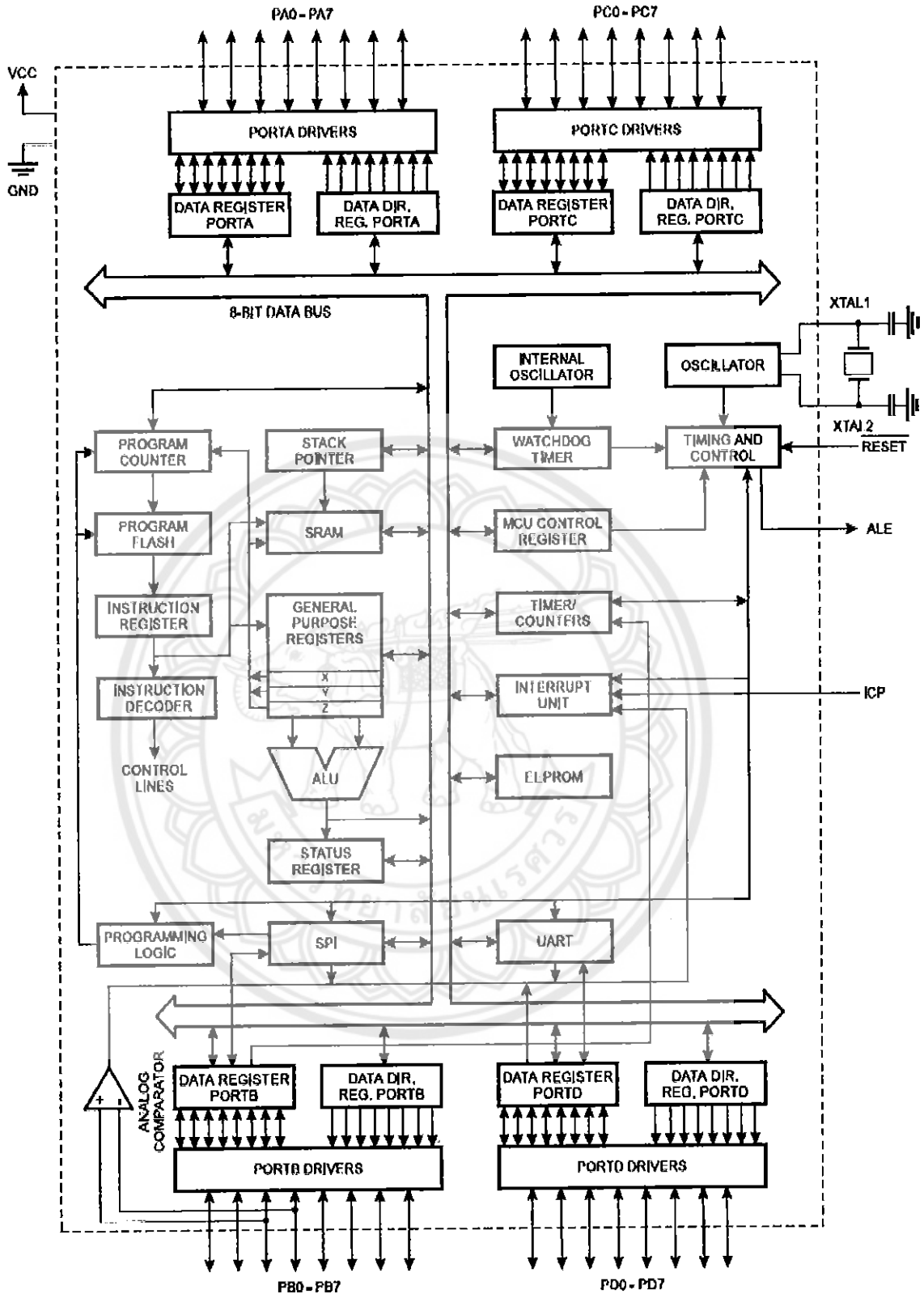
การควบคุมการทำงานของตัววัดรอบจะแสดงไว้ดังรูปที่ 2.2 เมื่อใบพัดหมุน sensor (คนละตัวกับขอมอเตอร์) จะเป็นตัวนับรอบคือใบพัดหมุนครบ 1 รอบจะทำการนับและเก็บค่าไว้บวกกันในรอบต่อๆ ไปและแสดงบวกออกมาที่จอ LCD ทุกรอบที่ใบพัดหมุน



รูปที่ 2.2 แสดงหลักการของตัววัดรอบ

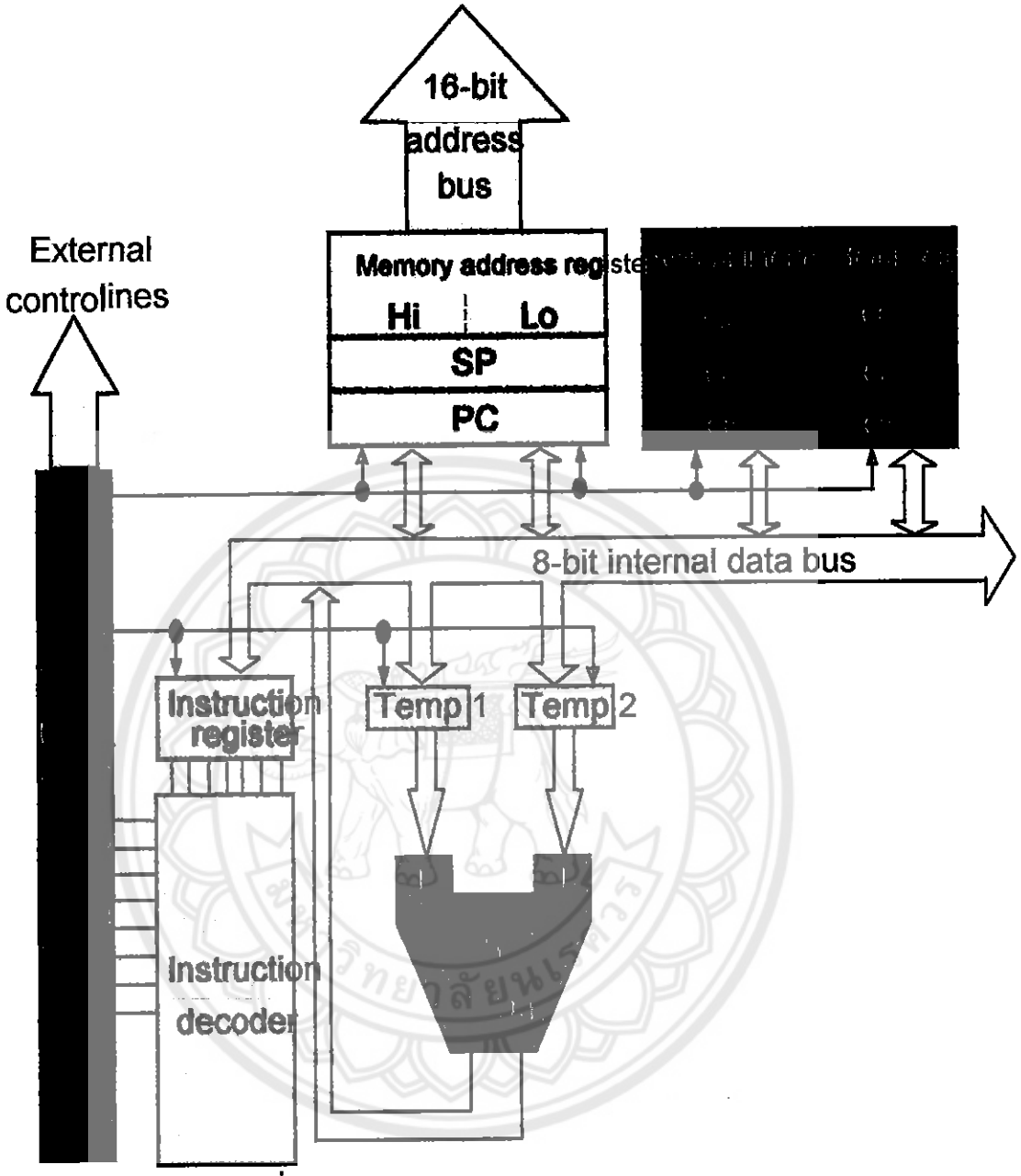
2.3 คุณสมบัติของการใช้งาน Board stk 200

ไมโครคอนโทรลเลอร์ที่มีสถาปัตยกรรมภายในถูกออกแบบให้ใช้สถาปัตยกรรมแบบ RISE (Reduce Instruction Set Computer) ดังแสดงในรูปที่ 2.3 คือ ทำให้การประมวลผลมีความเร็ว 1 คำสั่ง / 1 Clock หรือ CPU สามารถประมวลคำสั่งได้ 1 MIPS / MHz มีคำสั่งในการควบคุมการทำงานของไมโครคอนโทรลเลอร์จำนวน 118 คำสั่ง หน่วยความจำแบบ RAM ขนาด 512 Byte SRAM EEPROM มีรีจีสเตอร์ใช้งานทั่วไปขนาด 8 บิต จำนวน 32 ตัว พอร์ตอินพุตและเอาท์พุตขนาด 8 บิต จำนวน 4 พอร์ตสามารถใช้ ความถี่สัญญาณนาฬิกา สูงถึง 10 MHz ระบบการรีเซตแบบอัตโนมัติเมื่อเริ่มจ่ายกระแสไฟฟ้าเข้าไมโครคอนโทรลเลอร์(Power on reset) มีระบบการกำเนิดความถี่สัญญาณแบบ Pulse Width Modulator มีระบบการตรวจจับระดับสัญญาณอะนาลอก(Analog Comparator) ระบบการป้องกันการ COPY ข้อมูลภายในหน่วยความจำ(LOCK FOR SOLFWARE SECURITY) ระบบการอินเตอร์รัปต์จากภายนอก(EXTERNAL INTERRUPT) TIMER/COUNTER ขนาด 16 บิต และ 8 บิต



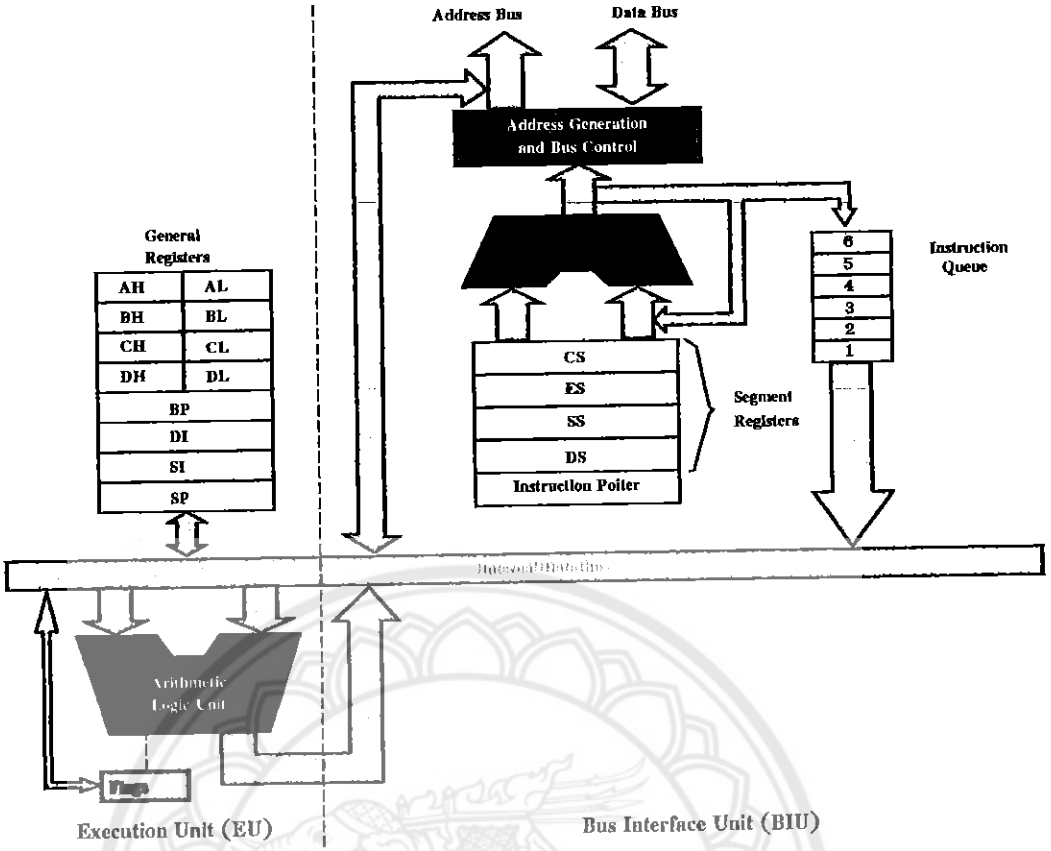
รูปที่ 2.3 Block diagram stk 200

บอร์ด stk 200 มีส่วนประกอบสำคัญ 3 คือ ALU(Arithmetic Logic Unit), Register และ Control Logic



รูปที่ 2.4 แสดงส่วนประกอบสำคัญของ board stk 200

หรืออาจแบ่งออกเป็น BIU(Bus Interface Unit) กับ EU.(Execution Unit)



รูปที่ 2.5 การทำงานระหว่าง ALU กับ registor

หน้าที่และหลักการทำงานของ ALU

จากรูปที่ 2.5 การทำงานระหว่าง ALU กับ registor นั้น ALU เป็น Major Logic function ของไมโครโปรเซสเซอร์ทุกตัวโดยทั่วไป ALU จะมี 2 Input และ 1 Output โดย Input ทั้ง 2 จะมี Buffer ซึ่งก็คือ Temporary Register จะเป็นตัวเก็บข้อมูลให้กับ ALU และยังต่ออยู่กับ Internal Bus เพื่อรับหรือส่งข้อมูลให้กับ ALU โดยผ่าน Temporary Register ทั้ง 2 โดยที่ ALU จะรับข้อมูลจาก Accumulator Register ALU จะส่งข้อมูลเป็น Word ไปเก็บที่ Accumulator Register หลังจากที่มีการกระทำตาม Logic Function ต่างๆ เสร็จเรียบร้อยแล้วหน้าที่ที่ ALU จะกระทำมีดังนี้

Add	Complement
Subtract	Shift right
AND	Shift lift
OR	Increment
Exclusive OR	Decrement

จำไว้ว่า ALU เป็นส่วนที่ใช้ในการ Process Data ไม่ใช่เก็บ Data

หน้าที่และหลักการทำงานของ Registers

จากรูปที่ 2.5 การทำงานระหว่าง ALU กับ registor ซึ่ง Registers เป็นส่วนที่ต้องใช้ในไมโครโปรเซสเซอร์ ทุกตัวนั้นเป็น Function หลักที่จะต้องมีอาจจะมีมากกว่าหนึ่งตัว Register ที่ไมโครโปรเซสเซอร์ทุกตัวจะต้องมี คือ

Accumulator Register

Program Counter Register

Stack Pointer Register

General-purpose Register

Memory address Register and logic

Instruction Register

Temporary data Register

Accumulator Register

Accumulator Register เป็น Register หลักของไมโครโปรเซสเซอร์ที่ใช้ในการเก็บและการเคลื่อนย้ายข้อมูล เช่น การคำนวณทางคณิตศาสตร์และลอจิกซึ่งต้องใช้ข้อมูลทั้ง ALU และ Accumulator ในการทำงานจะถูกแบ่งออกเป็น 2 Word โดยที่ Word แรกจะเก็บไว้ใน Accumulator Register และอีก Word หนึ่ง อาจจะเก็บที่ Register อื่นหรือในหน่วยความจำ นอกจากนี้ผลที่ได้จาก ALU จะเก็บไว้ใน Accumulator

การทำงานที่ใช้ Accumulator คือ การเคลื่อนย้ายข้อมูลจากที่หนึ่งในไมโครโปรเซสเซอร์ไปยังที่อื่น เช่น การย้ายข้อมูลระหว่าง I/O Port กับหน่วยความจำ หรือระหว่าง หน่วยความจำกับอุปกรณ์อื่น โดยมีการทำงานดังนี้คือ ข้อมูลจะถูกย้ายจาก Source ไปยัง Accumulator จากนั้นจึงจะย้ายไปยัง Destination

Program Counter Register

Program Counter Register เป็น Register อีกตัวหนึ่งที่มีความสำคัญต่อไมโครโปรเซสเซอร์ ซึ่งคำสั่งที่ใช้ในการทำงานจะมีการเรียงลำดับคำสั่งตามโปรแกรมที่เขียน โดยเก็บไว้ในหน่วยความจำของไมโครโปรเซสเซอร์และคำสั่งเหล่านี้จะเป็นตัวบอกให้ไมโครโปรเซสเซอร์ทำงานได้อย่างเที่ยงตรง

Program Counter คือ เก็บข้อมูลว่าคำสั่งใดเป็นคำสั่งเริ่มต้นและคำสั่งใดเป็นคำสั่งต่อไป โดยทั่วไป Program Counter จะมีขนาดมากกว่าขนาดข้อมูลของไมโครโปรเซสเซอร์ เช่น

ไมโครโปรเซสเซอร์มีขนาดข้อมูล 8 บิต ส่วน Program Counter จะมีขนาด 16 บิต ในส่วนการทำงานของโปรแกรมสามารถเริ่มและจบการทำงานจาก Memory address ระหว่าง 0-64 Kb โดยที่ Program Counter จะเริ่มต้นที่ตำแหน่ง 0 จากนั้นจะทำการเพิ่มค่าครั้งละ 1 เพื่อชี้ตำแหน่งต่อไปของคำสั่ง นอกจากนี้ Program Counter สามารถทำงานอื่นนอกเหนือจากลำดับการทำงานซึ่งในส่วนนี้เราจะเรียกว่าเป็นการเข้าไปทำงานใน "Subroutine" และกลับมายังลำดับการทำงานต่อไป

Stack Pointer Register

Stack Pointer Register เป็น Register ที่แสดงความแตกต่างระหว่างการคำนวณ การตรวจสอบผลของข้อมูลโดยจะใช้บิตแต่ละบิต ซึ่งหมายถึง Status แต่ละตัว โดยที่ Logic "0" หมายถึง ไม่มีการทำงาน และ "1" หมายถึง Set ให้เป็นตามเงื่อนไขของแต่ละบิต ดังนี้

1. Carry/Borrow bit หมายถึงการบวกเลขสองจำนวน

2. Zero bit บิตนี้จะเป็น "1" เมื่อมีการตรวจสอบค่าของตัวแปลที่ลดลงแล้วมีค่าเท่ากับ "0" โดยปกติ บิตนี้จะเป็น "1" ตลอด

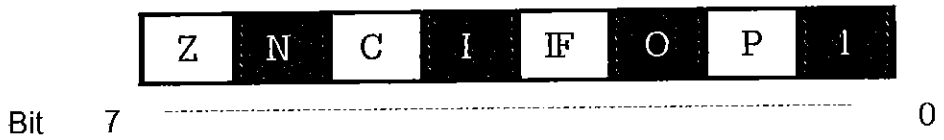
3. Negation bit บิตนี้จะใช้ในการแสดงให้รู้ว่าเลขที่ได้นั้นมีค่าเป็นลบหรือไม่ ซึ่งบิตนี้เรียกอีกอย่างหนึ่งว่า " บิตเครื่องหมาย (Signed bit)"

4. Intermediate Carry bit บิตนี้จะเป็นบิตที่แสดงถึงการบวกเลขของ 4 บิตแรกแล้วมีตัวทดเกิดขึ้นนอกจากนี้แล้วยังใช้ในการแปลงเลขฐาน จากรหัส BCD เป็นเลขฐานสอง

5 . Interrupt bit บิตนี้จะกำหนดได้จากการเขียนโปรแกรมควบคุม ซึ่งสามารถที่จะ Enable/Disable Interrupt ได้

6. Overflow bit บิตนี้แสดงถึงการทำงานทางคณิตศาสตร์ แล้วเกิดมีการทดพร้อมกับมีค่าของผลลัพธ์ที่ได้เป็นลบ หรือเกิดจากการทำ 2's Complement

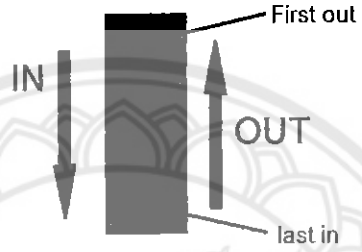
7. Parity bit บิตนี้จะแสดงถึงการตรวจสอบข้อมูลแล้วพบว่าเป็นข้อมูลเลขคี่ (คือนับจำนวนของลอจิก "1" แล้วได้เป็นเลขคี่) เช่น "10011000" Parity bit จึงจะให้ค่าเป็น "1"



รูปที่ 2.6 แสดง Status Register

General-purpose Register

General-purpose Register เป็นตัวชี้ตำแหน่งของการเก็บข้อมูลโดยลักษณะการเก็บจะเป็นแบบ Last in first out หรือ เข้าทีหลังออกก่อนซึ่งลักษณะจะเหมือนกับ Program Counter แต่จะต่างกันที่ Program Counter จะเป็นการเพิ่มค่า แต่ Stack Pointer จะเป็นการลดค่าลงและจะทำการกำหนดค่าโดยอัตโนมัติ และใช้จำนวน Byte ในการทำงาน 2 byte ซึ่งหมายถึง Stack Pointer จะลดค่าของ Address ครั้งละ 2 byte หรือ ลดตำแหน่งลง 2 ตำแหน่ง เช่น Stack ครั้งแรกอยู่ที่ Address $0FF4_H$ ตำแหน่ง Stack ครั้งต่อไปคือ $0FF4_H$ และนอกจากนี้ ผู้ใช้ยังสามารถกำหนด ตำแหน่งเริ่มต้นได้แต่ถ้าไม่มีการกำหนดตำแหน่งเริ่มต้น Stack จะทำการสุ่มตำแหน่งให้โดยอัตโนมัติ



รูปที่ 2.7 แสดง Stack Pointer Register

General -Purpose Register เป็น Register ที่ใช้งาน คือ B C D E H และ L และยังสามารถเรียกใช้งานรวมกันได้ เช่น BC หรือ DE ซึ่งการใช้งานจะต้องเป็นการใช้งานขนาด 16 bit ด้วย

Memory address Register and logic

Memory address Register and logic เป็นตัวแสดงตำแหน่งหน่วยความจำของไมโครโปรเซสเซอร์ ที่ต้องการใช้ โดยจะเก็บตำแหน่งของหน่วยความจำไว้ในรูปของเลขฐานสองขนาด 16 byte หรือมากกว่าขึ้นอยู่กับขนาดของ Address bus ซึ่งใช้เป็นตัวเลือกตำแหน่งหน่วยความจำหรือในกรณีของ input/output post ในการ fetch cycle จากคำสั่งต่างๆ ที่มีในหน่วยความจำเป็นตัวบอกให้ Memory Address Register และ Program Counter เพื่อบอกตำแหน่งของคำสั่งและข้อมูล โดย Memory Address Register จะเป็นตัวชี้ตำแหน่งของคำสั่งที่จะทำการ fetch จากหน่วยความจำ และเมื่อมีการ Decode แล้ว Program Counter จะทำการเพิ่มค่าอีก 1 เพื่อชี้คำสั่งต่อไปซึ่ง Memory Address Register จะไม่มีการเพิ่มหรือลดค่าแต่จะใช้เป็นตัวบอกตำแหน่งของข้อมูลที่ชี้โดย Program Counter

ใน 16-bit Memory Address Register จะแบ่ง Register เป็น 2 ส่วน คือ High byte กับ Low byte และในบางคำสั่งที่มีการกำหนดค่าจากการคำนวณ โดยคำนวณจากค่าของ Program Counter บวกหรือลบกับค่าของคำสั่ง ซึ่งในกลุ่มนี้เราจะเรียกว่าเป็น "Offset Addressing"

Instruction Register

Instruction Register เป็น Register ที่เก็บคำสั่งที่กำลังทำการ Execute และจะ Load การทำงานขึ้นอยู่กับการ fetch Execute Cycle ซึ่งรวมเรียกว่า "Instruction Cycle"

ในการทำงานเมื่อ fetch คำสั่ง Program Counter จะชี้คำสั่งต่อไปที่อยู่ในหน่วยความจำเมื่อ Instruction Register ทำการ fetch คำสั่งแล้วจะทำการนำคำสั่งนั้นไปไว้ที่ Internal Data bus และขณะกำลัง Exccute อยู่นั้นในส่วนของ Instruction Decoder จะอ่านรายละเอียดของ Instruction Register Instruction Decoder จะทำการ Decoder คำสั่งในรายการเพื่อบอกให้ Microprocessor ทำงานและส่งคำสั่งออกมา จากนั้นทำการส่งไปยังส่วนของ Microprocessor Control logic ต่อไป เพื่อส่งไปควบคุมการทำงานของส่วนต่างๆ

Temporary Data Register

Temporary Data Register เป็น Register ที่เก็บข้อมูลได้เพียง 1 Word และเป็นส่วนสำคัญของไมโครโปรเซสเซอร์ กล่าวคือถ้าไม่มี Temporary Data Register ALU จะไม่สามารถทำงานได้

หน้าที่และหลักการทำงานของ Control Logic

Control Logic เป็นส่วนหลักอีกส่วนหนึ่งของไมโครโปรเซสเซอร์ คือ เป็นการรวมงานต่างๆ เข้าด้วยกันอย่างถูกต้องตามลำดับซึ่งส่วนของ Control logic นั้น เรียกอีกอย่างหนึ่งว่าเป็น "Micro program" ซึ่งจะมีการเรียนรู้คำสั่งการ Decode คำสั่งที่ส่งมาจาก Instruction Register และจะส่งสัญญาณควบคุมของคำสั่งออกไปยังส่วนต่าง ๆ ที่คำสั่งนั้นมีการเรียกใช้งาน

Internal Data bus เป็นระบบ bus ที่ใช้ติดต่อภายในไมโครโปรเซสเซอร์จะมีขนาดเท่าใดขึ้นอยู่กับไมโครโปรเซสเซอร์ด้วยว่า มีขนาดของ Data bus เท่าใดเป็น 8 บิต, 16 บิต หรือ 32 บิต เป็นต้น ซึ่งชนิดของ Internal bus จะเป็นแบบ 2 ทิศทาง (Bi-directional bus)

2.4 คุณสมบัติของชิพเบอร์ AT90S8515

จากรูปที่ 2.8 แสดงคุณสมบัติของขาต่างๆ ในชิพเบอร์ AT90S8515 ซึ่งมีขาทั้งหมด 40 ขา รายละเอียดต่างๆ ประจำขาต่างมีดังนี้

2.4.1 Pin Descriptions

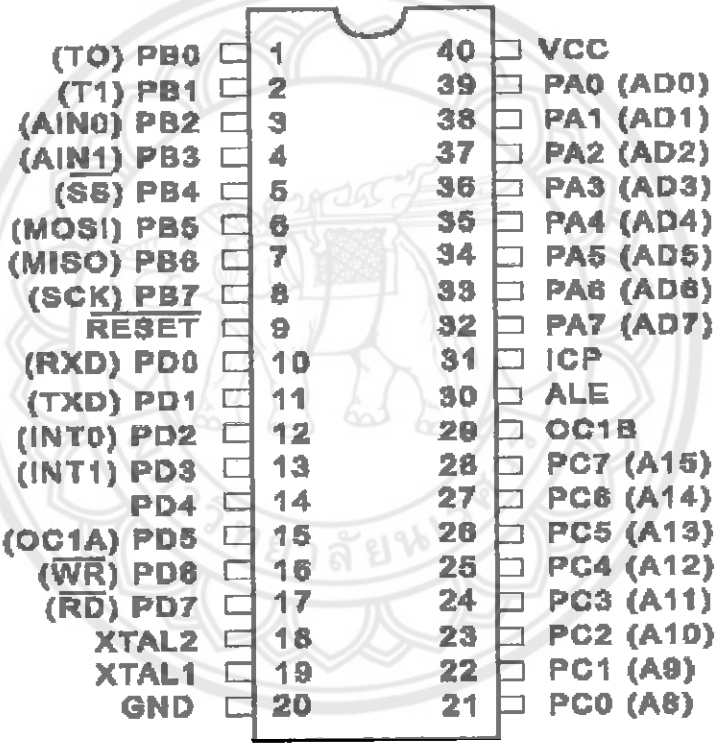
ในชิพเบอร์ AT90S8515 นี้ มีทั้งหมด 40 ขา แต่ขาอาจมีหลายหน้าแล้วแต่จะใส่ค่า registor

2.4.2 VCC

ไฟเลี้ยง จะอยู่ขาที่ 40 จะใช้ไฟ 2.7 – 6.0 v

2.4.3 GND ground.

จะอยู่ที่ขา 20



รูปที่ 2.8 แสดงคุณสมบัติของขาต่างๆ ในชิพเบอร์ AT90S8515

2.4.4 Prot A (PA7-PA0) มีขนาด 8 บิต

Port A สื่อสารได้สองทิศทาง สามารถเลือกได้ว่าจะพูลอัพหรือพูลดาวน์ registor ภายในหรือไม่ สามารถเลือกขาใดขาหนึ่งก็ได้ เมื่อ Port A ส่งลอจิก 1 จะสามารถ source กระแสได้ 20 mA ดังนั้น สามารถขับ LED ได้โดยตรงและ Port A สามารถรับกระแสได้ 20 mA และ Port A สามารถแปลง

analog inputs เป็น digital ขนาด ความละเอียด 10 บิต ได้ 8 ช่อง พร้อมทั้งเลือกได้ว่าจะกำจัดสัญญาณรบกวนหรือไม่

2.4.5 Prot B (PB7-PB0) มีขนาด 8 บิต

Port B (PB7-PB0) Port B มีขนาด 8 บิต สื่อสารได้สองทิศทาง input/output อย่างใดอย่างหนึ่ง สามารถเลือกได้ว่าจะพูลอัพ registor ภายในหรือไม่ สามารถเลือกขาใดขาหนึ่งก็ได้ เมื่อ port B ส่งลอจิก 1 จะสามารถ source กระแสได้ 20 mA ดังนั้นสามารถขับ LED ได้โดยตรง และ Port B สามารถรับกระแสได้ 20 mA และสามารถเลือกใช้งานในด้าน Specific ต่างๆ ได้

2.4.6 Port C (PC7-PC0) มีขนาด 8 บิต

Port C สื่อสารได้สองทิศทาง input/output สามารถเลือกได้ว่าจะพูลอัพหรือพูลดาวน์ registor ภายในหรือไม่ สามารถเลือกขาใดขาหนึ่งก็ได้ เมื่อ port C ส่งลอจิก 1 จะสามารถ source กระแสได้ 20 mA ดังนั้นสามารถขับ LED ได้โดยตรง และ Port C สามารถรับกระแสได้ 20 mA การ พูลดาวน์ นั้นต้องทำภายนอก Port C สามารถใช้งานเป็น Timer/Counter2.

2.4.6 Port D (PD7-PD0) มีขนาด 8 บิต

Port D สื่อสารได้สองทิศทาง input/output สามารถเลือกได้ว่าจะพูลอัพหรือพูลดาวน์ registor ภายในหรือไม่ สามารถเลือกขาใดขาหนึ่งก็ได้ เมื่อ port D ส่งลอจิก 1 จะสามารถ source กระแสได้ 20 mA ดังนั้นสามารถขับ LED ได้โดยตรง และ Port D สามารถรับกระแสได้ 20 mA Port D การพูลดาวน์นั้น ต้องทำภายนอก Port D นั้นสามารถเลือกใช้งานแบบ specific ได้

2.4.7 RESET

ที่ขา RESET ทำงานที่ลอจิก "0" โดยต้องให้เวลาไม่น้อยไปกว่า 2 Machine cycle จึงจะรีเซ็ตสมบูรณ์ ซึ่งการรีเซตจะมีสองแบบ คือ รีเซตครั้งแรกเมื่อเริ่มจ่ายไฟ และหากจะรีเซตอีกครั้ง มาจากสองแหล่งคือ เมื่อมีการใช้ watchdog และการต่อ switch reset นอก

2.4.8 AVCC

เป็นไฟเลี้ยงวงจรแปลงสัญญาณ A/D Converter หากไม่ได้ใช้งาน ขานี้ให้ต่อเข้ากับ VCC เพื่อเป็น lowpass Filter

2.4.9 AREF

เป็นขา voltage Referrent สำหรับเลือกช่วงของ Analog voltage REF ในช่วง 2 v to AVCC

2.4.10 AGND Analog ground

กรณีไม่ได้แยก gnd ก็ให้ต่อกับ gnd ของ mcu แสดงขาของ 90S8515

2.5 การเซตรีจิสเตอร์ DDR ในการเขียนโปรแกรมลงบอร์ด stk 200

ตารางที่ 2.1 ตารางตัวอย่างการเซตค่ารีจิสเตอร์

	ทิศทางของแต่ละบิต หลังจากเซตค่า DDRa,DDRb,DDRc,DDRd								
DDR	ค่า	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
DDRa	ffh	out	out	out	out	out	out	out	out
DDRa	f0h	out	out	out	out	in	in	in	in
DDRa	00h	in	in	in	in	in	in	in	in
DDRb	ffh	out	out	out	out	out	out	out	out
DDRb	f0h	out	out	out	out	in	in	in	in
DDRb	00h	in	in	in	in	in	in	in	in
DDRc	ffh	out	out	out	out	out	out	out	out
DDRc	f0h	out	out	out	out	in	in	in	in
DDRc	00h	in	in	in	in	in	in	in	in
DDRd	ffh	out	out	out	out	out	out	out	out
DDRd	f0h	out	out	out	out	in	in	in	in
DDRd	00h	in	in	in	in	in	in	in	in

จากตารางที่ 2.1 แสดงตารางตัวอย่างการเซตค่ารีจิสเตอร์ การใช้ port ใดๆ ก็ตาม ถ้าต้องการให้เป็น input ต้องเซตค่าประจำบิตของ DDR เป็น 0 ถ้าต้องการให้เป็น output ต้องเซตค่าเซตค่าประจำบิตของ DDR เป็น 1

เช่น ตัวอย่างตารางด้านบน

หากให้ DDRb = 0xff ทุกๆบิตของ Port b จะเป็น output หมด

หากให้ DDRa = 0xff ทุกๆบิตของ Port a จะเป็น output หมด

หากให้ DDRb = 0xf0 สี่บิตบนของ Port b จะเป็น output หมด

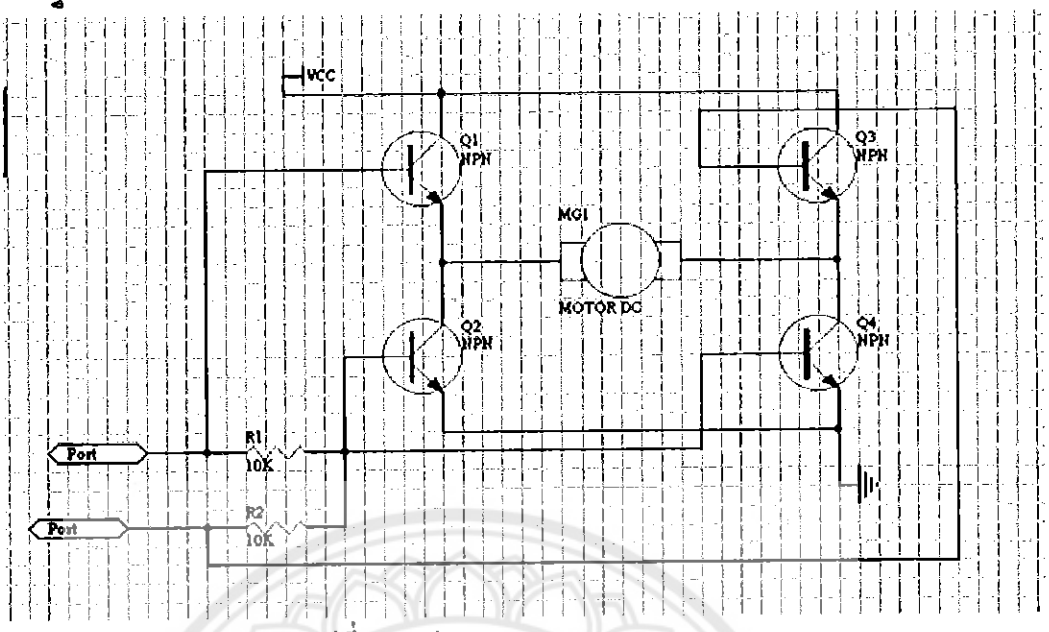
สี่บิตล่างของ Port b จะเป็น input หมด

2.5.1 Register Summary ที่ใช้กับงานต่างๆ

ตารางที่ 2.2 ตาราง รีจิสเตอร์ ต่างๆ

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C
$3E ($5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
$3C ($5C)	Reserved								
$3B ($5B)	GIMSK	INT1	INT0	-	-	-	-	-	-
$3A ($5A)	GIFR	INTF1	INTF0						
$39 ($59)	TIMSK	TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	-
$38 ($58)	TIFR	TOV1	OCF1A	OCF1B	-	ICF1	-	TOV0	-
$37 ($57)	Reserved								
$36 ($56)	Reserved								
$35 ($55)	MCUCR	SRE	SRW	SE	SM	ISC11	ISC10	ISC01	ISC00
$34 ($54)	Reserved								
$33 ($53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00
$32 ($52)	TCNT0	Timer/Counter0 (8 Bit)							
...	Reserved								
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10
$2E ($4E)	TCCR1B	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10
$2D ($4D)	TCNT1H	Timer/Counter1 - Counter Register High Byte							
$2C ($4C)	TCNT1L	Timer/Counter1 - Counter Register Low Byte							
$2B ($4B)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte							
$2A ($4A)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte							
$29 ($49)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte							
$28 ($48)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte							
...	Reserved								
$25 ($45)	ICR1H	Timer/Counter1 - Input Capture Register High Byte							
$24 ($44)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte							
...	Reserved								
$21 ($41)	WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0
$20 ($40)	Reserved								
$1F ($3F)	EEARH[1]	-	-	-	-	-	-	-	EEAR8
$1E ($3E)	EEARL	EEPROM Address Register Low Byte							
$1D ($3D)	EEDR	EEPROM Data Register							
$1C ($3C)	EECR	-	-	-	-	-	EEMWE	EEWE	EERE
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
$0F ($2F)	SPDR	SPI Data Register							
$0E ($2E)	SPSR	SPIF	WCOL	-	-	-	-	-	-
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
$0C ($2C)	UDR	UART I/O Data Register							
$0B ($2B)	USR	RXC	TXC	UDRE	FE	OR	-	-	-
$0A ($2A)	UCR	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8
$09 ($29)	UBRR	UART Baud Rate Register							
$08 ($28)	ACSR	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
...	Reserved								
$00 ($20)	Reserved								

2.6 พื้นฐาน STEPPING MOTOR



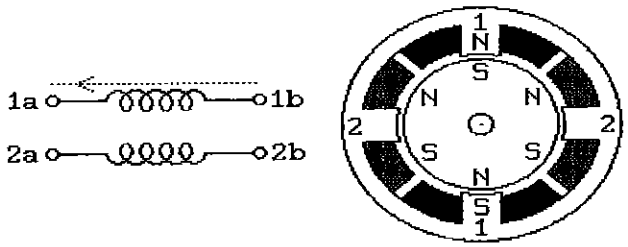
รูปที่ 2.9 รูปวงจร STEPPING MOTOR

จากรูปที่ 2.9 แสดงรูปวงจร STEPPING MOTOR การบังคับให้ DC Motor ทำงาน มีอยู่ 2 แนวทางที่นิยมใช้กันคือ การใช้ Relay และการใช้ Transistor ต่อกันเป็น H-Bridge ซึ่งมีลักษณะคล้ายกัน ในที่นี้จะขอกล่าวเพียงลักษณะของวงจร H-Bridge โดยทั่วไปหากต้องการให้ Motor หมุนตามเข็มสามารถทำได้โดยให้ A เป็น 1 และ B เป็น 0 และหากต้องการให้หมุนกลับทิศก็จะใช้ B เป็น 1 และ A เป็น 0 ส่วนที่ A และ B เป็น 1 หรือเป็น 0 ทั้งคู่จะไม่มีการหมุนเกิดขึ้น Transistor ทั้ง 4 ตัวจะทำหน้าที่เป็น Switch เปิดปิดสลับไปมา ในการควบคุมความเร็วของ DC นิยมใช้การจ่ายสัญญาณ เป็น PWM โดยหากต้องการให้หมุนเร็วก็จะจ่ายที่ Pulse ที่ความถี่สูง และหากต้องการให้หมุนช้าก็จะจ่าย Pulse ความถี่ต่ำ อย่างไรก็ตามการควบคุมความเร็วให้คงที่จะทำได้ค่อนข้างยากเว้นแต่ว่าจะใช้ระบบควบคุมที่มีการป้อนกลับเข้ามาช่วย(Feed back control)

2.6.1 Stepper motor

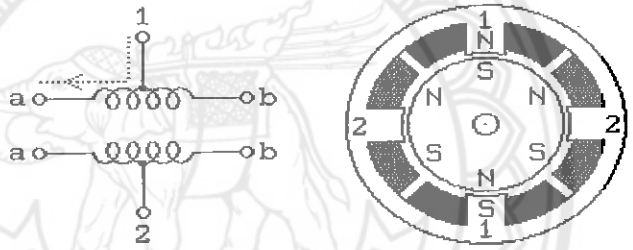
Stepper motor แบ่งออกได้เป็น 2 แบบตามลักษณะของโครงสร้างคือ Bipolar Stepper motor และ Unipolar Stepper motor ซึ่งทั้งสองแบบจะแตกต่างกันตามลักษณะของโครงสร้างภายใน อย่างไรก็ตามหลักในการขับให้ Stepper motor ทั้งสองแบบทำงานจะคล้ายคลึงกัน คือการป้อน Pulse เป็นช่วงๆ เข้าไปยังส่วนต่างๆ เพื่อให้ stepper motor หมุนตามจำนวนองศาที่ต้องการ

ปกติ Stepper motor แบบ Bipolar จะมีราคาถูกและสามารถหาได้ง่ายกว่า(ตัวอย่างเช่น Stepper ที่ใช้ขับ Floppy Drive) การจะหมุนก็ทำได้โดยการใส่ Pulse เข้าไปที่ Coil ในกรณีนี้อาจจะต้องใช้วงจร H-Bridge เข้ามาช่วยในลักษณะเดียวกันกับ DC Motor



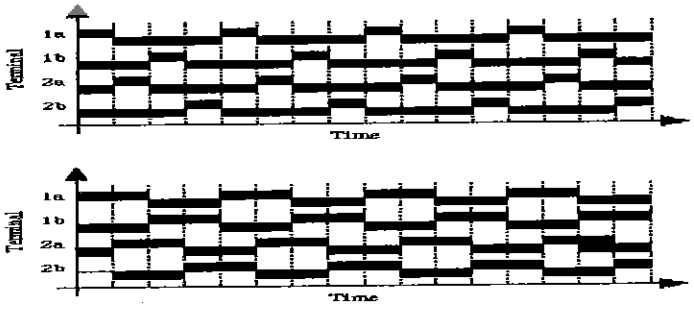
รูปที่ 2.10 Bipolar stepper motor layout (diagram by Douglas W. Jones, used with permission)

Stepper motor แบบ Unipolar จะมีสายสัญญาณทั้งสิ้น 6 ขา ซึ่งข้อดีคือผู้ใช้ไม่จำเป็นต้องต่อวงจร H-Bridge เพื่อขับวงจรดังกล่าว แสดงไว้ดังรูปที่ 2.11 Unipolar stepper motor layout



รูปที่ 2.11 Unipolar stepper motor layout (diagram by Douglas W. Jones, used with permission)

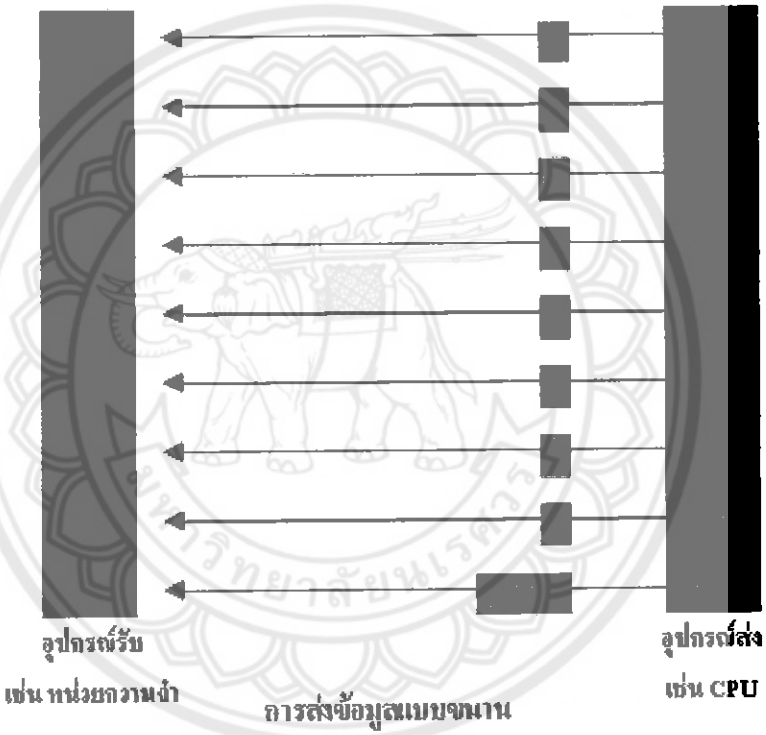
Sequence ทั้ง 2 อันต่อไปนี้ใช้สำหรับขับวงจร Stepper motor ทั้งสองแบบ แต่ Sequence ที่ 2 จะใช้ค่าของ Torque(หรือแรงบิด) สูงกว่า โดยสีแดงคือ VCC และ สีน้ำเงินคือ Ground ดังตารางที่ 2.3 แสดงตารางเปรียบเทียบสอง sequence



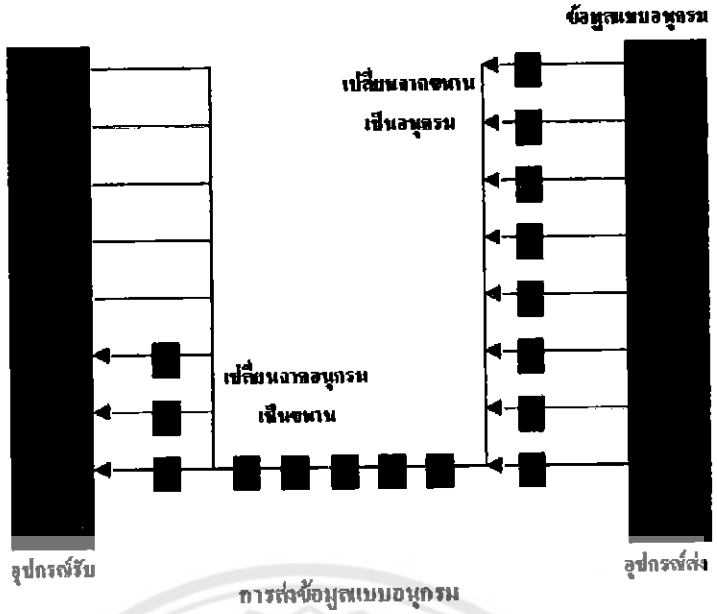
ตารางที่ 2.3 แสดงตารางเปรียบเทียบสอง sequence

2.7 การสื่อสารข้อมูลอนุกรมจากคอมพิวเตอร์ไปไมโครคอนโทรลเลอร์

การสื่อสารข้อมูลอนุกรมเป็นการรับส่งข้อมูลลักษณะของบิตหรือกลุ่มของบิต คราวละหนึ่งบิต เป็นลำดับเรื่อยไปจนสิ้นสุด การสื่อสารแบบนี้มีข้อแตกต่างจากการสื่อสารแบบขนานเป็นอย่างมาก เนื่องจาก ข้อมูลมีการโอนย้ายมาพร้อมกันจึงมีความจำเป็นต้องใช้จำนวนเส้นสัญญาณมากขึ้นตาม จำนวนบิตของข้อมูลด้วยในขณะที่การสื่อสารแบบอนุกรมนั้น ต้องการเส้นสัญญาณเพียงสองเส้นหรือ สามเส้นเท่านั้น ดังนั้นการสื่อสารแบบขนานจึงไม่เหมาะสมในการสื่อสารกับอุปกรณ์ภายนอกเป็น ระยะทางไกลๆ เพราะจะทำให้สิ้นเปลืองค่าใช้จ่ายมาก ลองพิจารณาเปรียบเทียบการสื่อสารทั้งสอง ประเภทได้จากรูป 2.12 แสดงการส่งข้อมูลแบบขนาน และรูปที่ 2.13 แสดงการส่งข้อมูลแบบอนุกรม



รูปที่ 2.12 แสดงการส่งข้อมูลแบบขนาน



รูปที่ 2.13 แสดงการส่งข้อมูลแบบอนุกรม

การจัดการข้อมูลอนุกรมของไมโครคอนโทรลเลอร์

พอร์ตอนุกรมของ 8051 มีโครงสร้างการทำงานในแบบที่เรียกว่าฟูลดูเพล็ก(Full Duplex) ซึ่งหมายถึงความสามารถในการรับและส่งข้อมูลอนุกรมในเวลาเดียวกัน จากรูป 2.12 แสดงให้เห็นถึงแผนการทำงานอย่างง่ายของวงจรส่วนจัดการข้อมูลของไมโครคอนโทรลเลอร์ โดยทางด้านวงจรตัวส่งประกอบด้วยรีจีสเตอร์ SBUF ทำหน้าที่เก็บข้อมูลที่จะส่งออก การใช้คำสั่งเขียนหรือโอนย้ายข้อมูลมายังรีจีสเตอร์นี้ จะเป็นการส่งข้อมูลนั้นออกไปยังพอร์ตอนุกรมทางขาสัญญาณ TxD โดยอัตโนมัติ ส่วนวงจรด้านตัวรับ ประกอบด้วยรีจีสเตอร์ SBUF เช่นเดียวกันแต่ทำหน้าที่เก็บข้อมูลที่นำมาจากส่วนของวงจรเลื่อนบิตหรือชิพรีจีสเตอร์ของวงจรจัดการข้อมูลอนุกรมภายในสัญญาณข้อมูลอนุกรมที่รับเข้าจะผ่านทางสัญญาณ RxD

2.8 UART ทำให้คอมพิวเตอร์กับไมโครคอนโทรลเลอร์

UART ย่อมาจาก Universal Asynchronous Receiver - Transmitter อุปกรณ์ที่ต่ออนุกรม(serial devices) ทุกชนิดใช้อินเตอร์เฟสชิพ UART ในการติดต่อสื่อสาร กับเครื่องคอมพิวเตอร์ ชิพ UART เปลี่ยนข้อมูลจากแบบขนาน ให้เป็นแบบอนุกรม หรือจากแบบอนุกรม ให้เป็นแบบขนาน เครื่องคอมพิวเตอร์รุ่นเก่าๆ จะมีชิพ UART แบบ 8250 หรือ 16450 ชิพสองแบบนี้ ถึงแม้ว่าจะสามารถ รับ-ส่งข้อมูลด้วยความเร็วค่อนข้างสูง(ขนาดที่จะทำให้โมเด็ม ที่มีความเร็ว 28.8 Kbps ทำงานได้ตลอดเวลา)

แต่บัฟเฟอร์สามารถ เก็บตัวอักษรได้เพียงตัวเดียว ปัญหาที่เกิดขึ้นคือ โปรแกรมที่ทำงาน บนเครื่องคอมพิวเตอร์ อาจจะไม่เร็วพอ ที่จะอ่านตัวอักษรนั้น ก่อนที่มันจะถูกเขียนทับ ผลก็คือจะเกิดข้อผิดพลาดแบบที่เรียกว่า CRC error หรือ "comm overrun" เครื่อง 486 หรือเพนเทียมส่วนมาก จะมี UART ชิพแบบ 16550 ซึ่งมีบัฟเฟอร์ที่สามารถเก็บได้ถึง 16 ตัวอักษรแบบ FIFO(First-In, First-Out หรือเข้าก่อนออกก่อน) ซึ่งขนาดของบัฟเฟอร์ที่เพิ่มมากขึ้น ช่วยลดข้อผิดพลาดหรือ CRC error ได้ อย่างไรก็ตาม การติดตั้งโปรแกรมหรือเครื่องคอมพิวเตอร์อย่างไม่ถูกต้องหรือไม่เหมาะสมก็สามารถทำให้เกิด CRCerrorได้

2.9 การใช้งาน Timer /Counter

Timer/Counter0 ขนาด 8 บิต ซึ่งสามารถเลือกสัญญาณ Clock จาก CK(Clock ของระบบ) หรือสัญญาณ Clock ของระบบที่ถูกหาร(Prescaling) หรือ สัญญาณจากภายนอก โดยการใช้งานรีจีสเตอร์ TCCRO และ TIFR สัญญาณควบคุมสามารถทราบรายละเอียดจากรีจีสเตอร์ TCCRO ซึ่งการควบคุมการอินเตอร์รัปต์จะควบคุมได้จาก รีจีสเตอร์ TIMSK เมื่อ Timer/Counter0 รับสัญญาณจากภายนอก ซึ่งสัญญาณจะซิงโครไน(Synchronized) กับสัญญาณนาฬิกาภายในCPU โดย TIMER/COUNTER 0 จะเป็นวงจรนับขึ้นที่สามารถเขียนและอ่านข้อมูลได้ตลอดเวลา การเขียนข้อมูลลงใน TIMER/COUNTER 0 ในขณะที่มีสัญญาณ Clock จะทำให้ TIMER/COUNTER 0 นับค่าต่อเนื่องจากค่าที่ถูกเขียนลงไป

การใช้งาน Timer คือ การจับเวลา โดยการจับเวลานั้นมีสองทางเลือก ทางเลือกแรก คือการอ้างอิงเวลาจากวงจรภายนอก เช่น วงจรตั้งเวลาที่ทำจาก Ic ne555 ซึ่งจะต้องมีวงจรข้างนอกเข้ามาช่วยบอกเวลา แต่อีกแบบ คือ การนับจาก Machine cycle จากภายใน ก็คือ การอ้างจาก crytal ที่จ่ายให้กับ MCU นั่นเอง ซึ่งวิธีนี้ดีตรงที่ไม่ต้องมีวงจรภายนอกมาต่อ นับได้ทันทีแถมตัว avr ยังมี prescaler ให้ด้วย ความหมาย คือ มันสามารถที่จะหารหรือคูณความถี่ได้นั้นเอง เช่น เราตั้งเวลาให้มันจับเวลา 50usec แต่ต่อมาเราไม่พอใจ เราก็สามารถเลือก เป็นจำนวนเท่าได้ เช่น 8 เท่า ดังนั้นก็สามารถจับเวลาได้ เท่ากับ 50usec*8 เท่า = 400usec ได้ ก่อนอื่นเราต้องมาดูกันก่อนว่า avr นี้คำนวณเรื่อง Machine cycle ได้อย่างไร ก็ขอยกสูตรจาก MCU 8051

เวลาการกระทำคำสั่ง 1 Machine cycle = (จำนวน clock ใน 1 รอบ*จำนวนวงรอบที่คำสั่งสิ่งนั้นใช้)/(ความถี่ของ crytal ที่ใช้งาน)

ในที่นี้หากใช้งาน 90s8515 ใช้ crytal = 8MHZ จะได้เวลาในการกระทำคำสั่ง 1 Machine cycle คือ 1*1 (8MHZ) เท่ากับ(1/8) usec

การใช้งาน Timer/Counter 0 นั้นมีขนาด 8 bit หมายความว่าถ้าใช้ prescaler(คูณ 1) จะสามารถนับ pluse หรือ 256 ชิ้น งานนั้นเอง และจะสามารถจับเวลาได้สูงสุดโหมดนี้ได้ 256 Machine cycle หรือในเวลาสูงสุดเท่ากับ(cycle = 8 MHz) = (1/8)*256usec แต่หากใช้ prescaler ขนาดคูณ 2 เวลาสูงสุด (crytal =8MHz) = (1/8)*256*2usec หรือถ้าเราใช้เป็น counter ก็จะนับ pluse ได้ถึง 256*2 pluse เราไม่สามารถจับเวลาและนับพัลส์ พร้อมกันไม่ได้ ต้องเลือกว่าจะใช้ counter หรือ timer อย่างใดอย่างหนึ่ง สำหรับโหมด 8 bit มีวิธีคำนวณ เหมือนกันกับตัว mcu 8051 เลยคือ

ค่าการรับที่เซตในรีจีสเตอร์ = 256 - ค่าที่ต้องการนับ

ค่าการรับที่เซตในรีจีสเตอร์ = 256 – 24 = 232 แต่หากว่าคุณจะจับเวลา 24usec เราต้องคำนึงถึง Machine cycle ด้วย

ตัวอย่างโปรแกรม

```
timer_init(void)
{
TCCR0 = 0x00;    //stop หยุดการใช้งาน timer ก่อน
TCNT0 = 0x040;  //set count ใส่ค่าการนับ
TCCR0 = 0x01;    // start timer เริ่มทำการรับ prescaler = 1
}
```

ดังนั้น สรุปได้ว่า TCCR0 ใช้สำหรับเลือกโหมด pescale และ เริ่มการนับ

TCNT0 ใช้ใส่ค่าการนับ

ส่วนตัวรีจีสเตอร์ที่ใช้ตรวจสอบการนับคือ TOV0 เป็นตัวบอกว่าการนับครบหรือยัง หากครบแล้วจะมีค่าเป็นลอจิก 1 หากใช้ interrupt บิตนี้จะถูกเคลียร์เองอัตโนมัติ เมื่อมีการนับครบเกิดขึ้น

2.9.1 การใช้ Timer0 แบบใช้ interrupt

ตัวอย่างแบบ interrupt แบบจับเวลาทุกๆ 24 usec สังเกตรีจีสเตอร์ที่เปลี่ยนไปจากโปรแกรมที่แล้ว

```
void_ timer0_init(void)
{
TCCR0 = 0x00; //stop
TCNT0 = 0x40; //set count
TCCR0 = 0x01; //start timer
}
////////////////ตรงนี้เป็นส่วนของ interrupt sevice ruetine ของ timer 0
#program inerrut_hand timer0_ovf_isr:10
Void timer0_ovf_idr(void)
{
TCNT0 =0x40;  //reload counter value สังเกตว่าสามารถโหลดค่าได้ในทันที
              // โดยไม่ต้อง ตรวจสอบหรือเคลียร์ค่า TOV0
}
```

2.9.2 การใช้งาน Timer /Counter 1

จะมีขนาด 16 บิต โดยสามารถเลือกสัญญาณนาฬิกาได้จาก CK หรือสัญญาณที่ได้รับการหารจาก CK(Prescelling) สัญญาณควบคุมอยู่ในรีจีสเตอร์ TCCR1A และ TCCR1B การควบคุมสัญญาณอินเตอร์รัปต์จะควบคุมได้จากรีจีสเตอร์ TIMSK(TIMER/COUNTER INTERRUPT MASK REGISTER) เมื่อ TIMER1/COUNTER1 จะประกอบด้วยส่วนของการเปรียบเทียบเอาท์พุต(Output Compear Function) 2 ฟังก์ชั่น โดยจะใช้ รีจีสเตอร์ OCR1A(Output Compare Register 1 A) และ OCR1B(Output Compare Register 1B) เป็นส่วนของการเก็บค่าข้อมูลของการเปรียบเทียบ TIMER1/COUNTER1 จะสามารถเลือกใช้ฟังชั่น PWM ได้ทั้ง 8 บิต,9 บิต และ 10 บิต

The Timer/Counter 1 Control Register B-TCCR1B

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

รูปที่ 2.14 แสดง registor B TCCR1B

การเซ็ตค่า Bit 7 หรือ ICN1

คือ Input Capture 1 Noise Canceler(4 CKs) บิตนี้เป็นบิตที่กำหนดให้ Input Capture 1 Noise Canceler ทำงานหรือไม่ทำงาน โดยเมื่อบิตนี้เป็น 1 จะเป็นการกำหนดค่า Input Capture 1 Noise Canceler ทำงาน แต่เมื่อบิตนี้เป็น 0 จะเป็นการกำหนดไม่ให้ Input Capture 1 Noise Canceler ทำงาน ชุด Noise Canceler จะถูกกำหนดให้ทำงานโดยการ Sampling สัญญาณที่เข้ามาที่ชุด Input Capture 1 โดยสัญญาณ Sampling แรกจะเริ่มที่ขอบแรกของสัญญาณขาขึ้นหรือขาลงขึ้นอยู่กับการกำหนดใน บิต ICES1 โดยชุด Noise Canceler จะ Sampling ด้วยความถี่เท่ากับความถี่ของ XTAL ซึ่งจะ Sampling ทั้งหมด 4 ครั้ง โดยลอจิกที่ได้จากการ Sampling จะต้องมีลอจิกเดียวกันกับลอจิกที่กำหนดในบิต ICES1

การเซ็ตค่า Bit 6 หรือ ICES1

คือ Input Capture 1 Edge Select เป็นบิตที่ใช้กำหนดให้ชุด Input Capture 1 จะต้อง Detect ถ้าบิต ICES1 เซ็ต เป็น 1 จะเป็นการกำหนดให้ชุด Input Capture1 ทำหน้าที่ Detect สัญญาณที่ขอบขาขึ้น แต่ถ้าบิต ICES 1 ถูกเคลียเป็น 0 จะเป็นการกำหนดให้ชุด Input Capture 1 ทำหน้าที่ Detect สัญญาณที่ขอบขาลง

การเซ็ตค่า Bit 5, 4 หรือ RES

คือ Reserved bits บิตนี้ถูกสงวนไว้

การเซ็ตค่า Bit 3 หรือ CTC1

คือ Clear Timer1/Counter1 on Compare Match บิตนี้เป็นที่ใช้ในการกำหนดว่าเมื่อเกิด Output Compare แล้วจะให้เกิดการนับต่อไปหรือจะให้มีการรีเซตค่าให้เป็น 00000 แล้วจึงทำการนับต่อไป โดยถ้าเป็นบิตนี้เป็น 1 จะเป็นการกำหนดให้มีการรีเซตค่าให้เป็น 0 เมื่อเกิดการ Output Compare แต่ถ้าบิตนี้เคลียร์เป็น 0 จะเป็นการกำหนดให้มีการนับค่าต่อเมื่อเกิด Output Compare

การเซ็ตค่า Bit 2, 1 และ 0 หรือ CS12, CS11 และ CS10

คือ Clock Select1, บิต 2, 1 and 0 เป็นบิตที่ใช้ในการเลือกสัญญาณ Clock

The Timer/Counter In Capture Register – ICR1H AND ICR1L

เป็นรีจีสเตอร์ขนาด 16 บิต ที่ใช้เก็บค่า Timer/Couner1 ที่อยู่ในรีจีสเตอร์ TCNT1 เมื่อ Input Capture สามารถ Detect ได้ เมื่อ Input Capture สามารถ Detect สัญญาณได้ตามที่กำหนดในบิต ICES1 จะทำให้ CPU โหลดค่าในรีจีสเตอร์ TCNT1 ลงในรีจีสเตอร์ และในเวลาเดียวกับบิต ICF1จะเซ็ตเป็น 1 โดยการอ่านค่ารีจิสเตอร์ ICR1 ของ CPU จะใช้รีจีสเตอร์ TEMP เป็นรีจีสเตอร์พักข้อมูล ซึ่งการใช้รีจีสเตอร์ TEMP ช่วยในการอ่านข้อมูลเพื่อให้ค่าที่อยู่ในรีจีสเตอร์ ICR1H และ ICR1L เหมือนถูกอ่านออกมาพร้อมกัน การอ่านค่าจากรีจีสเตอร์ ICR1 จะต้องอ่านค่าจากรีจีสเตอร์ ICR1L ก่อน โดยเมื่อ CPU

อ่านค่าจาก ICR1L จะทำให้ค่าใน รีจีสเตอร์ ICR1H ถูกโหลดลงในรีจีสเตอร์ TEMP เมื่อ CPU อ่านค่าจาก ICR1H จะทำให้ค่าในรีจีสเตอร์ TEMP ถูกส่งให้ CPU ส่วนการใช้งาน Timer คือ การจับเวลานั้นเอง โดยการจับเวลานั้นมีสองทางเลือก ทางเลือกแรก คือการอ้างอิงเวลาจากวงจรภายนอก เช่น วงจรตั้งเวลาที่ทำจาก Ic ne555 ซึ่งจะต้องมีวงจรข้างนอกเข้ามาช่วยบอกเวลา แต่อีกแบบ คือ การนับจาก Machine cycle จากภายใน ก็คือ การอ้างจาก crytal ที่จ่ายให้กับ MCU นั่นเอง ซึ่งวิธีนี้ดีตรงที่ไม่ต้องมีวงจรภายนอกมาต่อ นับได้ทันทีแถมตัว avr ยังมี prescaler ให้ด้วย ความหมาย คือ มันสามารถที่จะหารหรือคูณความถี่ได้นั้นเอง เช่น เราตั้งเวลาให้มันจับเวลา 50usec แต่ต่อมาเราไม่พอใจ เราก็สามารถเลือกเป็นจำนวนเท่าได้ เช่น 8 เท่า ดั้งนั้นก็สามารถจับเวลาได้เท่ากับ 50usec*8 เท่ากับ 400usec ได้ครับ ก่อนอื่นเราต้องมาดูกันก่อนว่า avr นี้คำนวณเรื่อง Machine cycle ได้อย่างไร ก็ขอยกสูตรจาก MCU 8051

เวลาการกระทำคำส่ง 1 Machine cycle = (จำนวน clock ใน 1 รอบ*จำนวนวงรอบที่คำสั่งสิ่งนั้นใช้)/
(ความถี่ของ crytal ที่ใช้งาน)

ในที่นี้หากใช้งาน 90s8515 ใช้ crytal = 8MHZ จะได้เวลาในการกระทำคำสั่ง 1 Machine cycle

= 1*1 (8MHZ) = (1/8) usec

การใช้งาน Timer/Counter 1 นั้นมีขนาด 16 บิต หมายความว่าถ้าไม่ใช้ prescaler(คูณ 1) จะสามารถนับ pluse ได้ 65536 pluse หรือ 65536 ชิ้น งานนั้นเอง และจะสามารถจับเวลาได้สูงสุดโหมดนี้ได้ 65536 machen cycle หรือแนวเวลาสูงสุดเท่ากับ(cycle = 8 MHz) = (1/8)*65536usec แต่หากใช้ prescaler ขนาดคูณ 2 เวลาสูงสุด(crytal =8MHz) = (1/8)*65536*2usec หรือถ้าเราใช้เป็น counter ก็จะนับ pluse ได้ถึง 65536*2 pluse เราไม่สามารถจับเวลาและนับพัลส์ พร้อมกันไม่ได้ ต้องเลือกว่าจะใช้ counter หรือ timer อย่างใดอย่างหนึ่ง สำหรับโหมด 8 บิต มีวิธีคำนวณ เหมือนกันกับตัว mcu 8051 เลย คือ ค่าการรับที่เซตในรีจีสเตอร์ =65536 - ค่าที่ต้องการนับ

ค่าการรับที่เซตในรีจีสเตอร์ = 65536 – 24 = 65512 แล้วแยกไบต์บนและไบต์ล่างใส่ในรีจีสเตอร์การนับตามลำดับดังนี้

OCR1AH = byte บน;

OCR1AL = byte ล่าง;

OCR1BH = byte บน;

OCR1BL = byte ล่าง;

2.9.3 การใช้งาน Timer/Counter1 ในโหมด PWM

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

รูปที่ 2.15 แสดงการใช้งาน Timer /Counter 1

การเซ็ตค่า Bits 7 กับ 6 หรือ COM1A1,COM1A0

คือ Compare Output Mode 1A,บิต 1 and 0 บิต COM1A1 และ COM1A0 เป็นบิตที่ใช้ในการกำหนดลักษณะของสัญญาณที่เกิดขึ้นที่ขา OC1A เมื่อTimer/Counter1 เกิด Compare Match ซึ่งเมื่อใช้ฟังชั่น Output Compare Match ของ Timer/Counter1 จะต้องควบคุมให้ขา OC1A มีสถานะเป็นเอาท์พุต โดยการเลือกลักษณะของสัญญาณแสดงในตาราง

การเซ็ตค่า Bit 5 กับ 4 หรือ COM1B1 กับ COM1B0

คือ Compare Output Mode 1 B, บิต 1 and 0 บิต Com1A1 และ COM1A0 เป็นบิตที่ใช้ในการกำหนดลักษณะของสัญญาณที่เกิดขึ้นที่ขา OC1A เมื่อTimer/Counter1 เกิด Compare Match ซึ่งเมื่อใช้ฟังชั่น Output Compare Match ของ Timer/Counter1 จะต้องควบคุมให้ขา OC1A มีสถานะเป็นเอาท์พุต

การเซ็ตค่า Bit 3 กับ 2 หรือ Res :Reserved bits

คือ ในส่วนของ AT mega32 จะสงวนบิตในกลุ่มนี้ไว้

การเซ็ตค่า Bit 1 กับ 0 หรือ PWM11 กับ PWM10

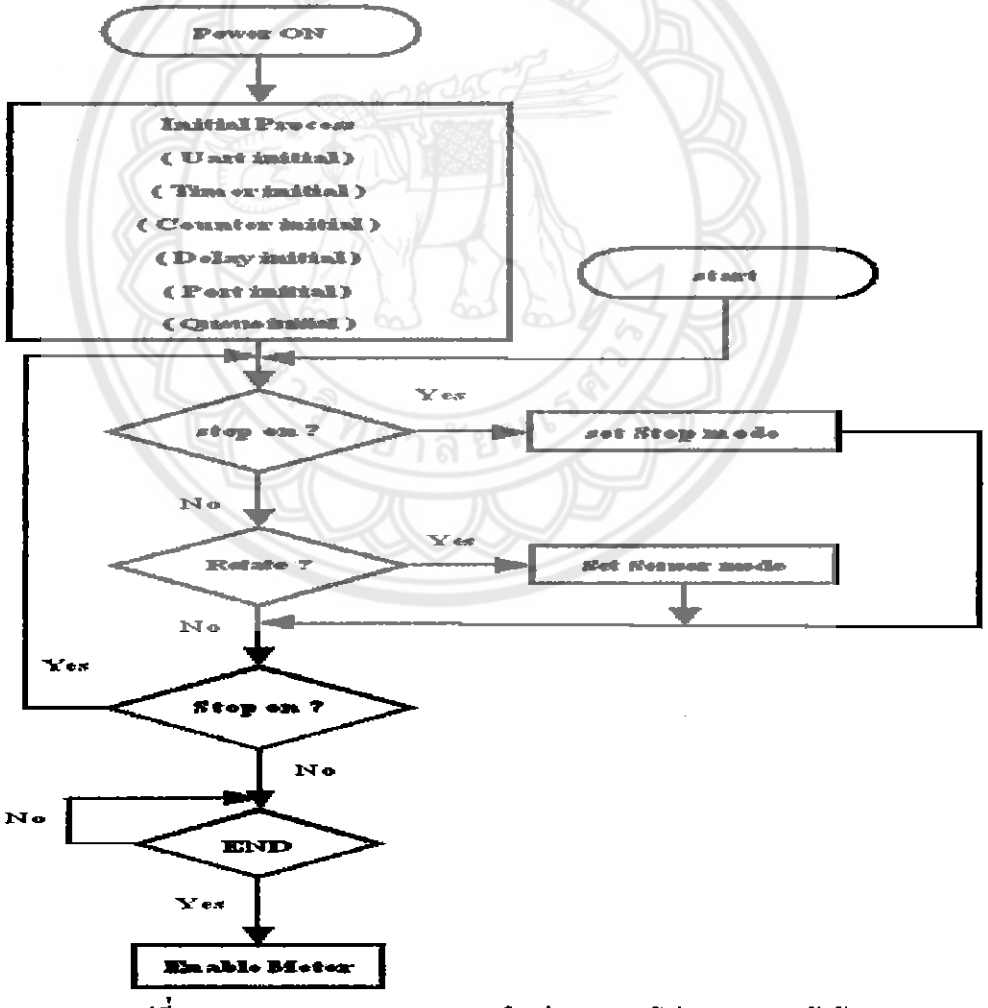
คือ Pulse Width Modulator Select Bit เป็นบิตที่ใช้ในการกำหนดการทำงานของ PWM

บทที่ 3
การสร้างมอเตอร์ทดสอบและตัววัดรอบ

ในบทนี้จะพูดถึงการสร้างมอเตอร์กระแสตรงและเครื่องวัดรอบซึ่งการทำงานจะแบ่งออกเป็นสองส่วนคือ เขียนโปรแกรมควบคุม และสร้างตัวมอเตอร์กระแสรงและเครื่องวัดรอบ และจากบทที่ 2 อธิบายละเอียดของมอเตอร์กระตรงและเครื่องวัดรอบ รวมถึงศึกษาหลักการเขียนโปรแกรมลงบอร์ดจะนำมาใช้ในบทนี้

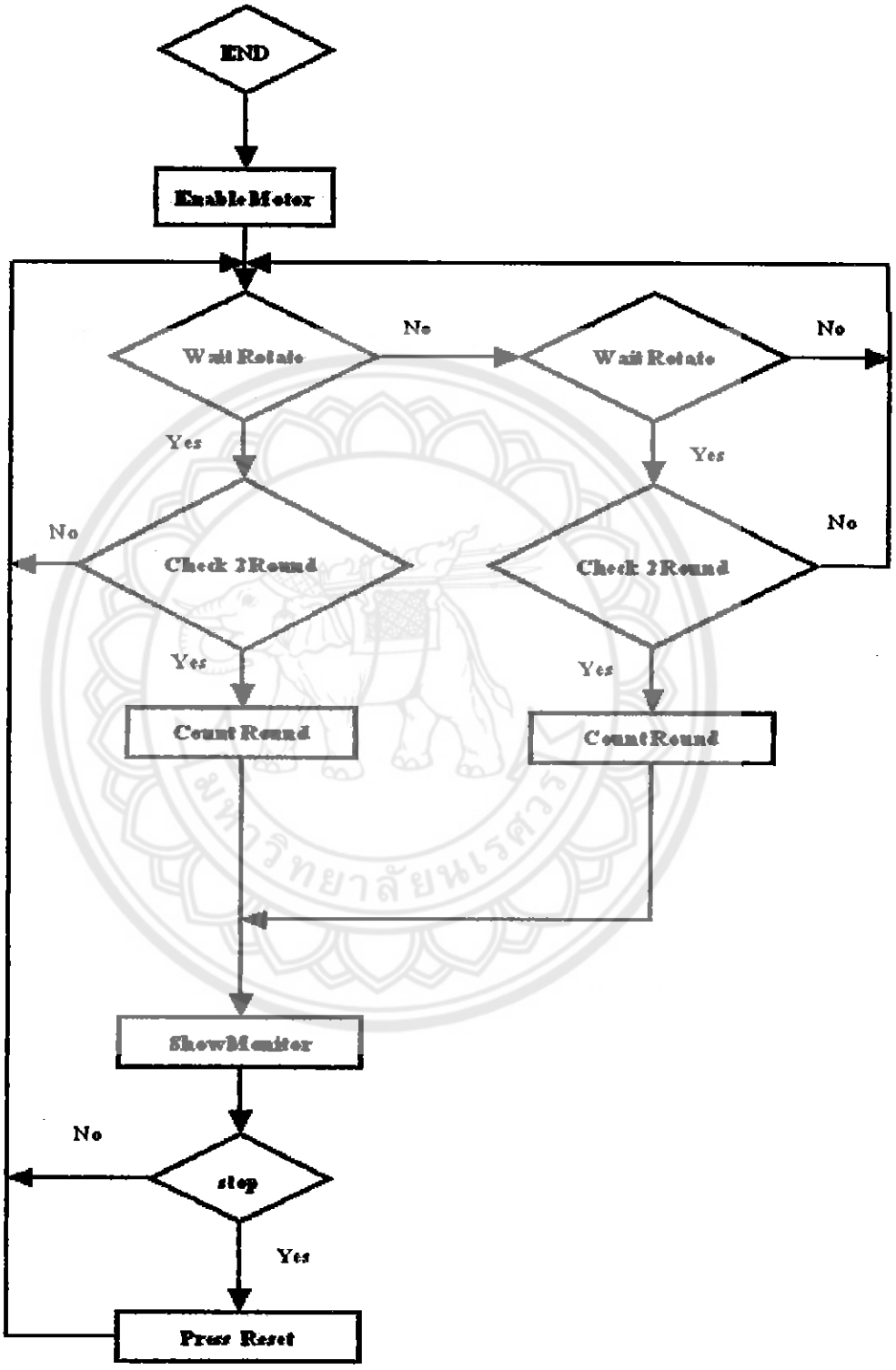
3.1 การเขียนโปรแกรมควบคุมเครื่องวัดรอบและมอเตอร์กระแสตรง

การเขียนโปรแกรมของเครื่องวัดรอบนี้จะแบ่งเป็นสองส่วนใหญ่คือ ส่วนแรกคือการเขียนให้ตัววัดรอบติดต่อกับคอมพิวเตอร์และการเซ็ตค่า registor โปรแกรมดังรูปที่ 3.1 แสดง Software flow chart ในส่วน main โปรแกรมของตัววัดรอบ



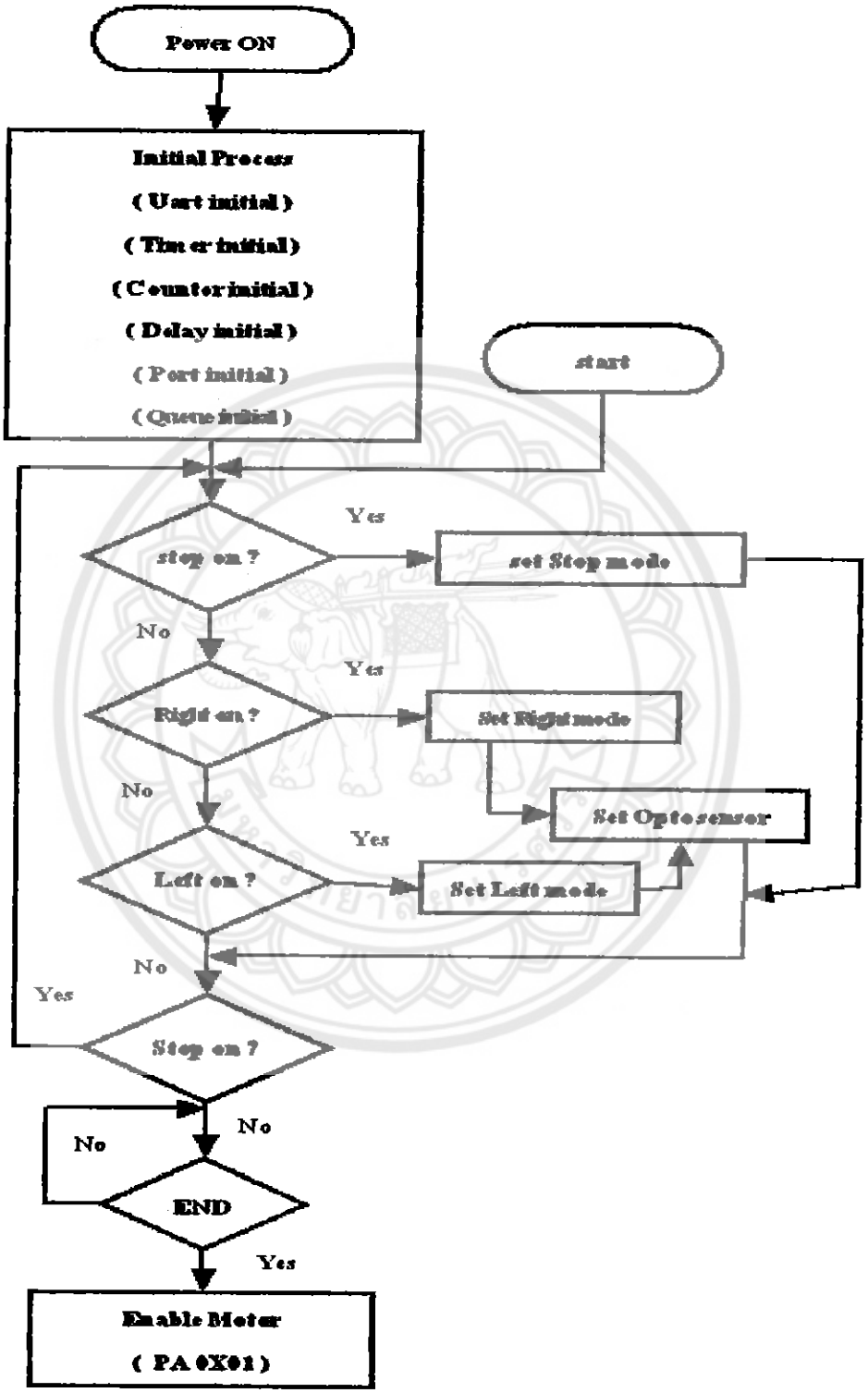
รูปที่ 3.1 แสดง Software flow chart ในส่วน main โปรแกรมของตัววัดรอบ

ส่วนที่สองจะเป็นการเขียนโปรแกรมสั่งงานตัววัดรอบ ว่าให้รอใบพัดหมุนและเช็คการหมุนว่าหรือไม่ จากนั้นจะแสดงการนับออกทาง LCD ดังรูปที่ 3.2 แสดง Software flow chart ในส่วนของการควบคุมเครื่องวัดรอบ



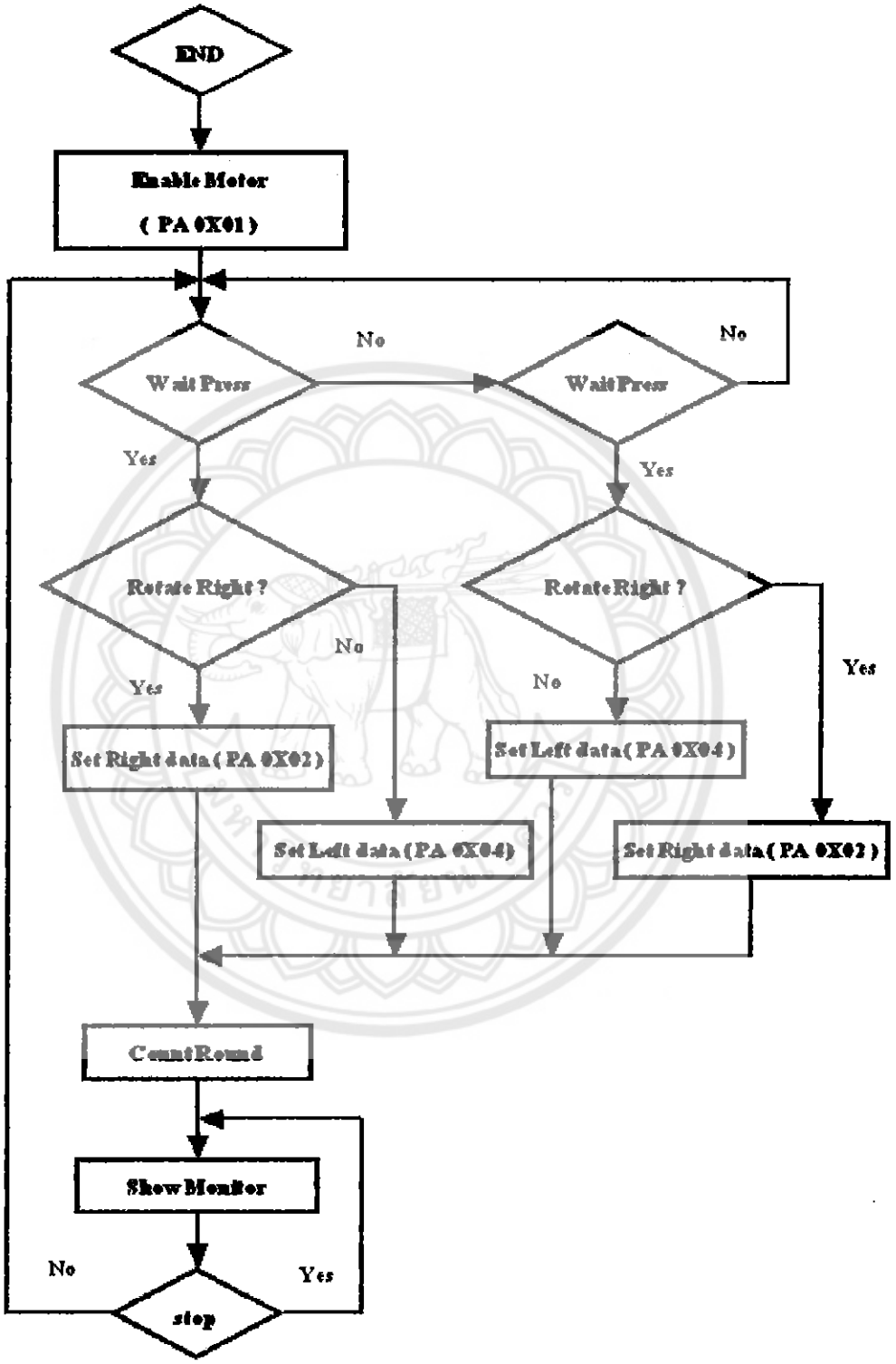
รูปที่ 3.2 แสดง Software flow chart ในส่วนของการควบคุมเครื่องวัดรอบ

การเขียนโปรแกรมของมอเตอร์นี้จะแบ่งเป็นสองส่วนใหญ่คือ ส่วนแรกคือการเขียนให้มอเตอร์ติดต่อกับคอมพิวเตอร์และการเซ็ตค่า registor โปรแกรมดังรูปที่ 3.3 แสดง Software flow chart ในส่วน main โปรแกรมของมอเตอร์



รูปที่ 3.3 แสดง Software flow chart ในส่วน main โปรแกรมของมอเตอร์

ส่วนที่สองจะเป็นการเขียนโปรแกรมสั่งงานตัวมอเตอร์ ว่าให้มอเตอร์หมุนใบพัดซ้ายหรือ และเขียนโปรแกรมตัว sensor จากนั้นจะแสดงการนับออกทางจอคอมพิวเตอร์ ดังรูปที่ 3.4 แสดง Software flow chart ในส่วนของการควบคุมมอเตอร์



รูปที่ 3.4 แสดง Software flow chart ในส่วนของการควบคุมมอเตอร์

3.2 การเขียนโปรแกรมให้คอมพิวเตอร์กับไมโครคอนโทรลเลอร์เชื่อมต่อกัน

ในการเชื่อมต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรเลอร์นั้น จะต้องเขียนโปรแกรม UART ขึ้นมาก่อน UART จะเปลี่ยนข้อมูลจากแบบขนาน ให้เป็นแบบอนุกรม หรือจากแบบอนุกรม ให้เป็นแบบขนาน จะทำให้การส่งข้อมูลนั้นจะเป็นแบบ stack เพราะไมโครคอนโทรลเลอร์นั้นจะทำงานที่ละคำสั่งไม่สามารถทำงานหลายคำสั่งได้และจะไม่กระโดดข้ามการทำงาน โดยจะเขียนด้วยภาษา C โดยเขียนผ่าน port RS232

ตัวอย่าง UART ที่เขียนลงในไมโครคอนโทรลเลอร์

```
#include <io8515v.h>            // เรียกไฟล์ io8515v.h มาใช้ ซึ่งประกอบด้วยค่า register ต่างๆ
#include <stdio.h>
#include <macros.h>
#include "standard.h"
#include "queue.h"              // เรียกไฟล์ queue.h ซึ่งใน queue.h จะเขียนให้การส่งข้อมูลเป็นแบบ
                                // stack
u8 PC232 = 0;                   //variable for receive value from PC232 buffer queue
Queue PC232_buff;
void init_PC232(void)
{
 Init_queue(&PC232_buff);
}
//Wait here until receive one byte from PC and store in global variable PC232
void wait_PC232(void)
{
        while(Empty(&PC232_buf))        //Wait in this routine if queue empty
        {
        }
        DeleteQueue(&PC232,&PC232_buff);        //PC232 keeps latest data from PC
}
#pragma interrupt_handler uart0_rx_isr:10
void uart0_rx_isr(void)
{
//uart has received a character in UDR
AddQueue(UDR,&PC232_buff);      }
```

3.3 เขียนโปรแกรมควบคุมการหมุนของ dc motor

สร้างตัว DC Motor 5V สามารถควบคุมทิศทางการหมุนได้ทั้ง ซ้ายและขวา และอ่านรอบการหมุนได้โดยใช้ Opto sensor เป็น Sensor ลักษณะของขั้วต่อใช้งาน และการส่งลอจิกควบคุมการหมุน โดย ขั้วต่อ IN1,IN2 จะเป็นตัวกำหนดทิศทางการหมุน ส่วนขั้วต่อ EN จะทำหน้าที่ Enable ชุดขับกระแส โดยจะทำงานที่ลอจิก "1" ขั้วต่อ OPA และ OPB จะทำหน้าที่ส่งสัญญาณลอจิก "1" ไปให้ MCU เมื่อใบพัดของ DC Motor ไปตัดที่ตัว Opto Sensor โดยถ้าไม่มีอะไรมาตัดที่ตัว Sensor ที่ขั้วต่อ OPA และ OPB จะมีสถานะเป็น "0" ซึ่งใช้อ่านรอบการหมุนของ Motor โดยเลือกอ่านสัญญาณที่ต่อ OPA หรือ OPB ก็ได้ แสดงดังรูป 3.5 โครงสร้าง STEPPING MOTOR และตำแหน่งขา Port ใช้งาน

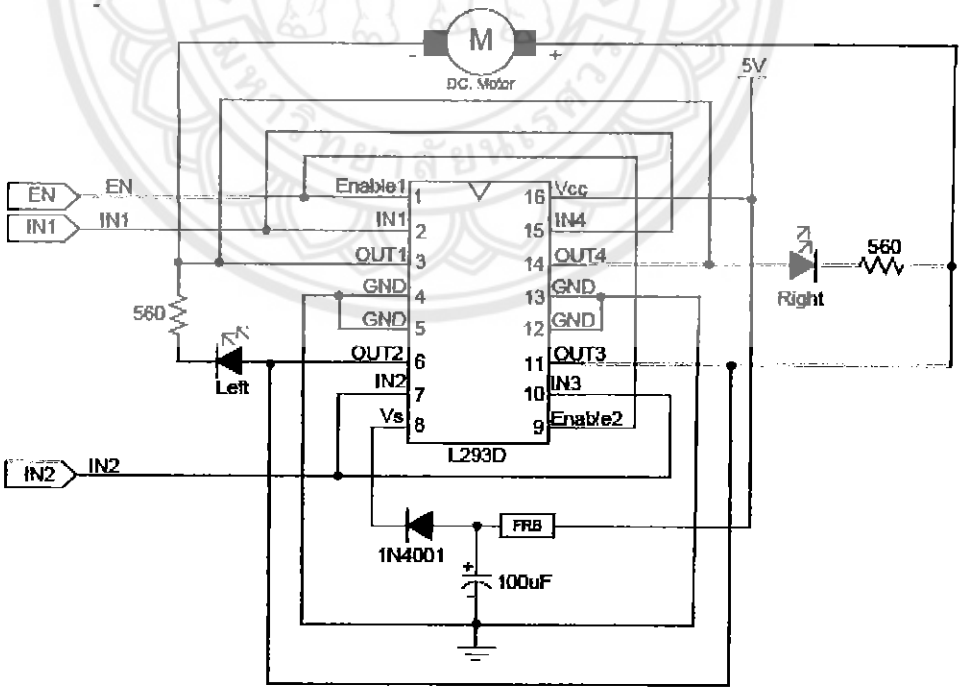
การเขียนโปรแกรมควบคุมมอเตอร์

Port PA0 เป็น EN จะทำหน้าที่ Enable ชุดขับกระแส โดยจะทำงานที่ลอจิก "1" แต่ถ้ากด S จะเป็นการสั่งให้ dc motor หยุดหมุน

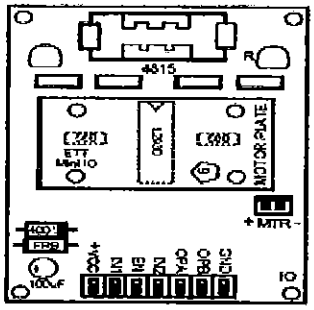
Port PA1 เป็นการควบคุมให้ dc motor หมุน ขวา และมี LED บอกว่าหมุนขวา โดยการหมุนนั้นจะให้ User กด R สัญญาณลอจิก "1"

Port PA2 เป็นการควบคุมให้ dc motor หมุน ซ้าย และมี LED บอกว่าหมุนซ้าย โดยการหมุนนั้นจะให้ User กด L สัญญาณลอจิก "1"

Port PB0 เป็น การใช้อ่านรอบการหมุนของ Motor โดยเลือกอ่านสัญญาณจากข้อต่อ OPA

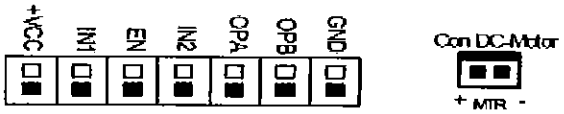


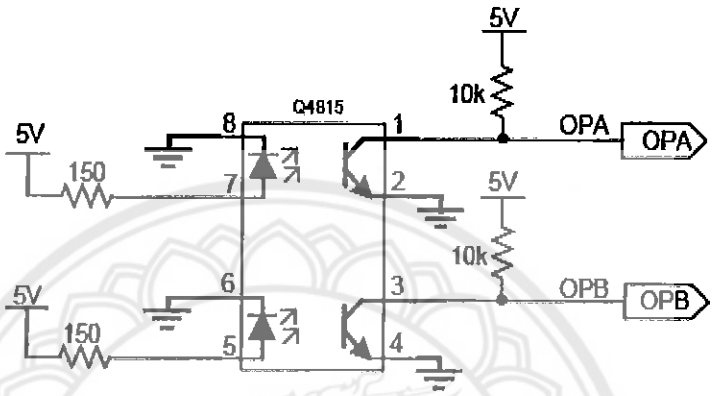
รูปที่ 3.5 โครงสร้าง STEPPING MOTOR และตำแหน่งขา Port ใช้งาน



ตารางแสดงการ Control DC-MOTOR

Control DC MOTOR								
RIGHT(R)			LEFT(L)			STOP		
EN	IN1	IN2	EN	IN1	IN2	EN	IN1	IN2
1	1	0	1	0	1	0	x	x





รูปที่ 3.6 รูปวงจร STEPPING MOTOR

3.4 การสร้างเครื่องวัดรอบ

สร้างตัวเครื่องวัดรอบที่สามารถอ่านรอบการหมุนได้โดยใช้ sensor ดังรูปที่ 3.7 แสดงแท่งที่พันด้วยทองแดง 2 อัน เป็น sensor ลักษณะของสนามไฟฟ้า คือถ้าใบพัดที่มีส่วนของทองแดงติดอยู่ไปตัดจะมีค่าเป็น "1" แต่ถ้าใบพัดที่ไม่มีแผ่นทองแดงติดอยู่ไปตัดจะมีค่าเป็น "0" การที่เครื่องวัดรอบจะอ่านรอบการหมุนได้โดยใช้ Sensor ลักษณะของเซนเซอร์คือเป็นแท่งสองแท่งพันด้วยทองแดง

รูปที่ 3.7 แสดงแท่งที่พันด้วยทองแดง 2 อัน

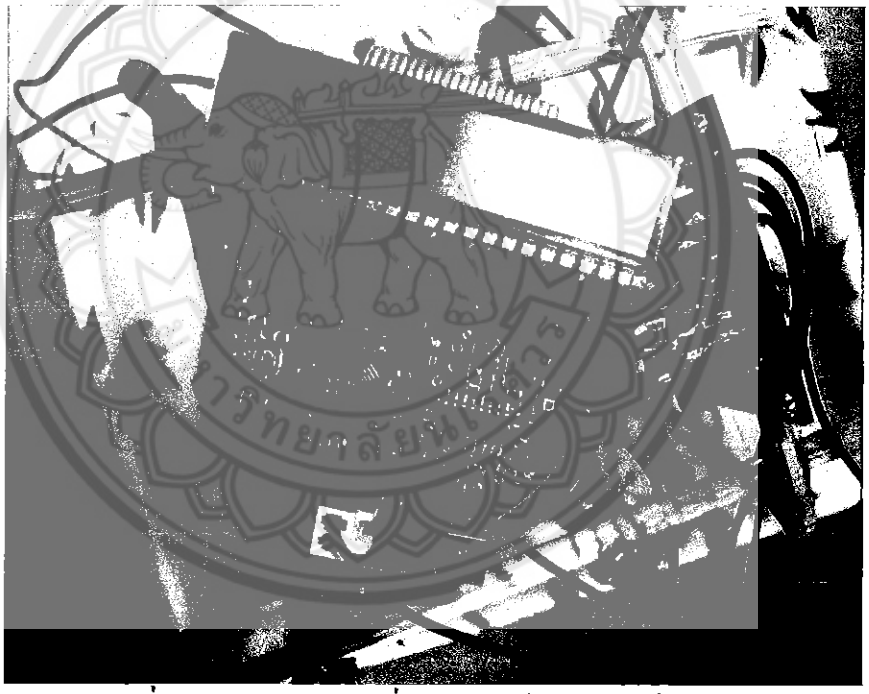
จากรูปที่ 3.8 แสดงตัวตัดสัญญาณ ซึ่งเป็นแผ่นวงกลมที่ครึ่งหนึ่งของวงกลมมีทองแดงขนาดบางไว้ทำหน้าที่ตัดทำสัญญาณให้เกิด damp(1) กับ undamp(0) เพื่อการนับรอบของเครื่องวัดรอบ

รูปที่ 3.8 แสดงแผ่นวงกลมที่มีแผ่นทองแดงติดอยู่ครึ่งหนึ่ง

เมื่อสร้างมอเตอร์กับเครื่องวัดรอบเสร็จ มอเตอร์ทดสอบจะว่างอยู่ด้านล่างมีใบพัดอยู่ตรงกลาง และมี Opto sensor ของมอเตอร์ติดอยู่ส่วนข้างบนสุดจะมีตัววัดรอบวางอยู่บนสุด ระหว่างใบพัดกับตัววัดรอบจะมี sensor ของเครื่องวัดรอบวางกันอยู่ ดังรูปที่ 3.9 แสดงการต่อเครื่องวัดรอบเข้ากับมอเตอร์ทดสอบ(ด้านหน้า) รูปที่ 3.10 แสดงการต่อเครื่องวัดรอบเข้ากับมอเตอร์(ด้านข้าง) และรูปที่ 3.11 แสดงการต่อเครื่องวัดรอบเข้ากับมอเตอร์(ด้านบน)

รูปที่ 3.9 แสดงการต่อเครื่องวัดรอบเข้ากับมอเตอร์(ด้านหน้า)

รูปที่ 3.10 แสดงการต่อเครื่องวัดรอบเข้ากับมอเตอร์(ด้านข้าง)



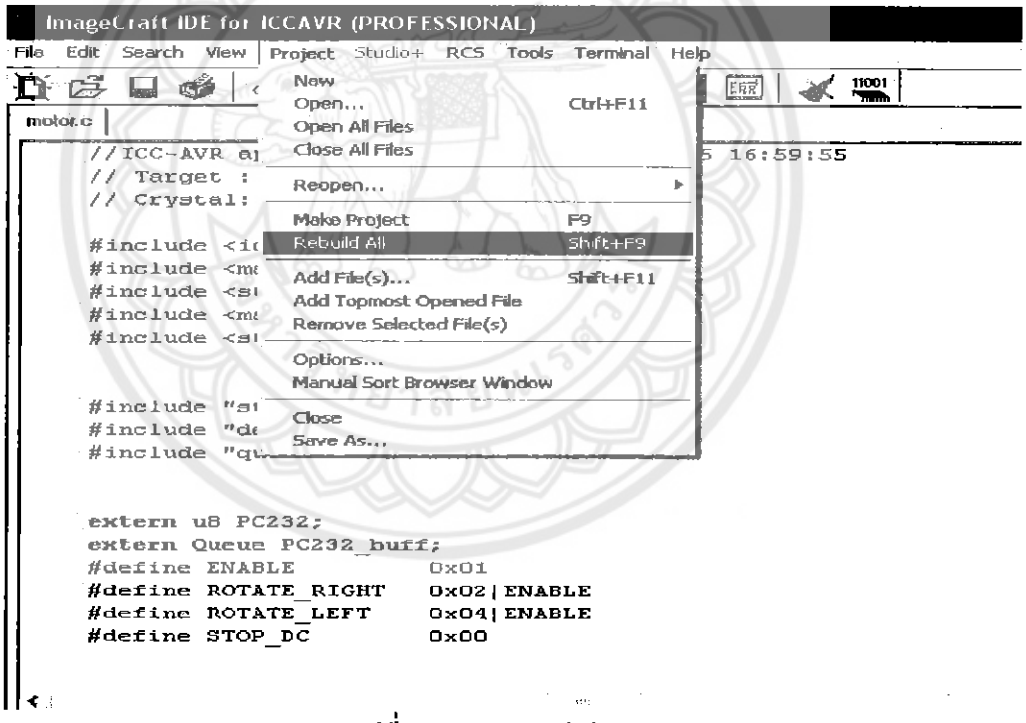
รูปที่ 3.11 แสดงการต่อเครื่องวัดรอบเข้ากับมอเตอร์(ด้านบน)

บทที่ 4
การทดสอบมอเตอร์และเครื่องวัดรอบ

บทนี้จะเป็นการทดสอบมอเตอร์หมุนใบพัดกับเครื่องวัดรอบว่ามีขั้นตอนอย่างไร ในการทดสอบเครื่องวัดรอบและตัววัดรอบใบพัดโดยให้ทั้งสองนับรอบพร้อมกัน โดยจะใช้เวลาในการทดสอบ 5 นาที ทำซ้ำ 10 ครั้ง โดยการทดสอบแต่ละครั้งจะไม่เปลี่ยนแปลงอะไรเลย และนำผลที่ได้จากการทดสอบมาคำนวณหาค่าเฉลี่ยเพื่อดูว่าเครื่องวัดรอบนั้นสามารถนับรอบตรงตามรอบใบพัดหรือไม่ และหาเปอร์เซ็นต์ความผิดพลาด

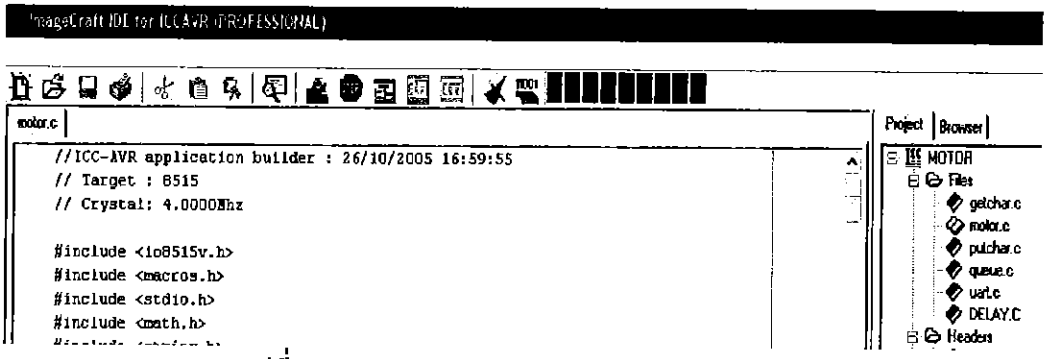
4.1 เริ่มการ run ทดสอบโปรแกรมของมอเตอร์

เมื่อเขียนโปรแกรมควบคุมมอเตอร์ให้หมุนใบพัดเสร็จ ให้เลือก Project > Rebuild all ดังรูปที่ 4.1 จากนั้นจะทำการ debug และโปรแกรมจะบอกสถานะว่าหน่วยความจำเหลือเท่าไหร่ ในบอร์ด



รูปที่ 4.1 การ debug โปรแกรม

เมื่อโปรแกรมคอมไพล์ผ่าน โปรแกรมจะทำการลบสิ่งที่เคยโปรแกรมลงบน Board ก่อนนี้ จากนั้นโปรแกรมจะทำการเขียนใหม่ลงไปใน board ดังรูปที่ 4.2 แสดงการเขียนโปรแกรมลงบอร์ด



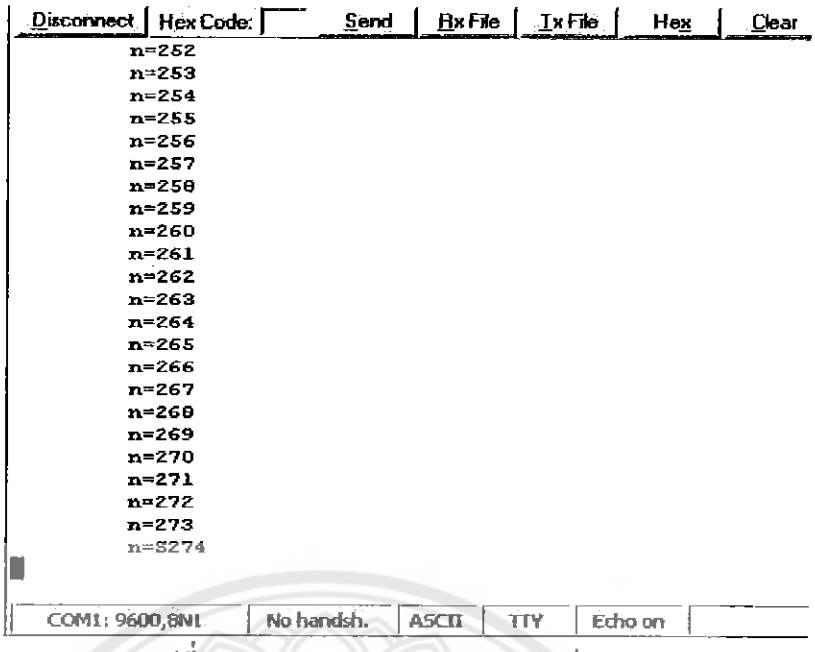
รูปที่ 4.2 แสดงการ write โปรแกรมลง Board stk 200

4.2 ผลการทดสอบให้มอเตอร์หมุนใบพัด

เมื่อเขียนโปรแกรมลง Board เสร็จให้เปิดโปรแกรม AVR > Terminal โปรแกรมจะทำการเชื่อมต่อกับ Board เมื่อโปรแกรมทำการเชื่อมต่อเสร็จ ให้กด " L" จะทำให้ IN2 มีลอจิกเป็น "1" มอเตอร์จะหมุนซ้าย และถ้าต้องการให้มอเตอร์หมุนขวาให้กด "R" เมื่อกด "R" ที่ขา IN1 จะมีค่าลอจิกเป็น 1 โปรแกรมจะสั่งให้มอเตอร์หมุนขวา รูปที่ 4.3 แสดงการหมุนใบพัดและตัว sensor ที่กำลังทำการนับรอบใบพัด

รูปที่ 4.3 แสดงการหมุนใบพัดและ sensor นับรอบที่หมุน

จากรูปที่ 4.3 เมื่อสั่งให้มอเตอร์หมุน ขวา หรือ ซ้าย โปรแกรมจะทำการนับจำนวนรอบที่มอเตอร์หมุนแสดงออกมาเป็น n เท่ากับจำนวนรอบที่มอเตอร์หมุน ดังรูปที่ 4.4 แสดงการนับรอบของใบพัดและเมื่อต้องการให้มอเตอร์หยุดหมุนให้กด "S" เป็นการทำให้ ขา EN นั้นมีค่าลอจิกเป็น "0"



รูปที่ 4.4 แสดงรอบการหมุนของมอเตอร์ทอสอบ

4.3 การทดสอบเครื่องวัดรอบ

เมื่อมอเตอร์ทดสอบหมุนใบพัดของเครื่องวัดรอบนั้นจะหมุนตามทำให้ sensor ของตัววัดรอบนั้นจะนับรอบพร้อมกันกับมอเตอร์ทดสอบด้วย เมื่อตัววัดรอบเกิดการ damp(1) กับ undamp(0) สองครั้งจะนับเป็นหนึ่งรอบการหมุนขอบใบพัดน้ำ ดังรูปที่ 4.5 แสดงผลการทดสอบเครื่องวัดรอบ

รูปที่ 4.5 แสดงผลการทดสอบเครื่องวัดรอบ

การสอบนี้จะสั่งให้ใบพัดหมุนและนับรอบของใบพัดและเครื่องวัดรอบโดยใช้เวลาในการทดสอบ 10 ครั้งโดยไม่มีการเปลี่ยนแปลงใดๆ ทั้งสิ้นแต่ละครั้งเท่ากับ 5 นาที เพื่อแสดงให้เห็นว่าความสามารถในการนับรอบนั้นเป็นอย่างไรและมีความผิดพลาดเท่ามากน้อยขนาดไหนและบันทึกผลลงตารางดังตารางที่ 4.1 แสดงการเปรียบเทียบการนับรอบของใบพัดและเครื่องวัดรอบ

ตารางที่ 4.1 แสดงการเปรียบเทียบการนับรอบของใบพัดและเครื่องวัดรอบ

ครั้งที่	รอบของใบพัด (รอบ)	รอบของเครื่องวัดรอบ (รอบ)
1	4,015	1,024
2	4,009	1,021
3	4,015	1,025
4	4,019	1,018
5	4,024	1,020
6	4,012	1,017
7	4,021	1,023
8	4,017	1,028
9	4,014	1,019
10	4,020	1,024

สูตร ค่าเฉลี่ย = ผลรวมของรอบทั้งหมด / จำนวนครั้ง

ค่าเฉลี่ยรอบของใบพัด เท่ากับ 40,166 / 10 = 4,016.6 รอบต่อ 5 นาที

ทำเป็นรอบต่อนาทีได้ 40,166.6 / 5 = 803.32 รอบ / นาที

ค่าเฉลี่ยรอบของเครื่องวัดรอบเท่ากับ 10,219 / 10 = 1,021.9 รอบต่อ 5 นาที

ทำเป็นรอบต่อนาทีได้ 1,021.9 / 5 = 204.38 รอบ / นาที

ดังนั้น เมื่อเปรียบเทียบรอบของใบพัดกับเครื่องวัดรอบจะได้ 803.32 / 204.38 = 3.93 เท่าหรือประมาณ 4 เท่า

จากการเปรียบเทียบรอบของใบพัดกับเครื่องวัดรอบจะได้ 1 ต่อ 4 รอบ/นาที

การหาเปอร์เซ็นต์ความคลาดเคลื่อนจะได้ (1 /4) X 100 = 25 %

จากตารางเมื่อนำมาหาค่าเฉลี่ยแล้วเปรียบเทียบค่าระหว่างเครื่องวัดรอบกับรอบใบพัดจะได้ว่าในการทดลองทั้ง 10 ครั้ง ครั้งละ 5 นาที ค่าของเครื่องวัดรอบนั้นมีความเร็วรอบอยู่ที่ 204 รอบ/นาที จากความเร็วรอบขนาดนี้จะช้ากว่าใบพัดที่มีความเร็วอยู่ที่ 803 รอบ/ นาที หรือห่างกันประมาณ 1 ต่อ 4 รอบ/นาที และมีเปอร์เซ็นต์ความผิดพลาดจะเท่ากับ 25%

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการทดสอบรอบของเครื่องวัดรอบเทียบกับรอบของใบพัด

จากการทดสอบเครื่องวัดรอบกับใบพัดโดยสั่งให้มอเตอร์หมุนแล้วนับรอบเป็นเวลา 5 นาที แล้วบันทึกผลทำทั้งหมด 10 ครั้งปรากฏว่าค่าเฉลี่ยที่ได้จากการทดสอบนั้นมีความแตกต่างกันคือ เครื่องวัดรอบนับรอบได้ช้ากว่าใบพัดประมาณ 4 เท่า สาเหตุมาจากเครื่องวัดรอบนั้นนับรอบ สัญญาณ "1" กับ "0" ไม่ทันเพราะมอเตอร์หมุนใบพัดเร็วประมาณ 803 รอบต่อนาที ทำให้มีเปอร์เซ็นต์ความผิดพลาดจะเท่ากับ 25%

5.2 ข้อเสนอแนะและแนวทางแก้ไข

จากการทดสอบน่าจะทำตัว sensor ของตัววัดรอบใหม่ ให้เมื่อใบพัดตัดกับตัว sensor แล้วทำให้โปรแกรมนับค่าได้ ถูกต้องมากยิ่งขึ้น และน่าจะเปลี่ยนบอร์ดใหม่ให้มีตัว Timer มากกว่าสองตัวขึ้นไปจะได้ทำ PWM ได้ เมื่อทำ PWM จะสามารถทำให้มอเตอร์หมุนใบพัดช้าลงได้สามขั้นคือ 8 บิต , 9 บิต และ 10 บิต

5.3 แนวทางพัฒนาต่อไป

1. พัฒนาตัว sensor ใหม่เพื่อที่จะให้โปรแกรมจับสัญญาณการเป็น 1 กับ 0 ได้ง่ายกว่าเดิม
2. เมื่อนับรอบได้ตรงแล้วควรจะพัฒนาให้ในส่วนอื่น เช่น ส่งข้อมูลไร้สาย ส่งแบบ IRDA
3. ควรพัฒนาให้มีการเก็บข้อมูลที่นับรอบได้และสามารถลบข้อมูลได้

เอกสารอ้างอิง

[1] ธีรบูลย์ หล่อวิเชียรรุ่ง และคณะ. **ปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 ด้วยโปรแกรมภาษาซี.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : อินโนตีฟ เอ็กเพอริเมนต์ จำกัด. 2547.

[2] อ.ประเสริฐ. **C Language for micro AVR .** พิมพ์ครั้งที่ 1.

[3] ธีรวัฒน์ ประกอบผล. การพัฒนาไมโครคอนโทรลเลอร์ด้วยภาษาซี. กรุงเทพฯ : สำนักพิมพ์ ส.ส.ท. 2546.

ประวัติผู้เขียนโครงงาน

ชื่อ มนตรี วิฑูรทัศน์
ภูมิลำเนา 1/14 ถ.เจริญธรรม ต.ท่าอิฐ อ.เมือง จ.อุตรดิตถ์ 53000
ประวัติการศึกษา

- จบการศึกษาระดับมัธยมศึกษาจาก โรงเรียนอุตรดิตถ์
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยนเรศวร

E-mail: chengnalux@hotmail.com

ชื่อ พัชมณฑ์ คเชนทร์ไพร
ภูมิลำนา 68/27 ถ.เอกาทศรถ อ.เมือง จ.พิษณุโลก 65000
ประวัติการศึกษา

- จบการศึกษาระดับมัธยมศึกษาจากโรงเรียนพิษณุโลกพิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยนเรศวร

E-mail: reejunk@hotmail.com