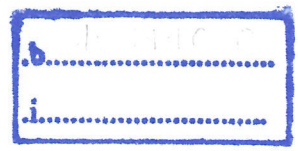
ระบบควบคุมมอเตอร์ด้วยไมโครคอนโทรลเลอร์ dsPIC

MOTOR CONTROL SYSTEM USING dsPIC MICROCONTROLLER

นายเจษฎา บางประเสริฐ
นายณัฐวุฒิ สีหสุทธิฤทธิ์
นางสาวปทุมรัตน์ เยาวบุตร



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2551

MOTOR CONTROL SYSTEM USING dsPIC MICROCONTROLLER

JESSADA BANGPRASERT

NATTHAWUT SRIHUSATHIRITH

PATHUMRAT YAOWABUT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENT ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUE OF TECHNOLOGY LADKRABANG

2008

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์	ระบบควบคุมมอเตอร์ด้วยไมโครคอนโทรลเลอร์ dsPIC		
	MOTOR CONTROL SYSTEM USING dsPIC MICROCONTROLLER		
นักศึกษาผู้จัดทำ	นายเจษฎา	บางประเสริฐ	รหัสนักศึกษา 48010147
	นายณัฐวุฒิ	สีหสุทธิฤทธิ์	รหัสนักศึกษา 48010281
	นางสาวปทุมรัตน์	เยาวบุตร	รหัสนักศึกษา 48010496
ปริญญา	วิศวกรรมศาสตรบัณฑิต		
สาขาวิชา	วิศวกรรมการวัดคุม		
ปีการศึกษา	2551		

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รองศาสตราจารย์ ดร.ทวีพล ซื่อสัตย์	

หัวข้อปริญญานิพนธ์	ระบบควบคุมมอเตอร์ด้วยไมโครคอนโทรลเลอร์ dsPIC		
	MOTOR CONTROL SYSTEM USING dsPIC MICROCONTROLLER		
นักศึกษาผู้จัดทำ	นายเจษฎา	บางประเสริฐ	รหัสนักศึกษา 48010147
	นายณัฐวุฒิ	สีหสุทธิฤทธิ์	รหัสนักศึกษา 48010281
	นางสาวปทุมรัตน์	เยาวบุตร	รหัสนักศึกษา 48010496
อาจารย์ที่ปรึกษา	รศ.ดร.ทวีพล	ซื่อสัตย์	
ปีการศึกษา	2551		

บทคัดย่อ

โครงงานนี้นำเสนอเครื่องควบคุมมอเตอร์ไฟฟ้ากระแสตรง โดยใช้ไมโครคอนโทรลเลอร์ตระกูล dsPIC (dsPIC30F2010) ไมโครคอนโทรลเลอร์เบอร์นี้เป็นหนึ่งในเทคโนโลยีสมองกลฝังตัวทำงานที่ 16 บิต และมีอุปกรณ์ภายในหลายชนิด เช่น ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล ตัวขับสัญญาณพัลซ์วิดท์มอดูเลเตอร์และอุปกรณ์ในการติดต่อสื่อสารระบบควบคุมนี้ ซึ่งใช้มอเตอร์กระแสตรงขนาด 24 โวลต์ โดยนำการควบคุมแบบลูปเปิดขับเคลื่อนโดยใช้ไอซีเบอร์ L298 N การสั่งการควบคุมสามารถทำได้ 2 ทาง คือ ผ่านทางสวิทช์ปุ่มกดและผ่านพอร์ตอนุกรม RS232C ในโครงงานนี้ได้ออกแบบข้อตกลงในการติดต่อสื่อสาร ให้สามารถใช้กับโปรแกรมคอมพิวเตอร์ได้ทุกชนิดโปรแกรม Labview และ Visual basic ซึ่งถูกใช้เป็นตัวอย่างในการประยุกต์ใช้งาน นอกจากนี้

โครงงานนี้ได้นำไปประยุกต์ใช้กับระบบการวัดแบบลำแสงเลเซอร์ตัดผ่าน เพื่อนำไปใช้สำหรับการวัดค่าทางเรขาคณิตกับวัตถุ ระบบควบคุมมอเตอร์นี้สามารถใช้ได้อย่างมีประสิทธิภาพ

Thesis Title	Motor Control System Using dsPIC Microcontroller	
Authors	Mr. Jessada	Bangprasert
	Mr.Natthawut	Srihusuthirith
	Miss.Pathumrat	Yaowabut
Thesis Advisor	Assoc.Prof.Dr.Taweepol	Suesut
Year	2008	

ABSTRACT

This project present a DC motor controller using a family of dsPIC microcontroller. (dsPIC30F2010) which is one of embedded technology. This microcontroller 16 bit consist of a variety of pheriperal devices such ADC (Analog to Digital) module , PWM (Pulse Width Modulator) function , serial communication and so on. With in this thesis , the open loop control system was applied to control 24 VDC motor driving by IC L298N. The manipulative control can be employed through push button swiches and RS232C. serial communication. In this hardware, The communication protocol , can support all computer programming. Visual basic and Labview were used as a case study.

Moreover, the motor controller has been applied to conveyor system for the 3D laser scanning instrument .The conveyor and the laser light sectioning can perform well for the geometric object measurement.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี เนื่องจากได้รับการชี้แนะทางด้านวิชาการและด้านเทคนิคที่เป็นประโยชน์อย่างยิ่งสำหรับอาจารย์ที่ปรึกษาปริญญานิพนธ์ รศ.ดร.ทวีพล ซื่อสัตย์ ที่ได้ให้คำแนะนำแก่ผู้วิจัยตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์และเครื่องมือต่าง ๆ ในการทำปริญญานิพนธ์ฉบับนี้ ผู้วิจัยรู้สึกซาบซึ้งและขอขอบพระคุณเป็นอย่างสูง ขอขอบพระคุณ รศ.ประภาษ อุคคกิมาพันธุ์ และอาจารย์กฤษณ์ เสมอพิทักษ์ และอาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่านและเพื่อนที่รักที่คอยให้คำแนะนำอันเป็นประโยชน์ และให้ความช่วยเหลือตลอดมา

ขอขอบพระคุณ คุณพ่อ คุณแม่ และคุณอาอันเป็นที่รักยิ่ง ที่สนับสนุนและเป็นแรงบันดาลใจในการทำปริญญานิพนธ์ฉบับนี้ คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

ในยุคสมัยปัจจุบัน การคอนโทรลมอเตอร์กระแสตรงหรือดีซีมอเตอร์ (DC Motor) จะพบได้โดยทั่วไป โดยเฉพาะในงานอุตสาหกรรมสมัยใหม่ส่วนมาก ด้วยความเจริญก้าวหน้าในส่วนของเทคโนโลยีทางไมโครอิเลคทรอนิคส์ (Microelectronic) และไมโครโปรเซสเซอร์ (Microprocessor) ทำให้การออกแบบระบบคอนโทรลมอเตอร์กระแสตรง และ การวิเคราะห์ปัญหาเป็นสิ่งที่น่าสนใจและมีความสำคัญมาก ระบบการคอนโทรลแบบดั้งเดิมมีพลังจักรกลที่สำคัญ คือ พวกมอเตอร์ไฟฟ้าต่าง ๆ ไฮโดรลิคและพวกเบรกและคลัช เป็นต้น แต่ด้วยความเจริญของเทคโนโลยีการสร้างแม่เหล็กถาวรที่มีคุณภาพสูงทำให้มอเตอร์กระแสตรงกลายเป็นพลังจักรกลที่สำคัญ ในส่วนระบบควบคุมเกือบทุกชนิดในปัจจุบัน ซึ่งพบเห็นได้ตั้งแต่การใช้มอเตอร์กระแสตรงในเครื่องใช้ไฟฟ้าภายในบ้านโดยปกติ การใช้ในรถยนต์ การใช้ในเครื่องคอมพิวเตอร์ การใช้ในหุ่นยนต์ตลอดถึงเครื่องจักรกลที่ทำงานแบบอัตโนมัติในอุตสาหกรรมต่าง ๆ ฯลฯ บวกกับความก้าวหน้าของตัวไมโครโปรเซสเซอร์ ซึ่งสามารถนำมาประยุกต์ใช้กับการคอนโทรลมอเตอร์กระแสตรงได้ จึงทำให้ได้รับความนิยมอย่างมากในปัจจุบัน

ซึ่งโครงงานนี้จะเป็นการสร้างและออกแบบระบบควบคุมของดีซีมอเตอร์ เพื่อขับสายพานโดยใช้ไมโครคอนโทรลเลอร์ตระกูล dsPIC โดยการควบคุมความเร็วและทิศทางของในระบบขับสายพาน จะผ่านทางฮาร์ดแวร์และซอฟตแวร์ โดยจะใช้โปรแกรมที่สร้างขึ้นจาก Visual Basic เป็นตัวติดต่อรับส่งค่าและ ในการควบคุมให้มอเตอร์หมุนตามความเร็วและไปยังทิศทางที่ต้องการนั้นจะผ่านวงจรขับมอเตอร์อีกวงจรในการขับมอเตอร์โดยใช้ไอซี L298N โดยตัว dsPIC30F2010 จะเป็นตัวสร้าง Pulse Width Modulate ในการขับมอเตอร์ที่อยู่ในระบบสายพานขับเคลื่อนนั้นเอง

1.2 วัตถุประสงค์

1. ศึกษาโครงสร้างของหลักการของมอเตอร์ไฟฟ้ากระแสตรง

2. ประยุกต์ใช้ไมโครคอนโทรลเลอร์ dsPIC30F2010 ควบคุมมอเตอร์ในการขับเคลื่อนระบบสายพาน

3. ควบคุมการทำงานของมอเตอร์เพื่อใช้ขับเคลื่อนสายพานได้ โดยสามารถปรับความเร็วและปรับทิศทางการหมุนได้

4. สามารถใช้ซอฟต์แวร์ควบคุมระบบขับเคลื่อนสายพานได้

1.3 ขอบเขต

1. ศึกษาโครงสร้างและหลักการ ตลอดจนถึงวิธีการใช้งานของตัวไมโครคอนโทรลเลอร์ dsPIC30F2010 เพื่อควบคุมมอเตอร์กระแสตรงได้

2. ใช้ dsPIC30F2010 เขียนโปรแกรมเพื่อควบคุมความเร็วและทิศทางของระบบขับขับเคลื่อนสายพานแบบ open loop โดยสามารถควบคุมได้ทั้งจากซอร์ฟแวร์ผ่านโปรแกรม Visual Basic 6 และ LABVIEW พร้อมทั้งจากผ่านทางฮาร์ดแวร์

3. สามารถเขียนโปรแกรมควบคุมการทำงานของระบบสายพาน ซึ่งกำหนดการทำงานโดยประมวลผลผ่าน dsPIC ได้

4. สร้างและออกแบบวงจรขับเคลื่อนเพื่อมาทำการทดลองงานจริง

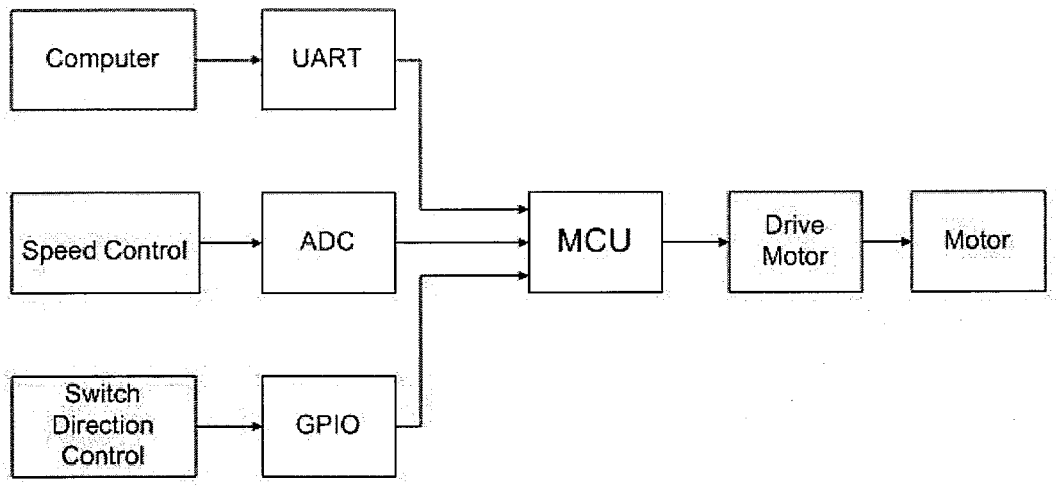
1.4 ขั้นตอนการศึกษา

1. เลือกหัวข้อในการทำโครงงาน
2. ศึกษาวัตถุประสงค์และขอบเขตการทำงานของโครงงาน
3. ศึกษาค้นคว้าข้อมูลที่เกี่ยวข้องกับโครงงาน
4. ศึกษาโครงสร้างของหลักการของมอเตอร์และหลักการการทำงานของ dsPIC30F2010
5. ทำการสร้างและออกแบบโครงงานเพื่อนำประยุกต์ใช้กับโครงงาน
6. ทำการทดลองและศึกษาผลทดลองที่ได้และนำผลทดลองที่ได้มาวิเคราะห์
7. สรุปผลการทดลองรวมถึงปัญหาและอุปสรรคในการทำงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

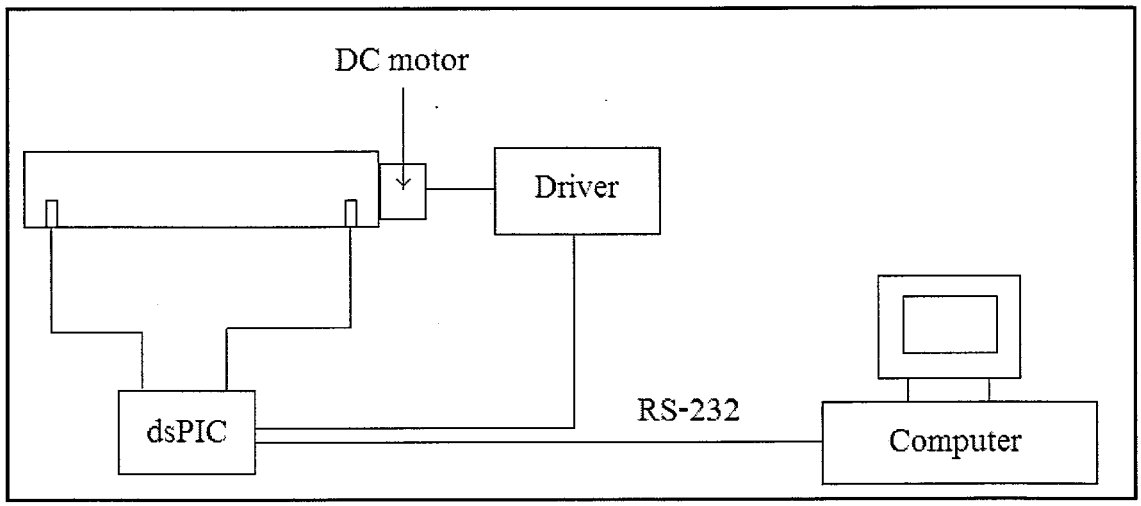
1. สามารถรู้หลักการทำงานของ dsPIC30F2010 ได้และสามารถเขียนโปรแกรมเพื่อใช้ควบคุมได้

2. สามารถนำความรู้ที่ได้จากการศึกษาตัวไมโครคอนโทรลเลอร์ dsPIC30F2010 นำไปประยุกต์การใช้งานควบคุมระบบขับเคลื่อนของสายพานได้



รูปที่ 1.1 ไดอะแกรมของระบบการควบคุม

ระบบขับสายพานนี้เป็นการประยุกต์ใช้ Microcontroller dsPIC ควบคุมความเร็วของสายพานผ่าน DC motor โดยการสั่งการสามารถทำได้จากคอมพิวเตอร์ผ่านทาง port RS232 ส่งเข้าโมดูล UART ของ dsPIC และจากแผงควบคุมที่เป็นสวิทซ์โดยควบคุมความเร็วจากการปรับเปลี่ยนค่าความต้านทาน ส่งเป็นสัญญาณอะนาล็อคเข้าสู่โมดูล ADC ของ dsPIC โปรแกรมที่เขียนไว้ใน MCU จะทำการประมวลผลเป็นสัญญาณ PWM ส่งให้กับ DC drive ต่อไป



รูปที่ 1.2 ภาพระบบโดยรวม

บทที่ 2
ทฤษฎีและความรู้ที่เกี่ยวข้อง

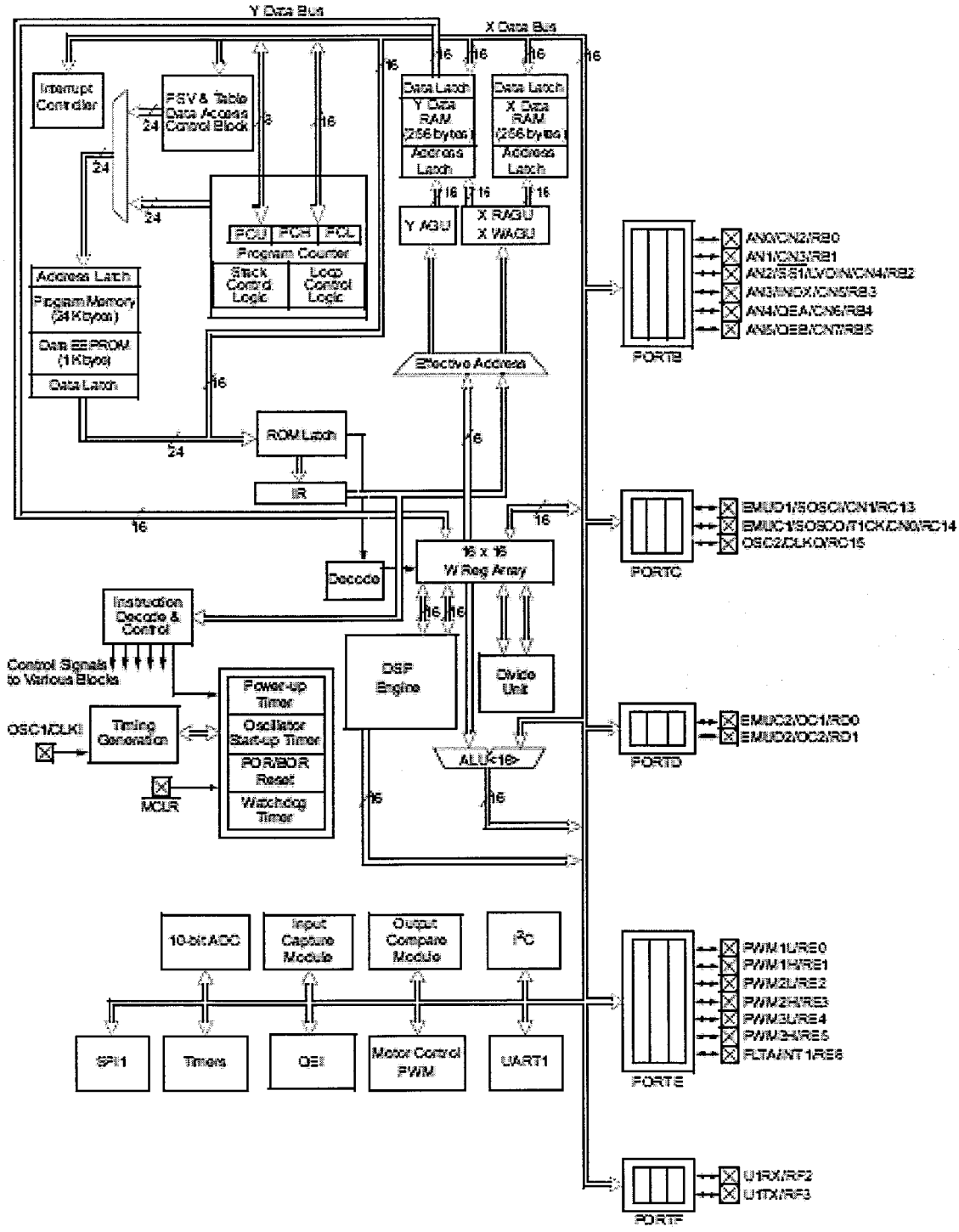
จากบทที่ได้กล่าวมาแล้วในบทที่ 1 แล้ว ก่อนที่จะทำการออกแบบและจัดสร้างระบบควบคุมความเร็วดีซีมอเตอร์นั้นจำเป็นต้องศึกษาองค์ประกอบต่าง ๆ ที่จำเป็นของระบบควบคุมที่สนใจให้เข้าใจเสียก่อน จึงได้พบว่าการควบคุมความเร็วมอเตอร์นั้นมีส่วนที่สำคัญอยู่หลายส่วน ดังนั้นในบทนี้จะศึกษาและอธิบายองค์ประกอบต่าง ๆ ที่สามารถนำไปใช้ในการควบคุมความเร็วดีซีมอเตอร์ซึ่งประกอบไปด้วย ไมโครคอนโทรลเลอร์ มอเตอร์กระแสตรง วงจรขับเคลื่อนมอเตอร์ การควบคุมความเร็วมอเตอร์รวมทั้งการอ่านค่าความเร็วมอเตอร์ที่ขับเคลื่อนในระบบสายพาน

2.1 ไมโครคอนโทรลเลอร์ตระกูล dsPIC

dsPIC เป็น MCU ซึ่งใช้การประมวลผลข้อมูลแบบ 16 บิต จากค่าย Microchips ซึ่งมีจุดเด่นในด้านของความสามารถในการประมวลผลข้อมูลสัญญาณแบบดิจิตอล สำหรับการไปประยุกต์ใช้ในงานควบคุมต่าง ๆ โดยในโครงสร้างภายในจะเป็นการผสมผสานระหว่างไมโครคอนโทรลเลอร์ (MCU) และวงจร DSP (Digital Signal Processing) รวมเข้าไว้ด้วยกัน หรืออาจเรียก MCU ตระกูล dsPIC ว่าเป็นDSC หรือ Digital Signal Controller ก็ได้ซึ่งในปัจจุบัน MCU ในตระกูล dsPIC ของ Microchips นั้นจะมีการผลิตออกมาจำหน่ายให้ผู้ใช้งานได้เลือกใช้งานกันอยู่มากมายหลายเบอร์ตามความเหมาะสมของงาน โดยปัจจุบัน (ตุลาคม 2548) dsPIC จะแบ่งออกเป็น 5 กลุ่ม ใหญ่ ๆ ด้วยกัน ได้แก่ dsPIC30F20xx ,dsPIC30F30xx ,dsPIC30F40xx,dsPIC30F50xx และ dsPIC30F60xx ซึ่งทุกเบอร์จะใช้โครงสร้างและสถาปัตยกรรมการประมวลผลแบบเดียวกันทั้งหมด แต่จะมีความแตกต่างกันในเรื่องของทรัพยากรภายใน เช่น ขนาดของหน่วยความจำใช้งานจำนวนของ Peripheral I/O แบบต่าง ๆ ซึ่งอาจมีการบรรจุไว้ในแต่ละเบอร์ด้วยจำนวนที่ไม่เท่ากัน ซึ่งในที่นี้จะขอกล่าวถึงเพียงเฉพาะข้อมูลในส่วนที่เป็นของ dsPIC30F2010 เท่านั้น

2.1.1 คุณสมบัติของ ไมโครคอนโทรลเลอร์ dsPIC

- ใช้สถาปัตยกรรมแบบ RISC โดยมี 84 คำสั่งมาตรฐาน รองรับการอ้างตำแหน่งแอดเดรสแบบต่าง ๆ ได้โดยอิสระ ซึ่งโดยรูปแบบโครงสร้างการจัดผังหน่วยความจำจะเป็นส่วนที่ดัดแปลงมาจากสถาปัตยกรรมของ “Harvard Architecture”
- ชุดคำสั่งใช้การอ้างแอดเดรสแบบ 24 บิต และการอ้างถึงข้อมูลขนาด 16 บิต
- มีหน่วยความจำโปรแกรมแบบ Flash ขนาด 12 KByte (4KWord) สามารถทำการลบและโปรแกรมซ้ำใหม่ได้กว่า 100,000 ครั้ง พร้อมระบบป้องกันการอ่าน
- มีหน่วยความจำ RAM ขนาด 512 Byte
- มีหน่วยความจำข้อมูลถาวรแบบ EEPROM ขนาด 1 KByte สามารถลบและเขียนซ้ำได้กว่า 1,000,000 ครั้ง และสามารถเก็บรักษาข้อมูลไว้ได้แม้ไม่ได้จ่ายไฟเลี้ยงให้ MCU
- มีรีจิสเตอร์ขนาด 16 บิต ให้ใช้งานจำนวน 16 ชุด
- สามารถประมวลผลด้วยความเร็วสูงสุดที่ 30 MIPS (30 ล้านคำสั่งต่อวินาที)
- รองรับสัญญาณนาฬิกาจากแหล่งกำเนิดภายนอก 0-40 MHz
- รองรับการใช้งานกับแหล่งกำเนิดความถี่แบบ XTAL ค่า 4-10 MHz
- มีวงจรคูณความถี่ภายในแบบ Phase-Lock-Loop โดยสามารถกำหนดค่าอัตราการคูณความถี่ได้ 3 ระดับ คือ 4 เท่า , 8 เท่า และ 16 เท่า
- รองรับการ Interrupt ได้ถึง 27 แหล่ง พร้อมสัญญาณ Interrupt จากภายนอก 3 แหล่ง และสามารถจัดระดับความสำคัญของการ Interrupt ได้ 8 ระดับ



รูปที่ 2.1 ไดอะแกรมการทำงานและส่วนประกอบทั้งหมดของ dsPIC30F2010

2.1.2 โครงสร้างทางฮาร์ดแวร์ของ dsPIC30F2010

dsPIC เบอร์ dsPIC30F2010 มีขาสำหรับต่อใช้งาน 28 ขา ดังภาพที่ 2.2 มีพอร์ตให้ใช้งานมากถึง 5 พอร์ต รวม 20 ขา

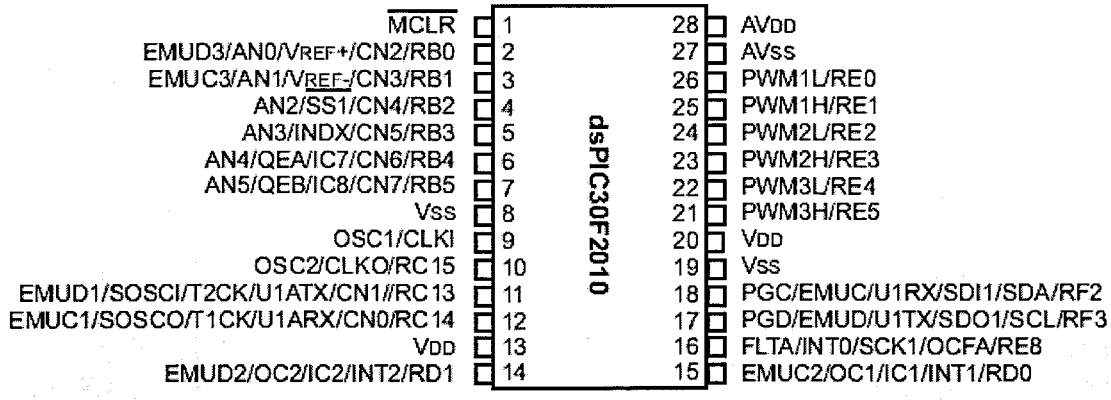
พอร์ต B มี 6 ขา คือ RB0-RB5 โดยทุกขาสามารถกำหนดให้เป็นพอร์ตอินพุตหรือเอาต์พุตได้และสามารถขับกระแสแบบซิงก์และซอร์สได้สูงถึง 25 mA

- พอร์ต C มี 3 ขา คือ RC13-RC15
- พอร์ต D มี 2 ขา คือ RD0-RD1
- พอร์ต E มี 2 ขา คือ RE0-RE5 และ RE8
- พอร์ต F มี 2 ขา คือ RF2 และ RF3
- ขาสัญญาณ I/O สามารถจ่ายกระแส (Source) และ รับกระแส (Sink) ได้มากถึง 25 mA
- มี Timer ขนาด 16 บิต จำนวน 3 ชุด และสามารถโปรแกรมใช้งานเป็น Timer แบบ 32 บิต ได้โดยใช้ Timer 16 บิต 2 ช่องรวมกัน
- มี Input Capture ขนาด 16 บิต จำนวน 4 ช่อง
- มี Output Compare/PWM ขนาด 16บิต จำนวน 2 ช่อง
- มีวงจรสื่อสารอนุกรมแบบ SPI จำนวน 1 ช่อง
- มีวงจรสื่อสารอนุกรมแบบ I2C จำนวน 1 ช่อง
- มีวงจรสื่อสารอนุกรมแบบ UART จำนวน 1 ช่อง
- มีวงจร DCPWM สำหรับใช้ควบคุมมอเตอร์ 3 ช่อง
- มีวงจรถอดรหัสแบบ QEIM ขนาด 16บิต จำนวน 1 ช่อง
- มีวงจร A/D ขนาด 10บิต จำนวน 6 ช่อง

ตารางที่ 2.1 แสดงรายละเอียดของ dsPIC เบอร์ต่าง ๆ

Device	Pins	Program Mem. Bytes/ Instructions	SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/Std PWM	Motor Control PWM	A/D 10-bit 500 Ksps	Quad Enc	UART	SPI™	I²C™	CAN
dsPIC30F2010	28	12K/4K	512	1024	3	4	2	6 ch	6 ch	Yes	1	1	1	–
dsPIC30F3010	28	24K/8K	1024	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	–
dsPIC30F4012	28	48K/16K	2048	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	1
dsPIC30F3011	40/44	24K/8K	1024	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	–
dsPIC30F4011	40/44	48K/16K	2048	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	1
dsPIC30F5015	64	66K/22K	2048	1024	5	4	4	8 ch	16 ch	Yes	1	2	1	1
dsPIC30F6010	80	144K/48K	8192	4096	5	8	8	8 ch	16 ch	Yes	2	2	1	2

2.1.3 ความรู้เบื้องต้นเกี่ยวกับ ไมโครคอนโทรลเลอร์ dsPIC30F2010



รูปที่ 2.2 แสดงไมโครคอนโทรลเลอร์ dsPIC30F2010

2.1.3.1 คุณสมบัติด้านการประมวลผล

- มีแอกคิวมูเลเตอร์ขนาด 40 บิต 2 ตัว รองรับการประมวลผลทางคณิตศาสตร์ได้อย่างดี

- มีหน่วยประมวลผลด้านการคูณและหารเลข 17 บิต ในรูปของฮาร์ดแวร์ จึงทำให้สามารถคูณเลขได้อย่างรวดเร็ว

- ทำการคูณเลข 16 บิต ได้ภายในสัญญาณนาฬิกาเพียง 1 ไซเคิล

2.1.3.2 คุณสมบัติของโมดูลฟังก์ชันพิเศษ

- มีไทเมอร์ขนาด 16 บิต จำนวน 3 ชุด สามารถโปรแกรมใช้งานเป็นไทเมอร์แบบ 32 บิต ได้โดยใช้ ไทเมอร์ 16 บิต 2 ช่อง รวมกัน

- มีโมดูลเปรียบเทียบข้อมูลและกำเนิดสัญญาณ PWM ขนาด 16 บิต จำนวน 2 ช่อง

- มีวงจรแปลงสัญญาณแอนะลอกเป็นดิจิตอลขนาด 10 บิต จำนวน 6 ช่อง

2.1.4 โมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล(ADC)

ไมโครคอลโทรลเลอร์ dsPIC30F2010 ได้ทำการบรรจุโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอลจำนวน 6 ช่อง ที่มีความละเอียดของการแปลงข้อมูล 10 บิต ทำให้ได้ข้อมูลดิจิตอลมีค่าตั้งแต่ 0-1023

คุณสมบัติโดยสรุปของโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล

- ใช้วิธีการแปลงสัญญาณแบบประมาณค่า (Successive Approximation)
- มีอัตราความเร็วในการสุ่มสัญญาณสูงสุด 500 กิโลแซมเปิลต่อวินาที (ksps)
- สามารถกำหนดให้ทำงานได้ขณะเข้าสู่โหมดสลีป (Sleep mode)
- สามารถกำหนดระดับแรงดันอ้างอิงได้ทั้งจากภายในผ่านทางขา AVDD กับ AVSS และภายนอกผ่านทางขา VREF+ หรือ VREF

การทำงานของโมดูลแปลงแอนะลอกเป็นดิจิตอลใน dsPIC30F2010 ซึ่งมีพอร์ตอินพุตแอนะลอกทั้งสิ้น 6 ขา คือ AN0 - AN5 โดยมี 2 ขา ที่สามารถใช้รับแรงดันอ้างอิงเพื่อขยายย่านของแรงดันอินพุตภายในโมดูลมีวงจรสุ่มและเก็บค่าสัญญาณ (Sample and Hold : S/H จำนวน 4 ชุด โดยทำงานร่วมกับส่วนควบคุมการมัลติเพล็กซ์สัญญาณอินพุตทำให้สามารถจัดสรรวงจร S/H ให้สามารถรองรับกับสัญญาณอินพุตแอนะลอกทั้ง 6 ช่อง ได้ด้วยความเร็วสูง สัญญาณที่ผ่านจากวงจร S/H จะถูกป้อนเข้าสู่วงจรแปลงสัญญาณแอนะลอกเป็นดิจิตอลแบบซักเซสซีฟแอ็ปพร็อกซีเมชั่นขนาด 10 บิต ข้อมูลที่ได้จากการแปลงจะถูกพักไว้ในหน่วยความจำแรมจากนั้นจะได้รับการจัดรูปแบบตามที่ผู้พัฒนาโปรแกรมกำหนด จากนั้นข้อมูลจะถูกถ่ายทอดลงบนบัสข้อมูลเพื่อส่งไปยังซีพียูต่อไป

2.1.5 โมดูลเปรียบเทียบข้อมูล (Output Compare : OC)

ใน dsPIC30F2010 ได้บรรจุโมดูลเปรียบเทียบข้อมูลไว้ 2 ชุด ซึ่งมีการทำงานหลักคือ เปรียบเทียบข้อมูล ที่ค่าฐานเวลาหนึ่ง ๆ กับข้อมูลในส่วนรีจิสเตอร์หากเท่ากันเมื่อใดก็จะกำหนดรูปสัญญาณพัลส์เดี่ยวหรือขบวนสัญญาณพัลส์ออกมาทางเอาต์พุต ซึ่งจะขึ้นอยู่กับการกำหนดโหมดทำงาน ดังนั้นจึงมักนำโมดูลมาใช้ในการกำเนิดสัญญาณพัลส์ โดยเฉพาะอย่างยิ่งกับการนำมาใช้สร้างสัญญาณ PWM

คุณสมบัติโดยสรุปของโมดูลเปรียบเทียบข้อมูล

- สามารถกำหนดสัญญาณ PWM
- ใช้ไทเมอร์ 2 หรือ 3 ร่วมในการทำงาน
- สามารถกำหนดเงื่อนไขเมื่อถึงคาบเวลาการนับ
- อินเตอร์รัปต์เนื่องจากถึงคาบเวลาการนับ
- มีอินพุตป้องกันความผิดพลาดในการสร้างสัญญาณ PWM

ในการกำเนิดสัญญาณของโมดูลเปรียบเทียบข้อมูล เริ่มต้นด้วยการกำหนดบิต OCM2 และ OCM0 ในรีเจสเตอร์ OCxCON เป็น “110” หรือ “111” ขึ้นอยู่กับว่าต้องการใช้งานขาอินพุต OCFA หรือ OCFB เพื่อป้องกันความผิดพลาดหรือไม่ เนื่องจากใน dsPIC30F2010 มีขาพอร์ต 20 ขามีโมดูล OC 2 ชุด จึงมีขาอินพุตเพื่อช่วยตรวจสอบข้อมูลความผิดพลาดเพียงขา

เดียวคือ OCFA เนื่องจาก OCFA จะสามารถทำงานได้จากโมดูล OC1-OC4 ในโหมดกำเนิดสัญญาณ PWM รีจีสเตอร์ OCxR จะถูกใช้ในการเก็บค่าดิวตี้ไซเกิลได้เพียงอย่างเดียว ดังนั้นในการเขียนข้อมูลเพื่อกำหนดค่าของดิวตี้ไซเกิลจึงต้องกระทำผ่านรีจีสเตอร์ OCxR แทน ทุกครั้งที่ค่าของรีจีสเตอร์คาบเวลา (Period Register) ซึ่งได้ค่ามาจากไทเมอร์ 2 หรือ 3 มีค่าตรงกับเงื่อนไขที่กำหนด ค่าดิวตี้ไซเกิลที่เก็บไว้ในรีจีสเตอร์ OCxR จะถูกโหลดมายัง OCxR เพื่อนำไปใช้สร้างสัญญาณ PWM ต่อไป พร้อมกันนั้นบิตแฟลก TyIF จะเซตเพื่อแจ้งสถานะที่เกิดขึ้น

$$T = [(PRx)+1] \times 4 \times OSC\ T \times (\text{TMRx prescaler value})$$

โดยที่ x คือ 2 หรือ 3 ขึ้นอยู่กับการเลือกไทเมอร์มาใช้งาน

PRx คือ รีจิสเตอร์เก็บค่าการนับของไทเมอร์ที่ถูกเลือก

OSC T คือ คาบเวลาของระบบ

TMRx prescaler value คือ ค่าปรีสเกลเลอร์ของไทเมอร์ที่ถูกเลือก

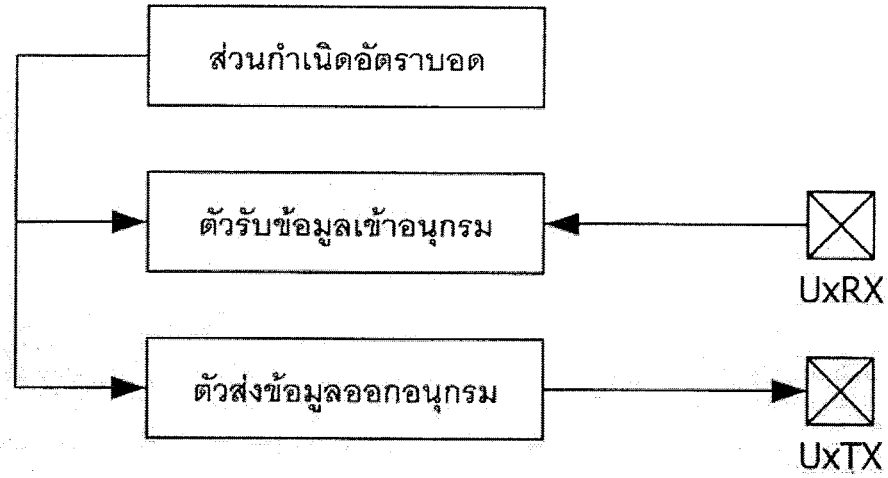
2.1.6 โมดูลสื่อสารข้อมูลอนุกรม (UART) ใน dsPIC

ในส่วนของการควบคุมจากคอมพิวเตอร์โมดูลนี้จะถูกใช้ติดต่อสื่อสารระหว่าง dsPIC กับคอมพิวเตอร์ผ่านพอร์ต RS-232

คุณสมบัติโดยสรุปของโมดูล UART

- สื่อสารข้อมูลแบบสองทิศทางในแบบ 8 และ 9 บิต
- สารมารถเลือกการสื่อสารข้อมูลแบบตรวจสอบบิตพาริตี้ คู่ หรือ คี่ และไม่ตรวจสอบบิตพาริตี้ สำหรับรูปแบบสื่อสารข้อมูลในแบบ 8 บิต มีบิตหยุด 1 หรือ 2 บิต
- มีส่วนกำเนิดอัตราบอดขนาด 16 บิต สำหรับกำหนดจังหวะและอัตราเร็วในการสื่อสารข้อมูลอนุกรมแยกอิสระเพ่อลดภาระการทำงานของโมดูลไทเมอร์
- กำเนิดอัตราบอดได้ตั้งแต่ 38 บิตต่อวินาที ถึง 1.875 เมกะบิตต่อวินาที
- บัฟเฟอร์ข้อมูลขาส่ง (TX) และขารับ (RX) ขนาด 4 เวิร์ดแยกส่วนกัน
- มีบิตแฟลกแจ้งข้อผิดพลาดในกรณีต่างๆของการสื่อสาร สามารถตรวจจับความผิดพลาดในการสื่อสารข้อมูลอนุกรม ได้แก่
  - ความผิดพลาดทางพาริตี้
  - รับข้อมูลไม่ทัน
  - เฟรมข้อมูลผิดพลาด
- สนับสนุนความสามารถในการอินเตอรัปต์แอดเดรส

- อินเตอร์รัปต์เวกเตอร์แยกตำแหน่งกันระหว่างการส่งและรับข้อมูล สามารถทำงานในโหมด Loopback



รูปที่ 2.3 ไดอะแกรมแสดงส่วนประกอบหลักของโมดูลUART

2.1.7 โครงสร้างทางโปรแกรมที่ควรทราบ

โครงสร้างทางโปรแกรมทีหรือ Programmer's model ของ dsPIC ที่นักพัฒนาต้องทราบ ซึ่งประกอบไปด้วย

1. รีจิสเตอร์หลักที่ใช้ในการทำงาน คือ รีจิสเตอร์ W (Working register) สำหรับใน dsPIC จะแตกต่างจากไมโครคอนโทรลเลอร์ PIC อย่างมากโดยรีจิสเตอร์ W ได้รับการจัดโครงสร้างเป็นอาเรย์ขนาด 16 บิต จึงทำให้สามารถรองรับทั้งข้อมูลค่าแอดเดรส หรือค่าของรีจิสเตอร์ใด ๆ ที่ต้องนำมาประมวลผล โดยใน dsPIC มีรีจิสเตอร์ W ให้ใช้งานถึง 16 ตัว ส่วนใหญ่ใช้ในการประมวลผลโดยตัวที่ใช้งานเป็นหลักคือ W0 ส่วนตัวที่ถูกนำไปใช้ในส่วนประมวลผลสัญญาณดิจิตอล

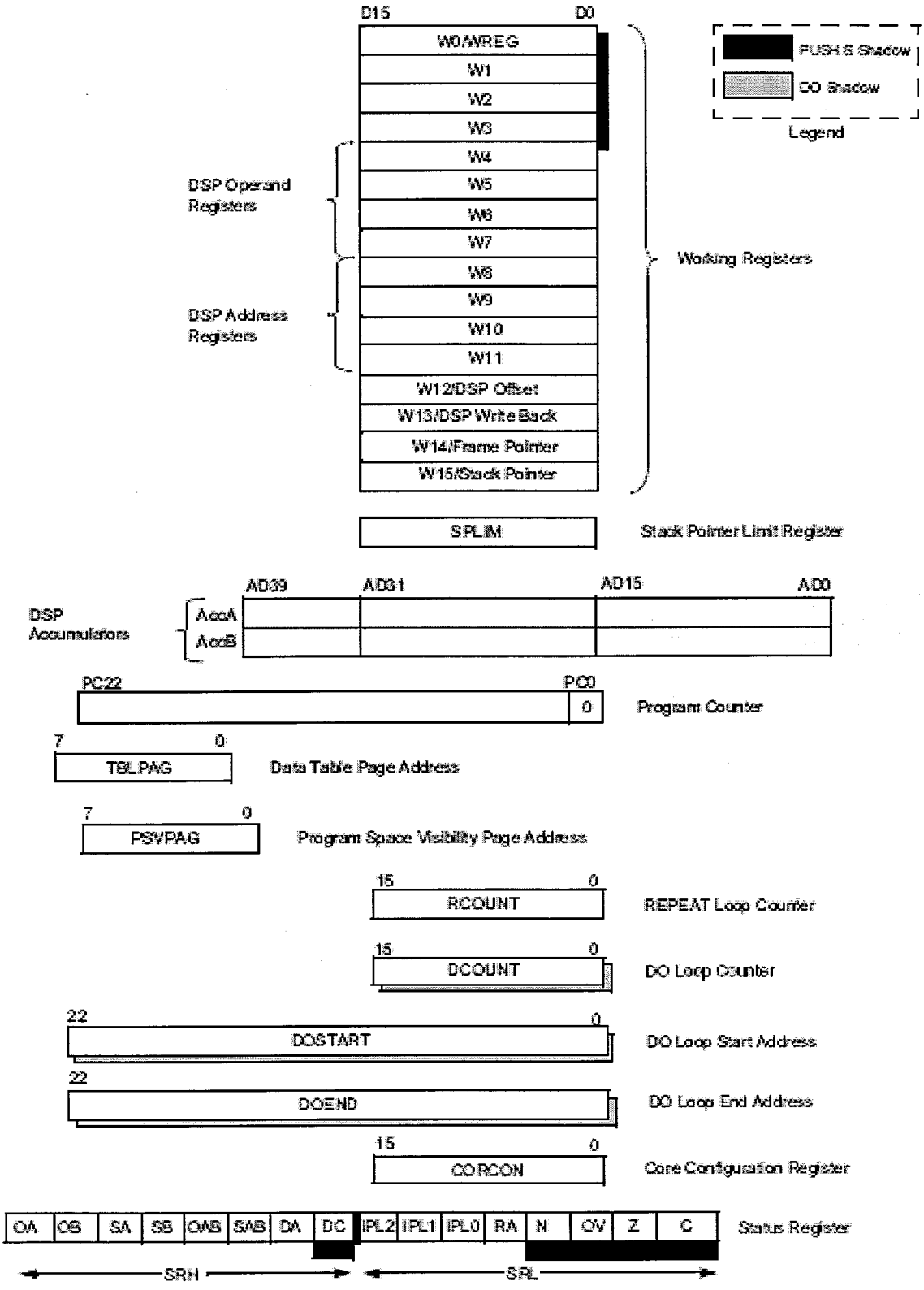
2. แอกคิวเลเตอร์ 40 บิต ใช้ในงานประมวลผลสัญญาณดิจิตอลเป็นหลัก

3. โปรแกรมเคาน์เตอร์ ขนาด 24 บิต นำมาใช้งานในการแจ้งแอดเดรส 23 บิต โดยไม่สนใจบิต MSB และ บิต LSB ต้องเป็น 0

4. รีจิสเตอร์หลัก อันประกอบด้วย

- STATUS ซึ่งใช้แสดงสถานการณ์ทำงานมีขนาด 16 บิต
- CORCON ใช้ควบคุมการทำงานของหน่วยประมวลผลกลาง มีขนาด 16 บิต
- TBLPAG เป็นรีจิสเตอร์กำหนดเพจของช่วง ตารางข้อมูลในหน่วยความจำโปรแกรมมีขนาด 8 บิต
- PSVPAG เป็นรีจิสเตอร์แสดงเพจแอดเดรสของพื้นที่โปรแกรม มีขนาด 8 บิต
- COUNT เป็นรีจิสเตอร์เก็บค่าตัวนับจำนวนรอบของลูปที่ทำซ้ำ
- DCOUNT เป็นรีจิสเตอร์เก็บค่าตัวนับจำนวนรอบของลูปที่ทำงาน
- DOSTART เป็นรีจิสเตอร์กำหนดแอดเดรสเริ่มต้นทำงานของโปรแกรมลูป
- DOEND เป็นรีจิสเตอร์กำหนดแอดเดรสปลายทางที่ต้องการทำโปรแกรม

สำหรับรีจิสเตอร์ DCOUNT , DOSTART และ DOEND เป็นรีจิสเตอร์เงา (shadow register) หมายความว่าเป็นรีจิสเตอร์ที่ถูกสร้างขึ้นชั่วคราวเพื่อเก็บค่าก่อนที่จะมีการถ่ายทอดออกไปทำงาน จึงไม่สามารถ เข้าถึงรีจิสเตอร์เหล่านี้ได้โดยตรง



รูปที่ 2.4 โครงสร้างทางโปรแกรมหรือ Programmer's model ของไมโครคอนโทรลเลอร์ dsPIC

2.1.7.1 รีจิสเตอร์ควบคุมเพิ่มเติมที่เกี่ยวข้องกับการทำงานของหน่วยประมวลผลกลางของ dsPIC

ประกอบด้วย

1. MODCON (Modulo Control Register)
2. XMODRST และ XMODEND (X Modulo Start Address Regieter)
3. YMODSRT และ YMODEND (Y Modulo Start Address Regieter และ Y Modulo End Address Regieter)
4. XBREV (X Modulo Bit-Reverse Register)
5. DISICNT (Disable Interrups Count Register)

2.1.7.2 หน่วยประมวลผลสัญญาณดิจิตอล (DSP Engine)

นี่คือจุดเด่นของไมโครคอนโทรเลอร์อนุกรมนี้ dsPIC ได้รับการออกแบบให้มีหน่วยคำนวณทางคณิตศาสตร์ที่มีประสิทธิภาพสูงมาก เพื่อจะได้รองรับการประมวลผลสัญญาณหรือข้อมูลขนาดใหญ่ทางดิจิตอล

องค์ประกอบสำคัญของหน่วยประมวลผลสัญญาณดิจิตอลใน dsPIC ได้แก่

- มัลติพลายเออร์หรือตัวคูณ 17 X 17 บิตความเร็วสูง
- ตัวยเลื่อนข้อมูลแบบบาร์เรล ขนาด 40 บิต
- แอกคิวมูเลเตอร์ขนาด 40 บิต ซึ่งมีด้วยกัน 2 ตัว คือ AccA และ AccB
- ส่วนประมวลผลลอจิกที่สามารถเลือกโหมดการทำงานได้
- ส่วนประมวลผลลอจิกในค่าอิ่มตัวที่สามารถเลือกโหมดการทำงานได้

ข้อมูลที่นำมาประมวลผลในหน่วยประมวลผลสัญญาณดิจิตอล (DSP) นี้มาได้จาก 2 แหล่ง

- เข้ามาโดยตรงจากรีจิสเตอร์ W4 ถึง W7
- จากบัสข้อมูลของหน่วยความจำข้อมูล X

ส่วนข้อมูลเอาต์พุตที่ได้จากหน่วย DSP นี้จะถูกส่งไปยัง 2 แหล่ง

- ส่งไปยังแอกคิวมูเลเตอร์เป้าหมาย
- หน่วยความจำข้อมูล X

แสดงถึงขอบเขตของข้อมูลในรีจิสเตอร์ที่มีขนาดต่างกัน ทั้งในแบบจำนวนเต็มและเลขเศษ

1. รูปแบบของตัวเลขที่ใช้ประมวลผลในหน่วย DSP และ ALU

เห็นถึงการกำหนดรูปแบบของข้อมูลเมื่อนำมาประมวลผลในหน่วย DSP และ ALU ทั้งในแบบเลขจำนวนเต็มและเลขเศษส่วนหรือทศนิยม ในกรณีเลขจำนวนเต็มจะคำนวณตามปกติ โดยบิต LSB จะมีค่าน้ำหนักประจำหลักเป็นและที่บิต MSB หากเป็นเลขคิดเครื่องหมายบิตนี้จะเป็นบิตแสดงเครื่องหมายว่าเป็นค่าบวกหรือลบ (ถ้าเป็น "0" หมายถึง เป็นค่าบวก ถ้าเป็น "1" หมายถึง เป็นค่าลบ) ส่วนน้ำหนักประจำหลักจะเท่ากับ 2^15 ในกรณีเป็นเลข 16 บิต , 2^31 กรณีเป็นเลข 32 และ 2^39 กรณีเป็นเลข 40 บิต

ในกรณีเป็นเลขเศษส่วนหรือทศนิยม บิตซ้ายสุดซึ่งปกติเป็นบิต MSB แต่ในกรณีเลขเศษส่วนหรือทศนิยมแล้วบิตซ้ายสุดจะเป็นเลขจำนวนเต็มหลักหน่วย และถ้าเป็นเลขแบบคิดเครื่องหมายบิตนี้ก็จะใช้ในการแสดงเครื่องหมายบิตนี้ก็จะใช้ในการแสดงเครื่องหมายด้วย นั่นคือ บิตซ้ายสุดจะมีค่าน้ำหนักประจำหลักเป็น 2^0 ในกรณีเป็นค่าบวก และ -2^0 ในกรณีเป็นเลขคิดเครื่องหมาย ส่วนบิตถัดมาทางขวาจะมีค่าน้ำหนักเป็น 2^ -1 ไล่เรียงไปจนถึง 2^ -15 ในกรณีเป็นเลข 16 บิต เป็นต้น

2. การรองรับการหารเลขใน dsPIC

คุณสมบัติเด่นอีกประการหนึ่งของไมโครคอนโทรลเลอร์ dsPIC คือ การหารเลข (divice) ในหน่วยคำนวณของไมโครคอนโทรลเลอร์ทั่วไปมักจะไม่รองรับการหารเลข ทำให้ต้องสร้างโปรแกรมย่อยขึ้นมาจัดการคำนวณในส่วนนี้เอง หรือถ้าจะรองรับการหารเลขได้อย่างเต็มรูปแบบ โดยแบ่งเป็น 5 รูปแบบคือ

1. การหารเศษส่วนหรือทศนิยม 16 บิตแบบคิดเครื่องหมาย โดยใช้คำสั่ง DIVF
2. การหารเลขตัวตั้ง 32 บิต ด้วยตัวหาร 16 บิต แบบคิดเครื่องหมายโดยใช้คำสั่ง DIV.SD
3. การหารเลขตัวตั้ง 32 บิต ด้วยตัวหาร 16 บิต แบบไม่คิดเครื่องหมายโดยใช้คำสั่ง DIV.UD
4. การหารเลขจำนวนเต็ม 16 บิต แบบคิดเครื่องหมาย โดยใช้คำสั่ง DIV.SW
5. การหารเลขจำนวนเต็ม 16 บิต แบบไม่คิดเครื่องหมาย โดยใช้คำสั่ง DIV.UW

ผลหารของทุกคำสั่งหารเลขจะเก็บไว้ในรีจิสเตอร์ W0 ส่วนในการหารเก็บไว้รีจิสเตอร์ W1 ในขณะที่ค่าของตัวหาร 16 บิต สามารถเก็บไว้ในรีจิสเตอร์ W ตัวใดก็ได้ เช่น เดียวกับตัวตั้งแต่ถ้าหากเป็นตัวตั้ง 32 บิต จะต้องนำค่าไปเก็บไว้ในรีจิสเตอร์ W จำนวน 2 ตัว ที่อยู่ติดกัน

2.1.8 การจัดสรรหน่วยความจำของ dsPIC30F2010

ในการจัดสรรพื้นที่หน่วยความจำโปรแกรมของ dsPIC30F2010 จะเห็นว่ามีการแบ่งออกเป็น 2 พื้นที่ใหญ่ ๆ คือ

พื้นที่สำหรับใช้งาน (User Memory Space)

พื้นที่สำหรับกำหนดค่าหรือคอนฟิกูเรชั่น (Configuration Memory Space)

ในช่วงแรกของพื้นที่สำหรับใช้งานจะเป็นพื้นที่ของตารางเวกเตอร์ ไม่ว่าจะเป็นรีเซตเวกเตอร์ เวกเตอร์ของการตรวจจับความผิดพลาดแบบต่าง ๆ อินเตอร์รัปต์เวกเตอร์ที่มีมากถึง 54 ตำแหน่ง (เริ่มต้นที่ 0x000014) แต่สำหรับ dsPIC30F2010 จะมีใช้งาน 48 ตำแหน่ง ส่วนของหน่วยความจำโปรแกรมที่ใช้เก็บโปรแกรมของผู้พัฒนาขนาด 4 กิโลเวิร์ด ถูกจัดสรรไว้ที่แอดเดรส 0x000100 และ ที่ช่วงแอดเดรส 0x7FFC00 ถึง 0x7FFFFE จัดสรรไว้ สำหรับหน่วยความจำข้อมูลอีอีพรอมความจุ 1 กิโลไบต์

ส่วนในพื้นที่สำหรับกำหนดค่า หรือ คอนฟิกูเรชั่นมีแอดเดรสเริ่มต้นที่ 0x800000 ถึง 0xFFFFE โดยพื้นที่ส่วนใหญ่จะเป็นพื้นที่สำรองไว้ที่สำคัญคือในช่วงแอดเดรส 0xf80000 ถึง 0xF8000E จะเป็นพื้นที่ของรีจิสเตอร์กำหนดค่าทางฮาร์ดแวร์หรือคอนฟิกูเรชั่น (Device Configuration Register)

2.1.9 การรีเซตในไมโครคอนโทรลเลอร์ dsPIC30F2010

dsPIC30F2010 รองรับกระบวนการรีเซตอยู่หลายรูปแบบดังนี้

1. เพาเวอร์-ออนรีเซต (Power Reset : POR) เป็นการรีเซตจากการจ่ายไฟเลี้ยงใหม่
2. การรีเซตที่ขา MCLR ในขณะทำงานปกติ (EXTR)
3. การรีเซตที่ขา MCLR ในโหมดสลีป (SLEEP หรือ IDLE)
4. การรีเซตเนื่องจากวอตช์ด็อกไทรเมอร์ในขณะปกติ (WDTR)
5. บราวเอาต์รีเซต (Programmable Brown-out Reset : BOR) เป็นการรีเซตเนื่องจากระดับไฟเลี้ยงลดต่ำกว่าค่าที่กำหนดไว้
6. การรีเซตเนื่องจากคำสั่ง RESET (SWR)
7. การรีเซตเนื่องจากการอินเตอร์รัปต์ของซีพียู (TRAPR)
8. การรีเซตเนื่องจากออปโค้ดผิดพลาดหรือจากการใช้งานรีจิสเตอร์ W ตัว ที่ยังไม่พร้อมทำงานไปทำงานเป็นตัวชี้แอดเดรส (legal opcode or by using an uninitialized W register as an address pointer : IOPUWR)

2.1.10 รีจิสเตอร์กำหนดค่าทางฮาร์ดแวร์ (Device Configuration Register)

ใน dsPIC30F2010 มีรีจิสเตอร์พิเศษที่ใช้ในการกำหนดค่าทางฮาร์ดแวร์ หรือ Device Configuration Registers ขนาด 24 บิตอยู่ 4 ตัว

1. FOSC รีจิสเตอร์กำหนดการทำงานของวงจรกำเนิดสัญญาณนาฬิกา
2. FWDT รีจิสเตอร์กำหนดการทำงานของวอตช์ด็อกไทเมอร์
3. FBORPOR รีจิสเตอร์กำหนดการทำงานของวงจรบราวน์เอาต์รีเซต (BOR) และ วงจรเพาเวอร์ออนนรีเซต (POR)
4. FGS รีจิสเตอร์กำหนดการป้องกันหน่วยความจำ

การกำหนดค่าลงในรีจิสเตอร์ทั้ง 4 ตัว นี้ซึ่งสามารถกระทำได้ในขณะทำงานหรือจากการโปรแกรมผ่านกระบวนการ ICSP หรือจากเครื่องโปรแกรมภายนอก

2.1.10.1 พอร์ตอินพุตเอาต์พุตของไมโครคอนเลอร์ dsPIC

ด้วยคุณสมบัติของไมโครคอนเลอร์สมัยใหม่ขาต่อใช้งานหรือพอร์ตอินพุต พอร์ตเอาต์พุตนั้นมักมีความสามารถพิเศษร่วมอยู่ด้วย โดยไม่ได้เป็นแค่เพียงพอร์ตอินพุตเอาต์พุตอย่างใดอย่างหนึ่งเพียงอย่างเดียวอีกต่อไป dsPIC ก็เช่นกันในแต่ละขาพอร์ตของ dsPIC ก็สามารถกำหนดให้ทำงานได้ทั้งเป็นพอร์ตอินพุตเอาต์พุตและขาสำหรับเชื่อมต่อใช้งานโมดูลพิเศษ dsPIC แบ่งการทำงานเป็น 2 ส่วน คือ โมดูลพอร์ต (Port module) และ เซลอินพุตเอาต์พุต (I/O Cell) โดยโมดูลพอร์ตจะใช้การกำหนดทิศทางของสัญญาณและเป็นที่พักข้อมูลชั่วคราว ก่อนรับเข้าหรือส่งออก ส่วนเซลอินพุตเอาต์พุตเป็นส่วนที่เชื่อมต่อกับขาพอร์ตจริง บรรจุวงจรบัฟเฟอร์เพื่อจัดการสัญญาณที่เข้าออกขาพอร์ตให้มีเสถียรภาพ

2.1.10.2 รีจิสเตอร์ควบคุมขาอินพุตเอาต์พุต

ทุกขาพอร์ตอินพุตเอาต์พุตของ dsPIC มีรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานโดยตรง 3 ตัว คือ

TRISx รีจิสเตอร์กำหนดทิศทางข้อมูลของขาพอร์ต

PORTx รีจิสเตอร์ข้อมูลของขาพอร์ต

LATx รีจิสเตอร์แลตช์ข้อมูลของขาพอร์ต

x เป็นค่าหรือตัวอักษรที่ใช้แทนตำแหน่งของขาพอร์ตใด ๆ เนื่องจาก dsPIC มีกลุ่มของขาพอร์ตอินพุตเอาต์พุตจำนวนมาก

2.1.11 การติดต่อกับ ICD (In- Circuit Debuger) เพื่อตรวจสอบการทำงานหรือดีบักในวงจรของ dsPIC30F2010

dsPIC30F2010 สามารถรองรับการตรวจสอบการทำงาน หรือ การดีบักด้วยเครื่องมือทางฮาร์ดแวร์ภายนอกที่เรียกว่า In-Circuit Debugger หรือ ICD ได้

ซึ่งใน dsPIC30F2010 ต้องจัดสรรความสามารถบางส่วนเพื่อนำไปใช้ร่วมกับการดีบักเกอร์ อันประกอบด้วย

- หน่วยความจำข้อมูลแรม 80 ไบต์ แรก
- ขาพอร์ตสำหรับติดต่อ 2 ขา ซึ่งสามารถเลือกได้คู่ตามที่ MPLAB IDE กำหนด ดังนี้

EMUD-EMUC (ขา 17 และ 18)

EMUD1-EMUC1 (ขา 11 และ 12)

EMUD2-EMUC2 (ขา 14 และ 15)

EMUD3-EMUC3 (ขา 2 และ 3)

ขา EMUD (Emulation/Debug Data) เป็นขาข้อมูลของการดีบัก ส่วนขา EMUC (Emulation/Debug Clock) เป็นขาสัญญาณนาฬิกาของการดีบักโดยขาพอร์ตทั้งสองจะเชื่อมต่อกับ ICD เพื่อรับและส่งข้อมูลร่วมกับขาแรงดันอีก 3 ขา คือ ขารีเซตและจ่ายแรงดันสำหรับโปรแกรม ซึ่งใช้ขา MCLR , ขาไฟเลี้ยง ซึ่งปกติเท่ากับ + 5v และขากราวด์ รวมทั้งสิ้น 5 ขา

แต่ถ้าหากใช้สัญญาณคู่อื่นที่เหลือทั้งสามคู่จะต้องใช้ขาสัญญาณเพิ่มอีก 2 ขา เพื่อใช้ในการเลือกขาพอร์ตดีบัก , 3 ขา สำหรับจ่ายแรงดัน (MCLR,+5V,GND) และ 2 ขา สำหรับติดต่อกับ ICD คือ EMUDx และ EMUCx รวมทั้งสิ้น 7 ขา

2.2 มอเตอร์กระแสตรง (DC MOTOR)

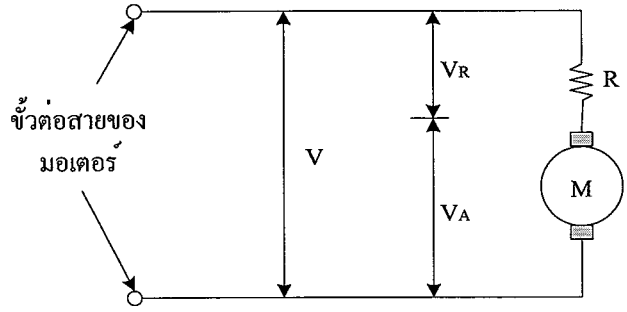
หลักการทำงานของมอเตอร์กระแสตรง

เมื่อมีการผ่านกระแสไฟฟ้าเข้าไปยังขดลวด ในสนามแม่เหล็กจะทำให้เกิดแรงแม่เหล็กซึ่งมีสัดส่วนของแรงขึ้นกับกระแสแรงของสนามแม่เหล็ก โดยแรงจะเกิดขึ้นเป็นมุมฉากกับกระแสและในสนามแม่เหล็ก ในขณะที่ทิศทางของแรงกลับตรงกันข้ามกัน ถ้าหากกระแสของสนามแม่เหล็กไหลย้อนกลับจะทำให้เกิดการเปลี่ยนแปลงของกระแสและสนามแม่เหล็ก เป็นผลทำให้ทิศทางของแรงเปลี่ยนไป ด้วยคุณสมบัตินี้ทำให้มอเตอร์กระแสตรงกลับทิศทางการหมุนได้

สนามแม่เหล็กของมอเตอร์ส่วนหนึ่งเกิดขึ้น จากแม่เหล็กถาวรซึ่งจะถูกยึดติดกับแผ่นเหล็กหรือเหล็กกล้า โดยปกติส่วนนี้จะเป็นส่วนที่ยึดอยู่กับที่และขดลวดเหนี่ยวนำจะพันอยู่กับส่วนที่เป็นแกนหมุนของมอเตอร์

2.2.1 คุณสมบัติของมอเตอร์กระแสตรง

ในการอธิบายคุณสมบัติของมอเตอร์กระแสตรงให้ละเอียดนั้น ต้องพิจารณาแรงดันที่ป้อนและความต้านทานของโรเตอร์ด้วย วงจรภายในของมอเตอร์เขียนได้ดังรูปที่ 2.5



รูปที่ 2.5 วงจรภายในของมอเตอร์กระแสตรง

โดยสมมติให้ทุ่นโรเตอร์ไม่มีความต้านทานอยู่เลยต่ออนุกรมกับความต้านทาน ซึ่งในที่นี้ก็คือความต้านทานของขดลวดนั่นเอง แรงดันที่ขั้วต่อสายของมอเตอร์ก็คือผลบวกระหว่างแรงดันที่ทุ่นโรเตอร์ (V_A) และ แรงดันตกคร่อมความต้านทานขดลวด (V_R)

แรงดัน V_A ถูกเรียกว่า แรงเคลื่อนเหนี่ยวนำป้อนกลับ (BACK EMF) ซึ่งเกิดขึ้นในโรเตอร์ขณะที่หมุนแรงดันที่เกิดขึ้นนี้เป็นไปตามกฎของการเหนี่ยวนำแม่เหล็กไฟฟ้า จากการเคลื่อนที่ของตัวนำในสนามแม่เหล็กสัมพันธ์กับแรงเคลื่อนเหนี่ยวนำแม่เหล็ก และความเร็วในการเคลื่อนที่ของตัวนำแรงดันที่เกิดขึ้น ซึ่งจะมีขั้วตรงกันข้ามกับแรงดันที่ป้อนให้กับมอเตอร์และแปรผันตรงกับ

ส่วนความเร็วในการหมุน ผลบวกของแรงดันที่ทุ่นโรเตอร์ (V_A) และแรงดันตกคร่อมขดลวด (V_R) ต้องเท่ากับแรงดันที่ป้อนให้กับมอเตอร์ (V)

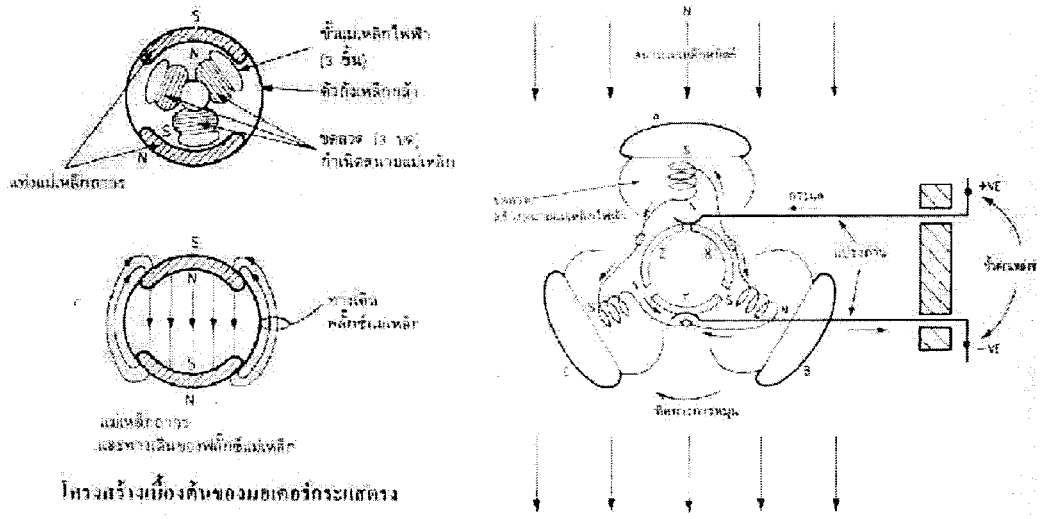
$$V = V_A + V_R \qquad (V)$$

เมื่อพิจารณาตั้งแต่มอเตอร์หยุดนิ่ง ความเร็วมีค่าเป็นศูนย์ ดังนั้น $V_A = 0$, $V_R = V$ กระแสที่ไหลในมอเตอร์หาได้จาก

$$I = V_R / R \qquad (A)$$

เมื่อมอเตอร์เริ่มหมุนจะมีความเร็วและ V_A เพิ่มขึ้นเป็นเส้นตรงตามความเร็ว V_R ซึ่งมีค่าเท่ากับความแตกต่างระหว่าง V_A และ V จะเริ่มลดลงกระแส I ก็จะเริ่มลดลงเช่นกันขณะที่มอเตอร์ยังมีความเร่งอยู่ ซึ่งค่าความเร็วจะเพิ่มขึ้นแรงบิดจะลดลงจนกว่าจะถึงจุดซึ่งแรงบิดจองมอเตอร์รับภาระโหลดได้สมดุลพอดี ขณะที่มอเตอร์ไม่มีโหลดและหมุนอย่างอิสระจะมีเพียงค่าความฝืดของแบริ่งและแรงต้านอากาศทำให้ V_A เกือบเท่ากับค่า V

มอเตอร์กระแสตรงจะมีหลักการทำงาน โดยวิธีการผ่านกระแสให้กับขดลวดในตัวตัดในขดลวดสนามแม่เหล็ก ซึ่งจะทำให้เกิดแรงแม่เหล็กโดยส่วนของแรงนี้จะขึ้นอยู่กับกระแสและกำลังของสนามแม่เหล็ก



รูปที่ 2.6 แสดงโครงสร้างทั่วไปของมอเตอร์กระแสตรง

จากในรูปทางเดินของฟลั๊กซ์แม่เหล็กและสนามแม่เหล็ก จะเกิดจากแท่งแม่เหล็กเฟอร์ไรต์ 2 ชิ้น ที่ขึ้นรูปเป็นแบบโค้งยึดติดกับตัวถังได้พอดีเพื่อที่จะให้เส้นแรงแม่เหล็กวิ่งเข้าสู่ใจกลางของมอเตอร์ได้ ดังนั้นความเข้มของแม่เหล็กจะขึ้นอยู่กับขนาดความหนาของแม่เหล็กด้วย ซึ่งส่งผลให้ฟลั๊กซ์แม่เหล็กวิ่งไปบนตัวถังโลหะ กระแสไฟฟ้าในขดลวดที่พันกับทุ่นโรเตอร์ก็จะทำให้เกิดสนามแม่เหล็กไฟฟ้าและต้านกับสนามแม่เหล็กถาวร จึงเกิดเป็นแรงบิดเพื่อที่จะหมุนทุ่นโรเตอร์ ให้ไปในทิศทางเดียวกันกับทิศทางของสนามแม่เหล็กที่มีแรงมากกว่ากระแสก็จะไหลผ่านไปยังตัวทุ่นโรเตอร์ โดยผ่านแปรงถ่านซึ่งจะสัมผัสกับแหวนตัวนำในทุ่นโรเตอร์และแหวนคอมมิวเตเตอร์ ซึ่งจะถูกแบ่งออกเป็น 3 เซกเมนต์ เพื่อที่จะทำหน้าที่นำกระแสเข้าขดลวดนั้นเอง

2.2.2 ทฤษฎีหลักการของมอเตอร์

2.2.2.1 หลักการของมอเตอร์ไฟฟ้ากระแสตรง

ทิศทางของแรงที่เกิดขึ้นในตัวนำ เมื่อมีกระแสไฟฟ้าผ่านตัวนำซึ่งอยู่ในสนามแม่เหล็กจะเป็นไปตามกฎมือซ้ายของเฟลมมิ่ง ดังนั้นจะได้

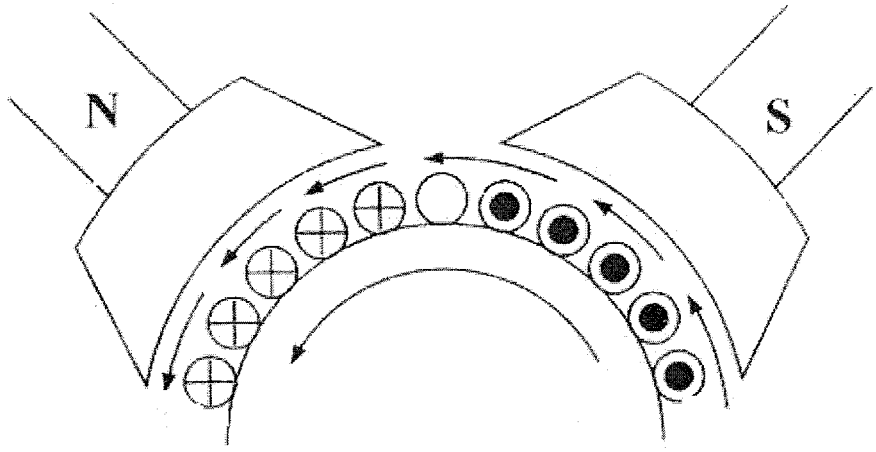
$$F = BLI \qquad (N)$$

โดยที่ :

	F	เป็นแรงที่เกิดบนตัวนำ	(N)
	B	เป็นความหนาแน่นของเส้นแรงแม่เหล็ก	$(\frac{Wb}{m^2})$
	L	เป็นความยาวของตัวนำ	(m)
	I	เป็นกระแสไฟฟ้าที่ไหลในตัวนำ	(A)

พิจารณาภาพที่ 2.7 แสดงมอเตอร์ไฟฟ้ากระแสตรงหลายขั้ว เมื่อเกิดการกระตุ้นที่ขดลวดสนามและตัวนำบนอาร์เมเจอร์ได้รับกระแสไฟฟ้าจากแหล่งจ่ายหลักซึ่งทำให้เกิดแรงในแนวตั้งฉากกับเส้นแรงแม่เหล็กและกระแสไหลผ่านในตัวนำนั้น ๆ ดังนั้นเมื่อมีกระแสไหลในขดลวดตัวนำที่พันอยู่บนแกนเหล็กอาร์เมเจอร์ ซึ่งจะเกิดเส้นแรงแม่เหล็กขึ้นรอบๆตัวนำและจะเกิดปฏิกิริยากับเส้นแรงแม่เหล็กที่เกิดจากขั้วแม่เหล็กของมอเตอร์ ทำให้เกิดแรงผลักบนตัวนำอาร์เมเจอร์จึงหมุนได้

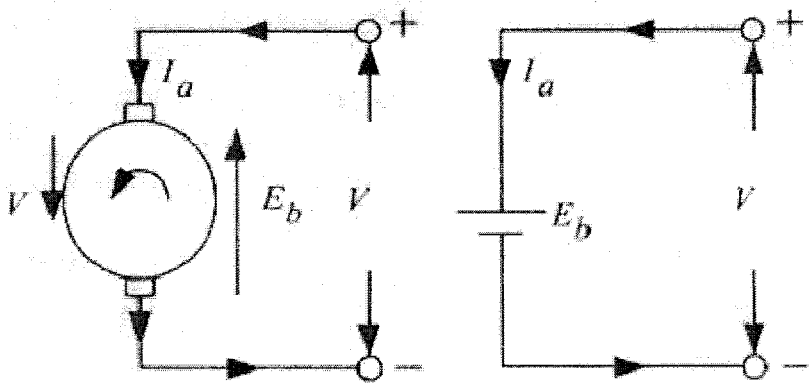
สมมติว่าตัวนำภายใต้ขั้ว N นำกระแสในทิศทางลงล่าง (+) ภายใต้ขั้ว S นำกระแสในทิศทางขึ้นบน (-) ใช้กฎมือซ้ายของเฟลมมิ่งในการพิจารณาทิศทางของแรงในแต่ละตัวนำซึ่งแสดงด้วยหัวลูกศรที่อยู่ด้านบนของแต่ละตัวนำ โดยแต่ละตัวนำมีแรง F ทำให้อาร์เมเจอร์หมุนในทิศทางทวนเข็มนาฬิกาแรงนี้ทำให้เกิดแรงบิดเริ่มหมุน



รูปที่ 2.7 มอเตอร์ไฟฟ้ากระแสตรงหลายขั้วพื้นฐาน

2.2.2.2 Significance of the back e.m.f

เมื่ออาร์เมเจอร์หมุนตัวนำที่อาร์เมเจอร์จะตัดกับเส้นแรงแม่เหล็ก ทำให้เกิดแรงเคลื่อนไฟฟ้าเหนี่ยวนำขึ้นในอาร์เมเจอร์ โดยมีทิศทางเป็นไปตามกฎมือขวาของเฟลมมิ่งซึ่งจะตรงกันข้ามกับแรงเคลื่อนไฟฟ้าที่จ่ายให้กับมอเตอร์เรียกแรงเคลื่อนไฟฟ้านี้ ซึ่งเรียกว่าแรงเคลื่อนไฟฟ้าต้านกลับ (back e.m.f) ใช้ตัวย่อ E_b วงจรสมมูลของมอเตอร์ไฟฟ้ากระแสตรงแสดงในรูปที่ 2.2.2.2



รูปที่ 2.8 ทิศทางของแรงเคลื่อนไฟฟ้าต้านกลับ

เมื่ออาร์เมเจอร์หมุนจะเกิดแรงเคลื่อนไฟฟ้าต้านกลับ จะได้

$$I_a = \frac{(V - E_b)}{R_a} \qquad (A) \qquad (2.2)$$

$$E_b = \phi ZN \times \left(\frac{P}{A}\right) \qquad (V) \qquad (2.3)$$

โดยที่ : R_a = ความต้านทานในวงจรของอาร์เมเจอร์ (Ω)

N = เป็นความเร็วรอบของมอเตอร์ (r.p.m)

พิจารณาสมการ (2.3) พบว่าแรงเคลื่อนไฟฟ้าต้านกลับจะขึ้นอยู่กับความเร็วรอบของอาร์เมเจอร์ ถ้าความเร็วรอบสูงแรงเคลื่อนไฟฟ้าต้านกลับก็จะมากกระแสที่ไหลในอาร์เมเจอร์ (I_a) จากสมการที่ (2.2) จะมีค่าน้อย ถ้าความเร็วรอบต่ำแรงเคลื่อนไฟฟ้าต้านกลับก็จะน้อย I_a ก็จะมาก แรงบิดจะมีค่าสูงเมื่อมีความเร็วรอบสูง

2.2.2.3 สมการแรงเคลื่อนไฟฟ้าของมอเตอร์ (Voltage Equation of a motor)

การศึกษาลักษณะสมบัติ (performance) ของมอเตอร์ จำเป็นต้องทราบถึงสมการพื้นฐานของมอเตอร์ก่อน สมการดังกล่าวคือสมการแรงเคลื่อนไฟฟ้าสามารถเขียนได้ดังนี้

$$V = E_b + I_a R_a \qquad (2.4)$$

จากสมการที่ (2.4) คูณด้วย I_a ตลอด จะได้

$$VI_a = E_b I_a + (I_a)^2 R_a \qquad (2.5)$$

โดยที่ : V = แรงเคลื่อนไฟฟ้าที่จ่ายให้กับวงจรของมอเตอร์ (V)

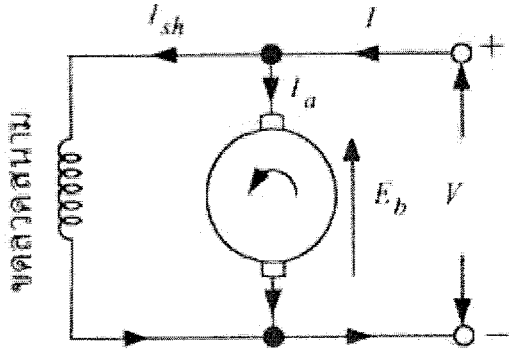
E_b = แรงเคลื่อนไฟฟ้าต้านกลับที่เกิดขึ้นในอาร์เมเจอร์ (V)

$I_a R_a$ = แรงเคลื่อนไฟฟ้าตกคร่อมที่ความต้านทานของอาร์เมเจอร์ (V)

VI_a = กำลังไฟฟ้าที่จ่ายให้กับอาร์เมเจอร์ (electrical input in armature) (V)

$E_b I_a$ = กำลังไฟฟ้าในอาร์เมเจอร์ที่สมมูลกับกำลังทางกลที่เกิดในอาร์เมเจอร์, P_m (W)

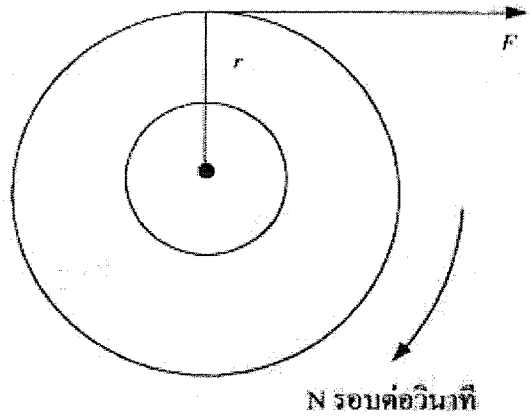
$(I_a)^2 R_a$ = กำลังสูญเสียที่ขดลวดในอาร์เมเจอร์ (W)



รูปที่ 2.9 วงจรมอเตอร์กระแสตรงแบบขนาน

2.2.2.4 แรงบิด (Torque)

แรงบิดคือโมเมนต์การบิดตัวของแรงเหนือแกนที่พิจารณาซึ่ง สามารถคำนวณได้โดยการคูณแรงกับรัศมีที่แรงกระทำ พิจาณาภาพที่ 2.7 แสดงพูเลย์ที่มีรัศมี r เมตร มีแรงกระทำในแนวเส้นรอบวง F นิวตันทำให้พูเลย์หมุนด้วยความเร็ว N รอบต่อวินาที



รูปที่ 2.10 การเกิดแรงบิด

จะได้แรงบิด $T = F \times r$ (N- m)

งานจากแรงในการหมุน 1 รอบ = แรง × ระยะทาง

$= F \times 2\pi r$ (J)

กำลังที่เกิดขึ้น $= F \times 2\pi r \times N$ $(\frac{J}{S})$

$= (F \times r) \times 2\pi N$ $(\frac{J}{S})$

โดยที่ : $2\pi N$ = ความเร็วเชิงมุม ω มีหน่วยเป็น $(\frac{rad}{s})$

และ F× r = แรงบิด (N- m)

∴ กำลังที่เกิดขึ้น $= T \times \omega$ $(\frac{J}{S})$ หรือ (W) (2.8)

2.2.2.4.1 แรงบิดในอาร์เมเจอร์(Torque in Armature, T_a)

ถ้า T_a เป็นแรงบิดที่เกิดขึ้นในอาร์เมเจอร์มีหน่วยเป็น(N-m)ขณะที่มอเตอร์หมุนอยู่ที่ N (r.p.s)

กำลังที่เกิดขึ้นจะมีค่า $P_m = T_a \times 2\pi N$ (W) (2.9)

กำลังกลที่เกิดในอาร์เมเจอร์ $= E_b I_a$ (W) (2.10)

จากสมการที่ (2.9) และ (2.10) จะได้

$$T_a \times 2\pi N = E_b I_a \quad (2.11)$$

เมื่อ $$E_b = (\phi ZN)\left(\frac{P}{A}\right) \quad (V)$$

$$\therefore T_a \times 2\pi N = (\phi ZN)\left(\frac{P}{A}\right) \times I_a$$

หรือ $$T_a = \left(\frac{1}{2\pi}\right) Z\phi I_a \left(\frac{P}{A}\right)$$

ดังนั้น $$T_a = 0.159$$

$$\phi ZI_a\left(\frac{P}{A}\right) N - m = 0.01629\phi ZI_a\left(\frac{P}{A}\right) \quad \text{(kg-m)} \quad (2.12)$$

โดยที่ : 1กิโลกรัม-น้ำหนัก (kg-wt) = 9.81(N)

จากสมการที่(2.11) ข้างบนจะเห็นว่า

$$T_a = \left(\frac{1}{2\pi}\right)\frac{(E_b I_a)}{(N)} = 0.159\frac{E_b I_a}{(N)} \text{ (N-m)} \quad (2.13)$$

$$= 0.0162\frac{E_b I_a}{(N)} \quad \text{(kg-m)} \quad (2.14)$$

2.2.2.4.2 แรงบิดเพลาของมอเตอร์ (Shaft Torque, T_{sh})

แรงบิดที่อาร์เมเจอร์สามารถคำนวณหาได้โดยใช้สมการที่ (2.13) หรือสมการ (2.14) แต่ยังไม่ได้คิดการสูญเสียที่แกนเหล็กและแรงเสียดทานในมอเตอร์ แรงบิดที่นำมาใช้ประโยชน์คือแรงบิดที่เพลา (T_{sh}) กำลังที่เกิดขึ้นที่เพลาเรียกว่า กำลังม้า (brake horse-power :B.H.P.)

ขนาดกำลังม้า (B.H.P.) $$= \frac{(T_{sh} \times 2\pi N)}{746} \quad \text{(กำลังม้า)} \quad (2.15)$$

$$T_{sh} = \frac{746 \times B.H.P.}{(2\pi N)} \quad \text{(N-m)} \quad (2.16)$$

$T_a - T_{sh}$ = เป็นแรงบิดสูญเสีย (N-m)

แรงบิดที่สูญเสีย=0.159 × สูญเสียที่แกนเหล็กและแรงเสียดทาน(N-m) (2.17)

= 0.0162 × สูญเสียที่แกนเหล็กและแรงเสียดทาน (kg-m) (2.18)

2.2.3 ข้อดีและข้อเสียของมอเตอร์ไฟฟ้ากระแสตรง

ข้อได้เปรียบของมอเตอร์ไฟฟ้ากระแสตรงกับมอเตอร์ไฟฟ้ากระแสสลับ

1. มอเตอร์กระแสตรงมีคุณสมบัติเหมาะสมในการควบคุมความเร็วของการเคลื่อนที่โดยมีพิสัยของอัตราเร็วมากกว่ามอเตอร์กระแสสลับ ซึ่งทำให้เราสามารถเพิ่มเติมอัตราเร็วให้สูงหรือต่ำกว่าอัตราเร็วปกติ

2. มอเตอร์กระแสตรงมีแรงบิดขณะสตาร์ทสูงมากเหมาะสมกับงาน ในลักษณะที่ต้องยกฉุดลากและลากงานขับเคลื่อน

3. วิธีการควบคุมมอเตอร์กระแสตรงง่ายและนุ่มนวลกว่ามอเตอร์กระแสสลับ ที่ทำงานในลักษณะเดียวกันแต่ก็ยังพบว่ามอเตอร์กระแสตรงนั้น ก็ยังมีข้อด้อยกว่ามอเตอร์กระแสสลับอยู่หลายประการ ได้แก่

1. ต้องจัดหาแหล่งจ่ายไฟตรงไว้สำหรับใช้งานให้เป็นพิเศษ

2. สำหรับขนาดแรงม้าที่เท่ากัน มอเตอร์กระแสตรงจะมีขนาดใหญ่และมีราคาสูงกว่ามอเตอร์เหนี่ยวนำ

3. มีความยุ่งยากในการสตาร์ทมากกว่ามอเตอร์กระแสสลับ เว้นแต่ว่าเป็นมอเตอร์ขนาดเล็กที่ไม่ต้องการมีการสตาร์ทแบบพิเศษ

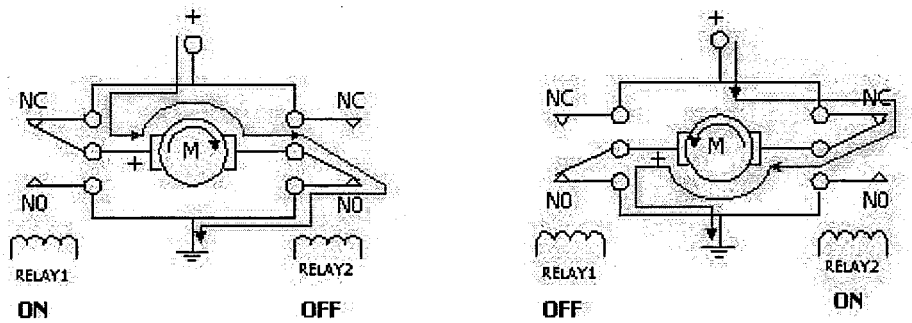
4. ต้องหมั่นซ่อมบำรุงเนื่องจากคอมมิวเตเตอร์ (Commutator) มีการสึกหรออันเนื่องมาจากประกายไฟ การอาร์คและการสึระหว่างแปรงถ่านกับคอมมิวเตเตอร์

2.2.4 การขับและกลับทิศทางของมอเตอร์กระแสตรง (DC MOTOR)

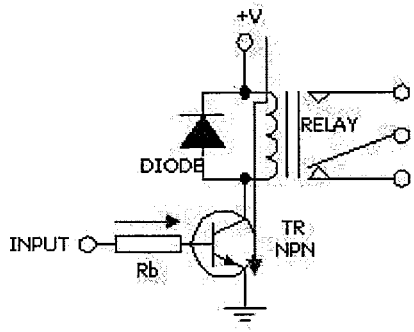
มอเตอร์ไฟฟ้ากระแสตรง (DC Motor) ในแง่ของทฤษฎีการทำงานจะกล่าวถึงพื้นฐานการควบคุมการทำงานของมอเตอร์ไฟฟ้ากระแสตรง ทั้งการควบคุมทิศทางการหมุนและการควบคุมความเร็ว ซึ่งในการควบคุมการหมุนมอเตอร์กระแสตรงเป็นเครื่องกลทางไฟฟ้าที่เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานกลผ่านทางแกนหมุนหรือเพลามอเตอร์ สามารถที่จะหมุนได้เนื่องจากมีแรงของสนามแม่เหล็ก 2 แหล่ง กระทำต่อกัน โดยที่สนามแม่เหล็กทั้ง 2 แหล่ง อาจจะเป็นแบบที่ได้จากการจ่ายกระแสไฟฟ้าผ่านขดลวดสเตเตอร์ (Stator Winding) และขดลวดอาร์เมเจอร์ (Armature) แต่มอเตอร์กระแสตรงที่นิยมใช้เป็นแบบที่มีแม่เหล็กถาวร (Permanent-Magnet) และเป็นส่วนในการสร้างสนามแม่เหล็กแทนขดลวดสเตเตอร์ (Stator Winding) และใช้การผ่านกระแสไฟฟ้าเข้าไปที่ขดลวดอาร์เมเจอร์เนื่องจากจะลดความสูญเสียได้เพราะไม่มี Field Winding นั่นคือประสิทธิภาพที่ดีขึ้นนอกจากนี้มอเตอร์ยังมีขนาดเล็ก และราคาถูกสนามแม่เหล็กที่เกิดจากแม่เหล็กถาวร (Permanent Magnet) และเกิดจากการจ่ายกระแสไฟฟ้ากระแสตรงเข้าไปในขดลวดอาร์เมเจอร์จะเป็นส่วนทำให้เกิดค่าแรงบิด (Torque) เกิดขึ้นที่โรเตอร์ซึ่งจะทำให้เกิดการหมุนได้นั่นเอง

ในการใช้ตัวไอซีไมโครคอนโทรเลอร์ ซึ่งจะเป็นตัวควบคุมความเร็วและทิศทางการหมุนของมอเตอร์กระแสตรงนั้น เราจะต้องมีส่วนของวงจรที่เรียกว่าวงจรขับมอเตอร์ (Drive) ในส่วนของวงจรกลับทิศทางของมอเตอร์นั้น สามารถที่จะใช้รีเลย์ต่อวงจรสวิตซ์เพื่อกลับทิศทางของขั้วไฟกระแสตรงหรืออาจใช้อุปกรณ์สารกึ่งตัวนำที่เป็นวงจรขับกำลังเช่น ทรานซิสเตอร์ มอสเฟต แล้วแต่วิธีที่เราจะเลือกใช้งาน

จากรูปเป็นการใช้รีเลย์ ควบคุมการเปลี่ยนทิศทางการหมุนของมอเตอร์ โดยการควบคุมการปิด - เปิดที่รีเลย์ 2 ตัว ซึ่งจะทำหน้าที่กลับทิศทางของขั้วไฟที่ป้อนให้กับมอเตอร์ โดยการสลับการทำงานของรีเลย์ เช่นให้รีเลย์ตัวที่ 1 ทำงาน (ON) และรีเลย์ตัวที่ 2 หยุดทำงาน (OFF) จะทำให้มอเตอร์หมุนไปทางซ้าย และในทำนองเดียวกันถ้าหากรีเลย์ตัวที่ 1 หยุดทำงาน (OFF) และรีเลย์ตัวที่ 2 ทำงาน (ON) ก็จะทำให้มอเตอร์หมุนไปทางขวา



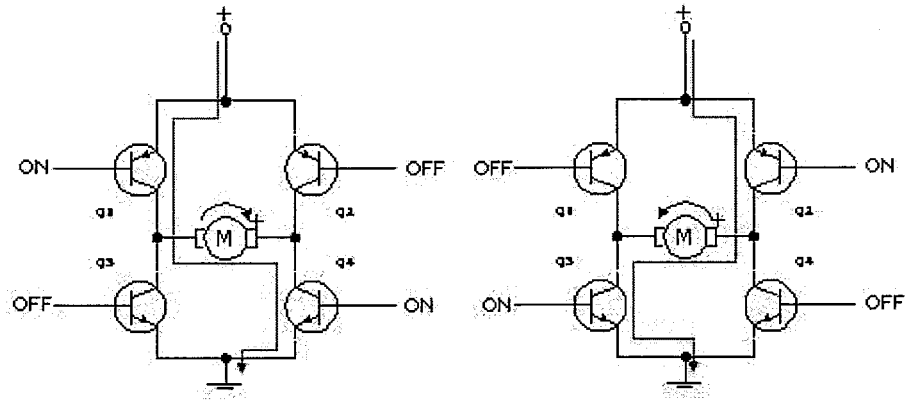
รูปที่ 2.11 แสดงการกลับทิศทางของมอเตอร์กระแสตรงโดยใช้รีเลย์



รูปที่ 2.12 แสดงการใช้ทรานซิสเตอร์เพื่อขับรีเลย์ให้ทำงาน

จากรูปเป็นวงจรขับรีเลย์โดยใช้ทรานซิสเตอร์ทำหน้าที่ขยายกระแส ด้วยเหตุผลเพราะไม่สามารถจะใช้ที่ขาเอาต์พุตของไมโครคอนโทรลเลอร์ป้อนกระแสไฟเข้าที่ขดลวดของรีเลย์โดยตรงได้ เนื่องจากว่ากระแสที่จ่ายออกมาจากขาเอาต์พุตของไมโครคอนโทรลเลอร์มีค่าน้อยเกินไป ดังนั้นเราจึงต้องมีส่วนของวงจรทรานซิสเตอร์ เพื่อที่จะทำการขยายกระแสให้เพียงพอในการป้อน

ให้กับขดลวดของรีเลย์ ส่วนไดโอดนำมาต่อไว้สำหรับป้องกันแรงดันย้อนกลับที่เกิดจากการเหนี่ยวนำของสนามแม่เหล็กในขณะเกิดการยุบตัว ซึ่งอาจจะทำให้ทรานซิสเตอร์เสียหายได้



รูปที่ 2.13 แสดงการใช้ทรานซิสเตอร์เป็นวงจรขับและกำหนดทิศทางของมอเตอร์กระแสตรง

จากรูปเป็นวงจรลิเนียร์บริดจ์แอมป์ ซึ่งจะประกอบไปด้วยทรานซิสเตอร์กำลัง 4 ตัว ที่ทำหน้าที่ขับและควบคุมทิศทางการหมุนของมอเตอร์ ถ้าหากกำหนดให้ทรานซิสเตอร์ Q1 และ Q4 อยู่ในสภาวะทำงาน (Active) กระแสไฟฟ้าจะไหลผ่านทรานซิสเตอร์จากซ้ายไปขวา โดยผ่านมอเตอร์กระแสตรงทำให้มอเตอร์หมุนไปทางขวา ในทำนองเดียวกันถ้าหากเราทำให้ทรานซิสเตอร์ Q2 และ Q3 อยู่ในสภาวะทำงาน (Active) กระแสไฟฟ้าก็จะไหลจากทางขวาไปทางซ้าย ซึ่งจะส่งผลให้มอเตอร์กลับทิศทางการหมุนจากทางขวาไปทางซ้าย

2.2.5 การควบคุมความเร็วของมอเตอร์กระแสตรง

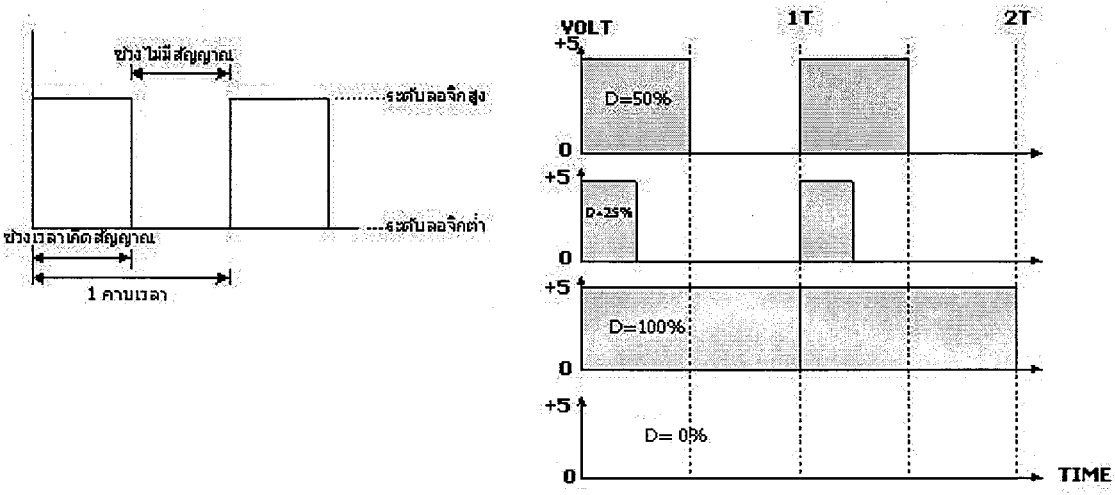
การควบคุมความเร็วของมอเตอร์กระแสตรงมีหลายวิธีด้วยกัน ซึ่งอาจจะใช้วิธีการควบคุมแบบพื้นฐานทั่วไป เช่น การควบคุมด้วยวิธีการใช้ตัวต้านทานปรับค่าโดยต่ออนุกรมกับมอเตอร์ หรือใช้วิธีการการควบคุมโดยการเปลี่ยนค่าของระดับแรงดันที่ป้อนให้กับมอเตอร์ แต่การควบคุมในวิธีดังกล่าว ถึงแม้ว่าจะควบคุมความเร็วมอเตอร์ให้คงที่ได้แต่ที่ความเร็วต่ำจะส่งผลให้แรงบิดต่ำไปด้วย ดังนั้นเราจึงเลือกใช้วิธีการควบคุมโดยการจ่ายกระแสไฟให้กับมอเตอร์เป็นช่วง ๆ โดยอาศัยกระแสไฟที่ป้อนให้กับมอเตอร์ให้เป็นค่าเฉลี่ยที่เกิดขึ้นในแต่ละช่วง ซึ่งเราเรียกว่าวิธีการของการมอดูเลชั่นทางความกว้างของพัลส์ PWM (Pulse Width Modulation)

2.2.6 วิธีการมอดูเลชั่นทางความกว้างของพัลส์ (PWM)

การมอดูเลชั่นทางความกว้างของพัลส์ PWM (Pulse Width Modulation) จะเป็นการปรับเปลี่ยนที่สัดส่วนและความกว้างของสัญญาณพัลส์ โดยซึ่งความถี่ของสัญญาณพัลส์จะไม่มีการเปลี่ยนแปลง หรือเป็นการเปลี่ยนแปลงที่ค่าของดิวตี้ไซเกิล (duty cycle) นั้นเองซึ่งค่าของดิวตี้

ไซเคิล คือช่วงความกว้างของพัลส์ที่มีสถานะลอจิกสูง โดยคิดสัดส่วนเป็นเปอร์เซนต์จากความกว้างของพัลส์ทั้งหมดยกตัวอย่าง เช่น ถ้าหากค่าดิวตี้ไซเคิลมีค่าเท่ากับเท่ากับ 50 % ก็หมายถึงใน 1 รูปสัญญาณพัลส์จะมีช่วงของสัญญาณที่เป็นสถานะลอจิกสูงอยู่ครึ่งหนึ่งและสถานะลอจิกต่ำอยู่อีกครึ่งหนึ่ง ดังรูป 2.14 และในทำนองเดียวกันถ้าหากค่าดิวตี้ไซเคิลมีค่ามาก หมายความว่าความกว้างของพัลส์ที่เป็นสถานะลอจิกสูงจะมีความกว้างมากขึ้น หากค่าดิวตี้ไซเคิลมีค่าเท่ากับ 100 % ก็หมายความว่าจะไม่มีสถานะลอจิกต่ำเลยซึ่งค่าดิวตี้ไซเคิลสามารถจะหาได้จากค่าความสัมพันธ์ดังนี้

ค่าดิวตี้ไซเคิล = (ช่วงของสัญญาณพัลส์/คาบเวลาทั้งหมดของสัญญาณ) X100%

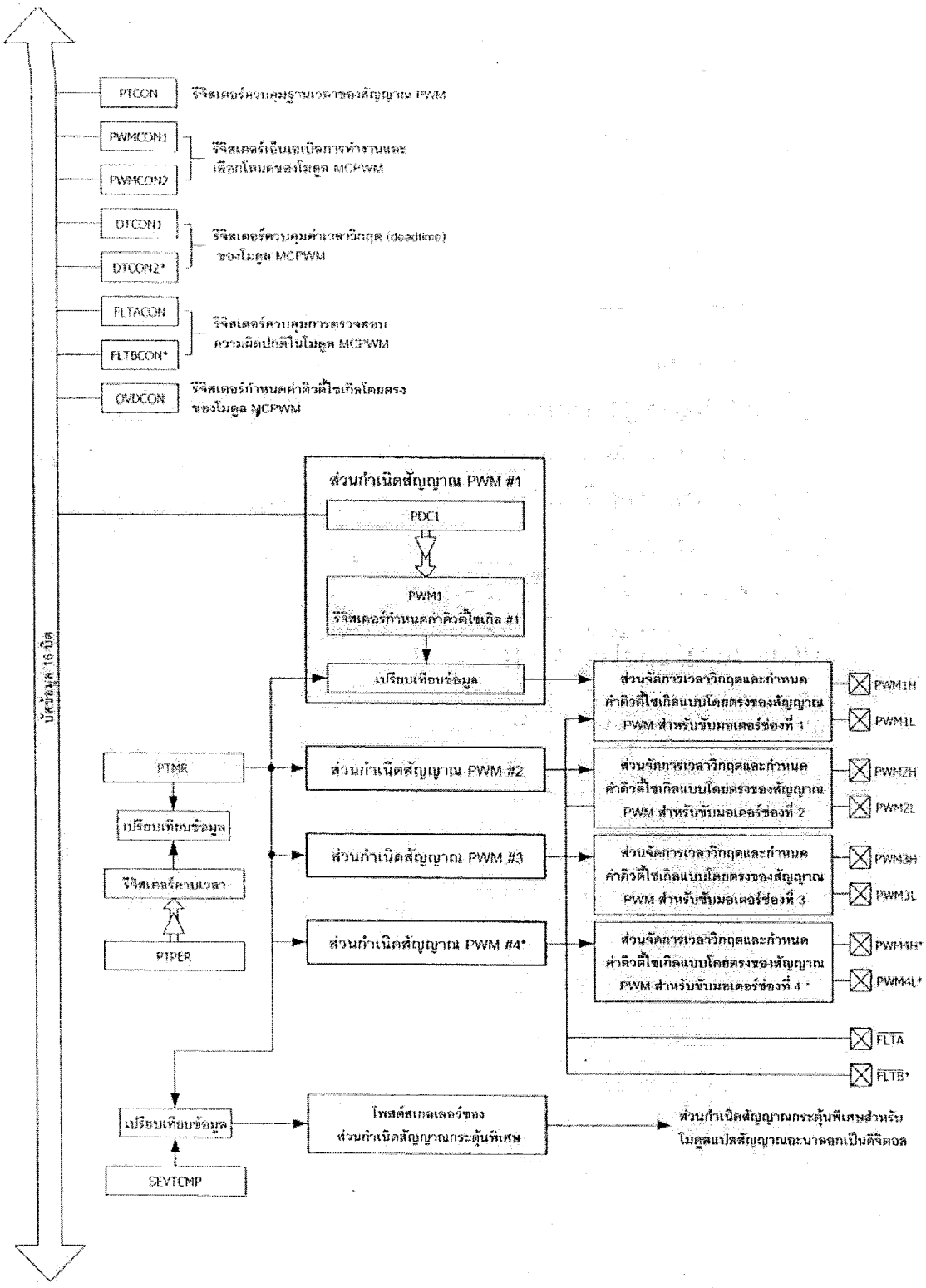


รูปที่ 2.14 แสดงความกว้างของพัลท์ขนาดต่างๆ และค่าดิวตี้ไซเคิล ของช่วงพัลท์ที่มีความถี่คงที่

บทที่ 3
การใช้งานโมดูล MCPWM เพื่อควบคุมมอเตอร์

3.1 คุณสมบัติโดยสรุปของโมดูล MCPWM

1. ความละเอียดของสัญญาณ PWM ที่สร้างขึ้นเท่ากับ $T_{cy}/2$
2. ในโมดูล MCPWM 1 ชุด มี 2 เอาต์พุต
3. สามารถใช้งานเอาต์พุตของโมดูล MCPWM แยกกันอย่างอิสระและร่วมกัน
4. เมื่อทำงานในแบบร่วมกัน หรือคอมพลีเมนทารีสามารถกำหนดค่าวิกฤต (dead time) เพื่อให้ช่วยการขับมอเตอร์ 3 เฟสให้เป็นไปอย่างมีประสิทธิภาพ
5. สามารถเลือกโหมดเอาต์พุตได้ 4 โหมด
    a. โหมดปรับขอบสัญญาณ (Edge aligned mode)
    b. โหมดสัญญาณเดี่ยว (Single event mode)
    c. โหมดปรับสัญญาณกึ่งกลาง (Center aligned mode)
    d. โหมดปรับสัญญาณกึ่งกลางพร้อมปรับปรุงค่า (Center aligned mode with double updates)
6. มีอินพุตสำหรับจับความผิดพลาดในการทำงาน (FAULT) แบบโปรแกรมได้
7. สามารถสร้างสัญญาณกระตุ้นส่งไปยังโมดูลแปลงสัญญาณอนาลอก เพื่อกำหนดจังหวะการทำงานให้สัมพันธ์กัน



รูปที่ 3.1 แสดงไดอะแกรมการทำงานของโมดูลMCPWMในไมโครคอนโทรลเลอร์ dsPIC

ในแผนภาพไดอะแกรมแสดงการทำงานของโมดูล MCPWM ส่วนประกอบหลักของโมดูลนี้คือส่วนกำเนิดสัญญาณ PWM ที่มีอยู่ด้วยกันสูงสุด 4 ชุด ซึ่งจะได้ค่าฐานเวลามาจากรีจิสเตอร์ PTMR และ PTPER ส่วนค่าดิวตี้ไซเกิลของสัญญาณ PWM นั้นสามารถกำหนดได้จากรีจิสเตอร์ที่แยกกันเป็นอิสระในแต่ละชุดของสัญญาณ นอกจากนั้นยังสามารถกำหนดการทำงานของขาพอร์ตเอาต์พุตของโมดูล MCPWM โดยผ่านทางรีจิสเตอร์ OVDCON

ส่วนกำเนิดสัญญาณ PWM สำหรับควบคุมมอเตอร์แต่ละชุดในโมดูล MCPWM สามารถกำหนดให้ทำงานแยกจากกันเป็นอิสระ (independent mode) หรือทำงานร่วมกัน (Complementary mode) เพื่อขับมอเตอร์ 3 เฟสได้ โดยใช้การกำหนดผ่านรีจิสเตอร์ PWMCON 1 และ PWMCON 2 เมื่อกำหนดให้ทำงานร่วมกันต้องมีการจัดการสัญญาณในแต่ละเฟส นั่นคือการจัดการค่าเวลาการหน่วงเฟส (detetime control) โดยใช้รีจิสเตอร์ DTCON 1 และ DTCON 2

สัญญาณที่ออกจากโมดูล MCPWM จะมีขาพอร์ต 2 ขาต่อช่องให้ใช้งาน นั่นคือขาเอาต์พุตด้านแรงดันสูง PWMxH และขาเอาต์พุตด้านแรงดันต่ำ PWMxL (x คือหมายเลขของช่องเอาต์พุตมี 4 ค่าคือ 1-4) หรือเรียกว่าคู่เอาต์พุต นอกจากนั้นสัญญาณเอาต์พุตสามารถส่งผ่านโพสต์สเกลเลอร์เพื่อสร้างสัญญาณกระตุ้นให้แก่โมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล เพื่อให้โมดูล MCPWM สามารถทำงานสัมพันธ์กับโมดูล ADC ได้ด้วย

นอกจากนั้นในโมดูล MCPWM ยังมีอินพุตใช้สำหรับรับสัญญาณตรวจสอบความผิดปกติ FAULT เพื่อป้องกันไม่ให้โมดูล MCPWM ทำงานผิดพลาดหรือเสียหายเมื่อเกิดความผิดปกติขึ้น โดยในส่วนนี้มีพอร์ตอินพุตสำหรับรับสัญญาณ 2 ขา คือ FLTA และ FLTB

3.2 รีจิสเตอร์ที่ใช้งานในโมดูล MCPWM

ในโมดูล MCPWM ของไมโครคอนโทรลเลอร์ dsPIC มีรีจิสเตอร์ที่ใช้ควบคุมและกำหนดค่า อันประกอบด้วย

1. PTCON รีจิสเตอร์ควบคุมฐานเวลาในการกำเนิดสัญญาณ PWM
2. PTMR รีจิสเตอร์กำหนดค่าฐานเวลาของการกำเนิดสัญญาณ PWM
3. PTPER รีจิสเตอร์กำหนดคาบเวลาของฐานเวลาสำหรับการกำเนิดสัญญาณ PWM
4. SEVTCMP รีจิสเตอร์เปรียบเทียบค่า
5. PWMCON1 รีจิสเตอร์ควบคุม PWM#1
6. PWMCON2 รีจิสเตอร์ควบคุม PWM#2
7. DTCON1 รีจิสเตอร์ควบคุมค่าเวลาวิกฤตหรือ Dead Time#1
8. DTCON2 รีจิสเตอร์ควบคุมค่าเวลาวิกฤตหรือ Dead Time#2
9. FLTACON รีจิสเตอร์ควบคุมการตรวจจับความผิดปกติของการขับมอเตอร์ชุด A

10. FLTBCON รีจิสเตอร์ควบคุมการตรวจจับความผิดปกติของการขับมอเตอร์ ชุด B
11. PDC1 รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำหนดสัญญาณ PWM ชุดที่ 1
12. PDC2 รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำหนดสัญญาณ PWM ชุดที่ 2
13. PDC3 รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำหนดสัญญาณ PWM ชุดที่ 3
14. PDC4 รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำหนดสัญญาณ PWM ชุดที่ 4

3.3 ฐานเวลาสัญญาณ PWM

ในรูปแสดงไดอะแกรมการกำหนดฐานเวลาของสัญญาณ PWM ที่ใช้ในโมดูล MCPWM ซึ่งแยกอิสระจากโมดูลเอาต์พุตเปรียบเทียบ (Output Compare : OC) ค่าฐานเวลาจะได้มาจากการทำงานของไทเมอร์ 15 บิตต่อร่วมกับพรีสเกลเลอร์และโพสเกลเลอร์ในโมดูลของ MCPWM โดยข้อมูล 15 บิต นั้นบรรจุอยู่ใน 15 บิตล่างของรีจิสเตอร์ PTMR ส่วนบิต MSB คือบิต PTDIR เป็นบิตที่อ่านได้อย่างเดียว ใช้ในการแสดงทิศทางในการนับค่าในปัจจุบันของฐานเวลา PWM นี้ โดยถ้าบิตนี้เป็น “0” แสดงว่า PTMR กำลังนับค่าขึ้นและเป็น “1” เมื่อกำลังนับค่าลง

การเอ็นเอเบิลให้ส่วนฐานเวลาทำงานจะต้องเซ็ตบิต PTEN ซึ่งเป็นบิต 15 ของรีจิสเตอร์ PTCON อย่างไรก็ตามค่าในรีจิสเตอร์ PTMR จะไม่ถูกเคลียร์แม้ว่าส่วนฐานเวลาของ PWM นี้จะถูกดิสเอเบิลด้วยการเคลียร์บิต PTEN

ฐานเวลา PWM ในโมดูล MCPWM สามารถกำหนดให้ทำงานได้ 4 โหมด คือ

1. โหมดเปลี่ยนแปลงค่าอิสระ (Free Running mode)
2. โหมดทำงานครั้งเดียว (Single Event mode)
3. โหมดนับค่าขึ้นหรือลงอย่างต่อเนื่อง (Continuous Up/Down mode)
4. โหมดการนับขึ้นลงอย่างต่อเนื่องพร้อมการอินเตอร์รัปเพื่อปรับปรุงค่า (Continuous Up /Down Count mode with interrupts for double-updates) การเลือกโหมดทำได้ด้วยการกำหนดค่าที่บิต PTMOD1 และ PTMOD0 ซึ่งเป็นบิต 1 และ 0 ในรีจิสเตอร์PTCON

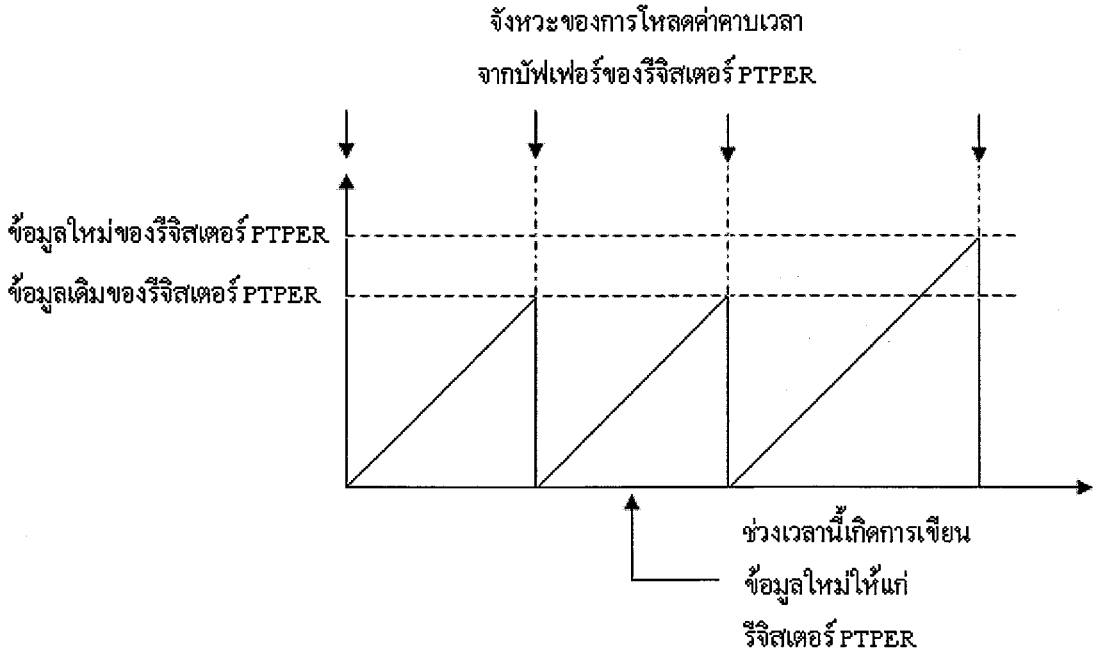
3.3.1 คาบเวลาของสัญญาณ PWM

รีจิสเตอร์ PTPER ถูกใช้สำหรับกำหนดค่าการนับคาบเวลาของรีจิสเตอร์ PTMR ผู้พัฒนาต้องระบุข้อมูลขนาด 15 บิต ลงในบิต 0 ถึง 14 ของรีจิสเตอร์ PTPER เมื่อโมดูลนี้ทำงานจนกระทั่งค่าของรีจิสเตอร์ PTMR เท่ากับ PTPER ค่าฐานเวลาจะรีเซตเป็น “0” หรือเปลี่ยนทิศทางการนับค่าในสัญญาณนาฬิกาลูกถัดไป ขึ้นอยู่กับการกำหนดโหมดทำงาน

คาบเวลาของฐานเวลาจะมีขนาดของบัฟเฟอร์เป็น 2 เท่า เพื่อรองรับการเปลี่ยนแปลงค่าในระหว่างการทำงานได้โดยปราศจากการรบกวน นั่นคือในรีจิสเตอร์ PTPER จะมีรีจิสเตอร์บัฟเฟอร์

สำรองรับค่าที่ต้องการเปลี่ยนแปลงใหม่ในระหว่างที่กำลังทำงานกับค่าเดิม โดยรีจิสเตอร์บัฟเฟอร์นี้ผู้ใช้งานไม่สามารถเข้าถึงได้ ข้อมูลสำหรับกำหนดค่าคาบเวลาจะถูกเขียนลงในรีจิสเตอร์ PTPER แตกต่างกันไปตามโหมดการทำงานของฐานเวลา PWM

ในโหมดเปลี่ยนแปลงค่าอิสระและโหมดทำงานครั้งเดียว ข้อมูลที่ถูกเก็บอยู่ในรีจิสเตอร์ PTPER จะถูกโหลดลงในรีจิสเตอร์คาบเวลาเมื่อรีจิสเตอร์ PTMR ถูกรีเซตเป็น "0" หลังจากที่ค่าของรีจิสเตอร์ PTMR ตรงกับค่าของ PTPER ดังแสดงด้วย



รูปที่ 3.2 ไดอะแกรมแสดงการโหลดค่าของรีจีสเตอร์ PTPER

ข้อมูลสำหรับกำหนดคาบเวลาของสัญญาณ PWM ที่เขียนไปยังรีจิสเตอร์ PTPER สามารถคำนวณได้จากสมการต่อไปนี้

$$PTPER = \frac{F_{CY}}{F_{PWM} \times PTMR\_\text{Pr}escaler} - 1$$

F_{CY} = ความถี่ของการทำงาน

F_{PWM} = ความถี่ของสัญญาณ

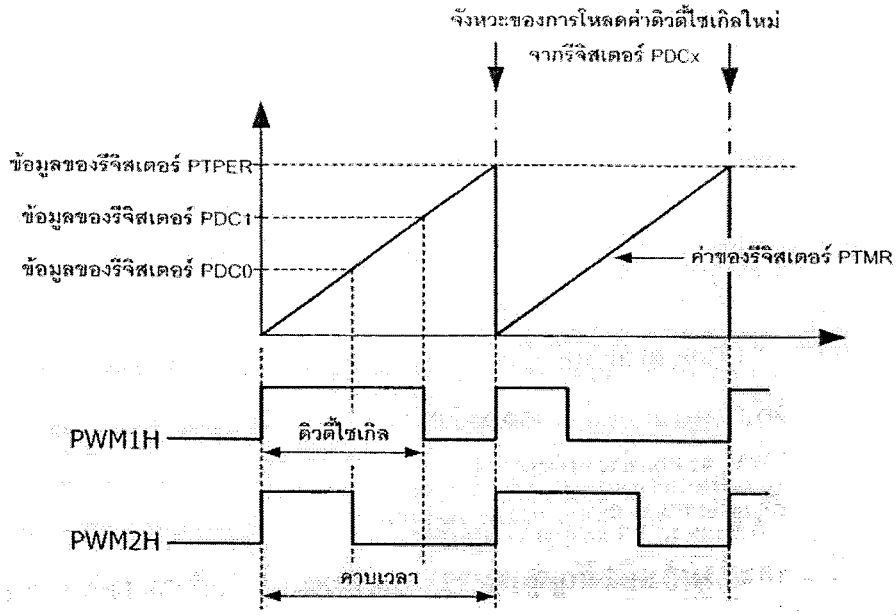
3.3.2 โหมดการทำงานของส่วนกำเนิดสัญญาณ PWM ในโมดูล MCPWM

มีด้วยกัน 4 แบบคือ

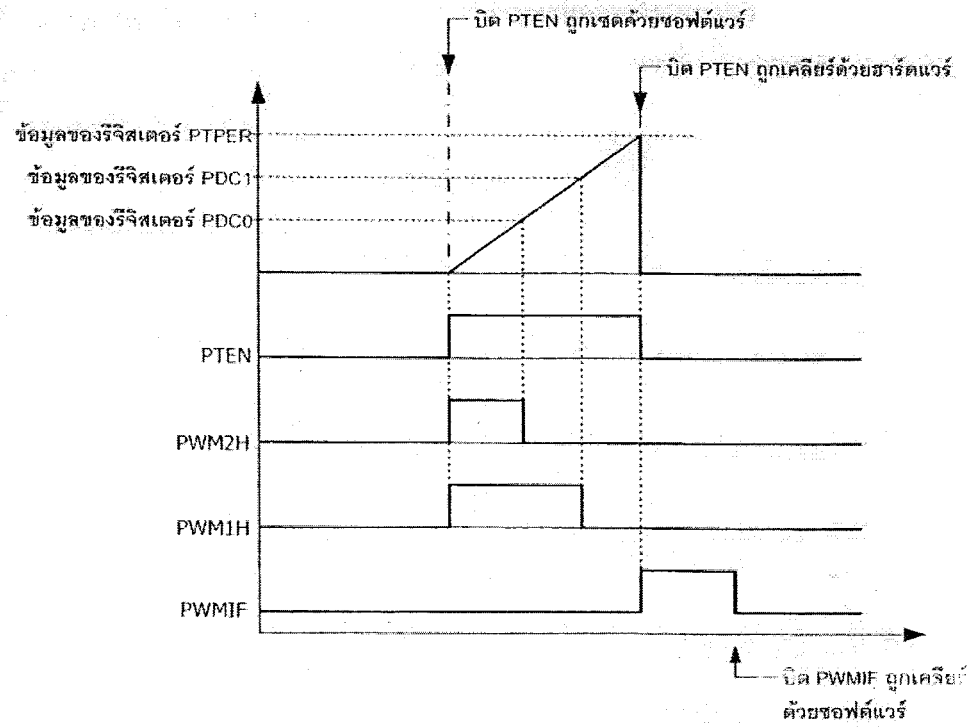
1. โหมดปรับขอบสัญญาณ (Edge aligned mode)
2. โหมดสัญญาณเดี่ยว (Single event mode)
3. โหมดปรับสัญญาณกึ่งกลาง (Center aligned mode)
4. โหมดปรับสัญญาณกึ่งกลางพร้อมปรับปรุงค่า (Center aligned mode with double updates) ซึ่งสัมพันธ์กับโหมดการทำงานของฐานเวลา PWM

เมื่อฐานเวลา PWM ทำงานในโหมดเปลี่ยนแปลงค่าอิสระ ส่วนกำเนิดสัญญาณ PWM จะทำงานในโหมดปรับขอบสัญญาณ เมื่อฐานเวลา PWM ทำงานในโหมดทำงานครั้งเดียว ส่วนกำเนิดสัญญาณ PWM จะทำงานในโหมดสัญญาณเดี่ยว เมื่อฐานเวลา PWM ทำงานในโหมดนับค่าขึ้นลงอย่างต่อเนื่อง ส่วนกำเนิดสัญญาณ PWM จะทำงานในโหมดปรับสัญญาณกึ่งกลาง เมื่อฐานเวลา PWM ทำงานในโหมดนับค่าขึ้นลงอย่างต่อเนื่องพร้อมปรับปรุงค่า ส่วนกำเนิดสัญญาณ PWM จะทำงานในโหมดปรับสัญญาณกึ่งกลางพร้อมปรับปรุงค่า

3.3.3 การทำงานของส่วนกำเนิดสัญญาณ PWM ในโหมดปรับขอบสัญญาณ



รูปที่ 3.3 ไดอะแกรมเวลาแสดงการเกิดสัญญาณ PWM เมื่อทำงานในโหมดปรับขอบสัญญาณ



รูปที่ 3.4 ไดอะแกรมเวลาแสดงการเกิดสัญญาณ PWM เมื่อทำงานในโหมดสัญญาณเดี่ยว

ไดอะแกรมเวลาแสดงการเกิดสัญญาณ PWM เมื่อมีการทำงานในโหมดปรับขอบสัญญาณ เริ่มต้นด้วยการกำหนดให้ฐานเวลา PWM ทำงานในโหมดเปลี่ยนแปลงค่าอิสระ ดังนั้นคาบเวลาของสัญญาณ PWM จะถูกกำหนดโดยค่าที่โหลดให้แก่รีจิสเตอร์ PTPER ส่วนดิวตี้ไซเกิลถูกกำหนดโดยค่าในรีจิสเตอร์ PDCx

ในกรณีที่ค่าดิวตี้ไซเกิลไม่เป็นศูนย์ ส่วนวงจรเอาต์พุตของส่วนกำเนิดสัญญาณ PWM ทุกชุดที่ได้รับการเอ็นเอเบิล จะซึ่งทำงานที่จุดเริ่มต้นของคาบเวลาของสัญญาณ PWM หรือเมื่อค่าในรีจิสเตอร์ PTMR เท่ากับศูนย์ และหยุดทำงานเมื่อค่าของรีจิสเตอร์ PTMR ตรงกับค่าดิวตี้ไซเกิลของส่วนกำเนิดสัญญาณ PWM ถ้าหากค่าในรีจิสเตอร์ PDCx เป็นศูนย์ วงจรเอาต์พุตของส่วนกำเนิดสัญญาณ PWM จะไม่ทำงานใด ๆ นั่นหมายความว่าในโหมดนี้จะสามารถกำเนิดสัญญาณ PWM ได้ก็ต่อเมื่อค่ารีจิสเตอร์ PTPER ต้องมากกว่าค่าที่กำหนดในรีจิสเตอร์ PDCx

3.3.4 การเปลี่ยนแปลงค่าดิวตี้ไซเกิลสัญญาณ PWM ของโมดูล MCPWM

รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลทั้ง 4 ตัว PDC1 ถึง PDC4 ต่างก็มีรีจิสเตอร์บัฟเฟอร์เพื่อป้องกันสัญญาณรบกวน เมื่อมีการปรับปรุงค่าของสัญญาณ PWM โดยดิวตี้ไซเกิลของสัญญาณ PWM จะถูกปรับปรุงค่าตามข้อมูลที่เขียนลงในรีจิสเตอร์ PDCx จากนั้นจากค่ารีจิสเตอร์ PDCx จะถูกโหลดไปยังบัฟเฟอร์เพื่อทำการเปรียบเทียบ เมื่อมีการเปลี่ยนแปลงข้อมูลในรีจิสเตอร์ PDCx เรียบร้อย ข้อมูลนั้นจะถูกส่งไปยังบัฟเฟอร์เพื่อทำงานต่อไป ทำให้ไม่เกิดการติดขัดหรือเกิดความผิดพลาดในขณะเปลี่ยนค่าดิวตี้ไซเกิล

เมื่อฐานเวลา PWM ทำงานในโหมดนับค่าขึ้นลงอย่างต่อเนื่อง (ค่าของ PTMOD1 และ PTMOD0 เท่ากับ “0x”) ค่าดิวตี้ไซเกิลของสัญญาณ PWM จะถูกปรับปรุงเมื่อค่าของรีจิสเตอร์ PTMR เท่ากับ PTPER และเมื่อรีจิสเตอร์ PTMR เกิดการรีเซตเป็นศูนย์

3.3.5 การทำงานร่วมกันของส่วนกำเนิดสัญญาณ PWM หรือทำงานแบบคอมพลีเมนตารี (Complementary PWM Output Mode)

การทำงานในลักษณะนี้ของโมดูล MCPWM เหมาะสำหรับการนำไปสร้างสัญญาณเพื่อขับโหลดแบบอินเวอร์เตอร์ ตัวอย่างของโหลดแบบอินเวอร์เตอร์ ได้แก่ มอเตอร์อินดักชั่นไฟฟ้ากระแสสลับ 3 เฟส (ACIM:AC Induction Motor) และมอเตอร์แบบไม่มีแปรงถ่าน (BLDC:Brushless DC motor) ในการทำงานแบบคอมพลีเมนตารีนี้ วงจรเอาต์พุตของโมดูล PWM ในคู่ที่จะนำมาใช้งานนั้นจะไม่สามารถกำหนดให้ทำงานได้พร้อมกันนั่นคือ PWMxH และ PWMxL ต้องมีสภาวะที่ตรงข้ามกันทำให้มอสเฟตที่ต่อกับขาเอาต์พุตนั้นสลับกันทำงาน โดยมีส่วนประกอบ

ที่สำคัญที่เพิ่มขึ้นมา 2 ส่วน คือ ส่วนกำเนิดช่วงเวลาวิกฤต (Dead Time Generation) และส่วนการตรวจสอบความผิดปกติและกำหนดการทำงานของขาเอาต์พุตโดยตรง (Override and Fault Logic)

3.3.6 การควบคุมเวลาวิกฤต (Dead Time Control)

โดยธรรมชาติในส่วนของอุปกรณ์เหนี่ยวนำ เมื่อได้รับแรงดันกระตุ้นให้ทำงานแล้วสั่งให้หยุดทำงานตัวอุปกรณ์จะไม่สามารถหยุดอุปกรณ์ได้ในทันที และหากทำการกระตุ้นสั่งให้วงจรชุดขับต่อไปทำงานทันที จะทำให้เกิดการชนกันของแรงดันซึ่งอาจนำมาซึ่งความเสียหายของวงจรได้ นั่นคือที่มาของคำว่า dead time หรือเวลาวิกฤต ดังนั้นเพื่อแก้ไขปัญหานี้จะต้องมีการหน่วงเวลาให้อุปกรณ์ที่ถูกกำหนดหยุดทำงานลงอีกเล็กน้อย เพื่อให้เกิดความแน่ใจว่าได้มีการหยุดทำงานลงอย่างแท้จริง ก่อนที่จะกระตุ้นให้เกิดสภาวะการทำงานในเฟสถัดไป

ส่วนที่ทำหน้าที่นี้คือ ส่วนกำเนิดช่วงเวลาวิกฤต (Dead Time Generator) โดยส่วนกำเนิดช่วงเวลาวิกฤตจะได้รับการเอ็นเอเบิลโดยอัตโนมัติเมื่อกำหนดให้โมดูล MCPWM ทำงานในแบบคอมพลีเมนตารีใน dsPIC ที่มีโมดูล MCPWM แบบ 6 เอาต์พุต จะสามารถกำหนดค่าเวลาวิกฤตได้ 1 ค่า ส่วนแบบ 8 เอาต์พุต สามารถกำหนดได้ 2 ค่า ในกรณีที่สามารถกำหนดได้ 2 ค่า สามารถเลือกใช้วิธีการกำหนดค่าได้จาก 2 วิธี ดังต่อไปนี้

1. สามารถเลือกกำหนดค่าเวลาวิกฤตในช่วงหยุดทำงานของเอาต์พุตด้านแรงดันสูงหรือต่ำก็ได้ โดยเวลาวิกฤตค่าแรกจะกำหนดลงระหว่างช่วงหยุดทำงานของเอาต์พุตด้านแรงดันต่ำ
2. ค่าเวลาวิกฤตทั้งสองค่าสามารถกำหนดลงในแต่ละคู่ของเอาต์พุตได้อย่างอิสระ

3.4 การกำหนดการทำงานขาพอร์ตเอาต์พุตของโมดูล MCPWM โดยตรง

นอกจากการกำหนดให้ขาพอร์ตของโมดูล MCPWM ทำงานกับส่วนกำเนิดสัญญาณ PWM แล้วยังสามารถทำการกำหนดได้โดยตรงหรือ PWM Output Overeride โดยกระทำผ่านรีจิสเตอร์ OVDCON โดยการกำหนดค่าที่รีจิสเตอร์นี้ จะเป็นการกำหนดในลักษณะเข้าถึงโดยตรงที่ขาพอร์ตเอาต์พุตของโมดูล MCPWM

ข้อมูล 8 บิต บนที่อยู่ในรีจิสเตอร์ OVDCON จะใช้กำหนดการทำงานของขาพอร์ตเอาต์พุตว่าต้องการให้ควบคุมจากส่วนสัญญาณ PWM ตามวิธีการปกติ (เซตเป็น “1”) หรือจากค่าของบิต POUTxx (เคลียร์บิตเป็น “0”) ซึ่งเป็นข้อมูลใน 8 บิต ล่างของรีจิสเตอร์ OVDCON นี้ โดยแบ่งเป็นคู่ ๆ คือ บิต 15 และ 14 ใช้กำหนดการทำงานของขาเอาต์พุตโมดูล PWM 4 ไล่ไปตามลำดับ ดังนั้นบิต 9 และ 8 จึงใช้กำหนดการทำงานของขาเอาต์พุตในโมดูล PWM1 ส่วน 8 บิต ล่างของรีจิสเตอร์ OVDCON คือบิต POUTxx ใช้กำหนดการทำงานของขาเอาต์พุตโมดูล PWM โดยตรง ซึ่งจะมีการใช้งานก็ต่อเมื่อบิต POVDxx ใน 8 บิต บนเป็น “0”

อย่างไรก็ตามเมื่อกำหนดให้โมดูล MCPWM ทำงานในคอมพลีเมนตารีหรือแบบทำงานร่วมกัน จะไม่สามารถกำหนดการทำงานของขาพอร์ตโดยตรงได้

ในการทำงานให้ควบคุมขาพอร์ตเอาต์พุตของโมดูล MCPWM โดยตรงด้วยการกำหนดค่าในรีจิสเตอร์ OVDCON ยังมีอีก 1 บิตควบคุมในรีจิสเตอร์ PWMCON 2 ที่ต้องให้ความสนใจนั่นคือ บิต OSYNC (บิต 1 ของรีจิสเตอร์ PWMCON2) โดยบิตนี้ใช้ในการกำหนดจังหวะการเปลี่ยนแปลงค่าของรีจิสเตอร์ OVDCON เพื่อจะได้สัมพันธ์กับการทำงานของฐานเวลา PWM หรือเกิดการซิงโครไนซ์จะต้องเซตบิตนี้เป็น "1" โดยการซิงโครไนซ์จะเกิดขึ้นเมื่อ

(1) ค่าของรีจิสเตอร์ PTMR เป็นศูนย์ในโหมดปรับขอบสัญญาณ

(2) ค่าของรีจิสเตอร์ PTMR เป็นศูนย์หรือเมื่อค่าของ PTMR เท่ากับค่าของ PTPER ในโหมดปรับสัญญาณกึ่งกลาง

3.5 การใช้งานโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอลภายใน dsPIC

คุณสมบัติโดยสรุปของโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล

1. เป็นโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอลที่มีความละเอียด 10 บิต จำนวน 6 ช่อง

2. ใช้วิธีการแปลงสัญญาณแบบประมาณค่าหรือซักเซฟซีฟ แอ็ปพร็อกซิเมชั่น

3. มีอัตราเร็วในการสุ่มสัญญาณสูงสุด 500 kilo sample ต่อวินาที (ksps) หรือ 500000 จุดตัวอย่างต่อวินาที

4. สามารถกำหนดให้ทำงานได้ขณะเข้าสู่โหมดสลีป (Sleep mode)

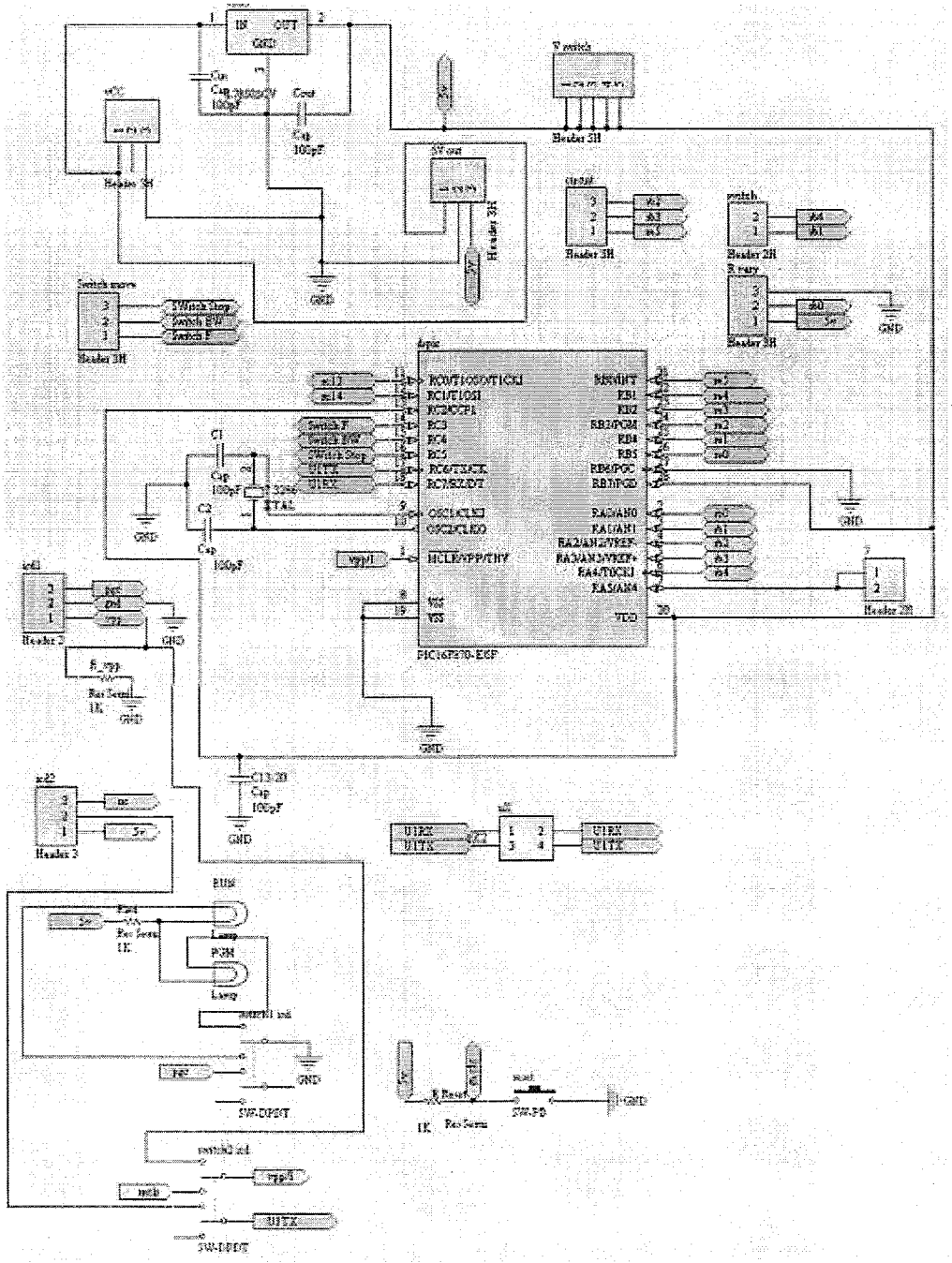
5. สามารถกำหนดระดับแรงดันอ้างอิงได้ทั้งจากภายในผ่านทางขา AV_{DD} กับ AV_{SS} และภายนอกผ่านทางขา V_{REF+} และ V_{REF-}

บทที่ 4

การออกแบบและการสร้างวงจร

4.1 ออกแบบทางฮาร์ดแวร์

4.1.1 dsPIC Programmer Board



รูปที่ 4.1 dsPIC Program Board

ในการเขียนโปรแกรมลงบนตัวไมโครคอนโทรลเลอร์ dsPIC จำเป็นจะต้องมีบอร์ดที่ทำหน้าที่ในการโปรแกรมข้อมูลสู่ dsPIC หรือที่เรียกว่า Programmer Board โดย dsPIC30F2010 Control Board ได้ออกแบบให้ขึ้นเพื่อความสะดวกในการนำไมโครคอนโทรลเลอร์ dsPIC30F2010 ไปใช้งานโดยสามารถเชื่อมต่อระหว่างพอร์ตของ dsPIC30F2010 เพื่อนำไปเชื่อมต่อกับอุปกรณ์ภายนอกเพื่อทำการควบคุม

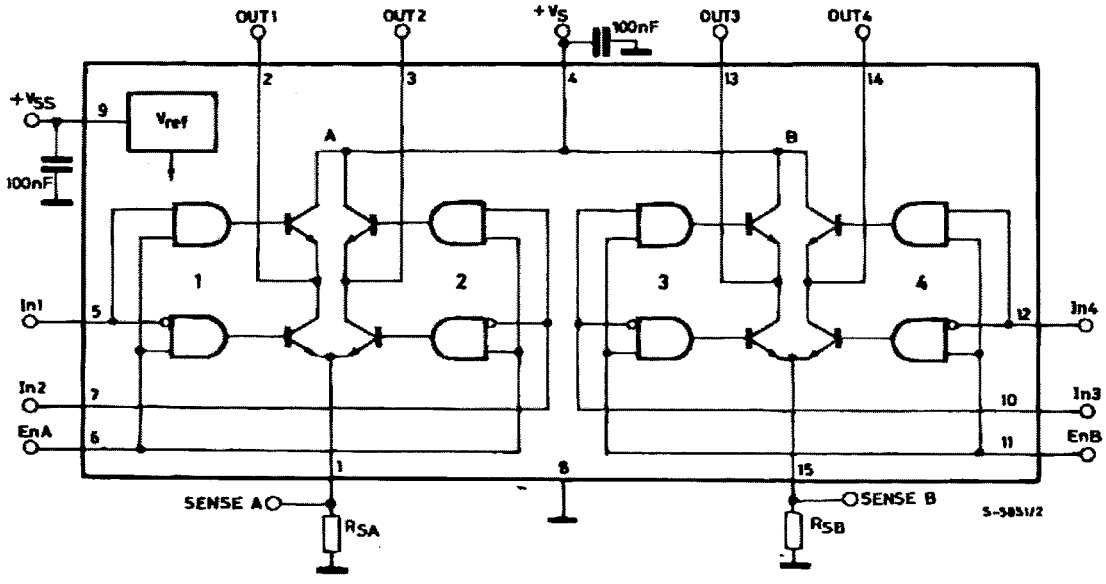
4.1.2 DC Motor Board

เป็นชุดสร้างมาเพื่อควบคุม DC Motor ที่ควบคุมการทำงานเนื่องจากไมโครคอนโทรลเลอร์ dsPIC30F2010 โดยใช้โมดูล Pulse Width Modulator (PWM) ในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

โครงสร้างและการทำงาน Dual Full-bridge Driver L298

เป็นไอซีควบคุมการทำงานของมอเตอร์ในลักษณะของวงจรฟูลบริดจ์ ซึ่งภายในประกอบด้วยวงจรฟูลบริดจ์ 2 วงจร โดยแต่ละวงจรสามารถให้แรงดันเอาต์พุตสูงสุดที่สามารถให้แรงดันเอาต์พุตสูงสุดที่ 46 v และจ่ายกระแสสูงสุดที่ 2 A สำหรับการต่อใช้งานคู่กันทั้ง 2 วงจร ในแบบขนานจะสามารถจ่ายกระแสได้สูงสุดที่ 4 A

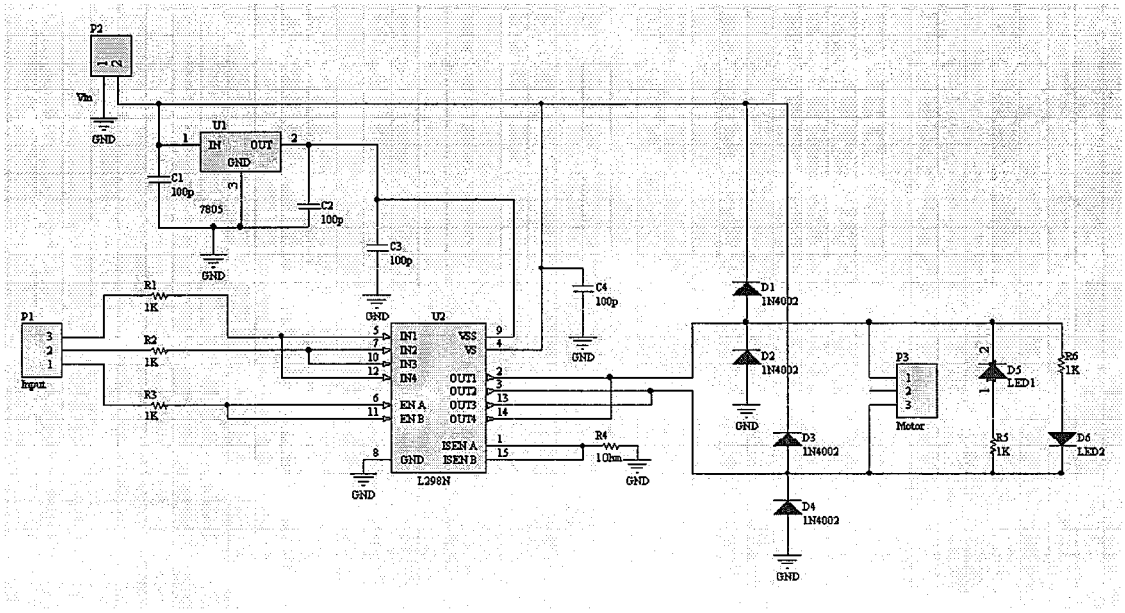
สำหรับการควบคุมทิศทางของมอเตอร์ จะป้อนสัญญาณเข้าที่ In1, In2, In3 และ In4 ในการควบคุมความเร็วจะป้อนสัญญาณ PWM เข้าที่ขา EnA และ EnB ดังรูป



รูปที่ 4.2 Block Diagram ของ L298

ตารางที่ 4.1 การควบคุมทิศทางของ L298

Input			Function
EN = H	IN1 = L	IN2 = L	Fast Motor Stop
	IN1 = L	IN2 = H	Reverse
	IN1 = H	IN2 = L	Forward
	IN1 = H	IN2 = H	Fast Motor Stop
EN = L	IN1 = X	IN2 = X	Free Running Motor Stop

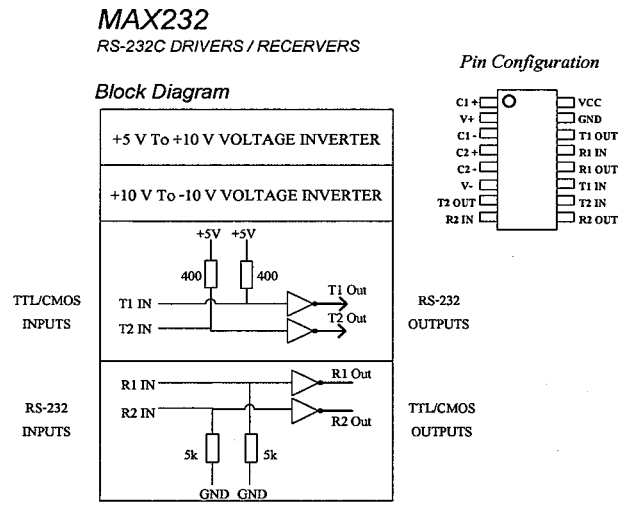


รูปที่ 4.4 แผนภาพวงจรขับเคลื่อน

4.1.3 RS232 INTERFACE

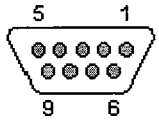
ในการเชื่อมต่อแบบอนุกรมเข้ากับอุปกรณ์คอมพิวเตอร์ต่าง ๆ เช่น คอมพิวเตอร์ เทเลกซ์ หรือ โทรพิมพ์ เป็นต้น มักจะกำหนดใช้การเชื่อมต่อตามมาตรฐาน RS-232C ทั้งนี้เพื่อให้มีการใช้งานเส้นสัญญาณหรือรูปแบบของตัวเชื่อมต่อที่สอดคล้องกัน ซึ่งจะได้ใช้ลดปัญหาของการเข้ากันไม่ได้ระหว่างสัญญาณของอุปกรณ์ที่มาเชื่อมต่อกันทั้งสองด้านให้น้อยลง เนื่องจากระดับโวลเตจที่ใช้และการแทนความหมายของระดับลอจิกตามมาตรฐานนี้แตกต่างไป จากที่ใช้งานกันในระบบ

ของดิจิตอลทั่วไปโดยระดับสัญญาณของ RS-232C เป็นแบบไบโพลาร์ ระดับโวลเตจทางด้านลบช่วง -3V ถึง -20V แทนค่าลอจิก 1 และโวลเตจทางด้านบวกช่วง +3V ถึง +20V แทนค่าลอจิก 0 ซึ่งจะทำให้เห็นได้ว่ามีความจำเป็นต้องเพิ่มเติมอุปกรณ์หรือวงจรพิเศษเข้าไป เพื่อที่จะเปลี่ยนระดับโวลเตจจากระบบ 0V ถึง +5V จากขาสัญญาณของ PIC เป็นระดับโวลเตจที่สูงกว่าค่า +3.0V หรือต่ำกว่า -3.0V

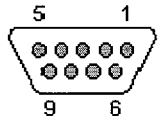


รูปที่ 4.5 ไอซีเบอร์MAX232ซึ่งเป็นวงจรเชื่อมต่อแบบRS-232Cโดยการใช้ไฟเลี้ยง +5 เพียงชุดเดียววงจรการเชื่อมต่อระหว่าง PIC กับ PC ทาง Serial Port

การต่อสัญญาณ Serial Port



(To Computer 1).



(To Computer 2).

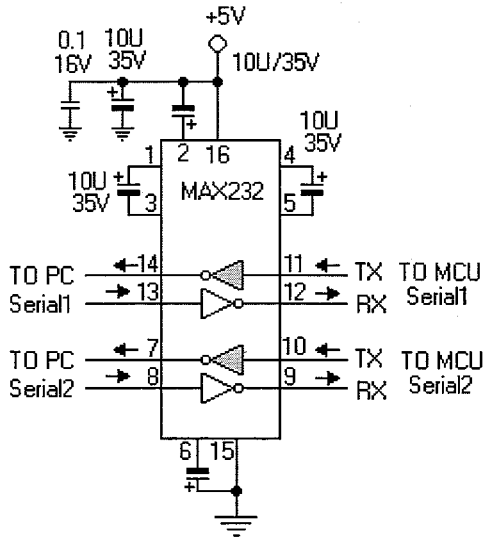
9 PIN D-SUB FEMALE to Computer 1.

9 PIN D-SUB FEMALE to Computer 2.

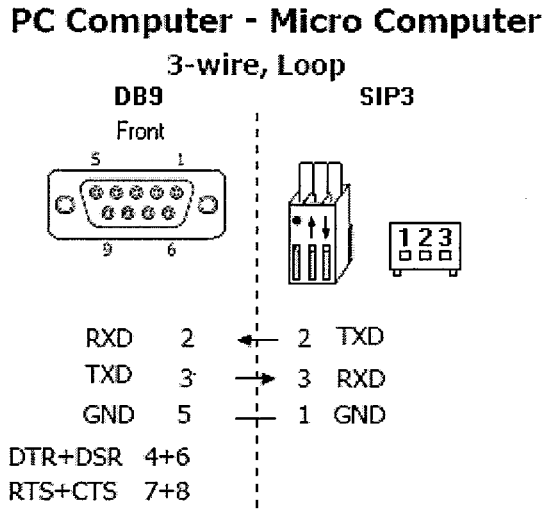
	D-Sub 1	D-Sub 2	
Receive Data	2	3	Transmit Data
Transmit Data	3	2	Receive Data
Data Terminal Ready	4	6+1	Data Set Ready + Carrier Detect
System Ground	5	5	System Ground
Data Set Ready + Carrier Detect	6+1	4	Data Terminal Ready
Request to Send	7	8	Clear to Send
Clear to Send	8	7	Request to Send

รูปที่ 4.6 แสดงขาสัญญาณภายในการเชื่อมต่อ RS232

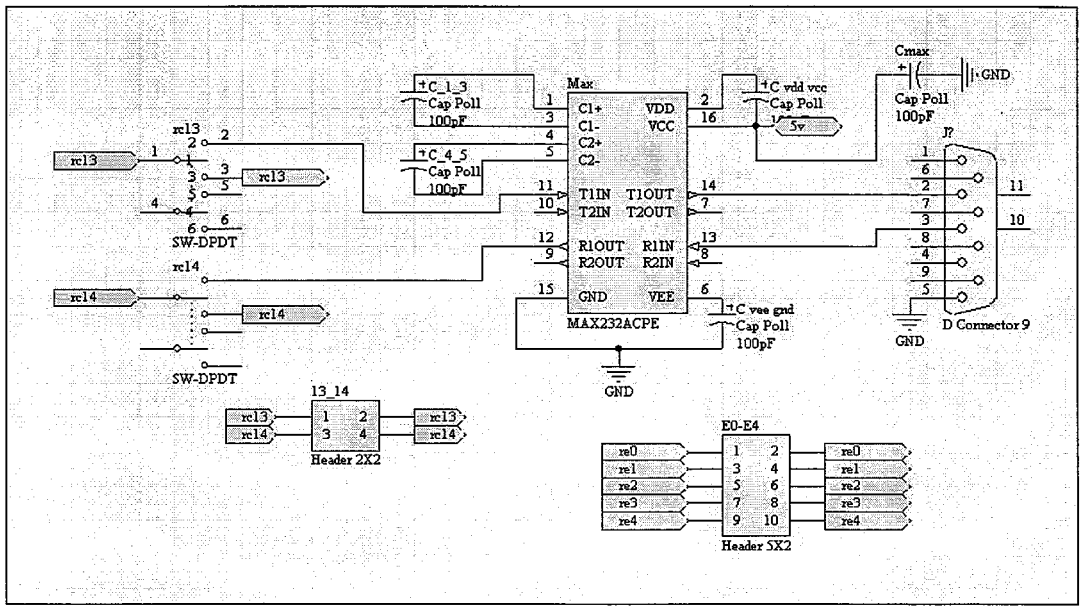
ในที่นี้จะใช้ไอซี MAX 232 เป็นตัวส่งและรับข้อมูลการทำงานของระดับสัญญาณระหว่างการเชื่อมต่อไมโครคอนโทรลเลอร์ติดต่อกับคอมพิวเตอร์ โดย MAX232 , CL 232 จะเป็นไอซีที่แปลงระดับสัญญาณของ RS-232 มาเป็นระดับ TTL และในทำนองเดียวกันก็แปลงระดับสัญญาณ TTL ไปเป็นระดับสัญญาณ RS-232



รูปที่ 4.7 แสดงตำแหน่งของขาไอซีการต่อใช้งานไอซี MAX232 การต่อใช้งาน MAX232 กับไมโครคอนโทรลเลอร์

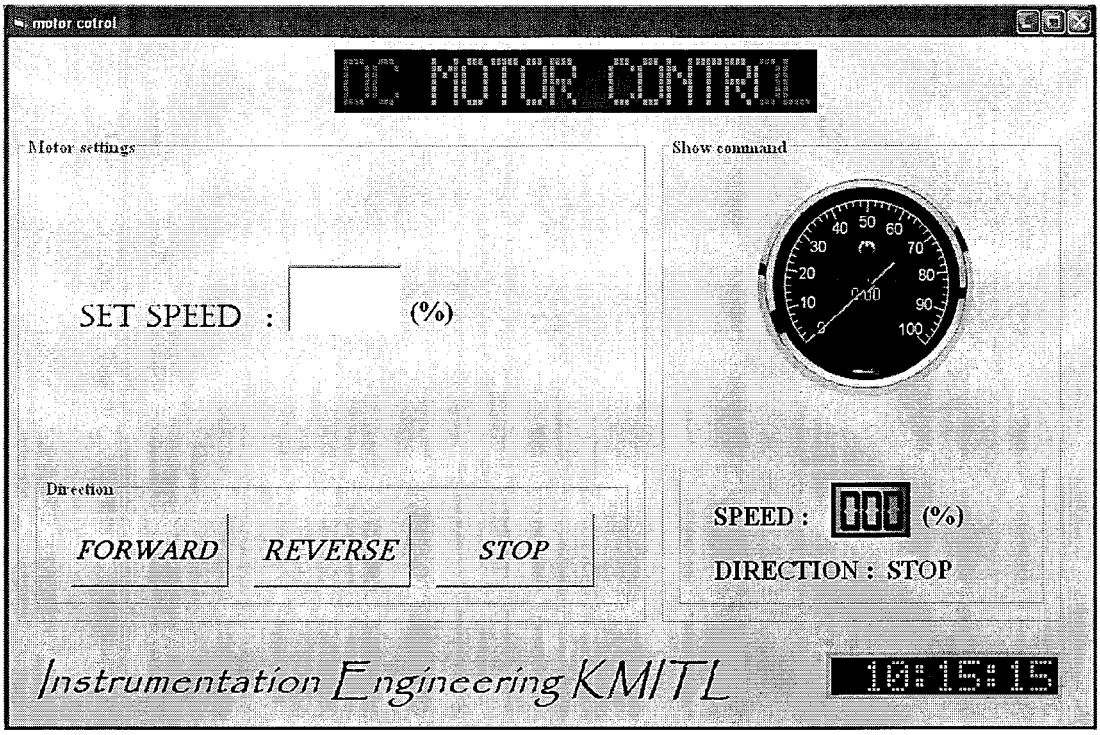


รูปที่ 4.8 แสดงการต่อใช้งานระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์



รูปที่ 4.9 แผนภาพวงจร RS 232

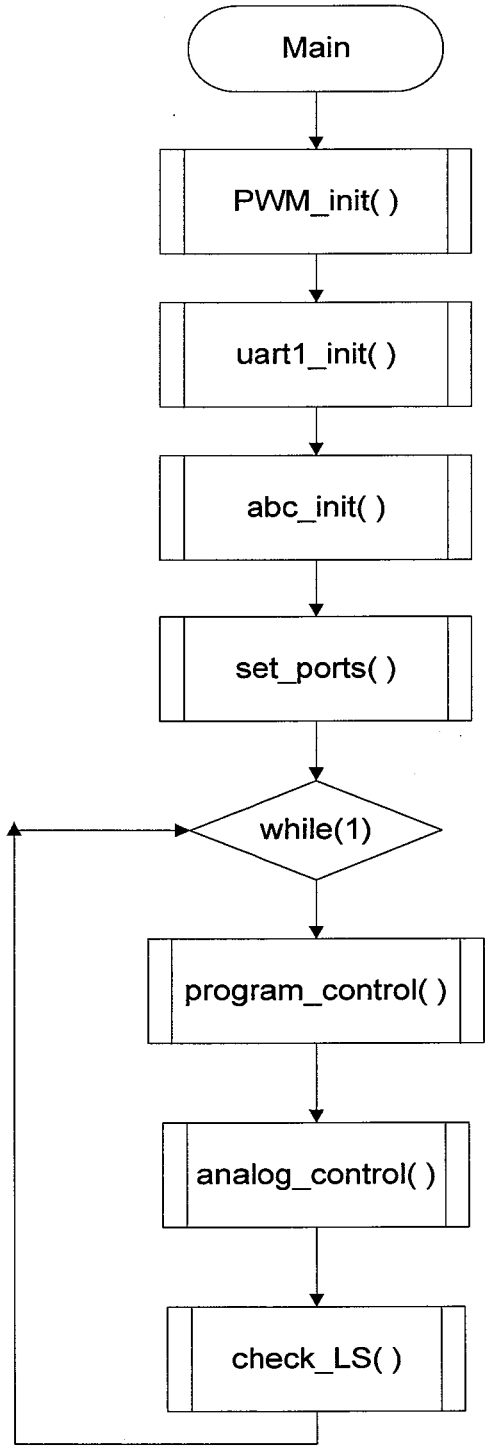
4.2 ออกแบบทางซอฟต์แวร์



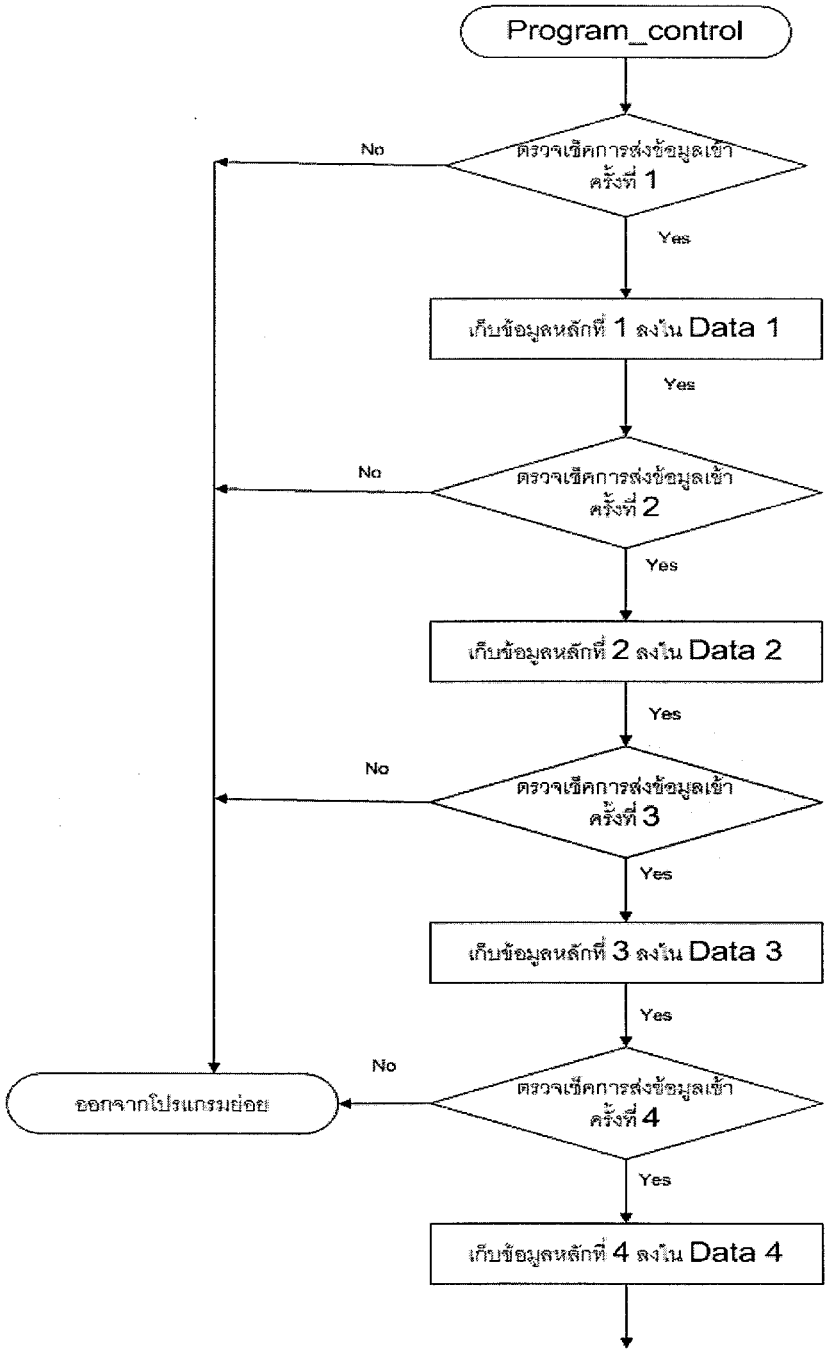
รูปที่ 4.10 หน้าต่างโปรแกรม motor control

การใช้งานโปรแกรม

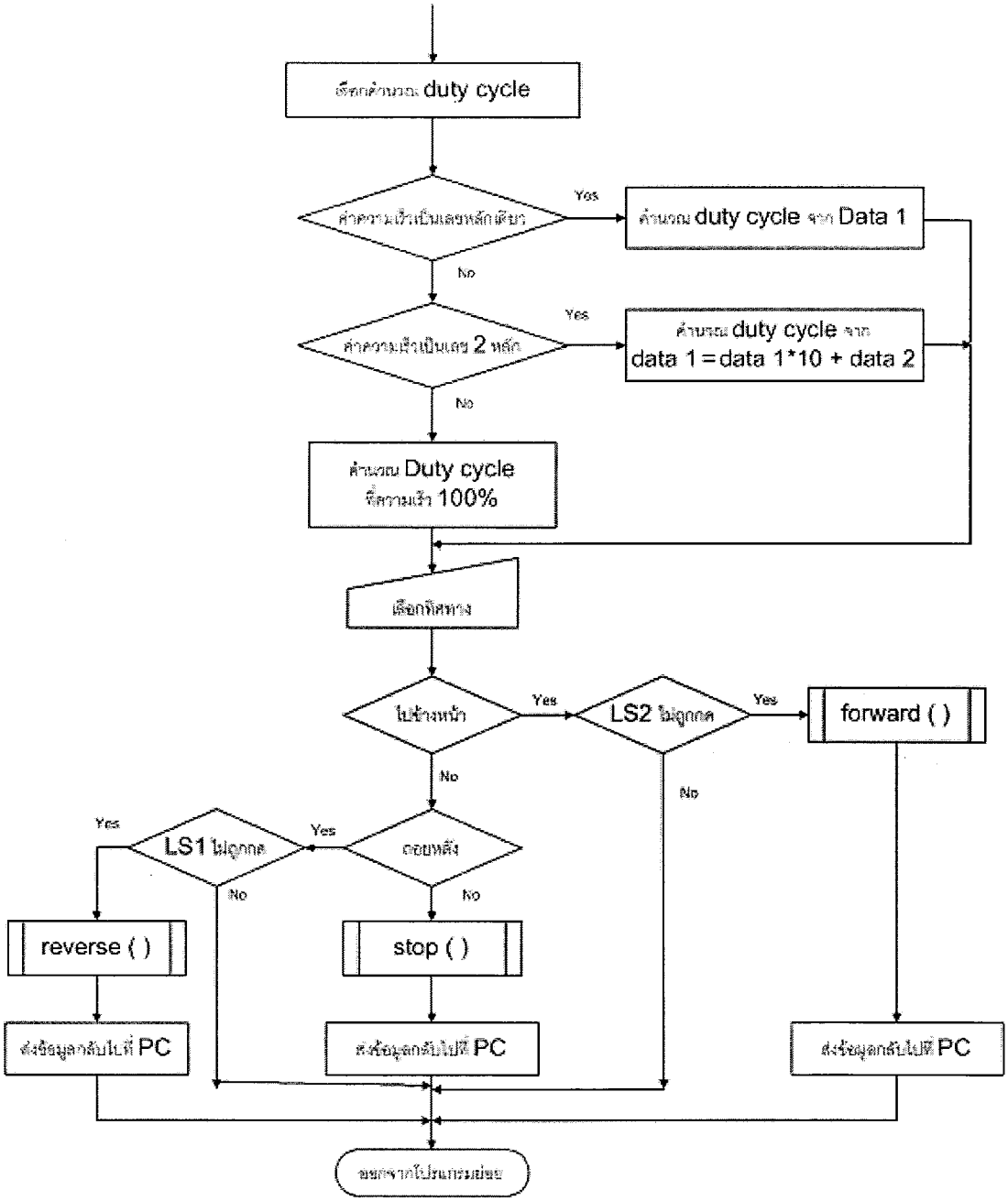
การใช้งานโปรแกรมเริ่มจากใส่ค่าความเร็วในรูปของ (%) ลงในช่อง SET SPEED จากนั้นกดเลือกทิศทางที่ต้องการ ความเร็วและทิศทางที่เลือกจะถูกส่งไปให้ dsPIC ประมวลผลต่อไป หน้าจอฝั่งขวาจะเป็นการแสดงความเร็วและทิศทางที่ dsPIC สั่งงานไปที่มอเตอร์ทั้งการสั่งงานจากซอร์ฟแวร์และฮาร์ดแวร์ก็จะถูกนำมาแสดงพร้อมกันในหน้าต่างนี้



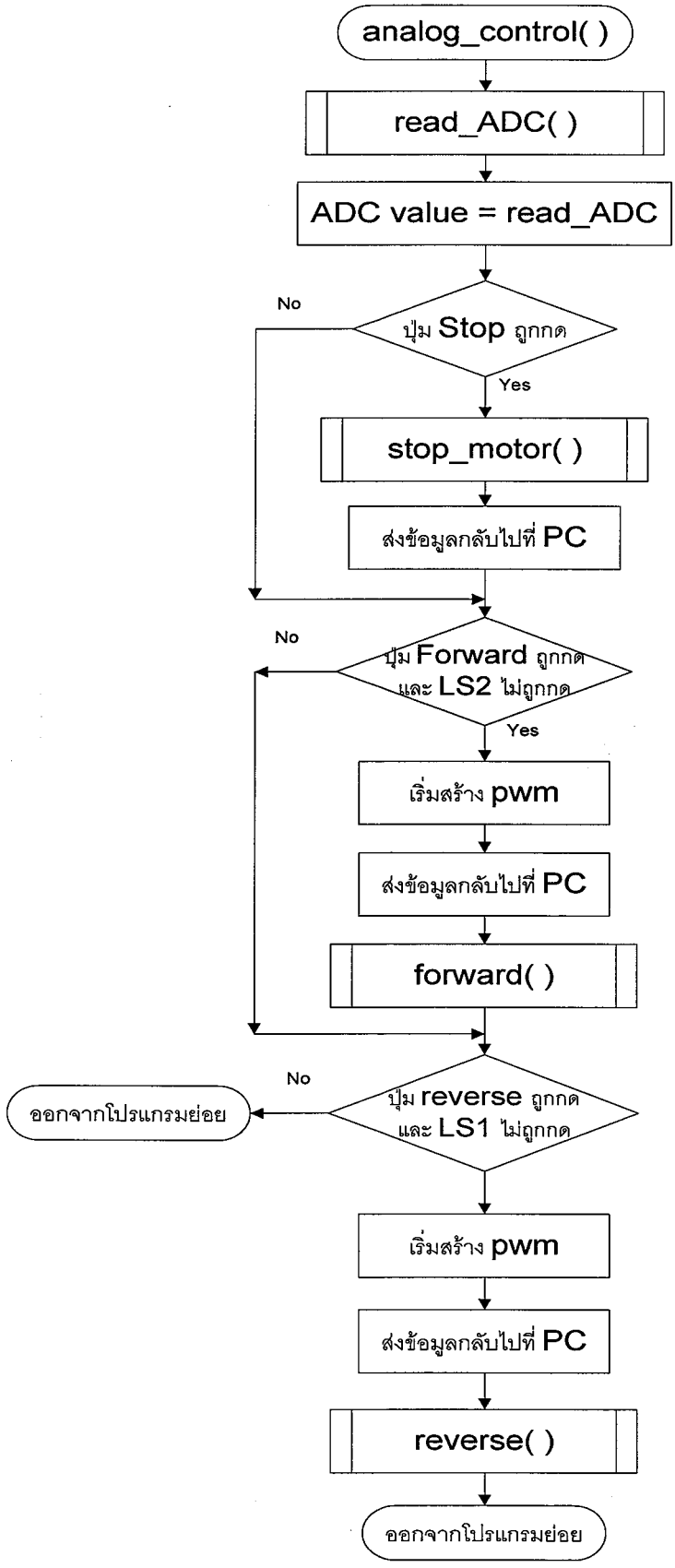
รูปที่ 4.11 การทำงานของโปรแกรมหลัก



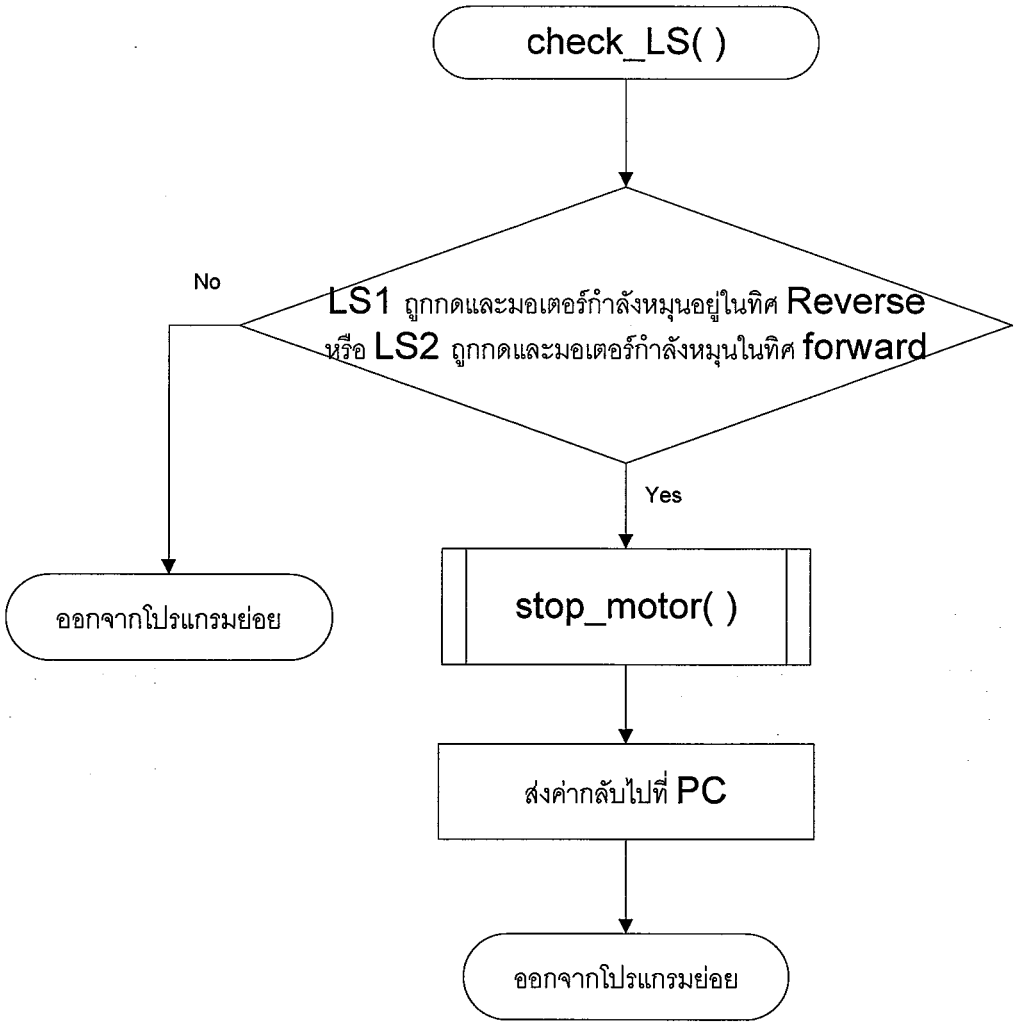
รูปที่ 4.12 การทำงานของโปรแกรมย่อย (Program Control)



รูปที่ 4.13 การทำงานของโปรแกรมย่อย (Program Control) ต่อ



รูปที่ 4.14 การทำงานของโปรแกรมย่อย (Program Analog Control)



รูปที่ 4.15 การทำงานของโปรแกรมย่อย (Program Analog Control) ต่อ

อธิบายการทำงานของซอฟแวร์

การทำงานของโปรแกรมหลักเริ่มจากการเซ็ทค่า Configuration ให้แก่โมดูล MCPWM, UART, ADC และกำหนดขาพอร์ตที่ต้องการเรียนใช้งานทั้งหมดก่อนที่จะเข้า infinite loop เพื่อวนตรวจสอบ input ที่เข้ามา ทั้งจากปุ่มกดและจาก PC เพื่อนำไปสร้างพัลส์และตรวจสอบการชนของลิมิตสวิทซ์

Program Control ทำหน้าที่สร้างสัญญาณ PWM ตามค่าที่ PC ส่งมาให้โดย PC จะส่งตัวเลขมาทั้งหมด 4 ตัว โดย 2 ตัวแรกเป็นค่าความเร็ว ตัวที่ 3 เป็นตัวบ่งบอกถึงจำนวนหลัก ถ้าเป็น "0" หมายถึงความเร็วที่ใส่เป็นเลขหลักเดียว ถ้าเป็น "1" หมายถึงเป็นเลขสองหลัก และถ้าเป็น "2" จะหมายถึงหมุนด้วยความเร็ว 100 % ตัวที่ 4 จะบอกถึงทิศทาง ถ้าเป็น "0" หมายถึง Reverse ถ้าเป็น "1" จะหมายถึง Forward ซึ่งโปรแกรมจะทำการตรวจสอบค่าที่รับมา เช่น 2001 หมายถึง หมุน Forward ด้วยความเร็ว 2 % , 5010 หมายถึง หมุน Reverse ด้วยความเร็ว 50 % เป็นต้น

หลังจากนั้น dsPIC จะส่งค่ากลับไปที่ VB เพื่อแสดงผลความเร็วและทิศทางที่ได้สั่งออกทางขาพอร์ตไป

Analog Control ทำหน้าที่สร้างสัญญาณ PWM จากการปรับค่าความต้านทานโดยจะสุ่มสัญญาณแรงดันที่รับมาเข้ามา 0-5 Vdc แล้วแปลงเป็นค่าเป็นตัวเลขฐานสิบ 0-1023 แล้วคำนวณออกมาเป็นสัญญาณ PWM โดยดูทิศทางจากปุ่มกดแล้วจึงส่งค่ากลับไปแสดงผลที่คอมพิวเตอร์

Check_LS มีหน้าที่ตรวจสอบว่าแท่นเลื่อนมาถึงจุดสิ้นสุดแล้วหรือยังโดยรับสัญญาณจาก limit switch ที่ติดอยู่ทั้งสองด้าน เพื่อเรียกใช้ฟังก์ชั่น stop motor แล้วจึงส่งค่ากลับไปแสดงผลที่ PC

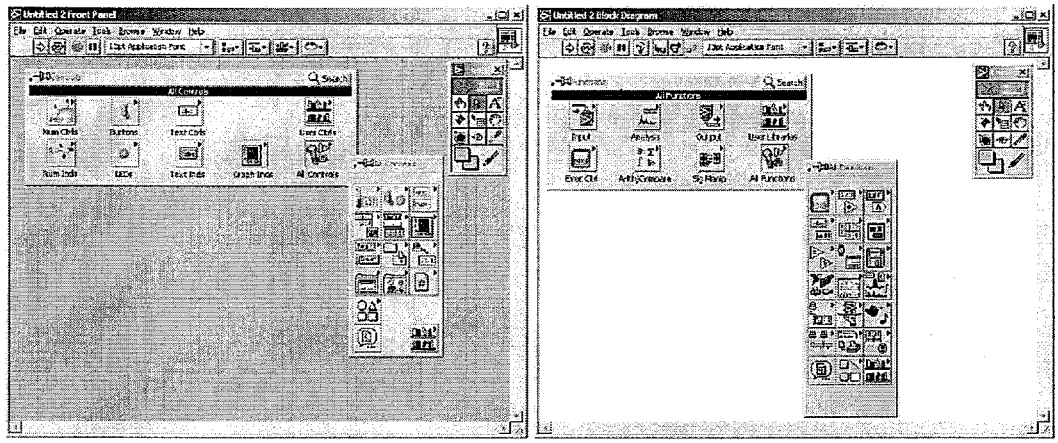
4.3 แนะนำโปรแกรม LabVIEW เบื้องต้น

LabVIEW ย่อมาจาก (Laboratory Virtual Instrument Engineering Workbench) เป็น Software ที่พัฒนาขึ้นโดย บริษัท National Instruments หรือ NI โดย LabVIEW จัดเป็น Software ประเภทภาษารูปภาพ (Graphical Programming Language) คือจะใช้บล็อกฟังก์ชันซึ่งแทนด้วยรูปไอคอน (Icon) แทนการเขียนด้วยตัวอักษร (Text-based Program) และใช้เส้นเชื่อมต่อระหว่างบล็อกฟังก์ชันแทนการไหลของข้อมูลระหว่างโปรแกรมย่อยคล้ายกับการเขียน Flow chart ของโปรแกรมแต่ไม่เหมือนกับการเขียนโปรแกรมด้วยภาษา (Complier) อื่น ๆ เช่น C , Pascal ซึ่งเป็นการเขียนลักษณะ Text mode โดยส่วนมากต้องคำนึงถึงชนิดของข้อมูลที่รับส่งระหว่างโปรแกรมย่อย ๆ เช่น real, Integer เป็นต้น การเขียนโปรแกรมด้วย LabVIEW สามารถลดเวลาในการพัฒนาโปรแกรมลงได้อย่างมากเมื่อเทียบกับการเขียนโปรแกรมด้วยภาษาอื่นที่ใช้ตัวอักษร สิ่งที่ทำให้โปรแกรม LabVIEW แตกต่างจาก Software อื่นก็คือ ความสามารถในการใช้งานด้าน งานวัดและการควบคุมอัตโนมัติ โดยมีเครื่องมือ (Tools) ต่าง ๆ ที่สนับสนุนการใช้งานด้านนี้ไว้อย่างมากมาย และให้ผลลัพธ์ออกมาในรูปแบบของเครื่องมือเสมือนจริง (Virtual Instrument หรือ VI) LabVIEW

LabVIEW เป็นโปรแกรมที่ใช้เพื่อสร้างโปรแกรมสำหรับการเก็บข้อมูล, การเชื่อมต่อกับอุปกรณ์หรือเครื่องมือวัดต่าง ๆ และการควบคุมระบบต่าง ๆ LabVIEW จะใช้ภาษากราฟฟิกในการสร้างโปรแกรม ซึ่งแตกต่างจากโปรแกรมอื่น ๆ ที่ใช้ตัวอักษรเพื่อสร้างโปรแกรม เช่น C/C++, Visual C++, Visual Basic, etc.โปรแกรม LabVIEW มีฟังก์ชั่น (Functions) และเครื่องมือ (Tools) ดังรูปที่ 1 เพื่อช่วยให้ผู้ใช้สามารถนำมาสร้างโปรแกรมสำหรับงานประยุกต์ต่าง ๆ เช่น loops, case statements, arrays, string, file I/O, data acquisition, instrument control, analysis tools เป็นต้น โปรแกรม LabVIEW โดยปกติจะเรียกว่า Virtual Instrument (VI) ซึ่งประกอบด้วย 2 ส่วนคือ

Front Panel (รูปที่ 4.16 (ก))

Block Diagram (รูปที่ 4.16 (ข)) โดยแต่ละส่วนจะใช้สำหรับวัตถุประสงค์ที่แตกต่างกัน

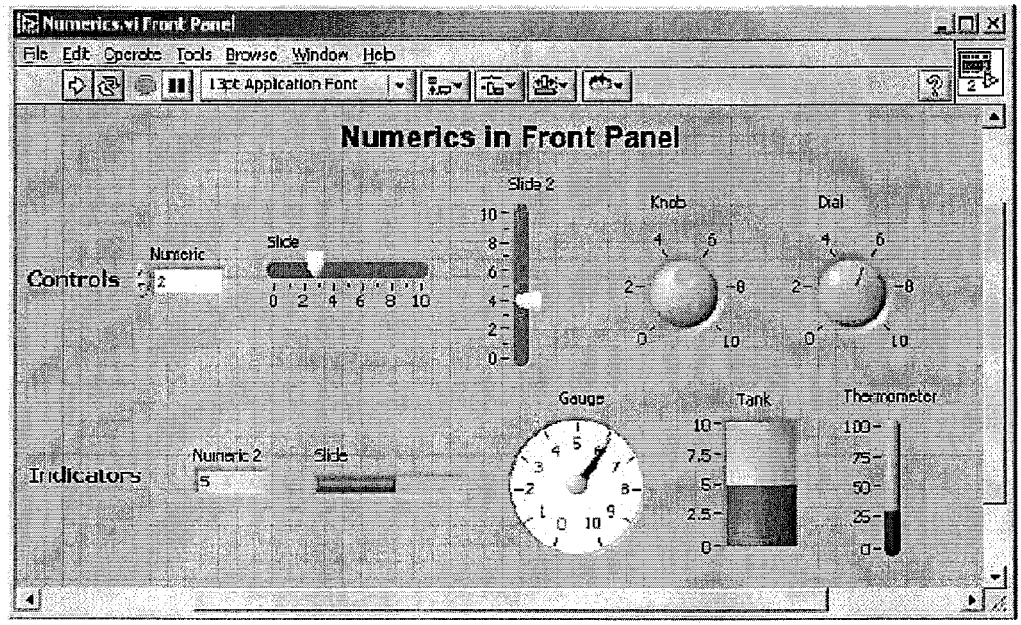


รูปที่ 4.16 Front Panel (ก) และ Block Diagram (ข) รวมทั้ง Tools, Functions และ Controls ต่าง ๆ ในโปรแกรม LabVIEW

Front Panel

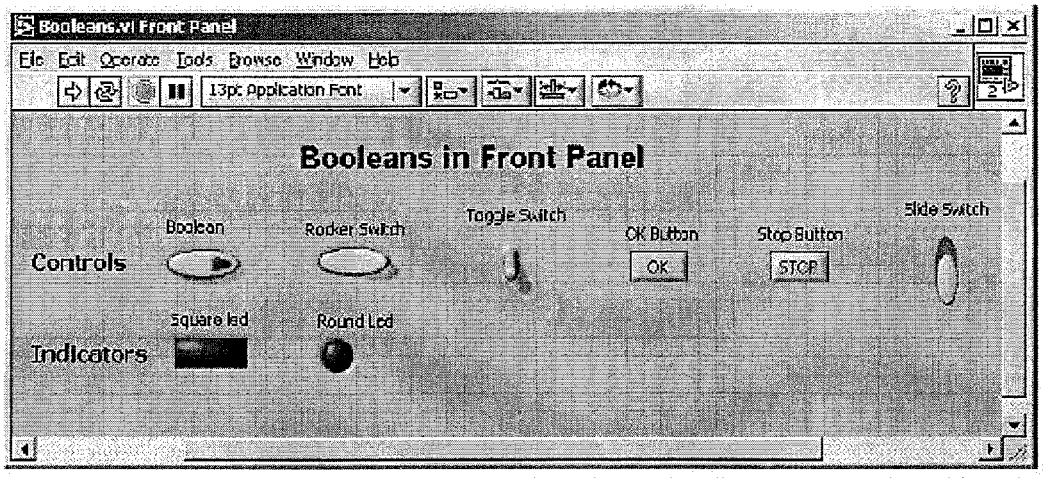
Front Panel เป็นส่วนที่ใช้สำหรับเชื่อมต่อกับผู้ใช้งาน ใช้สำหรับใส่ค่า (input) และแสดงผล (output) ของตัวโปรแกรมที่สร้างขึ้นมา โดยส่วน input จะถูกเรียกว่า "control" และส่วน output จะเรียกว่า "indicator" ตัว control และ indicator ที่ถูกนำมาใช้ใน Front Panel ซึ่งจะมีจุดต่อเชื่อมปรากฏอยู่ที่ Block Diagram ด้วย เมื่อโปรแกรมเริ่มทำงานตัว control ที่ Front Panel จะทำการส่งข้อมูลผ่านไปยัง Block Diagram และตัว output ก็จะส่งค่าจาก Block Diagram กลับมาแสดงผลที่ Front Panel ผ่านตัว indicator ที่กำหนดไว้ข้อมูลที่ใช้ในตัว control , indicator

Numerics มีอยู่ด้วยกันหลายรูปแบบ เช่น ตัวเลขดิจิตอล, เกจ,หรือ สเกลแบบต่าง ๆ ดังรูปที่ 4.17



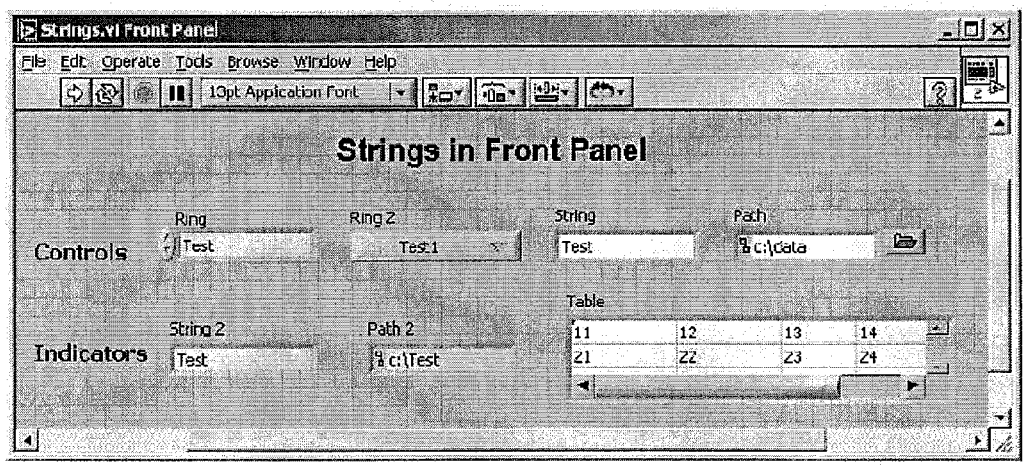
รูปที่ 4.17 ตัวอย่างของตัว Control และ Indicator แบบ Numerics ใน Front Panel

Booleans เป็นเงื่อนไข มี 2 สถานะคือ on/off หรือ True/False รูปแบบของ Boolean จะเป็นปุ่มสี, ปุ่มกดหยุด หรือปุ่มเลื่อนเปิดหรือปิด ดังแสดงในรูปที่ 4.18



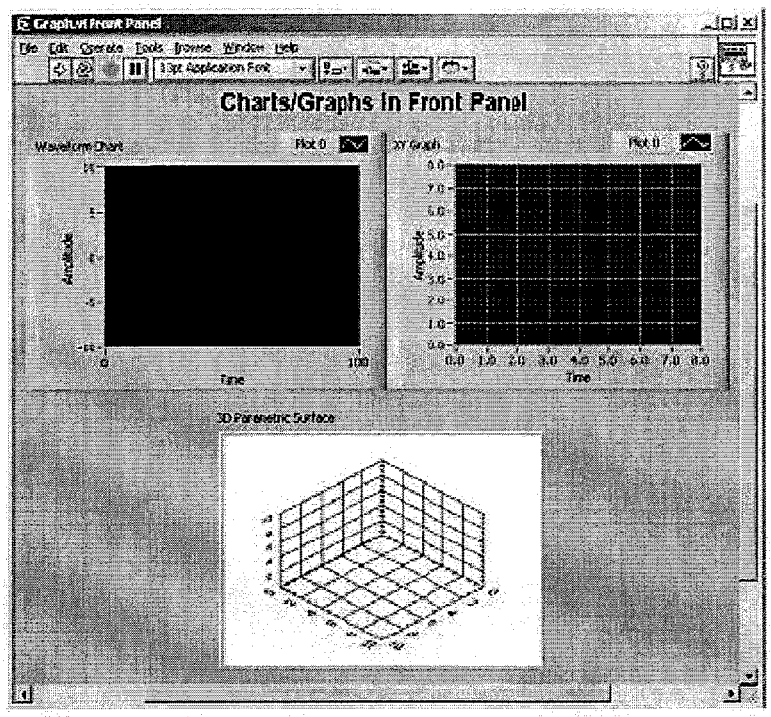
รูปที่ 4.18 ตัวอย่างของตัว Control และ Indicator แบบ Strings Boolean ใน Front Panel

String เป็นตัวอักษรที่ใช้สำหรับป้อนค่า, แสดงผล หรือจัดเก็บไว้ในรูปของไฟล์ ดังแสดงในรูปที่ 4.19



รูปที่ 4.19 ตัวอย่างของตัว Control และ Indicator แบบ Strings ใน Front Panel

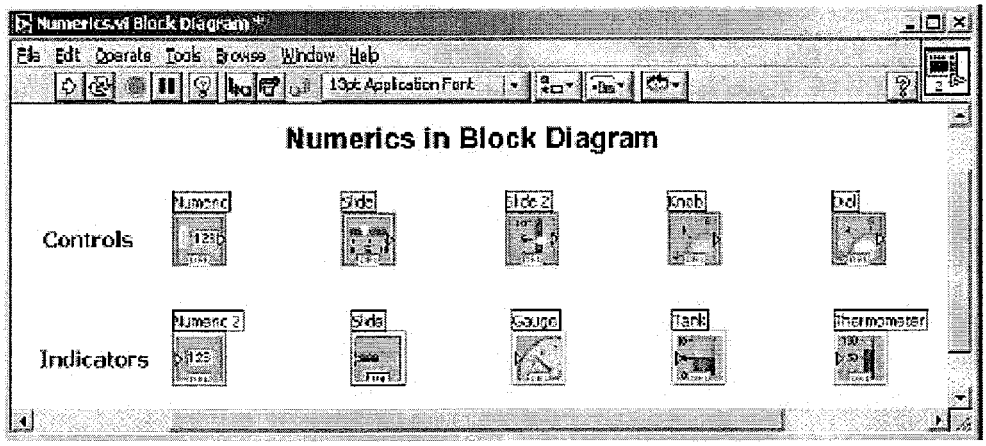
Charts/Graphs เป็น indicator ที่ใช้แสดงข้อมูลเทียบกับเวลา ซึ่งมีทั้งแบบ 2 มิติ และ 3 มิติ ดังแสดงในรูปที่ 4.20



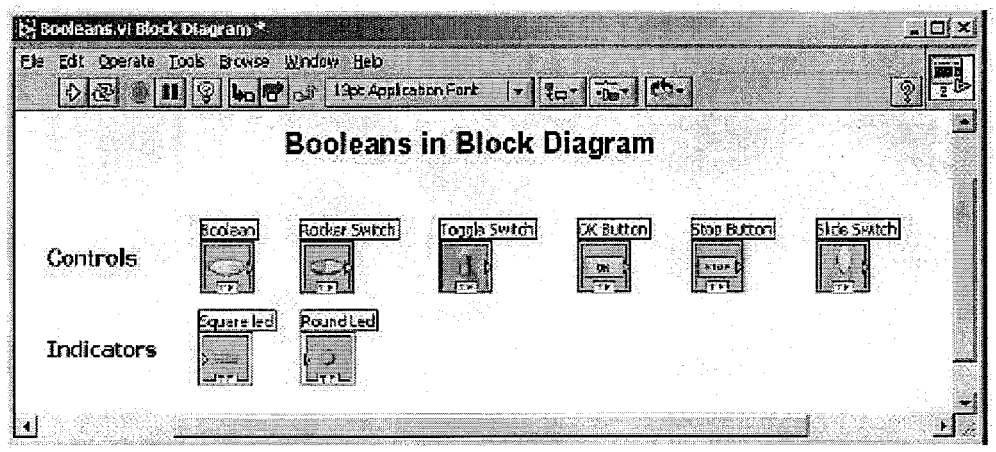
รูปที่ 4.20 ตัวอย่างของ Charts/Graphs แบบต่าง ๆ ใน Front Panel

Block Diagram

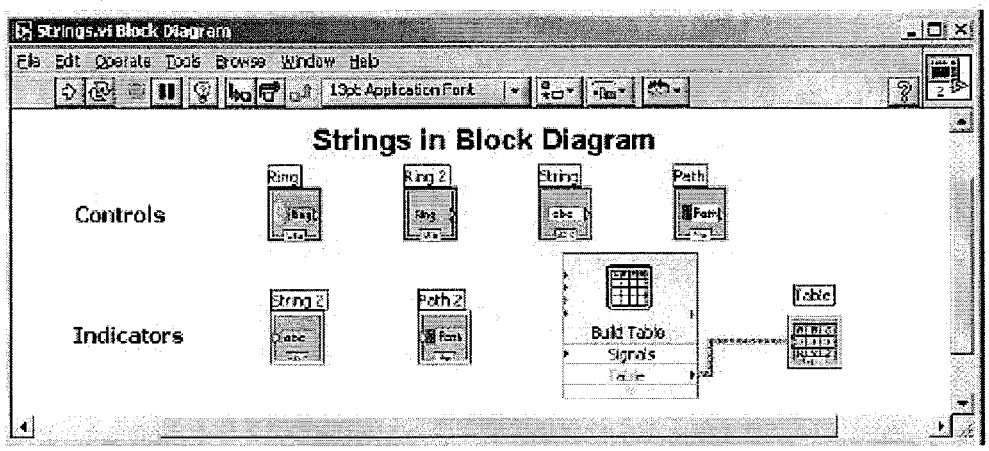
Block Diagram เป็นส่วนที่เก็บ "Source Code" ของโปรแกรม LabVIEW ซึ่งตัวโปรแกรมใน LabVIEW จะเรียกว่า "VI" ตัว Code ในโปรแกรม LabVIEW เป็นกราฟฟิกที่เรียกกันว่า G (Graphical) programming หลักการของโปรแกรมจะเชื่อมต่อตัวจุดเชื่อมต่าง ๆ เข้าด้วยกันแทนที่จะเขียนโดยใช้คำสั่งต่าง ๆ ดังที่ใช้ทั่วไปในโปรแกรมอื่น ๆ เช่น C/C++, Visual C++ ซึ่งอาจจะกล่าวได้ว่า LabVIEW ใช้หลักการเดียวกันกับการเขียน flow chart ตัวอย่างของ Numerics, Booleans, Strings และ Charts/Graphs ใน Block Diagram มีรูปแบบแตกต่างกันดังแสดงในรูปที่ 4.21



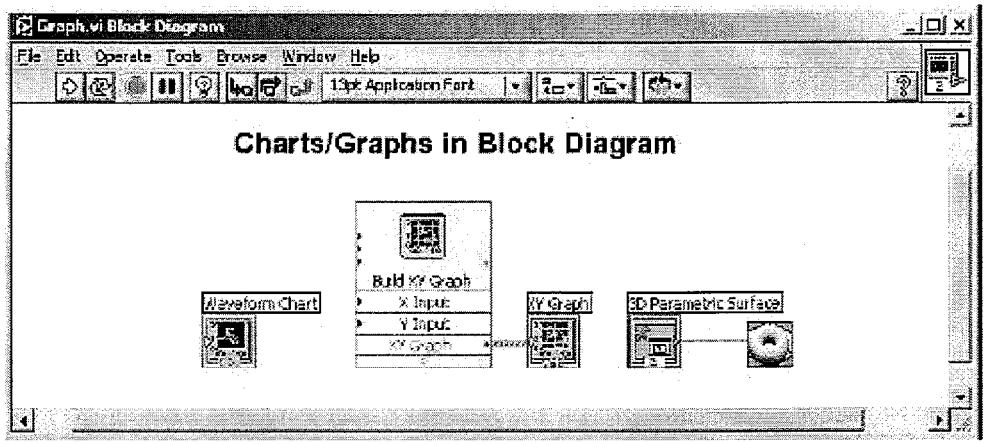
(ก) Numerics



(ข) Booleans



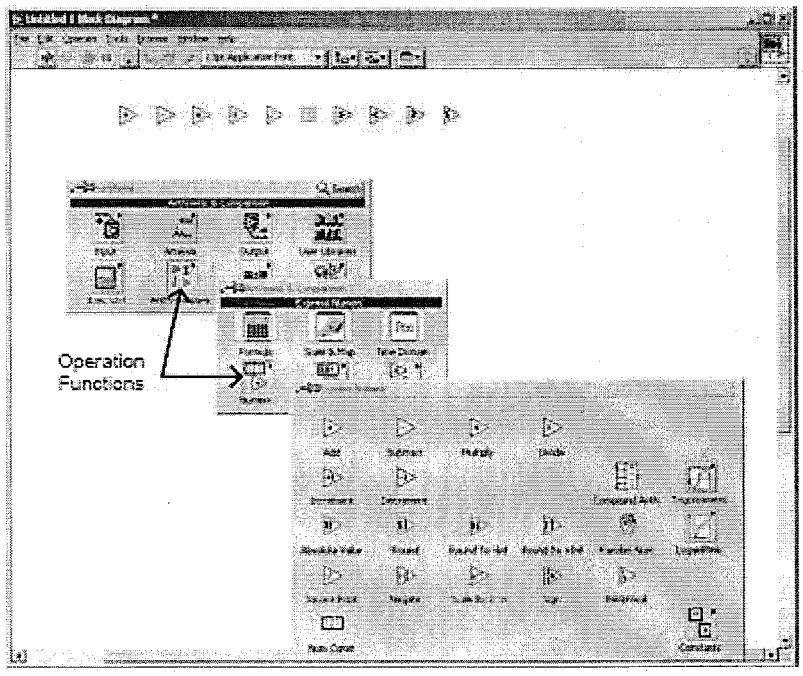
(ค) Strings



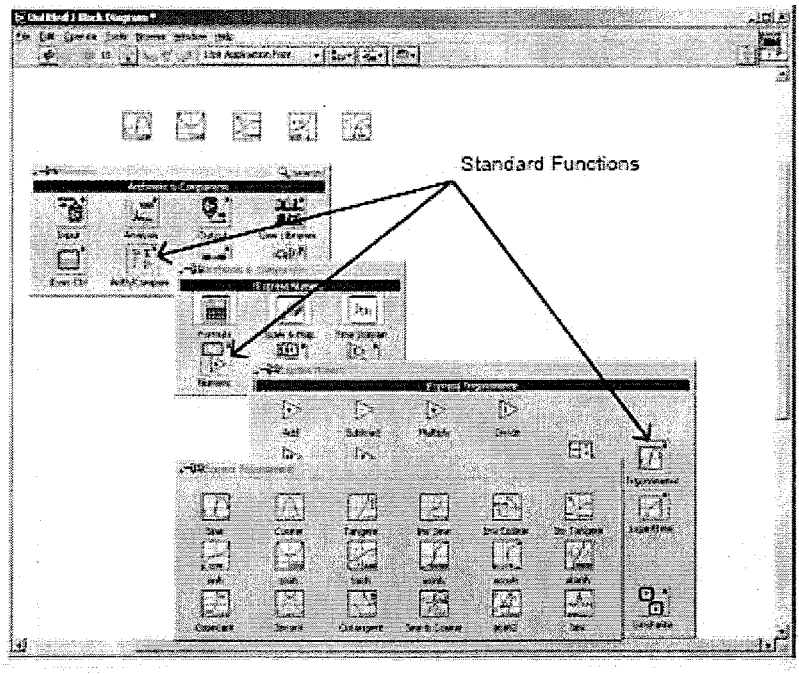
(ง) Charts/Graphs

รูปที่ 4.21 ตัวอย่างของ Numerics, Booleans, Strings และ Charts/Graphs ใน Block Diagram

โปรแกรม LabVIEW มีฟังก์ชันต่าง ๆ ที่ใช้ช่วยในการพัฒนาโปรแกรมที่มีความซ้ำซ้อน เช่น ตัว operation (+, -, *. /, ^ เป็นต้น) ค่าคงที่ (π ,*e*,ln 2,ln *e*,log10 เป็นต้น) ฟังก์ชันมาตรฐานต่าง ๆ (sin, cos, tan, atan เป็นต้น) Loop & Case Structures (while loop, for loop, case เป็นต้น) และ ฟังก์ชัน File input/output (File I/O) ฟังก์ชันต่าง ๆ ที่กล่าวถึงจะปรากฏอยู่ใน Block diagram เท่านั้นดังแสดงในรูปที่ 4.16-4.21 Operation เป็นฟังก์ชันที่ใช้ช่วยเพื่อสร้างสมการหรือคำสั่งในโปรแกรม LabVIEW ตามวัตถุประสงค์ที่กำหนดไว้ (ดังรูปที่ 4.22) ส่วนรูปที่ 4.23 แสดงฟังก์ชันพื้นฐานที่นิยมใช้สำหรับการเขียนโปรแกรมทางวิศวกรรม

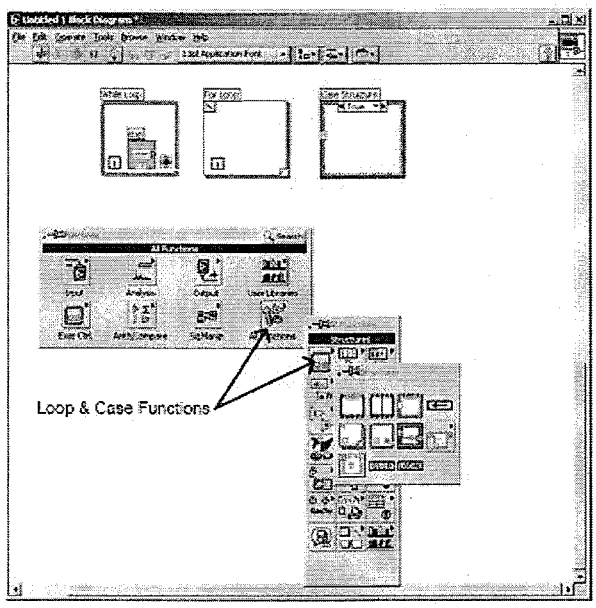


รูปที่ 4.22 ตัวอย่างของ Operation ต่าง ๆ



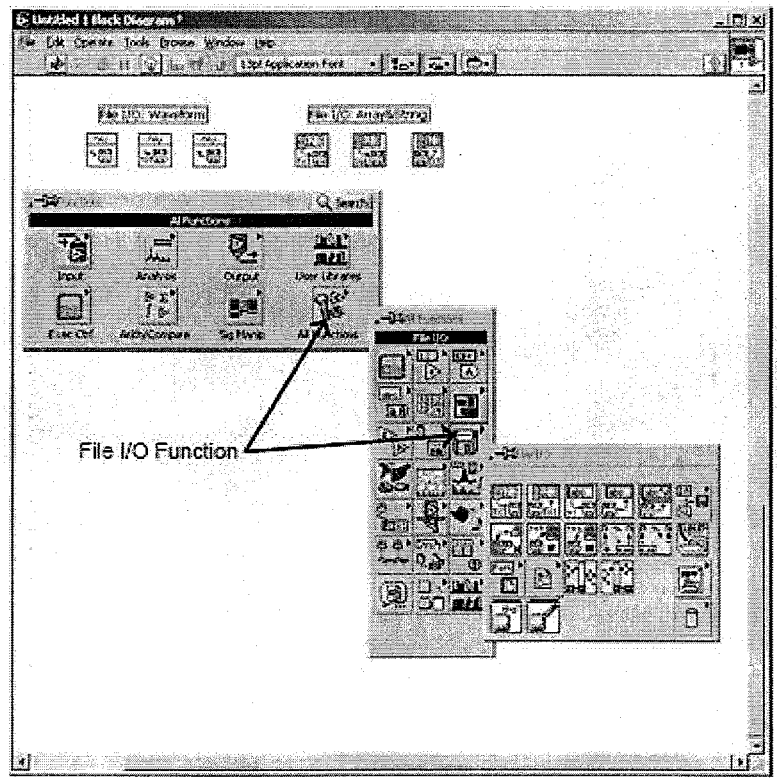
รูปที่ 4.23 ตัวอย่างของฟังก์ชันมาตรฐานต่าง ๆ

Loop & Case Structures เป็นฟังก์ชันพื้นฐานสำหรับการเขียนโปรแกรมดังรูปที่ 4.24 ซึ่งมีไว้ช่วยสำหรับการเขียนโปรแกรมที่ต้องมีขั้นตอนการทำงานซ้ำหลาย ๆ ครั้งโดยรูปแบบของฟังก์ชันขึ้นอยู่กับลักษณะของกระบวนการ เช่น While Loop กระบวนการจะทำซ้ำจนกระทั่งบรรลุตามเงื่อนไขที่กำหนดไว้ หากเป็น For Loop ตัวโปรแกรมจะกระทำซ้ำตามจำนวนครั้งที่กำหนด และในกรณีของ Case Structure ตัวโปรแกรมจะกระทำเมื่อค่าเริ่มต้นตรงตามข้อกำหนดของเงื่อนไขนั้น ๆ

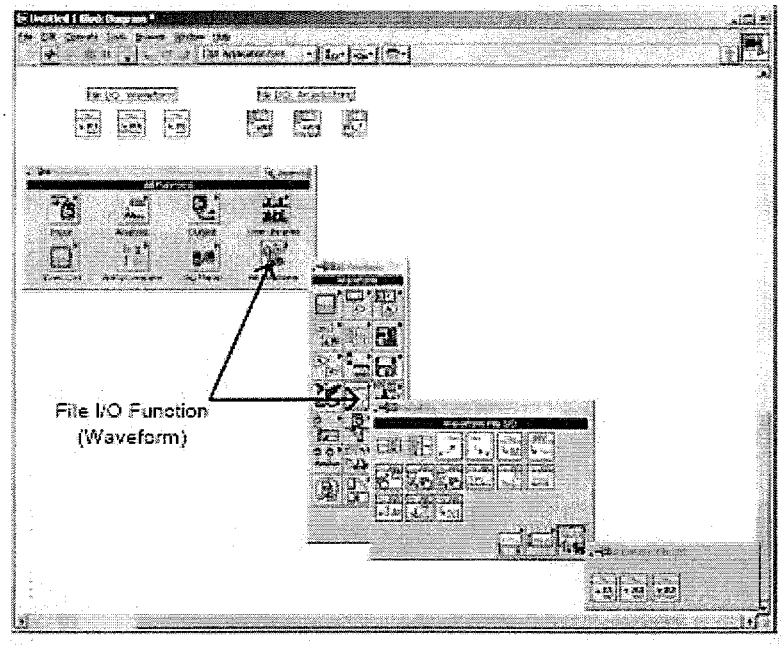


รูปที่ 4.24 ตัวอย่างของ Loop & Case Structures

โปรแกรม LabVIEW มีฟังก์ชัน File Input/Output (I/O) ที่สามารถนำมาใช้เพื่อจัดเก็บข้อมูลในรูปของไฟล์ทั้งแบบ Binary และ ASCII (text) Format โดยสามารถนำมาวิเคราะห์และจัดทำรายงานหรือกราฟในภายหลังได้ ฟังก์ชัน File I/O ประกอบด้วยฟังก์ชันอ่าน (Read) ข้อมูลจากไฟล์และจัดเก็บ (Write) ข้อมูลในรูปของไฟล์ ส่วนวิธีการใช้ฟังก์ชันขึ้นอยู่กับชนิดของข้อมูล (String, Array, Integer, Cluster เป็นต้น) และformat ของไฟล์ (binary หรือ ASCII) ที่ต้องการอ่านหรือจัดเก็บ ดังแสดงในรูปที่ 4.25-4.26

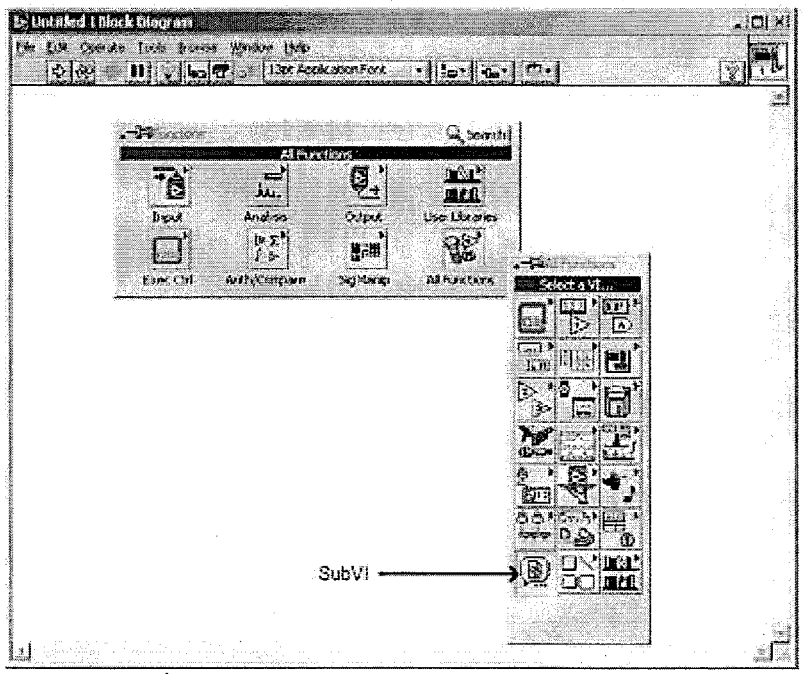


รูปที่ 4.25 ฟังก์ชัน File I/O สำหรับข้อมูลแบบต่าง ๆ



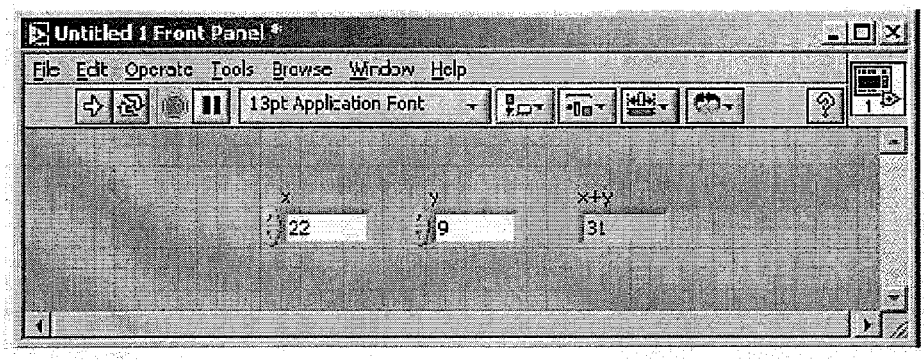
รูปที่ 4.26 ฟังก์ชัน File I/O สำหรับข้อมูลแบบ Waveform (Cluster)

นอกจากฟังก์ชันต่าง ๆ ที่มีอยู่ในโปรแกรม LabVIEW ผู้ใช้สามารถที่จะสร้างฟังก์ชันขึ้นเองตามวัตถุประสงค์การใช้งาน โดยตัวฟังก์ชันที่สร้างขึ้นนี้จะถูกเรียกว่า Sub-VI ซึ่งสามารถเรียกใช้ร่วมกับฟังก์ชันพื้นฐานอื่น ๆ วิธีเรียกใช้ฟังก์ชัน Sub-VI กระทำได้ดังรูปที่ 4.27 และเลือกฟังก์ชันที่ต้องการตามขั้นตอนที่โปรแกรม LabVIEW กำหนดไว้

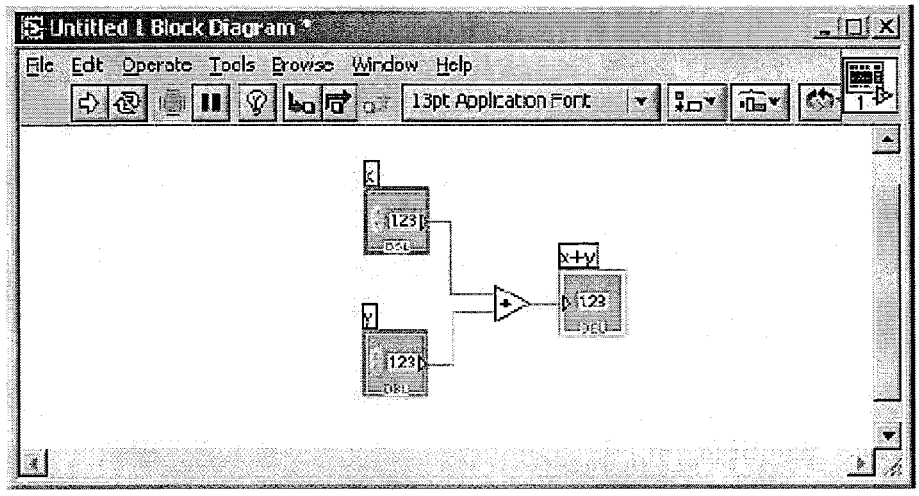


รูปที่ 4.27 Menu สำหรับเรียก SubVI

การเขียนโปรแกรม LabVIEW นั้นขึ้นอยู่กับวัตถุประสงค์ที่ต้องการใช้งาน ซึ่ง Front Panel และ Block Diagram จะถูกใช้ร่วมกันเพื่อเขียนโปรแกรม โดย Front Panel จะใช้เพื่อแสดงผลและติดต่อกับผู้ใช้ ส่วน Block Diagram จะใช้เพื่อเขียนและแสดงเส้นทางเดินของข้อมูล รวมทั้งเป็นตัวกำหนดค่าตัวแปรเริ่มต้นหรือตัวแปรควบคุมต่าง ๆ ที่จำเป็นต้องใช้ในโปรแกรม ตัวอย่างโปรแกรม LabVIEW เบื้องต้น เช่น โปรแกรมบวกเลข ซึ่งมีค่าเริ่มต้น 2 ค่า และมีผลลัพท์ 1 ค่า Front Panel และ Block Diagram ของตัวอย่างดังแสดงในรูปที่ 4.28 วิธีการ Run โปรแกรมกระทำได้โดยกดปุ่มลูกศรที่อยู่ทางด้านซ้ายบน



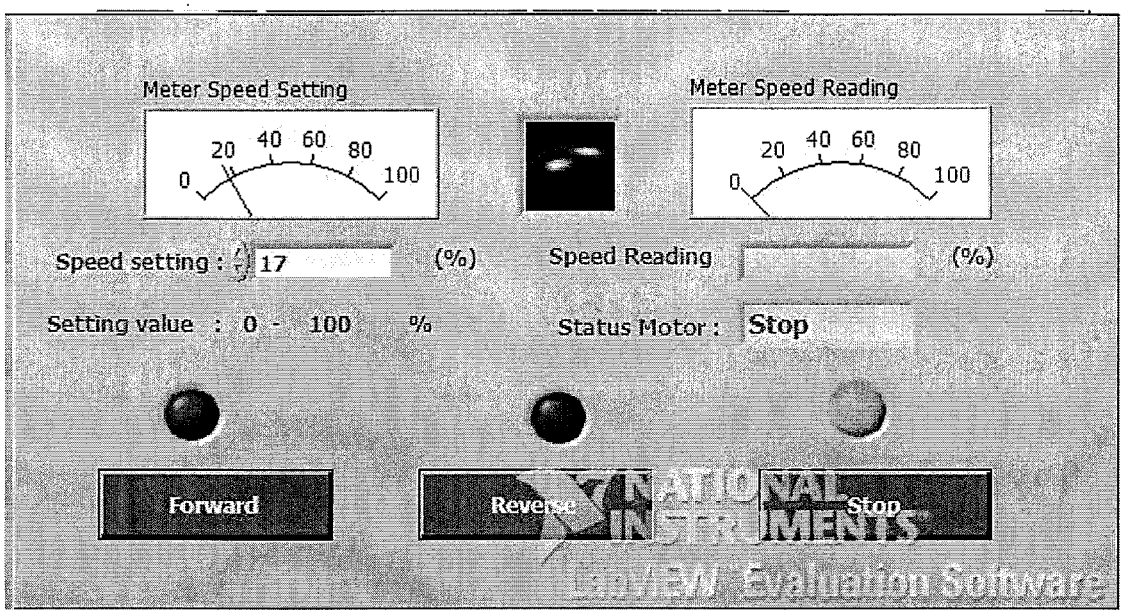
(ก) Front Panel ของโปรแกรมบวกเลข



(ข) Block Diagram ของโปรแกรมบวกเลข

รูปที่ 4.28 ตัวอย่างโปรแกรมบวกเลข

4.3.1 อธิบายการใช้งานโปรแกรม



รูปที่ 4.29 หน้าต่างโปรแกรม motor control (labview 8.0)

ขั้นตอนการใช้งาน

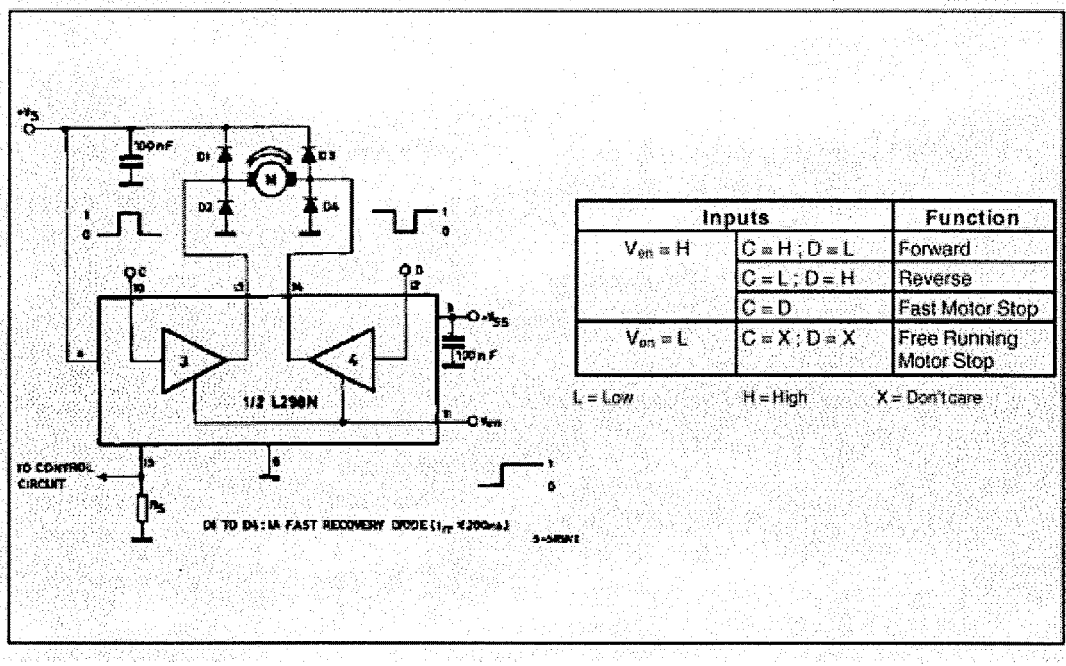
การใช้งานโปรแกรมเริ่มจากใส่ค่าความเร็วที่ต้องการในรูปของ (%) ลงในช่อง SPEED SETTING จากนั้นกดเลือกทิศทางที่ต้องการความเร็วและทิศทางที่เลือกจะถูกส่งไปให้ dsPIC ประมวลผลต่อไปหน้าจอ ฝั่งขวาจะเป็นการแสดงความเร็วและทิศทางที่ dsPIC สั่งงานไปที่มอเตอร์ ทั้งการสั่งงานจากซอร์ฟแวร์และฮาร์ดแวร์ก็จะถูกนำมาแสดงพร้อมกันในหน้าต่างนี้ โดยจะแสดงสถานะที่ทำงานอยู่ว่าหมุนไปในทิศทางไหน เมื่อต้องการหยุดการทำงานหรือสิ้นสุดการทำงานซอฟต์แวร์จะแสดงสถานะการหยุดขึ้น

บทที่ 5

การทดลอง

5.1 การขับเคลื่อนมอเตอร์กระแสตรง

: Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

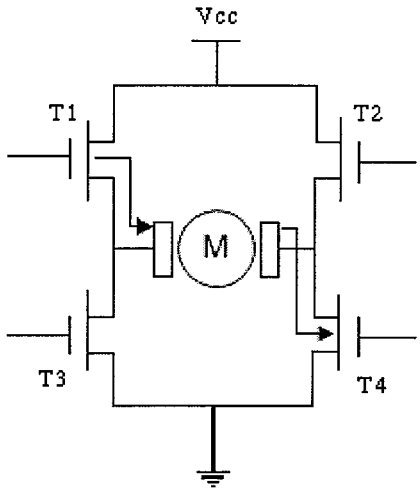
L = Low H = High X = Don't care

รูปที่ 5.1 แสดงการขับเคลื่อนมอเตอร์กระแสตรงด้วยไอซี L298

วงจรขับเคลื่อนมอเตอร์กระแสตรงจะใช้ Gate 4 ตัว ในการขับมอเตอร์ ซึ่ง Gate Driver ทำหน้าที่สวิตชิ่ง V_{Supply} (V_{cc}) จะได้ $V_{average}$ จ่ายให้แก่มอเตอร์และทิศทางการหมุนของมอเตอร์นั้น โดยทำการกำหนดค่า PWM1 และ PWM2 ดังตารางด้านล่าง

ตารางที่ 5.1 แสดงการกำหนดค่า PWM1 และ PWM2

PWM1		PWM2		ผลที่เกิดจากdsPIC			
H	L	H	L	Gate T1	Gate T2	Gate T3	Gate T4
ON	-	-	ON	ON	-	-	ON
กระแสจะไหลจาก Gate T1 ไปยัง Gate T4							

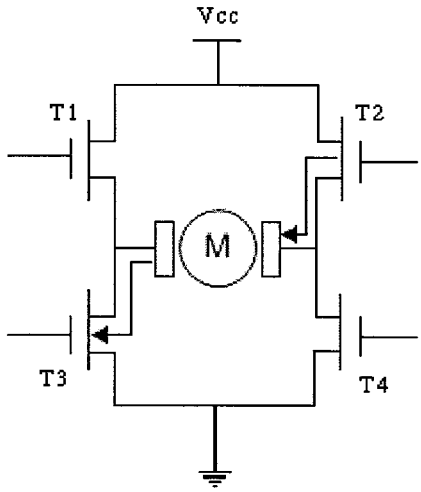


รูปที่ 5.2 แสดงการไหลของกระแสจาก Gate T1 ไปยัง Gate T4

การกลับทิศทางการหมุนมอเตอร์ทำได้โดยกำหนด PWM1 และ PWM2 ดังตารางด้านล่าง

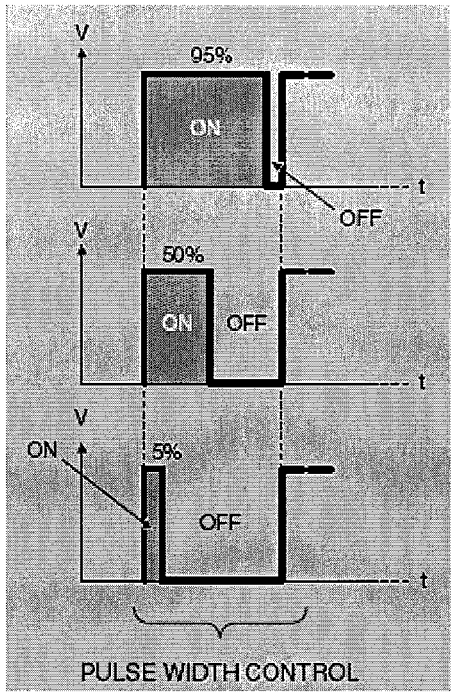
ตารางที่ 5.2 แสดงการกำหนดค่า PWM1 และ PWM2

PWM1		PWM2		ผลที่เกิดจากdsPIC			
H	L	H	L	Gate T1	Gate T2	Gate T3	Gate T4
-	ON	ON	-	-	ON	ON	-
กระแสจะไหลจาก Gate T2 ไปยัง Gate T3							



รูปที่ 5.3 แสดงการไหลของกระแสจาก Gate T2 ไปยัง Gate T3

ส่วนการปรับความเร็วรอบนั้นทำได้ โดยการปรับเปลี่ยนค่าของดิวตี้ไซเคิลด้วยตัวต้านทานปรับค่าได้ที่ต่ออยู่กับโมดูล A/D ของไมโครคอนโทรลเลอร์ dsPIC เช่นกันซึ่งโมดูล A/D นี้จะทำหน้าที่กำหนดค่าอ้างอิงให้กับรีจิสเตอร์ PDC ในโมดูล MCPWM โดยเทคนิคนี้สามารถใช้ได้ทั้งมอเตอร์กระแสตรงชนิดไม่มีแปรงถ่านและมอเตอร์เหนี่ยวนำ



รูปที่ 5.4 แสดงการปรับค่าดิวตี้ไซเคิล

จากสมการ $V_{average}$ = V_{Supply} x Duty Cycle หากกำหนดค่า Duty Cycle เป็น 100% $V_{average}$ จะเท่ากับ V_{Supply} ซึ่งจะทำให้มอเตอร์หมุนได้เร็วที่สุด แต่ถ้าหากลดค่า Duty Cycle ลงมาเหลือ 50% $V_{average}$ จะมีค่าประมาณครึ่งหนึ่งของ V_{Supply} ความเร็วก็จะลดลงเป็นครึ่งหนึ่งด้วย แต่ในความเป็นจริงมอเตอร์จะไม่สามารถหมุนได้ตามกำลังอินพุตที่ใส่เข้าไป

จากสมการแรงดันไฟฟ้าของมอเตอร์

$$V=Eg+I_aR_a$$

ใช้กระแสอาร์เมเจอร์I_a คูณเข้าไปทั้งสองข้างของสมการ

$$VI_a=EgI_a+I_a^2R_a$$

จะได้

VI_a = กำลังอินพุตที่จ่ายให้กับมอเตอร์ (Watt)

EgI_a = กำลังไฟฟ้าส่วนที่เปลี่ยนรูปเป็นกำลังกลในอาร์เมเจอร์ (Watt)

$I_a^2R_a$ = การสูญเสียในขดลวดอาร์เมเจอร์ (Watt)

จากสมการแสดงให้เห็นว่าจะต้องมีกำลังสูญเสียในตัวมอเตอร์ เพราะฉะนั้นจะต้องมีการชดเชยค่า IR Compensate ด้วยการใช้ Current Sensor วัดกระแสจากมอเตอร์ แล้วจึงทำการส่งไปยังโมดูล A / D ต่อจากนั้น dsPIC จะทำการชดเชยให้แก่มอเตอร์ด้วยเทคนิคทางซอฟท์แวร์ ซึ่งการทำเช่นนี้จะทำให้กำลังเอาท์พุตของมอเตอร์มีความใกล้เคียงกับกำลังอินพุตที่ใส่เข้าไป

5.2 การทดลองหาค่าพารามิเตอร์ของมอเตอร์

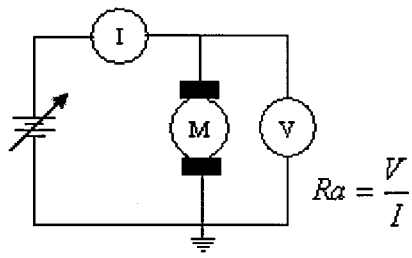
จุดมุ่งหมายในการทดสอบค่าพารามิเตอร์ต่างๆของมอเตอร์นั้น เพื่อให้ทราบข้อมูลที่จำเป็นเพื่อพิจารณาถึงเสถียรภาพการทำงานของระบบสำหรับนำไปประยุกต์การใช้งานตามที่ต้องการ

วิธีการทดสอบที่ใช้ในที่นี้ ซึ่งจะเป็นวิธีการทดสอบที่ได้พัฒนาขึ้นมาให้เหมาะสมกับผู้ที่มีเครื่องมือสำหรับทดสอบที่จำกัด คือเราได้เลือกเอาเฉพาะวิธีที่ง่ายที่สุดในการทดสอบหรือวิธีการทดสอบที่ไม่ต้องใช้เครื่องมือที่ยุ่งยาก

เนื่องจากการใช้มอเตอร์ในลักษณะของงานบางอย่าง จะต้องคำนึงถึงขีดจำกัดการทำงานของเอาท์พุทซึ่งอาจเกิดขึ้นได้ เช่น ค่าของความร้อน เราจะต้องตัดสินใจในขั้นแรกว่าพารามิเตอร์อะไรสัมพันธ์กับงานที่ใช้อยู่

5.2.1 การหาค่าความต้านทานอาร์เมเจอร์ของมอเตอร์ (Ra)

การวัดค่าความต้านทานของอาร์มาเจอร์ที่สะดวกที่สุดคือ การเปลี่ยนจากการใช้เครื่องวัดโอมห์มิเตอร์ มาใช้วิธีการวัดด้วย โวลท์-แอมป์มิเตอร์



รูปที่ 5.5 การวัดความต้านทานของอาร์เมเจอร์ด้วยการทดสอบกระแส

เมื่อเราทำการล็อคอาร์เมเจอร์ของมอเตอร์ให้หยุดนิ่งอยู่กับที่ตำแหน่งหนึ่ง จากนั้นก็จ่ายโวลท์เตจตกคร่อมตัวมอเตอร์แล้วทำการวัดค่ากระแสที่ไหลผ่านมอเตอร์ ซึ่งสามารถคำนวณค่าความต้านทานจากสูตร $Ra = \frac{V}{I}$

เราต้องทำการวัดค่ากระแสหลายๆ ค่าแล้วจึงหาค่าเฉลี่ย จะได้เป็นค่า Ra ของมอเตอร์ตัวนั้นๆ

มอเตอร์กระแสตรง ทำการจ่าย V_{in} (V_{supply}) ให้แก่มอเตอร์ จากนั้นเบรกมอเตอร์ให้หยุดนิ่งอยู่กับที่ ค่า Eg = 0 จะเกิดค่ากระแส I_a (A) ขึ้น ทำการบันทึกค่า Ia จากนั้นเปลี่ยนการจ่าย V_{in} และทดลองซ้ำและคำนวณค่า R_a

ตารางที่ 5.3 ผลการวัดค่าต่างๆของมอเตอร์กระแสตรง

V_{in}(V)	I_a(A)	$R_a(\Omega)$
3.00	0.28	10.714
4.00	0.39	10.256
5.00	0.48	10.416
6.00	0.59	10.169
7.00	0.69	10.145

จากทฤษฎี Linear Least Square ซึ่งเป็นสมการประมาณค่าเส้นตรง

$$\sum xy = A\sum x^2 + B\sum x \qquad (1)$$

$$\sum y = A\sum x + NB \qquad (2)$$

เมื่อกำหนดให้ X คือ I_a

Y คือ V_{in}

จะได้ $\sum XY$ = 13.17

$\sum X^2$ = 1.2851

$\sum Y$ = 25

$\sum X$ = 2.43

n = 5

แทนค่าลงในสมการที่ (1) และ (2)

$$13.71 = 1.2851A + 2.43B$$

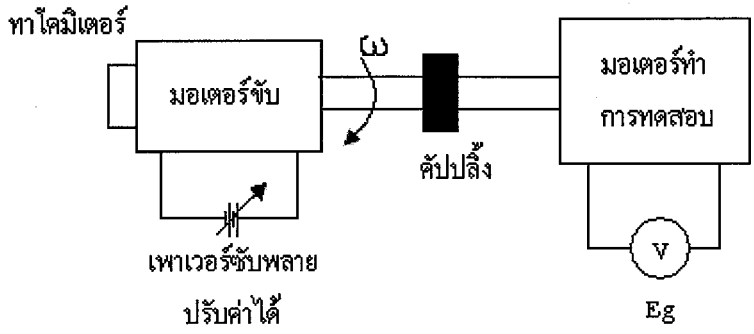
$$25 = 2.43A + 5B$$

แก้สมการจะได้ค่า A = 14.9827

∴ Ra ของมอเตอร์กระแสตรงมีค่า = 14.9827 Ω

5.2.2 การวัดค่าโวลท์เตจคงที่ (ค่า back-emf constant ของมอเตอร์)

การวัดค่า Back-emf constant (K_e) เรากระทำได้โดยขับมอเตอร์ที่ต้องการทดสอบให้หมุนเหมือนเป็นเจนเนอร์เรเตอร์ด้วยมอเตอร์อีกตัวหนึ่ง และวัดค่าโวลท์เตจที่เกิดขึ้นในตัวมอเตอร์ที่ทดสอบ (E_g) ขณะเดียวกันก็ทำการวัดความเร็วรอบของแกนมอเตอร์ ω ด้วย ตัวอย่างของวงจรที่ใช้ในการทดสอบแสดงดังรูปด้านล่าง



รูปที่ 5.6 วิธีการวัดค่าแบ็ค-อีเอ็มเอฟคอนสแต้นซ์ของมอเตอร์

เราหาค่าแบ็ค-อีเอ็มเอฟคอนสแต้นซ์ของมอเตอร์ได้จากความสัมพันธ์ต่อไปนี้

$$K_e = \frac{E_g}{\omega}$$

เราสามารถหาความสัมพันธ์ของ K_t และ K_e ได้จากสมการ

$$K_t = K_e[Nm/A; V/rads^{-1}]$$

- การทดลองหาค่า Ke ของมอเตอร์

มอเตอร์กระแสตรง

ทำการป้อนแรงดันให้แก่มอเตอร์ จากนั้นวัดค่าแรงดันย้อนกลับ (E_g) ขณะเดียวกันก็ทำการวัดความเร็วรอบของมอเตอร์ด้วยทาโคมิเตอร์ซึ่งมีหน่วยเป็น rpm (รอบต่อนาที) ดังนั้น

จะต้องเปลี่ยนให้เป็นหน่วย (rad/sec) จากสูตร $\omega(rpm)\times\frac{2\pi}{60} = \omega(rad/\sec)$ สามารถคำนวณ Ke ได้จากสมการ $K_e = \frac{E_g}{\omega}$ จากนั้นทำการเปลี่ยนค่า V_{Supply} และทดลองซ้ำ

ตารางที่ 5.4 การวัดค่าต่างๆของมอเตอร์กระแสตรง

V_{Supply}	Eg(V)	ω (rpm)	ω (rad/sec)	$K_e(V/rad/\sec)$
10	3.49	850.5	89.064	0.0392
11	3.95	937.6	98.185	0.0402
12	4.53	1045	109.432	0.0413
13	5.18	1135	118.856	0.0435
14	5.73	1234	129.224	0.0443
15	6.29	1330	139.277	0.0451

$$\sum xy = A\sum x^2 + B\sum x \qquad (1)$$
$$\sum y = A\sum x + NB \qquad (2)$$

เมื่อกำหนดให้ X = ω

Y = E_g

จะได้ $\sum XY$ = 3426.568

$\sum X^2$ = 79771.724

$\sum Y$ = 29.71

$\sum X$ = 684.038

n = 6

แทนค่าลงในสมการที่ (1) และ (2)

3426.568 = 79771.724A + 684.038B (1)

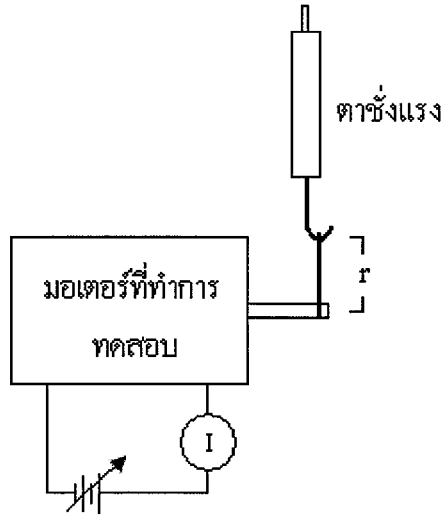
29.71 = 684.038A + 6B (2)

แก้สมการจะได้ค่า A = 0.02206

∴ ค่า K_e ของมอเตอร์กระแสตรงมีค่าเท่ากับ 0.02206 $\frac{volts}{rad/\sec}$

5.2.3 การวัดค่าทอร์กคงที่ (torque constant; **Kt**)

เราทำการป้อนโวลท์เตจค่าหนึ่ง เพื่อให้มอเตอร์หมุนด้วยความเร็วคงที่ทำการวัดกระแสอาร์เมเจอร์ (Ia) ขณะนั้น และที่ปลายของมอเตอร์จะทำการผูกกับตาชั่งแรง เมื่อมอเตอร์หมุนจะทำให้เกิดแรงดึงตาชั่งอ่านค่าแรงดึงขณะนั้นซึ่งมีหน่วยเป็น mg โดยทำการเปลี่ยนค่าแรงดึงให้เป็นหน่วยนิวตันทอร์กของมอเตอร์จะได้จากสูตร $\tau = F \times r$ ซึ่ง r คือระยะทางจากปลายมอเตอร์ไปยังตาชั่ง (หน่วย m)



รูปที่ 5.7 การวัดค่าทอร์กคงที่ของมอเตอร์

เราหาค่าทอร์กคงที่ของมอเตอร์ได้จากความสัมพันธ์ต่อไปนี้

$$K_e = \frac{\tau}{I_a}$$

มอเตอร์กระแสตรง

ทำการป้อน V_{supply} แล้ววัดค่า Ia และแรงดึงของมอเตอร์จากตาชั่ง แต่เนื่องจากแรงดึงของตาชั่งมีหน่วยเป็น mg จึงทำการเปลี่ยนหน่วยให้เป็นนิวตัน

จากสูตร F(g) × 9.81 = F(N)

สามารถคำนวณค่าทอร์กได้จาก $\tau = F \times r$; เมื่อ r มีระยะทางเท่ากับ 0.147 เมตร

ตารางที่ 5.5 ผลการวัดค่าต่างๆของมอเตอร์กระแสตรง เมื่อ r = 0.147 เมตร

V_{supply}	F(g)	F(N)	Ia	τ	Kt
3	153.33	1.504	0.206	0.221	1.07
4	273.33	2.681	0.323	0.394	1.21
5	433.33	4.250	0.420	0.624	1.48
6	576.66	5.657	0.526	0.831	1.57
7	693.33	6.801	0.670	0.999	1.49

$$\sum xy = A\sum x^2 + B\sum x \qquad (1)$$
$$\sum y = A\sum x + NB \qquad (2)$$

เมื่อ $X=I_a$

$Y=\tau$

จะได้ $\sum XY = 1.5412$

$\sum X^2 = 1.0487$

$\sum Y = 3.069$

$\sum X = 2.145$

n = 5

แทนค่าลงในสมการที่ (1) และ (2)

1.5412 = 1.0487A + 2.145B (1)

3.069 = 2.145A + 5B (2)

แก้สมการจะได้ค่า A = 1.36538

∴ ค่า K_t ของมอเตอร์กระแสตรงมีค่าเท่ากับ 1.36538 Newton-meter/ampere

- การทดลองความสัมพันธ์ระหว่าง ความเร็วและแรงดันไฟฟ้า

F pwm = 56 Hz.

T pwm = 17.6 ms

Supply ที่จ่ายแก่มอเตอร์ คือ 18.66 v แต่ต้องการปรับย่านแรงดันให้น้อยลงเพื่อให้ได้ความเร็วที่อยู่ในช่วงที่ใช้งาน โดยใช้สมการ B = A(3/5) + 6 เมื่อ A คือ speed(%) B คือ Voltage (%) ของ 18.66 v ที่สั่ง เช่น สั่งที่ speed = 100% แทนในสมการจะได้ 100(3/5) + 6 = 66% ของ 18.66 v ได้ประมาณ 12.32 v ดังนั้น ย่านแรงดันที่ใช้งานคือ 0 ถึง 12.32 v

ตารางที่ 5.6 แสดงค่าจากการคำนวณ

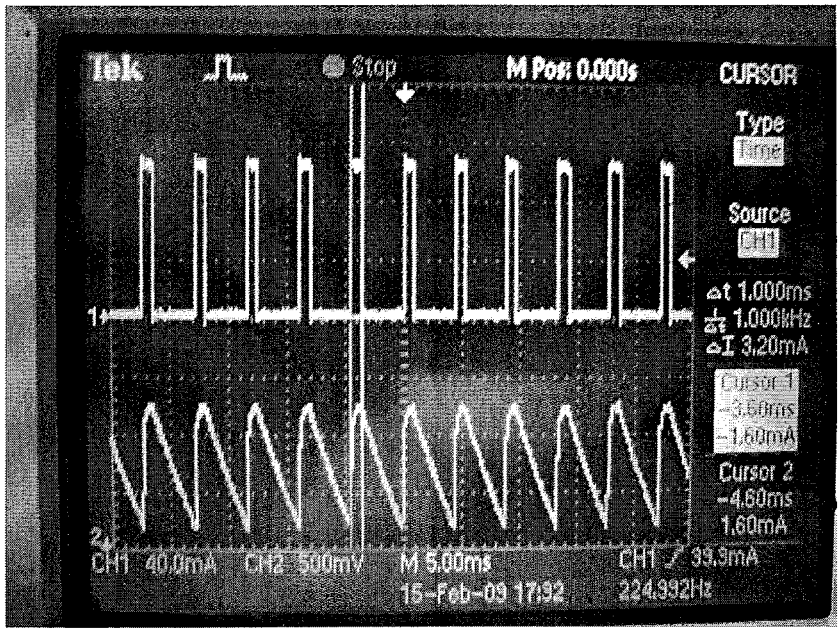
speed	R	Vout(v)	Duty cycle (%) of 18.66v
0	0	0	0
10	0.5	2.2	12
20	1.0	3.4	18
30	1.5	4.5	24
40	2.0	5.6	30
50	2.5	6.7	36
60	3.0	7.8	42
70	3.5	9.0	48
80	4.0	10.1	54
90	4.5	11.2	60
100	5.0	12.3	66

ตารางที่ 5.7 แสดงค่าที่ได้จากการทดลองจริง

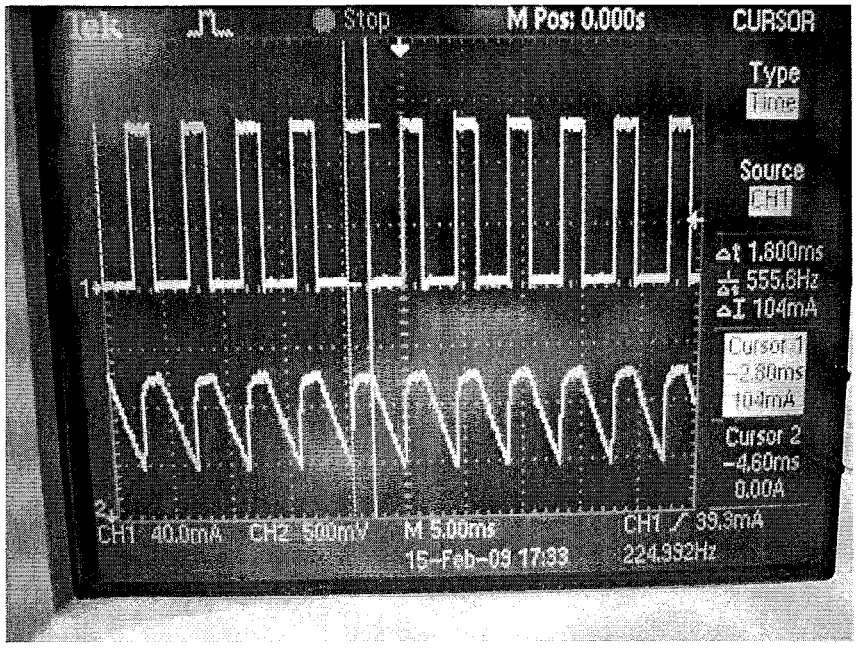
speed	R	Vout(v)	t_{on}(ms)	Duty cycle (%) of 18.66v
0	0	0	0	0
10	0.5	1.96	1.6	9.09
20	1.0	3.0	2.8	15.91
30	1.5	4.2	4.0	22.73
40	2.0	5.3	5.2	29.55
50	2.5	6.3	6.0	30.09
60	3.0	7.2	7.2	40.90
70	3.5	8.2	8.4	47.73
80	4.0	9.2	9.2	52.27
90	4.5	10.2	10.4	59.09
100	4.94	11	11.2	63.64

5.2.4 การทดลองวัดรูปสัญญาณ PWM

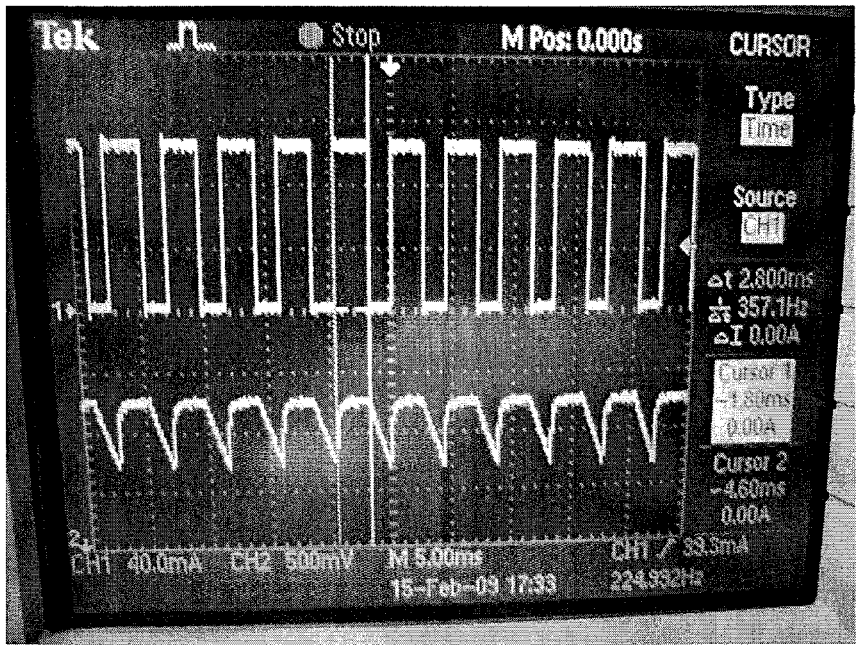
1. สัญญาณที่ Duty Cycle ที่ 20%



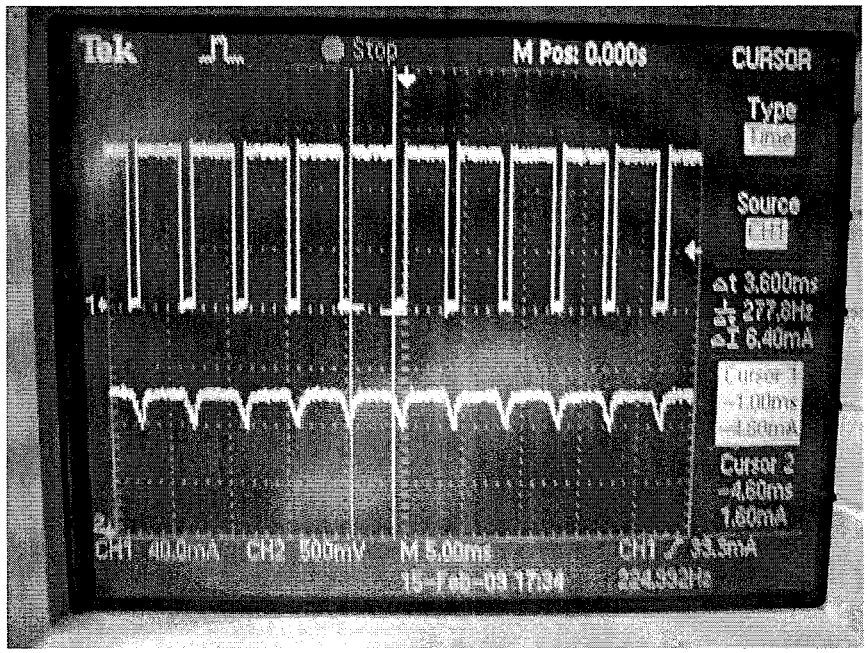
2. สัญญาณที่ Duty Cycle ที่ 40%



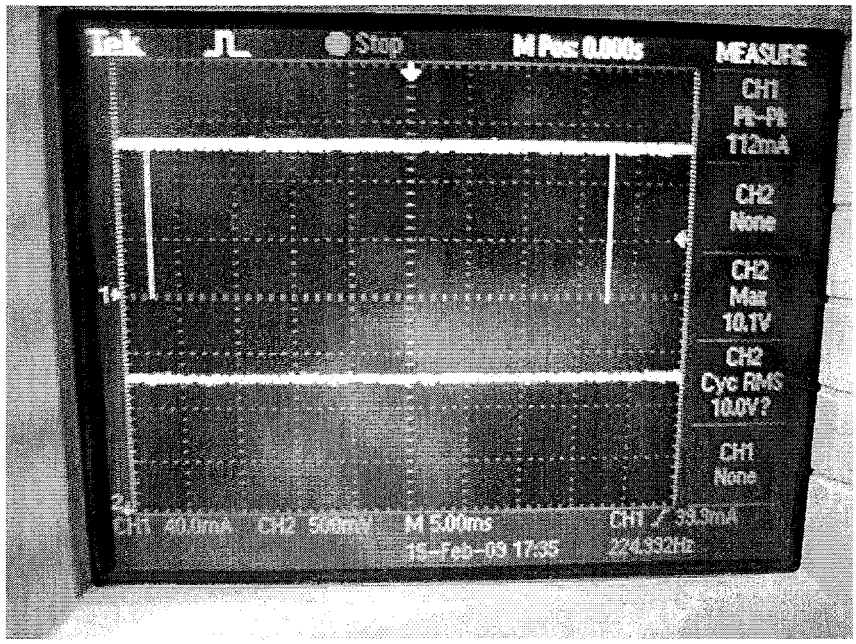
3. สัญญาณที่ Duty Cycle ที่ 60%



4. สัญญาณที่ Duty Cycle ที่ 80%



5. สัญญาณที่ Duty Cycle ที่ 100%



หมายเหตุ

-สัญญาณด้านบน คือ สัญญาณ PWM ที่ออกมาจาก dsPIC

-สัญญาณด้านล่าง คือ สัญญาณที่ออกจากวงจร Drive

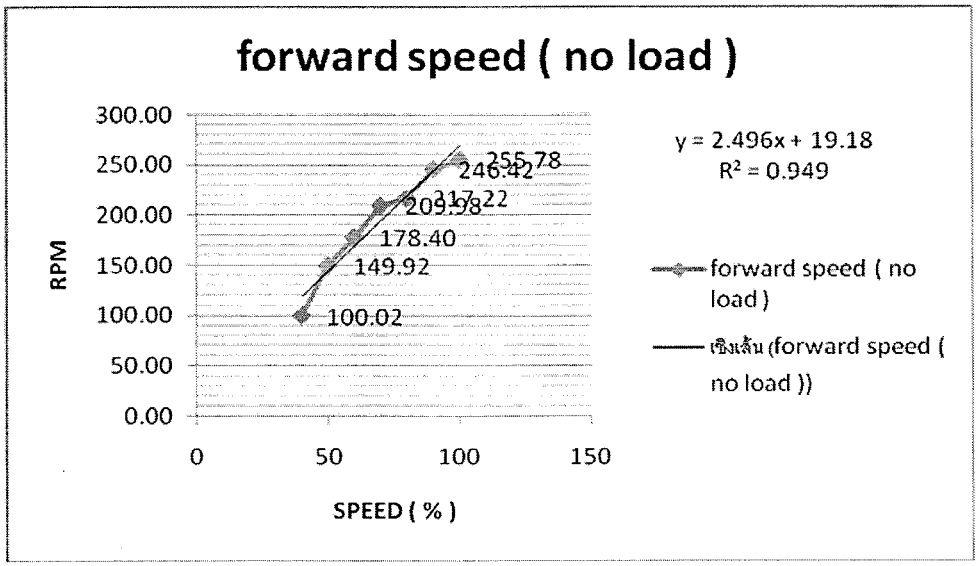
5.2.5 การทดลองวัดและทดสอบปรับความเร็ว

การสั่งงานปรับค่า SPEED ผ่านโปรแกรม

ตารางที่ 5.8 แสดงค่าระหว่างการปรับค่าความเร็ว (%) และความเร็วรอบ

FORWARD (NO LOAD)

speed	ความเร็วรอบ (RPM)					STD	AVERAGE
%	1	2	3	4	5	ค่าเบี่ยงเบนมาตราฐาน	ค่าเฉลี่ย
40	93.90	106.40	108.20	101.70	89.90	7.075	100.02
50	153.30	149.90	142.50	151.90	152.00	3.866	149.92
60	182.90	183.00	181.20	172.50	172.40	4.900	178.40
70	211.80	205.80	209.50	210.90	211.90	2.260	209.98
80	217.10	216.00	216.80	219.50	216.70	1.196	217.22
90	235.70	251.50	254.90	255.20	234.80	9.217	246.42
100	260.30	253.20	254.40	255.00	256.00	2.435	255.78

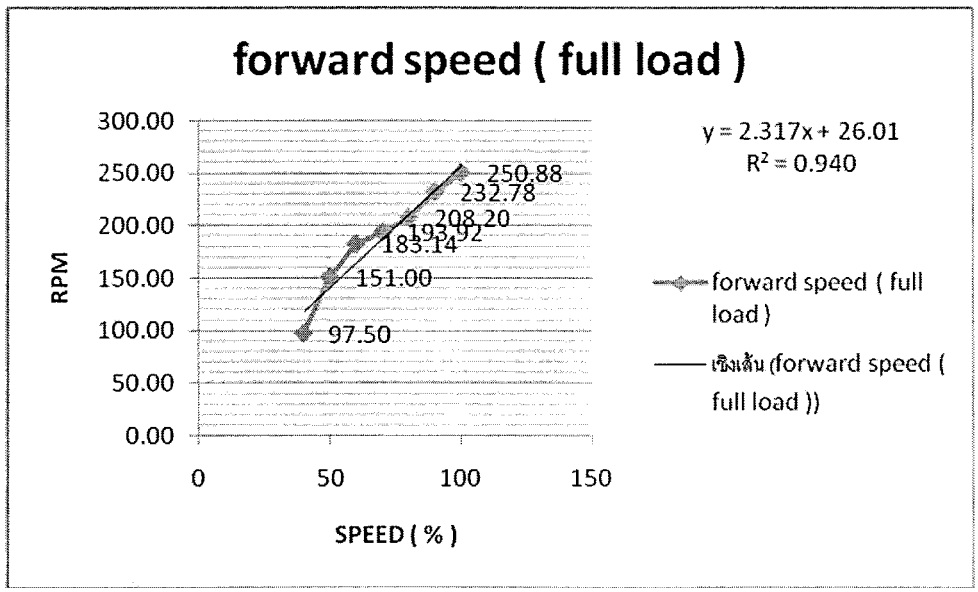


รูปที่ 5.8 แสดงกราฟที่ Forward speed (no load) จากทางโปรแกรม

ตารางที่ 5.9 แสดงค่าระหว่างการปรับค่าความเร็ว (%) และความเร็วรอบ

FORWARD (FULL LOAD)

speed	ความเร็วรอบ (RPM)					STD	AVERAGE
%	1	2	3	4	5	ค่าเบี่ยงเบนมาตราฐาน	ค่าเฉลี่ย
40	113.20	90.00	92.70	96.80	94.80	8.168	97.50
50	155.10	152.40	152.50	144.90	150.10	3.436	151.00
60	182.20	188.70	186.50	188.10	170.20	6.858	183.14
70	190.10	190.30	193.40	203.50	192.30	4.947	193.92
80	204.60	210.10	203.80	210.20	212.30	3.369	208.20
90	236.20	236.20	236.00	237.50	218.00	7.409	232.78
100	253.70	244.40	249.70	252.90	253.70	3.560	250.88



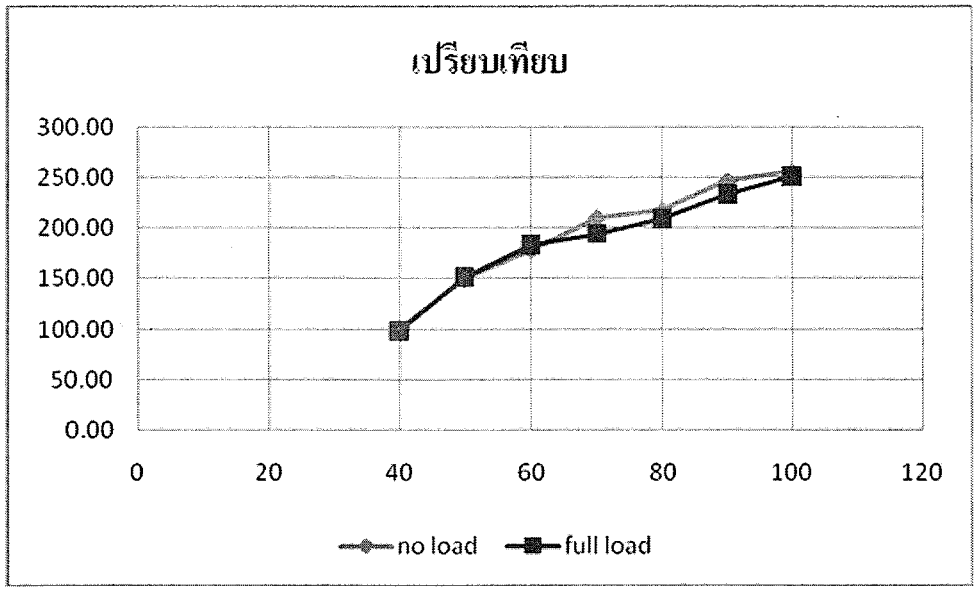
รูปที่ 5.9 แสดงกราฟที่ Forward speed (full load) จากทางโปรแกรม

ตารางที่ 5.10 แสดงค่าความผิดพลาดที่เกิดขึ้นจากการปรับความเร็วในทิศทาง forward จากทางซอฟต์แวร์

SPEED %	FORWARD no load	FORWARD full load	ERROR %
40	100.02	97.50	0.03
50	149.92	151.00	-0.01
60	178.4	183.14	-0.05
70	209.98	193.92	0.16
80	217.22	208.20	0.09
90	246.42	232.78	0.14
100	255.78	250.88	0.05

ตารางที่ 5.11 แสดงค่าความผิดพลาดที่เกิดขึ้นจากการปรับความเร็วในทิศทาง forward (cm/s) จากทางซอฟต์แวร์

SPEED %	RPM F.no load	SPEED cm/s	RPM F.full load	SPEED cm/s	ERROR %
40	100.02	3.33	97.50	3.25	0.0008
50	149.92	5.00	151.00	5.03	-0.0004
60	178.40	5.95	183.14	6.10	-0.0016
70	209.98	7.00	193.92	6.46	0.0054
80	217.22	7.24	208.20	6.94	0.0030
90	246.42	8.21	232.78	7.76	0.0045
100	255.78	8.53	250.88	8.36	0.0016

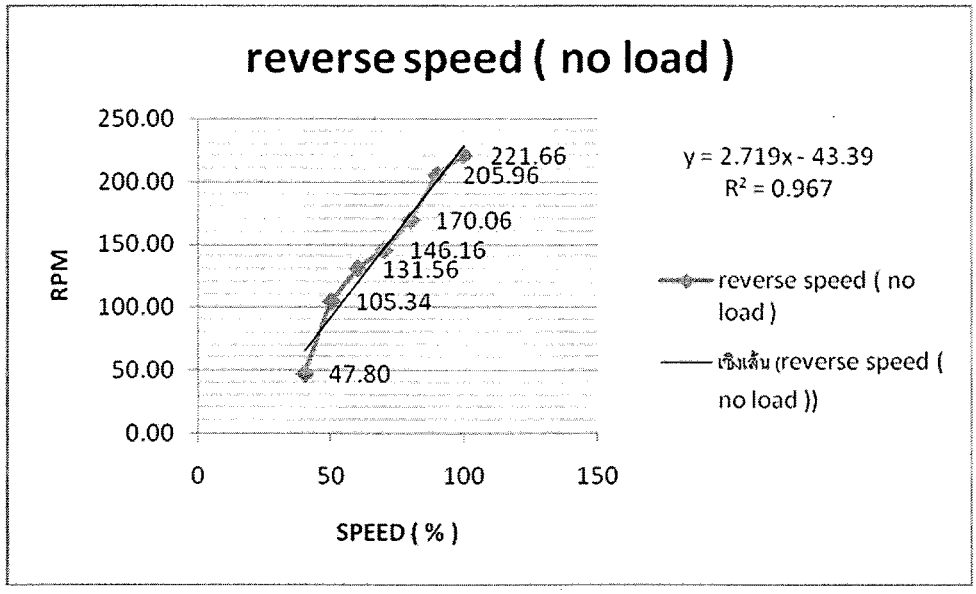


รูปที่ 5.10 เปรียบเทียบ Forward ที่ no load และ full load จากการสั่งงานผ่านทางโปรแกรม

ตารางที่ 5.12 แสดงค่าระหว่างการปรับค่าความเร็ว (%) และความเร็วรอบ

REVERSE (NO LOAD)

speed	ความเร็วรอบ (RPM)					STD	AVERAGE
%	1	2	3	4	5	ค่าเบี่ยงเบนมาตราฐาน	ค่าเฉลี่ย
40	56.90	45.40	45.40	45.40	45.90	4.554	47.80
50	104.20	107.90	107.40	106.20	101.00	2.517	105.34
60	131.90	123.40	133.60	135.90	133.00	4.284	131.56
70	143.10	143.80	151.30	146.80	145.80	2.895	146.16
80	174.80	168.90	164.40	171.80	170.40	3.435	170.06
90	212.20	196.00	209.60	207.10	204.90	5.546	205.96
100	219.00	216.00	221.10	223.70	228.50	4.251	221.66

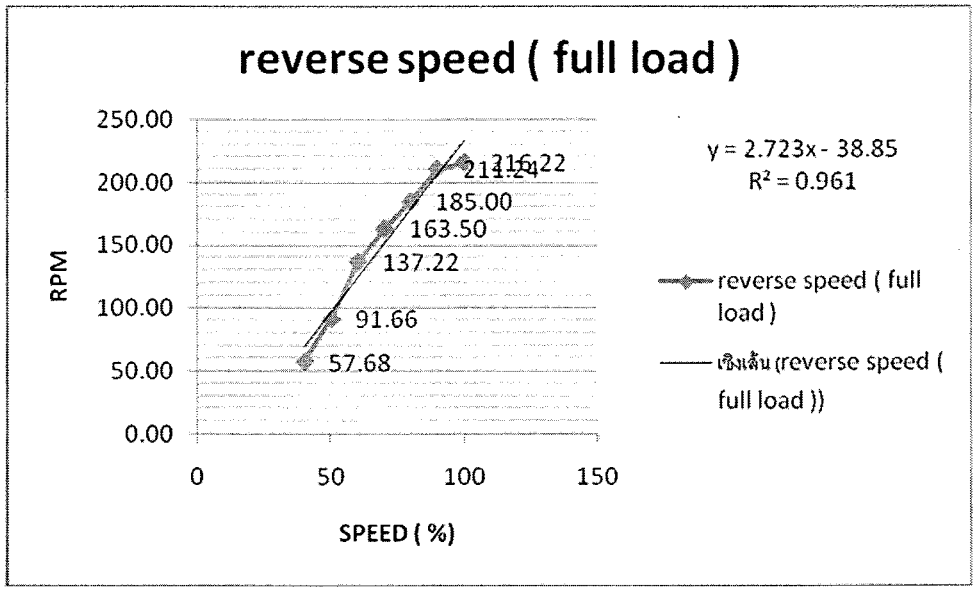


รูปที่ 5.11 แสดงกราฟที่ Reverse speed (no load) จากทางโปรแกรม

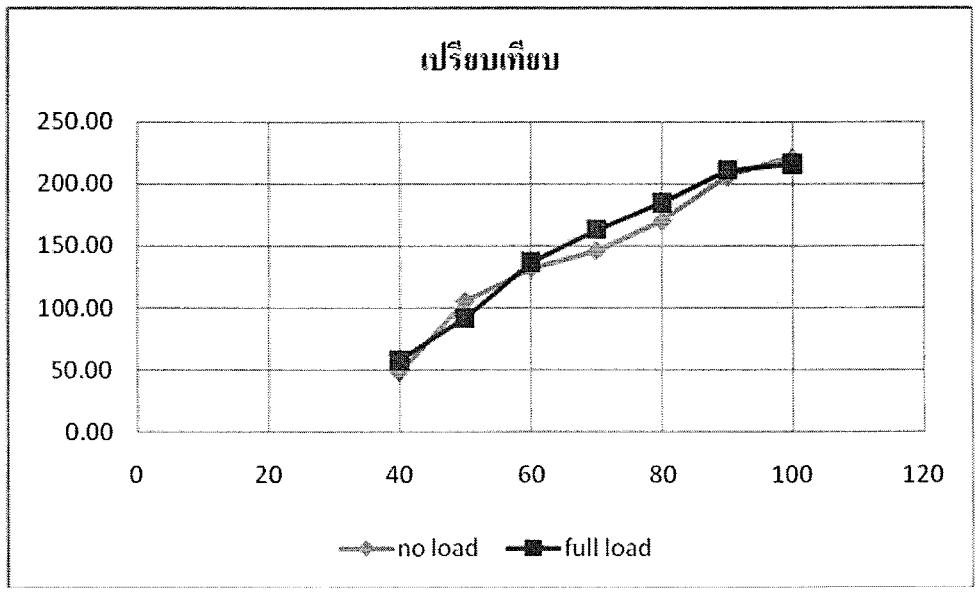
ตารางที่ 5.13 แสดงค่าระหว่างการปรับค่าความเร็ว (%) และความเร็วรอบ

REVERSE (FULL LOAD)

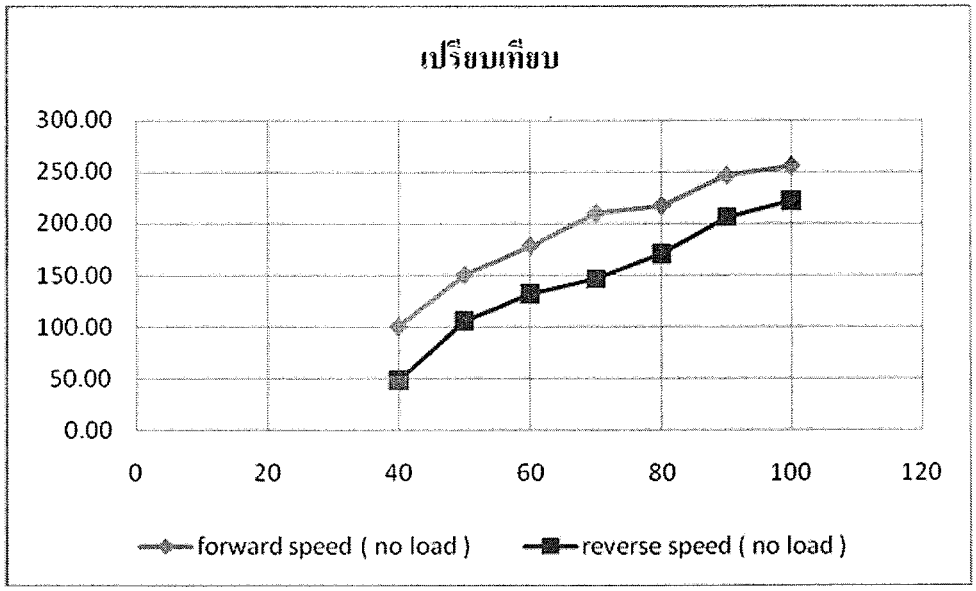
speed	ความเร็วรอบ (RPM)					STD	AVERAGE
%	1	2	3	4	5	ค่าเบี่ยงเบนมาตราฐาน	ค่าเฉลี่ย
40	59.90	57.50	53.50	56.30	61.20	2.710	57.68
50	97.20	87.50	83.30	92.60	97.70	5.572	91.66
60	126.20	137.80	138.90	144.00	139.20	5.907	137.22
70	162.60	165.30	162.60	165.40	161.60	1.554	163.50
80	186.50	177.80	187.10	186.90	186.70	3.606	185.00
90	213.10	207.10	211.40	210.90	213.70	2.315	211.24
100	211.20	217.20	215.00	219.70	218.00	2.930	216.22



รูปที่ 5.12 แสดงกราฟที่ Reverse speed (full load) จากทางโปรแกรม



รูปที่ 5.13 เปรียบเทียบ Reverse ที่ no load และ full load จากการสั่งงานผ่านทางโปรแกรม



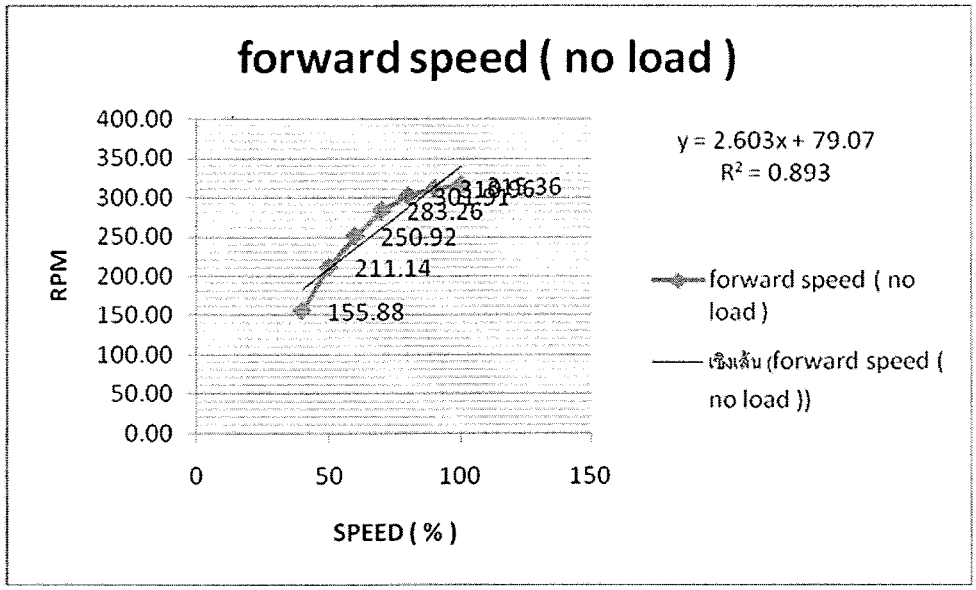
รูปที่ 5.14 เปรียบเทียบ Forward และ Reverse ที่ no load จากการสั่งงานผ่านทางโปรแกรม

การสั่งงานปรับค่า SPEED ผ่านปุ่มกดจากค่าความต้านทานปรับค่าได้

ตารางที่ 5.14 แสดงค่าระหว่างการปรับค่าความเร็ว (%) และความเร็วรอบ

FORWARD (NO LOAD)

SPEED	VOTAGE	ความเร็วรอบ (RPM)					STD	AVERAGE
%		1	2	3	4	5	ค่าเบี่ยงเบนมาตราฐาน	ค่าเฉลี่ย
40	1.63	157.1	154.9	155.1	155.5	156.8	0.90	155.88
50	2.14	220.9	205.6	199.5	214.8	214.9	7.60	211.14
60	2.69	244.6	243.2	257.4	252.4	257.0	6.01	250.92
70	3.2	282.2	282.6	284.7	287.7	279.1	2.85	283.26
80	3.65	302.7	308.7	303.2	289.6	306.0	6.46	301.91
90	4.25	311.2	310.9	315.9	310.1	306.7	2.94	310.96
100	4.99	311.7	317.5	321.1	314.0	312.5	3.49	315.36

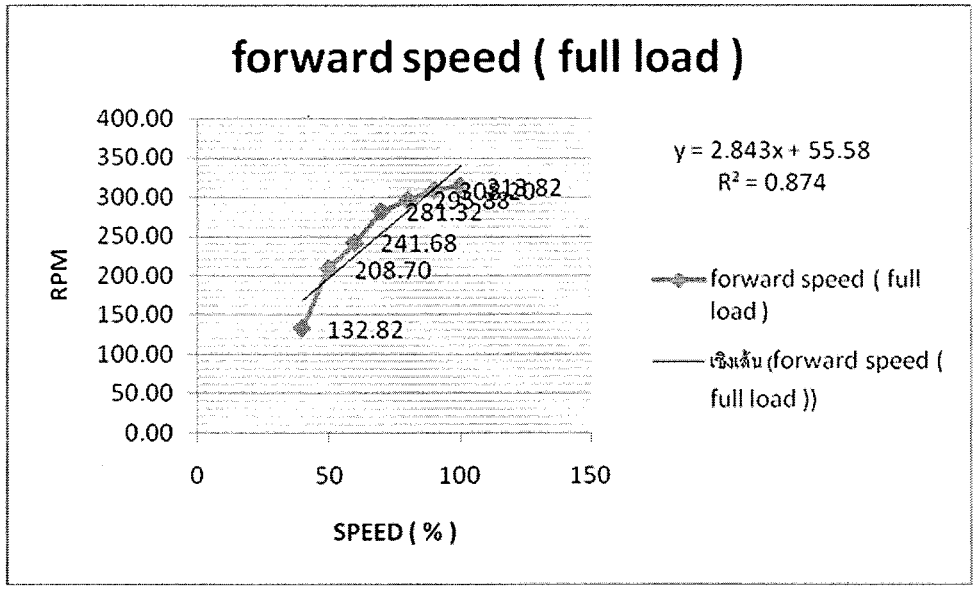


รูปที่ 5.15 แสดงกราฟที่ Forward speed (no load) จากทางค่าความต้านทานปรับค่าได้

ตารางที่ 5.15 แสดงค่าระหว่างการปรับค่าความเร็ว (%) และความเร็วรอบ

FORWARD (FULL LOAD)

SPEED %	VOTAGE	ความเร็วรอบ (RPM)					STD ค่าเบี่ยงเบนมาตราฐาน	AVERAGE ค่าเฉลี่ย
		1	2	3	4	5		
40	1.63	133.2	140.3	136.9	121.5	132.2	6.34	132.82
50	2.14	214.2	206.6	208.9	202.9	210.9	3.83	208.70
60	2.69	243.8	239.9	244.2	239.9	240.6	1.92	241.68
70	3.2	273.6	287.7	279.1	284.9	281.3	4.86	281.32
80	3.65	294.6	300.1	291.9	292.5	300.3	3.64	295.88
90	4.25	309.8	307	310.6	308.1	305.5	1.85	308.20
100	4.99	310.6	311.5	313.3	318.5	315.2	2.82	313.82



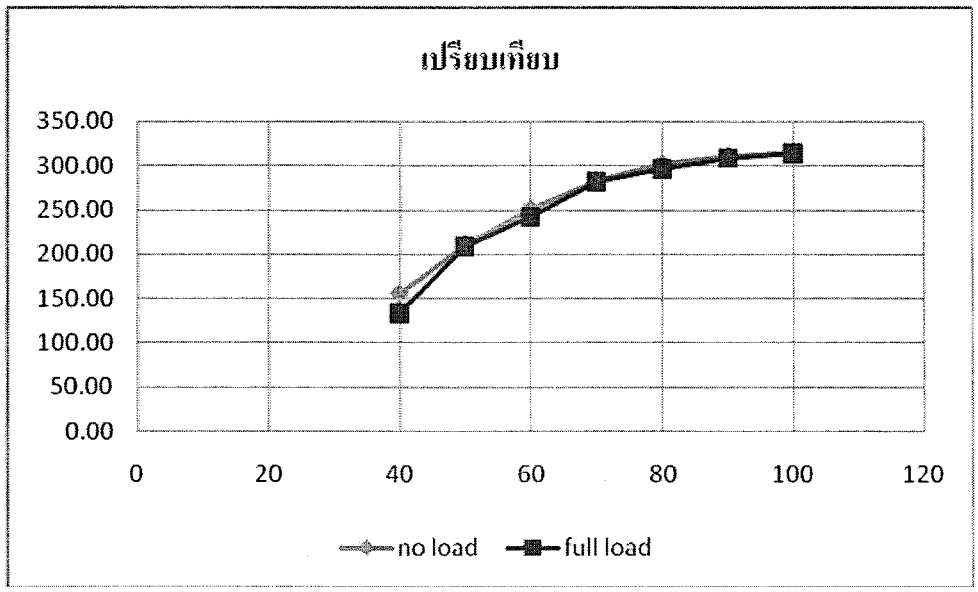
รูปที่ 5.16 แสดงกราฟที่ forward speed (full load) จากทางค่าความต้านทานปรับค่าได้

ตารางที่ 5.16 แสดงค่าความผิดพลาดที่เกิดขึ้นจากการปรับความเร็วในทิศทาง forward จากทางค่าความต้านทานปรับค่าได้

SPEED %	FORWARD no load	FORWARD full load	ERROR %
40	155.88	132.82	0.23
50	211.14	208.70	0.02
60	250.92	241.68	0.09
70	283.26	281.32	0.02
80	301.914	295.88	0.06
90	310.96	308.20	0.03
100	315.36	313.82	0.02

ตารางที่ 5.17 แสดงค่าความผิดพลาดที่เกิดขึ้นจากการปรับความเร็วในทิศทาง forward (cm/s) จากทางค่าความต้านทานปรับค่าได้

SPEED %	RPM F.no load	SPEED cm/s	RPM F.full load	SPEED cm/s	ERROR %
40	155.88	5.20	132.82	4.43	0.008
50	211.14	7.04	208.70	6.96	0.001
60	250.92	8.36	241.68	8.06	0.003
70	283.26	9.44	281.32	9.38	0.001
80	301.914	10.06	295.88	9.86	0.002
90	310.96	10.37	308.20	10.27	0.001
100	315.36	10.51	313.82	10.46	0.001

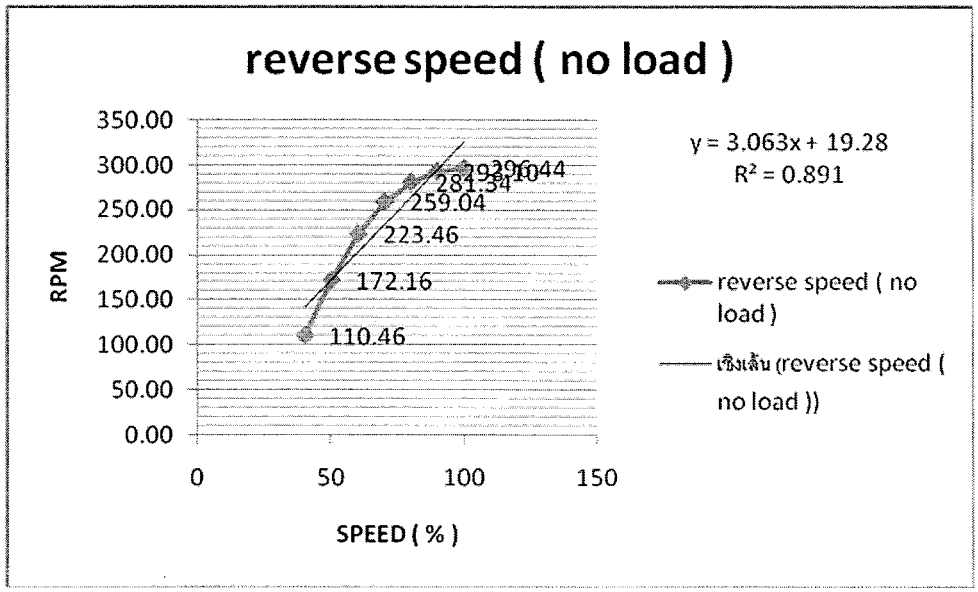


รูปที่ 5.17 เปรียบเทียบ Forward ที่ no load และ full load จากการสั่งงานจากค่าความต้านทานปรับค่าได้

ตารางที่ 5.18 แสดงค่าระหว่างการปรับค่าความเร็ว (%) และความเร็วรอบ

REVERSE (NO LOAD)

SPEED %	VOTAGE	ความเร็วรอบ (RPM)					STD ค่าเบี่ยงเบนมาตราฐาน	AVERAGE ค่าเฉลี่ย
		1	2	3	4	5		
40	1.63	113	102.2	109.7	116.6	110.8	4.76	110.46
50	2.14	169.1	181.4	170.8	174.5	165	5.54	172.16
60	2.69	222.1	227.1	222.8	222.1	223.2	1.87	223.46
70	3.2	261.6	260.7	258.6	258.4	255.9	1.99	259.04
80	3.65	276	282.8	288.2	279.6	280.1	4.06	281.34
90	4.25	294.1	285.5	297.1	297.5	291.3	4.41	293.10
100	4.99	294	293.7	299.6	295.8	299.1	2.49	296.44

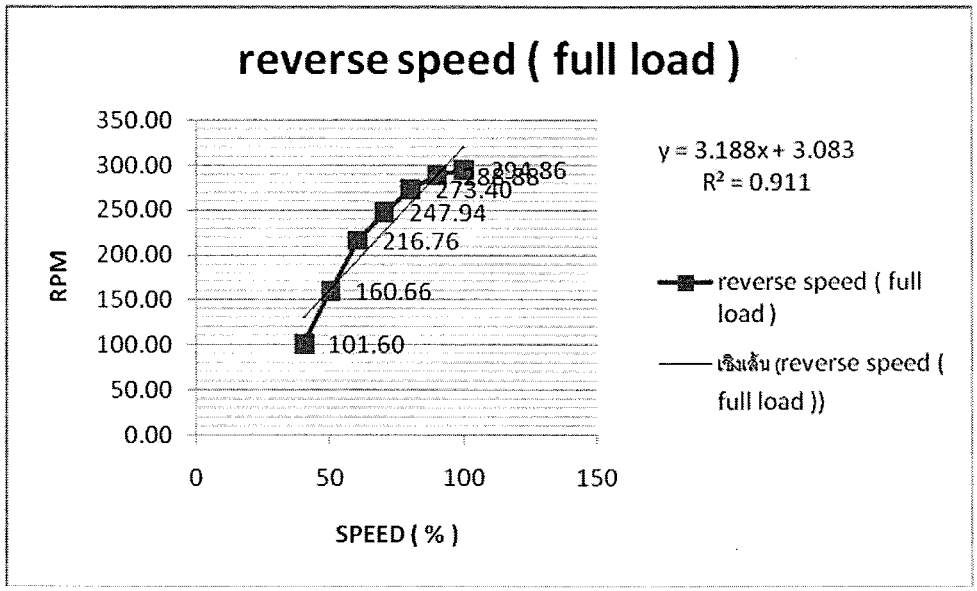


รูปที่ 5.18 แสดงกราฟที่ Reverse speed (no load) จากทางค่าความต้านทานปรับค่าได้

ตารางที่ 5.19 แสดงค่าระหว่างการปรับค่าความเร็ว (%) และความเร็วรอบ

REVERSE (FULL LOAD)

SPEED %	VOTAGE	ความเร็วรอบ (RPM)					STD ค่าเบี่ยงเบนมาตราฐาน	AVERAGE ค่าเฉลี่ย
		1	2	3	4	5		
40	1.63	105	99.7	96	105.8	101.5	3.58	101.60
50	2.14	146.8	168.3	153.3	168.7	166.2	8.94	160.66
60	2.69	217.4	214.5	216.2	220.4	215.3	2.06	216.76
70	3.2	259.3	248	244.8	245	242.6	5.93	247.94
80	3.65	280.4	265.5	267.9	275.5	277.7	5.74	273.40
90	4.25	293.2	289.9	283.1	288	290.2	3.34	288.88
100	4.99	294.3	299.1	297.5	299.01	284.4	5.51	294.86



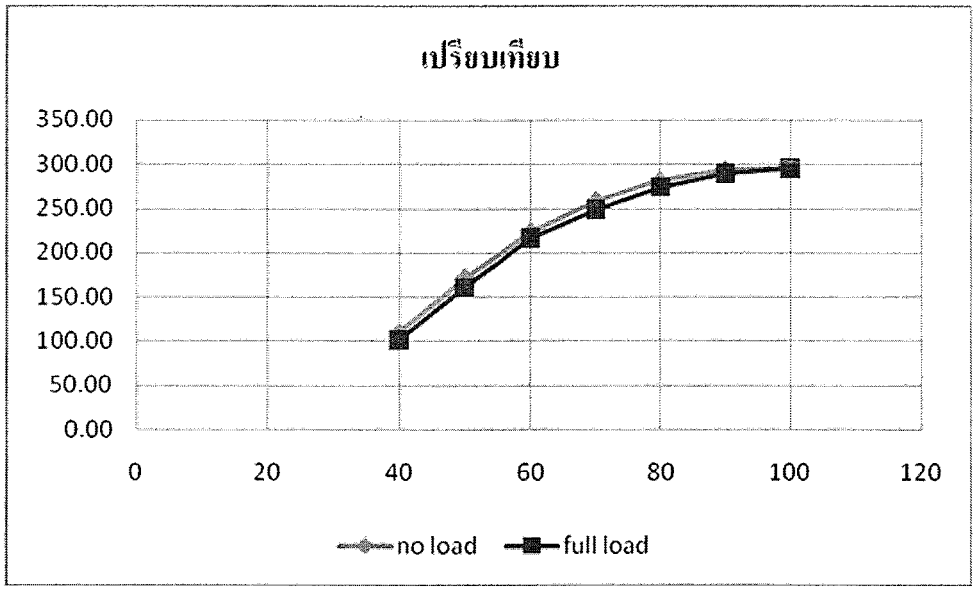
รูปที่ 5.19 แสดงกราฟที่ Reverse speed (full load) จากทางค่าความต้านทานปรับค่าได้

ตารางที่ 5.20 แสดงค่าความผิดพลาดที่เกิดขึ้นจากการปรับความเร็วในทิศทาง reverse จากทางค่าความต้านทานปรับค่าได้

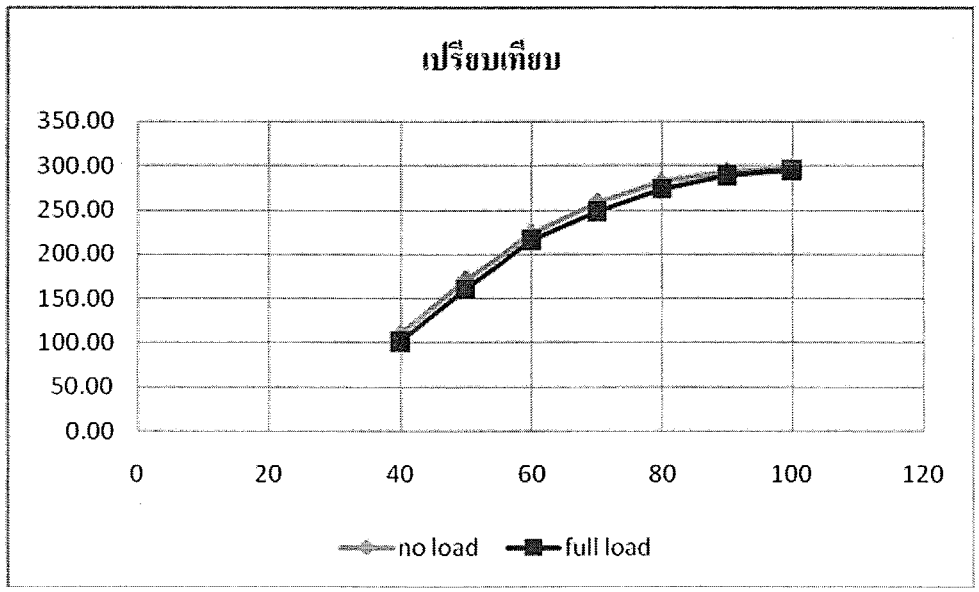
SPEED %	REVERSE no load	REVERSE full load	ERROR %
40	110.46	101.60	0.09
50	172.16	160.66	0.12
60	223.46	216.76	0.07
70	259.04	247.94	0.11
80	281.34	273.40	0.08
90	293.10	288.88	0.04
100	296.44	294.86	0.02

ตารางที่ 5.21 แสดงค่าความผิดพลาดที่เกิดขึ้นจากการปรับความเร็วในทิศทาง reverse (cm/s) จากค่าความต้านทานปรับค่าได้

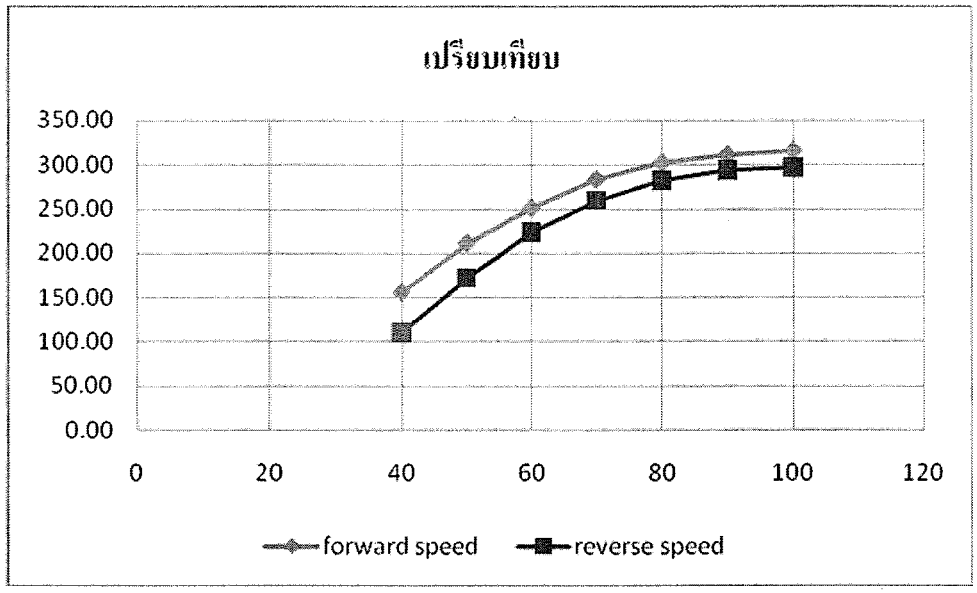
SPEED %	RPM R.no load	SPEED cm/s	RPM R.full load	SPEED cm/s	ERROR %
40	110.46	3.68	101.60	3.39	0.003
50	172.16	5.74	160.66	5.36	0.004
60	223.46	7.45	216.76	7.23	0.002
70	259.04	8.63	247.94	8.26	0.004
80	281.34	9.38	273.40	9.11	0.003
90	293.10	9.77	288.88	9.63	0.001
100	296.44	9.88	294.86	9.83	0.001



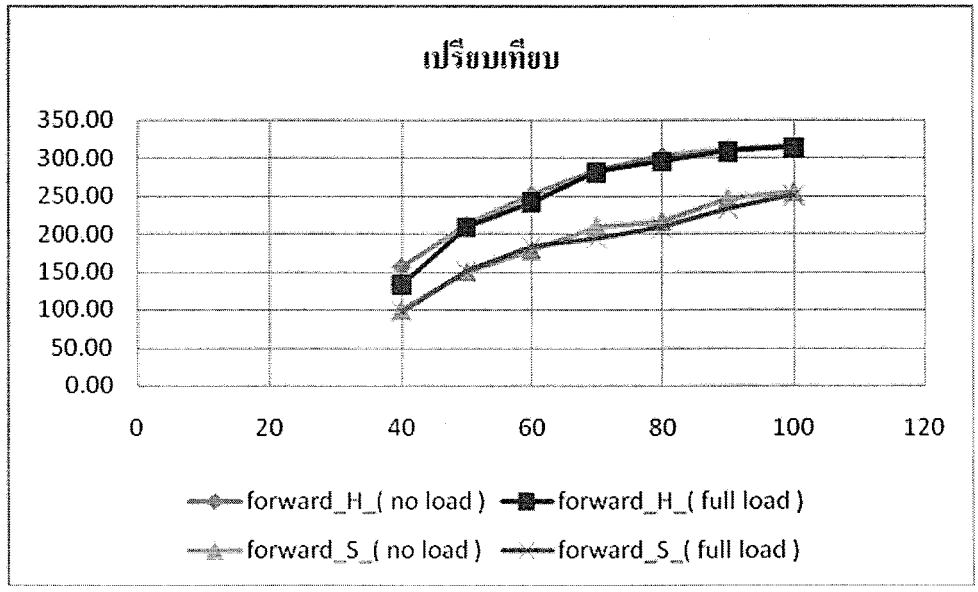
รูปที่ 5.20 เปรียบเทียบ Reverse ที่ no load และ full load จากการสั่งงานจากค่าความต้านทานปรับค่าได้



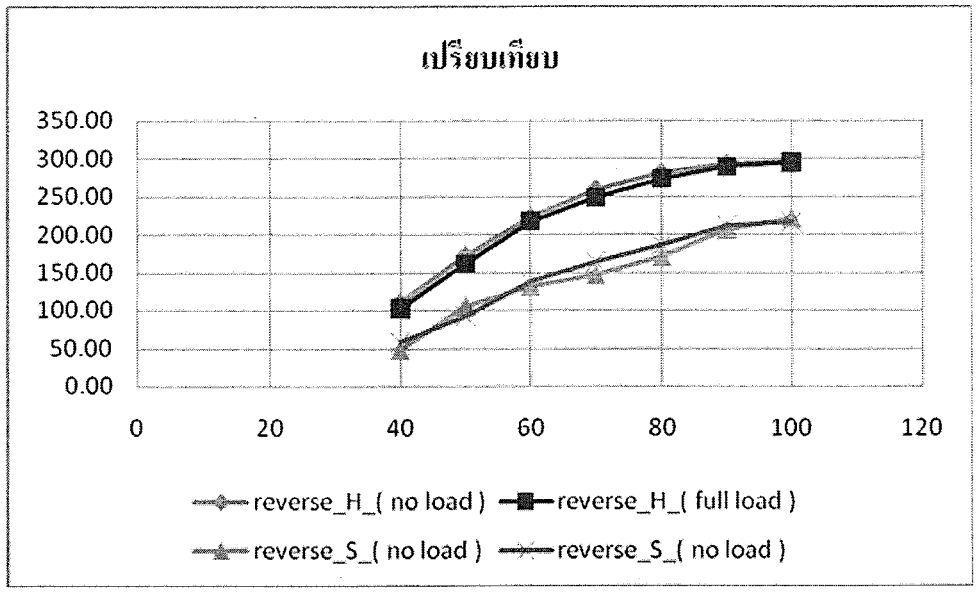
รูปที่ 5.21 เปรียบเทียบ forward และ reverse ที่ no load จากการสั่งงานจากค่าความต้านทานปรับค่าได้



รูปที่ 5.22 เปรียบเทียบ forward และ reverse ที่ no load จากการสั่งงานจากค่าความต้านทานปรับค่าได้



รูปที่ 5.23 เปรียบเทียบ Forward ที่ no load และ full load จากทั้งทางซอฟต์แวร์และฮาร์ดแวร์



รูปที่ 5.24 เปรียบเทียบ Reverse ที่ no load และ full load จากทั้งทางซอฟต์แวร์และฮาร์ดแวร์

สรุปผลการทดลอง

การทดลองนี้เป็นการทดลองเก็บค่าผลของโครงงานที่ได้จากชิ้นงาน โดยทำการวัดความเร็วการเคลื่อนที่ของแท่นวางด้วย Digital Tacho meter ที่ speed ต่าง ๆ เพิ่มที่ละ 10 % โดยแบ่งเป็นการสั่งงานจากซอฟแวร์และฮาร์ดแวร์แล้วจึงนำมาคำนวณเป็นความเร็วในหน่วย cm/s เพื่อให้ผู้ใช้งานสามารถเข้าใจได้ในหน่วยที่เป็นสากล จากผลการทดลองที่ได้พบว่าที่คำสั่ง speed ค่าเดียวกันการเคลื่อนที่ในทิศ Forward จะมีความเร็วมากกว่าการเคลื่อนที่ในทิศทาง Reverse ไม่ว่าจะสั่งงานจากซอฟแวร์หรือฮาร์ดแวร์ เมื่อพล็อตกราฟแสดงความสัมพันธ์ระหว่าง speed ที่สั่งงานกับความเร็วรอบที่วัดได้พบว่าได้กราฟที่มีแนวโน้มแบบ exponential

การสั่งงานจากซอฟแวร์ซึ่งได้ค่าที่แม่นยำตรงกับความต้องการมากกว่าการสั่งจากฮาร์ดแวร์ เนื่องจากการสั่งจากทางฮาร์ดแวร์ต้องผ่านการหมุนปรับแรงดันจากตัวต้านทานปรับค่าได้แล้วส่งให้ dsPIC ประมวลผลเป็นค่า dutycycle โดยต้องผ่านการ sampling สัญญาณ จาก 0 – 5 v ให้เป็นสัญญาณดิจิตอลที่มีค่าระหว่าง 0 – 1023 แล้วจึงปรับเทียบเป็น dutycycle 0 – 100 % ต่อไป ซึ่งขั้นตอนเหล่านี้ทำให้มีความคลาดเคลื่อนเกิดขึ้น แต่การสั่งงานจากซอฟแวร์ 0 – 100 % จะนำไปคิดเป็น dutycycle ได้ทันทีจึงมีความคลาดเคลื่อนน้อยกว่า

การสั่งงานที่ความเร็วต่ำกว่า 40 % ยังมีความเร็วไม่สม่ำเสมอและเคลื่อนที่ช้ามาก จึงไม่ได้ทำการบันทึกค่าลงในตารางผลการทดลอง และจากข้อกำหนดของผู้ใช้งานว่าน้ำหนักที่วางมีค่าไม่เกิน 5 kg. ผู้ทำโครงงานจึงได้ทดลองวาง load น้ำหนัก 5 kg ลงบนแท่น แล้วสั่งให้เคลื่อนที่ด้วยความเร็วต่างๆพบว่า ความเร็วลดลงไม่เกิน 1 %

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการทดลอง

จากการทดลองโครงงานนี้ จะสรุปได้ว่าการควบคุมความเร็วมอเตอร์กระแสตรงด้วยการใช้ตัวไมโครคอนโทรลเลอร์ dsPIC โดยใช้โมดูล MCPWM ในการสร้างสัญญาณพัลซ์วิดท์โมดูเลชั่น PWM สามารถเพิ่มหรือลดความเร็วได้ โดยการเพิ่มหรือลดค่าความกว้างของสัญญาณพัลซ์ให้มีค่ากว้างขึ้นหรือแคบลงซึ่งเป็นการเพิ่มค่าหรือลดค่ากระแสให้กับมอเตอร์แต่ถ้าสัญญาณพัลซ์ มีความกว้างเต็มที่แล้วจะไม่สามารถเพิ่มความเร็วได้อีก ในระบบการบังคับความเร็วสามารถจะกำหนดความเร็วให้คงที่หรือสามารถเปลี่ยนค่าได้ตามที่ต้องการ ซึ่งระบบนี้จะต้องมีหลักการอยู่บนพื้นฐานของการป้อนกลับเพื่อให้ได้ความเร็วที่มีความเที่ยงตรงสูง การบังคับความเร็วสามารถที่จะตอบสนองได้ทั้งสัญญาณคำสั่งบังคับให้ความเร็วคงที่หรือให้ความเร็วเปลี่ยนค่าไปได้ ซึ่งการเปลี่ยนไปตามของคำสั่งบังคับนั้นจะต้องอยู่ภายในแบนด์วิดท์ของระบบ ซึ่งจะสามารถควบคุมการหมุนซ้ายหมุนขวาได้และสามารถแสดงผลการควบคุมความเร็วมอเตอร์แบบตัวเลขได้ ซึ่งสามารถใช้สั่งงานที่แบบผ่านทางปุ่มกดหรือผ่านทางคอมพิวเตอร์ได้ซึ่งทำให้มีความหยืดหยุ่นในด้านการใช้งาน

เนื่องจากโปรเจคนี้ทำขึ้นเพื่อนำไปใช้งานจริง จากการทดสอบผู้ที่จะนำสายพานไปใช้งานพบว่าน้ำหนักของวัตถุที่จะนำมาวางบนสายพานมีน้ำหนักไม่เกิน 5 kg จึงได้นำมาทำการทดสอบโดยการวางน้ำหนักประมาณ 5 kg ลงบนสายพานแล้วสั่งงานให้หมุนด้วยความเร็วต่าง ๆ แล้วจับเวลาพบว่าได้ความเร็วไม่ต่างกันมากนักจึงไม่ได้ทำการชดเชยความเร็ว

จากผลการทดลองวัดสัญญาณและควบคุมความเร็วมอเตอร์ที่ระดับความเร็วต่าง ๆ พบว่าที่มีความเร็วตั้งแต่ 40% ขึ้นไปมอเตอร์หมุนได้เรียบและมีความเร็วคงที่แต่ที่ความเร็วต่ำ ๆ มอเตอร์เกิดเสียงดังในขณะหมุน การแก้ไขปัญหาในเรื่องแรงบิดที่น้อยอาจทำได้โดยการใส่ตัวเข้ารหัสเข้าไปเพื่อให้การควบคุมเป็นแบบ close loop

6.2 ปัญหา

1. จากการสังเกตการณ์แสดงผลความเร็วและทิศทางที่หน้าจอด้านขวาในโปรแกรม Visual Basic 6 พบว่าเฉพาะกรณีที่มีการสั่งงานจากฮาร์ดแวร์ยังมีการแสดงผลผิดพลาดบ้างเป็นบางครั้ง

2. การเคลื่อนที่ในทิศทาง Reverse ขณะที่แท่นวางกำลังเคลื่อนที่เข้าไปใกล้ limit switch ที่ระยะประมาณ 5 cm. พบว่าการหมุนของ screw มีความฝืดมากทำให้แท่นวางเคลื่อนที่ได้ช้าลงซึ่งจะเห็นได้อย่างชัดเจนที่ความเร็วต่ำ ๆ แต่ที่ความเร็ว 50 % ขึ้นไปจะไม่ได้รับผลกระทบมากนัก ผู้ทำโครงงานได้พยายามแก้ไขโดยทาสารหล่อลื่นที่เกลียวแต่ไม่ได้ช่วยให้ดีขึ้นมากนัก คาดว่าสาเหตุน่าจะมาจากปัญหาทางด้าน mechanic ของตัวอุปกรณ์ที่ได้รับมาในตอนแรก

3. เนื่องจากผู้ทำโครงงานยังไม่ใช่ผู้ชำนาญในด้านการใช้งาน dsPIC โปรแกรมที่เขียนขึ้นซึ่งมาจากการศึกษาด้วยตนเอง การทำงานของ dsPIC จึงมีความผิดพลาดบ้างเป็นบางครั้ง เช่น หยุดการทำงาน ถ้าเกิดกรณีเช่นนี้ขึ้นระหว่างใช้งานผู้ใช้งานสามารถแก้ไขได้ด้วยการกดปุ่มรีเซ็ตการทำงานที่ติดอยู่ที่บอร์ด

4. ไม่สามารถหาข้อมูลของมอเตอร์ที่ได้รับมา และสายสัญญาณที่ใช้กับตัวเข้ารหัสภายในได้

6.3 ข้อเสนอแนะ

เนื่องจากการสั่งงานที่ speed ต่ำ ๆ เช่น ที่ 0- 30 % แทบจะไม่ทำให้เกิดการเคลื่อนที่จึงไม่มีประโยชน์ที่จะสั่งงานด้วย speed ช่วงนี้เมื่อผู้ใช้งานสามารถกำหนดย่านความเร็วที่จะใช้งานได้แล้วสามารถพัฒนาปรับให้สั่งงานได้เฉพาะย่านที่ต้องการด้วยกระบวนการทางซอร์ฟแวร์ต่อไป

บรรณานุกรม

นคร ภักดีชาติ. **คู่มือการทดลองเบื้องต้น dsPIC Microcontroller ด้วยโปรแกรมภาษา C กับ MPLAB C30**. บริษัท อินโนเวทีฟ เอ็กเพอริเมนต์ จำกัด ; 2550

ศิวะ หงส์นภา. **การควบคุมและการประยุกต์ใช้งานดีซีไดรฟ์**. บริษัท กู๊ดวิลล์ไดเร็กชั่น จำกัด ; 2547

ไชยชาญ หินเกิด. **เครื่องกลไฟฟ้ากระแสตรง (Direct current machines)**. สำนักพิมพ์ ส.ส.ท. ; 2546

โยธิน เปรมปราณีรัชต์. **ระบบเซอร์โว และอิเล็กทรอนิกส์คอนโทรลมอเตอร์**. สำนักพิมพ์ Japan International Cooperation Agency (JICA) ; 2533

ศิวะ หงษ์นภา. **ระบบขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรง(วงจรภาคกำลัง)**. สำนักพิมพ์ ส.ส.ท. ; 2543

ภาคผนวก

'CODE VB6'

```
Public strIn As String
Public speed As Integer
Public a, i As Integer
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Private Sub cmdForward_Click()
   If Text1.Text = "" Then
   MsgBox "Please set speed", vbCritical + vbOKOnly, "Error"
   Else
   speed = Text1.Text
   Select Case speed
   Case Is < 10
         mscSender.Output = Text1.Text
         mscSender.Output = Chr(&H0)
         mscSender.Output = Chr(&H0)
         mscSender.Output = Chr(&H1)

   Case Is < 100
         mscSender.Output = Text1.Text
         mscSender.Output = Chr(&H1)
         mscSender.Output = Chr(&H1)

   Case 100
         mscSender.Output = Chr(&H1)
         mscSender.Output = Chr(&H0)
         mscSender.Output = Chr(&H2)
         mscSender.Output = Chr(&H1)
   End Select
   End If
End Sub

Private Sub cmdReverse_Click()
   If Text1.Text = "" Then
   MsgBox "Please set speed", vbCritical + vbOKOnly, "Error"
   Else
   speed = Text1.Text
   Select Case speed
   Case Is < 10
         mscSender.Output = Text1.Text
         mscSender.Output = Chr(&H0)
         mscSender.Output = Chr(&H0)
         mscSender.Output = Chr(&H0)

   Case Is < 100
         mscSender.Output = Text1.Text
         mscSender.Output = Chr(&H1)
         mscSender.Output = Chr(&H0)

   Case 100
```

```
        mscSender.Output = Chr(&H1)
        mscSender.Output = Chr(&H0)
        mscSender.Output = Chr(&H2)
        mscSender.Output = Chr(&H0)
    End Select
    End If
End Sub

Private Sub cmdStop_Click()
    If Text1.Text = "" Then
    Else
    mscSender.Output = Chr(&H0)
    mscSender.Output = Chr(&H0)
    mscSender.Output = Chr(&H0)
    mscSender.Output = Chr(&H2)
    End If
End Sub

Private Sub Form_Load()
    i = 1
    UniGaugeX1.ResourceImageIndex = 15
    mscSender.CommPort = 1
    mscSender.InBufferSize = 1024
    mscSender.Settings = "9600, n , 8 , 1"
    mscSender.PortOpen = True
    mscSender.RThreshold = 1
    Timer2.Interval = 1
    Timer3.Interval = 3000
    Timer3.Enabled = True
    Timer4.Interval = 100
    Timer4.Enabled = False
End Sub

Private Sub mscSender_OnComm()
    Sleep (10)
    strIn = mscSender.Input
    If i = 1 Then
    UniLCDX1.Value = Val(strIn)
    UniGaugeX1.Float = Val(strIn)
    i = i - 1
    Else
    Label8.Caption = strIn
    i = 1
    End If
    If UniLCDX1.Value = 0 Then
    Label8.Caption = "STOP"
    End If
End Sub

Private Sub Text1_Change()
```

```
    If Text1.Text <> "" Then
    If Text1.Text > 100 Then
        MsgBox "set speed 0-100 only", vbCritical + vbOKOnly, "Error"
        Text1.Text = ""
        Exit Sub
    End If
    If Text1.Text < 0 Then
        MsgBox "set speed 0-100 only", vbCritical + vbOKOnly, "Error"
        Text1.Text = ""
        Exit Sub
    End If
    End If
End Sub

Private Sub Timer2_Timer()
    UniMatrixX3.Text = Time$
End Sub

Private Sub Timer3_Timer()
    Timer3.Enabled = False
    Timer4.Enabled = True
    a = 30
End Sub

Private Sub Timer4_Timer()
    Select Case a
    Case Is = 30
        Label4.Caption = "I"
        a = a - 1
    Case Is = 29
        Label4.Caption = "In"
        a = a - 1
    Case Is = 28
        Label4.Caption = "Ins"
        a = a - 1
    Case Is = 27
        Label4.Caption = "Inst"
        a = a - 1
    Case Is = 26
        Label4.Caption = "Instr"
        a = a - 1
    Case Is = 25
        Label4.Caption = "Instru"
        a = a - 1
    Case Is = 24
        Label4.Caption = "Instrum"
        a = a - 1
    Case Is = 23
        Label4.Caption = "Instrume"
        a = a - 1
```

```
Case Is = 22
   Label4.Caption = "Instrumen"
   a = a - 1
Case Is = 21
   Label4.Caption = "Instrument"
   a = a - 1
Case Is = 20
   Label4.Caption = "Instrumenta"
   a = a - 1
Case Is = 19
   Label4.Caption = "Instrumentat"
   a = a - 1
Case Is = 18
   Label4.Caption = "Instrumentati"
   a = a - 1
Case Is = 17
   Label4.Caption = "Instrumentatio"
   a = a - 1
Case Is = 16
   Label4.Caption = "Instrumentation"
   a = a - 1
Case Is = 15
   Label4.Caption = "Instrumentation E"
   a = a - 1
Case Is = 14
   Label4.Caption = "Instrumentation En"
   a = a - 1
Case Is = 13
   Label4.Caption = "Instrumentation Eng"
   a = a - 1
Case Is = 12
   Label4.Caption = "Instrumentation Engi"
   a = a - 1
Case Is = 11
   Label4.Caption = "Instrumentation Engin"
   a = a - 1
Case Is = 10
   Label4.Caption = "Instrumentation Engine"
   a = a - 1
Case Is = 9
   Label4.Caption = "Instrumentation Enginee"
   a = a - 1
Case Is = 8
   Label4.Caption = "Instrumentation Engineer"
   a = a - 1
Case Is = 7
   Label4.Caption = "Instrumentation Engineeri"
   a = a - 1
Case Is = 6
   Label4.Caption = "Instrumentation Engineerin"
```

```
    a = a - 1
  Case Is = 5
    Label4.Caption = "Instrumentation Engineering"
    a = a - 1
  Case Is = 4
    Label4.Caption = "Instrumentation Engineering K"
    a = a - 1
  Case Is = 3
    Label4.Caption = "Instrumentation Engineering KM"
    a = a - 1
  Case Is = 2
    Label4.Caption = "Instrumentation Engineering KMI"
    a = a - 1
  Case Is = 1
    Label4.Caption = "Instrumentation Engineering KMIT"
    a = a - 1
  Case Is = 0
    Label4.Caption = "Instrumentation Engineering KMITL"
    a = a - 1
    Timer4.Enabled = False
    Timer3.Enabled = True
  End Select
End Sub
```

```
//*************************************//
//**        Target MCU  : dsPIC30F2010       **//
//**              X-TAL : 7.3728 MHz         **//
//*************************************//
#include <p30f2010.h>
#include <stdio.h>
#include <pwm.h>
#include <adc10.h>
#include <ports.h>
#include<uart.h>
#define PERIOD 0xFFFF
unsigned char speed[10];
char botton0,botton1,botton2;
unsigned int ADCvalue,dutycycle,char1,char2,char3,input;
unsigned int data1,data2,data3,data4,dir,LS1,LS2,TX,i=1;
//-------------------------------------------------------------------------------------------------//
//----------------------------Setup Configuration For dsPIC30F2010 ----------------------------//
//-------------------------------------------------------------------------------------------------//
_FOSC(CSW_FSCM_ON & XT_PLL4)
_FWDT(WDT_OFF)
_FBORPOR(MCLR_EN & PBOR_OFF)
_FGS(CODE_PROT_OFF)
```

```
//-------------------------------------------------------------------------------------------------//
//-----------------------------------Delay Time Function ------------------------------------------//
//-------------------------------------------------------------------------------------------------//
void delay_ms(unsigned int ms)
{
        unsigned int x,a;
        for(x=0;x<ms;x++)
{
        for(a=0;a<816;a++);
}
}
//-------------------------------------------------------------------------------------------------//
//-----------------------------------Function stop motor-------------------------------------------//
//-------------------------------------------------------------------------------------------------//
void stop_motor(void)
{
        LATBbits.LATB2 = 0;
        LATBbits.LATB3 = 0;
}
//-------------------------------------------------------------------------------------------------//
//-----------------------------------Function drive motor forward----------------------------------//
//-------------------------------------------------------------------------------------------------//
void forward(void)
{
        LATBbits.LATB2 = 0;
        LATBbits.LATB3 = 1;
}
```

```
//-------------------------------------------------------------------------------------------------//
//---------------------------------- Function drive motor reverse ---------------------------------//
//-------------------------------------------------------------------------------------------------//
void reverse(void)
{
        LATBbits.LATB2 = 1;
        LATBbits.LATB3 = 0;
}
//-------------------------------------------------------------------------------------------------//
//---------------------------------- Function read analog signal ----------------------------------//
//-------------------------------------------------------------------------------------------------//
int read_ADC(void)
{
        ADCON1bits.SAMP = 1;
        while(!ADCON1bits.SAMP);
        ConvertADC10();
        while(!BusyADC10());
        return (ReadADC10(0));
}
//-------------------------------------------------------------------------------------------------//
//----------------------- Interrupt service routine for External interrupt 0 ----------------------//
//-------------------------------------------------------------------------------------------------//
void _ISR _INT0Interrupt(void)
{
        delay_ms(10);
        IFS0bits.INT0IF = 0;
        botton0 = 1;
}
```

```
//-------------------------------------------------------------------------------------------------//
//----------------------- Interrupt service routine for External interrupt 1 ----------------------//
//-------------------------------------------------------------------------------------------------//
void _ISR _INT1Interrupt(void)
{
        delay_ms(10);
        IFS1bits.INT1IF = 0;
        if (LS2)
        botton1 = 1;
}
//-------------------------------------------------------------------------------------------------//
//----------------------- Interrupt service routine for External interrupt 2 ----------------------//
//-------------------------------------------------------------------------------------------------//
void _ISR _INT2Interrupt(void)
{
        delay_ms(10);
        IFS1bits.INT2IF = 0;
        if (LS1)
        botton2 = 1;
}
void _ISR _U1RXInterrupt(void)
{
        IFS0bits.U1RXIF = 0;
}
void _ISR _U1TXInterrupt(void)
{
        IFS0bits.U1TXIF = 0;
}
```

```
//-------------------------------------------------------------------------------------------------//
//---------------------- Interrupt service routine for MCPWM interrupt ----------------------//
//-------------------------------------------------------------------------------------------------//
void _ISR _PWMInterrupt(void)
{
        IFS2bits.PWMIF = 0;
}
//-------------------------------------------------------------------------------------------------//
//-------------------------------- Function set port input,output ---------------------------------//
//-------------------------------------------------------------------------------------------------//
void set_ports()
{
        TRISBbits.TRISB2 = 0;
        TRISBbits.TRISB3 = 0;
        TRISBbits.TRISB4 = 1;
        TRISBbits.TRISB5 = 1;
        TRISDbits.TRISD1 = 1;
        TRISDbits.TRISD0 = 1;
        TRISEbits.TRISE8 = 1;
        ConfigINT0(RISING_EDGE_INT & EXT_INT_ENABLE & EXT_INT_PRI_5);
        ConfigINT1(RISING_EDGE_INT & EXT_INT_ENABLE & EXT_INT_PRI_4);
        ConfigINT2(RISING_EDGE_INT & EXT_INT_ENABLE & EXT_INT_PRI_3);
}
void pwm_init()
{
        ConfigIntMCPWM (PWM_INT_EN &
        PWM_FLTA_DIS_INT &
        PWM_INT_PR7);
```

```
        PTCON = 0x8000;
        PWMCON1 = 0x0F20;
        PWMCON2 = 0x0002;
        PTPER = PERIOD;
}
//-------------------------------------------------------------------------------------------------//
//-------------------------------- Function for configuration UART1 ------------------------------//
//-------------------------------------------------------------------------------------------------//
void uart1_init()
{
        CloseUART1();
        ConfigIntUART1(UART_RX_INT_EN &
        UART_RX_INT_PR6 &
        UART_TX_INT_EN &
        UART_TX_INT_PR2);
        U1BRG = 47;
        U1MODE = 0x8400;
        U1STAbits.UTXISEL = 1;
        U1STAbits.URXISEL = 0;
        U1STAbits.ADDEN   = 0;
        U1STAbits.OERR    = 0;
        U1STAbits.UTXBRK  = 0;
        U1STAbits.UTXEN   = 1;
}
```

```
//-------------------------------------------------------------------------------------------//
//---------------------------- Function initialize ACD module ------------------------------//
//-------------------------------------------------------------------------------------------//
void adc_init()
{
        unsigned int Channel, PinConfig, Scanselect ;
        unsigned int Adcon3_reg, Adcon2_reg,Adcon1_reg;
        ADCON1bits.ADON = 0;
        Channel = ADC_CH0_POS_SAMPLEA_AN0 &
                  ADC_CH0_NEG_SAMPLEA_NVREF ;
        SetChanADC10(Channel);
        ADPCFGbits.PCFG4 = 1;
        ADPCFGbits.PCFG5 = 1;
        ConfigIntADC10(ADC_INT_DISABLE);
        PinConfig = ENABLE_AN0_ANA ;
        Scanselect = SKIP_SCAN_AN1 &
                     SKIP_SCAN_AN2 &
                     SKIP_SCAN_AN3 &
                     SKIP_SCAN_AN4 &
                     SKIP_SCAN_AN5 ;
        Adcon3_reg = ADC_SAMPLE_TIME_10 &
                       ADC_CONV_CLK_INTERNAL_RC &
                       ADC_CONV_CLK_13Tcy;

        Adcon2_reg = ADC_VREF_AVDD_AVSS &
                       ADC_SCAN_ON &
                       ADC_ALT_BUF_OFF &
```

```
                    ADC_ALT_INPUT_OFF &
                    ADC_CONVERT_CH0&
                    ADC_SAMPLES_PER_INT_16;
        Adcon1_reg = ADC_MODULE_ON &
                    ADC_IDLE_CONTINUE &
                    ADC_FORMAT_INTG &
                    ADC_CLK_MANUAL &
                    ADC_SAMPLE_SIMULTANEOUS &
                    ADC_AUTO_SAMPLING_OFF;
        OpenADC10(Adcon1_reg, Adcon2_reg,Adcon3_reg,PinConfig, Scanselect);
}
//-------------------------------------------------------------------------------------------------//
//--------------------------Function control motor with computer-----------------------------//
//-------------------------------------------------------------------------------------------------//
void program_control()
{
        if((DataRdyUART1())&&(i==1))
        {
                data1 = ReadUART1();
                data1 = data1&(0x000F);
                i = 2;
        }
        if((DataRdyUART1())&&(i==2))
        {
                data2 = ReadUART1();
                data2 = data2&(0x000F);
                i = 3;
        }
```

```
if((DataRdyUART1())&&(i==3))
{
        data3 = ReadUART1();
        data3 = data3&(0x000F);
        i = 4;
}
if((DataRdyUART1())&&(i==4))
{
        dir = ReadUART1();
        dir = dir&(0x000F);
        i = 5;
}
if(i==5)
{
        switch (data3)
        {
                case 0:
                        TX = data1;
                        dutycycle = data1*(PERIOD/100);
                        PDC2 = dutycycle;
                        break;
                case 1:
                        data1 = (data1)*10+data2;
                        TX = data1;
                        dutycycle = data1*(PERIOD/100);
                        PDC2 = dutycycle;
                        break;
                default:
```

```
                TX = 100;
                data1 = 100;
                dutycycle = data1*(PERIOD/100);
                PDC2 = dutycycle;
}
switch (dir)
{
        case 1:
                if(LS2)
                {
                        forward();
                        sprintf(speed,"%d",TX);
                        putsUART1((unsigned int *)speed);
                        while(BusyUART1());
                        delay_ms(100);
                        putsUART1((unsigned int *)"FORWARD");
                        while(BusyUART1());
                }
                break;
        case 0:
                if(LS1)
                {
                        reverse();
                        sprintf(speed,"%d",TX);
                        putsUART1((unsigned int *)speed);
                        while(BusyUART1());
                        delay_ms(100);
                        putsUART1((unsigned int *)"REVERSE");
```

```
                    while(BusyUART1());
                }
                break;
            default:
                stop_motor();
                TX = 0;
                sprintf(speed,"%d",TX);
                putsUART1((unsigned int *)speed);
                while(BusyUART1());
                delay_ms(100);
                putsUART1((unsigned int *)"STOP");
                while(BusyUART1());
        }
        i = 1;
    }
}
//-------------------------------------------------------------------------------------------------//
//----------------------------function control motor with switchs----------------------------------//
//-------------------------------------------------------------------------------------------------//
void analog_control()
{
    read_ADC();
    ADCvalue = read_ADC();
    if (botton0)
    {
        botton0 = 0;
        stop_motor();
        putsUART1((unsigned int *)"0");
```

```
        while(BusyUART1());
        delay_ms(10);
        putsUART1((unsigned int *)"STOP");
        while(BusyUART1());
}
if (botton1&&LS2)
{
        botton1 = 0;
        input = ((ADCvalue*50)/512);
        dutycycle = (input*(PERIOD/100));
        PDC2 = dutycycle;
        sprintf(speed,"%d",input);
        putsUART1((unsigned int *)speed);
        while(BusyUART1());
        forward();
        delay_ms(10);
        putsUART1((unsigned int *)"FORWARD");
        while(BusyUART1());
}
if (botton2&&LS1)
{
        botton2 = 0;
        input = ((ADCvalue*50)/512);
        dutycycle = (input*(PERIOD/100));
        PDC2 = dutycycle;
        sprintf(speed,"%d",input);
        putsUART1((unsigned int *)speed);
        while(BusyUART1());
```

```
            reverse();
            delay_ms(10);
            putsUART1((unsigned int *)"REVERSE");
            while(BusyUART1());
        }
}
//-------------------------------------------------------------------------------------------------//
//----------------------------------function check limit switchs-----------------------------------//
//-------------------------------------------------------------------------------------------------//
void check_LS()
{
        LS1 = PORTBbits.RB4;
        LS2 = PORTBbits.RB5;
        if((!LS1&&PORTBbits.RB2)||(!LS2&&PORTBbits.RB3))
        {
            stop_motor();
            putsUART1((unsigned int *)"0");
            while(BusyUART1());
            delay_ms(10);
            putsUART1((unsigned int *)"STOP");
            while(BusyUART1());
        }
}
```

```
//-------------------------------------------------------------------------------------------------//
//------------------------------------------ Main Program -----------------------------------------//
//-------------------------------------------------------------------------------------------------//
int main(void)
{
        pwm_init();
        uart1_init();
        adc_init();
        set_ports();
        while(1)
        {
                program_control();
                analog_control();
                check_LS();
        }
}
```

8.2 Configuring Analog Port Pins

The use of the ADPCFG and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bit set (input). If the TRIS bit is cleared (output), the digital output level (V_{OH} or V_{OL}) will be converted.

When reading the PORT register, all pins configured as analog input channel will read as cleared (a low level).

Pins configured as digital inputs will not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins), may cause the input buffer to consume current that exceeds the device specifications.

8.2.1 I/O PORT WRITE/READ TIMING

One instruction cycle is required between a port direction change or port write operation and a read operation of the same port. Typically this instruction would be a NOP.

EXAMPLE 8-1: PORT WRITE/READ EXAMPLE

```
MOV 0xFF00, W0; Configure PORTB<15:8>
      ; as inputs
MOV W0, TRISBB; and PORTB<7:0> as outputs
NOP   ; Delay 1 cycle
btssPORTB, #13; Next Instruction
```

8.3 Input Change Notification Module

The Input Change Notification module provides the dsPIC30F devices the ability to generate interrupt requests to the processor in response to a change-of-state on selected input pins. This module is capable of detecting input change-of-states even in Sleep mode, when the clocks are disabled. There are up to 22 external signals (CN0 through CN21) that may be selected (enabled) for generating an interrupt request on a change-of-state.

TABLE 8-1: dsPIC30F2010 PORT REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TRISB	02C6	—	—	—	—	—	—	—	—	—	—	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	0000 0000 0011 1111
PORTB	02C8	—	—	—	—	—	—	—	—	—	—	RB5	RB4	RB3	RB2	RB1	RB0	0000 0000 0000 0000
LATB	02CB	—	—	—	—	—	—	—	—	—	—	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	0000 0000 0000 0000
TRISC	02CC	TRISC15	TRISC14	TRISC13	—	—	—	—	—	—	—	—	—	—	—	—	—	1110 0000 0000 0000
PORTC	02CE	RC15	RC14	RC13	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
LATC	02D0	LATC15	LATC14	LATC13	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
TRISD	02D2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TRISD1	TRISD0	0000 0000 0000 0111
PORTD	02D4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	RD1	RD0	0000 0000 0000 0000
LATD	02D6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	LATD1	LATD0	0000 0000 0000 0000
TRISE	02D8	—	—	—	—	—	—	—	TRISE8	—	—	TRISE5	TRISE4	TRISE3	TRISE2	TRISE1	TRISE0	0000 0001 0011 1111
PORTE	02DA	—	—	—	—	—	—	—	RE8	—	—	RE5	RE4	RE3	RE2	RE1	RE0	0000 0000 0000 0000
LATE	02DC	—	—	—	—	—	—	—	LATE8	—	—	LATE5	LATE4	LATE3	LATE2	LATE1	LATE0	0000 0000 0000 0000
TRISF	02EE	—	—	—	—	—	—	—	—	—	—	—	—	TRISF3	TRISF2	—	—	0000 0000 0000 1100
PORTF	02E0	—	—	—	—	—	—	—	—	—	—	—	—	RF3	RF2	—	—	0000 0000 0000 0000
LATF	02E2	—	—	—	—	—	—	—	—	—	—	—	—	LATF3	LATF2	—	—	0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 8-2: INPUT CHANGE NOTIFICATION REGISTER MAP (BITS 15-0)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
CNEN1	00C0	CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE	CN9IE	CN8IE	CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE	0000 0000 0000 0000
CNEN2	00C2	—	—	—	—	—	—	—	—	—	—	CN21IE	CN20IE	CN19IE	CN18IE	CN17IE	CN16IE	0000 0000 0000 0000
CNPU1	00C4	CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE	CN9PUE	CN8PUE	CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE	0000 0000 0000 0000
CNPU2	00C6	—	—	—	—	—	—	—	—	—	—	CN21PUE	CN20PUE	CN19PUE	CN18PUE	CN17PUE	CN16PUE	0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to *dsPIC30F Family Reference Manual* (DS70046) for descriptions of register bit fields.

14.0 MOTOR CONTROL PWM MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046).

This module simplifies the task of generating multiple, synchronized Pulse Width Modulated (PWM) outputs. In particular, the following power and motion control applications are supported by the PWM module:

- Three Phase AC Induction Motor
- Switched Reluctance (SR) Motor
- Brushless DC (BLDC) Motor
- Uninterruptible Power Supply (UPS)

The PWM module has the following features:

- 6 PWM I/O pins with 3 duty cycle generators
- Up to 16-bit resolution
- 'On-the-Fly' PWM frequency changes
- Edge and Center Aligned Output modes
- Single Pulse Generation mode
- Interrupt support for asymmetrical updates in Center Aligned mode
- Output override control for Electrically Commutative Motor (ECM) operation
- 'Special Event' comparator for scheduling other peripheral events
- $\overline{\text{FLTA}}$ pins to optionally drive each of the PWM output pins to a defined state

This module contains 3 duty cycle generators, numbered 1 through 3. The module has 6 PWM output pins, numbered PWM1H/PWM1L through PWM3H/PWM3L. The six I/O pins are grouped into high/low numbered pairs, denoted by the suffix H or L, respectively. For complementary loads, the low PWM pins are always the complement of the corresponding high I/O pin.

A simplified block diagram of the Motor Control PWM modules is shown in Figure 14-1.

The PWM module allows several modes of operation which are beneficial for specific power control applications.

TABLE 14-1: PWM REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000
PTMR	01C2	PTDIR	PWM Timer Count Value															0000 0000 0000 0000
PTPER	01C4	—	PWM Time Base Period Register															0000 0000 0000 0000
SEVTCMP	01C6	SEVTDIR	PWM Special Event Compare Register															0000 0000 0000 0000
PWMCON1	01C8	—	—	—	—	—	PTMOD3	PTMOD2	PTMOD1	—	PEN3H	PEN2H	PEN1H	—	PEN3L	PEN2L	PEN1L	0000 0000 0111 0111
PWMCON2	01CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	—	OSYNC	UDIS	0000 0000 0000 0000
DTCON1	01CC	—	—	—	—	—	—	—	—	DTAPS<1:0>		Dead Time A Value						0000 0000 0000 0000
FLTACON	01D0	—	—	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	—	FAEN3	FAEN2	FAEN1	0000 0000 0000 0000
OVDCON	01D4	—	—	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	—	—	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	0011 1111 0000 0000
PDC1	01D6	PWM Duty Cycle #1 Register																0000 0000 0000 0000
PDC2	01D8	PWM Duty Cycle #2 Register																0000 0000 0000 0000
PDC3	01DA	PWM Duty Cycle #3 Register																0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to *dsPIC30F Family Reference Manual* (DS70046) for descriptions of register bit fields.

17.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046).

This section describes the Universal Asynchronous Receiver/Transmitter Communications module.

Note: Since dsPIC30F2010 devices have only one UART, all references to Ux... imply that x = 1 only.

17.1 UART Module Overview

The key features of the UART module are:

- Full-duplex, 8 or 9-bit data communication
- Even, Odd or No Parity options (for 8-bit data)
- One or two Stop bits
- Fully integrated Baud Rate Generator with 16-bit prescaler
- Baud rates range from 38 bps to 1.875 Mbps at a 30 MHz instruction rate
- 4-word deep transmit data buffer
- 4-word deep receive data buffer
- Parity, Framing and Buffer Overrun error detection
- Support for Interrupt only on Address Detect (9th bit = 1)
- Separate Transmit and Receive Interrupts
- Loopback mode for diagnostic support

FIGURE 17-1: UART TRANSMITTER BLOCK DIAGRAM

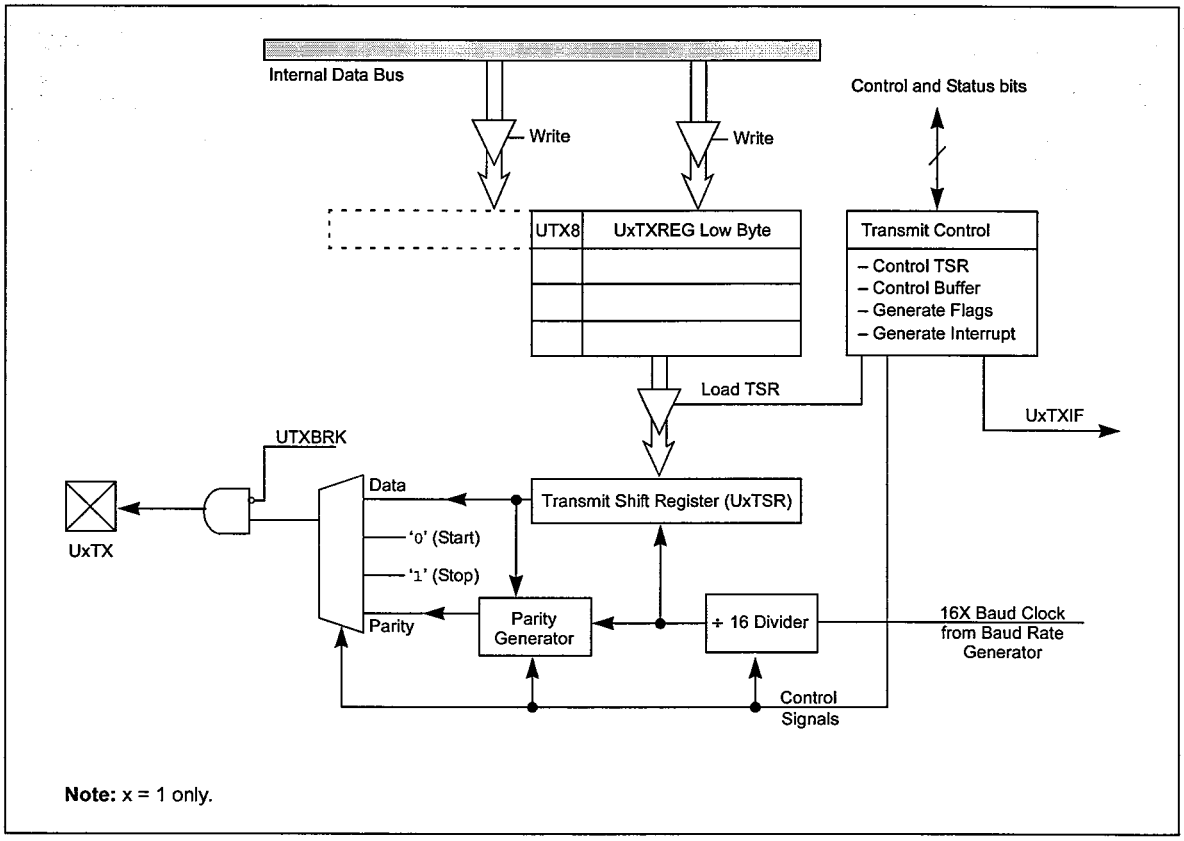
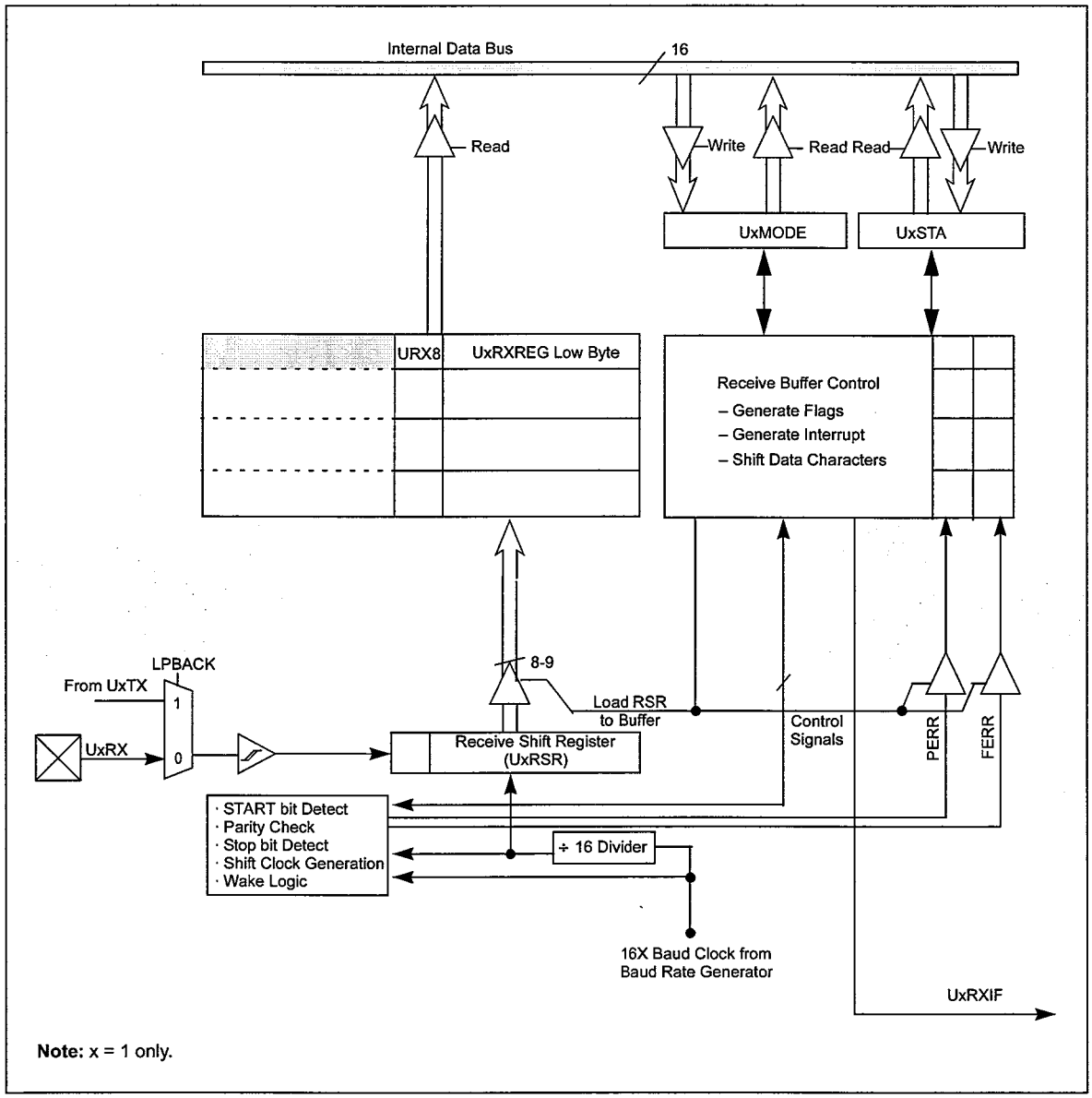


FIGURE 17-2: UART RECEIVER BLOCK DIAGRAM



Note: x = 1 only.

TABLE 17-1: UART1 REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
U1MODE	020C	UARTEN	—	USIDL	—	—	ALTIO	—	—	WAKE	LPBACK	ABAUD	—	—	PDSEL1	PDSEL0	STSEL	0000 0000 0000 0000
U1STA	020E	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0000 0001 0001 0000
U1TXREG	0210	—	—	—	—	—	—	—	UTX8	Transmit Register								0000 000u uuuu uuuu
U1RXREG	0212	—	—	—	—	—	—	—	URX8	Receive Register								0000 0000 0000 0000
U1BRG	0214	Baud Rate Generator Prescaler																0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to *dsPIC30F Family Reference Manual* (DS70046) for descriptions of register bit fields.

18.0 10-BIT HIGH SPEED ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046).

The10-bit high-speed analog-to-digital converter (A/D) allows conversion of an analog input signal to a 10-bit digital number. This module is based on a Successive Approximation Register (SAR) architecture, and provides a maximum sampling rate of 500 ksps. The A/D module has up to 16 analog inputs which are multiplexed into four sample and hold amplifiers. The output of the sample and hold is the input into the converter, which generates the result. The analog reference voltages are software selectable to either the device supply voltage (AVDD/AVSS) or the voltage level on the (VREF+/VREF-) pin. The A/D converter has a unique feature of being able to operate while the device is in Sleep mode.

The A/D module has six 16-bit registers:

- A/D Control Register1 (ADCON1)
- A/D Control Register2 (ADCON2)
- A/D Control Register3 (ADCON3)
- A/D Input Select Register (ADCHS)
- A/D Port Configuration Register (ADPCFG)
- A/D Input Scan Selection Register (ADCSSL)

The ADCON1, ADCON2 and ADCON3 registers control the operation of the A/D module. The ADCHS register selects the input channels to be converted. The ADPCFG register configures the port pins as analog inputs or as digital I/O. The ADCSSL register selects inputs for scanning.

Note: The SSRC<2:0>, ASAM, SIMSAM, SMPI<3:0>, BUFM and ALTS bits, as well as the ADCON3 and ADCSSL registers, must not be written to while ADON = 1. This would lead to indeterminate results.

The block diagram of the A/D module is shown in Figure 18-1.

FIGURE 18-1: 10-BIT HIGH SPEED A/D FUNCTIONAL BLOCK DIAGRAM

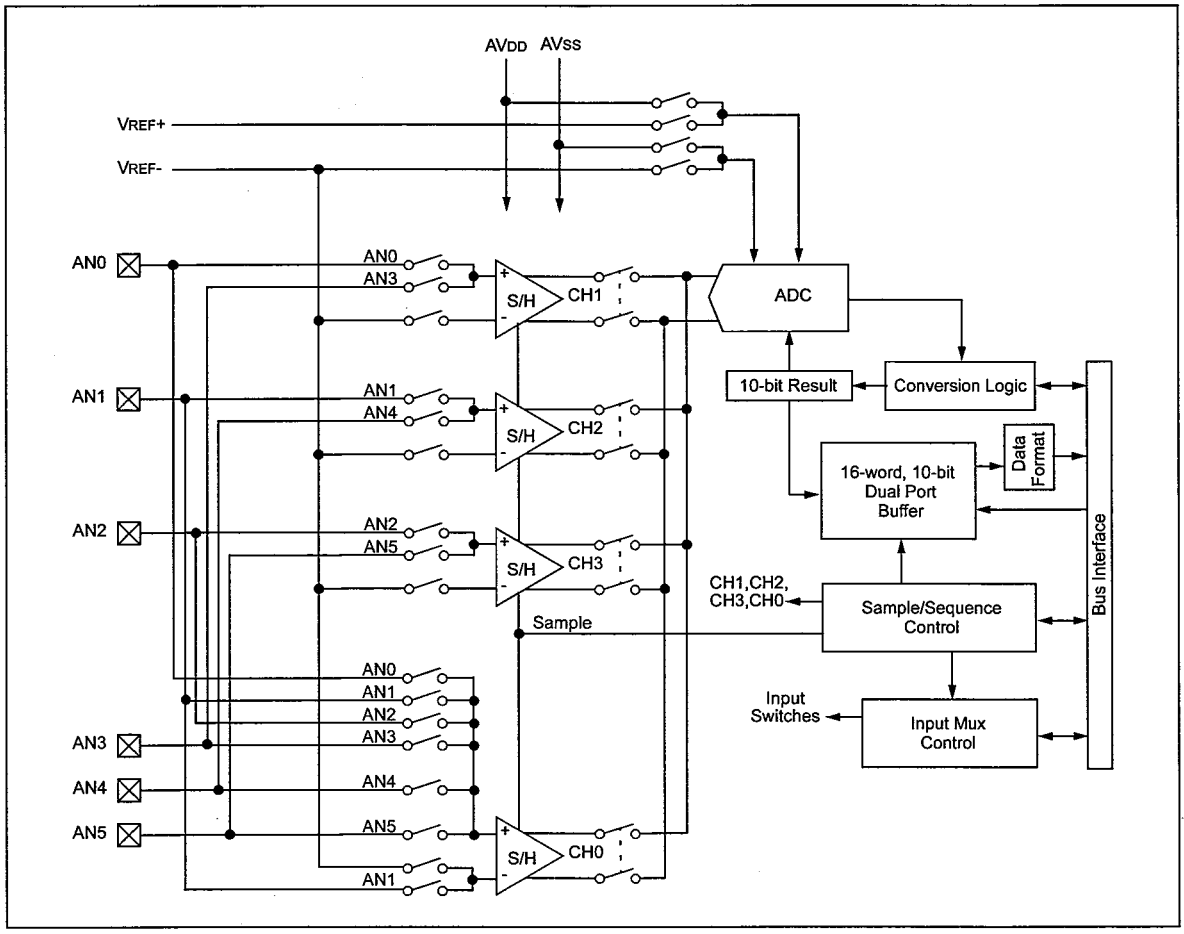


TABLE 18-1: ADC REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
ADCBUF0	0280	—	—	—	—	—	—	ADC Data Buffer 0										0000 00uu uuuu uuuu
ADCBUF1	0282	—	—	—	—	—	—	ADC Data Buffer 1										0000 00uu uuuu uuuu
ADCBUF2	0284	—	—	—	—	—	—	ADC Data Buffer 2										0000 00uu uuuu uuuu
ADCBUF3	0286	—	—	—	—	—	—	ADC Data Buffer 3										0000 00uu uuuu uuuu
ADCBUF4	0288	—	—	—	—	—	—	ADC Data Buffer 4										0000 00uu uuuu uuuu
ADCBUF5	028A	—	—	—	—	—	—	ADC Data Buffer 5										0000 00uu uuuu uuuu
ADCBUF6	028C	—	—	—	—	—	—	ADC Data Buffer 6										0000 00uu uuuu uuuu
ADCBUF7	028E	—	—	—	—	—	—	ADC Data Buffer 7										0000 00uu uuuu uuuu
ADCBUF8	0290	—	—	—	—	—	—	ADC Data Buffer 8										0000 00uu uuuu uuuu
ADCBUF9	0292	—	—	—	—	—	—	ADC Data Buffer 9										0000 00uu uuuu uuuu
ADCBUFA	0294	—	—	—	—	—	—	ADC Data Buffer 10										0000 00uu uuuu uuuu
ADCBUFB	0296	—	—	—	—	—	—	ADC Data Buffer 11										0000 00uu uuuu uuuu
ADCBUFC	0298	—	—	—	—	—	—	ADC Data Buffer 12										0000 00uu uuuu uuuu
ADCBUFD	029A	—	—	—	—	—	—	ADC Data Buffer 13										0000 00uu uuuu uuuu
ADCBUFE	029C	—	—	—	—	—	—	ADC Data Buffer 14										0000 00uu uuuu uuuu
ADCBUFF	029E	—	—	—	—	—	—	ADC Data Buffer 15										0000 00uu uuuu uuuu
ADCON1	02A0	ADON	—	ADSIDL	—	—	—	FORM<1:0>		SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE	0000 0000 0000 0000
ADCON2	02A2	VCFG<2:0>			—	—	CSCNA	CHPS<1:0>		BUFS	—	SMPI<3:0>				BUFM	ALTS	0000 0000 0000 0000
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>						0000 0000 0000 0000
ADCHS	02A6	CH123NB<1:0>		CH123SB	CH0NB	CH0SB<3:0>				CH123NA<1:0>		CH123SA	CH0NA	CH0SA<3:0>				0000 0000 0000 0000
ADPCFG	02A8	—	—	—	—	—	—	—	—	—	—	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000
ADCSSL	02AA	—	—	—	—	—	—	—	—	—	—	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0	0000 0000 0000 0000

Legend: u = uninitialized bit

Note: Refer to *dsPIC30F Family Reference Manual* (DS70046) for descriptions of register bit fields.

+5V Powered, Dual RS-232 Transmitter/Receiver

The ICL232 is a dual RS-232 transmitter/receiver interface circuit that meets all EIA RS-232C and V.28 specifications. It requires a single +5V power supply, and features two onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply.

The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to +30V, and have a 3kΩ to 7kΩ input impedance. The receivers also have hysteresis to improve noise rejection.

Ordering Information

PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. DWG. #
ICL232CPE	0 to 70	16 Ld PDIP	E16.3
ICL232CBE	0 to 70	16 Ld SOIC	M16.3
ICL232CBET	16 Ld SOIC Tape and Reel		M16.3
ICL232CBEZ (See Note)	0 to 70	16 Ld SOIC (Pb-free)	M16.3
ICL232CBEZT (See Note)	16 Ld SOIC Tape and Reel (Pb-free)		M16.3
ICL232IPE	-40 to 85	16 Ld PDIP	E16.3
ICL232IBE	-40 to 85	16 Ld SOIC	M16.3
ICL232IBET	16 Ld SOIC Tape and Reel		M16.3
ICL232MJE	-55 to 125	16 Ld CERDIP	F16.3

NOTE: Intersil Pb-free plus anneal products employ special Pb-free material sets; molding compounds/die attach materials and 100% matte tin plate termination finish, which are RoHS compliant and compatible with both SnPb and Pb-free soldering operations. Intersil Pb-free products are MSL classified at Pb-free peak reflow temperatures that meet or exceed the Pb-free requirements of IPC/JEDEC J STD-020.

Features

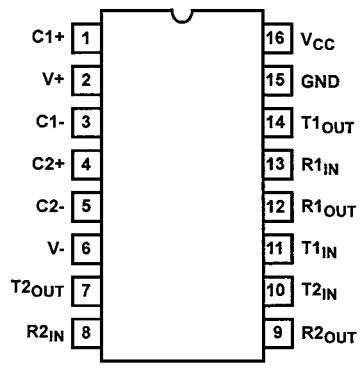
- Meets All RS-232C and V.28 Specifications
- Requires Only Single +5V Power Supply
- Onboard Voltage Doubler/Inverter
- Low Power Consumption
- 2 Drivers
  - ±9V Output Swing for +5V Input
  - 300Ω Power-off Source Impedance
  - Output Current Limiting
  - TTL/CMOS Compatible
  - 30V/µs Maximum Slew Rate
- 2 Receivers
  - ±30V Input Voltage Range
  - 3kΩ to 7kΩ Input Impedance
  - 0.5V Hysteresis to Improve Noise Rejection
- All Critical Parameters are Guaranteed Over the Entire Commercial, Industrial and Military Temperature Ranges
- Pb-Free Plus Anneal Available (RoHS Compliant)

Applications

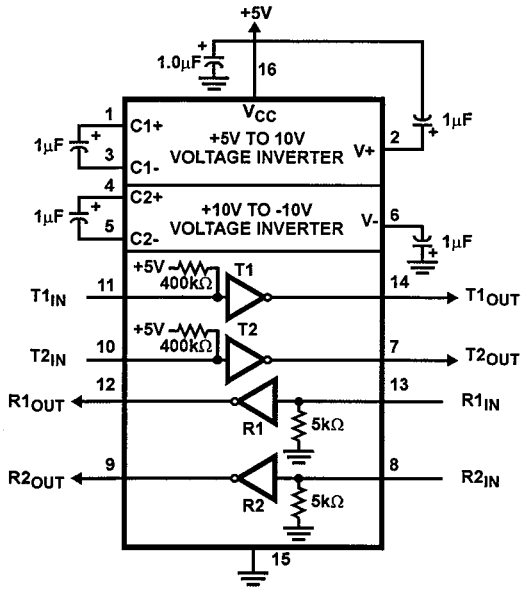
- Any System Requiring RS-232 Communications Port
  - Computer - Portable and Mainframe
  - Peripheral - Printers and Terminals
  - Portable Instrumentation
  - Modems
- Dataloggers

Pinout

ICL232 (PDIP, CERDIP, SOIC)
TOP VIEW



Functional Diagram



Absolute Maximum Ratings

V_{CC} to Ground . . . . . (GND -0.3V) < V_{CC} < 6V
V+ to Ground . . . . . (V_{CC} -0.3V) < V+ < 12V
V- to Ground . . . . . -12V < V- < (GND +0.3V)
Input Voltages
$T1_{IN}$, $T2_{IN}$ . . . . . (V_{-} -0.3V) < V_{IN} < (V_{+} +0.3V)
$R1_{IN}$, $R2_{IN}$ . . . . . ±30V
Output Voltages
$T1_{OUT}$, $T2_{OUT}$ . . . . . (V_{-} -0.3V) < V_{TXOUT} < (V_{+} +0.3V)
$R1_{OUT}$, $R2_{OUT}$ . . . . . (GND -0.3V) < V_{RXOUT} < (V_{CC} +0.3V)
Short Circuit Duration
$T1_{OUT}$, $T2_{OUT}$ . . . . . Continuous
$R1_{OUT}$, $R2_{OUT}$ . . . . . Continuous

Operating Conditions

Temperature Ranges
ICL232C . . . . . 0°C to 70°C
ICL232I . . . . . -40°C to 85°C
ICL232M . . . . . -55°C to 125°C

Thermal Information

Thermal Resistance (Typical, Note 1)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
CERDIP Package	80	18
PDIP Package	100	N/A
SOIC Package	100	N/A

Maximum Junction Temperature
Plastic Packages . . . . . 150°C
Ceramic Package . . . . . 175°C
Maximum Storage Temperature Range . . . . . -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) . . . . . 300°C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

1. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

Electrical Specifications

Test Conditions: V_{CC} = +5V ±10%, T_A = Operating Temperature Range. Test Circuit as in Figure 8 Unless Otherwise Specified

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Transmitter Output Voltage Swing, T_{OUT}	$T1_{OUT}$ and $T2_{OUT}$ Loaded with 3kΩ to Ground	±5	±9	±10	V
Power Supply Current, I_{CC}	Outputs Unloaded, T_A = 25°C	-	5	10	mA
T_{IN}, Input Logic Low, V_{IL}		-	-	0.8	V
T_{IN}, Input Logic High, V_{IH}		2.0	-	-	V
Logic Pullup Current, I_P	$T1_{IN}$, $T2_{IN}$ = 0V	-	15	200	µA
RS-232 Input Voltage Range, V_{IN}		-30	-	+30	V
Receiver Input Impedance, R_{IN}	V_{IN} = ±3V	3.0	5.0	7.0	kΩ
Receiver Input Low Threshold, V_{IN} (H-L)	V_{CC} = 5V, T_A = 25°C	0.8	1.2	-	V
Receiver Input High Threshold, V_{IN} (L-H)	V_{CC} = 5V, T_A = 25°C	-	1.7	2.4	V
Receiver Input Hysteresis, V_{HYST}		0.2	0.5	1.0	V
TTL/CMOS Receiver Output Voltage Low, V_{OL}	I_{OUT} = 3.2mA	-	0.1	0.4	V
TTL/CMOS Receiver Output Voltage High, V_{OH}	I_{OUT} = -1.0mA	3.5	4.6	-	V
Propagation Delay, t_{PD}	RS-232 to TTL	-	0.5	-	µs
Instantaneous Slew Rate, SR	C_L = 10pF, R_L = 3kΩ, T_A = 25°C (Notes 2, 3)	-	-	30	V/µs
Transition Region Slew Rate, SR_T	R_L = 3kΩ, C_L = 2500pF Measured from +3V to -3V or -3V to +3V	-	3	-	V/µs
Output Resistance, R_{OUT}	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V	300	-	-	Ω
RS-232 Output Short Circuit Current, I_{SC}	$T1_{OUT}$ or $T2_{OUT}$ Shorted to GND	-	±10	-	mA

NOTES:

2. Guaranteed by design.
3. See Figure 4 for definition.

Test Circuits

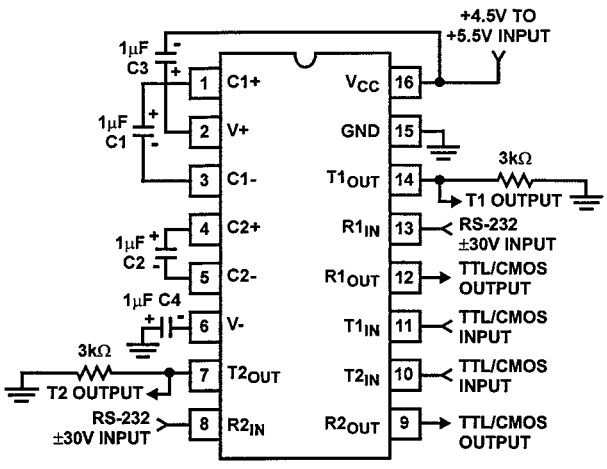


FIGURE 1. GENERAL TEST CIRCUIT

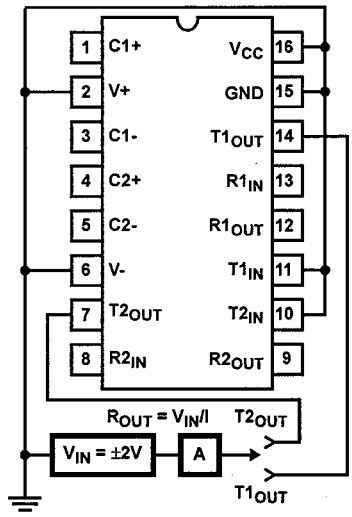


FIGURE 2. POWER-OFF SOURCE RESISTANCE CONFIGURATION

Typical Performance Curves

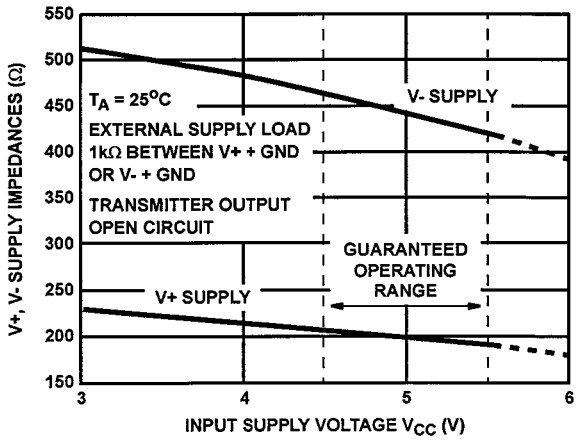


FIGURE 3. V+, V- OUTPUT IMPEDANCES vs V_{CC}

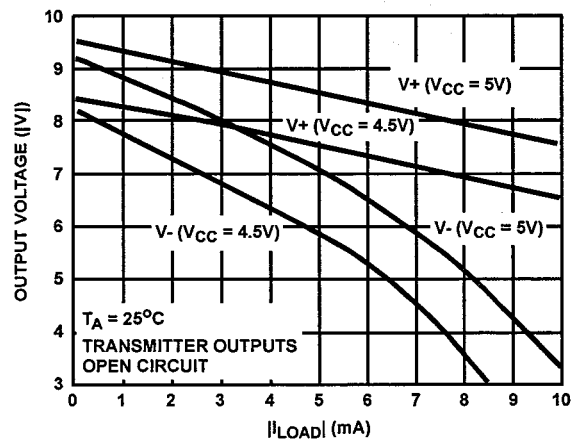


FIGURE 4. V+, V- OUTPUT VOLTAGES vs LOAD CURRENT

Pin Descriptions

PDIP, CERDIP	SOIC	PIN NAME	DESCRIPTION
1	1	C1+	External capacitor "+" for internal voltage doubler.
2	2	V+	Internally generated +10V (typical) supply.
3	3	C1-	External capacitor "-" for internal voltage doubler.
4	4	C2+	External capacitor "+" internal voltage inverter.
5	5	C2-	External capacitor "-" internal voltage inverter.
6	6	V-	Internally generated -10V (typical) supply.
7	7	$T2_{OUT}$	RS-232 Transmitter 2 output ±10V (typical).
8	8	$R2_{IN}$	RS-232 Receiver 2 input, with internal 5K pulldown resistor to GND.

Pin Descriptions (Continued)

PDIP, CERDIP	SOIC	PIN NAME	DESCRIPTION
9	9	R2out	Receiver 2 TTL/CMOS output.
10	10	$T2_{IN}$	Transmitter 2 TTL/CMOS input, with internal 400K pullup resistor to V_{CC}.
11	11	$T1_{IN}$	Transmitter 1 TTL/CMOS input, with internal 400K pullup resistor to V_{CC}.
12	12	$R1_{OUT}$	Receiver 1 TTL/CMOS output.
13	13	$R1_{IN}$	RS-232 Receiver 1 input, with internal 5K pulldown resistor to GND.
14	14	$T1_{OUT}$	RS-232 Transmitter 1 output ±10V (typical).
15	15	GND	Supply Ground.
16	16	V_{CC}	Positive Power Supply +5V ±10%

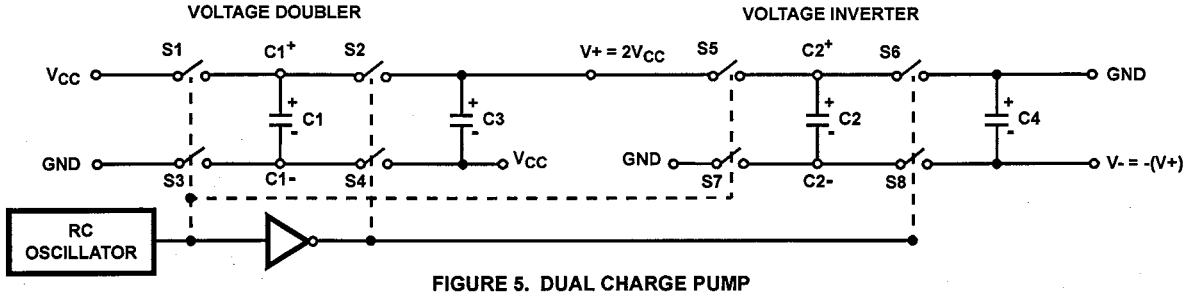


FIGURE 5. DUAL CHARGE PUMP

Detailed Description

The ICL232 is a dual RS-232 transmitter/receiver powered by a single +5V power supply which meets all EIA RS232C specifications and features low power consumption. The functional diagram illustrates the major elements of the ICL232. The circuit is divided into three sections: a voltage doubler/inverter, dual transmitters, and dual receivers Voltage Converter.

An equivalent circuit of the dual charge pump is illustrated in Figure 5.

The voltage quadrupler contains two charge pumps which use two phases of an internally generated clock to generate +10V and -10V. The nominal clock frequency is 16kHz. During phase one of the clock, capacitor C1 is charged to V_{CC}. During phase two, the voltage on C1 is added to V_{CC}, producing a signal across C2 equal to twice V_{CC}. At the same time, C3 is also charged to $2V_{CC}$, and then during phase one, it is inverted with respect to ground to produce a signal across C4 equal to $-2V_{CC}$. The voltage converter accepts input voltages up to 5.5V. The output impedance of the doubler (V+) is approximately 200Ω, and the output impedance of the inverter (V-) is approximately 450Ω. Typical graphs are presented which show the voltage converters output vs input voltage and output voltages vs load characteristics. The test circuit (Figure 3) uses 1µF capacitors for C1-C4, however, the value is not critical. Increasing the values of C1 and C2 will lower the output impedance of the voltage doubler and inverter, and increasing the values of the reservoir capacitors, C3 and C4, lowers the ripple on the V+ and V- supplies.

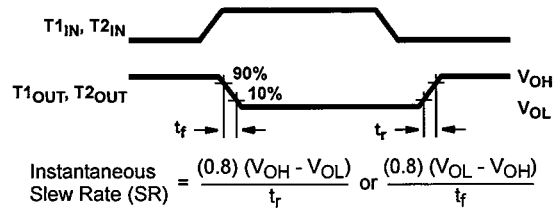


FIGURE 6. SLEW RATE DEFINITION

Transmitters

The transmitters are TTL/CMOS compatible inverters which translate the inputs to RS-232 outputs. The input logic threshold is about 26% of V_{CC}, or 1.3V for V_{CC} = 5V. A logic 1 at the input results in a voltage of between -5V and V- at the output, and a logic 0 results in a voltage between +5V and (V+ - 0.6V). Each transmitter input has an internal 400kΩ pullup resistor so any unused input can be left unconnected and its output remains in its low state. The output voltage swing meets the RS-232C specification of ±5V minimum with the worst case conditions of: both transmitters driving 3kΩ minimum load impedance, V_{CC} = 4.5V, and maximum allowable operating temperature. The transmitters have an internally limited output slew rate which is less than 30V/µs. The outputs are short circuit protected and can be shorted to ground indefinitely. The powered down output impedance is a

minimum of 300Ω with ±2V applied to the outputs and V_{CC} = 0V.

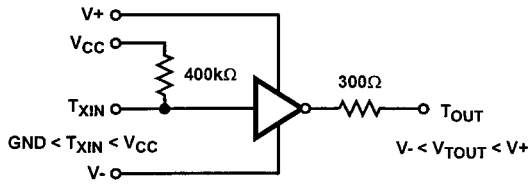


FIGURE 7. TRANSMITTER

Receivers

The receiver inputs accept up to ±30V while presenting the required 3kΩ to 7kΩ input impedance even it the power is off (V_{CC} = 0V). The receivers have a typical input threshold of 1.3V which is within the ±3V limits, known as the transition region, of the RS-232 specification. The receiver output is 0V to V_{CC}. The output will be low whenever the input is greater than 2.4V and high whenever the input is floating or driven between +0.8V and -30V. The receivers feature 0.5V hysteresis to improve noise rejection.

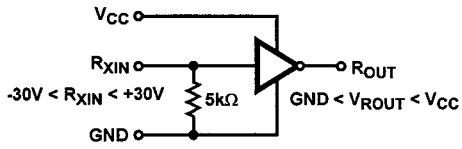
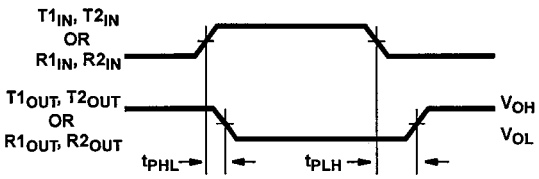


FIGURE 8. RECEIVER



$$\text{Average Propagation Delay} = \frac{t_{PHL} + t_{PLH}}{2}$$

FIGURE 9. PROPAGATION DELAY DEFINITION

Applications

The ICL232 may be used for all RS-232 data terminal and communication links. It is particularly useful in applications where ±12V power supplies are not available for conventional RS-232 interface circuits. The applications presented represent typical interface configurations.

A simple duplex RS-232 port with CTS/RTS handshaking is illustrated in Figure 10. Fixed output signals such as DTR (data terminal ready) and DSRS (data signaling rate select) is generated by driving them through a 5kΩ resistor connected to V+.

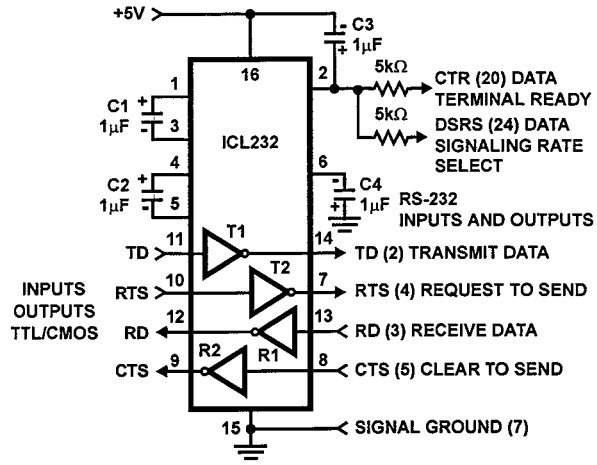


FIGURE 10. SIMPLE DUPLEX RS-232 PORT WITH CTS/RTS HANDSHAKING

In applications requiring four RS-232 inputs and outputs (Figure 11), note that each circuit requires two charge pump capacitors (C1 and C2) but can share common reservoir capacitors (C3 and C4). The benefit of sharing common reservoir capacitors is the elimination of two capacitors and the reduction of the charge pump source impedance which effectively increases the output swing of the transmitters.

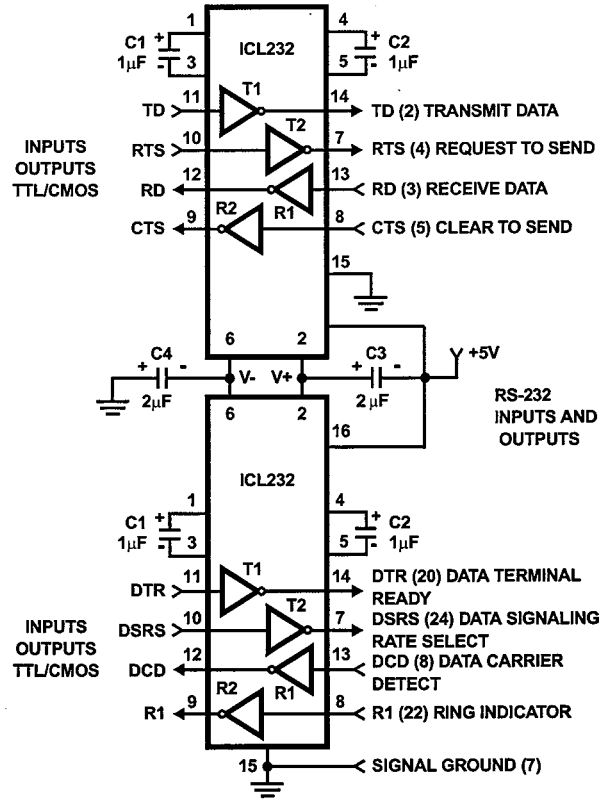
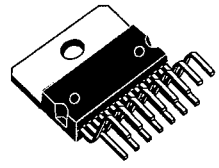


FIGURE 11. COMBINING TWO ICL232s FOR 4 PAIRS OF RS-232 INPUTS AND OUTPUTS

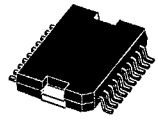
L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15



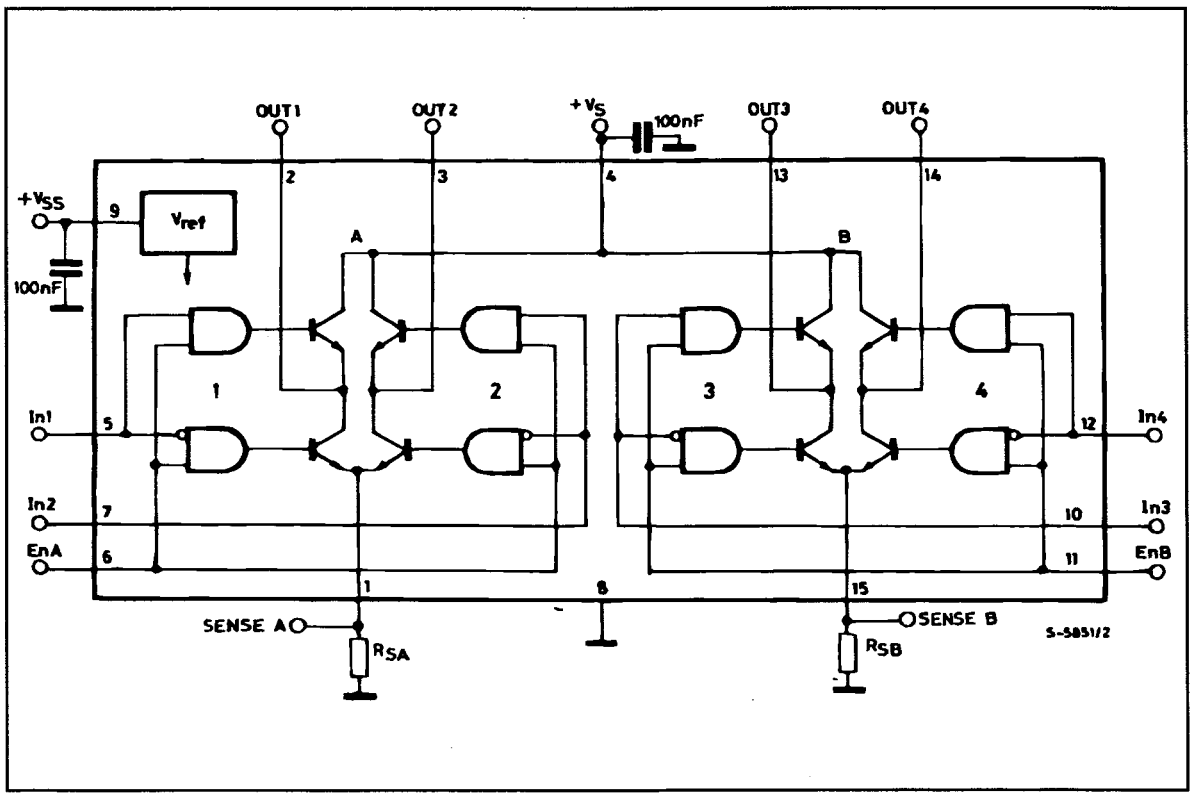
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

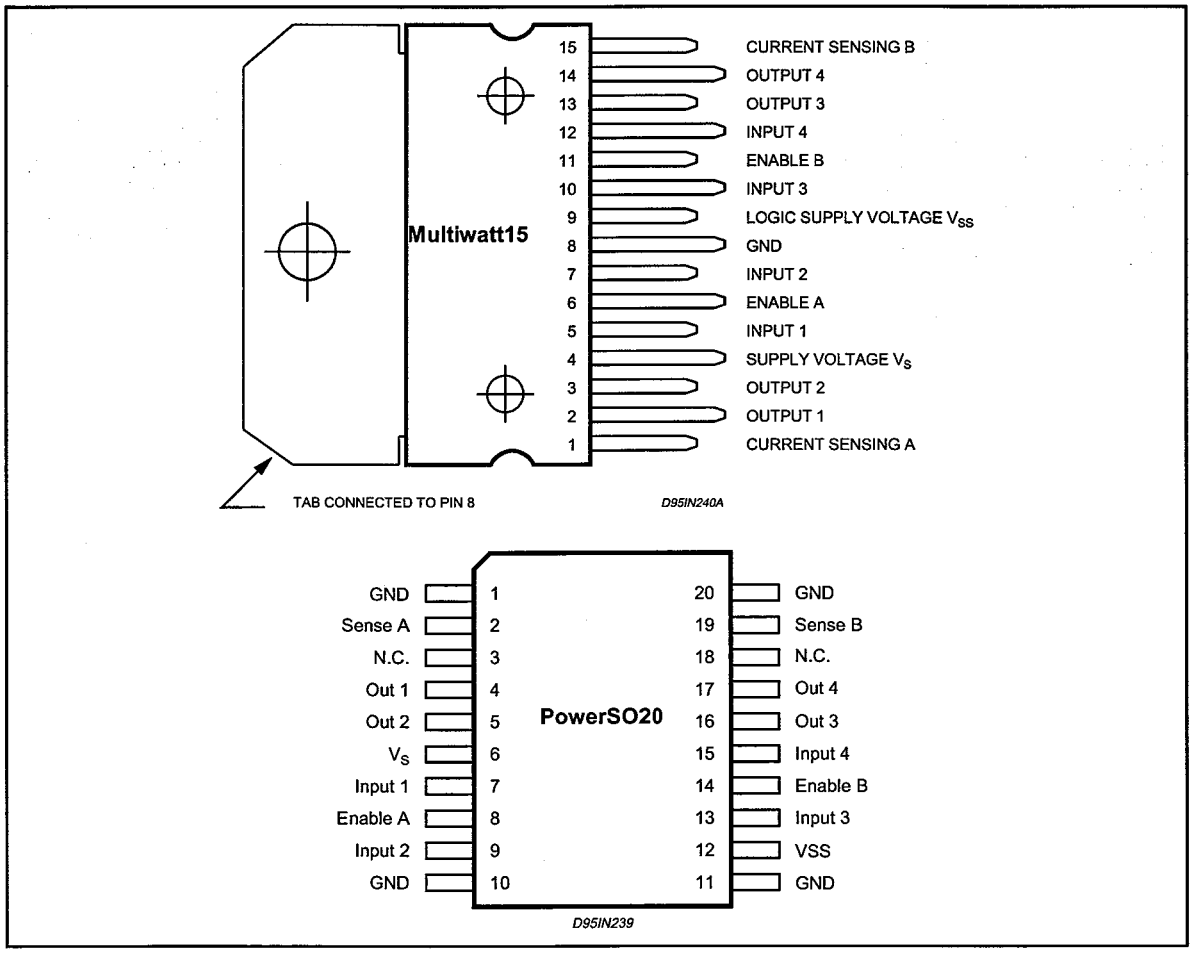
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

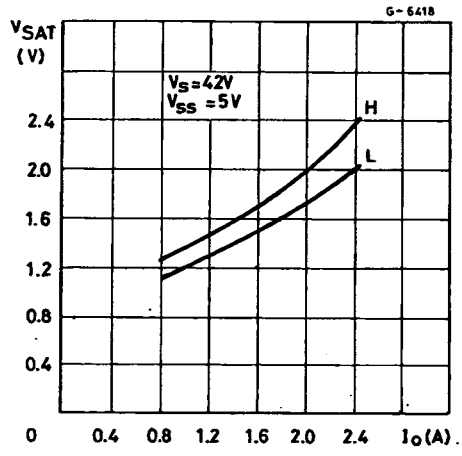
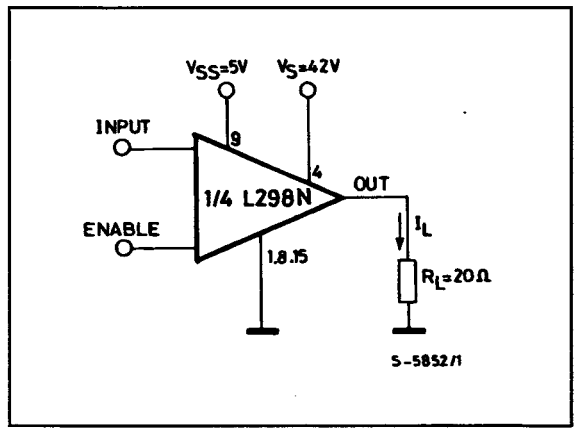


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

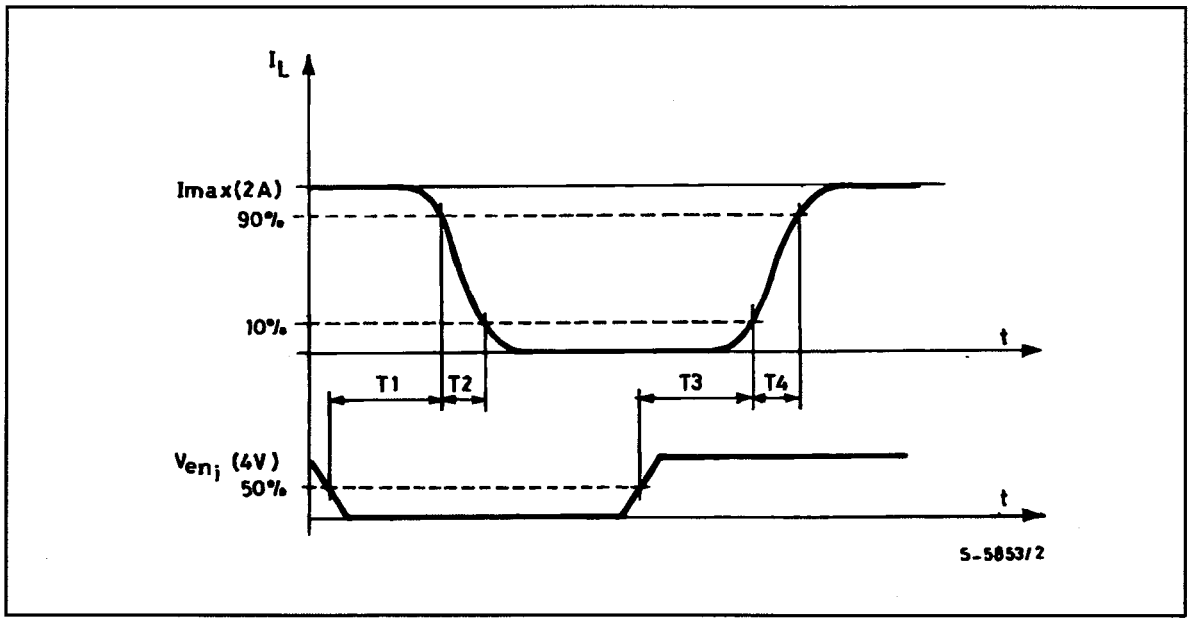
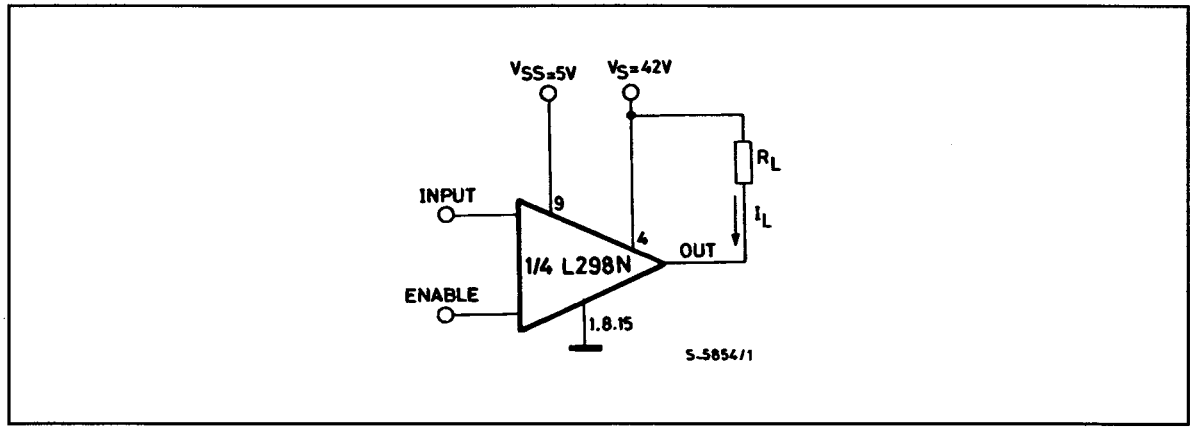


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

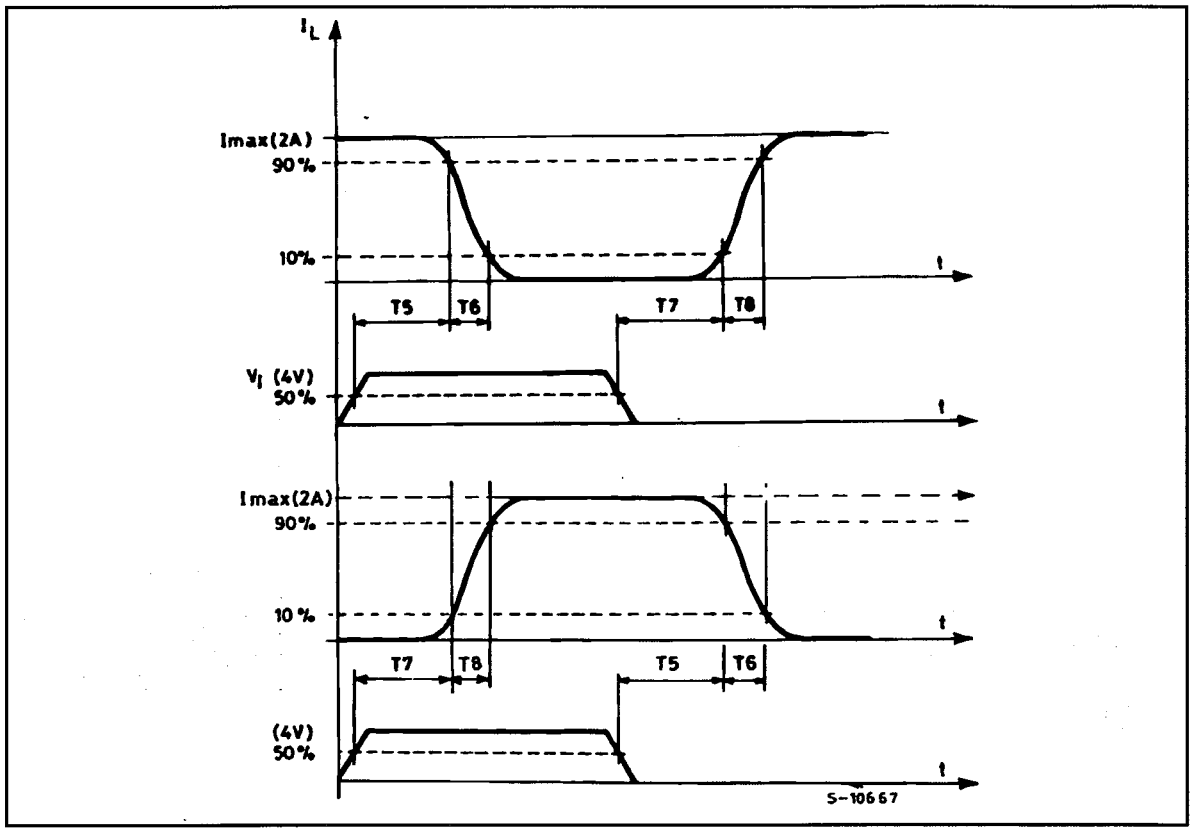
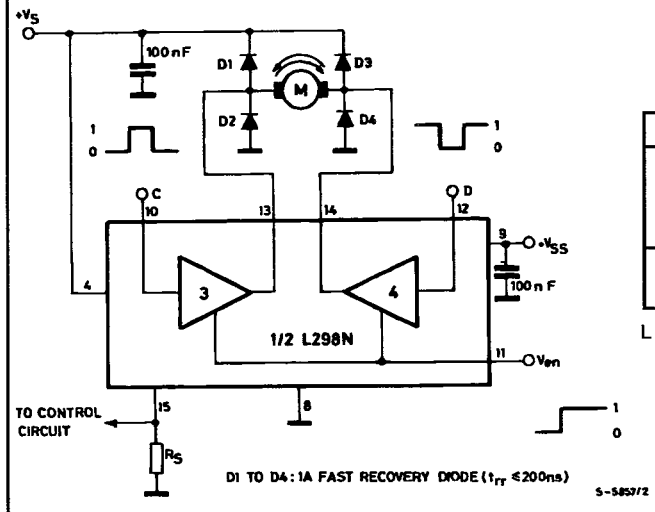


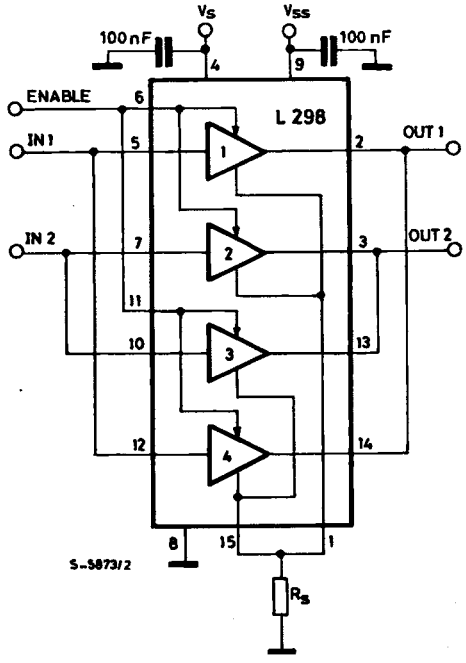
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements ($t_{rr} \leq 200$ nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped ; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

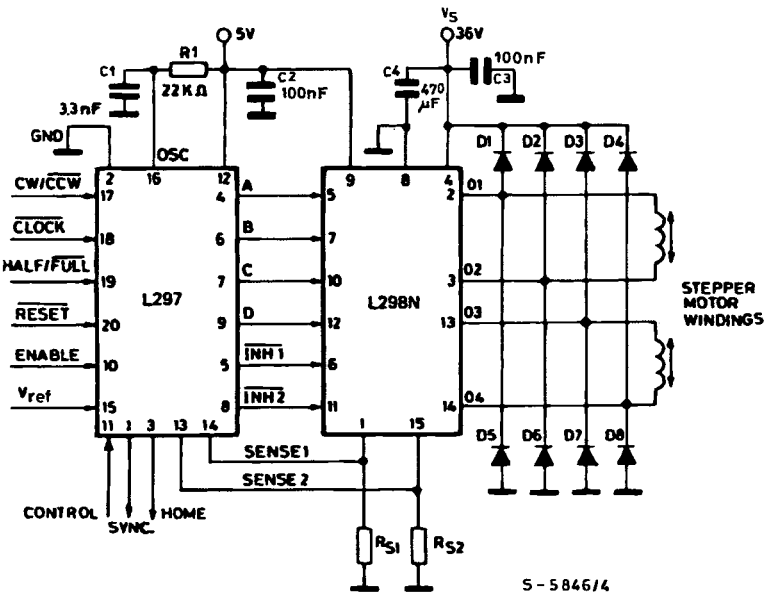
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2$ V @ I = 2 A
trr ≤ 200 ns

Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

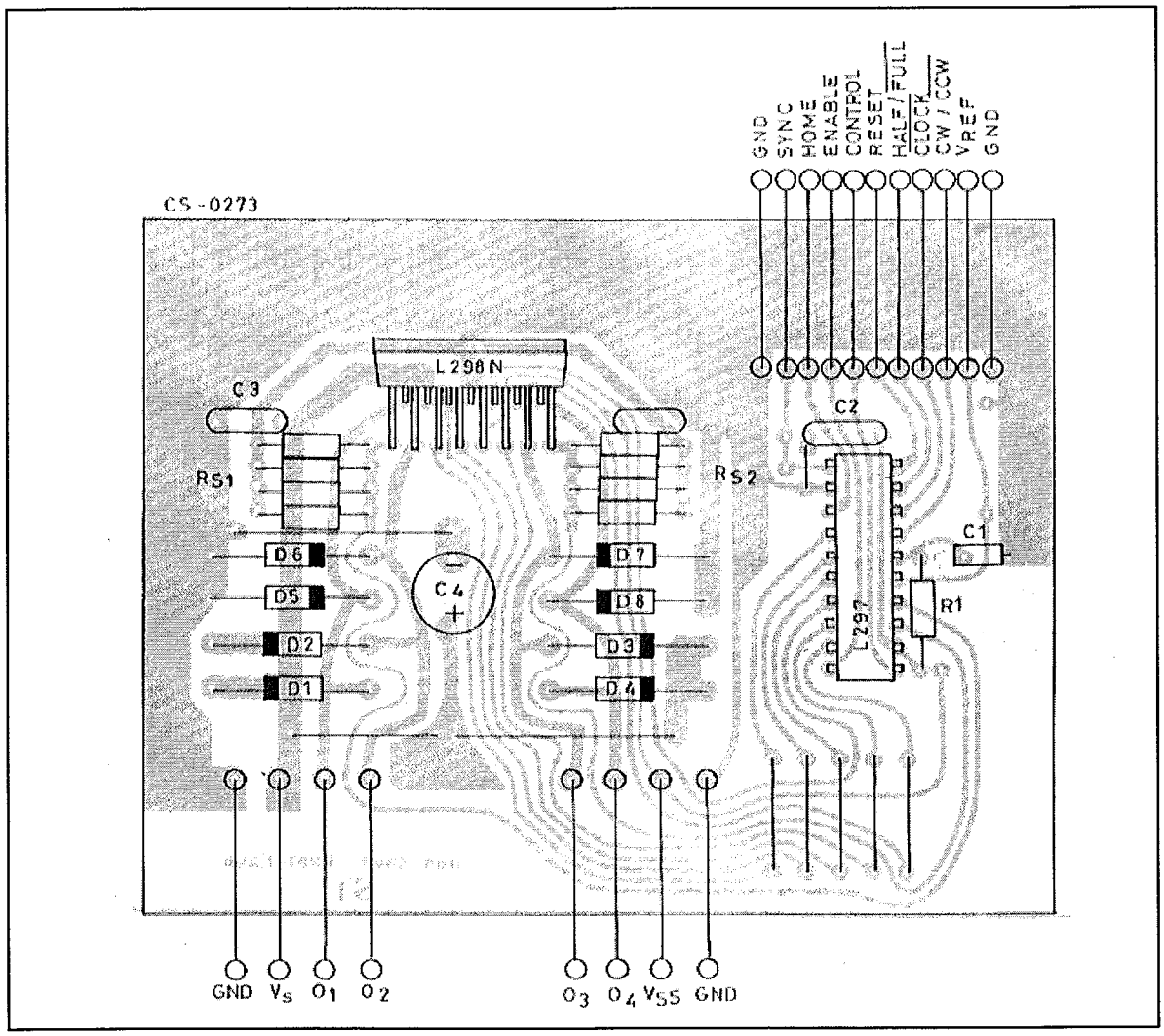
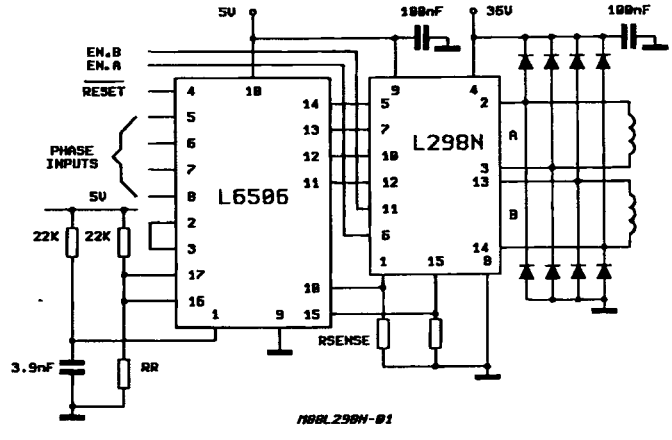


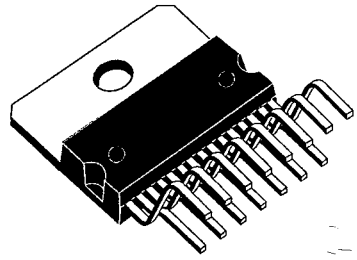
Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



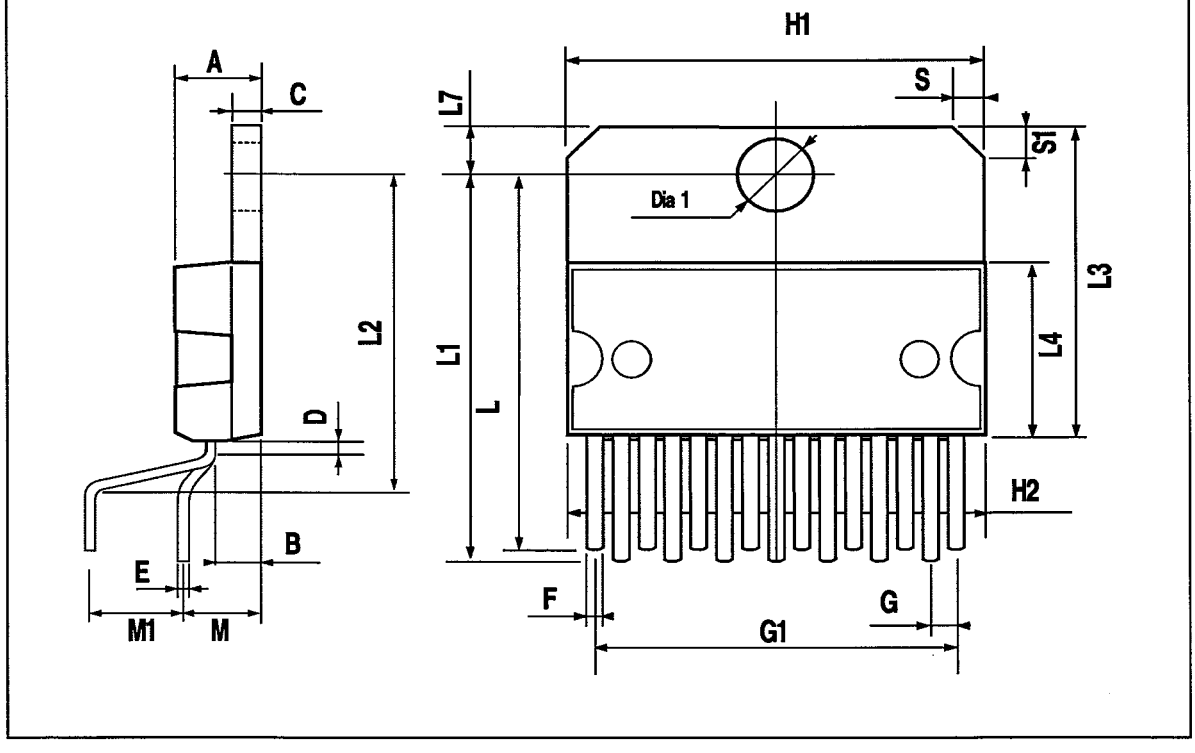
R_R and R_{sense} depend from the load current

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

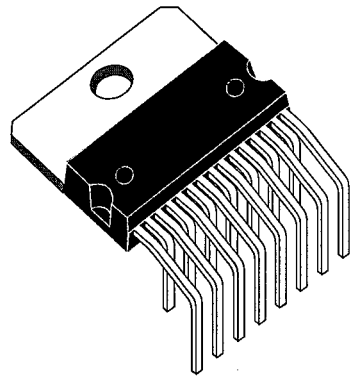


Multiwatt15 V

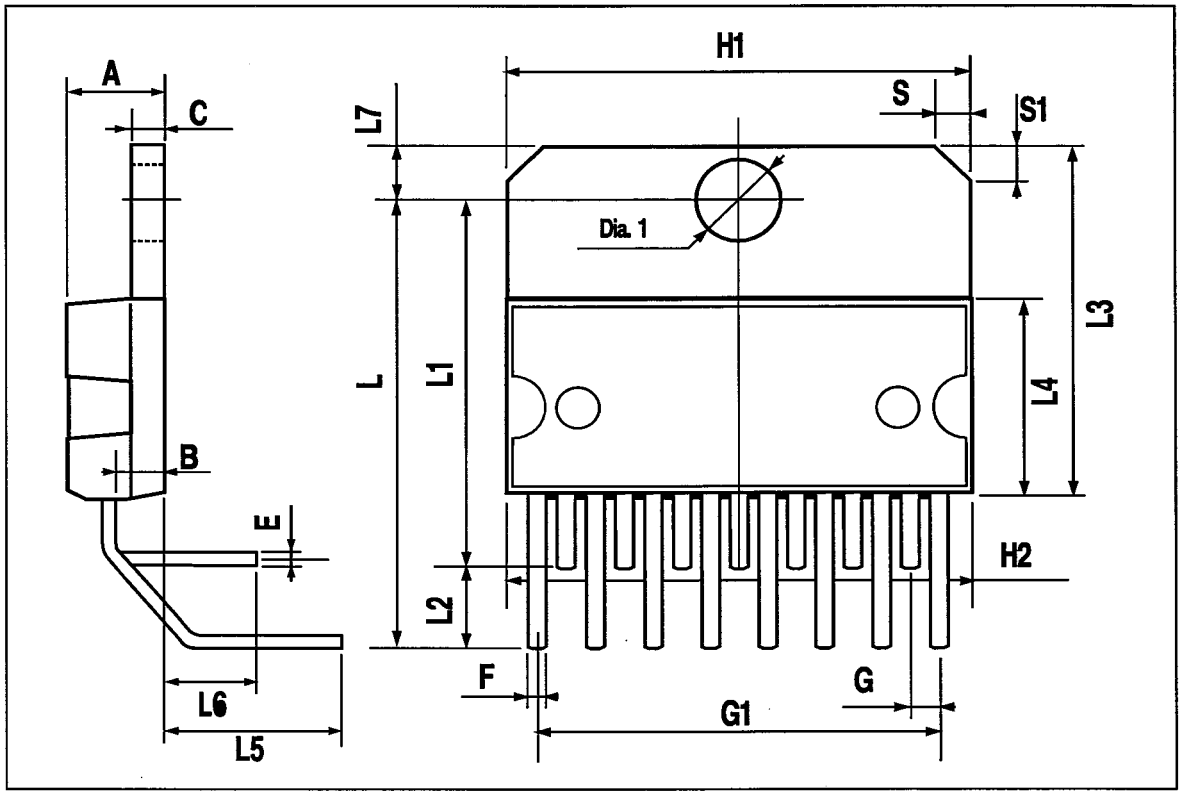


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L		20.57			0.810	
L1		18.03			0.710	
L2		2.54			0.100	
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L5		5.28			0.208	
L6		2.38			0.094	
L7	2.65		2.9	0.104		0.114
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



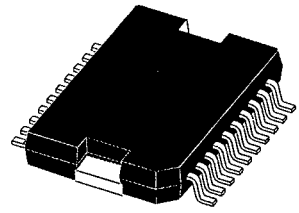
Multiwatt15 H



DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			3.6			0.142
a1	0.1		0.3	0.004		0.012
a2			3.3			0.130
a3	0		0.1	0.000		0.004
b	0.4		0.53	0.016		0.021
c	0.23		0.32	0.009		0.013
D (1)	15.8		16	0.622		0.630
D1	9.4		9.8	0.370		0.386
E	13.9		14.5	0.547		0.570
e		1.27			0.050	
e3		11.43			0.450	
E1 (1)	10.9		11.1	0.429		0.437
E2			2.9			0.114
E3	5.8		6.2	0.228		0.244
G	0		0.1	0.000		0.004
H	15.5		15.9	0.610		0.626
h			1.1			0.043
L	0.8		1.1	0.031		0.043
N	10° (max.)					
S	8° (max.)					
T		10			0.394	

(1) "D and F" do not include mold flash or protrusions.
- Mold flash or protrusions shall not exceed 0.15 mm (0.006").
- Critical dimensions: "E", "G" and "a3"

OUTLINE AND MECHANICAL DATA



JEDEC MO-166

PowerSO20