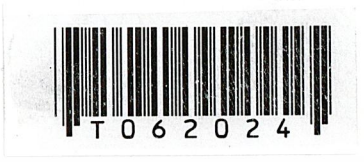
ระบบควบคุมโดยใช้ PIC ไมโครคอนโทรลเลอร์

PIC MICROCONTROLLER BASED CONTROL SYSTEMS



โดย

นาย เทพ จารุรัตน์มงคล

นาย ประทีป วงศ์จิรชยา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์เชิงกล

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ระบบควบคุมโดยใช้ PIC ไมโครคอนโทรลเลอร์

PIC MICROCONTROLLER BASED CONTROL SYSTEMS

โดย

นาย เทพ จารุรัตน์มงคล 44010186

นาย ประทีป วงศ์จิรชยา 44010282

อาจารย์ที่ปรึกษา

ดร. ชนินทร์ บุญลักษณานุสรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์เชิงกล

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์เชิงกล ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบควบคุมโดยใช้ PIC ไมโครคอนโทรลเลอร์**

PIC MICROCONTROLLER BASED CONTROL SYSTEMS

ผู้จัดทำ

1. นาย เทพ จารุรัตน์มงคล 44010186

2. นาย ประทีป วงศ์จิรชยา 44010282

..... อาจารย์ที่ปรึกษา

(ดร. ชนินทร์ บุญฉักษณานุสรณ์)

ระบบควบคุมโดยใช้ PIC ไมโครคอนโทรลเลอร์

นาย เทพ จารุรัตน์มงคล 44010186
นาย ประทีป วงศ์จิรชยา 44010282
ดร. ชนินทร์ บุญลักษณานุสรณ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอ การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล PIC 16F877 โดยจัดทำเป็นชุดทดลองเพื่อการเรียนรู้ ซึ่งจะศึกษาตั้งแต่โครงสร้างสถาปัตยกรรม ชุดคำสั่ง และการโปรแกรมข้อมูล ของไมโครคอนโทรลเลอร์ รวมถึงการเชื่อมต่อกับอุปกรณ์ อินพุต และเอาต์พุตภายนอก โดยออกแบบและจัดทำเป็นชุดปฏิบัติการในรูปแบบต่างๆ

PIC MICROCONTROLLER BASED CONTROL SYSTEMS

Mr. Thep Jaruratmongkol 44010186
Mr. Prateep Wongjirachaya 44010282
Dr. Chanin Bunlaksananusorn Advisor
2004

ABTRACT

This thesis presents development of the PIC microcontroller based control systems. Important features, instruction sets, and programming of the microcontroller are described, as well as the design of peripheral boards. The interface between the microcontroller and these peripheral boards allows several control tasks to be executed, making it suitable as a learning tool for mechatronics.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

การประยุกต์ใช้ไมโครคอนโทรลเลอร์ ในงานต่างๆ ปัจจุบันได้รับความนิยมเป็นอย่างมาก ไมโครคอนโทรลเลอร์ชนิดและตระกูลต่างๆ ถูกใช้งานกันอย่างแพร่หลาย หนึ่งในนั้น คือ ไมโครคอนโทรลเลอร์ PIC จาก บริษัทไมโครชิพ (Microchip) ซึ่งได้รับความนิยมอย่างสูง ทั้งนี้ด้วยประสิทธิภาพการทำงานที่ดี และมีทรัพยากรภายในที่ครบถ้วนในตัวมันเอง ไม่ว่าจะเป็น Program Memory, RAM, EEPROM, PWM Module, A/D Converter, โมดูลสื่อสารข้อมูลอนุกรม ฯลฯ โดยไม่จำเป็นต้องต่ออุปกรณ์เสริมจากภายนอก โดยในตัว PIC จะมีฟังก์ชันที่ใช้ประมวลผล รวมทั้งหน่วยความจำ ซึ่งทำให้มันมีความสามารถคล้ายกับเป็น CPU ขนาดเล็กที่มีความสามารถในหลายๆ ด้าน

1.2 ขอบเขตของปริญญานิพนธ์

โครงงานนี้จะเป็นการศึกษา เรียนรู้ เพื่อใช้งานไมโครคอนโทรลเลอร์ตระกูล PIC 16F877 เป็นหลัก โดยจะศึกษาตั้งแต่โครงสร้างสถาปัตยกรรม ชุดคำสั่ง และทดลองใช้งาน โดยออกแบบเป็นชุดปฏิบัติการในการควบคุมระบบพื้นฐานต่างๆ

1.3 วิธีการดำเนินงาน

เริ่มจากการศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้อง ได้แก่ ศึกษาการทำงานของไมโครคอนโทรลเลอร์ PIC 16F877 มอเตอร์ไฟฟ้ากระแสตรง รีเลย์ เมตริกซ์สวิตช์ ตัวแสดงผล LED 7 ส่วน และศึกษาการเขียนโปรแกรมภาษา Assembly เพื่อควบคุม PIC 16F877

จากนั้นก็ออกแบบ สร้าง และทดลองวงจรที่ใช้ในการลงโปรแกรม ควบคุม แสดงผล และชุดปฏิบัติการพื้นฐานต่างๆ

1.4 วัตถุประสงค์และประโยชน์ที่คาดว่าจะได้รับ

1. เพื่อศึกษาโครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์ตระกูล PIC16F877
2. เพื่อศึกษาชุดคำสั่งการใช้งานและการเชื่อมต่ออุปกรณ์พื้นฐานของไมโครคอนโทรลเลอร์ ตระกูล PIC16F877
3. เพื่อออกแบบและจัดทำชุดปฏิบัติการที่ควบคุมโดยไมโครคอนโทรลเลอร์ตระกูล PIC16F877

4. สามารถเขียนโปรแกรมสั่งงานไมโครคอนโทรลเลอร์ตระกูล PIC16F877 เพื่อควบคุมการทำงานของชุดปฏิบัติการได้ตามต้องการ
5. สามารถนำความรู้ที่ได้ไปประยุกต์ใช้กับไมโครคอนโทรลเลอร์ตระกูล PIC เบอร์อื่นๆ ตลอดจน เข้าใจการทำงานพื้นฐานของไมโครคอนโทรลเลอร์ตระกูลอื่นๆ

บทที่ 2
ทฤษฎีและหลักการ

ระบบควบคุมโดยใช้ PIC นั้นประกอบด้วย 2 ส่วนหลัก คือ ส่วนฮาร์ดแวร์ (Hardware) และส่วนซอฟต์แวร์ (Software) โดยทั้งสองส่วนนี้ต้องอาศัยทฤษฎีและหลักการที่เกี่ยวข้องดังนี้

1. โครงสร้าง สถาปัตยกรรม ชุดคำสั่งการใช้งาน และการเชื่อมต่อกับอุปกรณ์ภายนอกของไมโครคอนโทรลเลอร์ PIC 16F877
2. โครงสร้างและการทำงานของรีเลย์
3. โครงสร้าง การทำงาน และการควบคุม มอเตอร์ไฟฟ้ากระแสตรง

2.1 ไมโครคอนโทรลเลอร์ PIC 16F877

ในปัจจุบัน ไมโครคอนโทรลเลอร์ตระกูล PIC มีการพัฒนาและผลิตออกมาหลายเบอร์ โดยทั่วไปจะแบ่งออกเป็น 3 แบบ คือ

1. สถาปัตยกรรมแบบ 12 – Bit Core (Base-Line) เป็นกลุ่มของไมโครคอนโทรลเลอร์ที่มีขนาดเล็กมีโครงสร้างของคำสั่งเพียง 12 bits และค่อนข้างมีข้อจำกัดในการใช้งาน เนื่องจากมีหน่วยความจำRAM และ STACK ค่อนข้างจำกัด
2. สถาปัตยกรรมแบบ 14 – Bit Core (Mid-Range) เป็นกลุ่มของไมโครคอนโทรลเลอร์ที่มีขนาดกลางมีโครงสร้างของคำสั่ง 14 bits มีทั้งแบบที่โปรแกรมได้ครั้งเดียว (OTP : One Time Programmable) และแบบแฟลช (Flash Memory)
3. สถาปัตยกรรมแบบ 16 – Bit Core (High – End) เป็นกลุ่มของไมโครคอนโทรลเลอร์ที่อยู่ในกลุ่มระดับสูงซึ่งได้มีการพัฒนาโครงสถาปัตยกรรม ทั้งในเรื่องของหน่วยความจำ ความเร็ว และ คุณสมบัติอื่นๆ ที่เหนือกว่าสองกลุ่มที่ผ่านมามาก มีการจัดวางหน่วยความจำโปรแกรมอยู่ในเพจ (Page) เดียวกัน ทำให้ไม่มีปัญหาเรื่องรอยต่อของหน่วยความจำ

นอกจากนี้แล้ว PIC ยังแบ่งออกเป็นประเภทของหน่วยความจำอีกด้วย โดยจะมีการจำแนกเป็น 3 ประเภท คือ

1. C เช่น PIC16CXXX คือ มีโครงสร้างหน่วยความจำเป็น EPROM จัดอยู่ในจำพวกอุปกรณ์ OTP แต่สามารถลบได้ด้วยแสง UV
2. CR เช่น PIC16CRXXX คือ มีโครงสร้างหน่วยความจำเป็น ROM จัดอยู่ในจำพวกอุปกรณ์ OTP ไม่สามารถลบได้

3. F เช่น PIC16FXXX คือ มีโครงสร้างหน่วยความจำเป็น FLASH Memory สามารถทำการโปรแกรม ลบ แล้ว โปรแกรมซ้ำได้หลายครั้ง

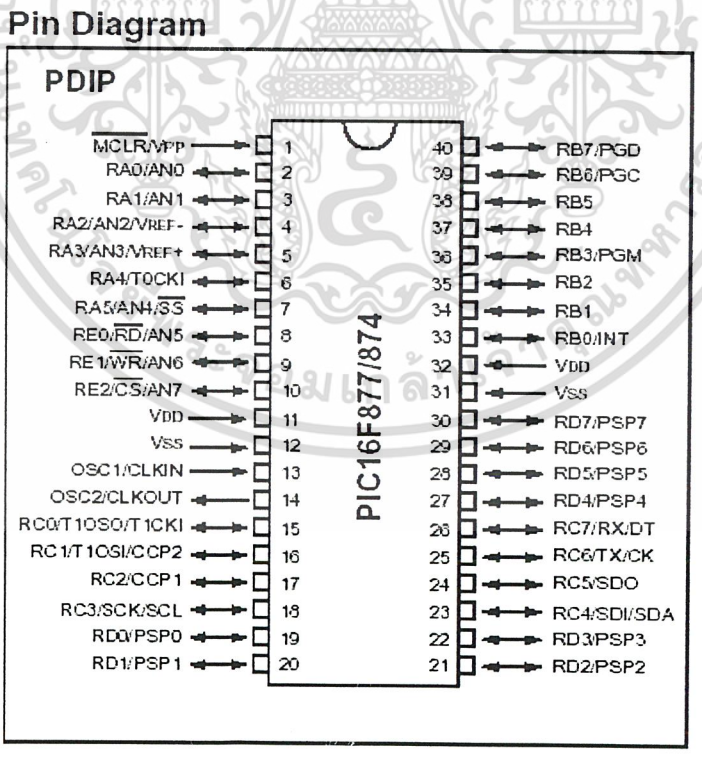
2.1.1 แรงดันในการทำงาน

ช่วงแรงดันการทำงานของ PIC โดยปกติมาตรฐานแล้วจะอยู่ระหว่าง 4.5 – 6.0 V แต่จะมีบางเบอร์ที่ออกแบบมาให้สามารถทำงานได้ในช่วงแรงดันต่ำ ประมาณ 2.5 -6 V ได้ ซึ่งจะมีการระบุโค้ดให้ทราบ โดยมีการเพิ่มรหัสตัว L เข้าไปในเบอร์ของอุปกรณ์ เช่น PIC16LFXXX เป็นต้น

2.1.2 สัญญาณนาฬิกา

PIC จะใช้สัญญาณนาฬิกา โดยมองเป็นลักษณะของ วงรอบ (Cycle) ซึ่งระบุเอาไว้ว่า 1 คำสั่งนั้นจะประกอบไปด้วย 1-2 วงรอบ โดยแต่ละวงรอบนั้นจะแบ่งเป็น 4 ส่วน คือ Q1, Q2, Q3 และ Q4 ด้วยเหตุนี้ ความเร็วโดยรวมของ PIC จึงเท่ากับ ค่าความถี่ของสัญญาณนาฬิกา หาร ด้วย 4

$$1cycle = Q_1 + Q_2 + Q_3 + Q_4 = \frac{XTAL}{4}$$



รูปที่ 2.1 การจัดขาของ PIC16F877 .

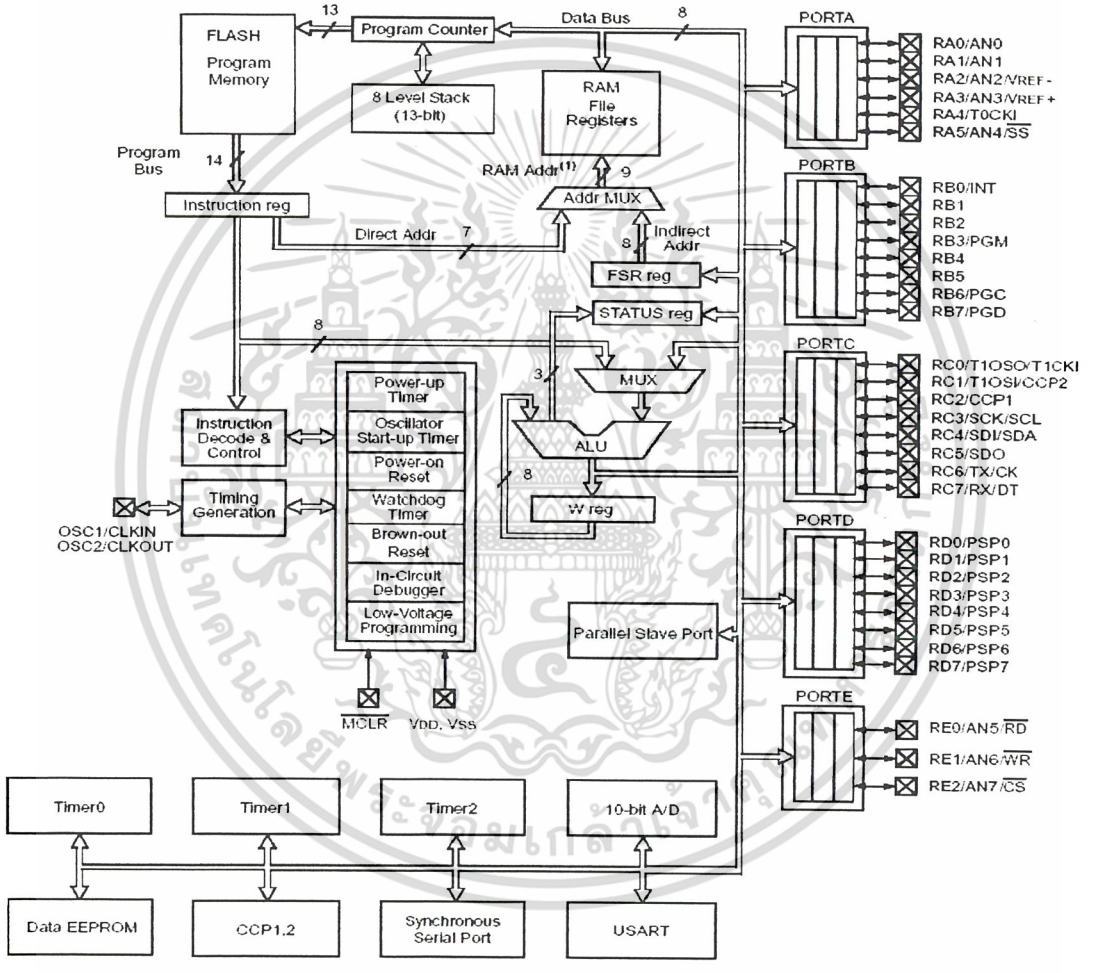
2.1.3 คุณสมบัติของ PIC16F877

- ซีพียูเป็นแบบ RISC (Reduced Instruction-Set) มีคำสั่งให้ใช้งาน 35 คำสั่ง
- คำสั่งหนึ่งๆใช้เวลาทำงาน 1 ถึง 2 Cycle
- ทำงานได้สูงสุดที่ 20MHz (PIC16F877-20/P)
- สัญญาณนาฬิกามีหลายโหมดให้เลือกใช้งาน
  - โหมด RC ใช้วงจรภายในไมโครคอนโทรลเลอร์กำเนิดสัญญาณนาฬิกา
  - โหมด LP ใช้คริสตอลพลังงานต่ำ ความถี่สูงสุดไม่เกิน 200 KHz
  - โหมด XT ใช้คริสตอลความถี่ตั้งแต่ 100 KHz – 4 MHz
  - โหมด HS ใช้คริสตอลความถี่สูง สูงสุดไม่เกิน 20 MHz
- มี I/O พอร์ตทั้งหมด 5 พอร์ต [PORT A (6 bits), B (8 bits), C(8 bits), D(8 bits), E (3 bits)]
- หน่วยความจำโปรแกรมเป็นแบบ Flash มีขนาด 8KWord (1 word=14 bits)
- มี RAM ขนาด 368 bytes
- มี EEPROM ขนาด 256 bytes
- ทำงานแบบ Pipe-line ทำให้ ณ เวลาหนึ่งสามารถทำคำสั่ง Fetch และ Execute ได้พร้อมกัน
- ตอบสนองกับอินเตอร์รัพท์ได้ทั้งหมด 14 แหล่ง
- มี Stack ให้ใช้ได้สูงสุด 8 ระดับ
- มีระบบ POR (Power On Reset), PWRT (Power Up Timer), OST (Oscillator Start-up timer)
- มีระบบ WDT (Watchdog Timer)
- มีระบบ CP (Code Protection) และสามารถเลือกระดับการป้องกันได้
- มีโหมด ประหยัดพลังงาน
- สามารถโปรแกรมด้วยไฟ +5 VDC ได้
- สามารถโปรแกรมแบบ In-Circuit Serial Programming
- ทำงานที่ไฟเลี้ยง 2 VDC ถึง 5.5 VDC
- Current Sink และ Current Source ของ PORT อยู่ที่ 25 mA
- มี Timer/Counter 3 ตัว
- มีโมดูล Capture/Compare/PWM 2 ชุด
- มี A/D Converter แบบ 10 bits จำนวน 8 ช่องนำเข้า ในตัวเอง
- มีระบบ USART สำหรับการสื่อสารแบบอนุกรม

- มีระบบตรวจระดับไฟเลี้ยง BODEN (Brown-Out Detection) เพื่อสร้างสัญญาณรีเซต ซีพียู BOR (Brown-Out Reset)

2.1.4 โครงสร้างการทำงานภายในของ PIC16F877

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



รูปที่ 2.2 โครงสร้างการทำงานภายใน PIC16F877

ชื่อขา	ตำแหน่งขา	ชนิดของขา	ชนิดของวงจรบัฟเฟอร์	รายละเอียด
OSC1/CLKIN	13	I	ST/CMOS	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
				PORTA is a bi-directional I/O port.
RA0/AN0	2	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/SS/AN4	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
				PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs. RB0 can also be the external interrupt pin.
RB0/INT	33	I/O	TTL/ST	
RB1	34	I/O	TTL	
RB2	35	I/O	TTL	
RB3/PGM	36	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	I/O	TTL	Interrupt-on-change pin.
RB5	38	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	I/O	TTL/ST	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	I/O	TTL/ST	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
				PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I2C modes.
RC4/SDI/SDA	23	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I2C mode).
RC5/SDO	24	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.

ตารางที่ 2.1 รายละเอียดของขา PIC16F877

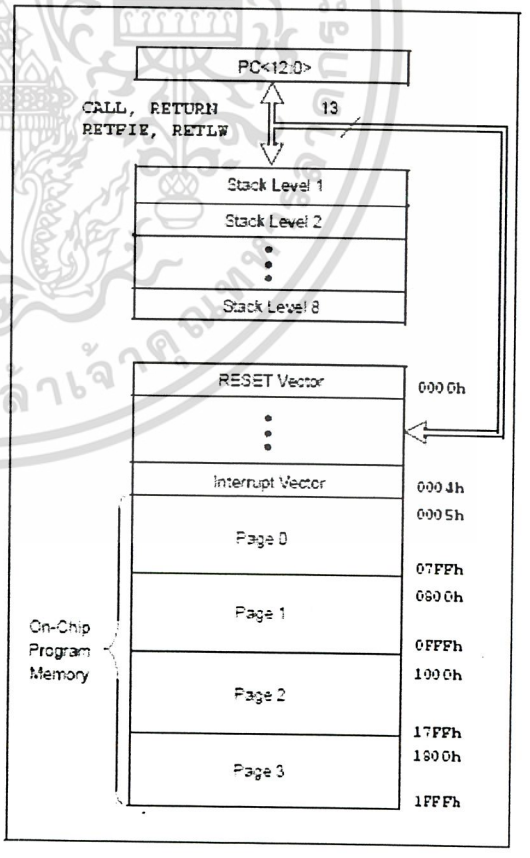
ชื่อขา	ตำแหน่งขา	ชนิดของขา	ชนิดของวงจรบัฟเฟอร์	รายละเอียด
RD0/PSP0 RD1/PSP1 RD2/PSP2 RD3/PSP3 RD4/PSP4 RD5/PSP5 RD6/PSP6 RD7/PSP7	19 20 21 22 27 28 29 30	I/O I/O I/O I/O I/O I/O I/O I/O	ST/TTL ST/TTL ST/TTL ST/TTL ST/TTL ST/TTL ST/TTL ST/TTL	PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RE0/RD/AN5 RE1/WR/AN6 RE2/CS/AN7	8 9 10	I/O I/O I/O	ST/TTL ST/TTL ST/TTL	PORTE is a bi-directional I/O port. RE0 can also be read control for the parallel slave port, or analog input5. RE1 can also be write control for the parallel slave port, or analog input6. RE2 can also be select control for the parallel slave port, or analog input7.
VSS	12 31	P	—	Ground reference for logic and I/O pins.
VDD	11 32	P	—	Positive supply for logic and I/O pins.
NC	—		—	These pins are not internally connected. These pins should be left unconnected.

ตารางที่ 2.1 (ต่อ) รายละเอียดของขา PIC16F877

2.1.5 การจัดสรรหน่วยความจำโปรแกรมของ PIC16F877

PIC16F877 จะมีขนาดของ Program Memory ซึ่งสามารถอ้างได้ถึง 8K byte โดย PIC16F877 จะมีขนาดหน่วยความจำเท่ากับ 8 K x 14 bits ซึ่งตำแหน่ง Reset Vector จะอยู่ที่ 0000h และ Interrupt Vector จะอยู่ที่ 0004h

PIC จะแบ่ง Program Memory ออกเป็น Page ซึ่งแต่ละ Page ก็จะมีขนาด 2 Kbytes ซึ่งคำสั่ง CALL และ GOTO สามารถสั่งให้ Program Counter กระโดดไปมาได้ในช่วง Page เท่านั้นแต่ถ้าเมื่อต้องการกระโดดจาก Page หนึ่งไปยังอีก Page หนึ่ง ดังนั้นจึงต้องไปควบคุม PCLATH<4:3> (Bit Address ที่ 12 และ 13 ให้ชี้ไปยัง Page ที่ต้องการเสียก่อน หลังจากนั้นจึงเรียกคำสั่ง CALL หรือ GOTO ตามอีกที



รูปที่ 2.3 การจัดสรรหน่วยความจำโปรแกรมของ PIC16F877

การเลือก Page ของหน่วยความจำโปรแกรม จะต้องเลือกในรีจิสเตอร์ PCLATH โดยการระบุตำแหน่งที่บิต 3 และบิต 4

PCLATH<4:3>	Page
00	0
01	1
10	2
11	3

ตารางที่ 2.2 ค่า PCLATH<4:3> ในการเลือก Page ของ PIC16F877

เมื่อใช้คำสั่ง CALL ไปที่ Routine ใด Routine หนึ่งแล้ว แล้วจะใช้คำสั่ง RETURN ในการกลับไปตำแหน่งเดิม การ RETURN กลับนั้นไม่จำเป็นต้องสั่ง PCLATH ให้ชี้ไปยัง Page ก่อนหน้าที่จะเรียก CALL เพราะค่า Address ดังกล่าวจะถูกเก็บไว้ใน STACK อยู่แล้ว แต่สำหรับคำสั่ง GOTO เวลาข้าม Page จะต้องสั่งให้ PCLATH ชี้ไปยัง Page ที่ต้องการจะไปทุกครั้ง

2.1.6 การจัดสรรหน่วยความจำข้อมูล RAM ของ PIC16F877

PIC16F877 มีหน่วยความจำข้อมูล RAM สำหรับใช้งานทั่วไป 368 bytes และมีรีจิสเตอร์ไฟล์ 8 bits 57 ตัว ดังรูปที่ 2.4 แต่ละBankมีขนาดสูงสุด 128 bytes แต่มีการใช้งานได้จริงในแต่ละ Bank ต่างกัน โดนในแต่ละ Bank มีการจัดสรรพื้นที่ ดังนี้

- Bank 0 มีช่วง Address 0x00-0x7F
  - Address 0x00-0x1F เป็นพื้นที่รีจิสเตอร์ไฟล์
  - Address 0x20-0x7F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 96 bytes

- Bank 1 มีช่วง Address 0x80-0xFF
  - Address 0x80-0x9F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบาง Address ไม่ใช้งาน
  - Address 0xA0-0xEF เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80 bytes
  - Address 0xF0-0xFF บรรจุข้อมูลเหมือนกับใน Address 0x70-0x7F ในBank 0 เพื่อช่วยให้สามารถใช้ข้อมูลจาก Address 0x70-0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยน Bank

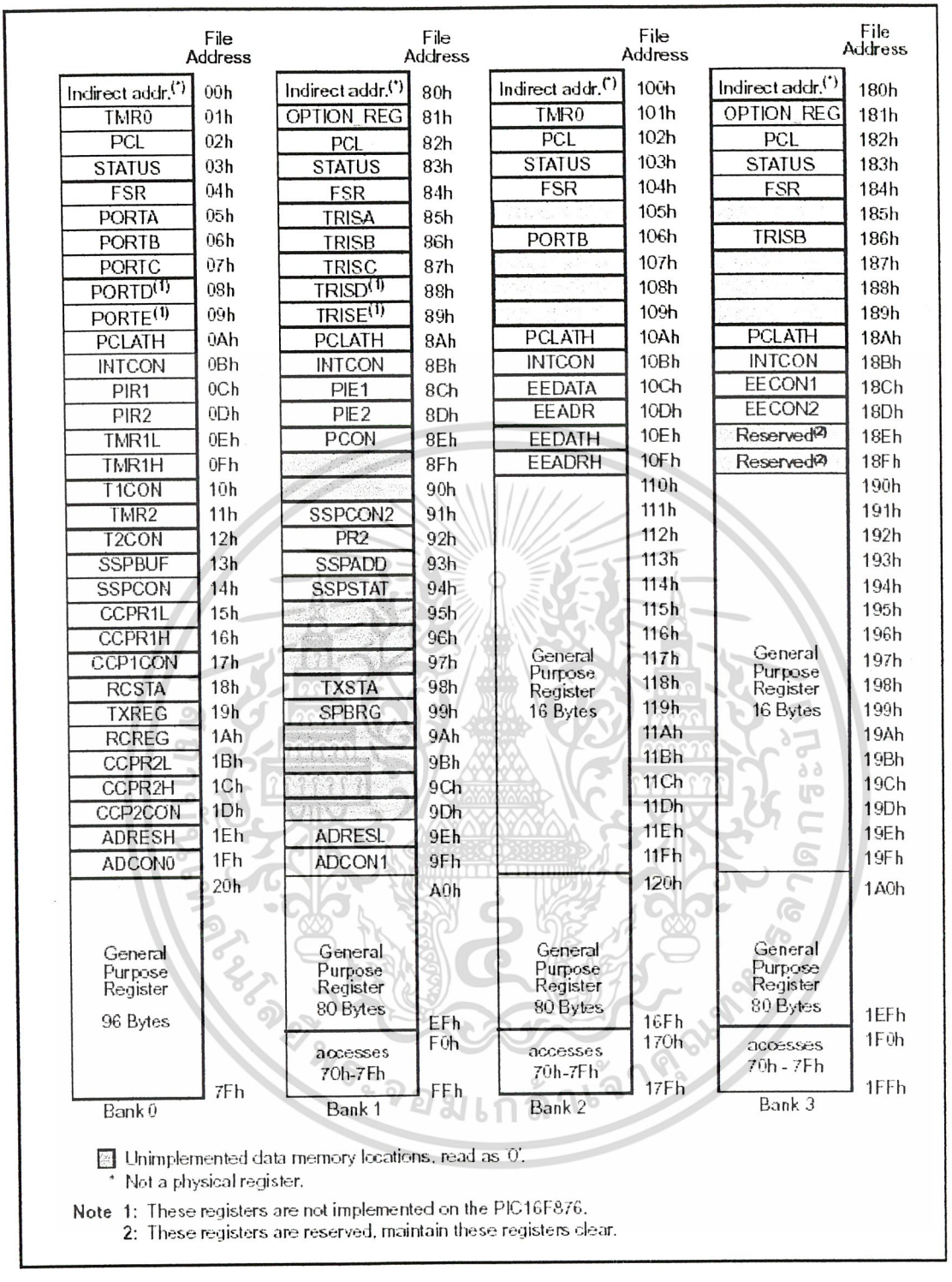
- Bank 2 มีช่วง Address 0x100-0x17F
  - Address 0x100-0x10F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบาง Address ไม่ใช้งาน
  - Address 0x110-0x11F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 16 bytes
  - Address 0x120-0x16F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80 bytes
  - Address 0x170-0x17F บรรจุข้อมูลเหมือนกับใน Address 0x70-0x7F ใน Bank 0 เพื่อช่วยให้สามารถใช้ข้อมูลจาก Address 0x70-0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยน Bank

- Bank 3 มีช่วง Address 0x180-0x1FF
  - Address 0x180-0x18F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบาง Address ไม่ใช้งาน
  - Address 0x190-0x19F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 16 bytes
  - Address 0x1A0-0x1EF เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80bytes
  - Address 0x1F0-0x1FF บรรจุข้อมูลเหมือนกับใน Address 0x70-0x7F ในBank 0 เพื่อช่วยให้สามารถใช้ข้อมูลจาก Address 0x70-0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยน Bank

การเลือก Bank ของหน่วยความจำข้อมูล RAM จะต้องเลือกในรีจิสเตอร์ STATUS โดยการระบุตำแหน่งที่บิต 5 และบิต 6 (RP0, RP1)

RP1:RP0	Bank
00	0
01	1
10	2
11	3

ตารางที่ 2.3 ค่า RP0 และ RP1 ในการเลือก Bank ของ PIC16F877



รูปที่ 2.4 การจัดสรรหน่วยความจำข้อมูล RAM และตำแหน่งรีจิสเตอร์ ของ PIC16F877

2.1.7 รีจิสเตอร์หลักของ PIC16F877

2.1.7.1 รีจิสเตอร์ Configuration word

มีขนาด 14 bits อยู่ที่ Address 2007h ใน Program Memory โดยการทำงานเบื้องต้นของ PIC จะถูกกำหนดที่หน่วยความจำตรงนี้ ไม่ว่าจะเป็น Enable/Disable Power-up timer, Enable/Disable Watchdog timer, Oscillator Selection bits (กำหนดที่มาของสัญญาณนาฬิกา) หน่วยความจำที่ตำแหน่งนี้ จะต้องกำหนดในขณะทำการเขียนโปรแกรมลงสู่ Flash Memory ของ PIC

2.1.7.2 รีจิสเตอร์ STATUS

เป็นรีจิสเตอร์ที่ใช้เก็บข้อมูลแสดงสถานะการทำงานของ PIC16F877

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

รูปที่ 2.5 รีจิสเตอร์ STATUS

บิต 7 **IRP** (Indirect Register Bank Select bit) ใช้เลือก Bank ของหน่วยความจำข้อมูล RAM เมื่อใช้การอ้างตำแหน่งโดยอ้อม

1 = Bank 2, 3 (100h - 1FFh)

0 = Bank 0, 1 (00h - FFh)

บิต 6-5 **RP1:RP0** (Register Bank Select bits) ใช้เลือก Bank ของหน่วยความจำข้อมูล RAM เมื่อใช้การอ้างตำแหน่งโดยตรง

11 = Bank 3 (180h - 1FFh)

10 = Bank 2 (100h - 17Fh)

01 = Bank 1 (80h - FFh)

00 = Bank 0 (00h - 7Fh)

บิต 4 **TO** (Time-out bit) บิตแสดงการเกิด Time Out เมื่อ Watchdog Timer ทำงานครบเวลาที่กำหนด

1 = After power-up, CLRWDT instruction, or SLEEP instruction

0 = A WDT time-out occurred

บิต 3 **PD** (Power-down bit) บิตแสดงการทำงานโหมดประหยัดพลังงาน

1 = After power-up or by the CLRWDT instruction

0 = By execution of the SLEEP instruction

บิต 2 **Z** (Zero bit) ใช้แสดงผลการกระทำคำสั่งทางคณิตศาสตร์

1 = The result of an arithmetic or logic operation is zero

0 = The result of an arithmetic or logic operation is not zero

บิต 1 **DC** (Digit carry/borrow bit) ใช้แสดงการทดหรือยืมระหว่างหลัก

ในกรณีคำสั่ง ADDWF หรือ ADDLW

1 = A carry-out from the 4th low order bit of the result occurred

0 = No carry-out from the 4th low order bit of the result

และแสดงค่ากลับกันเมื่อกระทำคำสั่ง SUBWF หรือ SUBLW

บิต 0 C (Carry/borrow bit) ใช้แสดงการทดหรือยืมของบิต MSB

ในกรณีคำสั่ง ADDWF หรือ ADDLW

1 = A carry-out from the Most Significant bit of the result occurred

0 = No carry-out from the Most Significant bit of the result occurred

และแสดงค่ากลับกันเมื่อกระทำคำสั่ง SUBWF หรือ SUBLW

2.1.7.3 รีจิสเตอร์ W เป็นรีจิสเตอร์ที่มีบทบาทสำคัญ เพราะในการประมวลผลทางคณิตศาสตร์ จะต้องกระทำผ่านรีจิสเตอร์ W และยังทำหน้าที่เป็นตัวกลางในการส่งผ่านสถานะของ Output ไปยัง I/O PORT อีกด้วย

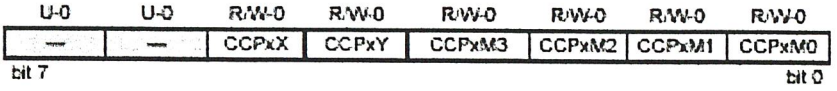
2.1.7.4 Program Counter (PC) เป็นรีจิสเตอร์พิเศษที่ใช้ระบุ Address ของ Program Memory ที่กำลังทำการประมวลผล ซึ่งจะเป็น Counter ขนาด 13 bits โดยทั่วไปแล้ว Counter ตัวนี้จะเพิ่มขึ้น 1 ทุกๆ ครั้งเมื่อมีการประมวลผลคำสั่งเกิดขึ้น 1 ครั้ง ซึ่งค่าที่แสดงก็คือตำแหน่งของคำสั่งต่อไปที่จะทำการประมวลผล แต่เมื่อประมวลคำสั่ง JUMP ตัว counter จะมีค่าเท่ากับตำแหน่งที่คำสั่ง JUMP นั้นอ้างถึง

2.1.7.5 Stack เป็นหน่วยความจำสำรองสำหรับเก็บค่าของ PC ขนาด 13 bits โดยเก็บข้อมูลได้ 8 ระดับ โดยเก็บตำแหน่งของ PC เข้าเมื่อมีคำสั่ง CALL และส่งตำแหน่งที่เก็บไว้ออกไปยัง PC เมื่อมีคำสั่ง RETURN โดยการเก็บจะเป็นแบบ LIFO (Last In First Out)

2.1.7.6 PORTA, PORTB, ... เก็บค่าสถานะของ PORT นั้นๆ

2.1.7.8 TRISA, TRISB, ... ใช้กำหนดทิศทางของขาของ PORT นั้นๆ ว่าขาใดเป็น Input หรือ Output โดยถ้ากำหมดให้เป็น 0 จะเป็น Output ถ้าให้เป็น 1 จะเป็น Input

2.1.7.9 รีจิสเตอร์ CCP1CON และ CCP2CON เป็นรีจิสเตอร์ควบคุมโมดูล CCP1 และ CCP2 (Address 17h, 1Dh)



รูปที่ 2.6 รีจิสเตอร์ CCPxCON

บิต 7-6 ไม่ใช้งาน อ่านค่าเป็น “0”

บิต 5-4 CCPxX : CCPxY ใช้เก็บค่าดิวตี้ไซเกิลของสัญญาณ PWM ใน 2 บิตล่าง จะใช้งานเมื่อโมดูล CCPx ได้รับการกำหนดให้ทำงานในโหมด PWM ส่วนข้อมูล 8 บิตบนจะใช้จากรีจิสเตอร์ CCPRxL

บิต 3-0 CCPxM3:CCPxM0 ใช้เลือกการทำงานของโมดูล CCPx

“0000” – หยุดหรือปิดการทำงานของโมดูล CCPx ทำให้โมดูล CCPx เกิดรีเซต

“0100” – ทำงานในโหมดตรวจจับสัญญาณ โดยตรวจจับที่ทุกๆ ขอบขาลงของสัญญาณ

“0101” – ทำงานในโหมดตรวจจับสัญญาณ โดยตรวจจับที่ทุกๆ ขอบขาขึ้นของสัญญาณ

“0110” – ทำงานในโหมดตรวจจับสัญญาณ โดยตรวจจับที่ทุกๆ ขอบขาขึ้นที่ 4 ของสัญญาณ

“0111” – ทำงานในโหมดตรวจจับสัญญาณ โดยตรวจจับที่ทุกๆ ขอบขาขึ้นที่ 16 ของสัญญาณ

“1000” - ทำงานในโหมดเปรียบเทียบข้อมูล แจ้งผลการเปรียบเทียบด้วยการเซตเอาต์พุตเมื่อเท่ากัน (CCPxIF จะเซต)

“1001” - ทำงานในโหมดเปรียบเทียบข้อมูล แจ้งผลการเปรียบเทียบด้วยการเคลียร์เอาต์พุตเมื่อเท่ากัน (CCPxIF จะเซต)

“1010” - ทำงานในโหมดเปรียบเทียบข้อมูล เมื่อค่าเท่ากันจะทำการอินเตอร์รัปต์ (CCPxIF จะเซต แต่สถานะที่ CCPx จะไม่ได้รับผลกระทบ)

“1011” - ทำงานในโหมดเปรียบเทียบข้อมูล เมื่อค่าเท่ากันจะทำการกำเนิดสัญญาณกระตุ้นเหตุการณ์พิเศษ Trigger Special Event (CCPxIF จะเซต แต่สถานะที่ CCPx จะไม่ได้รับผลกระทบ และทำการรีเซตค่า TMR1)

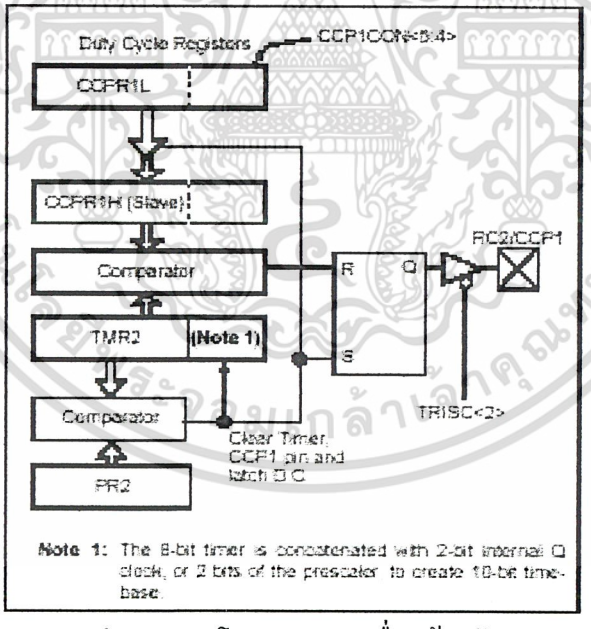
“11xx” - ทำงานในโหมดสร้างสัญญาณ PWM

2.1.8 การทำงานในโหมด PWM

โมดูล CCPx จะกำเนิดสัญญาณมอดูเลชั่นทางความกว้างพัลซ์ (Pulse Width Modulation: PWM) ความละเอียด 10 บิต สัญญาณ PWM ที่สร้างขึ้นจะส่งออกทางขา RC2/CCP1 หรือ RC1/T1OSI/CCP2

2.1.8.1 หลักการสร้างสัญญาณ PWM

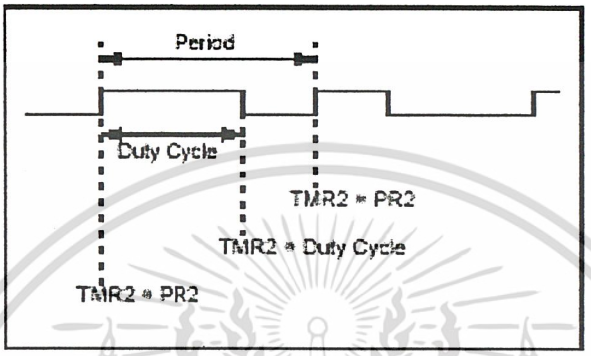
1. กำหนดค่าให้แก่รีจิสเตอร์ PR2 เพื่อกำหนดคาบเวลาของสัญญาณ PWM
2. กำหนดค่าดิวตี้ไซเกิล โดยเขียนข้อมูลลงในรีจิสเตอร์ CCPRxL ร่วมกับบิต 5-4 ของรีจิสเตอร์ CCPxCON
3. กำหนดให้ขาพอร์ต RC2/CCP1 หรือ RC1/T1OSI/CCP2 เป็นเอาต์พุต เพื่อเป็นทางออกของสัญญาณ PWM โดยการเคลียร์บิตที่ 2 หรือ 1 ของรีจิสเตอร์ TRISC
4. กำหนดค่าปรีสเกลเลอร์ของ TMR2 และเอ็นเนเบิลการทำงานของ ไทเมอร์2
5. กำหนดให้โมดูล CCPx ทำงานในโหมด PWM



รูปที่ 2.7 การทำงานของโมดูล CCP1 เพื่อสร้างสัญญาณ PWM

จากรูปที่ 2.7 และ 2.8 ทันทีที่โมดูล CCPx เริ่มทำงานค่าของ TMR2 จะเพิ่มขึ้นจนเท่ากับ PR2 ที่ขาพอร์ต RC2/CCP1 หรือ RC1/T1OSI/CCP2 จะเกิดลอจิก "1" และคงสถานะอยู่เช่นนั้น แล้วค่าของ TMR2 จะเคลียร์และเริ่มนับเพิ่มขึ้นใหม่ หลังจากนั้นค่าดิวตี้ไซเกิลที่กำหนดไว้ในรีจิสเตอร์ CCPRxL และ 2 บิตใน CCPxCON จะถูกถ่ายทอดไปยัง CCPRxH และ 2 บิตใน

หน่วยความจำพิเศษเพื่อเปรียบเทียบกับค่าใน TMR2 และค่าปรีสเกลเลอร์ 2 บิต ซึ่งเพิ่มค่าขึ้นเรื่อยๆ จนกระทั่งเมื่อข้อมูลทั้ง 2 กลุ่มเท่ากัน จะส่งสัญญาณไป ทำให้ขาพอร์ต RC2/CCP1 หรือ RC1/T1OSI/CCP2 กลับมาเป็นลอจิก "0" และคงสถานะอยู่เช่นนั้นจนกระทั่งค่าของ TMR2 เท่ากับ PR2 อีกครั้ง ก็จะเกิดสัญญาณลอจิก "1" เป็นการเริ่มต้นรอบใหม่ของสัญญาณ และจะทำงานวนเช่นนี้จนกระทั่งมีการดิสเอเบิ้ล



รูปที่ 2.8 เอาต์พุตของสัญญาณ PWM

ค่าดิวตี้ไซเกิลของสัญญาณ PWM ที่สร้างขึ้นนี้สามารถเปลี่ยนแปลงได้ตลอดเวลา จากการเปลี่ยนค่าที่ CCPRxL และ 2 บิตใน CCPxCON แต่ค่าของคาบเวลาหรือความถี่จะเปลี่ยนแปลงไม่ได้จนกว่าจะหยุดการทำงาน แล้วกำหนดค่าของคาบเวลาลงในรีจิสเตอร์ PR2 ใหม่ ดังนั้นสัญญาณ PWM ที่สร้างขึ้นจึงสามารถกำหนดค่าดิวตี้ไซเกิลได้ตามความต้องการ

2.1.8.2 การกำหนดคาบเวลาของสัญญาณ PWM ทำได้โดยการเขียนข้อมูลลงในรีจิสเตอร์ PR2 (รีจิสเตอร์คาบเวลาของไทเมอร์ 2) แล้วนำค่าของ PR2 มาคำนวณหาค่าคาบเวลาของสัญญาณ PWM

$$\textit{คาบเวลาของสัญญาณ PWM} = (\textit{ค่าในรีจิสเตอร์ PR2} + 1) \times 4 \times T_{OSC} \times \textit{ค่าปรีสเกลเลอร์ของ TMR2}$$

โดยที่ T_{OSC} คือ คาบเวลาของสัญญาณนาฬิกาหลักมีหน่วยเป็นวินาที
ค่าในรีจิสเตอร์ทั้งหมดคำนวณในรูปของเลขฐานสิบ

$$\textit{ความถี่ของสัญญาณ PWM} = 1/\textit{คาบเวลาของสัญญาณ PWM}$$

2.1.8.3 การกำหนดค่าดิวตี้ไซเกิลของสัญญาณ PWM ทำได้โดยการเขียนข้อมูลไปยังรีจิสเตอร์ CCPRxL ร่วมกับบิต 5-4 ของรีจิสเตอร์ CCPxCON ทำให้สามารถกำหนดความละเอียด

ของสัญญาณ PWM ได้สูงสุดถึง 10 บิต โดย 8 บิตบนจะใช้ข้อมูลในรีจิสเตอร์ CCPRxL ส่วนใน 2 บิตล่างใช้ข้อมูลในบิต 5-4 ของรีจิสเตอร์ CCP1CON

$$\text{ดิวตี้ไซเกิลของสัญญาณ PWM} = (CCPR1L:CCP1CON<5:4>)_{\text{10 บิต}} \times T_{OSC}$$

โดยที่ T_{OSC} คือ คาบเวลาของสัญญาณนาฬิกาหลักมีหน่วยเป็นวินาที
ค่าในรีจิสเตอร์ทั้งหมดคำนวณในรูปของเลขฐานสิบ

2.1.8.4 ความละเอียดของสัญญาณ PWM จะขึ้นอยู่กับบิตปรีสเกลเลอร์ของไทเมอร์ 2 ในขณะที่ความละเอียดสูงสุดของสัญญาณ PWM จะสัมพันธ์กับความถี่ที่กำหนด ความถี่ของสัญญาณนาฬิกา และค่าปรีสเกลเลอร์

$$\text{ความละเอียด (บิต)} = \frac{\log\left(\frac{f_{osc}}{f_{PWM} \times N}\right)}{\log 2}$$

โดยที่ f_{OSC} คือ ความถี่ของสัญญาณนาฬิกาหลักมีหน่วยเป็น Hz
f_{PWM} คือ ความถี่ของสัญญาณ PWM มีหน่วยเป็น Hz
N คือ ค่าปรีสเกลเลอร์ในไทเมอร์ 2
สัญญาณ PWM จะมีความละเอียดสูงสุดเมื่อค่าของปรีสเกลเลอร์เท่ากับ 1

ความถี่ของสัญญาณ PWM (kHz)	**1.22**	**4.88**	**19.53**	**78.12**	**156.3**	**208.3**
ค่าปรีสเกลเลอร์ (1, 4, 16)	16	4	1	1	1	1
ค่ารีจิสเตอร์ PR2	FFh	FFh	FFh	3Fh	1Fh	17h
ความละเอียดสูงสุด (บิต)	10	10	10	8	7	5.5

ตารางที่ 2.4 ตัวอย่างขนาดความถี่และความละเอียดของสัญญาณ PWM ที่ความถี่ 20 MHz

2.2 รีเลย์

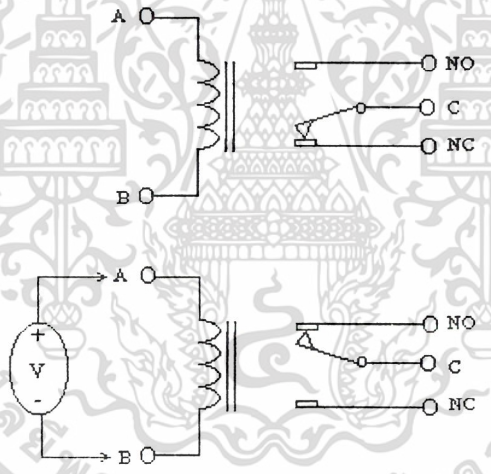
รีเลย์ เป็นอุปกรณ์ทำหน้าที่เป็นสวิตช์ใช้ในการควบคุมวงจรไฟฟ้าได้อย่างหลากหลาย

2.2.1 โครงสร้างการทำงานของรีเลย์

โครงสร้างภายในของ รีเลย์ จะประกอบไปด้วยขดลวด (Coil) 1 ชุด และหน้าสัมผัส (Contactor) ดังรูปที่ 2.9 ซึ่งจะประกอบไปด้วย

- หน้าสัมผัสแบบปกติปิด (Normally Close หรือ NC.) ซึ่งในสภาวะปกติ ขานี้จะต่ออยู่กับขาร่วม (Common)
- หน้าสัมผัสแบบปกติเปิด (Normally Open หรือ NO.) ขานี้จะต่อเข้ากับขาร่วม (Common) เมื่อขดลวดมีแรงดันตกคร่อม หรือกระแสไหลผ่าน (ในปริมาณที่เพียงพอ)

โดยที่ ในรีเลย์ 1 ตัว อาจมีหน้าสัมผัสมากกว่า 1 ชุดได้ เช่น 2 ชุด 4 ชุด แล้วแต่ผู้ผลิต



รูปที่ 2.9 โครงสร้างของรีเลย์

เมื่อขดลวดได้รับแรงดันตกคร่อม (ขา A และ B) จะทำให้มีกระแสไหลผ่านขดลวด ซึ่งจะทำให้เกิดอำนาจสนามแม่เหล็ก ดูดให้หน้าสัมผัส NO และ C ติดกัน

2.2.2 เวลาการตอบสนอง

รีเลย์ เป็นอุปกรณ์ที่มีความเร็วในการทำงานต่ำ เช่นรีเลย์ชนิดแรงดันต่ำ (กระตุ้นขดลวดไม่เกิน 24 V) จะใช้เวลาในการทำงานประมาณ 10 - 50 mS และรีเลย์ขนาดใหญ่ ที่ใช้ควบคุมมอเตอร์ในโรงงานอุตสาหกรรมนั้น อาจใช้เวลาในการทำงานมากกว่า 100 mS เลยทีเดียว

2.2.3 ผลกระทบจากอำนาจแม่เหล็ก

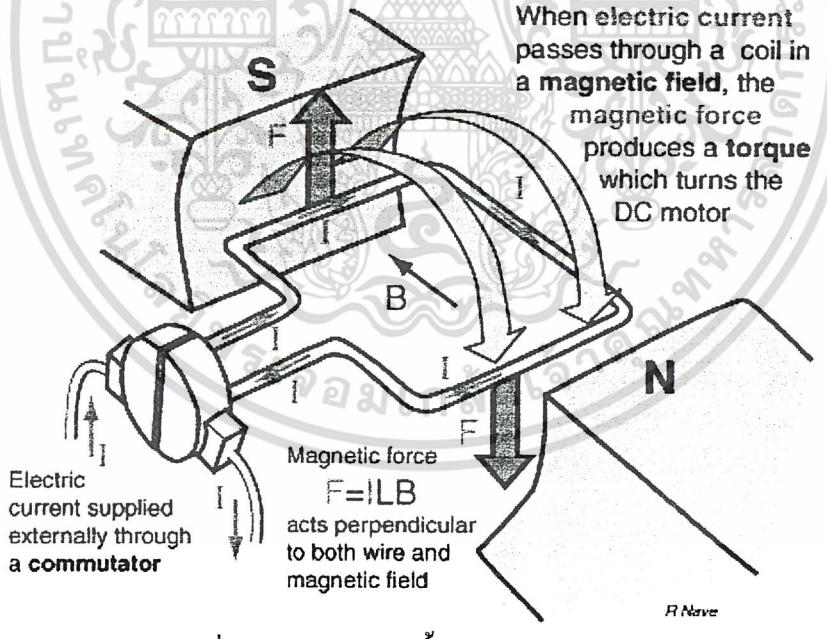
รีเลย์ เป็นอุปกรณ์แม่เหล็ก (Magnetic Device) ดังนั้นการทำงานของรีเลย์จะไปรบกวนการทำงานของวงจรไมโครคอนโทรลเลอร์ การแก้ไขมีหลายวิธี เช่น

- **แยกกราวด์** คือ การแยกกราวด์ของแหล่งจ่ายไฟสำหรับวงจรไมโครคอนโทรลเลอร์และแหล่งจ่ายไฟกระตุ้นรีเลย์ ออกจากกันโดยใช้อุปกรณ์ Optocouplers
- **แยกบอร์ด** คือ การแยกการทำงานในส่วนของวงจรรีเลย์ ออกไปจากบอร์ดไมโครคอนโทรลเลอร์ แล้วทำการชีลด์
- **ลดขนาดแรงดันกระตุ้น** คือ การเปลี่ยนตัวรีเลย์ เช่นเปลี่ยนจากรีเลย์ขนาด 24 V เป็นขนาด 12 V

2.3 มอเตอร์ไฟฟ้ากระแสตรง

2.3.1 โครงสร้างการทำงานของมอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์จะหมุนก็ต่อเมื่อได้รับแรงดันไฟฟ้า ซึ่งเมื่อป้อนแรงดันไฟตรงเข้าที่แปรงถ่าน ก็จะเกิดกระแสไหลผ่านขดลวดตัดสนามแม่เหล็ก ทำให้เกิดแรงผลักให้ขดลวดหมุน



รูปที่ 2.10 การทำงานเบื้องต้นของมอเตอร์

ขนาดของแรงที่เกิดขึ้นนี้หาได้จาก

$$F = Bil$$

เมื่อ **F** = แรงที่เกิดขึ้นบนตัวนำหนึ่งตัว (นิวตัน)

B = ความหนาแน่นสนามแม่เหล็ก (เทสลา)

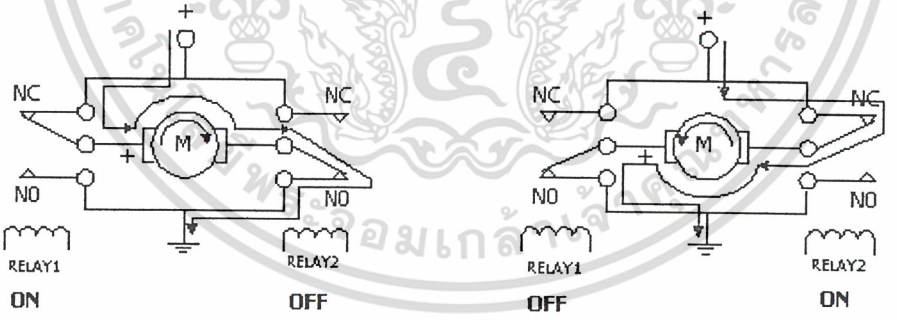
i = กระแสที่ไหลในตัวนำ (แอมแปร์)

l = ความยาวของตัวนำ (เมตร)

แรง F ที่เกิดขึ้นนี้ จะอยู่ในแนวที่ตั้งฉากกับสนามแม่เหล็ก และกระแสที่ผ่านในตัวนำนั้นๆ โดย คอมมิวเตเตอร์(Commentator) เป็นตัวทำให้กระแสไหลผ่านตัวนำไปในทิศทางเดียวตลอดเวลาภายใต้ขั้วแม่เหล็กแต่ละขั้ว ซึ่งช่วยทำให้เกิดแรงบิดไปในทิศทางเดียวกันอย่างต่อเนื่อง

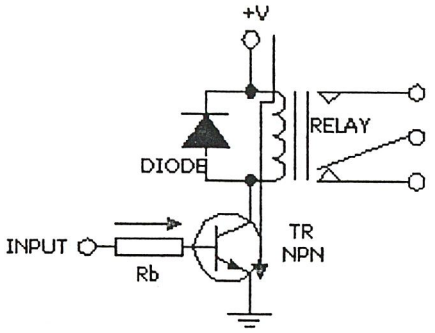
2.3.2 การขับมอเตอร์ไฟฟ้ากระแสตรง

ในการใช้ไมโครคอนโทรเลอร์เป็นตัวควบคุมการหมุนของมอเตอร์ไฟฟ้ากระแสตรงนั้น จะต้องมีส่วนของวงจร ที่เรียกว่าวงจรขับมอเตอร์ (Driver) ในส่วนของวงจรกลับทิศทางของมอเตอร์นั้น สามารถที่จะใช้รีเลย์ต่อวงจร สวิตซ์เพื่อกลับทิศทางของขั้วไฟกระแสตรง หรืออาจใช้อุปกรณ์สารกึ่งตัวนำที่เป็นวงจรขับกำลังเช่น ทรานซิสเตอร์ มอสเฟต



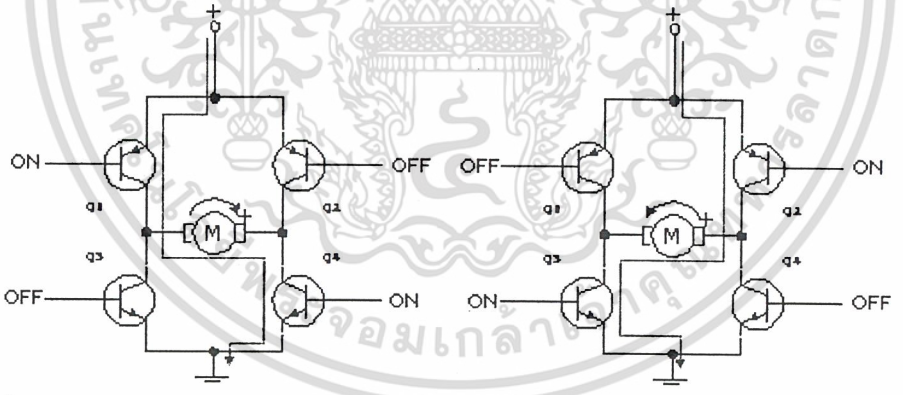
รูปที่ 2.11 การกลับทิศทางของมอเตอร์กระแสตรงโดยใช้รีเลย์

จากรูปที่ 2.11 เป็นการใช้รีเลย์ควบคุมการเปลี่ยนทิศทางการหมุนของมอเตอร์ โดยการควบคุมการปิด-เปิด ที่รีเลย์ 2 ตัว ซึ่งจะทำหน้าที่กลับทิศทางของขั้วไฟที่ป้อนให้กับมอเตอร์ โดยการสลับการทำงานของรีเลย์ เช่นให้รีเลย์ตัวที่ 1 ทำงาน (ON) และรีเลย์ตัวที่ 2 หยุดทำงาน (OFF) จะทำให้มอเตอร์หมุนไปทางซ้าย และในทำนองเดียวกันถ้าหากรีเลย์ตัวที่ 1 หยุดทำงาน (OFF) และรีเลย์ตัวที่ 2 ทำงาน (ON) ก็จะทำให้มอเตอร์หมุนไปทางขวา



รูปที่ 2.12 การใช้ทรานซิสเตอร์เพื่อขับรีเลย์ให้ทำงาน

จากรูปที่ 2.12 เป็นวงจรขับรีเลย์โดยใช้ทรานซิสเตอร์ทำหน้าที่ขยายกระแส ด้วยเหตุผลเพราะไม่สามารถจะใช้ขา เอาต์พุตของไมโครคอนโทรลเลอร์ป้อนกระแสไฟที่ขดลวดของรีเลย์โดยตรงได้ เนื่องจากว่ากระแสที่จ่ายออกมาจากขา เอาต์พุตของไมโครคอนโทรลเลอร์มีค่าน้อยเกินไป ดังนั้นจึงต้องมีส่วนของวงจรทรานซิสเตอร์เพื่อที่จะทำการขยายกระแสให้เพียงพอในการป้อนให้กับขดลวดของรีเลย์ ส่วนไดโอดนำมาต่อไว้สำหรับป้องกันแรงดันย้อนกลับที่เกิดจากการเหนี่ยวนำของสนามแม่เหล็กในขณะเกิดการยุบตัว ซึ่งอาจจะทำให้ทรานซิสเตอร์เสียหายได้



รูปที่ 2.13 การใช้ทรานซิสเตอร์เป็นวงจรขับและกำหนดทิศทางของมอเตอร์กระแสตรง

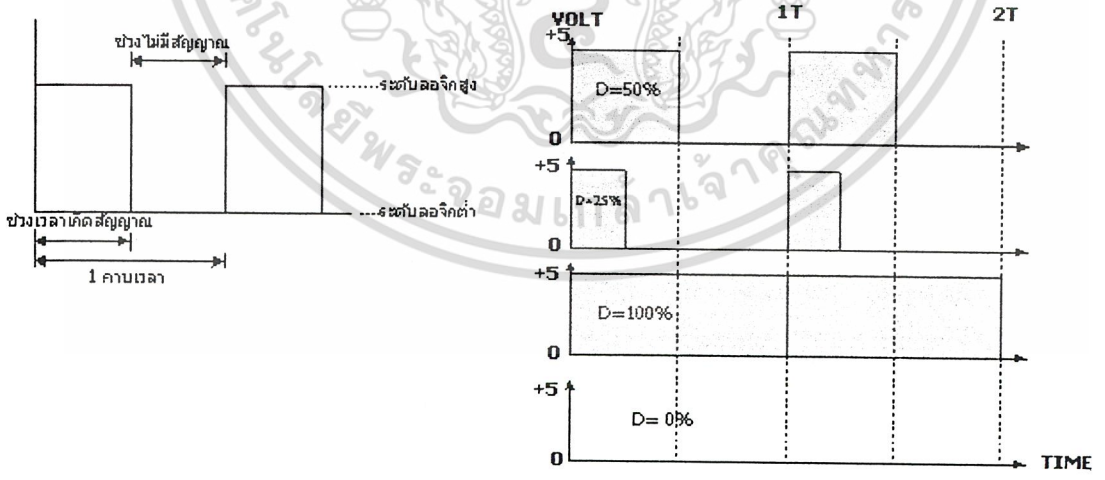
จากรูปที่ 2.13 เป็นวงจรฟูลบริดจ์ ซึ่งจะประกอบไปด้วยทรานซิสเตอร์กำลัง 4 ตัวที่ทำหน้าที่ขับ และควบคุมทิศทางการหมุนของมอเตอร์ ถ้าหากกำหนดให้ทรานซิสเตอร์ Q1 และ Q4 อยู่ในสภาวะทำงาน (Active) กระแสไฟฟ้าจะไหลผ่านมอเตอร์จากซ้ายไปขวา ทำให้มอเตอร์หมุนไปทางขวา ในทำนองเดียวกันถ้าหากเราทำให้ทรานซิสเตอร์ Q2 และ Q3 อยู่ในสภาวะทำงาน (Active) กระแสไฟฟ้าก็จะไหลจากทางขวาไปทางซ้ายซึ่งจะส่งผลให้มอเตอร์กลับทิศทางการหมุนจากทางขวาไปทางซ้าย

2.3.3 การควบคุมความเร็วของมอเตอร์กระแสตรง

การควบคุมความเร็วของมอเตอร์กระแสตรงมีหลายวิธีด้วยกัน ซึ่งอาจจะใช้วิธีการควบคุมแบบพื้นฐาน เช่นการควบคุมด้วยวิธีการใช้ตัวต้านทานปรับค่าได้ต่ออนุกรมกับมอเตอร์เพื่อลดแรงดันไฟฟ้าที่ป้อนให้กับมอเตอร์ หรือใช้วิธีการมอดูเลชั่นทางความกว้างของพัลส์ PWM (Pulse Width Modulation)

2.3.3.1 การมอดูเลชั่นทางความกว้างพัลส์ PWM (Pulse Width Modulation) จะเป็นการปรับเปลี่ยนความกว้างของสัญญาณพัลส์ โดยความถี่ของสัญญาณพัลส์จะไม่มีการเปลี่ยนแปลง หรือเป็นการเปลี่ยนแปลงที่ค่าของดิวตี้ไซเกิล (duty cycle) นั้นเอง ซึ่งค่าของดิวตี้ไซเคิล คือช่วงความกว้างของพัลส์ที่มีสถานะลอจิกสูง โดยคิดสัดส่วนเป็นเปอร์เซ็นต์จากความกว้างของพัลส์ทั้งหมด ยกตัวอย่างเช่น ถ้าหากค่าดิวตี้ไซเคิลมีค่าเท่ากับเท่ากับ 50% ก็หมายถึงใน 1 รูปสัญญาณพัลส์จะมีช่วงของสัญญาณที่เป็นสถานะลอจิกสูงอยู่ครึ่งหนึ่ง และสถานะลอจิกต่ำอยู่อีกครึ่งหนึ่ง ดังรูป 2.14 และในทำนองเดียวกันถ้าหากค่าดิวตี้ไซเคิลมีค่ามาก หมายความว่าความกว้างของพัลส์ที่เป็นสถานะลอจิกสูงจะมีความกว้างมากขึ้น หากค่าดิวตี้ไซเคิลมีค่าเท่ากับ 100% ก็หมายความว่าจะไม่มีสถานะลอจิกต่ำเลย ซึ่งค่าดิวตี้ไซเคิลสามารถ จะหาได้จากค่าความสัมพันธ์ดังนี้

ค่าดิวตี้ไซเคิล = (ช่วงของสัญญาณพัลส์/คาบเวลาทั้งหมดของสัญญาณ) × 100%



รูปที่ 2.14 ความกว้างของพัลซ์ขนาดต่างๆ ของสัญญาณพัลซ์ที่มีความถี่คงที่

บทที่ 3
การออกแบบและการสร้าง

3.1 PIC Programmer Board

3.1.1 วัตถุประสงค์

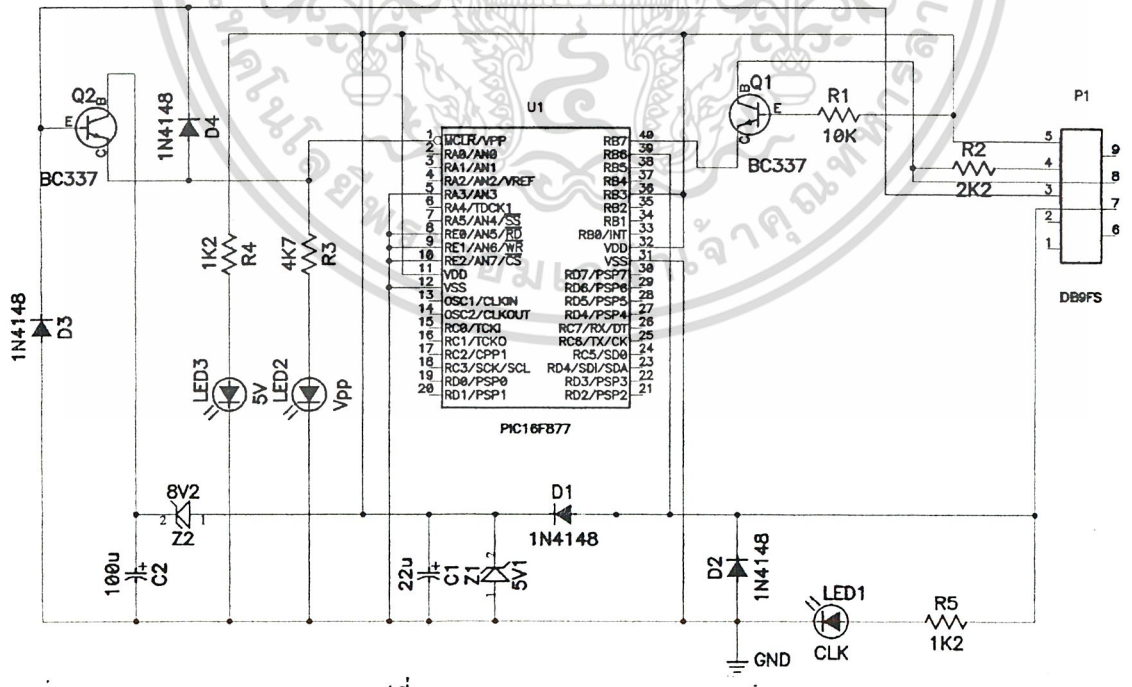
ในการเขียนโปรแกรมลงบนตัวไมโครคอนโทรลเลอร์ PIC จำเป็นจะต้องมี บอร์ดที่ทำหน้าที่ในการโปรแกรมข้อมูลเข้าสู่ PIC หรือที่เรียกว่า Programmer Board

3.1.2 โครงสร้างและการทำงาน

PIC Programmer ที่จัดทำขึ้น สามารถโปรแกรมข้อมูลได้ทั้ง PIC ตระกูล 16F87X ที่มี 28 Pins และ 40 Pins ผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์ (Serial Port) โดยไม่จำเป็นต้องใช้แหล่งจ่ายไฟจากภายนอก เนื่องจากอาศัยไฟจากขาของพอร์ต RS232 ที่เชื่อมต่อกับคอมพิวเตอร์

การทำงานในการโปรแกรมข้อมูลนั้น ข้อมูลจะถูกส่งผ่านทางพอร์ต RS232 ป้อนเข้าที่ขา RB7 ตามจังหวะการโปรแกรมข้อมูลที่ถูกกำหนดโดยสัญญาณ CLK ที่ป้อนเข้าที่ขา RB6

จากวงจรของ PIC Programmer Board ดังรูปที่ 3.1 ได้ออกแบบเป็นลายวงจร (PCB) สำหรับใช้ในการกัดแผ่นทองแดง เพื่อใช้ในการลงอุปกรณ์



รูปที่ 3.1 วงจรของ PIC Programmer

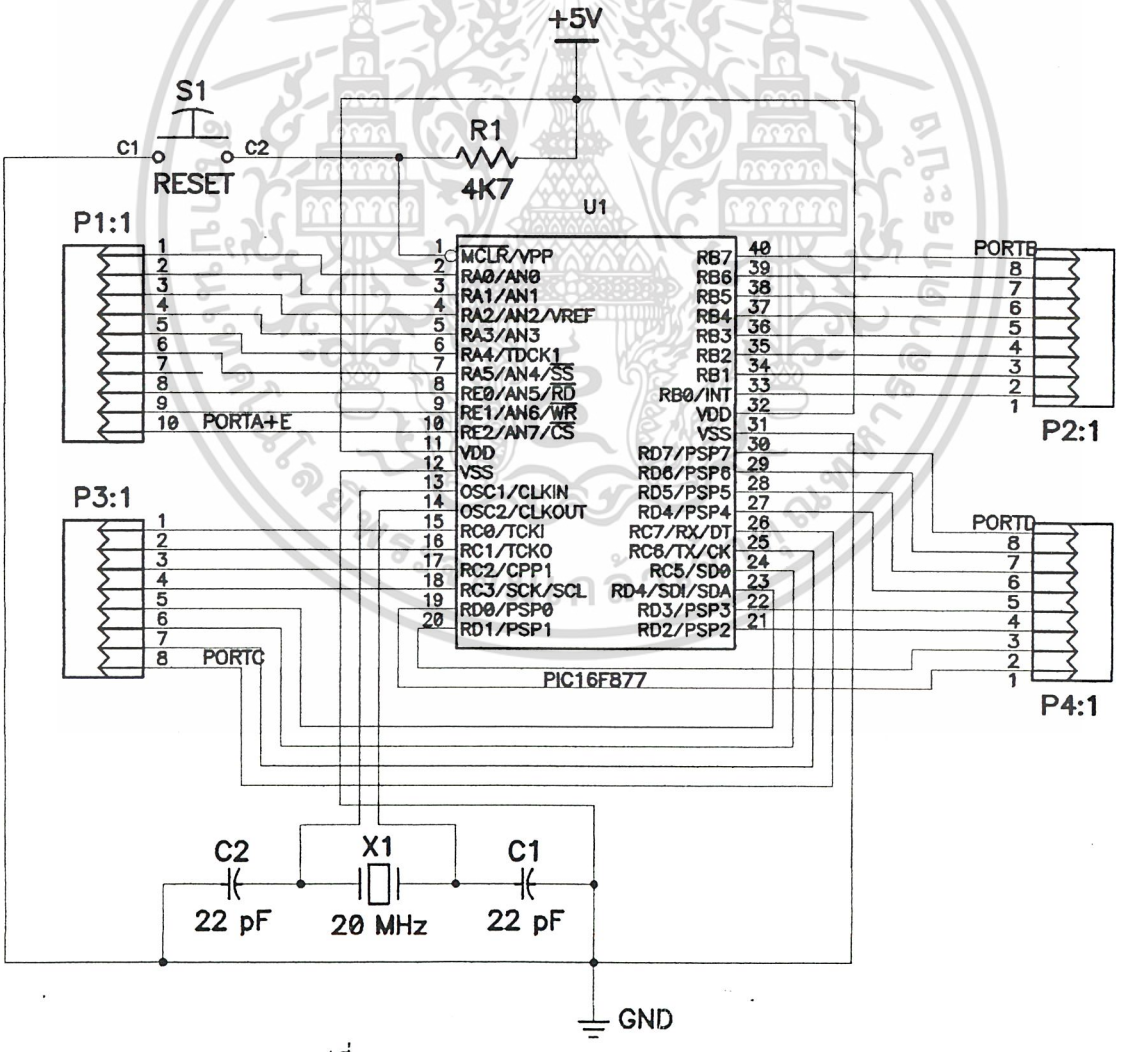
3.2 PIC16F877 Control Board

3.2.1 วัตถุประสงค์

PIC16F877 Control Board ได้ถูกออกแบบให้ขึ้นเพื่อความสะดวกในการนำไมโครคอนโทรลเลอร์ PIC16F877 ไปใช้งาน โดยสามารถเชื่อมต่อระหว่างพอร์ตของ PIC16F877 กับอุปกรณ์ภายนอก

3.2.2 โครงสร้างและการทำงาน

PIC16F877 Control Board ได้แยก Port A, Port B, Port C, Port D และ Port E เพื่อนำไปต่อใช้งานเป็นขา Input หรือ Output ได้โดยเป็นอิสระแยกจากกัน โดยมีคริสตอล (Crystal) ขนาด 20 MHz เป็นแหล่งกำเนิดสัญญาณนาฬิกา (Clock) เพื่อเป็นจังหวะในการเฟทซ์และเอ็กซีคิวต์คำสั่งตามโปรแกรมการทำงานของไมโครคอนโทรลเลอร์ PIC16F877



รูปที่ 3.2 วงจรของ PIC16F877 Control Board

3.3 LED & 7-Segments Board

3.3.1 วัตถุประสงค์

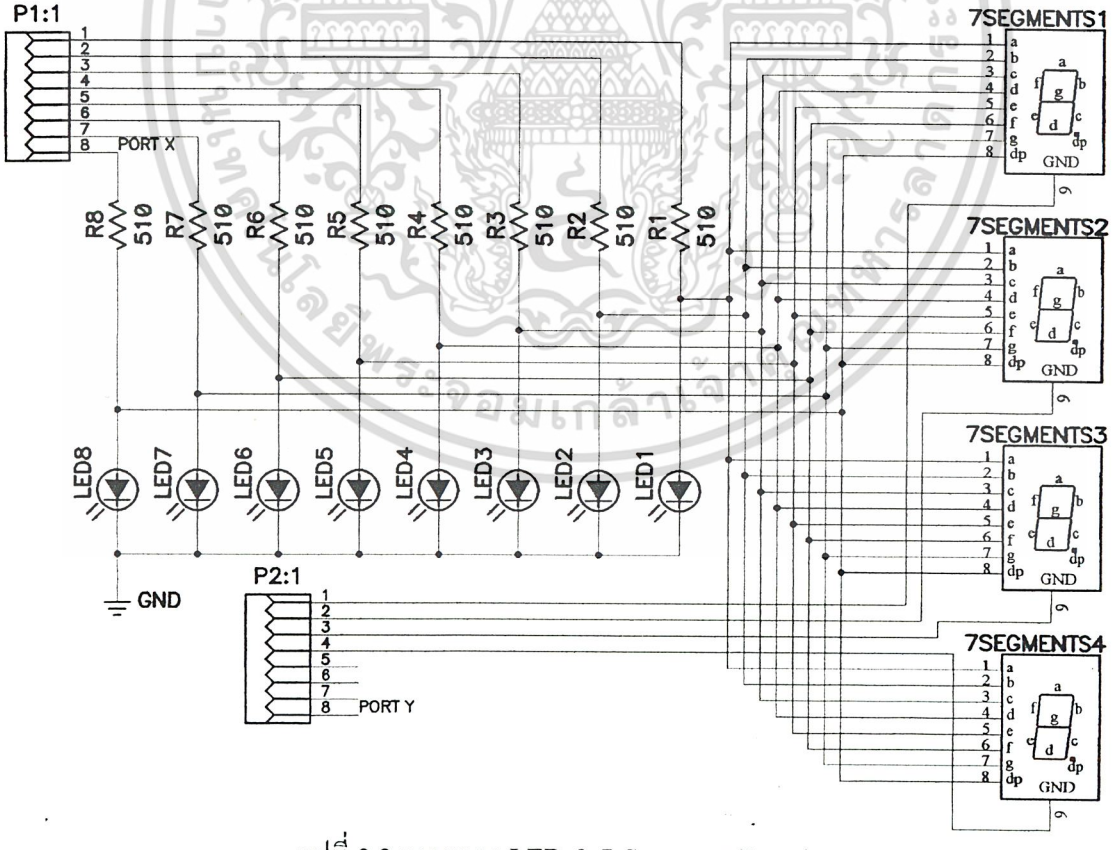
เป็นอุปกรณ์เอาต์พุตพื้นฐาน ที่สามารถเชื่อมต่อกับ PIC16F877 Control Board ได้ เพื่อใช้ในการแสดงผลที่ได้จากการทำงานของ PIC16F877

3.3.2 โครงสร้างและการทำงาน

การใช้งานจะมีอยู่ 2 ส่วน คือ การแสดงผลโดยใช้ LED 8 ดวง และ 7-Segments 4 หลัก

3.3.2.1 LED การแสดงผลจะใช้ข้อมูลจากพอร์ตของ PIC16F877 จำนวน 8 บิต โดยถ้าส่งมาเป็นลอจิก “1” หลอด LED จะสว่าง และถ้าเป็นลอจิก “0” หลอด LED จะดับ

3.3.2.2 7-Segments การแสดงผลจะใช้ข้อมูลจากพอร์ตเดียวกับที่ใช้แสดงผลของ LED และจะต้องใช้ข้อมูลจากพอร์ตของ PIC16F877 อีกพอร์ตเพื่อใช้ในการมัลติเพล็กซ์ 7-Segment ทั้ง 4 หลัก โดยหลักการ คือ การขับ 7-Segments ทีละหลักด้วยความเร็วสูงจนมองเห็นว่า ตัวเลขทุกหลักแสดงผลพร้อมกัน การควบคุมการติดดับของ 7-Segments จะกระทำที่ขา Common Cathode โดยถ้าเป็นลอจิก “1” 7-Segments จะติด และถ้าเป็นลอจิก “0” 7-Segments จะดับ



รูปที่ 3.3 วงจรของ LED & 7-Segments Board

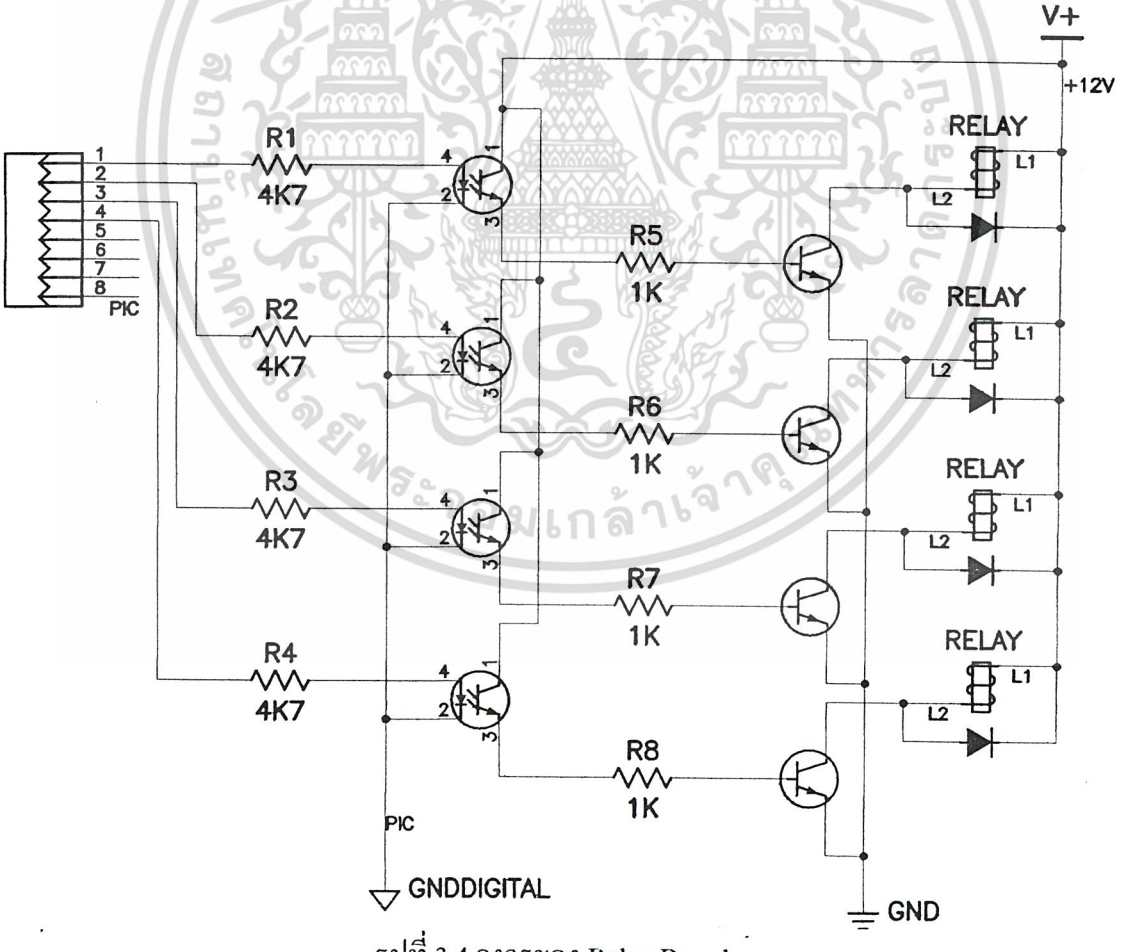
3.4 Relay Board

3.4.1 วัตถุประสงค์

Relay Board เป็นการประยุกต์ใช้งาน PIC16F877 เพื่อใช้ในการควบคุมการทำงานของ Relay ซึ่งสามารถนำไปใช้ประโยชน์ในการควบคุมวงจรไฟฟ้าได้หลากหลาย เช่น ใช้เป็นสวิตซ์ ใช้ในการควบคุมคอนโทรลวาล์วของระบบนิวแมติกส์ หรือใช้ในการควบคุมมอเตอร์

3.4.2 โครงสร้างและการทำงาน

เนื่องจากรีเลย์เป็นอุปกรณ์แม่เหล็ก (Magnetic Device) จึงจำเป็นต้องมีการแยกกราวด์ เพื่อไม่ให้รบกวนการทำงานของไมโครคอนโทรลเลอร์ โดยการใช้ Optocouplers ในการแยกวงจรรีเลย์ออกจากวงจรของไมโครคอนโทรลเลอร์ และใช้ Transistors ทำหน้าที่ขยายกระแส ในการกระตุ้นขดลวดของรีเลย์ สำหรับไดโอด ที่ต่อคร่อมขดลวดของรีเลย์ (free wheeling diode) ทำหน้าที่ป้องกันแรงดันไฟฟ้าสไปค์ เนื่องจากกระแสในขดลวดของรีเลย์หยุดไหลทันทีทันใด



รูปที่ 3.4 วงจรของ Relay Board

3.5 DC Motor Board

3.5.1 วัตถุประสงค์

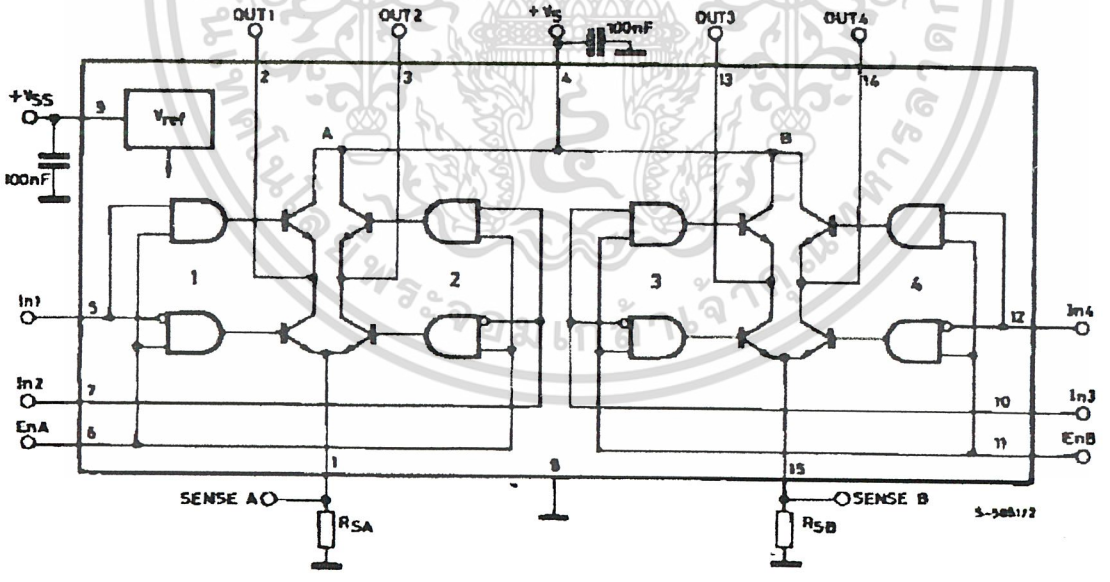
เป็นชุดทดลอง DC Motor ที่ควบคุมการทำงานจากไมโครคอนโทรลเลอร์ PIC16F877 โดยใช้โมดูล Pulse Width Modulator (PWM) ในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

3.5.2 โครงสร้างและการทำงาน

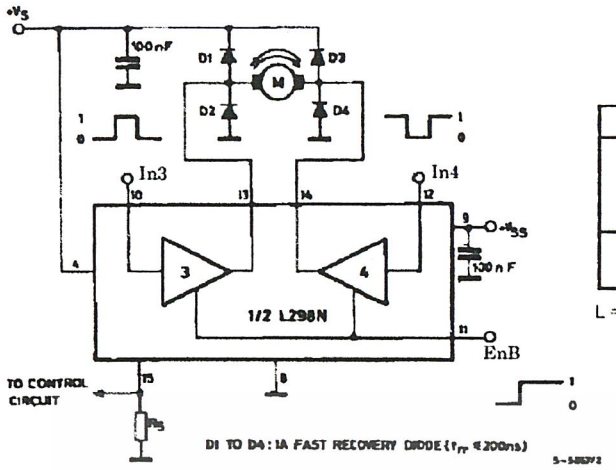
เพื่อป้องกันการรบกวนการทำงานของไมโครคอนโทรลเลอร์ เนื่องจากการทำงานของมอเตอร์ จึงจำเป็นต้องมีการแยกกราวด์ โดยการใช้ Optocouplers ในการแยกวงจรขับมอเตอร์ออกจากวงจรของไมโครคอนโทรลเลอร์

3.5.2.1 Dual Full-Bridge Driver L298 เป็นไอซีควบคุมการทำงานของมอเตอร์ในลักษณะของวงจรฟูลบริดจ์ ซึ่งภายในจะประกอบด้วยวงจรฟูลบริดจ์ 2 วงจร ดังรูปที่ 3.5 โดยแต่ละวงจรสามารถให้แรงดันเอาต์พุตสูงสุดที่ 46 V และจ่ายกระแสสูงสุดที่ 2 A สำหรับการต่อใช้งานคู่กันทั้ง 2 วงจร ในแบบขนาน จะสามารถจ่ายกระแสได้สูงสุดที่ 4 A

สำหรับการควบคุมทิศทางของมอเตอร์ จะป้อนสัญญาณเข้าที่ขา In1, In2 และ In3, In4 และในการควบคุมความเร็วจะป้อนสัญญาณ PWM เข้าที่ขา EnA และ EnB ดังรูปที่ 3.6



รูปที่ 3.5 Block Diagram ของ L298



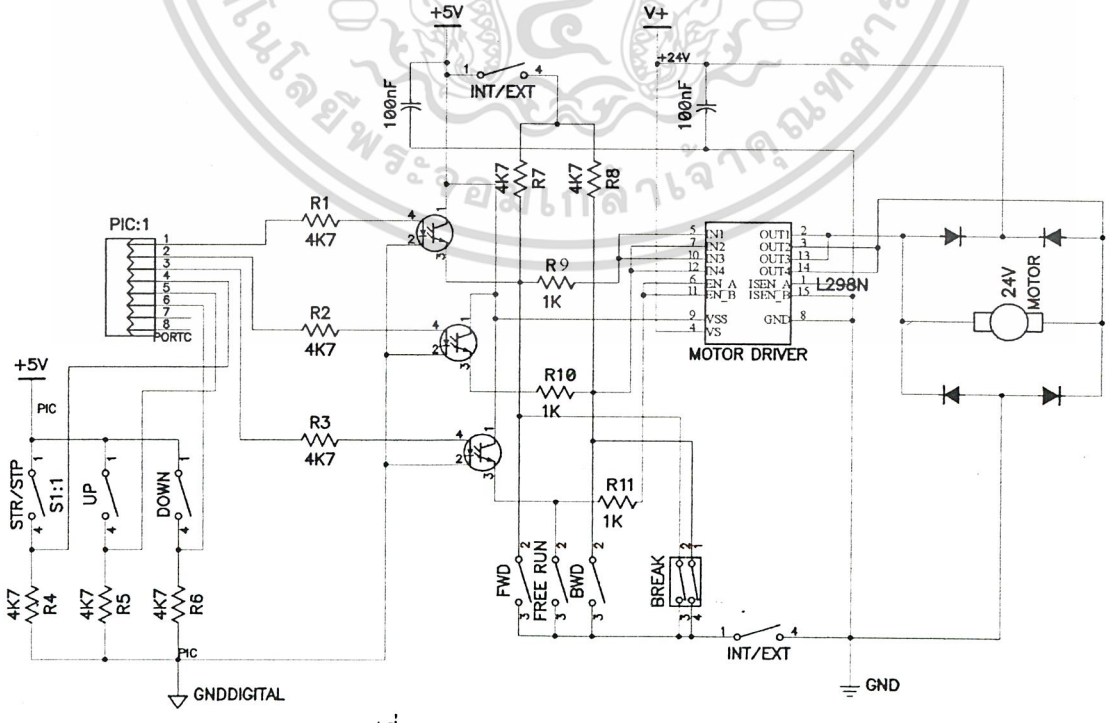
Inputs		Function
EnB = H	In3 = H ; In4 = L	Forward
	In3 = L ; In4 = H	Reverse
	In3 = In4	Fast Motor Stop
EnB = L	In3 = X ; In4 = X	Free Running Motor Stop

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

รูปที่ 3.6 การควบคุมทิศทางมอเตอร์ของ L298

3.5.2.2 การควบคุมมอเตอร์ สำหรับการควบคุมมอเตอร์ จะทำโดยการส่งสัญญาณไปยัง L298 โดยได้ออกแบบการทำงานไว้เป็น 2 ลักษณะ คือ

- โหมดการควบคุมจากภายใน DC Motor Board ตั้งค่าโดยการ On สวิตซ์ INT/EXT สัญญาณทิศทางที่ส่งไปยัง L298 จะถูกกำหนดโดยตรงจาก สวิตซ์ควบคุมทิศทาง และการเบรค ที่อยู่บน DC Motor Board
- โหมดการควบคุมจากภายนอก DC Motor Board ผ่านทาง PIC16F877 ตั้งค่าโดยการ Off สวิตซ์ INT/EXT สัญญาณควบคุมทิศทางที่ส่งไปยัง L298 จะมาจาก PIC16F877 โดยรับค่าการสั่งงานจาก สวิตซ์เมตริกซ์



รูปที่ 3.7 วงจรของ DC Motor Board

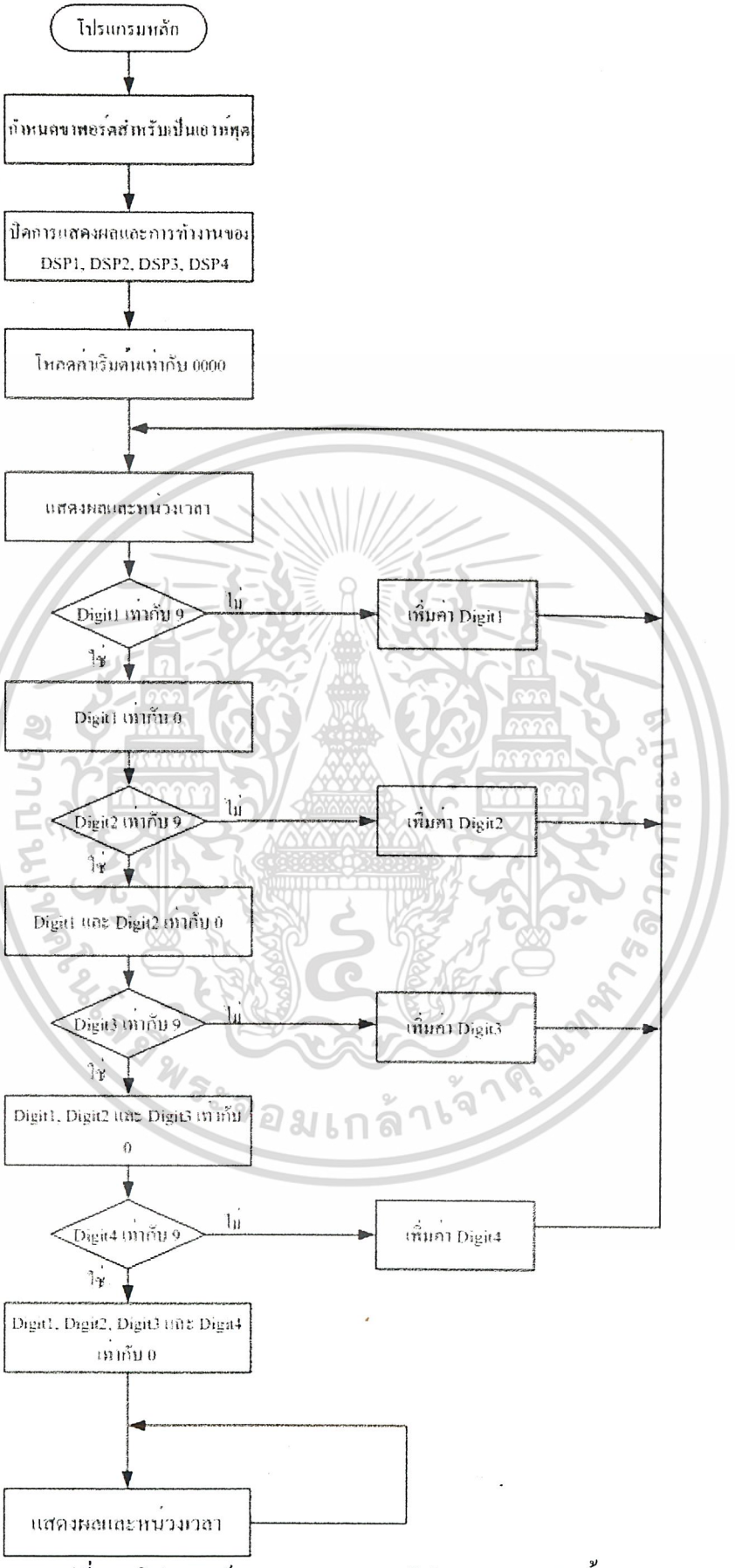
บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดสอบการลงโปรแกรมและการใช้งานโปรแกรม

ในการทดลองนี้จะเป็นการตรวจสอบทั้งในส่วนของ การทำงานของ PIC Programmer Board และในส่วนของ PIC16F877 Control Board โดยตรวจสอบการทำงานของโปรแกรมซึ่งแสดงผลผ่านทาง LED & 7-Segments Board

การทดสอบในการแสดงการทำงานของ LED & 7-Segments Board เพื่อตรวจสอบการลงโปรแกรมและการใช้งานโปรแกรม ได้ออกแบบในส่วนของการแสดงการทำงานของ 7-Segment ทั้ง 4 หลัก ให้มีการนับค่าขึ้นจาก “0000” ถึง “9999” โดยใช้หลักการมัลติเพล็กซ์ ซึ่งแสดงโฟลวชาร์ตการทำงานได้ดังรูปที่ 4.1

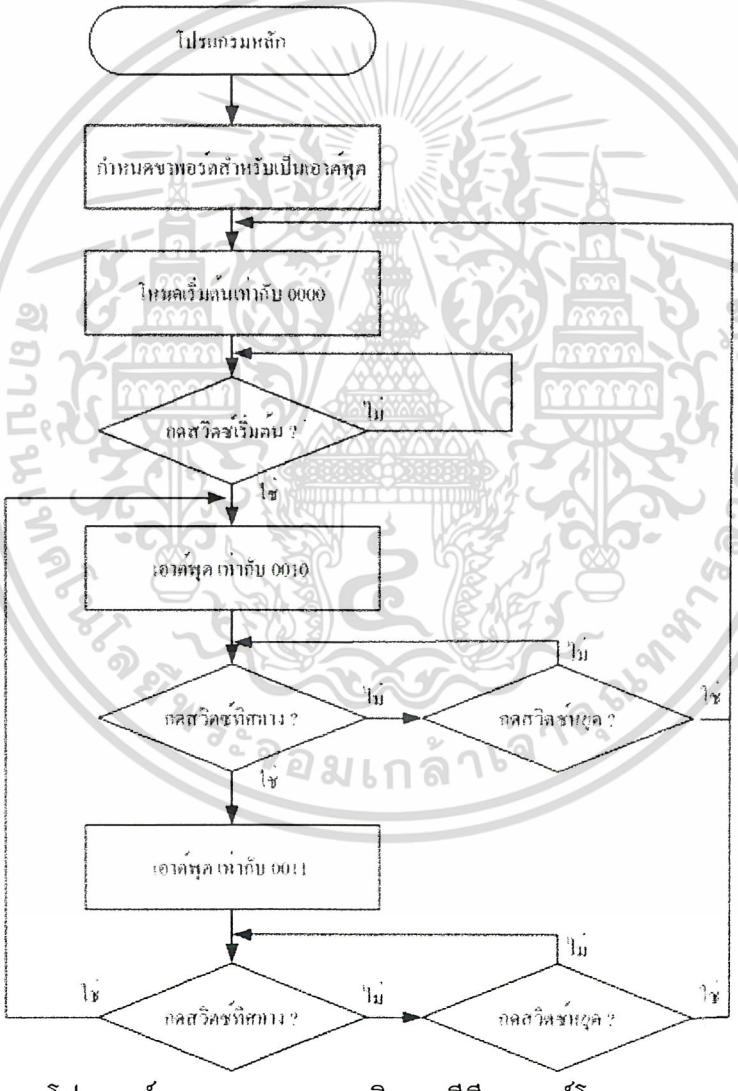


รูปที่ 4.1 โฟลวชาร์ตแสดงการทำงานโปรแกรมการนับขึ้น

4.2 การทดสอบการทำงานของ Relay Board

รีเลย์สามารถทำหน้าที่ได้หลายอย่าง ทั้งในส่วนการทำหน้าที่เป็นสวิตซ์ควบคุมการทำงานของวงจรไฟฟ้า หรือการควบคุมทิศทางของมอเตอร์ไฟฟ้ากระแสตรง ดังนั้นในการทดลองนี้จึงทำการออกแบบระบบเอาต์พุตพื้นฐาน เป็นหลอดไฟขนาด 24 V. ที่ติดตั้งอยู่บน Relay Board เพื่อแสดงผลการทำงานของรีเลย์

นอกจากนั้น Relay Board ยังได้ถูกออกแบบให้สามารถควบคุมทิศทางของมอเคอร์ไฟฟ้ากระแสตรง โดยการต่อการทำงานของ Relay ในแบบวงจร H-Bridge ซึ่งแสดงโฟลวชาร์ตการทำงานได้ดังรูปที่ 4.2

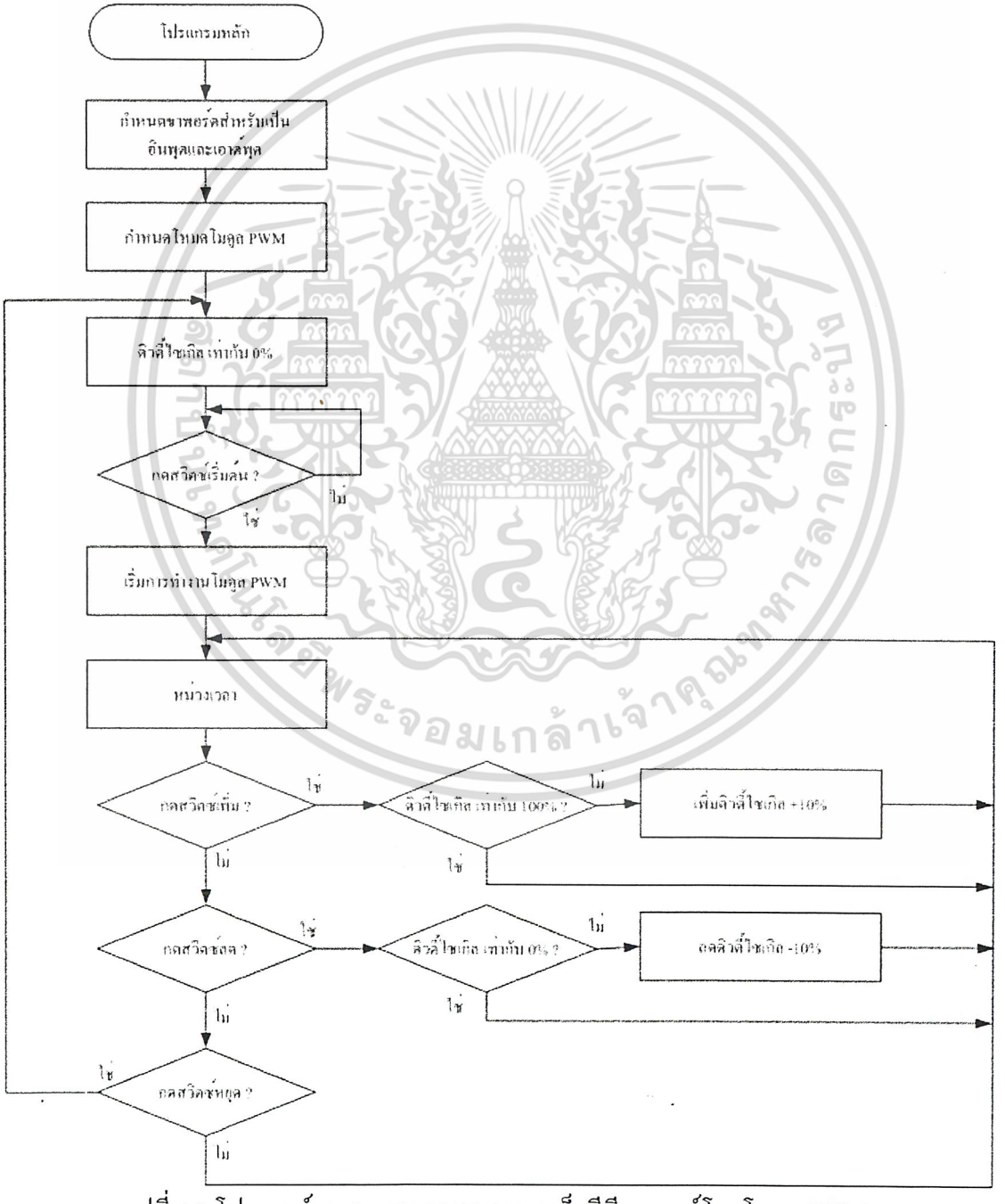


รูปที่ 4.2 โฟลวชาร์ตแสดงการควบคุมทิศทางดีซีมอเตอร์โดยวงจร H-Bridge

4.3 การทดสอบการทำงานของ DC Motor Board

ในการทดลองนี้จะเป็นการเน้นความสำคัญของการควบคุมความเร็วของ มอเตอร์ไฟฟ้ากระแสตรง โดยใช้โมดูล Pulse Width Modulator (PWM) ของ PIC16F877 ควบคุมความเร็วมอเตอร์ผ่านทาง ขา Enable ของไอซีขับมอเตอร์ L298

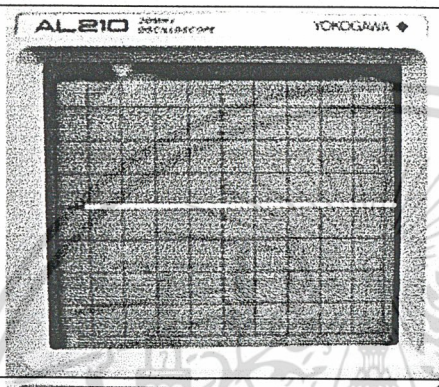
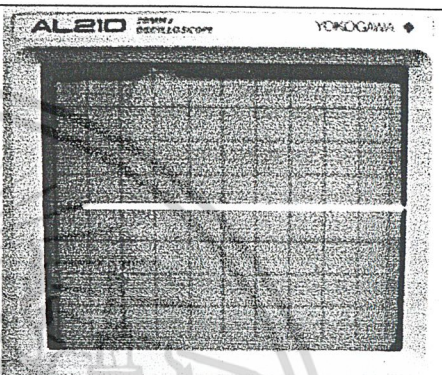
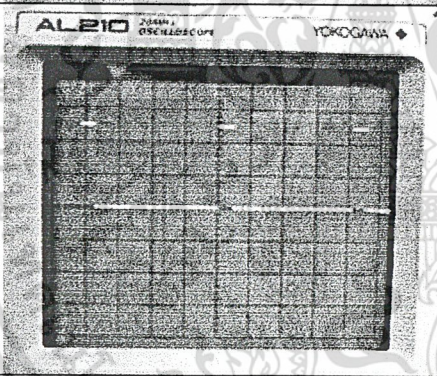
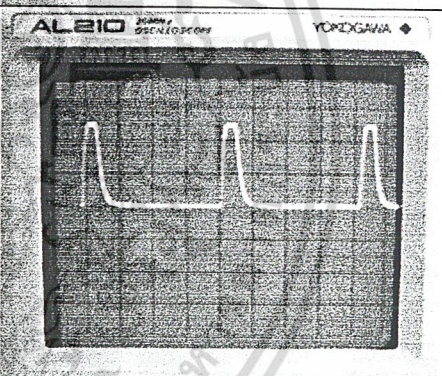
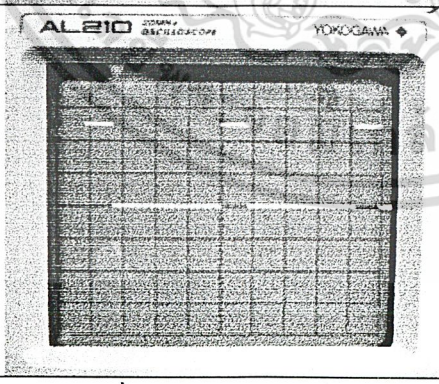
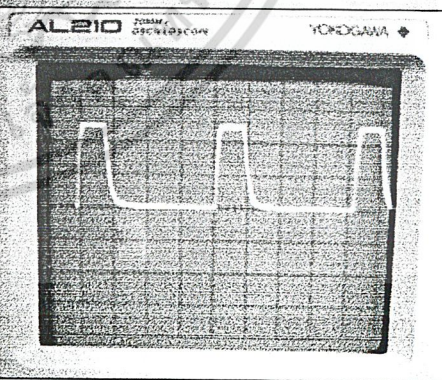
โดยในการทดสอบการทำงานได้ออกแบบโปรแกรม โดยใช้การควบคุมค่าดิวตี้ไซเกิลของโมดูล PWM โดยกำหนดค่าการทำงานของดิวตี้ไซเกิล ให้สามารถเพิ่มและลดค่าเปอร์เซนต์ของดิวตี้ไซเกิลได้ครั้งละ 10% เพื่อตรวจสอบการควบคุมความเร็วของมอเตอร์ที่ค่าดิวตี้ไซเกิลขนาดต่างๆ



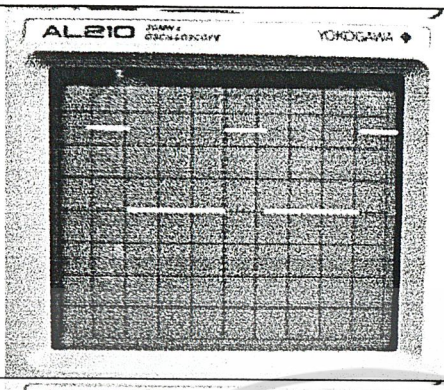
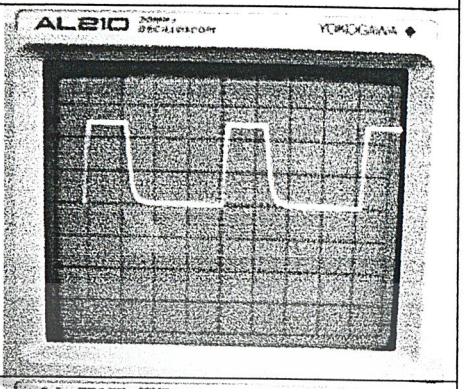
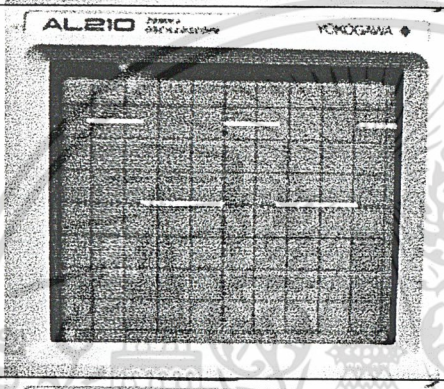
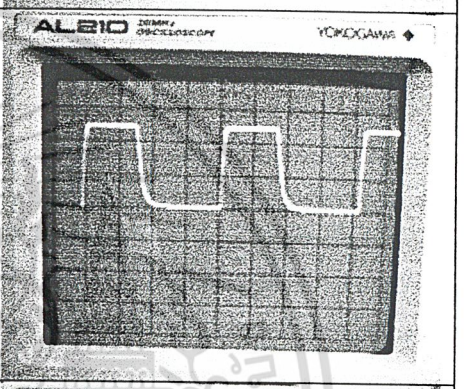
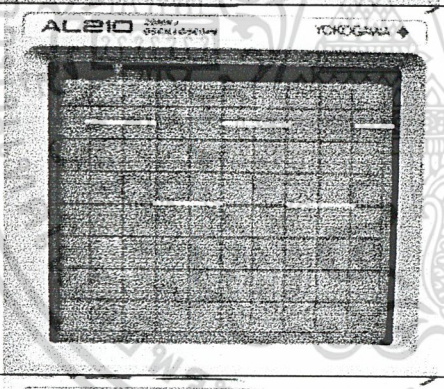
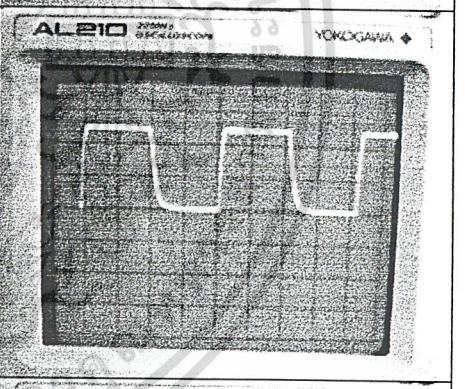
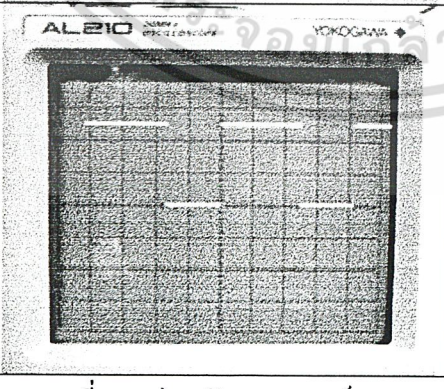
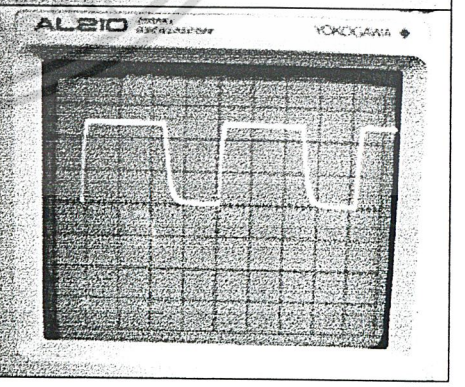
รูปที่ 4.3 โฟลวชาร์ตแสดงการควบคุมความเร็วดีซีมอเตอร์โดยโมดูล PWM

4.3.1 ผลการทดสอบการควบคุมความเร็วดีซีมอเตอร์

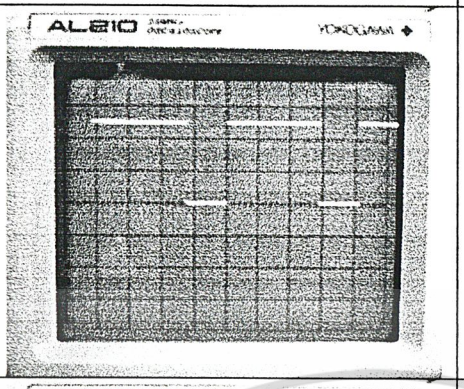
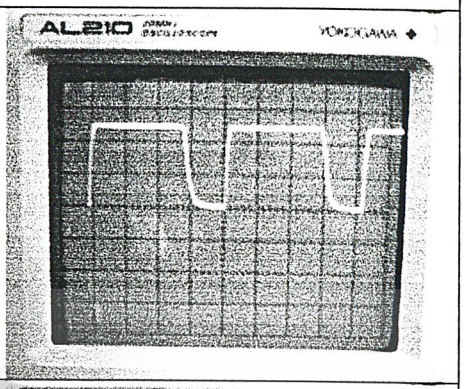
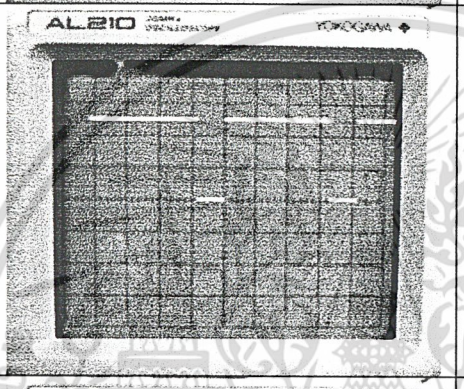
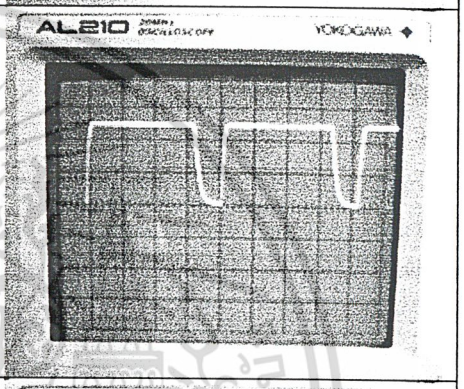
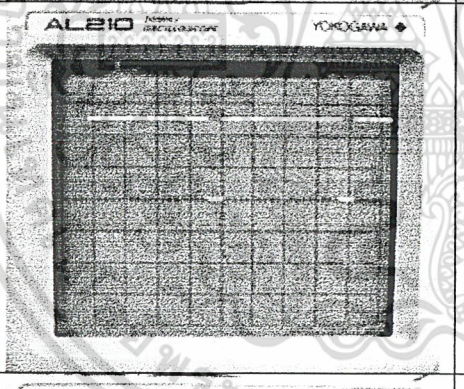
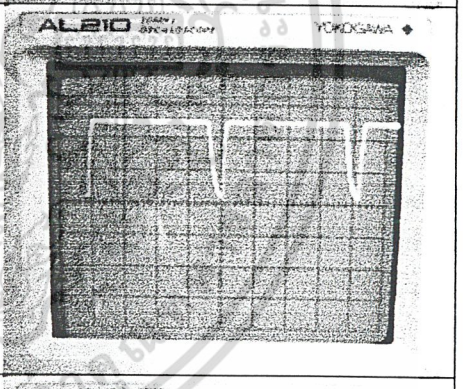
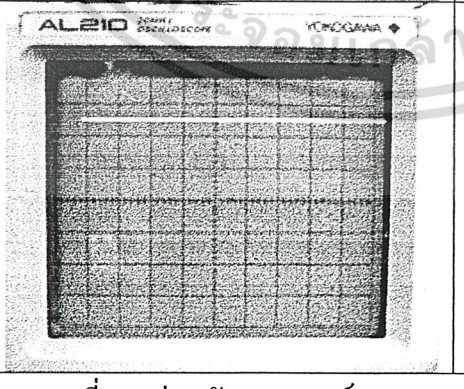
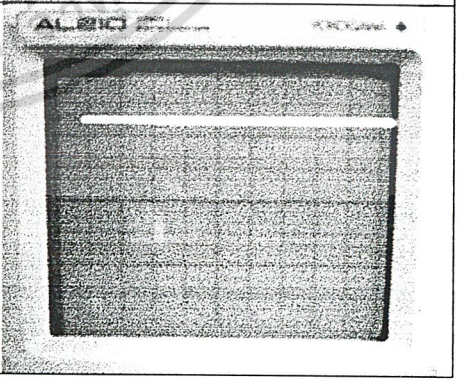
เนื่องจากการทำงานในการควบคุมความเร็วมอเตอร์ โดยใช้โมดูล PWM ค่าสัญญาณที่ผ่าน Optocoupler จะมีค่าความผิดเพี้ยนเมื่อเทียบกับสัญญาณที่ได้จาก PIC16F877 อันเนื่องมาจากค่า Rise time และ Fall time ของ Optocoupler

ค่าดิวตี้ไซเกิล	สัญญาณเอาต์พุตจาก PIC (2 volts / Div.)	สัญญาณเอาต์พุตจาก Optocoupler (2 volts / Div.)
0%		
10%		
20%		

ตารางที่ 4.1 สัญญาณเอาต์พุตจาก PIC และ Optocoupler

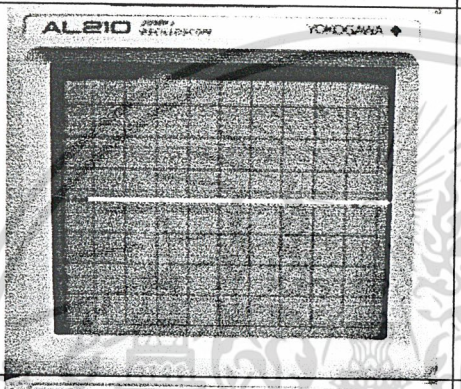
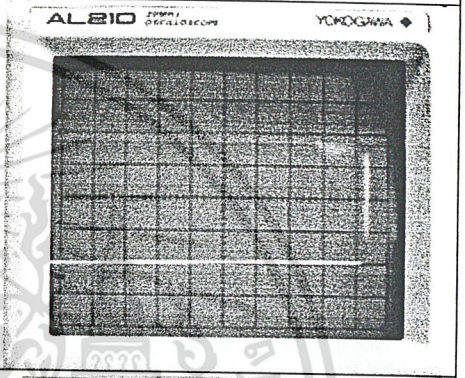
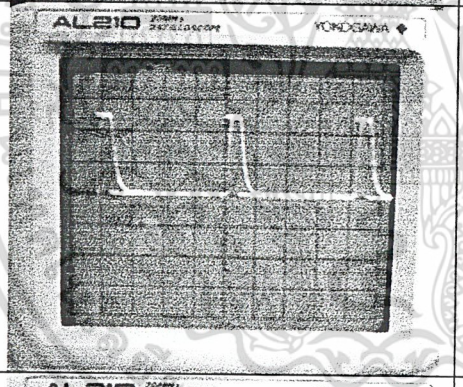
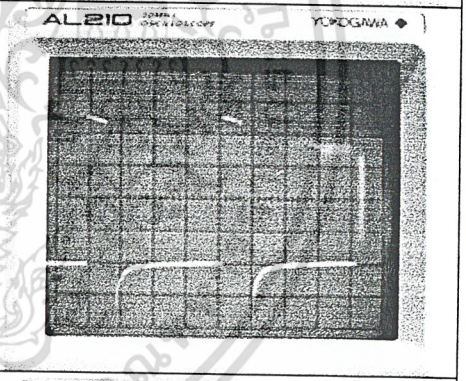
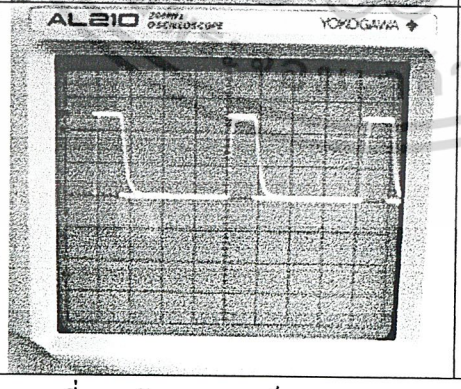
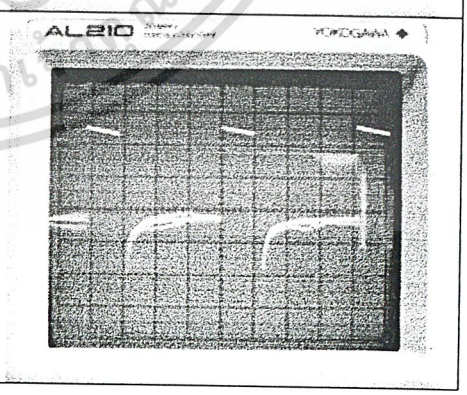
ค่าดิวตี้ไซเกิล	สัญญาณเอาต์พุตจาก PIC (2 volts / Div.)	สัญญาณเอาต์พุตจาก Optocoupler (2 volts / Div.)
30%		
40%		
50%		
60%		

ตารางที่ 4.1(ต่อ) สัญญาณเอาต์พุตจาก PIC และ Optocoupler

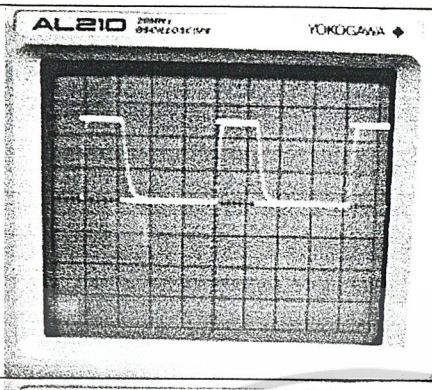
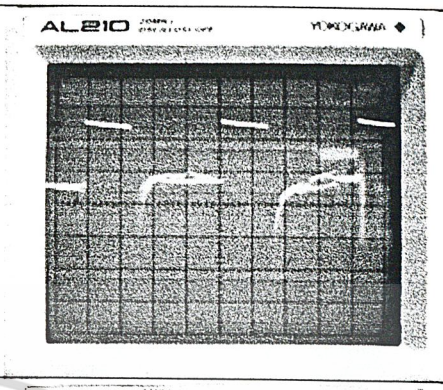
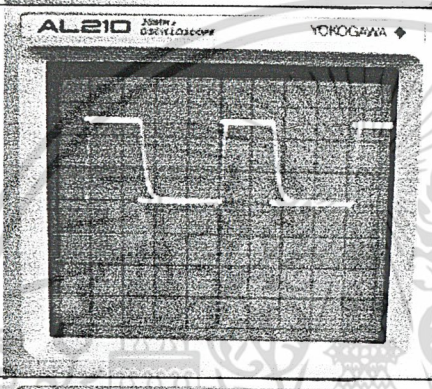
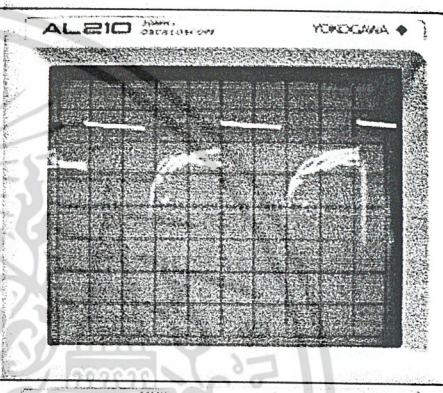
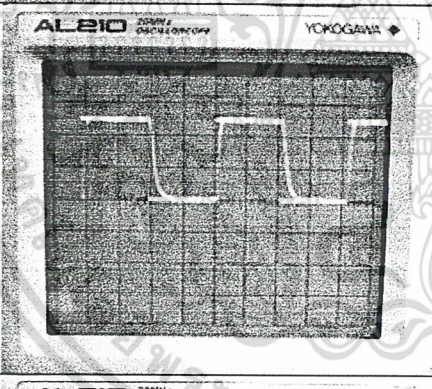
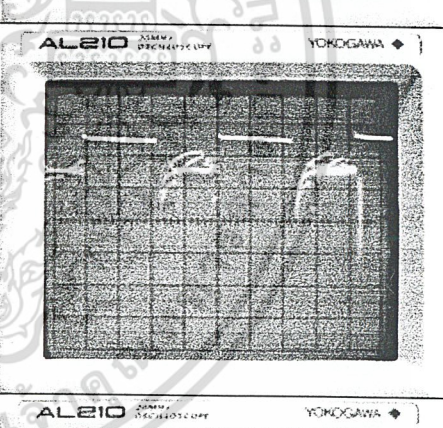
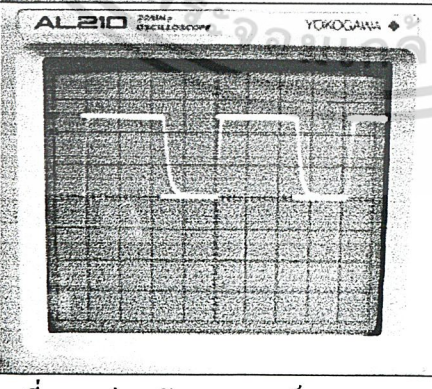
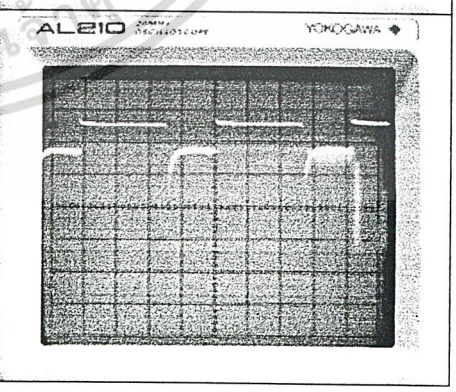
ค่าดิวตี้ไซเกิล	สัญญาณเอาต์พุตจาก PIC (2 volts / Div.)	สัญญาณเอาต์พุตจาก Optocoupler (2 volts / Div.)
70%		
80%		
90%		
100%		

ตารางที่ 4.1(ต่อ) สัญญาณเอาต์พุตจาก PIC และ Optocoupler

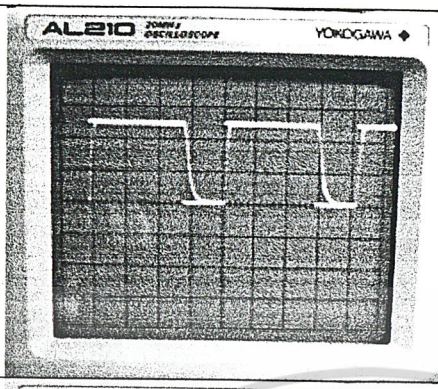
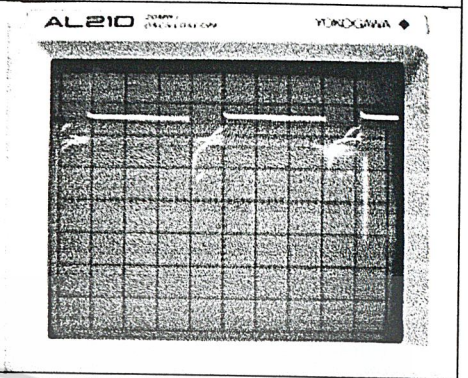
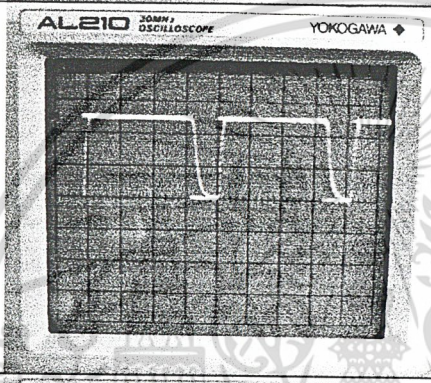
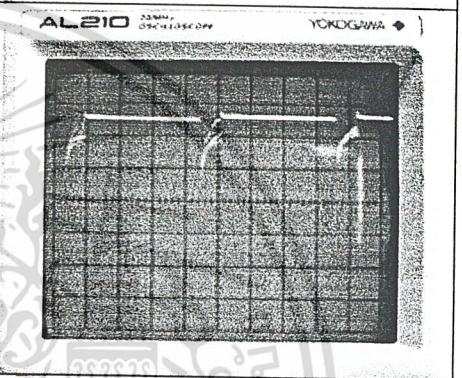
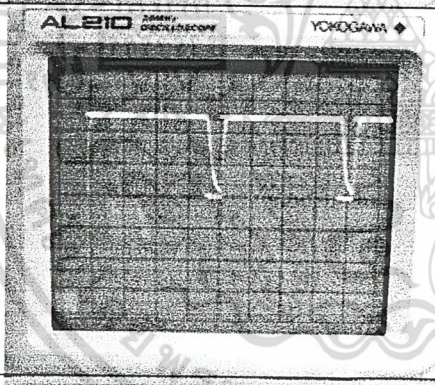
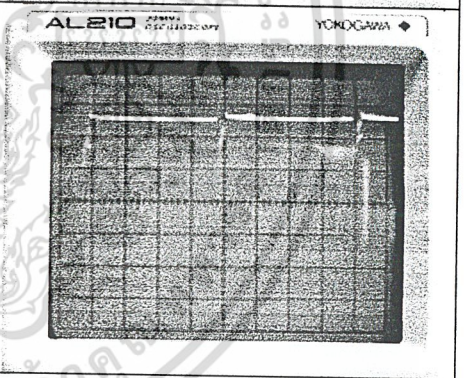
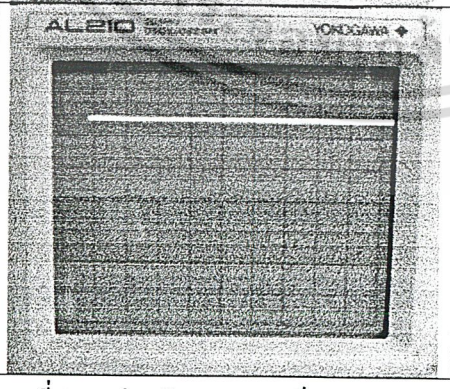
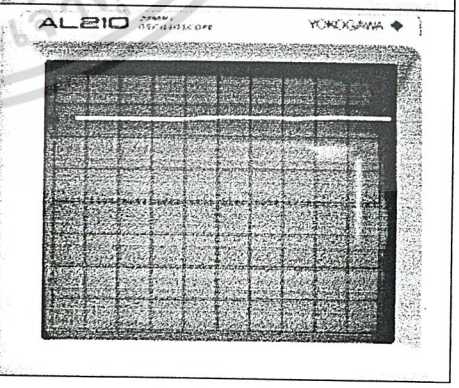
สัญญาณ PWM ขนาด 5 V จาก PIC ผ่าน Optocouplers จะเป็นอินพุทให้กับขา En ของ L298 เพื่อเป็นตัวกำหนดรูปแบบของสัญญาณ PWM ขนาด 24 V ที่เป็นเอาต์พุตของ L298 และเป็นแรงดันที่จ่ายเพื่อขับดีซีมอเตอร์ ซึ่งรูปร่างของสัญญาณเอาต์พุตจาก L298 ที่วัดค่าได้จะมีสัญญาณรบกวน อันเนื่องมาจากค่า Back EMF ที่เกิดขึ้นขณะที่มอเตอร์หมุนแบบ Free Run เมื่อสัญญาณ PWM มีค่าเป็น 0 V

ค่าดิวตี้ไซเกิล	สัญญาณเอาต์พุตจาก PIC vs Optocoupler (2 volts / Div.)	สัญญาณเอาต์พุทจาก L298 ขณะมอเตอร์ทำงาน (5 volts / Div.)
0%		
10%		
20%		

ตารางที่ 4.2 สัญญาณเอาต์พุตจาก PIC vs Optocoupler และ L298

ค่าดิวตี้ไซเกิล	สัญญาณเอาต์พุตจาก PIC vs Optocoupler (2 volts / Div.)	สัญญาณเอาต์พุทจาก L298 ขณะมอเตอร์ทำงาน (5 volts / Div.)
30%		
40%		
50%		
60%		

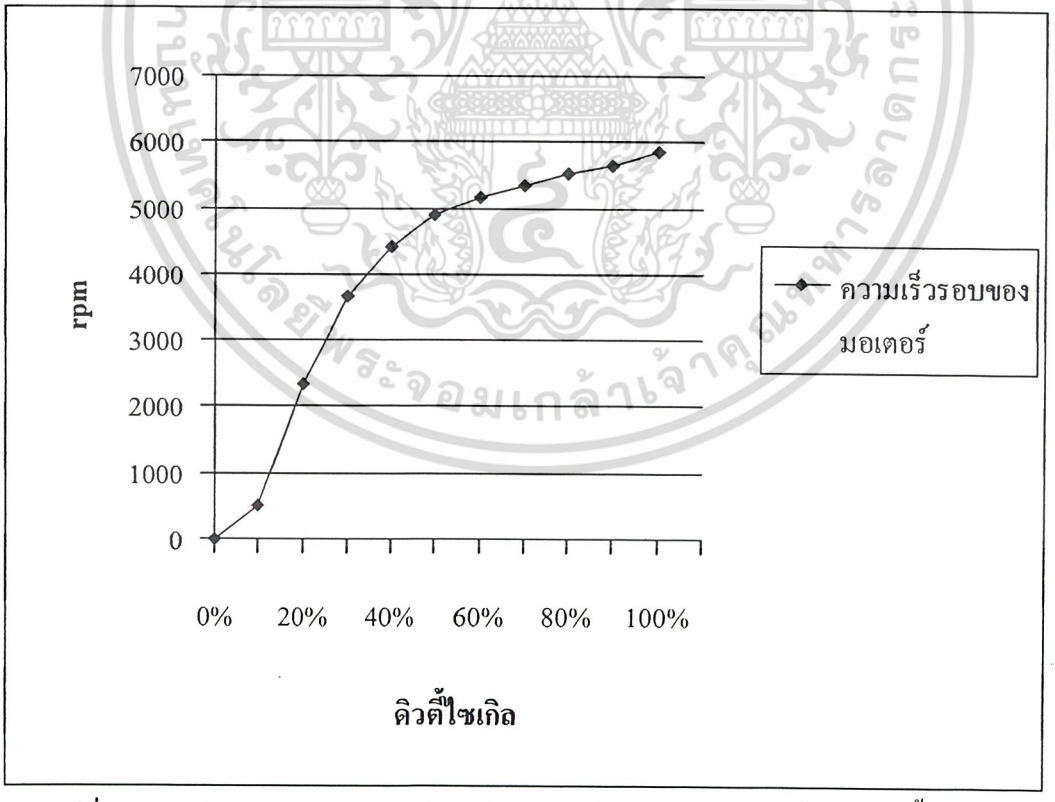
ตารางที่ 4.2 (ต่อ) สัญญาณเอาต์พุตจาก PIC vs Optocoupler และ L298

ค่าดิวตี้ไซเกิล	สัญญาณเอาต์พุตจาก PIC vs Optocoupler (2 volts / Div.)	สัญญาณเอาต์พุทจาก L298 ขณะมอเตอร์ทำงาน (5 volts / Div.)
70%		
80%		
90%		
100%		

ตารางที่ 4.2 (ต่อ)สัญญาณเอาต์พุตจาก PIC vs Optocoupler และ L298

ค่าดิวตี้ไซเกิล	แรงดันอุดมคติ (V.)	แรงดันเอาต์พุตที่ Motor (V.)	ความเร็วรอบของ Motor (rpm)
0%	0	0	0
10%	2.4	2.23	508
20%	4.8	8.74	2340
30%	7.2	13.87	3651
40%	9.6	16.75	4423
50%	12	18.55	4883
60%	14.4	19.62	5164
70%	16.8	20.06	5359
80%	19.2	21.01	5515
90%	21.6	21.51	5663
100%	24	22.42	5865

ตารางที่ 4.3 ผลการทดลองการควบคุมความเร็วดีซีมอเตอร์ โดยโมดูล PWM



รูปที่ 4.4 กราฟแสดงความสัมพันธ์ระหว่างความเร็วรอบของมอเตอร์กับค่าดิวตี้ไซเกิล

บทที่ 5
บทสรุปและวิจารณ์

5.1 บทสรุป

สำหรับโครงงานนี้จัดเป็นการศึกษาไมโครคอนโทรลเลอร์ PIC16F877 เพื่อนำมาประยุกต์ใช้งาน และออกแบบเป็นชุดปฏิบัติการในการควบคุมอุปกรณ์เอาต์พุตต่างๆ เพื่อการศึกษาและเรียนรู้การใช้งานไมโครคอนโทรลเลอร์ PIC16F877 ซึ่งถือเป็นการเรียนรู้เชิงปฏิบัติ ที่จะทำให้เข้าใจการทำงานของไมโครคอนโทรลเลอร์ได้ง่าย และรวดเร็วยิ่งขึ้น รวมทั้งยังสามารถฝึกเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ เพื่อควบคุมชุดปฏิบัติการให้เป็นไปตามที่ต้องการได้

โดยชุดทดลองที่จัดทำขึ้น ประกอบไปด้วยชุดปฏิบัติการย่อย ดังนี้

- PIC Programmer Board
- PIC16F877 Control Board
- LED & 7-Segments Board
- Relay Board
- DC Motor Board
- Matrix Switch 16 Keys

ซึ่งชุดปฏิบัติการย่อยทั้งหมด สามารถควบคุมการทำงานและติดต่อกับไมโครคอนโทรลเลอร์ PIC16F877 ได้ตามที่ต้องการ อีกทั้งยังสามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ

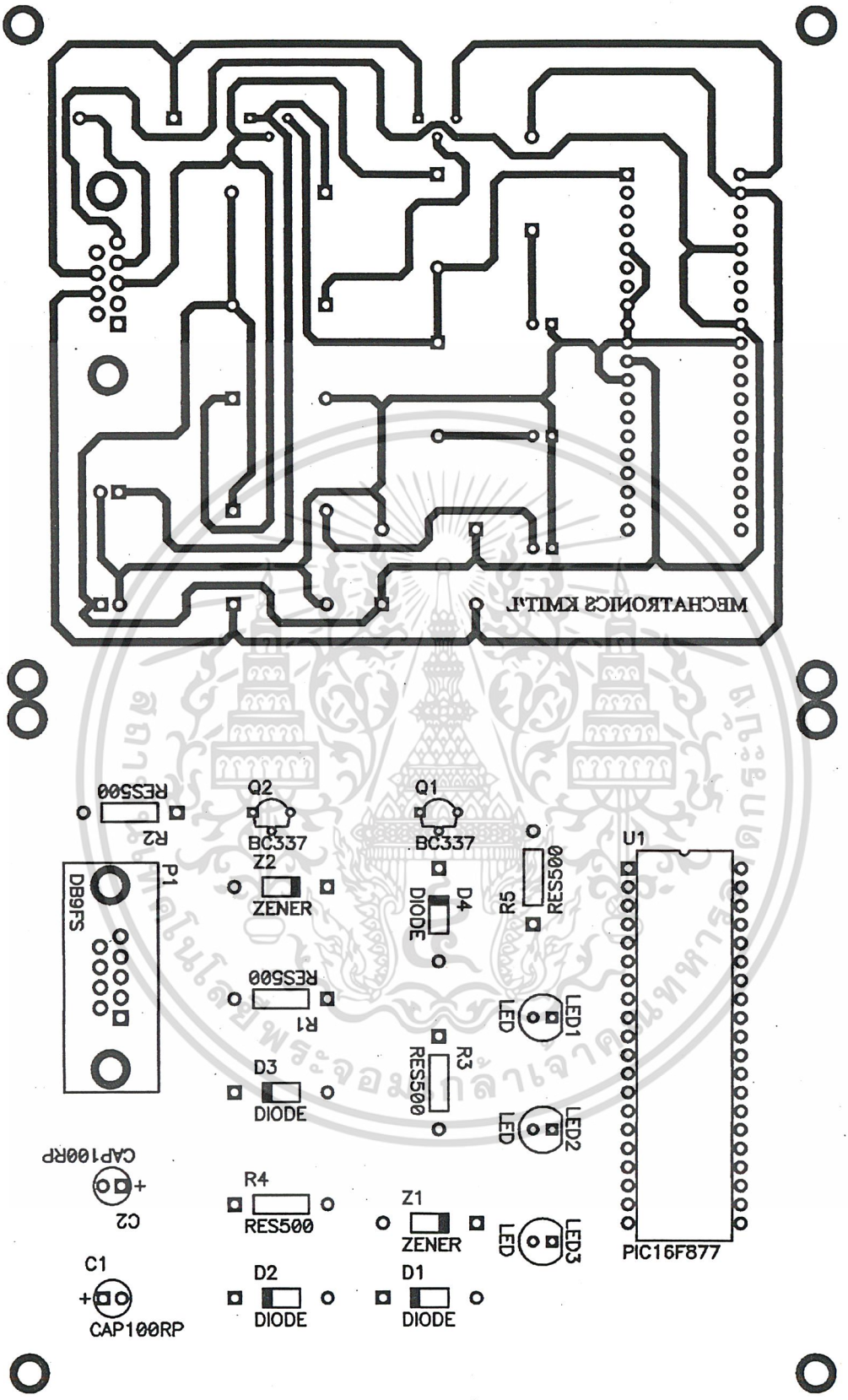
5.2 แนวทางพัฒนาต่อ

สามารถนำโครงงานนี้ไปพัฒนาเพิ่มเติม เพื่อเพิ่มประสิทธิภาพในการใช้งานเป็นชุดทดลองเพื่อการเรียนรู้ให้ได้มากยิ่งขึ้น โดย

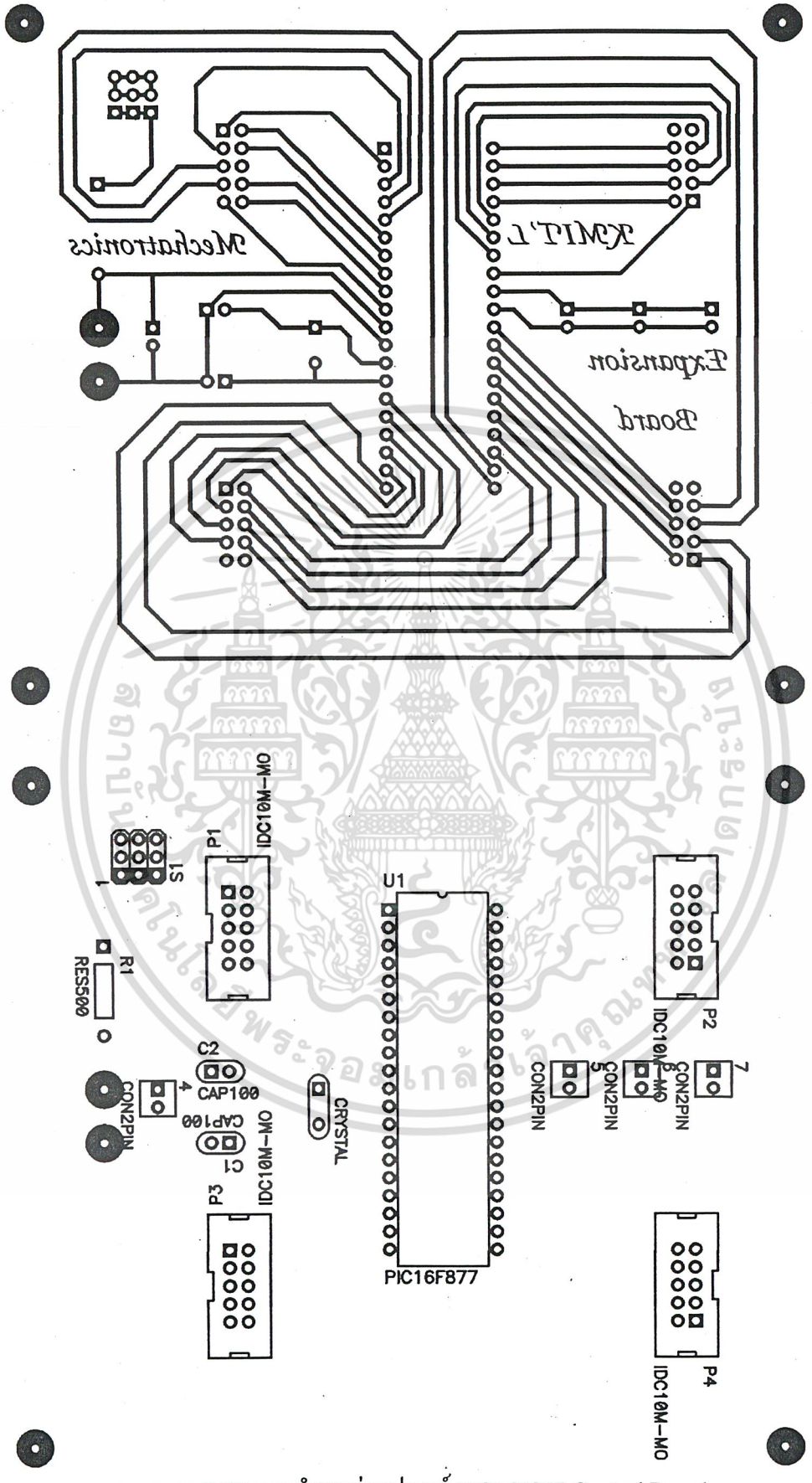
- จัดทำชุดปฏิบัติการควบคุมอุปกรณ์เอาต์พุตอื่นๆ ให้มีความหลากหลายมากยิ่งขึ้น อาทิเช่น Stepping Motor, LCD เป็นต้น
- ออกแบบให้สามารถเชื่อมต่อรับและส่งข้อมูลกับคอมพิวเตอร์ ผ่านทางพอร์ตอนุกรมได้
- จัดทำใบงานการทดลอง เพื่อใช้เป็นแนวทางในการศึกษาและเรียนรู้ สำหรับผู้ที่สนใจ

ภาคผนวก ก.

ลายวงจร PCB



ลายวงจร PCB และตำแหน่งอุปกรณ์ PIC Programmer Board



ลายวงจร PCB และตำแหน่งอุปกรณ์ PIC16F877 Control Board

ภาคผนวก ข.
โปรแกรมที่ใช้งาน

โปรแกรม ASSEMBLY วงจร 7-Segments

นับขึ้น 4 หลัก

```
list    p=16f877
#include <p16f877.inc>

__CONFIG _CP_OFF & _WDT_OFF &
_BODEN_ON & _PWRTE_ON &
_HS_OSC & _WRT_ENABLE_ON &
_LVP_OFF & _DEBUG_OFF &
_CPD_OFF

CBLOCK      0x20

DUMMY
DUMMY1
DUMMY2
DUMMY3
DIGIT0
DIGIT1
DIGIT2
DIGIT3

ENDC

        ORG     0x000
        clrf    PCLATH

        clrf    PORTA

        movlw   0x06
        movwf   ADCON1

        bsf     STATUS,RP0
        movlw   0x00
        movwf   TRISB
        movlw   b'11110000'
        movwf   TRISA
        bcf     STATUS,RP0

        movlw   b'11111111'
        movwf   PORTA
        clrf    DIGIT0
        clrf    DIGIT1
        clrf    DIGIT2
        clrf    DIGIT3
        goto    Inc1

Table_  addwf   PCL,f
        retlw   0x3f
        retlw   0x06
        retlw   0x5b
        retlw   0x4f
        retlw   0x66
        retlw   0x6d
        retlw   0x7d
        retlw   0x07
        retlw   0x7f
        retlw   0x6f
        return

Inc1    call    Delay
        movlw   0x09
```

```
        subwf   DIGIT0,w
        btfsc   STATUS,Z
        goto    Inc2
        incf    DIGIT0,f
        goto    Inc1

Inc2    clrf    DIGIT0
        movlw   0x09
        subwf   DIGIT1,w
        btfsc   STATUS,Z
        goto    Inc3
        incf    DIGIT1,f
        goto    Inc1

Inc3    clrf    DIGIT0
        clrf    DIGIT1
        movlw   0x09
        subwf   DIGIT2,w
        btfsc   STATUS,Z
        goto    Inc4
        incf    DIGIT2,f
        goto    Inc1

Inc4    clrf    DIGIT0
        clrf    DIGIT1
        clrf    DIGIT2
        movlw   0x09
        subwf   DIGIT3,w
        btfsc   STATUS,Z
        goto    Clear
        incf    DIGIT3,f
        goto    Inc1

Clear   clrf    DIGIT0
        clrf    DIGIT1
        clrf    DIGIT2
        clrf    DIGIT3
        call    Delay
        goto    $-1

Disp    movf    DIGIT0,w
        call    Table_
        movwf   PORTB
        movlw   b'11111110'
        movwf   PORTA
        call    Delay2
        movlw   b'11111111'
        movwf   PORTA

        movf    DIGIT1,w
        call    Table_
        movwf   PORTB
        movlw   b'11111101'
        movwf   PORTA
        call    Delay2
        movlw   b'11111111'
        movwf   PORTA

        movf    DIGIT2,w
        call    Table_
        movwf   PORTB
        movlw   b'11111011'
```

```
                movwf   PORTA
                call    Delay2
                movlw   b'11111111'
                movwf   PORTA

                movf    DIGIT3,w
                call    Table_
                movwf   PORTB
                movlw   b'11110111'
                movwf   PORTA
                call    Delay2
                movlw   b'11111111'
                movwf   PORTA
                return

Delay           clrf    DUMMY
Delay1          call    Disp
                decfsz  DUMMY,f
                goto    Delay1
                return
Delay2          movlw   0x01
                movwf   DUMMY1
Delay3          clrf    DUMMY2
                decfsz  DUMMY2,f
                goto    $-1
                decfsz  DUMMY1,f
                goto    Delay3
                return

        END
```

โปรแกรม ASSEMBLY วงจรรีเลย์ควบคุม

ทิศทางดีซีมอเตอร์

```
list    p=16f877
#include <p16f877.inc>

__CONFIG _CP_OFF & _WDT_OFF &
_BODEN_ON & _PWRTE_ON &
_HS_OSC & _WRT_ENABLE_ON &
_LVP_OFF & _DEBUG_OFF &
_CPD_OFF

DUMMY1      equ     0x21
DUMMY2      equ     0x22
DUMMY3      equ     0x23

            ORG     0x000
            clrf    PCLATH

            clrf    PORTB

            bsf     STATUS, RP0
            movlw   0x00
            movwf   TRISB
            bcf     STATUS,RP0
loop    movlw   b'00000010'
            movwf   PORTB
            call    Delay
            movlw   b'0000011'
            movwf   PORTB
            call    Delay
            goto    loop

Delay       movlw   0x8f
            movwf   DUMMY1
Delay2      clrf    DUMMY2
Delay3      clrf    DUMMY3
            decfsz  DUMMY3,f
            goto    $-1
            decfsz  DUMMY2,f
            goto    Delay3
            decfsz  DUMMY1,f
            goto    Delay2
            return

            END
```

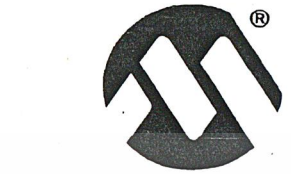
```
โปรแกรม ASSEMBLY วงจรควบคุม
ความเร็วดีซีมอเตอร์ โดยโมดูล PWM
list    p=16f877
#include <p16f877.inc>

__CONFIG _CP_OFF & _WDT_OFF &
_BODEN_ON & _PWRTE_ON &
_HS_OSC & _WRT_ENABLE_ON &
_LVP_OFF & _DEBUG_OFF &
_CPD_OFF

DUMMY1  equ  0x21
DUMMY2  equ  0x22
DUMMY3  equ  0x23
DUTY  equ 0x24

        ORG  0x000
        clrf  PCLATH
        clrf  PORTC
        bsf   STATUS, RP0
        movlw b'11111000'
        movwf TRISC
        bcf   STATUS,RP0
        clrf  CCP1CON
        bsf   STATUS,RP0
        movlw 0xff
        movwf PR2
        bcf   STATUS,RP0
        bcf   CCP1CON,4
Check0  bcf   CCP1CON,5
        movlw 0x00
        movwf CCPR1L
        movlw b'00000010'
        movwf T2CON
        clrf  TMR2
        movlw b'00001100'
        movwf CCP1CON
        bsf   T2CON,2

        call  Delay200

Check1  btfsc PORTC,4
        goto  Inc1
        goto  $-2
Check2  btfsc PORTC,4
        goto  Inc2
        goto  $-2
Check3  btfsc PORTC,4
        goto  Inc3
        goto  $-2
Check4  btfsc PORTC,4
        goto  Inc4
        goto  $-2
Check5  btfsc PORTC,4
        goto  Inc5
        goto  $-2
Check6  btfsc PORTC,4
        goto  Inc6
        goto  $-2
Check7  btfsc PORTC,4
```

```
                goto    Inc7
                goto    $-2
Check8          btfsc   PORTC,4
                goto    Inc8
                goto    $-2
Check9          btfsc   PORTC,4
                goto    Inc9
                goto    $-2
Check10         btfsc   PORTC,4
                goto    Inc10
                goto    $-2

Inc1            movlw   0x1a
                movwf   CCPR1L
                call    Delay200
                goto    Check2
Inc2            movlw   0x33
                movwf   CCPR1L
                call    Delay200
                goto    Check3
Inc3            movlw   0x4d
                movwf   CCPR1L
                call    Delay200
                goto    Check4
Inc4            movlw   0x66
                movwf   CCPR1L
                call    Delay200
                goto    Check5
Inc5            movlw   0x80
                movwf   CCPR1L
                call    Delay200
                goto    Check6
Inc6            movlw   0x99
                movwf   CCPR1L
                call    Delay200
                goto    Check7
Inc7            movlw   0xb3
                movwf   CCPR1L
                call    Delay200
                goto    Check8
Inc8            movlw   0xcc
                movwf   CCPR1L
                call    Delay200
                goto    Check9
Inc9            movlw   0xe5
                movwf   CCPR1L
                call    Delay200
                goto    Check10
Inc10           movlw   0xff
                movwf   CCPR1L
                call    Delay200
                btfsc   PORTC,4
                goto    Check0
                goto    $-2

Delay200        movlw   0x0a
                movwf   DUMMY1
Delay2          clrf    DUMMY2
Delay3          clrf    DUMMY3
                decfsz  DUMMY3,f
                goto    $-1
                decfsz  DUMMY2,f
```

```
        goto    Delay3
        decfsz  DUMMY1,f
        goto    Delay2
        return

END
```

ภาคผนวก ค.

DATA SHEET



MICROCHIP

PIC16F87X
Data Sheet

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

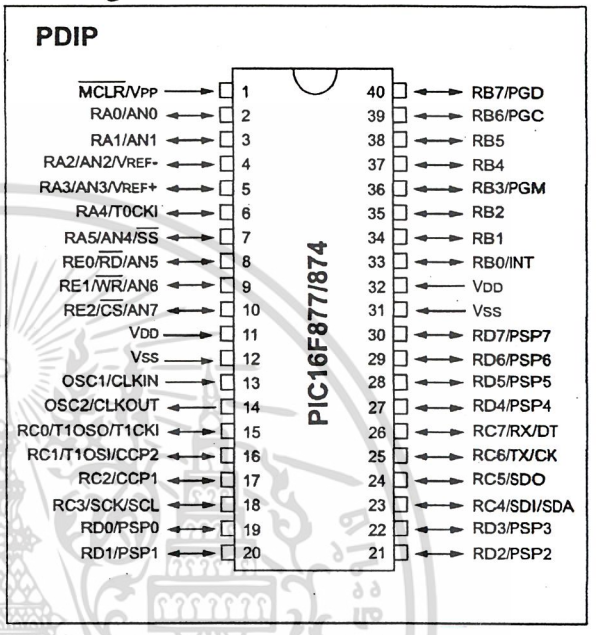
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 μA typical @ 3V, 32 kHz
  - < 1 μA typical standby current

Pin Diagram

PDIP

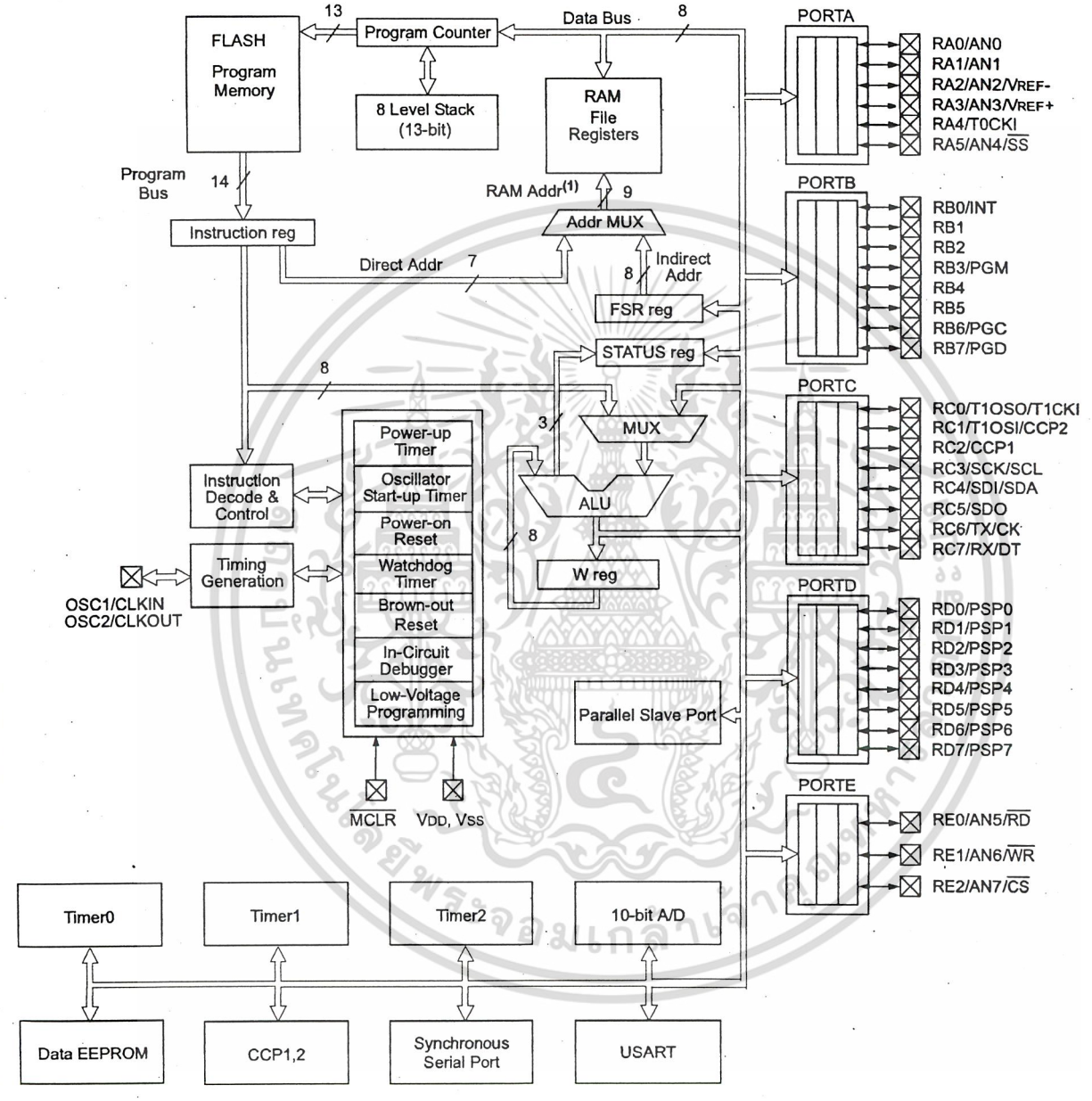


Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected, These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

2.0 MEMORY ORGANIZATION

There are three memory blocks in each of the PIC16F87X MCUs. The Program Memory and Data Memory have separate buses so that concurrent access can occur and is detailed in this section. The EEPROM data memory block is detailed in Section 4.0.

Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

2.1 Program Memory Organization

The PIC16F87X devices have a 13-bit program counter capable of addressing an 8K x 14 program memory space. The PIC16F877/876 devices have 8K x 14 words of FLASH program memory, and the PIC16F873/874 devices have 4K x 14. Accessing a location above the physically implemented address will cause a wraparound.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PIC16F877/876 PROGRAM MEMORY MAP AND STACK

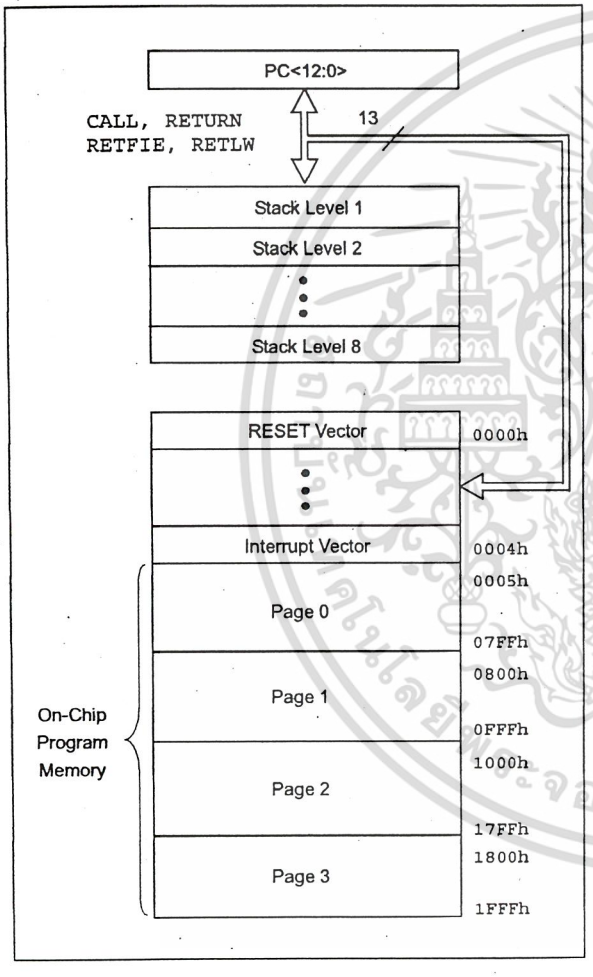
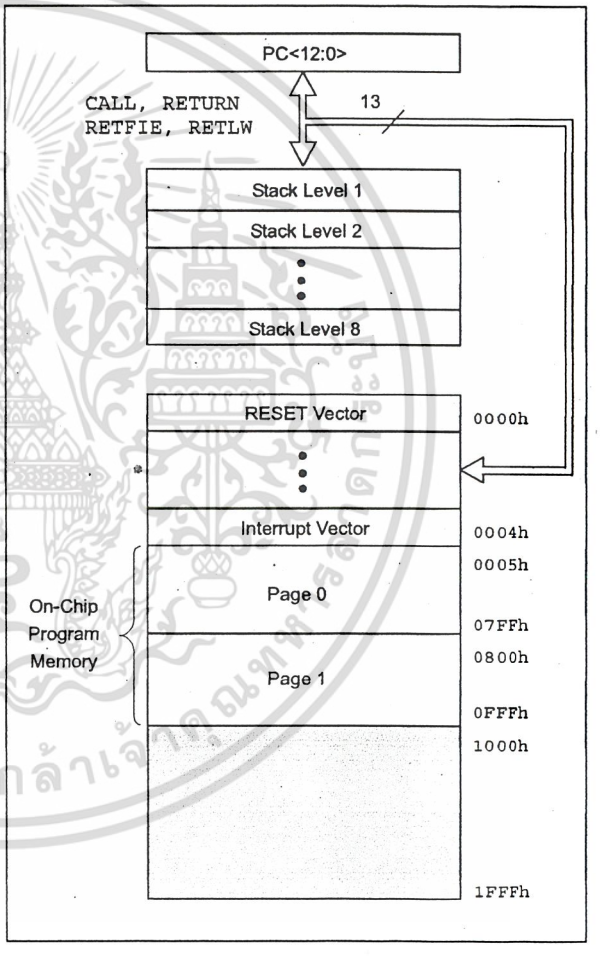


FIGURE 2-2: PIC16F874/873 PROGRAM MEMORY MAP AND STACK



2.2 Data Memory Organization

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 (STATUS<6>) and RP0 (STATUS<5>) are the bank select bits.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

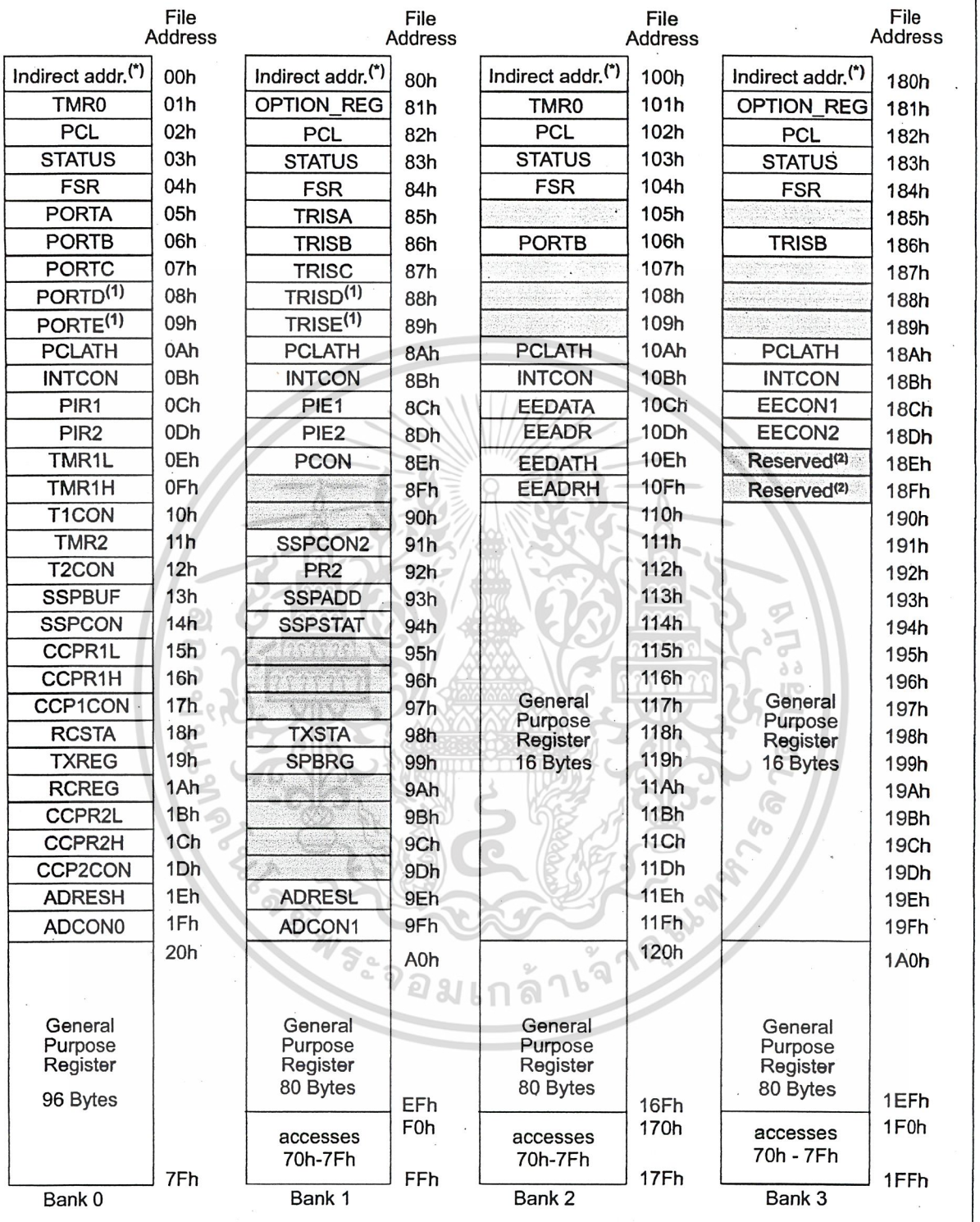
Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

Note: EEPROM Data Memory description can be found in Section 4.0 of this data sheet.

2.2.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly, or indirectly through the File Select Register (FSR).

FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP



Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
01h	TMR0	Timer0 Module Register								xxxx xxxx	47
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	29
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	33
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	35
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	36
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
0Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	22
0Dh	PIR2	—	(5)	—	EEIF	BCLIF	—	—	CCP2IF	-r-0 0--0	24
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	51
11h	TMR2	Timer2 Module Register								0000 0000	55
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	55
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	70, 73
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	67
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	57
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	57
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	58
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	96
19h	TXREG	USART Transmit Data Register								0000 0000	99
1Ah	RCREG	USART Receive Data Register								0000 0000	101
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	57
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	57
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	58
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	116
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	111

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	29
86h	TRISB	PORTB Data Direction Register								1111 1111	31
87h	TRISC	PORTC Data Direction Register								1111 1111	33
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	35
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000. -111	37
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0. 0000	26
8Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	21
8Dh	PIE2	—	(5)	—	EEIE	BCLIE	—	—	CCP2IE	-r-0 0--0	23
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	25
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	68
92h	PR2	Timer2 Period Register								1111 1111	55
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	73, 74
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	66
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	95
99h	SPBRG	Baud Rate Generator Register								0000 0000	97
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	—	Unimplemented								—	—
9Dh	—	Unimplemented								—	—
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	116
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000	112

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
101h	TMR0	Timer0 Module Register								xxxx xxxx	47
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	26
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
10Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	41
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	41
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						xxxx xxxx	41
10Fh	EEADRH	—	—	—	EEPROM Address Register High Byte					xxxx xxxx	41
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	31
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
18Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	41, 42
18Dh	EECON2	EEPROM Control Register2 (not a physical register)								---- ----	41
18Eh	—	Reserved maintain clear								0000 0000	—
18Fh	—	Reserved maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

2.2.2.1 STATUS Register

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory.

The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable, therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C or DC bits from the STATUS register. For other instructions not affecting any status bits, see the "Instruction Set Summary."

Note:	The C and DC bits operate as a $\overline{borrow}$ and digit $\overline{borrow}$ bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0:** Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

bit 4 **$\overline{TO}$:** Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$:** Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
(for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C:** Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

3.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

3.1 PORTA and the TRISA Register

PORTA is a 6-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, the value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

Other PORTA pins are multiplexed with analog inputs and analog VREF input. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

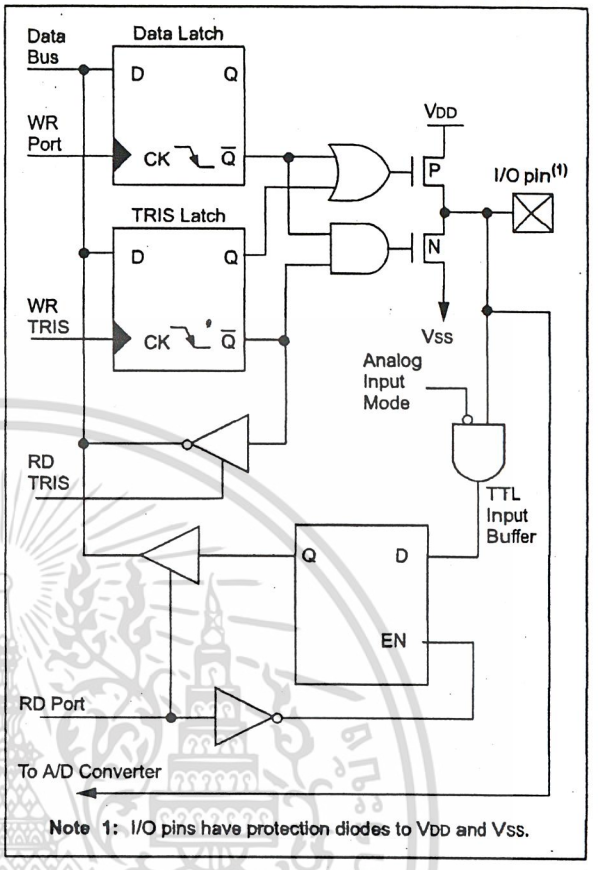
Note:	On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 3-1: INITIALIZING PORTA

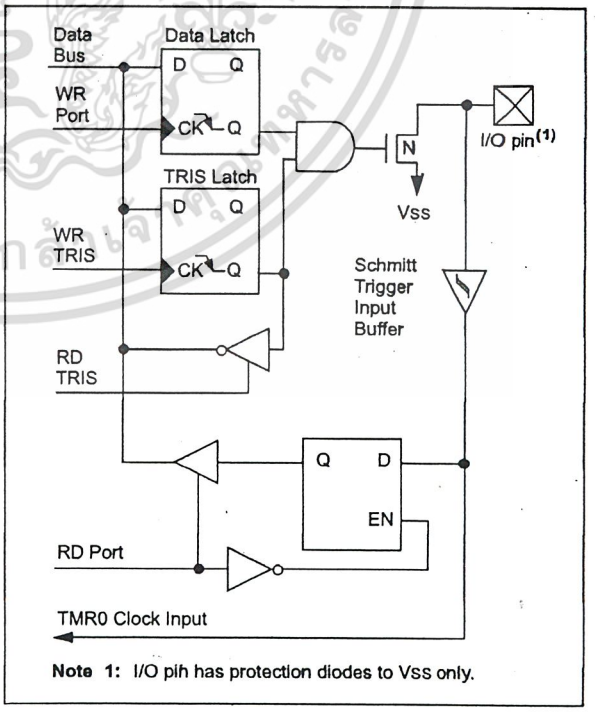
```
BCF     STATUS, RP0   ;
BCF     STATUS, RP1   ; Bank0
CLRF    PORTA         ; Initialize PORTA by
                      ; clearing output
                      ; data latches
BSF     STATUS, RP0   ; Select Bank 1
MOVLW   0x06          ; Configure all pins
MOVWF   ADCON1        ; as digital inputs
MOVLW   0xCF          ; Value used to
                      ; initialize data
                      ; direction
MOVWF   TRISA         ; Set RA<3:0> as inputs
                      ; RA<5:4> as outputs
                      ; TRISA<7:6>are always
                      ; read as '0'.
```

FIGURE 3-1: BLOCK DIAGRAM OF RA3:RA0 AND RA5 PINS



Note 1: I/O pins have protection diodes to VDD and Vss.

FIGURE 3-2: BLOCK DIAGRAM OF RA4/T0CKI PIN



Note 1: I/O pih has protection diodes to Vss only.

TABLE 3-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input.
RA1/AN1	bit1	TTL	Input/output or analog input.
RA2/AN2	bit2	TTL	Input/output or analog input.
RA3/AN3/VREF	bit3	TTL	Input/output or analog input or VREF.
RA4/T0CKI	bit4	ST	Input/output or external clock input for Timer0. Output is open drain type.
RA5/$\overline{SS}$/AN4	bit5	TTL	Input/output or slave select input for synchronous serial port or analog input.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 3-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by PORTA.

Note: When using the SSP module in SPI Slave mode and $\overline{SS}$ enabled, the A/D converter must be set to one of the following modes, where PCFG3:PCFG0 = 0100, 0101, 011x, 1101, 1110, 1111.

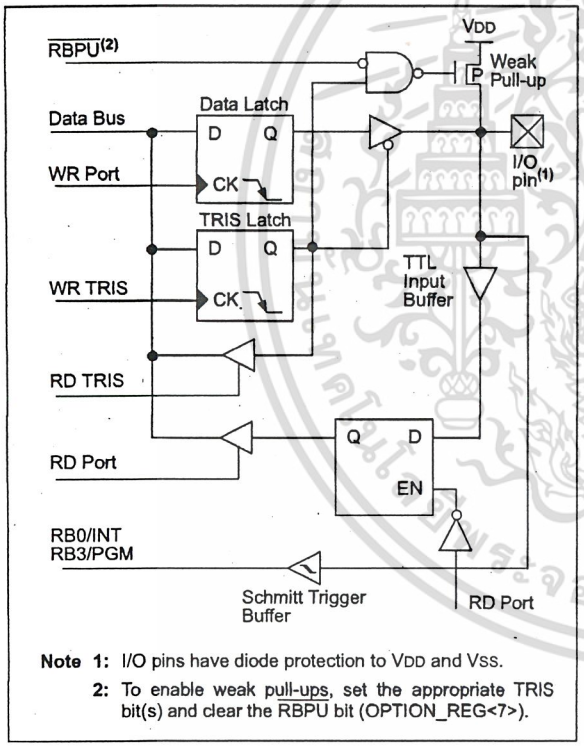
3.2 PORTB and the TRISB Register

PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Three pins of PORTB are multiplexed with the Low Voltage Programming function: RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in the Special Features Section.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit $\overline{\text{RBPU}}$ (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

FIGURE 3-3: BLOCK DIAGRAM OF RB3:RB0 PINS



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

Four of the PORTB pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.

b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

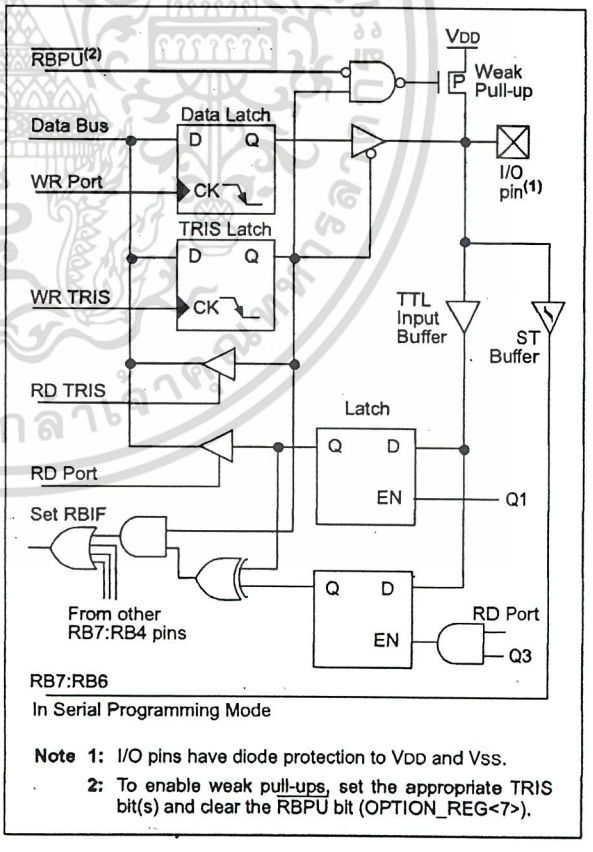
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

This interrupt-on-mismatch feature, together with software configureable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression. Refer to the Embedded Control Handbook, *"Implementing Wake-up on Key Strokes"* (AN552).

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

RB0/INT is discussed in detail in Section 12.10.1.

FIGURE 3-4: BLOCK DIAGRAM OF RB7:RB4 PINS



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

TABLE 3-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT	bit0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM(3)	bit3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/PGC	bit6	TTL/ST(2)	Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit7	TTL/ST(2)	Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.

2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.

3: Low Voltage ICSP Programming (LVP) is enabled by default, which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

TABLE 3-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

3.3 PORTC and the TRISC Register

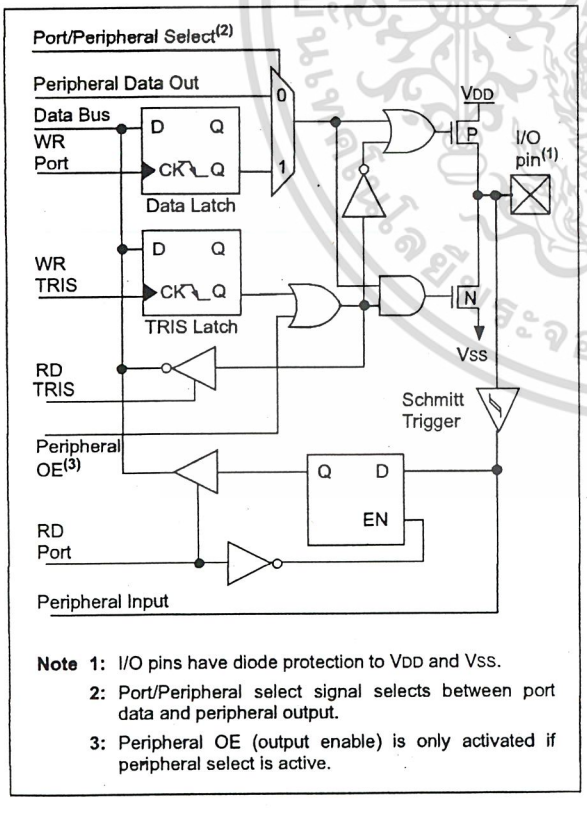
PORTC is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

PORTC is multiplexed with several peripheral functions (Table 3-5). PORTC pins have Schmitt Trigger input buffers.

When the I^2C module is enabled, the PORTC<4:3> pins can be configured with normal I^2C levels, or with SMBus levels by using the CKE bit (SSPSTAT<6>).

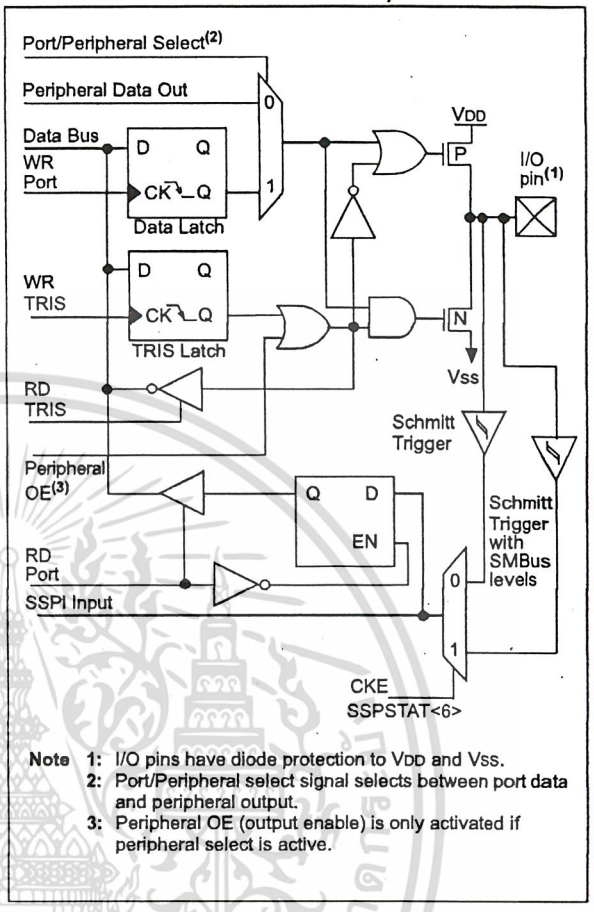
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (`BSF`, `BCF`, `XORWF`) with TRISC as destination, should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

FIGURE 3-5: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<2:0>, RC<7:5>



Note 1: I/O pins have diode protection to VDD and VSS.
2: Port/Peripheral select signal selects between port data and peripheral output.
3: Peripheral OE (output enable) is only activated if peripheral select is active.

FIGURE 3-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<4:3>



Note 1: I/O pins have diode protection to VDD and VSS.
2: Port/Peripheral select signal selects between port data and peripheral output.
3: Peripheral OE (output enable) is only activated if peripheral select is active.

TABLE 3-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit6	ST	Input/output port pin or USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	bit7	ST	Input/output port pin or USART Asynchronous Receive or Synchronous Data.

Legend: ST = Schmitt Trigger input

TABLE 3-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

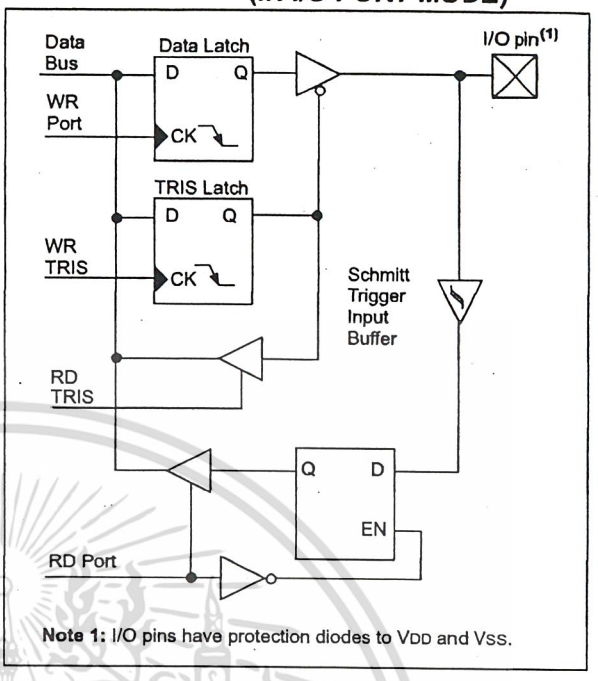
3.4 PORTD and TRISD Registers

PORTD and TRISD are not implemented on the PIC16F873 or PIC16F876.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configureable as an input or output.

PORTD can be configured as an 8-bit wide microprocessor port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

FIGURE 3-7: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 3-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL(1)	Input/output port pin or parallel slave port bit0.
RD1/PSP1	bit1	ST/TTL(1)	Input/output port pin or parallel slave port bit1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or parallel slave port bit2.
RD3/PSP3	bit3	ST/TTL(1)	Input/output port pin or parallel slave port bit3.
RD4/PSP4	bit4	ST/TTL(1)	Input/output port pin or parallel slave port bit4.
RD5/PSP5	bit5	ST/TTL(1)	Input/output port pin or parallel slave port bit5.
RD6/PSP6	bit6	ST/TTL(1)	Input/output port pin or parallel slave port bit6.
RD7/PSP7	bit7	ST/TTL(1)	Input/output port pin or parallel slave port bit7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 3-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

3.5 PORTE and TRISE Register

PORTE and TRISE are not implemented on the PIC16F873 or PIC16F876.

PORTE has three pins (RE0/$\overline{RD}$/AN5, RE1/$\overline{WR}$/AN6, and RE2/$\overline{CS}$/AN7) which are individually configureable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make certain that the TRISE<2:0> bits are set, and that the pins are configured as digital inputs. Also ensure that ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

Register 3-1 shows the TRISE register, which also controls the parallel slave port operation.

PORTE pins are multiplexed with analog inputs. When selected for analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Note: On a Power-on Reset, these pins are configured as analog inputs, and read as '0'.

FIGURE 3-8: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)

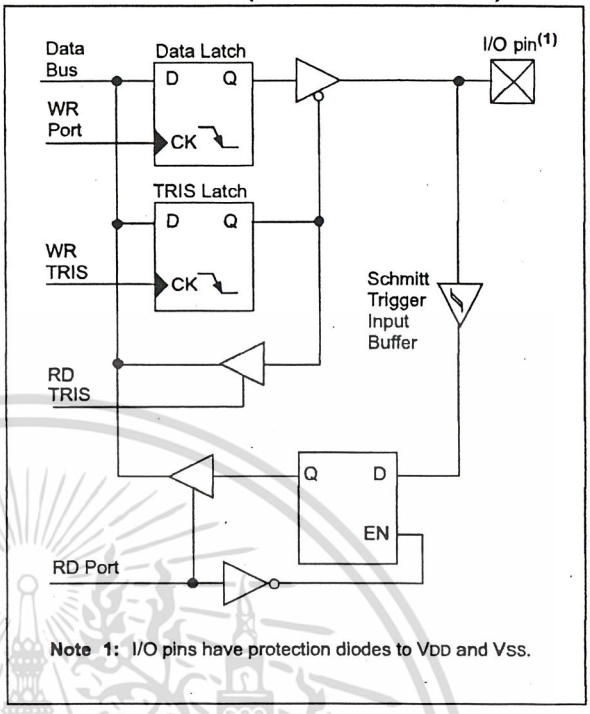


TABLE 3-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected)
RE1/$\overline{WR}$/AN6	bit1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected)
RE2/$\overline{CS}$/AN7	bit2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 3-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

REGISTER 3-1: TRISE REGISTER (ADDRESS 89h)

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	Bit2	Bit1	Bit0
bit 7							bit 0

Parallel Slave Port Status/Control Bits:

bit 7 **IBF:** Input Buffer Full Status bit
1 = A word has been received and is waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF:** Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV:** Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE:** Parallel Slave Port Mode Select bit
1 = PORTD functions in Parallel Slave Port mode
0 = PORTD functions in general purpose I/O mode

bit 3 **Unimplemented:** Read as '0'

PORTE Data Direction Bits:

bit 2 **Bit2:** Direction Control bit for pin RE2/$\overline{CS}$/AN7
1 = Input
0 = Output

bit 1 **Bit1:** Direction Control bit for pin RE1/$\overline{WR}$/AN6
1 = Input
0 = Output

bit 0 **Bit0:** Direction Control bit for pin RE0/$\overline{RD}$/AN5
1 = Input
0 = Output

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

5.0 TIMER0 MODULE

The Timer0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Figure 5-1 is a block diagram of the Timer0 module and the prescaler shared with the WDT.

Additional information on the Timer0 module is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

Timer mode is selected by clearing bit T0CS (OPTION_REG<5>). In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

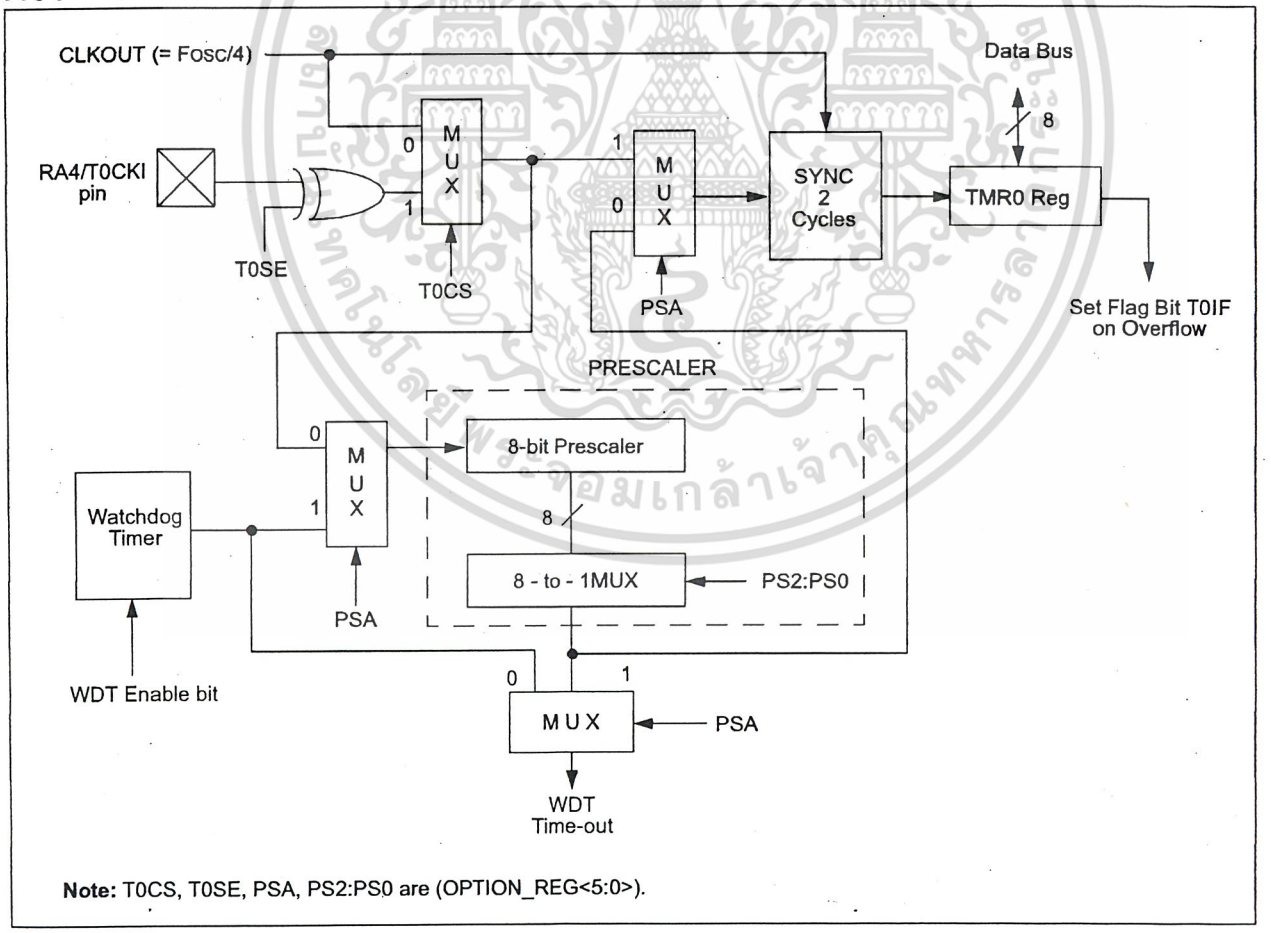
Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In Counter mode, Timer0 will increment either on every rising, or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (OPTION_REG<4>). Clearing bit T0SE selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 5.2.

The prescaler is mutually exclusively shared between the Timer0 module and the Watchdog Timer. The prescaler is not readable or writable. Section 5.3 details the operation of the prescaler.

5.1 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). The interrupt can be masked by clearing bit T0IE (INTCON<5>). Bit T0IF must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP, since the timer is shut-off during SLEEP.

FIGURE 5-1: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

5.2 Using Timer0 with an External Clock

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks. Therefore, it is necessary for T0CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

5.3 Prescaler

There is only one prescaler available, which is mutually exclusively shared between the Timer0 module and the Watchdog Timer. A prescaler assignment for the Timer0 module means that there is no prescaler for the Watchdog Timer, and vice-versa. This prescaler is not readable or writable (see Figure 5-1).

The PSA and PS2:PS0 bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g. `CLRF 1`, `MOVWF 1`, `BSF 1,x`....etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

Note: Writing to TMR0, when the prescaler is assigned to Timer0, will clear the prescaler count, but will not change the prescaler assignment.

REGISTER 5-1: OPTION_REG REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**

bit 6 **INTEDG**

bit 5 **T0CS:** TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE:** TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0:** Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: To avoid an unintended device RESET, the instruction sequence shown in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

TABLE 5-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
01h,101h	TMR0	Timer0 Module's Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by Timer0.

6.0 TIMER1 MODULE

The Timer1 module is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L), which are readable and writable. The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR1 Interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit TMR1IE (PIE1<0>).

Timer1 can operate in one of two modes:

- As a timer
- As a counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

In Timer mode, Timer1 increments every instruction cycle. In Counter mode, it increments on every rising edge of the external clock input.

Timer1 can be enabled/disabled by setting/clearing control bit TMR1ON (T1CON<0>).

Timer1 also has an internal "RESET input". This RESET can be generated by either of the two CCP modules (Section 8.0). Register 6-1 shows the Timer1 control register.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI/CCP2 and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored, and these pins read as '0'.

Additional information on timer modules is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

REGISTER 6-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Control bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

6.1 Timer1 Operation in Timer Mode

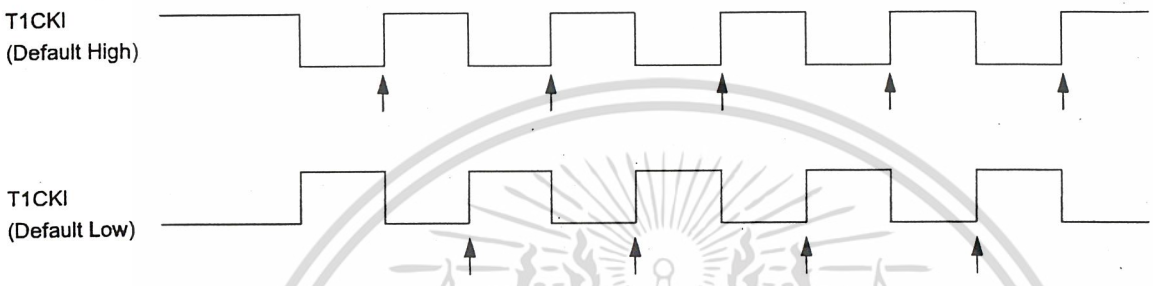
Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is Fosc/4. The synchronize control bit $\overline{\text{T1SYNC}}$ (T1CON<2>) has no effect, since the internal clock is always in sync.

6.2 Timer1 Counter Operation

Timer1 may operate in either a Synchronous, or an Asynchronous mode, depending on the setting of the TMR1CS bit.

When Timer1 is being incremented via an external source, increments occur on a rising edge. After Timer1 is enabled in Counter mode, the module must first have a falling edge before the counter begins to increment.

FIGURE 6-1: TIMER1 INCREMENTING EDGE



Note: Arrows indicate counter increments.

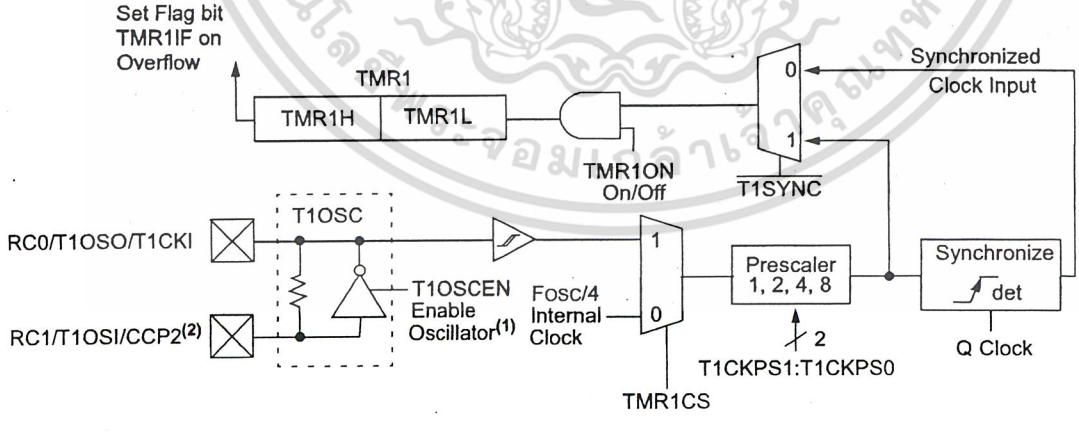
6.3 Timer1 Operation in Synchronized Counter Mode

Counter mode is selected by setting bit TMR1CS. In this mode, the timer increments on every rising edge of clock input on pin RC1/T1OSI/CCP2, when bit T1OSCEN is set, or on pin RC0/T1OSO/T1CKI, when bit T1OSCEN is cleared.

If $\overline{\text{T1SYNC}}$ is cleared, then the external clock input is synchronized with internal phase clocks. The synchronization is done after the prescaler stage. The prescaler stage is an asynchronous ripple-counter.

In this configuration, during SLEEP mode, Timer1 will not increment even if the external clock is present, since the synchronization circuit is shut-off. The prescaler, however, will continue to increment.

FIGURE 6-2: TIMER1 BLOCK DIAGRAM



Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

6.4 Timer1 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{T1SYNC}}$ (T1CON<2>) is set, the external clock input is not synchronized. The timer continues to increment asynchronous to the internal phase clocks. The timer will continue to run during SLEEP and can generate an interrupt-on-overflow, which will wake-up the processor. However, special precautions in software are needed to read/write the timer (Section 6.4.1).

In Asynchronous Counter mode, Timer1 cannot be used as a time-base for capture or compare operations.

6.4.1 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L while the timer is running from an external asynchronous clock, will guarantee a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the timer register.

Reading the 16-bit value requires some care. Examples 12-2 and 12-3 in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023) show how to read and write Timer1 when it is running in Asynchronous mode.

6.5 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON<3>). The oscillator is a low power oscillator, rated up to 200 kHz. It will continue to run during SLEEP. It is primarily intended for use with a 32 kHz crystal. Table 6-1 shows the capacitor selection for the Timer1 oscillator.

The Timer1 oscillator is identical to the LP oscillator. The user must provide a software time delay to ensure proper oscillator start-up.

TABLE 6-1: CAPACITOR SELECTION FOR THE TIMER1 OSCILLATOR

Osc Type	Freq.	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF
These values are for design guidance only.			
Crystals Tested:			
32.768 kHz	Epson C-001R32.768K-A		± 20 PPM
100 kHz	Epson C-2 100.00 KC-P		± 20 PPM
200 kHz	STD XTL 200.000 kHz		± 20 PPM

Note 1: Higher capacitance increases the stability of oscillator, but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

6.6 Resetting Timer1 using a CCP Trigger Output

If the CCP1 or CCP2 module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer1.

Note: The special event triggers from the CCP1 and CCP2 modules will not set interrupt flag bit TMR1IF (PIR1<0>).

Timer1 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this RESET operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1 or CCP2, the write will take precedence.

In this mode of operation, the CCPRxH:CCPRxL register pair effectively becomes the period register for Timer1.

6.7 Resetting of Timer1 Register Pair (TMR1H, TMR1L)

TMR1H and TMR1L registers are not reset to 00h on a POR, or any other RESET, except by the CCP1 and CCP2 special event triggers.

T1CON register is reset to 00h on a Power-on Reset, or a Brown-out Reset, which shuts off the timer and leaves a 1:1 prescale. In all other RESETS, the register is unaffected.

6.8 Timer1 Prescaler

The prescaler counter is cleared on writes to the TMR1H or TMR1L registers.

TABLE 6-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

7.0 TIMER2 MODULE

Timer2 is an 8-bit timer with a prescaler and a postscaler. It can be used as the PWM time-base for the PWM mode of the CCP module(s). The TMR2 register is readable and writable, and is cleared on any device RESET.

The input clock (Fosc/4) has a prescale option of 1:1, 1:4, or 1:16, selected by control bits T2CKPS1:T2CKPS0 (T2CON<1:0>).

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon RESET.

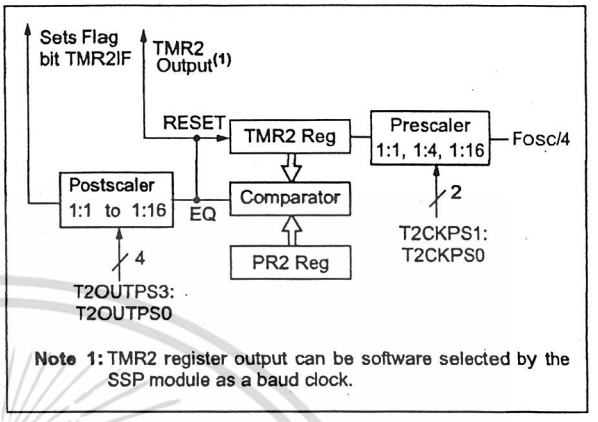
The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit TMR2IF, (PIR1<1>)).

Timer2 can be shut-off by clearing control bit TMR2ON (T2CON<2>), to minimize power consumption.

Register 7-1 shows the Timer2 control register.

Additional information on timer modules is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

FIGURE 7-1: TIMER2 BLOCK DIAGRAM



Note 1: TMR2 register output can be software selected by the SSP module as a baud clock.

REGISTER 7-1: T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
0010 = 1:3 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.1 Timer2 Prescaler and Postscaler

The prescaler and postscaler counters are cleared when any of the following occurs:

- a write to the TMR2 register
- a write to the T2CON register
- any device RESET (POR, $\overline{MCLR}$ Reset, WDT Reset, or BOR)

TMR2 is not cleared when T2CON is written.

7.2 Output of TMR2

The output of TMR2 (before the postscaler) is fed to the SSP module, which optionally uses it to generate shift clock.

TABLE 7-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
11h	TMR2	Timer2 Module's Register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

8.0 CAPTURE/COMPARE/PWM MODULES

Each Capture/Compare/PWM (CCP) module contains a 16-bit register which can operate as a:

- 16-bit Capture register
- 16-bit Compare register
- PWM Master/Slave Duty Cycle register

Both the CCP1 and CCP2 modules are identical in operation, with the exception being the operation of the special event trigger. Table 8-1 and Table 8-2 show the resources and interactions of the CCP module(s). In the following sections, the operation of a CCP module is described with respect to CCP1. CCP2 operates the same as CCP1, except where noted.

CCP1 Module:

Capture/Compare/PWM Register1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. The special event trigger is generated by a compare match and will reset Timer1.

CCP2 Module:

Capture/Compare/PWM Register2 (CCPR2) is comprised of two 8-bit registers: CCPR2L (low byte) and CCPR2H (high byte). The CCP2CON register controls the operation of CCP2. The special event trigger is generated by a compare match and will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

Additional information on CCP modules is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023) and in application note AN594, "Using the CCP Modules" (DS00594).

TABLE 8-1: CCP MODE - TIMER RESOURCES REQUIRED

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

TABLE 8-2: INTERACTION OF TWO CCP MODULES

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time-base
Capture	Compare	The compare should be configured for the special event trigger, which clears TMR1
Compare	Compare	The compare(s) should be configured for the special event trigger, which clears TMR1
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt)
PWM	Capture	None
PWM	Compare	None

REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS: 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPxX:CCPxY:** PWM Least Significant bits

Capture mode:
Unused

Compare mode:
Unused

PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0:** CCPx Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

8.3 PWM Mode (PWM)

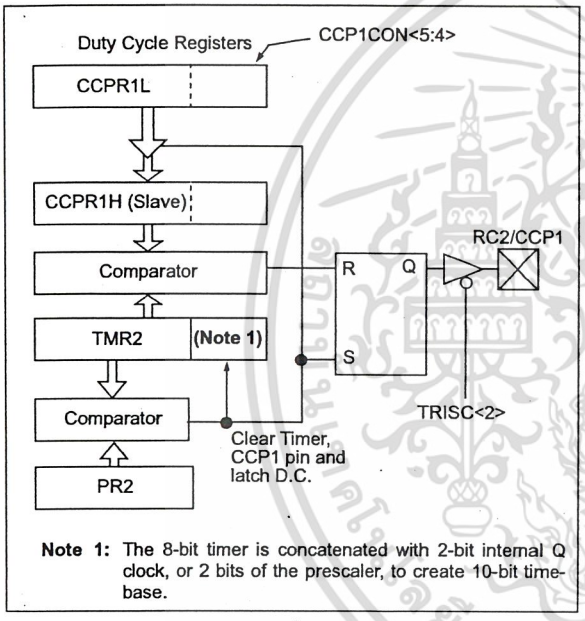
In Pulse Width Modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Note:	Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 8-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see Section 8.3.3.

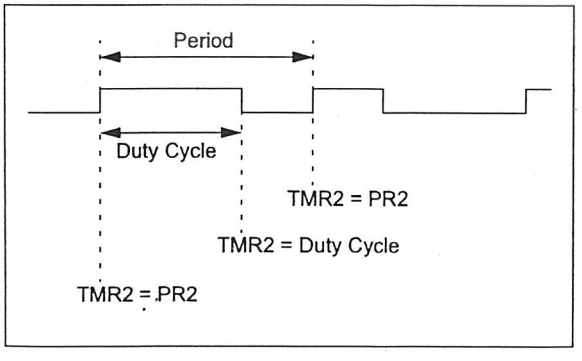
FIGURE 8-3: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: The 8-bit timer is concatenated with 2-bit internal Q clock, or 2 bits of the prescaler, to create 10-bit time-base.

A PWM output (Figure 8-4) has a time-base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 8-4: PWM OUTPUT



8.3.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

PWM frequency is defined as 1 / [PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note:	The Timer2 postscaler (see Section 7.1) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

8.3.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitch-free PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock, or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the formula:

$$\text{Resolution} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

8.3.3 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the TRISC<2> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

TABLE 8-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 20 MHz

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12kHz	156.3 kHz	208.3 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFFh	0xFFh	0xFFh	0x3Fh	0x1Fh	0x17h
Maximum Resolution (bits)	10	10	10	8	7	5.5

TABLE 8-4: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, AND TIMER1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by Capture and Timer1.
Note 1: The PSP is not implemented on the PIC16F873/876; always maintain these bits clear.

TABLE 8-5: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- ---0	---- ---0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
11h	TMR2	Timer2 Module's Register								0000 0000	0000 0000
92h	PR2	Timer2 Module's Period Register								1111 1111	1111 1111
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PWM and Timer2.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

13.1 Instruction Descriptions

ADDLW	Add Literal and W
Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

ANDLW	AND Literal with W
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.

ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

CALL	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	($\bar{f}$) → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

DECFSZ **Decrement f, Skip if 0**

Syntax: [*label*] DECFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - 1 → (destination);
skip if result = 0

Status Affected: None

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
If the result is 1, the next instruction is executed. If the result is 0, then a NOP is executed instead making it a 2TCY instruction.

GOTO **Unconditional Branch**

Syntax: [*label*] GOTO k

Operands: $0 \le k \le 2047$

Operation: k → PC<10:0>
PCLATH<4:3> → PC<12:11>

Status Affected: None

Description: GOTO is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction.

INCF **Increment f**

Syntax: [*label*] INCF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → (destination)

Status Affected: Z

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

INCFSZ **Increment f, Skip if 0**

Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → (destination),
skip if result = 0

Status Affected: None

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2TCY instruction.

IORLW **Inclusive OR Literal with W**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → (W)

Status Affected: Z

Description: The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.

IORWF **Inclusive OR W with f**

Syntax: [*label*] IORWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) .OR. (f) → (destination)

Status Affected: Z

Description: Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.

MOVLW	Move Literal to W
Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.

NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

RETFIE	Return from Interrupt
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None

RETLW	Return with Literal in W
Syntax:	[*label*] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W); TOS → PC
Status Affected:	None
Description:	The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

RLF **Rotate Left f through Carry**

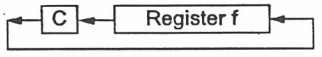
Syntax: [*label*] RLF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.



SLEEP

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Description: The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped.

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

SUBLW **Subtract W from Literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Description: The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register.

RRF **Rotate Right f through Carry**

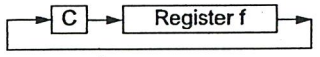
Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.



SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - (W) → (destination)

Status Affected: C, DC, Z

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

SWAPF **Swap Nibbles in f**

Syntax:	[*label*] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed in register 'f'.

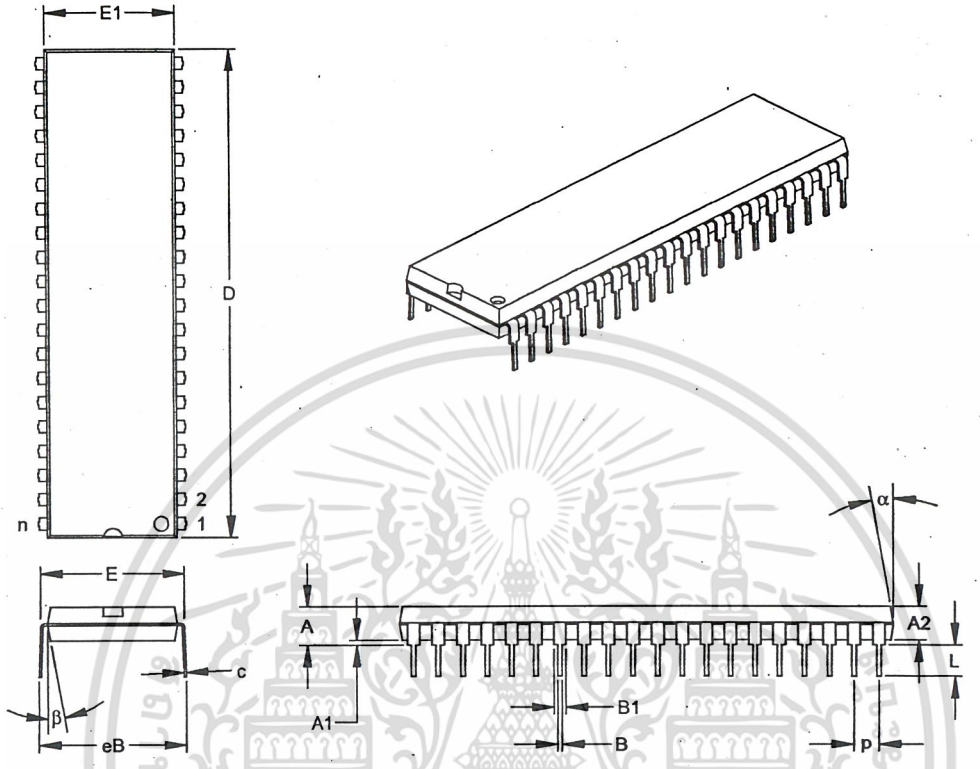
XORLW **Exclusive OR Literal with W**

Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR'ed with the eight-bit literal 'k'. The result is placed in the W register.

XORWF **Exclusive OR W with f**

Syntax:	[*label*] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

40-Lead Plastic Dual In-line (P) – 600 mil (PDIP)



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		40			40	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.160	.175	.190	4.06	4.45	4.83
Molded Package Thickness	A2	.140	.150	.160	3.56	3.81	4.06
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.595	.600	.625	15.11	15.24	15.88
Molded Package Width	E1	.530	.545	.560	13.46	13.84	14.22
Overall Length	D	2.045	2.058	2.065	51.94	52.26	52.45
Tip to Seating Plane	L	.120	.130	.135	3.05	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.030	.050	.070	0.76	1.27	1.78
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56
Overall Row Spacing §	eB	.620	.650	.680	15.75	16.51	17.27
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

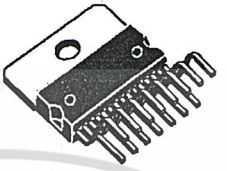
* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-011
Drawing No. C04-016

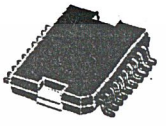
L298

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



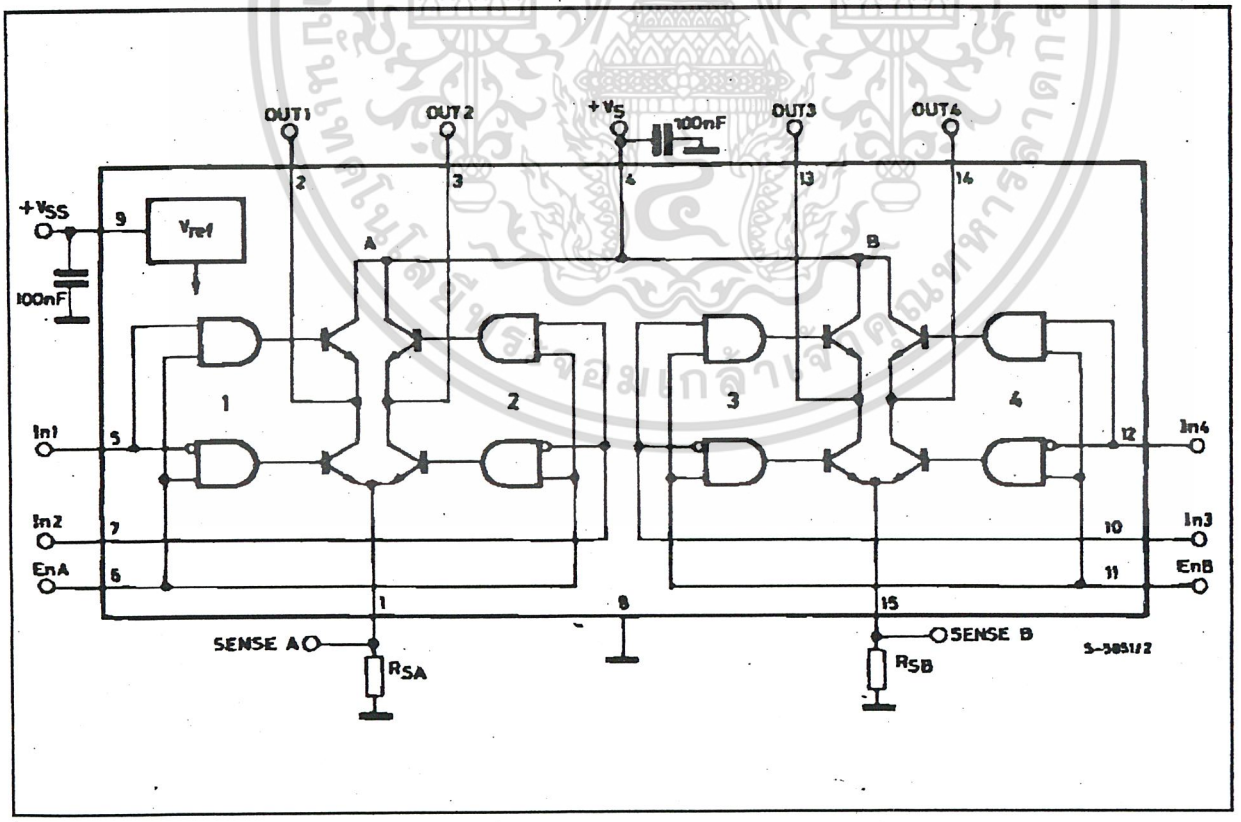
PowerSO20

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.

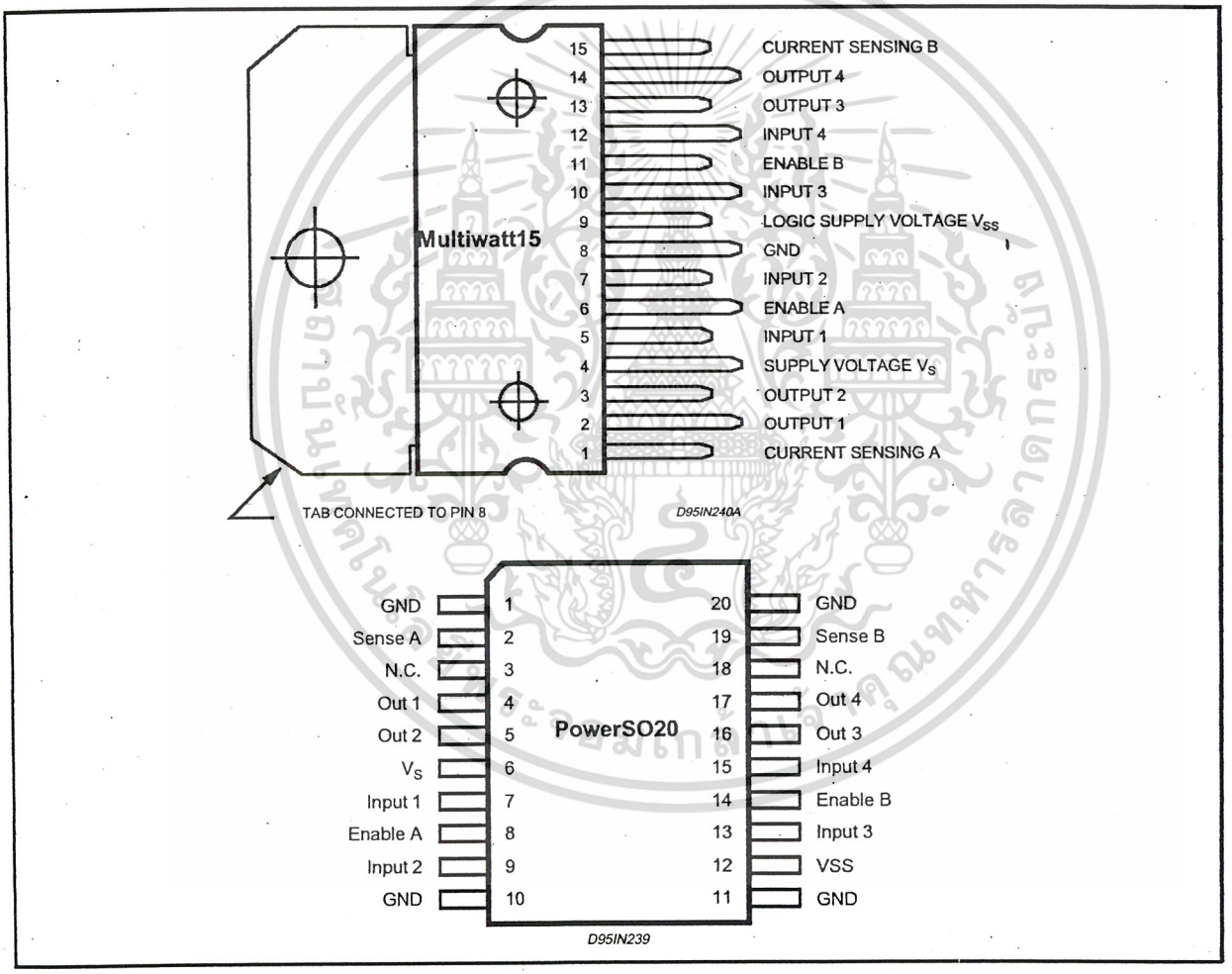
BLOCK DIAGRAM



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		13	22	mA
		V_i = H		50	70	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		24	36	mA
		V_i = H		7	12	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	0.95	1.35	1.7	V
		I_L = 2A		2	2.7	V
$V_{CEsat(L)}$	Sink Saturation Voltage	I_L = 1A (5)	0.85	1.2	1.6	V
		I_L = 2A (5)		1.7	2.3	V
V_{CEsat}	Total Drop	I_L = 1A (5)	1.80		3.2	V
		I_L = 2A (5)			4.9	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.

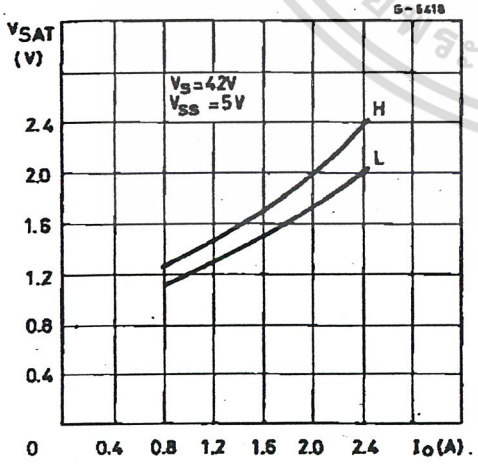
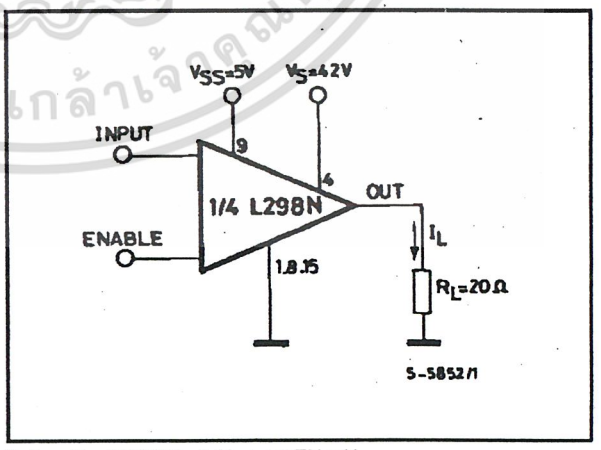


Figure 2 : Switching Times Test Circuits.



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

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

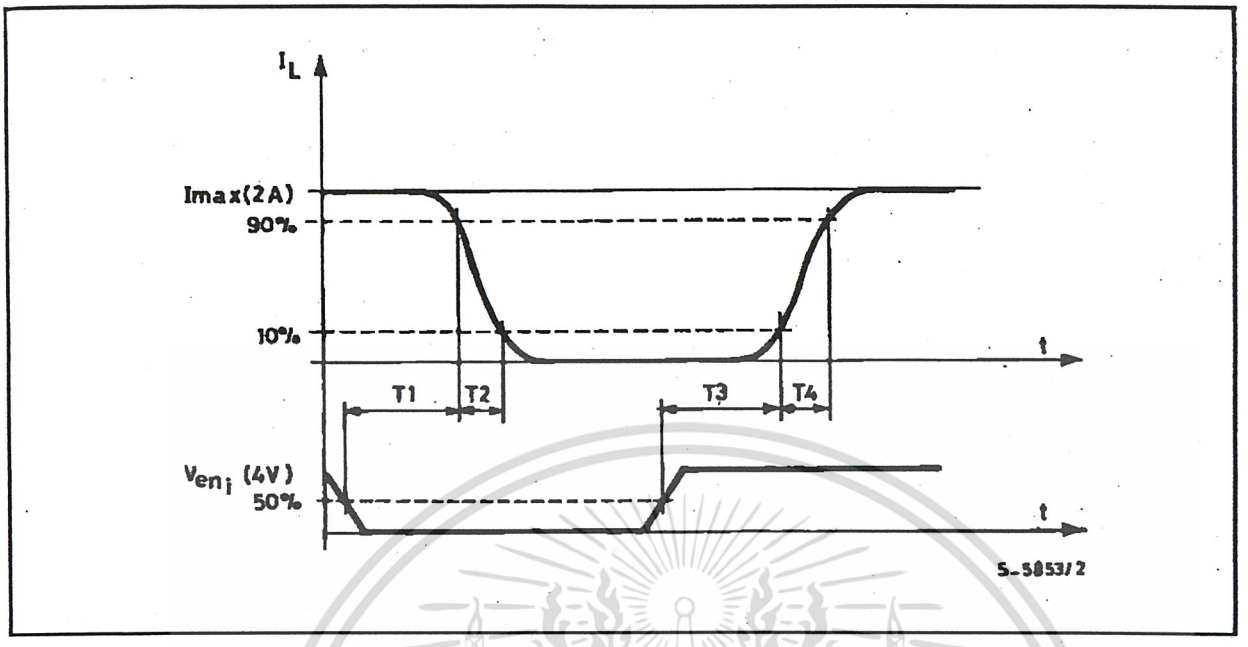
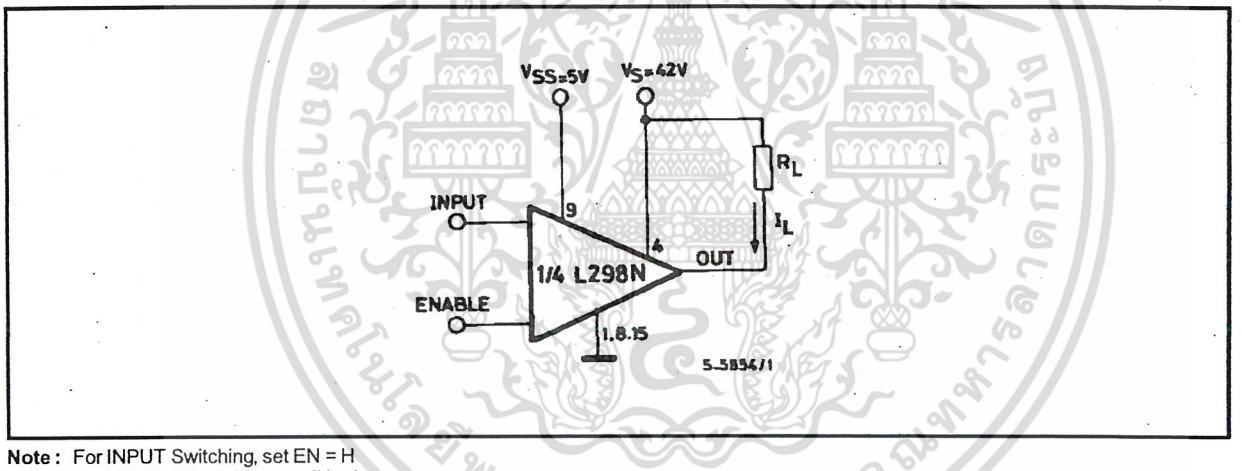


Figure 4 : Switching Times Test Circuits.



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.

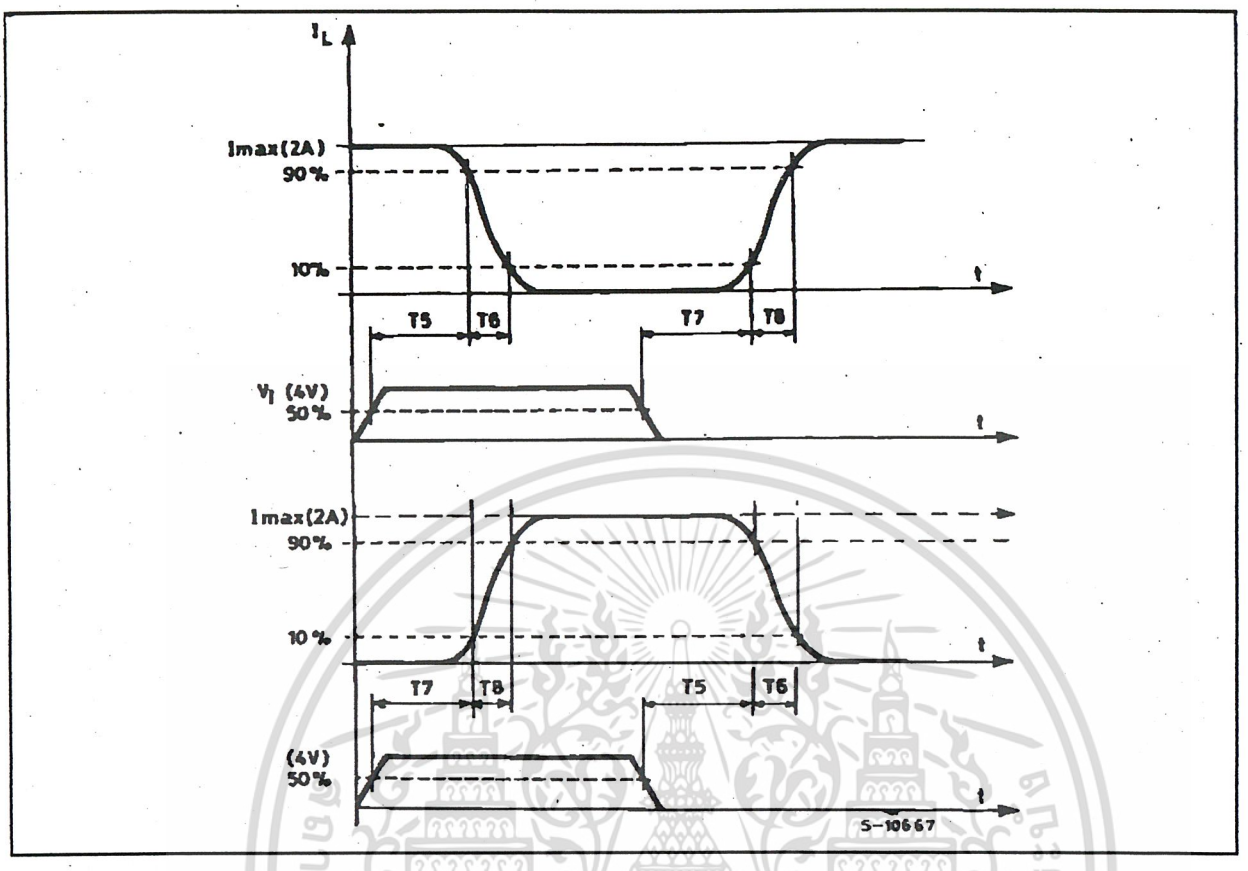
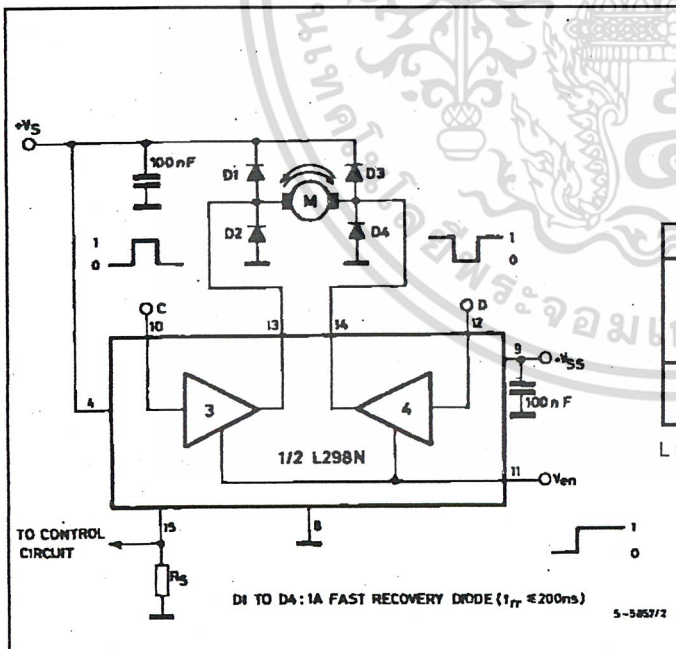


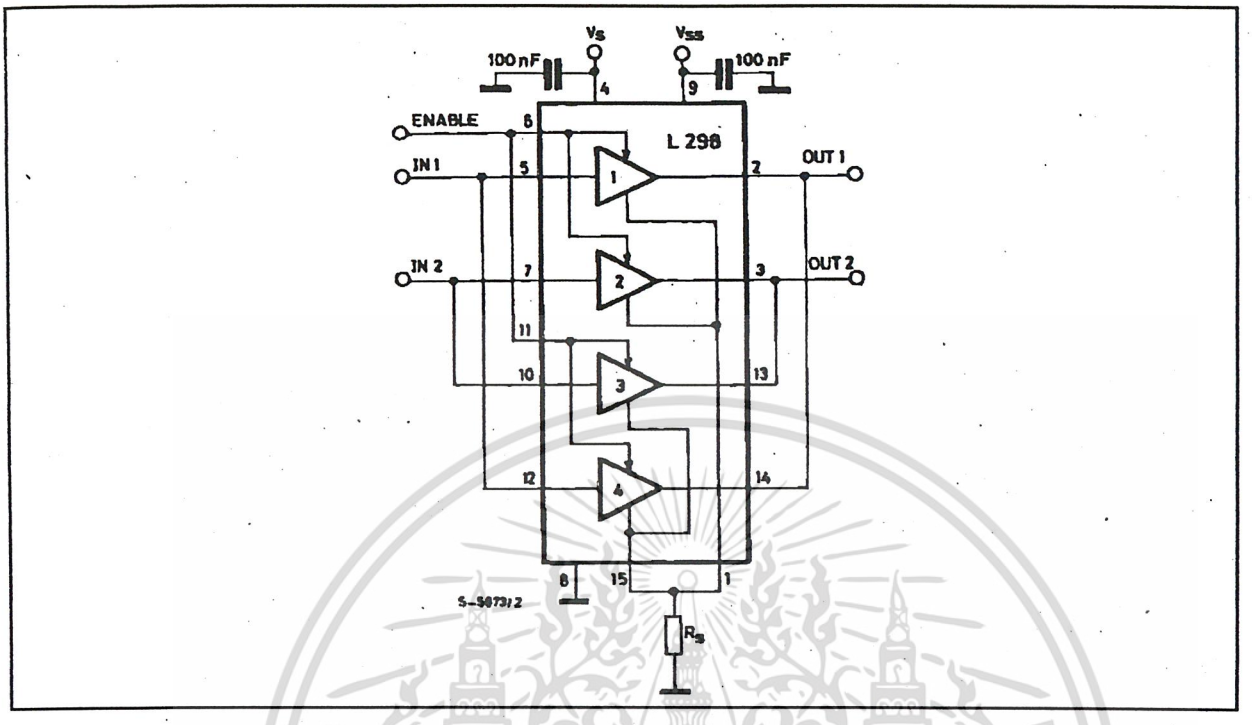
Figure 6 : Bidirectional DC Motor Control.



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.

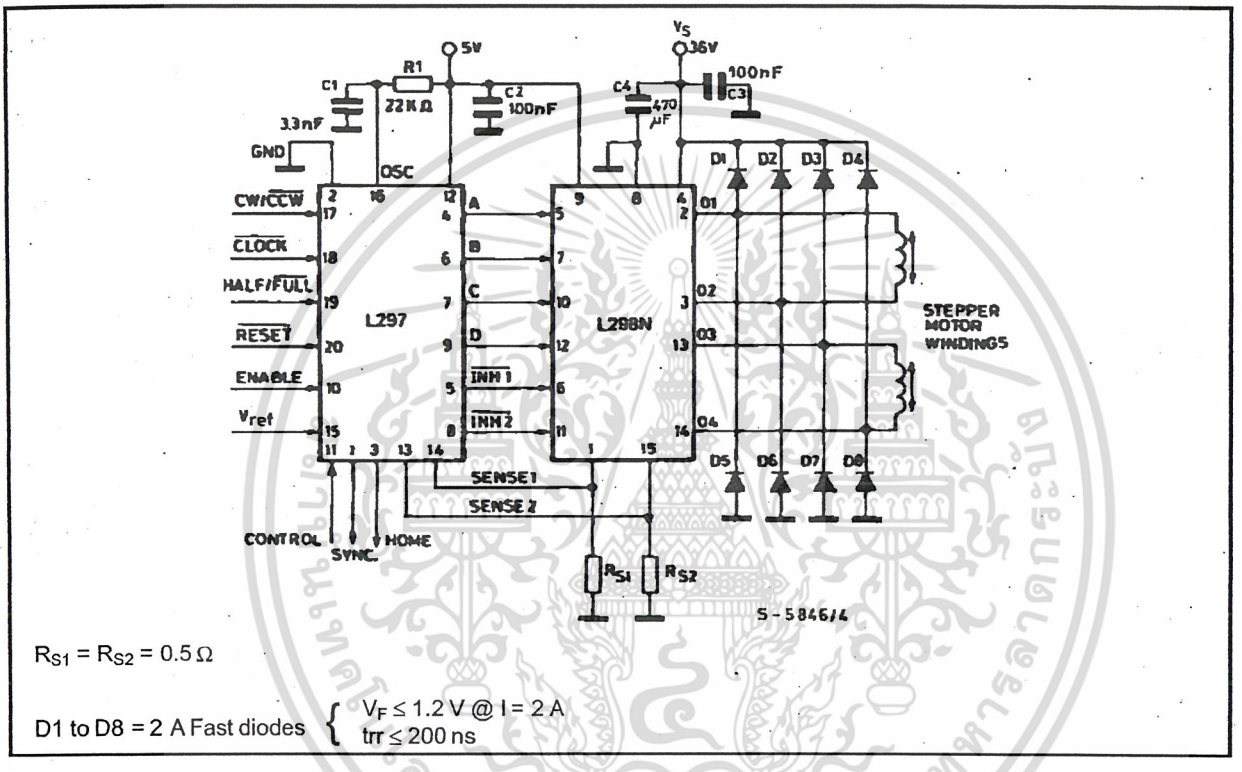
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

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).

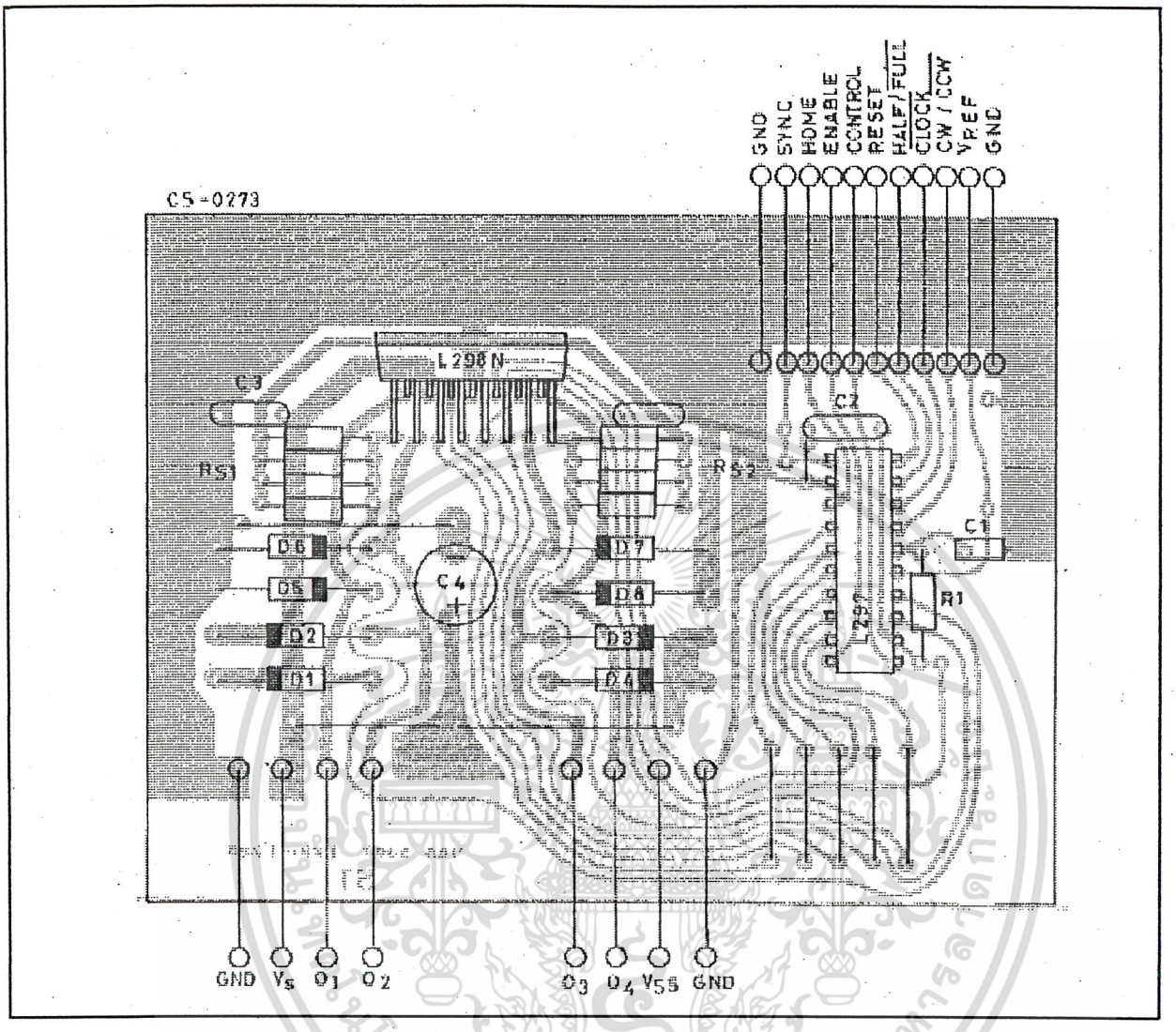
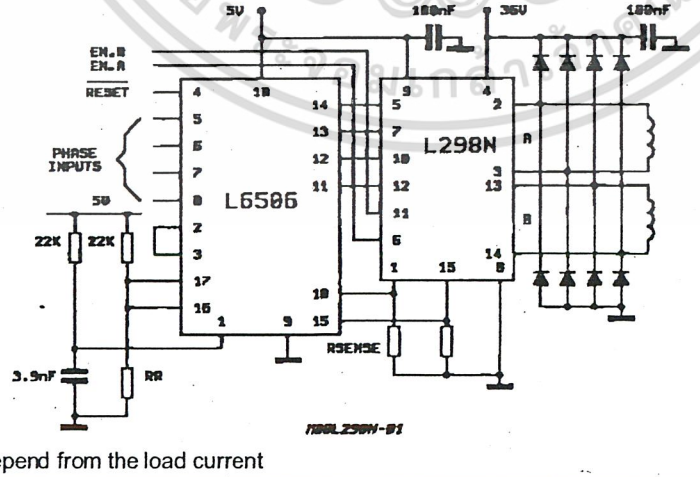


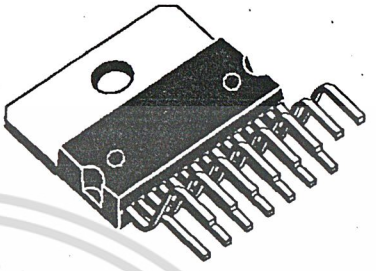
Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



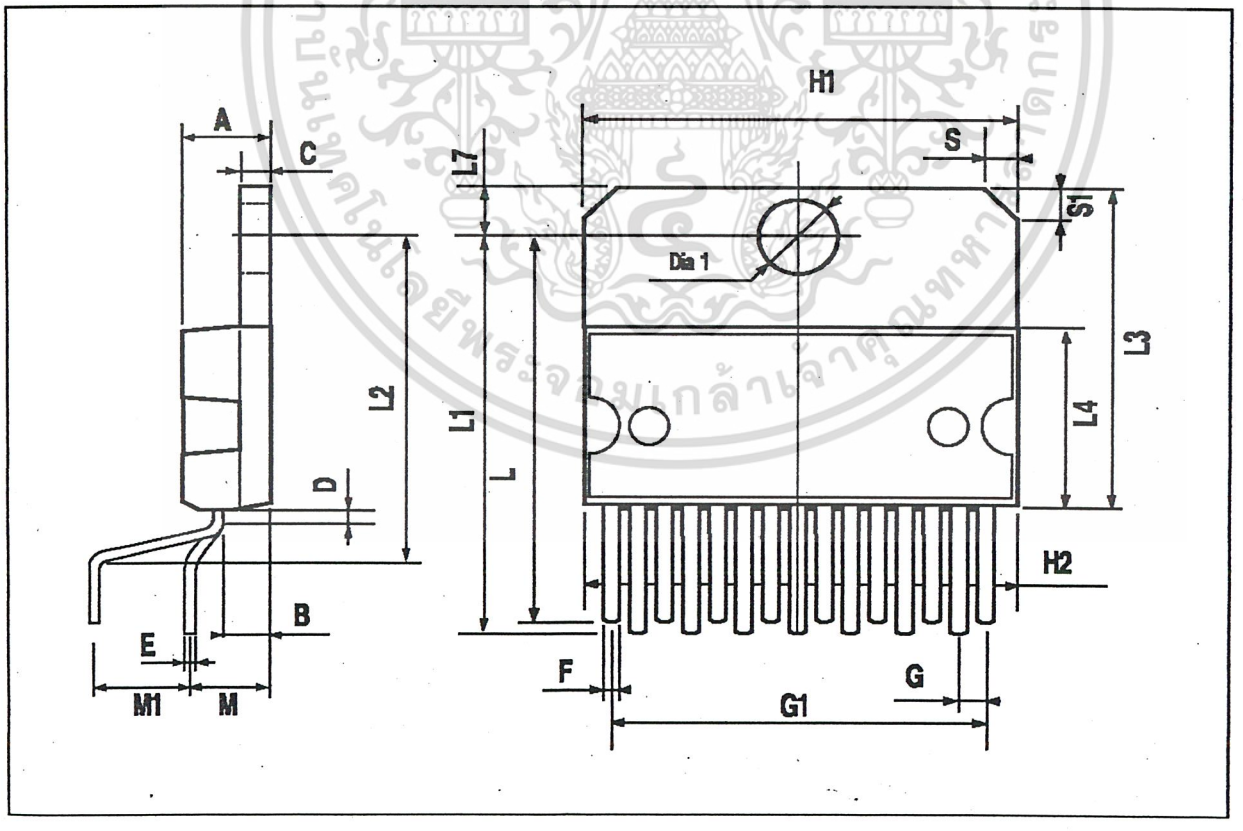
R_R and R_{sense} depend from the load current

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

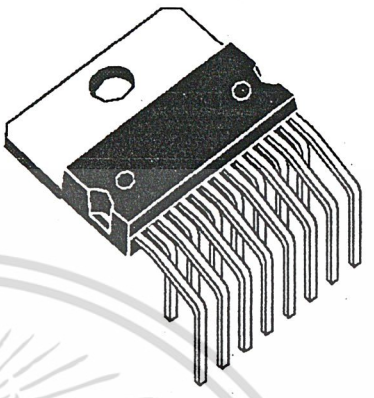


Multiwatt15 V

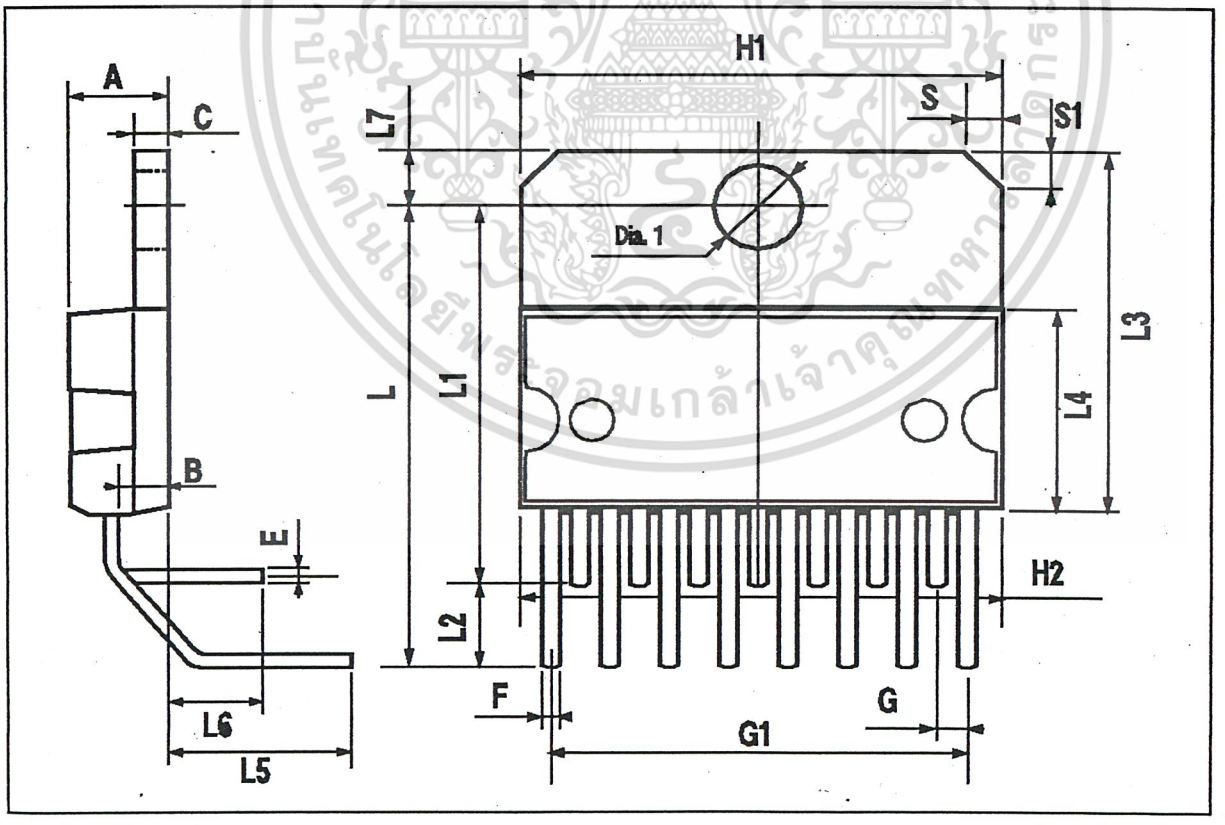


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L		20.57			0.810	
L1		18.03			0.710	
L2		2.54			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



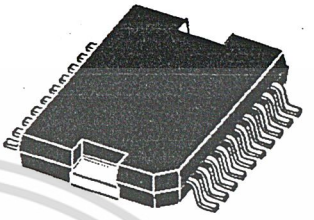
Multiwatt15 H



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	

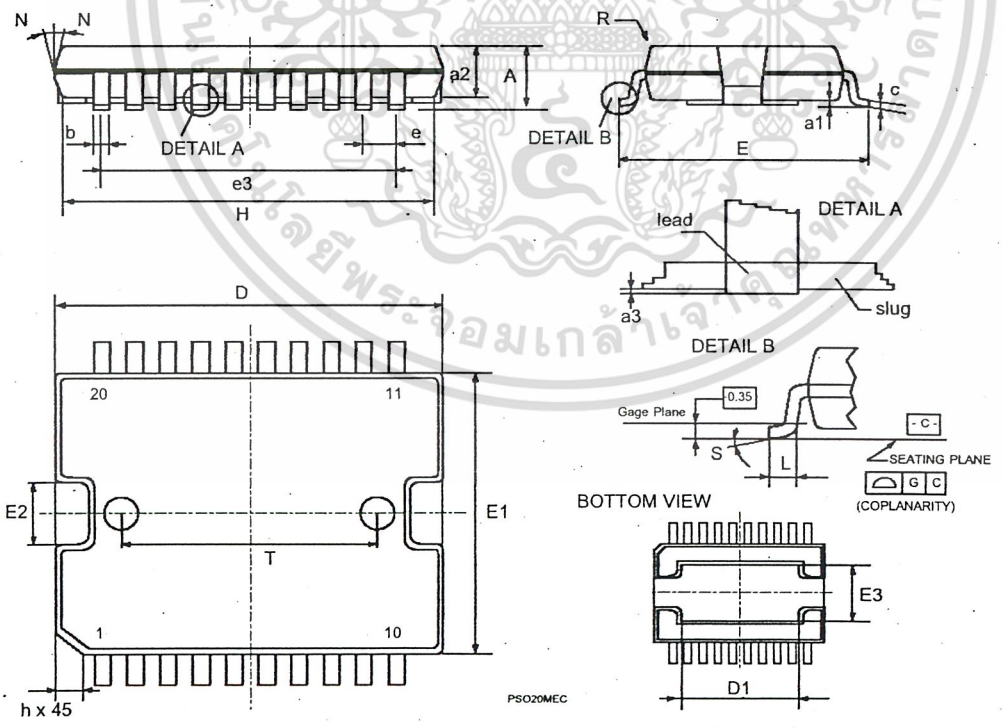
(1) "D and F" do not include mold flash or protrusions.
- Mold flash or protrusions shall not exceed 0.15 mm (0.006").
- Critical dimensions: "E", "G" and "a3"

OUTLINE AND MECHANICAL DATA



JEDEC MO-166

PowerSO20



กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงด้วยดีเนื่องมาจากได้รับความอนุเคราะห์จากบุคคลหลายๆ ท่าน โดยเฉพาะอย่างยิ่ง จากอาจารย์ที่ปรึกษา ผศ.ดร. ชนินทร์ บุญลักษณานุสรณ์ ที่คอยให้คำแนะนำและให้คำปรึกษาในการทำโครงงาน อีกทั้งยังเป็นแรงกระตุ้น และสร้างแรงผลักดันให้เกิดความคืบหน้าในการทำงาน รวมไปถึงอาจารย์ท่านอื่นๆ พี่ธุรการภาค พี่สโตร์ ที่ให้ความอนุเคราะห์ในการทำโครงงาน ขอขอบคุณเป็นอย่างยิ่งกับเพื่อนๆ ทุกๆ คน สำหรับการเป็นเพื่อนอย่างที่เป็น และสุดท้ายนี้ ขอขอบคุณเป็นอย่างสูงสำหรับผู้อ่านทุกท่าน

นายเทพ จารุรัตน์มงคล

นายประทีป วงศ์จิรชยา

ผู้จัดทำ

หนังสืออ้างอิง

1. กฤษดา ใจเย็น, "สนุกกับไมโครคอนโทรลเลอร์ฉบับ PIC", อินโนเวตีฟ เอ็กเพอริเมนต์, กรุงเทพฯ: ม.ป.ป.
2. วัชรินทร์ เคารพ, "เรียนรู้และเข้าใจไมโครคอนโทรลเลอร์ PIC ด้วยภาษาเบสิก PicBasic Pro", อีทีที, กรุงเทพฯ: พิมพ์ครั้งที่ 1, 2547
3. วัชรินทร์ เคารพ, "เรียนรู้และเข้าใจสถาปัตยกรรมไมโครคอนโทรลเลอร์ PIC16F877", อีทีที, กรุงเทพฯ: พิมพ์ครั้งที่ 1, 2547
4. วารี ปริยพงศ์, "เส้นทางสู่นักประดิษฐ์หุ่นยนต์", สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), กรุงเทพฯ: พิมพ์ครั้งที่ 1, 2547