รายงานโครงการหมายเลข EE 2003-69

ควบคุมมอเตอร์ไฟฟ้าด้วยแรงดันต่ำ

นาย สุนทร ทาพิลา เลขประจำตัว 443040246-4

นาย ปิยะวัติ กอนคอน เลขประจำตัว 443040177-7

รายงานนี้เป็นรายงาน งานโครงการของหนักศึกษาชั้นปีที่4 ซึ่งเสนอเป็นส่วนหนึ่งใน
หลักสูตรวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น

พ.ศ. 2546

Project Report No. **2003-69**

Low – Voltage Motor Drive

Mr. SOONTHORN TAPILA ID **443040246-4**

Mr. PIYAWAT KONCON ID **443040177-7**

This is report of fourth year project assignment submitted in partial fullfillment of the requirement for the degree of Bachelor of Engineering.

Department of Electrical Engineering

Faculty of Engineering, Khon Kaen University

2003

ใบประเมินผลงานโครงการ

ชื่อเรื่องภาษาไทย **มอเตอร์ไฟฟ้าเหนี่ยวนำสามเฟสด้วยแรงดันต่ำ**

ชื่อเรืองภาษาอังกฤษ Low – Voltage Induction Motor

ชื่อผู้เสนอโครงการ

นาย สุนทร ทาพิลา เลขประจำตัว 443040246-4

นาย ปิยะวัติ กอนคอน เลขประจำตัว 443040177-7

อาจารย์ที่ปรึกษา

_____

(อาจารย์ ดร. กฤษ เฉยไสย)

อาจารย์ผู้ร่วมประเมินผลโครงการ

_____

(อาจารย์ สถิพร พรนิมิตร)

_____

(อาจารย์ วิชัย เปรมชัยสวัสดิ์)

ประเมินผล ณ วันที่ 11 มกราคม พ.ศ. 2547

กิตติกรรมประกาศ

โครงการนี้ เป็นโครงการที่เกี่ยวกับการนำความรู้ทางด้านวิศวกรรมไฟฟ้าในแขนงมอเตอร์ไฟฟ้า อิเล็กทรอนิกส์กำลัง และไมโครคอนโทรลเลอร์ ในการออกแบบและสร้างมอเตอร์ไฟฟ้าเหนี่ยวนำสามเฟสที่สามารถใช้ได้กับแรงดันต่ำ ๆซึ่งจะเป็นประโยชน์ที่จะพัฒนาต่อไป ซึ่งโครงการนี้สำเร็จลุล่วงไปได้ดีทั้งนี้ได้รับความอนุเคราะห์จากท่านอาจารย์ ดร. กฤษ เฉยไสย อาจารย์ที่ปรึกษาโครงการที่คอยให้คำปรึกษาและให้คำแนะนำมาโดยตลอด รวมทั้งอาจารย์ สถิรพร พรนิมิตร อาจารย์ที่ปรึกษาร่วม

พร้อมกันนี้ใคร่ขอขอบคุณต่อเจ้าหน้าที่ควบคุมการเบิกจ่ายอุปกรณ์และเครื่องมือของภาควิชาวิศวกรรมไฟฟ้าทุกท่าน

นาย สุนทร ทาพิลา
นาย ปิยะวัติ กอนคอน

บทคัดย่อ

โครงการนี้มีจุดประสงค์ที่จะพัฒนาและออกแบบมอเตอร์ไฟฟ้ากระแสสลับสามเฟสที่มีอยู่ให้ใช้ประโยชน์อย่างอื่น อีกทั้งยังใช้งานได้กับไฟฟ้ากระแสตรงที่ไฟฟ้ากระแสสลับไม่สามารถเข้าถึงได้

จากการทดลองปรากฏว่า ใช้ไฟฟ้ากระแสตรงแรงดัน 48 v ให้กับวงจรและทำวงจรให้เหมาะสมกับการใช้งานโดยการควบคุมผ่านไมโครคอนโทรลเลอร์ ATmega 32 และจะเป็นประโยชน์ในการำไปใช้งานหรือพัฒนาต่อไปได้ด้วย

ABSTRACT

The aim of this projected Is to be development and designed Three Phase AC Induction Motor can been other used and can been Used Direct Current (DC) at Alternating Current (AC) not attended. Results form en experiment show use Direct Current 48 VoltageGive for Circuit Compete and Circuiting for using Control pass Microcontroller's ATmega32 and will be using for development.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

เนื่องจากปัจจุบันนี้การใช้เทคโนโลยีต่าง ๆ ได้มีการพัฒนาไปเรื่อย ๆ ฉะนั้นแล้วการที่เราจะดัดแปลงเครื่องใช้ไฟฟ้าที่มีอยู่ ให้สามารถใช้ได้กับอุปกรณ์ที่มีการพัฒนาความสามารถมากกว่าเดิมและมีเทคโนโลยีที่ทันสมัยนั้น จะเป็นการใช้ทรัพยากรให้เกิดประโยชน์มากขึ้นรวมทั้งจะสามารถนำไปใช้ในงานที่เป็นเฉพาะด้านมากขึ้น เช่น

1. แหล่งจ่ายไฟตรงหาได้ง่ายกว่าแหล่งจ่ายไฟสลับ ซึ่งสามารถขนไปไหนมาไหนได้ เช่น แบตเตอรี่ ซึ่งง่ายสำหรับการใช้ควบคุมสิ่งที่เคลื่อนที่ได้ อาทิเช่น รถไฟฟ้า
2. ใช้ควบคุมความเร็วของมอเตอร์กระแสสลับโดยการเปลี่ยนความถี่ เมื่อความถี่ของกระแสสลับเปลี่ยนแปลง ความเร็วของมอเตอร์จะเปลี่ยนแปลงตามสูตร

   $N = (120f)/P$; N= ความเร็วรอบหน่วยเป็นรอบหน่วยเป็นรอบต่อวินาที

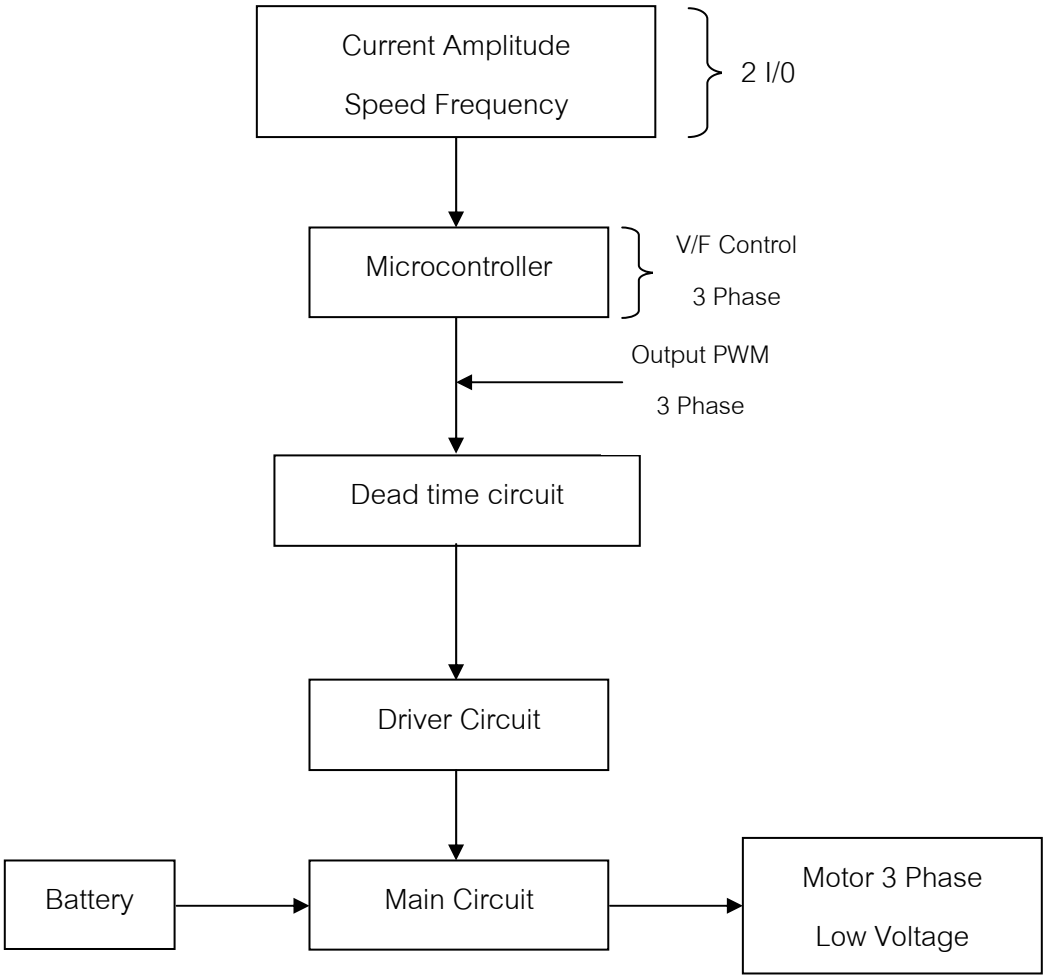
   ; f = ความถี่ของแหล่งจ่ายไฟเป็นไซเคิลต่อวินาที

   ; P = จำนวน pole ของมอเตอร์
3. ในการควบคุมเราควบคุมผ่าน Microcontroller ซึ่งเป็นอุปกรณ์ที่เล็กและมี function การใช้งานมากมาย สามารถปรับอัตราส่วนของแรงดันต่อความถี่ ที่จ่ายให้กับมอเตอร์ได้ที่ Program ที่เขียนให้กับ Microcontroller เลย

ดังนั้น โครงการนี้ได้เล็งเห็นประโยชน์ดังที่กล่าวไว้แล้ว จึงคิดที่พัฒนาอุปกรณ์ไฟฟ้าคือมอเตอร์ไฟฟ้าสามเฟสที่มีใช้ทั่วไปให้สามารถใช้ได้กับแบตเตอรี่ที่มีขายตามท้องตลาด เพื่อที่จะสามารถอุปกรณ์ชุดนี้ไปใช้ในที่ที่ระบบไฟฟ้าไม่สามารถเข้าถึงได้หรือจะพัฒนาใช้เป็นตัวขับเคลื่อน
ในรถยนต์ต่อไป

การทำงานของโครงการนี้จะเริ่มจากการใช้แบตเตอรี่ขนาดของแรงดัน 48 โวลท์เป็นแหล่งจ่ายพลังงาน โดยมีชุดขับที่จะเป็นตัวแปลงกระแสไฟฟ้าจากแบตเตอรี่ ให้สามารถใช้ขับเคลื่อนมอเตอร์ไฟฟ้าที่มีการดัดแปลงขนาดของขดลวดให้ใหญ่ขึ้นและทนกระแสได้มากขึ้น ในส่วนของการควบคุมนั้นจะใช้ไมโครคอนโทรลเลอร์ ATmega32 เป็นตัวสร้าง Pulse Width Modulation (PWM) ให้วงจร Dead Time Circuit จะทำให้มีสัญญาณ ON – OFF สลับกัน แล้วส่งให้วงจร MOSFET Driver Circuit ซึ่งเป็นวงจรที่จะเป็นตัวขับมอเตอร์ให้ทำงาน ดังแสดงในรูป 1.1

หลักการควบคุมการทำงานของ Low – voltage Induction Motor



รูปที่ 1.1 การควบคุมการทำงานของ Low – voltage Induction Motor

หลักการทำงาน คือ เมื่อเราป้อนไฟฟ้าจากแบตเตอรี่ ผ่าน Main Circuit **ซึ่งทำหน้าที่เป็นสวิตช์ โดยการสวิตช์จะควบคุมจาก** Microcontroller **ผ่าน** Microcontroller **ซึ่งเราจะโปรแกรมเอาไว้ ให้สามารถปรับค่าความถี่และ** Amplitude (V/f Control) **สัญญาณที่ออกจาก** Microcontroller **จะเป็น** function PWM 3 channel **เข้าวงจร** Dead time **เพื่อให้สัญญาณ** shitf phase **กัน ก่อนป้อนเข้าสู่** Driver circuit **และนำไปควบคุม** Main circuit **ต่อไป สัญญาณที่ออกจาก** Main circuit **จะเป็นรูป** Sine wave **สามารถควบคุมมอเตอร์ได้ ดังรูปที่ 1.1**

1.2 วัตถุประสงค์

1. ศึกษาการใช้งานโปรแกรม AVR Studio 4
2. ศึกษาหลักการทำงานและการใช้งานของฟังก์ชั่น PWMภายใน Microcontroller AVR ATMEGA 32
3. ศึกษาหลักการทำงานและการใช้งานของฟังก์ชั่น ADC ภายใน Microcontroller AVR ATMEGA 32
4. ศึกษาการเขียนโปรแกรมด้วยภาษา Assembly ให้กับ Microcontroller AVR ATMEGA 32
5. ศึกษาข้อมูลและดัดแปลงมอเตอร์ไฟฟ้าสามเฟสให้ใช้กับแรงดันต่ำ ๆ ได้
6. ศึกษาและทดลองสร้างวงจรขับมอเตอร์แรงดันต่ำได้

1.3 ขอบข่ายของการทำงาน

1. เขียนโปรแกรมสร้าง PWM ภายใน AVR ATMEGA 32
2. เขียนโปรแกรมเพื่อใช้งานฟังก์ชั่น ADC โดยใช้ AVR ATMEGA 32
3. ต่อวงจรทดลองเพื่อทดสอบการทำงานของโปรแกรมที่ใช้ฟังก์ชั่น ADC และ PWM
4. ดัดแปลง Parameter ของมอเตอร์ไฟฟ้าสามเฟสได้
5. สร้างวงจรที่จะใช้ขับมอเตอร์ได้

1.4 ผลที่คาดว่าจะได้รับ

1. สามารถเขียนโปรแกรมสร้าง PWM ได้
2. สามารถเขียนโปรแกรมเพื่อศึกษาการใช้งานฟังก์ชั่น ADC ได้
3. สามารถเขียนโปรแกรมการทำงานให้กับ AVR ATMEGA32 เพื่อใช้ในการควบคุมอินเวอร์เตอร์ได้
4. ได้เรียนรู้วิธีการเขียนโปรแกรมและใช้โปรแกรม AVR studio 4
5. เรียนรู้ไมโครคอนโทรลเลอร์ AVR ใน Mode ที่ใช้งาน
6. เรียนรู้และสร้างวงจร Dead Time , MOSFET Driver
7. เรียนรู้การพันขดลวดของมอเตอร์เหนี่ยวนำไฟฟ้าสามเฟส

บทที่ 2
ทฤษฎีพื้นฐาน

AVR เป็นไมโครคอนโทรลเลอร์(MCU)ที่ได้รวบรวมอุปกรณ์สนับสนุนการทำงานของ CPU ไว้มากมาย อาทิเช่น Analog to Digital , SPI , UART , Timer , Counter , PWM ซึ่งอุปกรณ์สนับสนุนการทำงานเหล่านี้ทำให้ MCUสามารถทำงานได้กว้างและใช้อุปกรณ์ต่อร่วมจากภายนอกน้อยมาก และสามารถประมวลคำสั่งได้ภายใน 1 clock ในบทนี้จะนำเสนอข้อมูลบางส่วนที่เป็นการทำงานภายในของ AVR - MCU แนะนำคุณสมบัติและขาต่อใช้งานของไมโครคอนโทรลเลอร์ สถาปัตยกรรมภายในและรีจีสเตอร์ใช้งานทั่วไป ตำแหน่ง I/O รีจีสเตอร์สถานะและการใช้งาน EEPROM การรีเซ็ตและการอินเตอร์รัพท์ การสื่อสารอนุกรม การเปรียบเทียบสัญญาณอนาลอกและการแปลงสัญญาณอนาลอกเป็นดิจิตอล การทำงานของพอร์ตอินพุต/เอาท์พุตการทำงานของTimer / Counter & Watch dog และการใช้กลุ่มคำสั่งต่าง ๆ

2.1 คุณสมบัติและขาต่อใช้งานของไมโครคอนโทรลเลอร์

คุณสมบัติ

1. สถาปัตยกรรมภายในถูกออกแบบให้ใช้สถาปัตยกรรมแบบ RISE (Reduce Instruction Set Computer) RISE คือ ทำให้การประมวลผลมีความเร็ว 1 คำสั่ง / 1 Clock หรือ CPU สามารถประมวลคำสั่งได้ 1 MIPS / MHz
2. มีคำสั่งในการควบคุมการทำงานของไมโครคอนโทรลเลอร์จำนวน 118 คำสั่ง
3. หน่วยความจ 3. หน่วยความจบบันทึก PROGRAM MEMORY ขนาด 32 Kbyte
4. หน่วยความจำแบบ EEPROMสำหรับบันทึก DATA MEMORY ขนาด 1024 Byte
5. หน่วยความจำแบบ RAM ขนาด 2K Byte
6. ระบบการเปลี่ยนสัญญาณ ANALOG TO DIGITAL ขนาด 10 บิท จำนวน 8 CHANNEL
7. กลุ่มรีจีสเตอร์ใช้งานทั่วไปขนาด 8 บิท จำนวน 32 ตัว
8. พอร์ตอินพุตและเอาท์พุตขนาด 8 บิท จำนวน 4พอร์ต
9. ระบบการสื่อสารข้อมูลดิจิตอลแบบอะซิงโครนัส(UART) 1 CHANEL
10. ระบบการสื่อสารข้อมูลดิจิตอลแบบซิงโครนัส(SPI) 1 CHANEL
11. ความถี่สัญญาณนาฬิกา 0 - 16 MHz (ATMEGA 32)
12. ระบบการรีเซ็ตแบบอัตโนมัติเมื่อเริ่มจ่ายกระแสไฟฟ้าเข้าไมโครคอนโทรลเลอร์(Power on reset)
13. ระบบการกำเนิดความถี่สัญญาณแบบ PWM จำนวน 4 CHANNEL (ATMEGA 32)
14. ระบบการตรวจจับระดับสัญญาณอนาลอก(Analog Comparator)
15. 6 SLEEP MOD:IDEL ,POWER SAVE , POWER DOWN ,ADC Noise , Reduction , Standby, and Extended stanby

16. ระบบการป้องกันการ COPY ข้อมูลภายในหน่วยความจำ (LOCK FOR SOLFWARE SECURITY)

17. ระบบตรวจจับการทำงานผิดพลาดของ CPU (WATCHDOG TIMER WITH ON-CHIP OSCILATOR)

18. ระบบการอินเตอร์รัพท์จากภายนอก (EXTERNAL INTERRUPT)

19. TIMER/COUNTER ขนาด 16 บิท 1 CHANNEL

20. TIMER/COUNTER ขนาด 8 บิท 2 CHANNEL

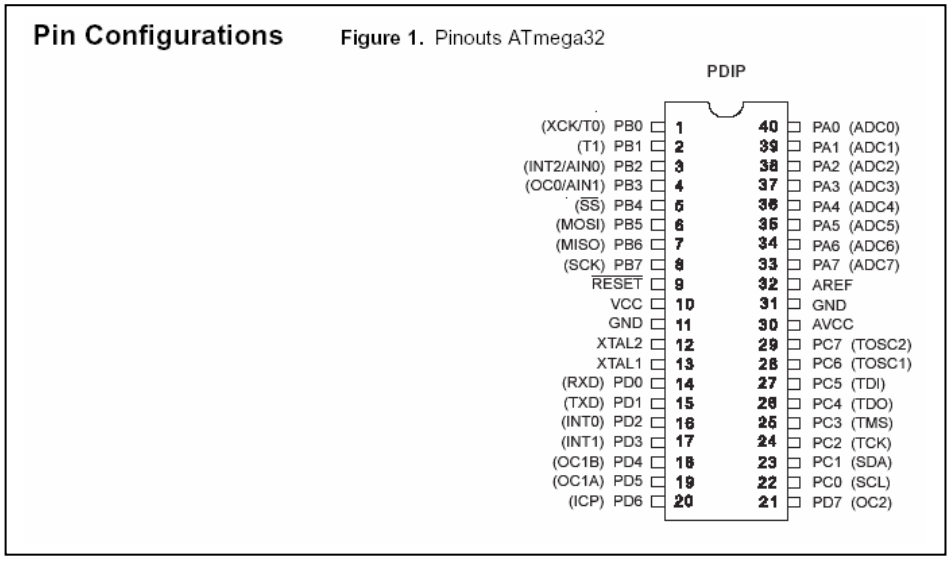
21. Vcc: 4.5 - 5.5 for ATMEGA 32

รายละเอียด

AT MEGA 32 เป็นไมโครคอนโทรลเลอร์ขนาด 10 บิทที่มีสถาปัตยกรรมแบบ RISE (reduce instruction set computer) ซึ่งทำให้การประมวลผลมีความเร็ว 1 คำสั่ง/ 1 clock หรือ CPU สามารถประมวลคำสั่งได้ 1 MIPS / MHz

2.2 โครงสร้างภายนอกและตำแหน่งขา

ภายในประกอบด้วยรีจีสเตอร์ใช้งานทั่วไปขนาด 8 บิท จำนวน 32 ตัวซึ่งแต่ละตัวจะต่อเข้ากับALU โดยตรง ทำให้การประมวลผล ต่อ 1 คำสั่งมีความเร็วกว่า CPU ที่มีสถาปัตยกรรมแบบCISC



รูปที่ 2.1 โครงสร้างภายนอกและตำแหน่งขา

2.3 รายละเอียดของขาสัญญาณและการใช้งาน

VCC

คือ ขาจ่ายไฟให้กับ CPU และ GND คือ กราวด์

Port A (PA7..PA0)

เป็นพอร์ต 2 ทิศทางขนาด 8 บิท โดยสามารถกำหนดให้แต่ละขาของพอร์ตสามารถ PULL UP ภายในแยกจากกันซึ่งสามารถรับกระแส SINK 20mA โดยพอร์ต Aยังใช้เป็นขาอินพุตเพื่อรับสัญญาณอนาลอกในส่วนของการแปลงสัญญาณ ANALOG TO DIGITAL

Port B (PB7..PB0)

เป็นพอร์ต 2 ทิศทางขนาด 8 บิท โดยสามารถกำหนดให้แต่ละขาของพอร์ตสามารถ PULL UP ภายในอิสระแยกจากกันซึ่งแต่ลขาสามารถรับกระแสSINK20mA และยังถูกนำไปใช้งานอื่น ๆอีก

Port C (PC7..PC0)

เป็นพอร์ต 2 ทิศทางขนาด 8 บิท โดยสามารถกำหนดให้แต่ละขาของพอร์ตสามารถ PULL UP ภายในอิสระแยกจากกันซึ่งแต่ละขาสามารถรับกระแสSINK20mAและยังถูกนำไปใช้งานอื่น ๆอีก

PorT D (PD7..PD0)

เป็นพอร์ต 2 ทิศทางขนาด 8 บิท โดยสามารถกำหนดให้แต่ละขาของพอร์ตสามารถ PULL UP ภายในอิสระแยกจากกันซึ่งแต่ละขาสามารถรับกระแสSINK20mAและยังถูกนำไปใช้งานอื่น ๆอีก

Reset คือ ขารีเซ็ต

XTAL 1 เป็นขาอินพุตของ OSE

XTAL 2 เป็นขาเอาท์พุตของ OSE

AVcc ใช้จ่ายไฟให้กับวงจร Analog to Digital

AREF เป็นขาแรงดันอ้างอิงที่ใช้งานในส่วนของวงจร Analog to Digital

AGND เป็นขากราวด์ของวงจร Analog to Digital

2.4 ฟังก์ชั่น ADC

การแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลนั้นมีความจำเป็นมากเพราะว่าในตัวไมโครคอนโทรลเลอร์ไม่สามารถประมวลผลแบบอนาล็อกได้มันจะประมวลผลแบบดิจิตอลเท่านั้นดังนั้นเราจึงจำเป็นต้องมีการแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอล

ปกติใน CPU ของ AVR - ATMEGA32 นั้นจะมีฟังชั่น ADC อยู่ภายในตัวไอซี ดังนั้นไม่จำเป็นต้องใช้ไอซี ADC ต่อภายนอก สำหรับฟังก์ชั่น ADC นี้สามารถรับสัญญาณอนาล็อกได้สูงสุด 8 Channel โดยรับสัญญาณเข้ามาทาง พอร์ต A เราสามารถเลือกใช้ฟังก์ชั่นนี้ทำการแปงสัญญาณอนาล็อกที่ละ Channel อย่างต่อเนื่อง หรือจะให้ทำการแปลงสัญญาณเฉพาะ channelที่เราต้องการ ก็ได้เช่นกัน โดยสัญญาณดิจิตอลที่แปลงได้จะมีความละเอียด 10 บิต โดยการรับสัญญาณแต่ละขาของพอร์ท Aโดยจะมีวงจร SAMPLE AND HOLD

เพื่อช่วยให้สัญญานอนาลอกที่รับเข้ามาเพื่อแปลงเป็นสัญาณานดิจิตอลที่มีระดับสัญญาณคงที่โดยปกติการใช้งานฟั่งก์ชั่นนี้เราจำเป็นต้องจัดแรงดัน AVCC AREF และ AGND ให้กับฟังก์ชั่นด้วย

คุณสมบัติ

1. 10bit resolution
2. 0.5LSB integral non-linearity
3. ±2 LSB Absolute Accuracy
4. 65-260 μσ Conversion Time
5. Up to 15κSPS at Maximum Resolution
6. 8 Multiplexed Single Ended Input Channels
7. 7 Differential Input Channels
8. 2 Differential Input Channels With Gain of 10xand 200x(1)
9. Optional left adjustment for ADC result readout
10. 0-Vcc ADC Input Voltage Range
11. selectable 2.56V ADC Reference Voltage
12. Free Running or Single conversion Complete
13. Sleep mode noise Canceler

การทำงาน

ในส่วนของการแปลงสัญญาณ อนาลอก เป็นดิจิตอล สามารถทำได้ 2 mode คือ

1. Single Conversion Mode
2. Free Running Mode

การทำงาน Single Conversion Mode ผู้ใช้ต้องเป็นผู้กำหนดการใช้งานขึ้นเอง แต่ในส่วนของ Free Running Mode วงจร Analog to digital จะเป็นตัวจัดการอ่านข้อมูลและเก็บใน ADC Data Register ซึ่งบิต ADFR ใน Register ADCSR จะเป็น บิตที่ใช้เลือก Mode การใช้งานของวงจร Analog to digital สำหรับการกำหนดให้วงจร Analog to digital ทำงานนั้น สามารถทำได้โดยการเซ็ตบิต ADEN ในรีจีสเตอร์ ADCHRA ให้เป็น 1 โดยบิตนี้จะเป็น 1 ไปตลอดจนกระทั่ง Conversion ของสัญญาณจะเรียบร้อยแล้วจึงทำให้บิตนี้เป็น 0 โดยอัตโนมัติ แต่ถ้าเป็นการเปลี่ยน Channel ของการแปลงสัญญาณขณะที่ Channel เดิมยัง Conversion อยู่ วงจร Analog to digital จะ Conversion สัญญาณ Channel เดิมให้เสร็จก่อนแล้วจึง Conversion สัญญาณ Channel ถัดไป โดยข้อมูลที่ได้จากการแปลงสัญญาณอนาล็อกเป็นดิจิตอลจะเก็บไว้ในรีจีสเตอร์ ADCH และ ADCL

2.5 ฟังก์ชั่น PWM

การมอดูเลตความกว้างของพัลส์ (PWM) เป็นเทคนิคสำคัญที่ใช้ในการปรับปรุงสมรรถนะของอินเวอร์เตอร์ ดังนั้นการศึกษาเกี่ยวกับ PWM จึงมีความจำเป็นอย่างยิ่งสำหรับอินเวอร์เตอร์ เพื่อที่อินเวอร์เตอร์จะได้มีสมรรถนะและประสิทธิภาพในการทำงานที่ดีขึ้น เนื่องจากว่า PWM เป็นฟังก์ชั่นการทำงานหนึ่งในโหมด PWM ของ Timer/Counter ที่อยู่ภายใน AVR – ATMEGA32 ดังนั้นในหัวข้อต่อไปจะแนะนำเกี่ยวกับการทำงานของ Timer/Counter ของ AVR – ATMEGA32

Timer /Counter

ภายใน AVR - ATMEGA32 จัดให้มี Timer/Counter 3 ชุด โดยจัดเป็น Timer/Counter ขนาด 8 บิท 2 ชุด และ Timer/Counter ขนาด 16 บิท 1 ชุด ดังนี้คือ Timer/Counter2 และ Timer/Counter0 และ Timer/Counter1 ซึ่ง Timer/Counter2 สามารถรับสัญญาณ Clock จากภายนอก ซึ่งเป็น Option ที่จะนำ Timer/Counter2 มาทำเป็น RTC โดยใช้ XTAL ที่มีค่าความถี่เท่ากับ 32.768KHz มาเป็นฐานเวลา และ Timer/Counter0 และ Timer/Counter1 ใช้วงจร Prescaling ขนาด 10 บิทร่วมกัน ส่วน Timer/Counter2 ใช้วงจร Prescaling แยกออกต่างหาก

แนะนำการใช้งาน Timer/Counter แต่ละประเภท

Timer/Counter0

โครงสร้างของ Timer/Counter0 ขนาด 8 บิท แสดงในรูปที่ 50 ซึ่งสามารถเลือกสัญญาณ Clock ได้จาก CK (Clock ของระบบ) หรือสัญญาณ Clock ของระบบที่ถูกหาร (Prescaling) หรือ สัญญาณจากภายนอก โดยการใช้งานจะอธิบายในรีจีสเตอร์ TCCRO และ TIFR ส่วนสัญญาณควบคุมสามารถทราบรายละเอียดได้จาก รีจีสเตอร์ TCCRO ซึ่งการควบคุมการอินเตอร์รับจะควบคุมไดจาก รีจีสเตอร์ TIMSK เมื่อ Timer/Counter0 ได้รับสัญญาณจากภายนอก ซึ่งสัญญาณดังกล่าวจะซิงโครไน (Synchronized) กับสัญญาณนาฬิกาภายในCPU

โดย TIMER/COUNTER 0 จะเป็นวงจรนับขึ้นที่สามารถเขียนและอ่านข้อมูลได้ตลอดเวลา โดยเมื่อทำการเขียนข้อมูลลงใน TIMER/COUNTER 0 ในขณะที่มีสัญญาณ Clock จะทำให้ TIMER/COUNTER 0 นับค่าต่อเนื่องจากค่าที่ถูกเขียนลงไป

2.Timer/Counter 1

จะมีขนาด 16 บิท โดยสามารถเลือกสัญญาณนาฬิกาได้จาก CK หรือสัญญาณที่ได้รับการหารจาก CK (Prescelling) ซึ่งการหยุด Timer/Counter 1 จะอธิบายไว้ในรีจีสเตอร์ TCCR1A (Timer/Counter 1 Control Register) และ TCCR1B โดยแฟร์กที่แสดงสถานะต่าง ๆ(Overflow,Compare math,Capture even) ส่วนสัญญาณควบคุมจะอธิบายไว้ในรีจีสเตอร์ TCCR1A และ TCCR1B การควบคุมสัญญาณอินเตอร์รัพต์จะควบคุมได้จากรีจีสเตอร์ TIMSK (TIMER/COUNTER INTERRUPT MASK REGISTER)

เมื่อ TIMER1/COUNTER1 จะประกอบด้วยส่วนของการเปรียบเทียบเอาท์พุต (Output Compear Function) 2 ฟังก์ชั่น โดยจะใช้ รีจีสเตอร์ OCR1A (Output Compare Register 1 A) และ OCR1B (Output

Compare Register 1B) เป็นส่วนของการเก็บค่าข้อมูลของการเปรียบเทียบ TIMER1/COUNTER1 จะสามารถเลือกใช้ฟังชั่น PWM ได้ทั้ง 8,9 และ 10 บิท

The Timer/Counter Control Register

Bits 7, 6-COM1A1, COM1A0: Compare Output Mode 1 A, bit 1 and 0 บิท COM1A1 และ COM1A0 เป็นบิทที่ใช้ในการกำหนดลักษณะของสัญญาณที่เกิดขึ้นที่ขา OC1A เมื่อ Timer/Counter1 เกิด Compare Match ซึ่งเมื่อใช้ฟังชั่น Output Compare Match ของ Timer/Counter1 จะต้องควบคุมให้ขา OC1A มีสถานะเป็นเอาท์พุต โดยการเลือกลักษณะของสัญญาณแสดงในตาราง

Bit 5, 4-COM1B1, COM1B0: Compare Output Mode 1 B, bit 1 and 0 บิท Com1A1 และ COM1A0 เป็นบิทที่ใช้ในการกำหนดลักษณะของสัญญาณที่เกดขึ้นที่ขา OC1A เมื่อTimer/Counter1 เกิด Compare Match ซึ่งเมื่อใช้ฟังชั่น Output Compare Match ของ Timer/Counter1 จะต้องควบคุมให้ขา OC1A มีสถานะเป็นเอาท์พุต

Bit 3…2-Res: Reserved bits ในส่วนของ AT mega32 จะสงวนบิทในกลุ่มนี้ไว้

Bit 1…0 - PWM11, PWM10: Pulse Width Modulator Select Bit
เป็นบิทที่ใช้ในการกำหนดการทำงานของ PWM

The Timer/Counter1 Control Register B-TCCRI1B

Bit 7-ICN1: Input Capture 1 Noise Canceler (4 CKs) บิทนี้เป็นบิทที่กำหนดให้ Input Capture 1 Noise Canceler ทำงานหรือไม่ทำงาน โดยเมื่อบิทนี้เป็น 1 จะเป็นการกำหนดให้ Input Capture 1 Noise Canceler ทำงาน แต่เมื่อบิทนี้เป็น 0 จะเป็นการกำหนดไม่ให้ Input Capture 1 Noise Canceler ทำงาน

ชุด Noise Canceler จะถูกกำหนดให้ทำงานโดยการ Sampling สัญญาณที่เข้ามาที่ชุด Input Capture 1 โดยสัญญาณ Sampling แรกจะเริ่มที่ขอบแรกของสัญญาณขาขึ้นหรือขาลงขึ้นอยู่กับการกำหนดในบิท ICES1 โดยชุด Noise Canceler จะ Sampling ด้วยความถี่เท่ากับความถี่ของ XTAL ซึ่งจะ Sampling ทั้งหมด 4 ครั้ง โดยลอจิกที่ได้จากการ Sampling จะต้องมีลอจิกเดียวกันกับลอจิกที่กำหนดในบิท ICES1

Bit 6-ICES1: Input Capture 1 Edge Select เป็นบิทที่ใช้กำหนดให้ชุด Input Capture 1 จะต้อง Detect ถ้าบิท ICES1 เซ็ต เป็น 1 จะเป็นการกำหนดให้ชุด Input Capture1 ทำหน้าที่ Detect สัญญาณที่ขอบขาขึ้น แต่ถ้าบิท ICES 1 ถูกเคลียเป็น 0 จะเป็นการกำหนดให้ชุด Input Capture 1 ทำหน้าที่ Detect สัญญาณที่ขอบขาลง

Bit 5, 4-RES: Reserved bits บิทนี้ถูกสงวนไว้

Bit 3: CTC1: Clear Timer1/Counter1 on Compare Match บิทนี้เป็นที่ใช้ในการกำหนดว่าเมื่อเกิด Output Compare แล้วจะให้เกิดการนับต่อไปหรือจะให้มีการรีเซ็ตค่าให้เป็น 00000 แล้วจึงทำการนับต่อไป โดยถ้าเป็นบิทนี้เป็น 1 จะเป็นการกำหนดให้มีการรีเซ็ตค่าให้เป็น 0 ขเมื่อเกิดการ Output Compare แต่ถ้าบิทนี้เคลียร์เป็น 0 จะเป็นการกำหนดให้มีการนับค่าต่อเมื่อเกิด Output Compare

Bit 2, 1, 0-CS12, CS11, CS10: Clock Select1, bit 2, 1 and 0 เป็นบิทที่ใช้ในการเลือกสัญญาณ Clock

The Timer/Counter In Capture Register – ICR1H AND ICR1L

เป็นรีจีสเตอร์ขนาด 16 ที่ใช้เก็บค่า Timer/Couner1 ที่อยู่ในรีจีสเตอร์ TCNT1 เมื่อ Input Capture สามารถ Detect ได้ เมื่อ Input Capture สามารถ Detect สัญญาณได้ตามที่กำหนดในบิท ICES1 จะทำให้ CPU โหลดค่าในรีจีสเตอร์ TCNT1 ลงในรีจีสเตอร์ และในเวลาเดียวกับบิท ICF1จะเซ็ตเป็น 1 โดยการอ่านค่าจากรีจิสเตอร์ ICR1 ของ CPU จะใช้รีจีสเตอร์ TEMP เป็นรีจีสเตอร์พักข้อมูล ซึ่งการใช้รีจีสเตอร์ TEMP ช่วยในการอ่านข้อมูลเพื่อให้ค่าที่อยู่ในรีจีสเตอร์ ICR1H และ ICR1L เสมือนถูกอ่านออกมาพร้อมกัน การอ่านค่าจาก รีจีสเตอร์ ICR1 จะต้องอ่านค่าจากรีจีสเตอร์ ICR1L ก่อน โดยเมื่อ CPU อ่านค่าจาก ICR1L จะทำให้ค่าในรีจีสเตอร์ ICR1H ถูกโหลดลงในรีจีสเตอร์ TEMP เมื่อ CPU อ่านค่าจาก ICR1H จะทำให้ค่าในรีจีสเตอร์ TEMP ถูกส่งให้ CPU

การใช้งาน Timer/Counter1 ในโหมด PWM

การทำงานในโหมด PWM ของ Timer/Counter1 จะสามารถเลือกใช้งานได้ 8,9 หรือ 10 บิท โดยเอาท์พุตที่ได้จะออกที่ขา PD5(OC1A) และขา PD(OC1B) ในการทำงาน Timer/Counter1 จะนับขึ้นและนับลง ซึ่งจะนับขึ้นจาก 0000 ถึงค่าสูงสุด(ตามที่กำหนดในตารางที่ 13) และจะนับจากค่าสูงสุดลงมาที่ 0000 แล้วจึงนับขึ้นอีกครั้ง

เมื่อค่าใน Timer/Counter1 เท่ากับค่าในรีจีสเตอร์ OCR1A หรือ OCR1B จะทำให้ขา PD5 (OC1A) /PD1 (OC1B) เปลี่ยนแปลงตามที่กำหนดในบิท COM1A/CoM1A0 หรือ CoM1B/COM1B0

เมื่อ OCR1 มีค่าเท่ากับ 0000 หรือค่าสูงสุดจะทำให้เอาท์พุตขา OC1A/OCA1B มีลอจิกเป็น LOW หรือ HIGH ตามที่กำหนดในบิท COM1A1/COM1A0 หรือ COM1B1/COM1B0และเมื่อ Timer/Counter1 เกิด Overflow และค่าการนับเป็น 0000 จะทำให้บิท TOV1 เซ็ตเป็น 1

Timer2&Counter

เป็น Timer / Counter ขนาด 8 บิท ต่อไปจะกล่าวถึงรายละเอียดของรีจีสเตอร์ใช้งานใน Timer/Counter 2

The Timer/Counter 2 Control Register – TCCR2

Bit 7 - Res:Reserved Bit

ใน AT90S4434/8535 บิทนี้สงวนไว้

Bit 6 - PWM2: Pulse Width Modulator Enable

เป็นบิทที่ใช้ Enable ให้โหมด PWM ใน Timer/Counter2 ให้ทำงาน โดยถ้าบิทนี้เซ็ตเป็น 1 จะเป็นการกำหนดให้โหมด PWM ถูก Enable ให้ทำงานแต่ถ้าบิทนี้ถูกเคลียเป็น 0 จะเป็นการ Disable ไม่ให้โหมด PWM ใน Timer/Counter2 ทำงาน

Bit 5, 4 - Com21, Com20: Compare Output Mode, bit 1 and 0

เป็นบิทที่ใช้กำหนดลักษณะสัญญาณที่ขา PD7 (OC2) **เมื่อ** Timer/Counter2 **ทำงานในโหมด** Compare **โดยเมื่อ** Compare Output Match **จะทำให้ขา** PD7 (OC2) **เป็นไปตามที่กำหนดในบิท** Com21 **และ** Com20

Bit 3-CTC2: Clear Timer/Counter on Compare Match

เป็นบิทที่ใช้กำหนดให้ Timer2/Counter2 **ทำการ** RESET **ค่าเป็น** 00 **หลังจากที่ค่าในรีจีสเตอร์** TCNT2 **มีค่าเท่ากับค่าที่ตั้งไว้ในรีจีสเตอร์** OCR **หรือ** Compare Output Match **ถ้าบิทนี้เซ็ตเป็น** 1 **จะทำให้** Timer/Counter2 **รีเซ็ต**

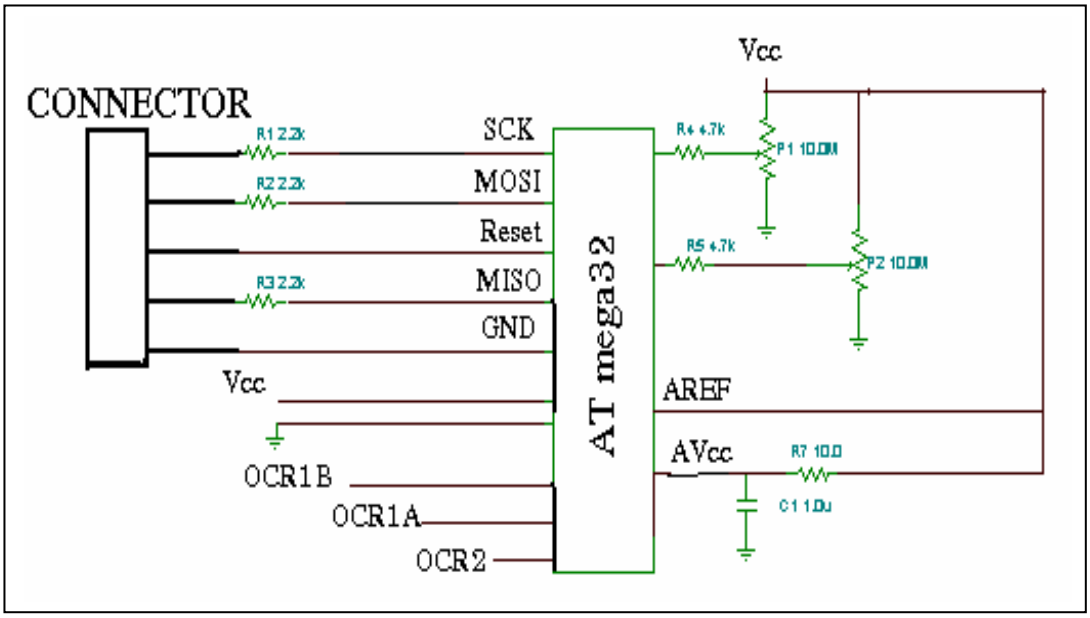
Bits 2, 1, 0-CS22, CS21, CS20: Clock Select bit 2, 1 and 0

เป็นบิทใช้ในการกำหนด ค่า Prescaling

บทที่ 3
การออกแบบ

3.1 ไมโครคอนโทรลเลอร์ ATmega32

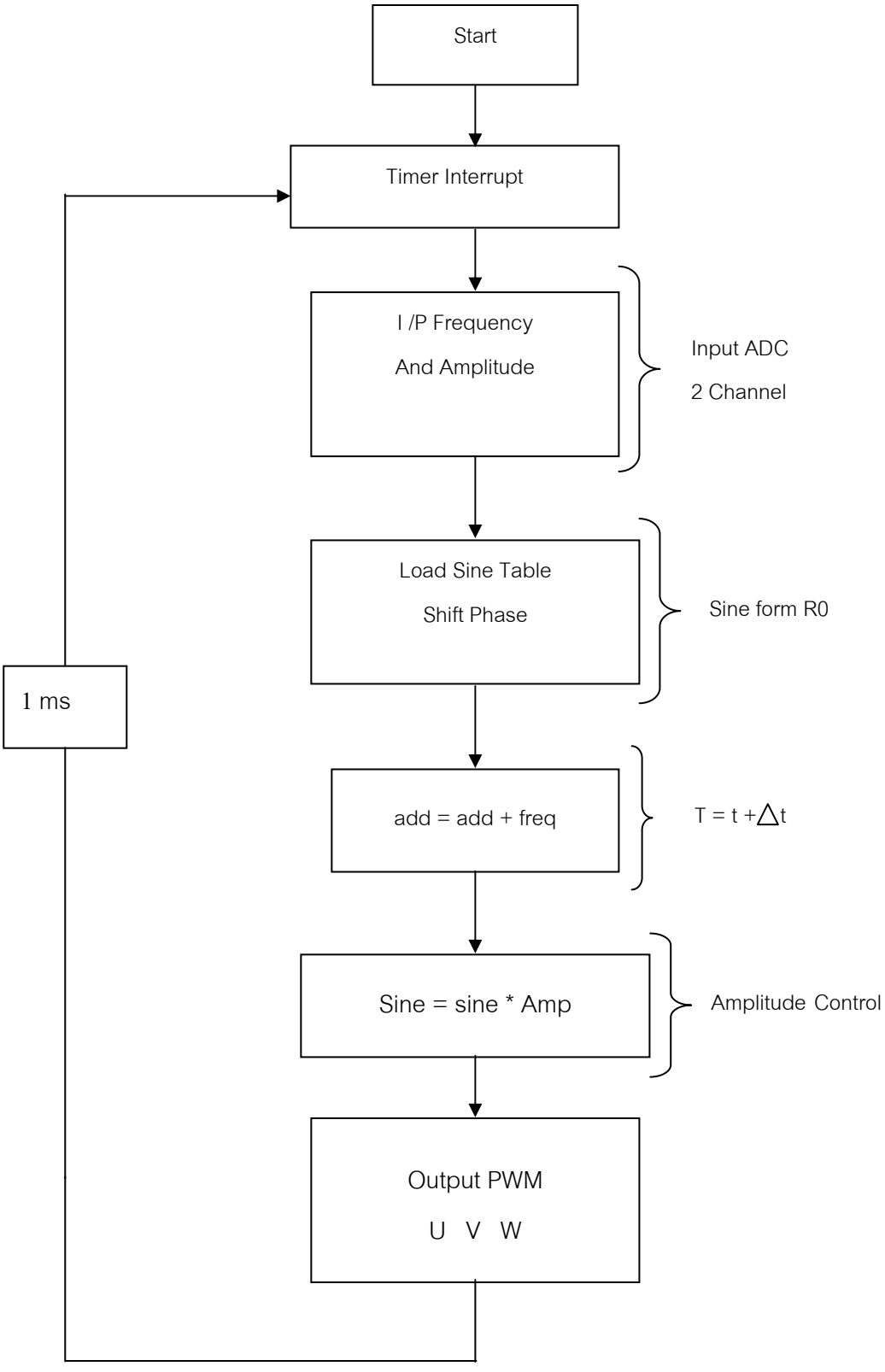
เป็นไมโครคอนโทรลเลอร์ในตระกูล AVR ทีมี Mode ของการทำงานใน Modeของ Pulse Width Modulation (PWM) จำนวนทั้งหมด 4 Port แต่เลือกใช้เพียง 3 Port และ Clock ภายในของ ATmega32 สูงถึง 8 MHz ก็เพียงพอกับการใช้งาน การต่อใช้งานจริงของไมโครคอนโทรลเลอร์เข้ากับบอร์ดสำเร็จรูปจะต่อเฉพาะส่วนที่ใช้งานและจำเป็นเท่านั้น



รูปที่ 3.1 การต่อใช้งานจริงของไมโครคอนโทรลเลอร์

การทำงานของโปรแกรม จะเริ่มจากเขียนค่าของ Sine Table เข้าสู่โปรมแกรมแล้วกระโดดเข้ามาทำ Timer Interrupt จากนั้นรับค่า Input จากภายนอกโดยผ่านทางความต้านทานปรับค่าได้ 2 ตัวโดนตัวแรกจะใช้เป็นค่าของ Frequency ส่วนตัวที่ 2 จะเป็นค่าของ Amplitude จากนั้นก็ Load Sine Table เข้ามาในโปรแกรมแล้วก็มีการ Shift Phase ให้ได้ทั้ง 3 Phase โดยแต่ละเฟสจะห่างกัน 120 องศา เพื่อให้ใช้ได้กับมอเตอร์สามเฟส

Flow Chart ของโปรแกรม

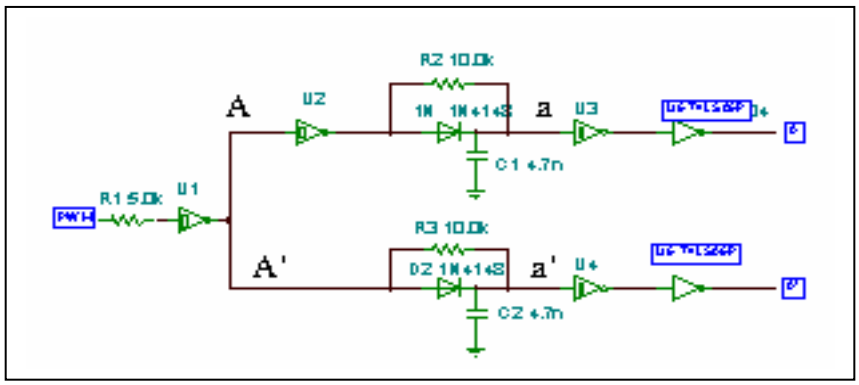


รูปที่ 3.2 Flowchart แสดงการทำงานของโปรแกรม

3.2 วงจรเดดไทม์ (Dead Time Circuit)

เนื่องจากวง ON - OFF ของ MOSFETไม่ได้เกิดขึ้นทันทีทันใด ฉะนั้นการนำสัญญาณของ PWM ที่ได้จากไมโครคอนโทรลเลอร์มาขับ โดยตรงนั้นจะทำให้เกิดการ Overlap กัน

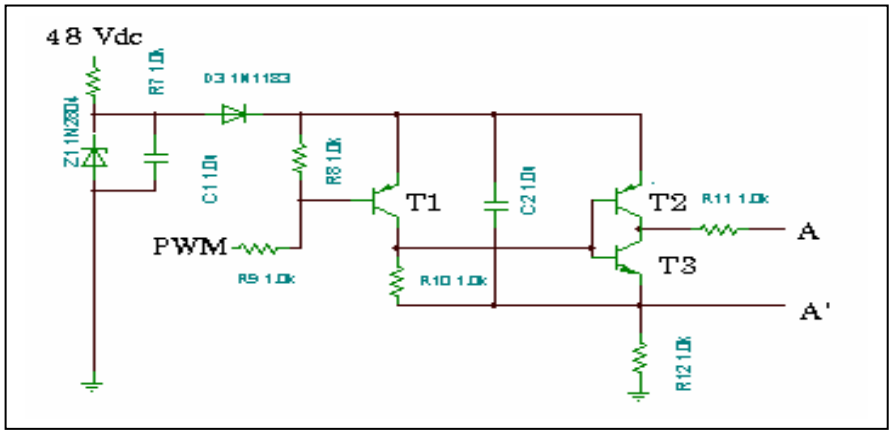
ดังนั้นจึงต้องสร้างสัญญาณที่เกิดจากการ Shift Phase ด้วยการสร้างวงจร Dead Time ระหว่างสัญญาณด้วย เพื่อที่จะให้ MOSFET ON – OFF สลับกัน



รูปที่ 3.3 แสดงวงจร Dead Time Circuit

3.3 วงจรขับ (Driver Circuit)

เป็นวงจรที่จะช่วยขยายสัญญาณจากวงจร Driver Circuit ให้เหมาะกับการทำงานของ MOSFET และจะเป็นวงจรที่แยกการทำงานของ MOSFET ออกเป็นแต่ละเฟส



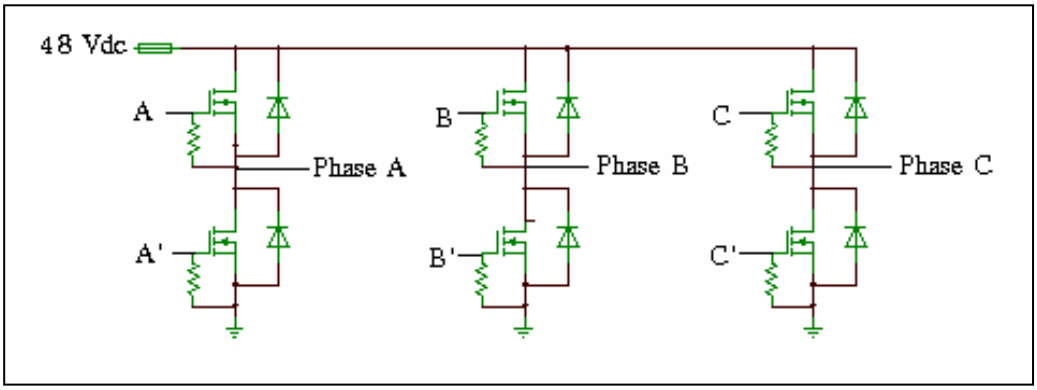
รูปที่ 3.4 แสดงวงจร Driver Circuit เพียงเฟสเดียว

การทำงานของวงจรจะเริ่มจากเมื่อมีสัญญาณ PWM ที่เป็น Low เข้ามาจะทำ T1 มีแรงดันตกคร่อม R8 ส่งผลให้ T1 ทำงานและเมื่อ T1 ทำงานจะเกิดแรงดันตกคร่อม R10 และจะทำให้การทำงานของ T2 และ T3 จะทำงานแต่ทั้งสองตัวจะต้องไม่ทำงานโดยจะต้องมีช่วงเวลาที่เหมาะสมหากแต่โดรงงานเป็นแบบแรงดันต่ำจึงสามารถให้ Transistor ทั้งสองตัวทำงานในเวลาที่ไม่ห่างกันมากนัก

รูปที่ 3.5 แสดงวงจร Driver Circuit **ทั้งหมด**

3.4 วงจรหลัก (Main Circuit)

การทำงานของวงจรจะอธิบายในส่วนของเฟส A ส่วนเฟสอื่นนั้นเหมือนกันเริ่มจากในวงจรกำลังนั้นจำเป็นอย่างยยิ่งที่จะต้องมีอย่างแรกคือ Fuse เพราะหากเกิดการทำงานที่ผิดพลาดแล้วจะได้มีตัวที่ช่วยป้องกันวงจรการทำงานนั้นตัวบนกับตัวล่างจะทำงานสลับกับตามที่ได้อธิบายก่อนหน้านี้การต่อ Mosfet กับอุปกรณ์ระบายความร้อนนั้นจะต้องระวังให้ดีและต้องต่อให้ถูกต้องตามวิธีคือมียางรองแท่นน๊อตด้วย เพราะหากการทำงานผิดจะเกิดการระเบิดขึ้นได้



รูปที่ 3.6 แสดงการต่อใช้งาน Main Circuit

3.5 มอเตอร์ไฟฟ้าสามเฟสแรงต่ำ

การดัดแปลงมอเตอร์ไฟฟ้าเพื่อใช้กับแรงดันไฟฟ้าต่ำนั้นจำเป็นต้องมีการเปลียนแปลง Parameter เดิมขมอเตอร์เพื่อให้เหมาะสมและสามารถใช้งานได้กับแรงดันไฟฟ้าต่ำได้

รายละเอียดของมอเตอร์ที่ใช้ในการทดลอง

Three Phase Induction Motor

ขนาด 2 Hp 4 Pole

ความถี่ 50 Hz

Voltage 220/380

Current 6.0/3.5

ชุดขดลวด	ค่า Parameter (mH)
U - X	16.13
V - Y	16.13
W - Z	16.13

ตารางที่ 3.1 ค่า Inductance ก่อนดัดแปลงพันขดลวดใหม่

การออกแบบมอเตอร์ไฟฟ้า 3 เฟสแรงดันต่ำ

นั่นก็คือค่า L ตัวใหม่จะน้อยกว่าเดิม 77/6 เท่า จะได้ค่า L_2 เท่ากับ 16 mH x (6/77) = 1.25 mH

เมื่อได้ค่า L ตัวใหม่แล้วต่อไปจะเป็นการคำนวณหาจำนวนรอบที่เราจะพันเส้นลวดลงในแต่ละร่องสล็อต โดยใช้ค่า L ตัวใหม่คือ 1.25 mH โดยเทียบอัตราส่วนจาก

$L_1 = KZ_1^2$

$K = 16\ mH / 46^2$

เมื่อ L_1 คือ ค่าอินดักแตนซ์ที่เราวัดก่อนทำการดัดแปลง

Z_1 คือ จำนวนตัวนำในหนึ่งร่องสล็อต

ในทำนองเดียวกันจาก $L_2 = KZ_2^2$

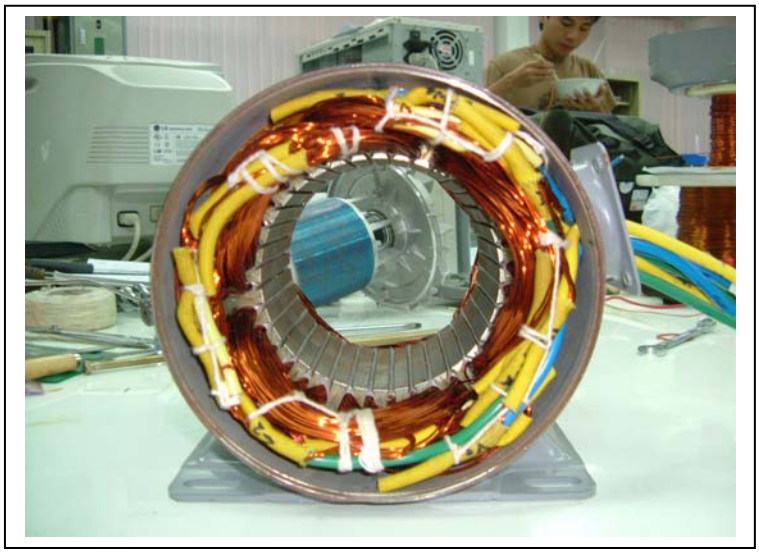
ดังนั้น $Z_2 = (1.25mH/7.65x10^{-6})^{1/2}$

$= 12.78 \approx 13$ รอบต่อสล็อต

ในการพันขดลวดเราต้องการจำนวนรอบของเส้นลวดน้อยลงและต้องการพื้นที่หน้าตัดเพิ่มขึ้น ดังนั้นเราจะใช้ขดลวดที่พันได้ขนานกัน 3 ขด เพื่อจะให้พื้นที่หน้าตัดใหญ่กว่าเดิม แต่ค่าอินดักแตนซ์เท่าเดิมนั่นก็คือในหนึ่งร่องสล็อตจะมีเส้นลวดตัวนำทั้งหมด 13 x 3 = 39 รอบ/สล็อต ซึ่งในการพันมอเตอร์เราจะพันในลักษณะเดิมโดยการพันจะเป็นแบบก้นหอย(Double Layer)

ขดลวด	ค่าอินดักแตนซ์ (mH)
U – X	1.10
V – Y	1.08
W - Z	1.28

ตารางที่ 3.2 เป็นการวัดค่า Inductance หลังจากที่ดัดแปลงมอเตอร์เสร็จแล้ว



รูปที่ 3.7 แสดงมอเตอร์หลังการพันใหม่

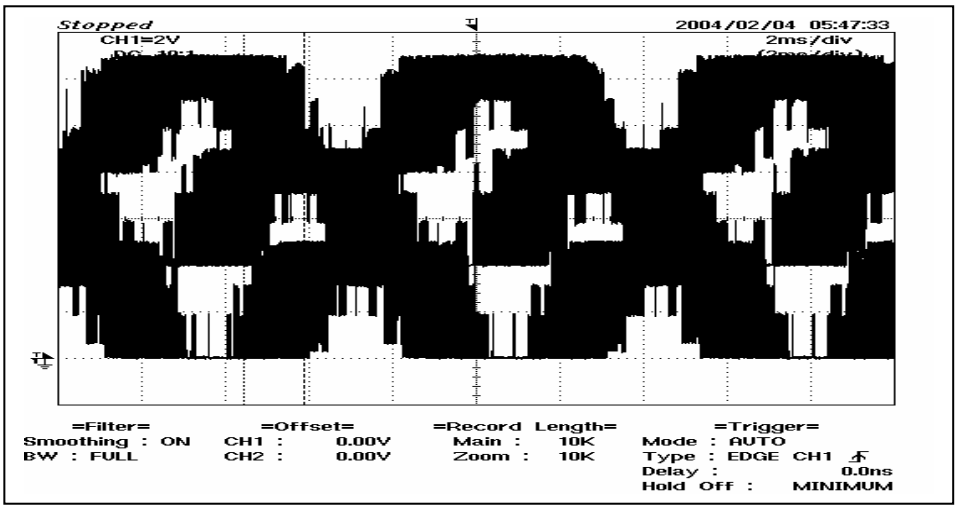
มอเตอร์หลังการพันขดลวดใหม่ต้องทำใหม่ภายนอกเหมือนเดิมเพื่อที่จะให้ Rotor สามารถใส่เข้าเหมือนเดิมได้โดนไม่ให้ขดลวดมีรอยขีดข่วนเมื่อพันขดลวดหากมองดูแล้ว สังเกตเห็นว่าขดลวดบริเวณโดยรอบข้าง Stator เกินออกมาเราก็ต้องใช้เชือกรัดโดยรอบให้แน่นหนา

บทที่ 4
การทดลองและผลการทดลอง

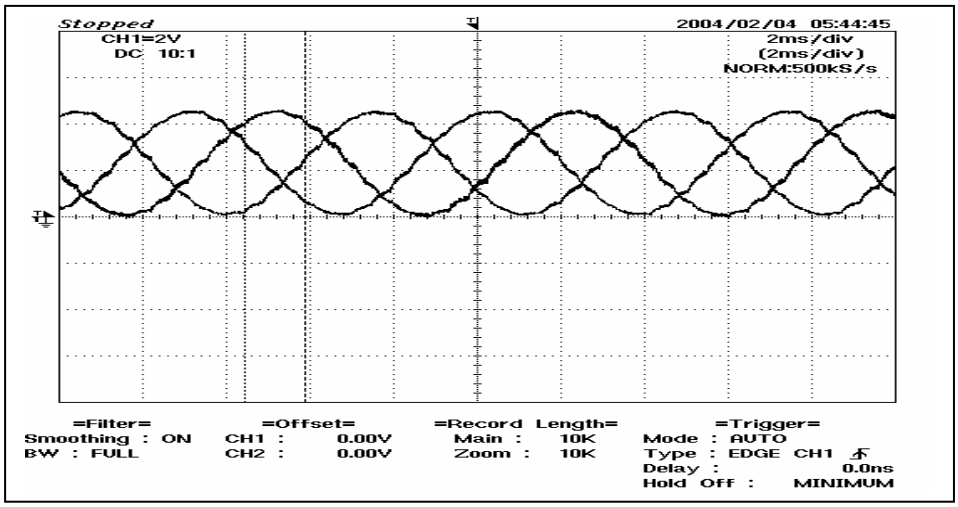
ในการทดลอง เราจะทำการทดลองเป็นขั้นตอนทีละขั้นตอนโดยจะเริ่มแต่ละวงจรดูผล output ที่ออกมา เราจะสังเกตว่าสามารถทำงานร่วมกับวงจรอื่นได้ ซึ่งผลที่ได้จะเป็นส่วน ๆ ตาม Block diagram ที่เราออกแบบเอาไว้ ซึ่งจะได้ผลเป็นข้อ ๆ ดังนี้

4.1 ผลการทดลองจากไมโครคอนโทลเลอร์

ผลการทดลองของไมโครคอนโทรลเลอร์โดย Output ที่ได้ออกมาเป็น Pulse PWM จับสัญญาณดูทั้งสามเฟส

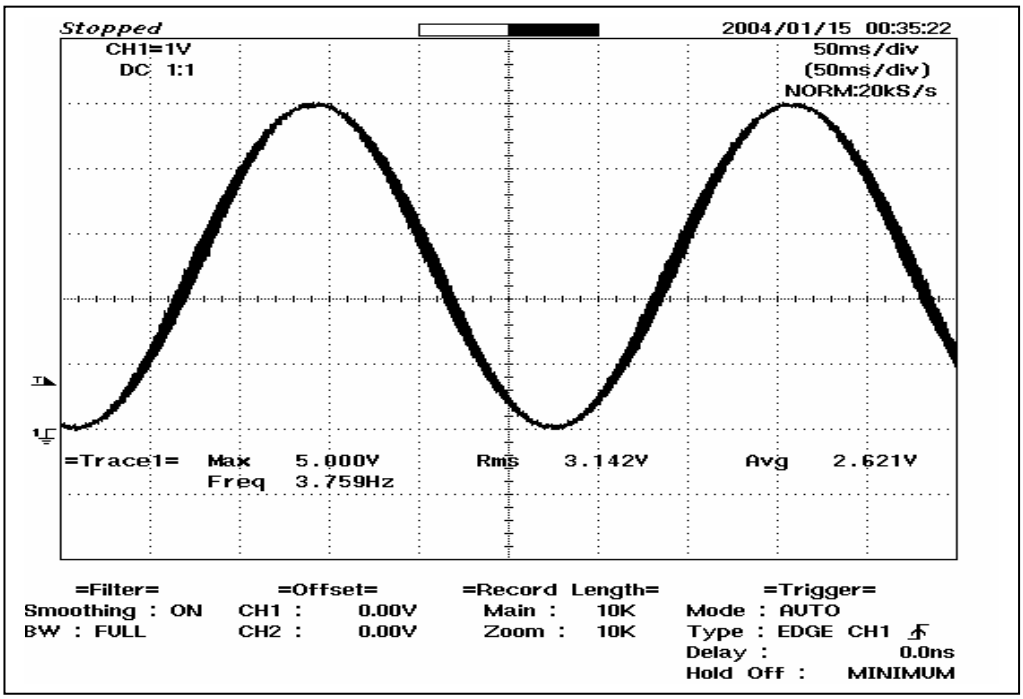


รูปที่ 4.1 แสดงสัญญาณ Output ที่ได้จาก ไมโครคอนโทรลเลอร์ขณะไม่ได้ปรับ Scope

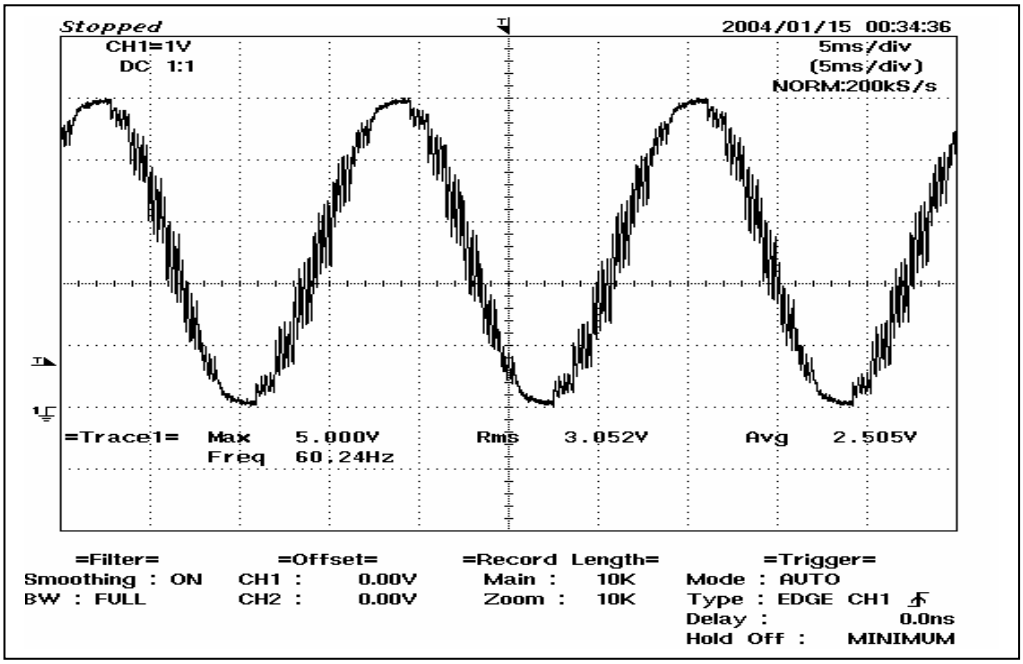


รูปที่ 4.2 แสดงสัญญาณ Output ที่ได้จาก ไมโครคอนโทรลเลอร์เพื่อให้ดูง่ายต่อผ่าน Filter

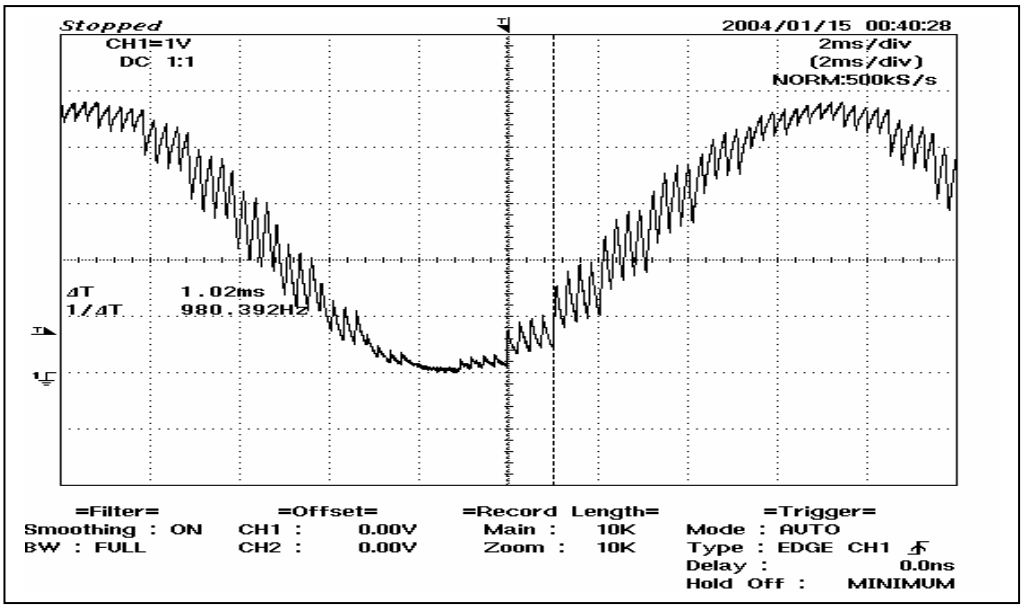
ทดลองปรับเปลี่ยนค่าความถี่ของสัญญาณที่ได้ออกมาเพื่อดูค่าความถี่ต่ำสุดและสูงสุดและเป็นการดูให้แน่ใจว่าการทำงานของโปรแกรมมีการเปลี่ยนแปลงแต่ละรอบมีค่าเท่าไร



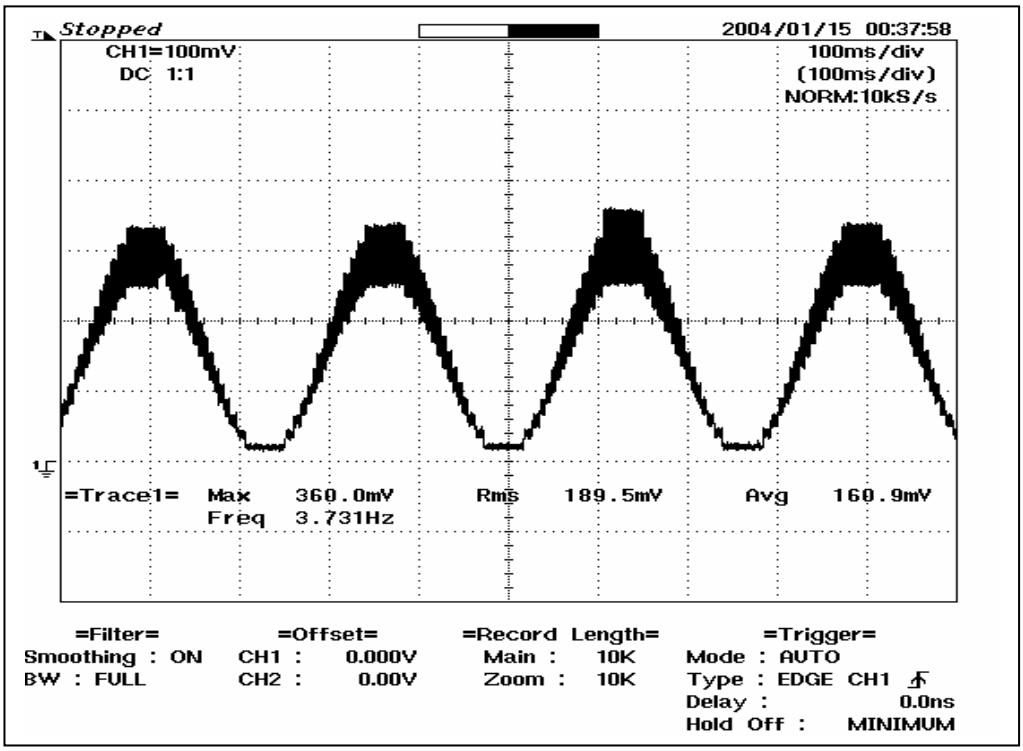
รูปที่ 4.3 แสดงการปรับค่าความถี่ของสัญญาณ Output ที่ความถี่ 3.759 Hz



รูปที่ 4.4 แสดงการปรับค่าความถี่ของสัญญาณ Output ที่ความถี่ 60.24 Hz



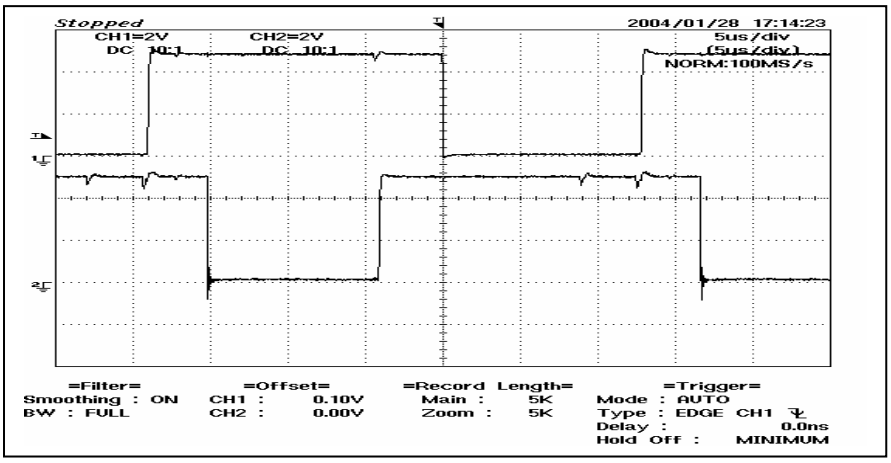
รูปที่ 4.5 แสดงสัญญาณที่ความถี่ต่ำสุดและสูงสุดพร้อมด้วยการเปลี่ยนแปลงการทำงาน



รูปที่ 4.6 การเปลี่ยนแปลงโดยให้ค่า V/f คงที่ ที่ความถี่ต่ำสุด

จากรูปเป็นการคูณสัญญาณโดยมีค่าคง V/f อยู่ที่ระดับแรงดันต่ำ ๆจะมีผลน้อยเมื่อเทียบกับสัญญาณเดิมแต่พอที่ระดับแรงดันของสัญญาณมาขึ้นจะทำให้ได้สัญญาณดังรูป

4.2 ผลการทดลองของวงจร Dead Time



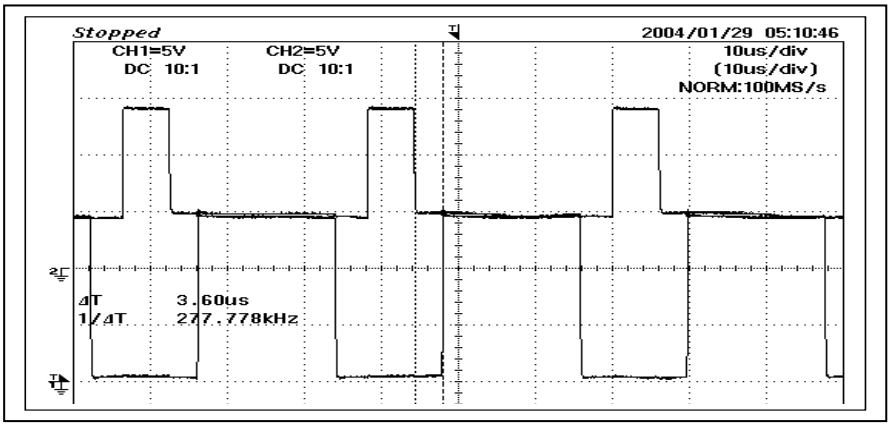
รูปที่ 4.7 แสดงสัญญาณOutput จาก Dead Time

การทำงานจะเริ่มจากตำแหน่ง A มี logic “0” แรงดัน a logic “1” Output ที่ได้เป็น “0” และเมื่อ A มี logic “1” จะค่อยลดลงจากผลของการ discharge ของ capacitor จนถึงระดับแรงดันที่รับรู้ว่าเป็น 1 จะทำให้ Output logic “0”

ขณะที่A’ มี logic “0” จะทำให้แรงดันของ a เป็น logic “0” ทำให้ Output ที่ได้เป็น logic “1” และในขณะที่ A’ มี logic “1” จะทำให้ระดับของแรงดันที่ a เป็น logic “1” จะทำให้ Output ที่ได้ออกมามี logic “0”

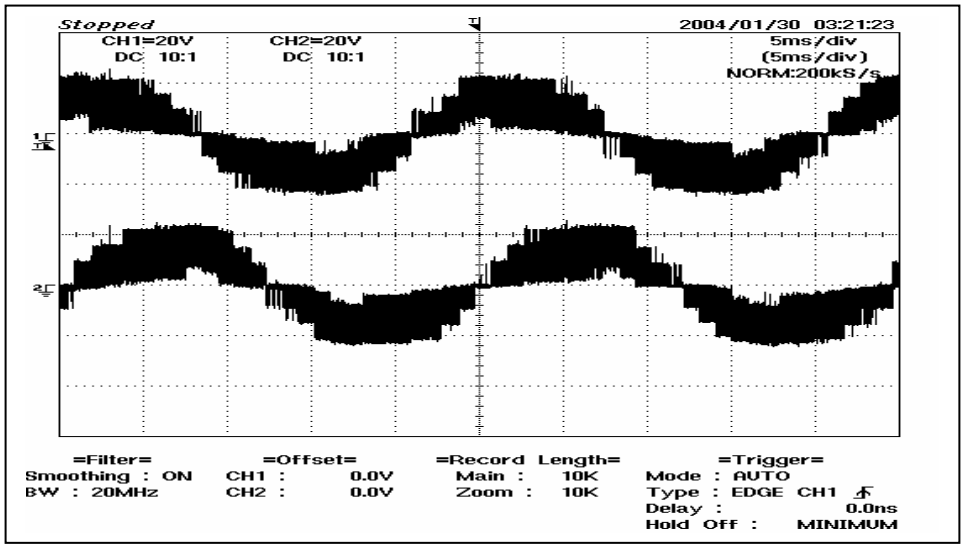
4.3 ผลการทดลองของวงจร Driver Circuit

จากรูปเป็นการทำงานของวงจร Driver Circuit แต่ยังไม่จ่ายไฟให้กับวงจรกำลังเพราะต้องตรวจสอบการทำงานของแต่ละวงจรไปเรื่อย ๆต้องให้แน่ใจว่า Driver Circuit ทำงานถูกต้องจริง สัญญาณรูปล่างจะเป็นสัญญาณที่ออกจาก A ส่วนสัญญาณรูปบนจะเป็นสัญญาณที่ออกจาก A’ ซึ่งทั้งสองสัญญาณจะมีช่วงการทำงานที่เป็นช่วง High ที่ไม่ทับซ้อนกัน

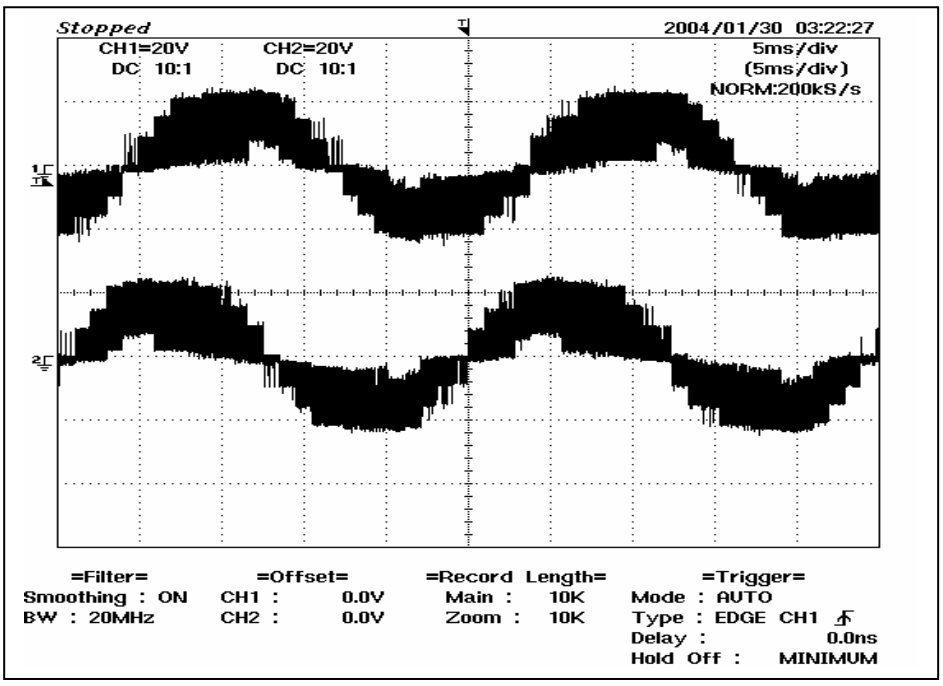


รูปที่ 4.8 แสดงสัญญาณOutput จาก Driver Circuit

4.4 ผลการทดลองของวงจร Main Circuit



รูปที่ 4.9 เปรียบเทียบสัญญาณระหว่าง Phase A กับ Phase B

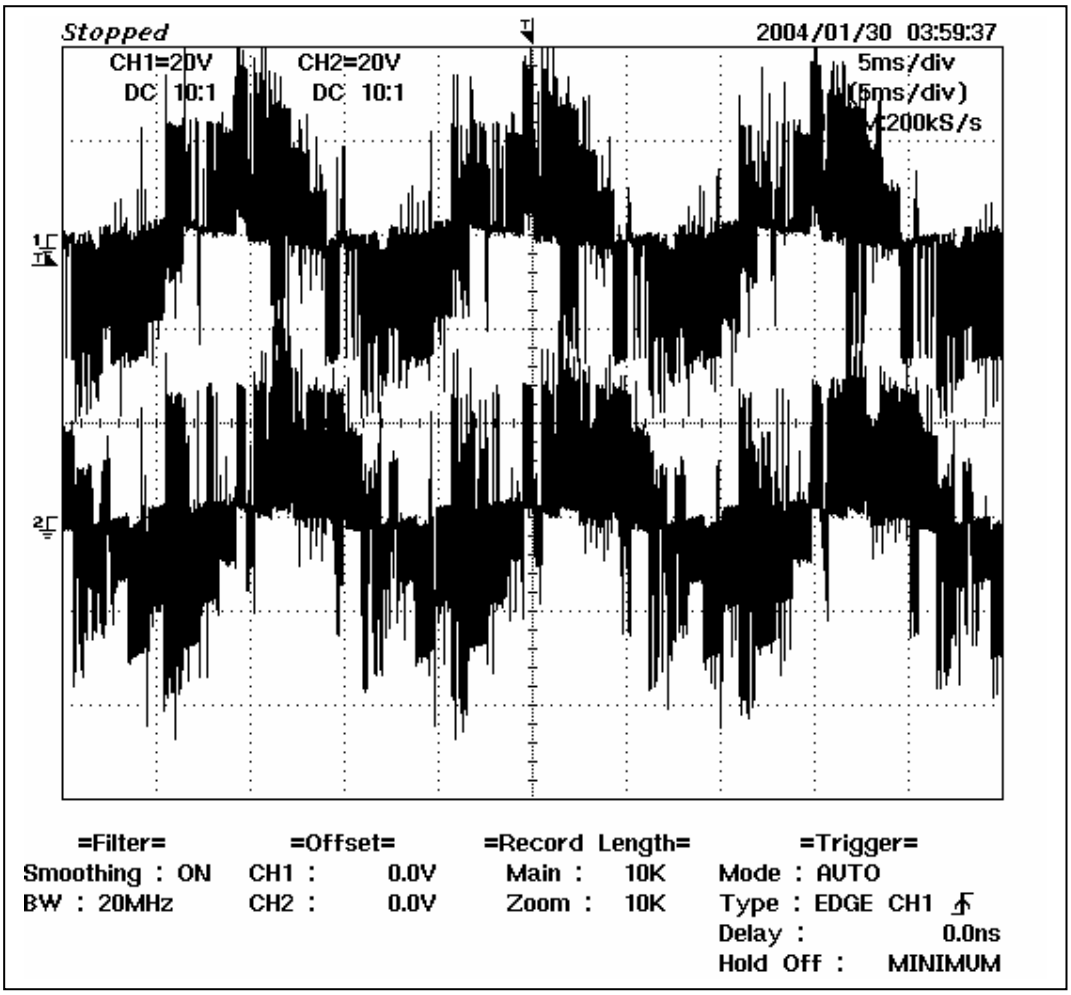


รูปที่ 4.10 เปรียบเทียบสัญญาณระหว่าง Phase A กับ Phase B

รูปข้างบนเป็นการแสดงการวัดสัญญาณขณะจ่ายไฟให้กับ Mosfet เพื่อทดลองดูว่าเมื่อจ่ายให้กับมอเตอร์แล้วจะเป็น Sine wave จริง ๆและมีการ Shift Phase กันด้วย

4.5 การทดลองขับมอเตอร์แรงดันต่ำ

สัญญาณที่ออกมาจาก Main Circuit จะเป็นสัญญาณ Sine wave ซึ่งแต่เฟสจะมีเฟสไม่ตรงกัน และเมื่อนำไปขับมอเตอร์ก็จะมีลักษณะคล้ายกันแต่จะมี Harmonic ปะปนอยู่ การที่ปรับค่าความถี่ที่ไมโครคอนโทรลเลอร์ จะมีผลต่อการเปลี่ยนแปลงของความเร็วมอเตอร์ด้วย ดังแสดงในรูป 4.1



รูปที่ 4.11 แสดงสัญญาณ sine wave ขณะต่อมอเตอร์หมุน

บทที่ 5
สรุปวิจารณ์และข้อเสนอแนะ

5.1 สรุปผลการทำงาน

ในการทดลอง เมื่อเราประกอบวงจรทั้งหมดเข้าด้วยกันเสร็จแล้ว ซึ่งเราจะทดลองสัญญาณที่ละจุดว่า output เป็นอย่างไร ก่อนที่จะนำไปขับมอเตอร์ที่เราออกแบบเอาไว้ โดยเราจะเริ่มพิจารณาจากสัญญาณที่ออกจากขาของไมโครคอนโทลเลอร์ และสัญญาณที่ออกจะเป็นรูป sine wave ที่สามารถคอนโทลค่า V/f ที่อยู่ในระดับแรงดันต่ำ ๆ จากนั้นจะนำไปต่อเข้ากับ input ของ Dead time circuit เพื่อที่จะให้สัญญาณที่ออกมาแต่ละเฟสเกิดการสลับกัน ซึ่งจะมีเฟสห่างกันประมาณ 5 $\mu\sigma$ และนำสัญญาณ Dead time ที่ได้ไปป้อนให้กับ Driver circuit และเมื่อเราใช้ Scope จับที่สัญญาณ Output Driver circuit แต่ยังไม่จ่ายไฟให้กับระบบกำลัง เพื่อจะดูรูปร่างสัญญาณที่ออกจาก Channel A และรูปร่างของ Channel A' ซึ่งทั้งสองสัญญาณจะมีช่องการทำงานเป็นช่วง High ที่ไม่ซับซ้อนกัน ก่อนจะจ่ายให้กับ Main circuit จากนั้นเราทำการวัดสัญญาณจาก output ของ Main circuit ทั้ง 3 channel ซึ่งต้อง shitf phase กัน พอที่จะขับมอเตอร์ นั่นคือประมาณ 120 องศาไฟฟ้า เมื่อได้เช่นนั้นเราจึงนำมอเตอร์ต่อและทดสอบขับมอเตอร์

จากการทดลองที่ผ่านมา คือ จ่ายแรงดันฟ้ากระแสตรงให้กับมอเตอร์ไฟฟ้าที่มีการพันขดลวดใหม่ปรากฏว่ามอเตอร์นั้น สามารถหมุนทำงานได้ตามวัตถุประสงค์ของโครงการ ซึ่งจะเห็นว่าหากเราต้องการนำมอเตอร์ไฟฟ้าสามเฟสไปใช้งานเฉพาะอย่างก็สามารถทำได้และโครงการนี้ยังจะสามารถพัฒนาต่อไปได้อีก เพียงแต่ต้องใช้ให้เหมาะกับงานนั้น ๆ

5.2 ปัญหาที่พบและแนวทางการแก้ปัญหา

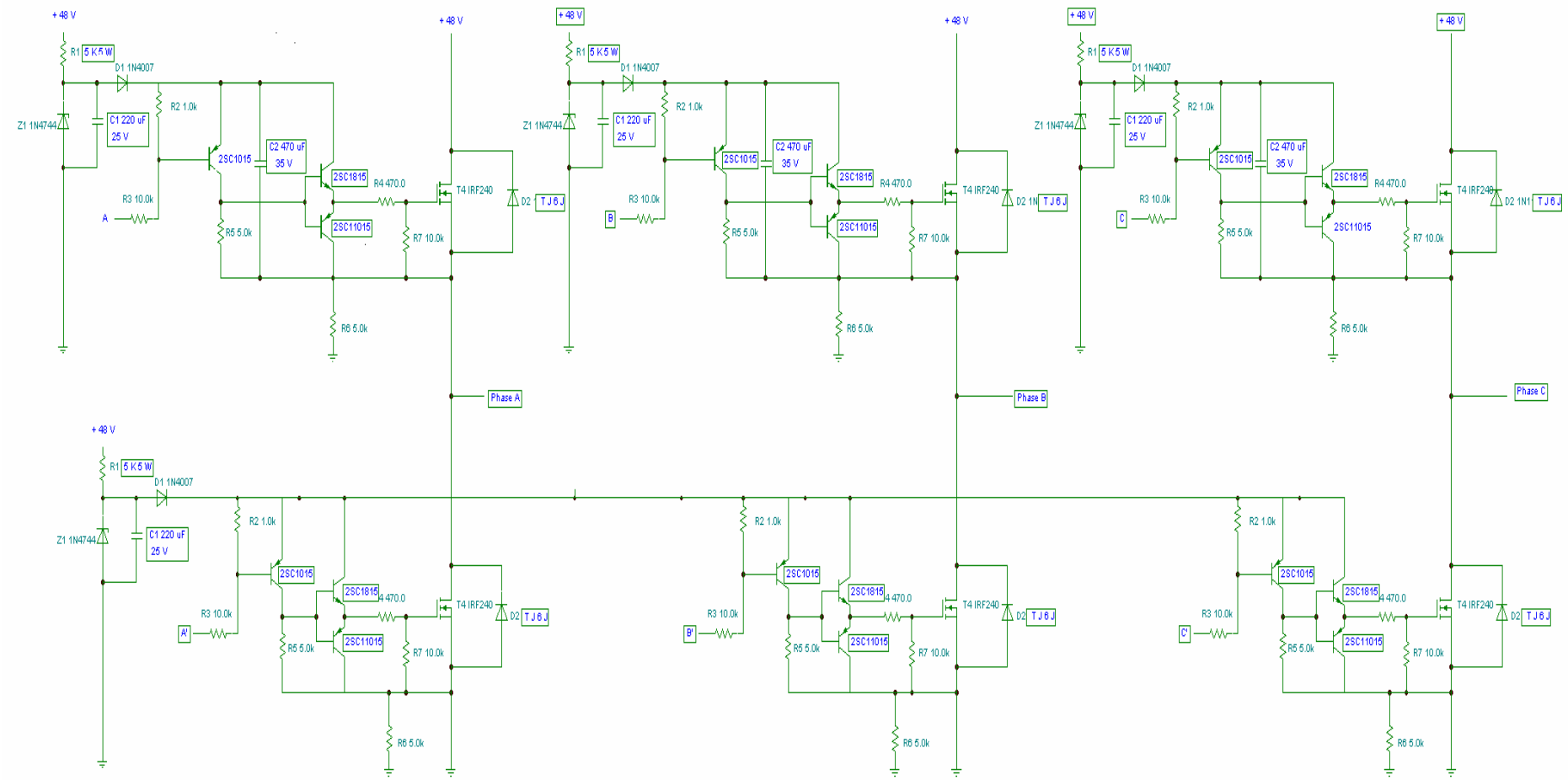
เมื่อเราทดสอบวงจรทั้งหมด การจ่ายไฟจาก Battery เพื่อทดสอบวงจร จะต้องเริ่มจ่ายไฟฟ้าจากแรงดันต่ำแล้วดูการทำงานของวงจรต่าง ๆก่อนเพราะถ้าหากเราจ่ายให้วงจรมากหรือนานเกินไปอาจจะทำให้อุปกรณ์อิเล็คทรอนิกส์บางตัวที่ทนกระแสได้ เกิดการเสียหายซึ่งโดยมากจะเป็นอุปกรณ์พวก MOSFET และ Transistor เพราะถ้าเกิดว่ามีอุปกรณ์ตัวใดตัวหนึ่งในวงจรเสียวงจรจะทำงานผิดพลาดทำให้เกิดการเพี้ยนและการหาว่าตรงจุดไหนเสียนั้นจะยุ่งยากพอสมควร

ในการทำงานนั้นจะต้องมีการลองผิดลองถูกพอสมควร เพราะเมื่อเราต่อวงจรใช้งานจริงวงจรอาจจะเกิดการเพี้ยนหรือทำงานผิดพลาดไม่ตรงตามที่เราต้องการ จึงจำเป็นต้องมีการดัดแปลงและประยุกต์วงจรเพื่อให้วงจรสามารถทำงานได้ในการทำงานเมื่อมีปัญหาควรปรึกษาผู้มีประสบการณ์ เพราะปัญหาในการทำงานส่วนมากจะด้านเทคนิคซึ่งนักศึกษาเองยังขาดอยู่ ดังนั้นคำปรึกษาจากท่านอาจารย์จะเป็นประโยชน์อย่างมากเกี่ยวกับการแก้ปัญหาในการทำโครการอย่างยิ่ง

ภาคผนวก ก. โปรแกรมการควบคุมมอเตอร์

ภาคผนวก ข. คู่มือการใช้งานโปรแกรม AVRstudio4

ภาคผนวก ค. รายละเอียดอุปกรณ์ IC



รูปที่ 3.5 วงจร Drive สำหรับขับ Main Circuit

เอกสารอ้างอิง

1. โคทม อารียา ,**อิเล็กทรอนิกส์ 2** ,2544 ,กรุงเทพมหานคร :ซีเอ็ดยูเคชั่น.
2. สมบูรณ์ มาลานนท์ และคณะ ,**แหล่งจ่ายไฟแบบสวิตชิ่ง** (Switching Power Supply), กรุงเทพมหานคร:ฟิกส์เซ็นเตอร์.
3. นภัทร วัจนเทพินทร์,**อิเล็กทรอนกส์กำลัง 2 ภาคปฏิบัติ**, 2544 ,กรุงเทพมหานคร : สกายบุกส์.

```
;// Program Low - Voltage Drive Motor
.include     "m32def.inc"
.def temp = r16
.def adcdata2 = r17
.def constant = r18
.def constant1 = r19
.def constant2 = r20
.def constant3 = r21
.def constant4 = r22
.def adcdata1  = r23
.def sin1  = r24
.def sin2  = r25
.def sin3  = r26
.equ phaseshift = 0x56
.def phaseconstant = r27

.org   0x0000
       rjmp RESET

.org   0x0016 ; t0 overflow
       rjmp  update_pwm

.org 0x0100

sine:          ; 256 step sinewave table
       .DB
       0x80,0x83,0x86,0x89,0x8c,0x8f,0x92,0x95,0x98,0x9c,0x9f,
0xa2,0xa5,0xa8,0xab,0xae
       .DB
       0xb0,0xb3,0xb6,0xb9,0xbc,0xbf,0xc1,0xc4,0xc7,0xc9,0xcc,0
xce,0xd1,0xd3,0xd5,0xd8
       .DB
       0xda,0xdc,0xde,0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xed,0
xef,0xf0,0xf2,0xf3,0xf5
```

```
.DB
        0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfc,0xfd,0xfe,0xfe,0xff,
0xff,0xff,0xff,0xff
        .DB
        0xff,0xff,0xff,0xff,0xff,0xff,0xfe,0xfe,0xfd,0xfc,0xfc,0xfb,0
xfa,0xf9,0xf8,0xf7
        .DB
        0xf6,0xf5,0xf3,0xf2,0xf0,0xef,0xed,0xec,0xea,0xe8,0xe6,0xe
4,0xe2,0xe0,0xde,0xdc
        .DB
        0xda,0xd8,0xd5,0xd3,0xd1,0xce,0xcc,0xc9,0xc7,0xc4,0xc1,0
xbf,0xbc,0xb9,0xb6,0xb3
        .DB
        0xb0,0xae,0xab,0xa8,0xa5,0xa2,0x9f,0x9c,0x98,0x95,0x92,0
x8f,0x8c,0x89,0x86,0x83
        .DB
        0x80,0x7c,0x79,0x76,0x73,0x70,0x6d,0x6a,0x67,0x63,0x60,
0x5d,0x5a,0x57,0x54,0x51
        .DB
        0x4f,0x4c,0x49,0x46,0x43,0x40,0x3e,0x3b,0x38,0x36,0x33,
0x31,0x2e,0x2c,0x2a,0x27
        .DB
        0x25,0x23,0x21,0x1f,0x1d,0x1b,0x19,0x17,0x15,0x13,0x12,
0x10,0x0f,0x0d,0x0c,0x0a
        .DB
        0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x03,0x02,0x01,0x01,
0x00,0x00,0x00,0x00,0x00
        .DB
        0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x01,0x02,0x03,0x03,
0x04,0x05,0x06,0x07,0x08
        .DB
        0x09,0x0a,0x0c,0x0d,0x0f,0x10,0x12,0x13,0x15,0x17,0x19,
0x1b,0x1d,0x1f,0x21,0x23
```

```
.DB
        0x25,0x27,0x2a,0x2c,0x2e,0x31,0x33,0x36,0x38,0x3b,0x3e,
0x40,0x43,0x46,0x49,0x4c
        .DB
        0x4f,0x51,0x54,0x57,0x5a,0x5d,0x60,0x63,0x67,0x6a,0x6d,
0x70,0x73,0x76,0x79,0x7c


RESET:
            cli
            ldi         temp, HIGH(RAMEND)
            out         SPH, temp
            ldi         temp, LOW(RAMEND)
            out         SPL, temp       ; setup stack pointer

            ldi         temp,0xF0       ; out pwm
OC2,OC1A,OC1B
            out         DDRD,temp

             ;ADC
            ldi         temp, 0x60              ; AVCC, Channel 0
(0x60)
            out         ADMUX, temp
            ldi         temp, 0x87              ; 8 bit mode (0x87)
            out         ADCSR, temp



            ; set sinewave output as default

            ldi         r31,HIGH(sine*2)    ; setup Z pointer high
            ldi         r30,LOW(sine*2)             ; setup Z pointer
low
```

```
ldi         temp,0xA1
            out         TCCR1A,temp         ;Timer 1 Phase 2
(0xA1)

            ldi         temp,0x0A
            out         TCCR1B,temp         ;Timer 1 Phase 1
(0x09)

            ldi         temp,0x6A
            out         TCCR2,temp          ;Timer 2 Phase 3
(0x69)

            ldi         temp,0x04       ;set timer of program(04)
            out         TCCR0,temp

            ldi         temp,0xFE       ;Old 0xF0
            out         TCNT0,temp

            ldi         temp,0x01
            out   TIMSK,temp

            ldi         phaseconstant,phaseshift   ; shift 120 degree

            sei


mainLoop:
            nop
            rjmp  mainLoop

update_pwm:

            cbi   ADMUX, MUX0           ; Channel 0
```

```
        sbi   ADCSR,ADSC        ; start conversion
_adc0_wait:
        sbic  ADCSR, ADSC       ; wait 'til complete
        rjmp  _adc0_wait
        in          adcdata1, ADCH      ; load_adc

        sbi   ADMUX, MUX0               ; Channel 1
        sbi   ADCSR,ADSC        ; start conversion
_adc1_wait:
        sbic  ADCSR, ADSC       ; wait 'til complete
        rjmp  _adc1_wait
        in          adcdata2, ADCH      ; load_adc

        lsr         adcdata2
        lsr         adcdata2
        lsr         adcdata2
        lsr         adcdata2        ;read adcdata 4 bit

        ldi         temp,0x01       ;adcdata is not 0
        add         adcdata2,temp
        add         r30, adcdata2   ; addr + freq
        push  r30

        lpm
        mov         sin1,r0                 ;load new sine1.

        adc         r30,phaseconstant
        lpm
        mov         sin2,r0                 ;load new sine2.

        adc         r30,phaseconstant
        lpm
        mov         sin3,r0                 ;load new sine3.
```

```
        ldi     constant1,0x0F
        mul     adcdata2,constant1      ;new_adcdata =
adcdata*15
        mov     constant3,r0            ;new_adcdata

        mul   adcdata1,constant3   ;current*15
        mov    constant4,r1

        mov     constant,constant4  ;Amp_cont

        mul     sin1,constant           ;phase1
        mov     sin1,r1

        mul     sin2,constant           ;phase2
        mov     sin2,r1

        mul     sin3,constant           ;phase3
        mov     sin3,r1


        out     OCR1AL,sin1   ;out PWM phase1
        out     OCR1BL,sin2   ;out PWM phase2
        out     OCR2,sin3             ;out PWM phase3


        ldi     temp,0xF0     ;reload to timer 0
        out     TCNT0,temp

        pop   r30

        reti
```