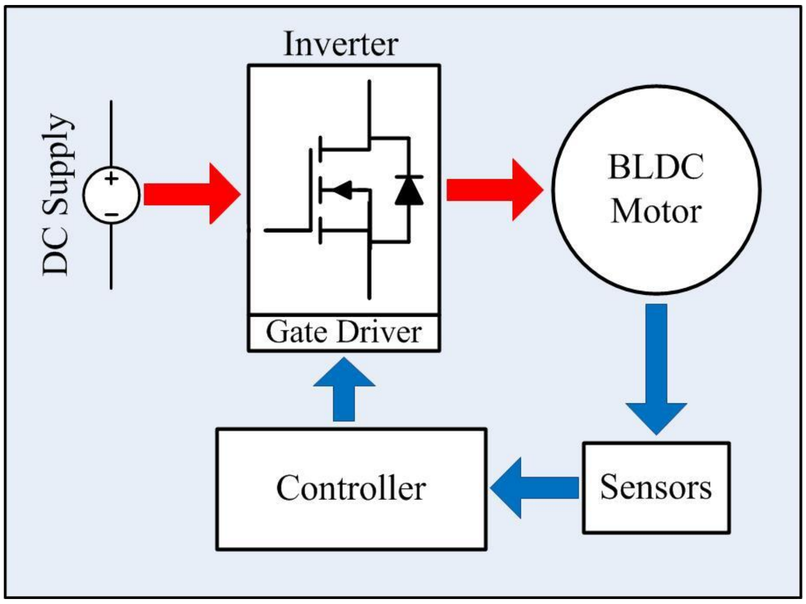
EE75

BLDC Motor Drive

This contributes 5% of the marks for ELEC2208.

This laboratory exercise is an introduction to work with brushless DC (BLDC) motor drive using three-phase inverter/electronic commutator based on Hall sensor signals. BLDC motor can be envisioned as a brushed DC motor which is turned inside out, with the permanent magnets on the rotor and windings on the stator. Rotation of rotor is achieved by creating a rotating stator flux which is always perpendicular to the rotor flux so as to create a maximum torque to "drag" the rotor. Hence, rotor position is sensed using Hall sensors to allow the controller to determine which stator windings to be energised by controlling the three-phase inverter.



Schedule

Preparation time	:	3 hours
Lab time	:	3 hours

Items provided

Tools	:	-
Components	:	1 Il Matto, experimental kit, hook-up wire, dip switch, resistors
Equipment	:	3 bench PSU, Oscilloscope
Software	:	Software for Il Matto programming e.g. AVRDUDE (v5.11 + C232HM patch), avr-gcc, UrJTAG

Items to bring

Essentials. A full list is available on the Laboratory website at https://secure.ecs.soton.ac.uk/notes/ellabs/databook/essentials/

Before you come to the lab, it is essential that you read through this document and complete ***all*** of the preparation work in section 2. If possible, prepare for the lab with your usual lab partner. Only preparation which is recorded in your laboratory logbook will contribute towards your mark for this exercise. There is no objection to several students working together on preparation, as long as all understand the results of that work. Before starting your preparation, read through all sections of these notes so that you are fully aware of what you will have to do in the lab.

Academic Integrity – *If you undertake the preparation jointly with other students, it is important that you acknowledge this fact in your logbook. Similarly, you may want to use sources from the internet or books to help answer some of the questions. Again, record any sources in your logbook.*

This exercise uses the standard **mark scheme** available on the Laboratory website at http://secure.ecs.soton.ac.uk/notes/ellabs/markscheme/

Revision History

March 12, 2017	Michail Sidorov (ms)	C codes added
August 7, 2015	Sze Sing Lee (ss.lee) & Richard Crowder (rmc)	First version of this lab created
April 11 2015	Richard Crowder	

1 Aims, Learning Outcomes and Outline

This laboratory exercise aims to:

- Develop the skills required to integrate power electronics with a microcontroller
- Demonstrate the operating principle of a BLDC motor and its drive system

Having successfully completed the lab, you will be able to:

- Implement a fix frequency three-phase inverter
- Construct the commutation tables for a BLDC
- Control a BLDC motor

The exercise introduces an electric drive system for BLDC motor. Each part of the drive system will be investigated to build up to the implementation of a fully functioning system. Experimental platform for reconfigurable power electronics converters is used. A three-phase inverter will be constructed and studied first. Then, the BLDC motor operating principle will be studied and tested manually to construct commutation tables. Having understand the theories and technical skills for three-phase inverter, Hall sensors and BLDC motor, a drive system will be implemented.

2 Preparation

Read through the course handbook statement on safety and safe working practices, and your copy of the standard operating procedure. Make sure that you understand how to work safely. Read through this document so you are aware of what you will be expected to do in the lab.

Before the lab you should read through appendices (in particular Appendix A) and understand the theoretical operating principle of a BLDC motor and its control strategy. It is also recommended that you do initial development of the two Il Matto programmes required.

- *What is a Hall sensor or a Hall effect sensor?*
- *The Hall sensor used in this experiment has "open collector" output. What is required to make this a digital output with a defined logic state, 0 or 1?*
- *What is a gate driver and why is an isolated supply (isolated DC-DC converter) required?*
- *What is the "dead-time" as applied to an inverter?*
- *If all three motor windings are energised with the same current ($i_a=i_b=i_c$), what is the torque produced and the response of the rotor?*

3 Laboratory Work

3.1 Three-Phase Voltage Source Inverter (Six Step Operation)

1. Program your Il Matto to produce 6 PWM signals as illustrated in Fig. 1. The frequency is 50Hz and the dead time is 1μs.

2. Verify the sequence of device signal you generated with the following measurements:
   <u>Measurement 1:</u>
   Time: 5m s/div, Channel 1: S1, Channel 2: S3, Channel 3: S5
   <u>Measurement 2:</u>
   Time: 5m s/div, Channel 1: S1, Channel 2: S2, Channel 3: S4, Channel 4: S6

3. Verify the dead time of PWM signal you generated with the following measurements:
   <u>Measurement 3:</u>
   Time: 500n s/div, Channel 1: S1, Channel 2: S2, Trigger at channel 1 rising edge
   <u>Measurement 4:</u>
   Time: 500n s/div, Channel 1: S1, Channel 2: S2, Trigger at channel 1 falling edge
   Repeat measurement 3 and 4 for inverter leg 2 (S3 & S4) and 3 (S5 & S6).

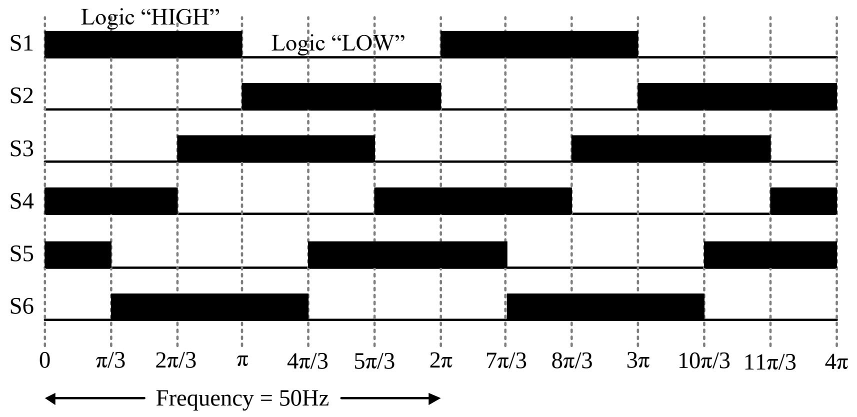


Figure 1 Device signals for three-phase inverter operating in six-step operation.

4. Construct a three-phase inverter by using the platform (shown in Appendix B) and 6 power MOSFETs.
   Note: Make sure you recognise the correct Drain and Source terminal of power MOSFETs and plug-in the correct terminals.

5. Six gate drivers are required to drive the inverter as each power MOSFET must be driven by a separate gate driver. Connect the output of each gate driver to the designated MOSFET's control terminals by using female-to-female jumpers. For power MOSFET S1, for instance "PWM(+)_S1" and "PWM(-)_S1" are connected to Gate and Source terminals (pin header) respectively. Similar connections should be done for the rest 5 power MOSFETs.

6. Connect a 12V supply to power to the gate drivers.

7. Connect PWM signals generated from Il Matto to gate drivers through "PWM" connectors. A link connector is provided.

8. Verify that all the MOSFETs are driven by the correct signals by measurement across Gate and Source terminals of individual MOSFETs.

9. By using a separate bench PSU, connect 12V to supply power to the inverter (across DC link terminals).

10. Measure the inverter output between phase A – C, B – C and A – B. *Note: Take care with the 0v connection to the oscilloscope, a third reading may require the use of the oscilloscopes math function.*

3.2 Rotor Position of BLDC Motor and Hall Sensor Signals

1. Referring to Fig. 2, Figure A4 and Table A1, energise the stator winding manually by using a bench power supply to align the rotor to a specific angle and determine the Hall sensor signals (H1, H2 and H3). Refer to steps 1 – 4 in appendix A.
Note: Do not make connections (connect or disconnect) while bench power supply is on. Turn on the supply only after the connections are properly made.

◇ *When the power supply is turned on, a short circuit is observed in the bench power supply. Why?*

2. Compute commutation tables for both clockwise (CW) and counter-clockwise (CCW) directions.

◇ *What is the relationship between both tables?*

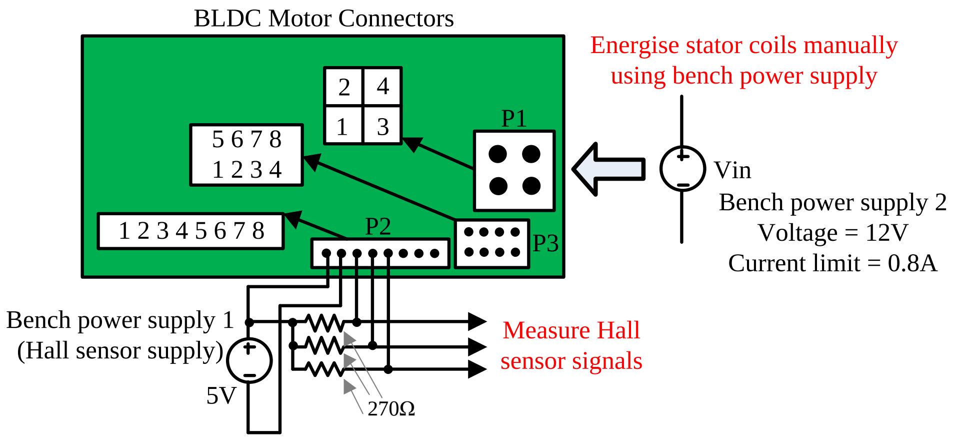


Figure 2. Energising stator winding manually using bench power supply to find out relationship between rotor position and Hall sensor signals.

3.3 Controlling BLDC Motor

1. Sketch the PWM signals of BLDC motor in CW and CCW directions with respect to the Hall sensor signals in Fig. 3.

2. Program the Il Matto to detect Hall sensor signals and produce appropriate signals to control the motor in CW direction.
Note: You can verify your program by rotating the rotor shaft slightly to different angle. Then, measure whether the device signals generated with respect to Hall sensor signals are correct.

3. Construct a system to drive the BLDC motor as shown in Fig. 4. Set V_{in} of bench power supply 3 to zero with current limit of 0.8A. Turn on the power supply and increase the voltage slowly. *Note: Do not increase the voltage beyond 24V.*

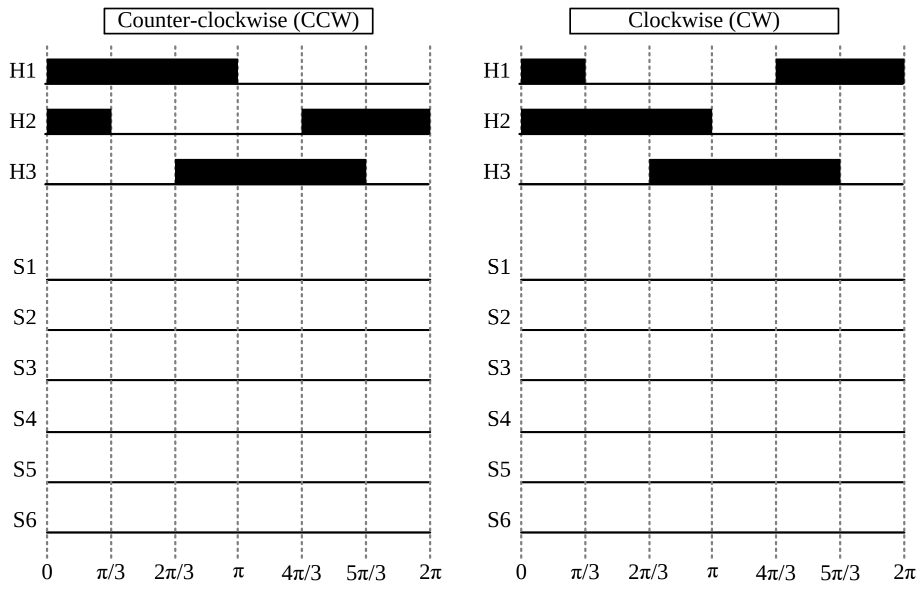


Figure 3. Device signals for BLDC motor drive based on Hall sensor signals.

Based on the drives signals in Fig. 3, how long is the dead time? Illustrated your answer in radian.

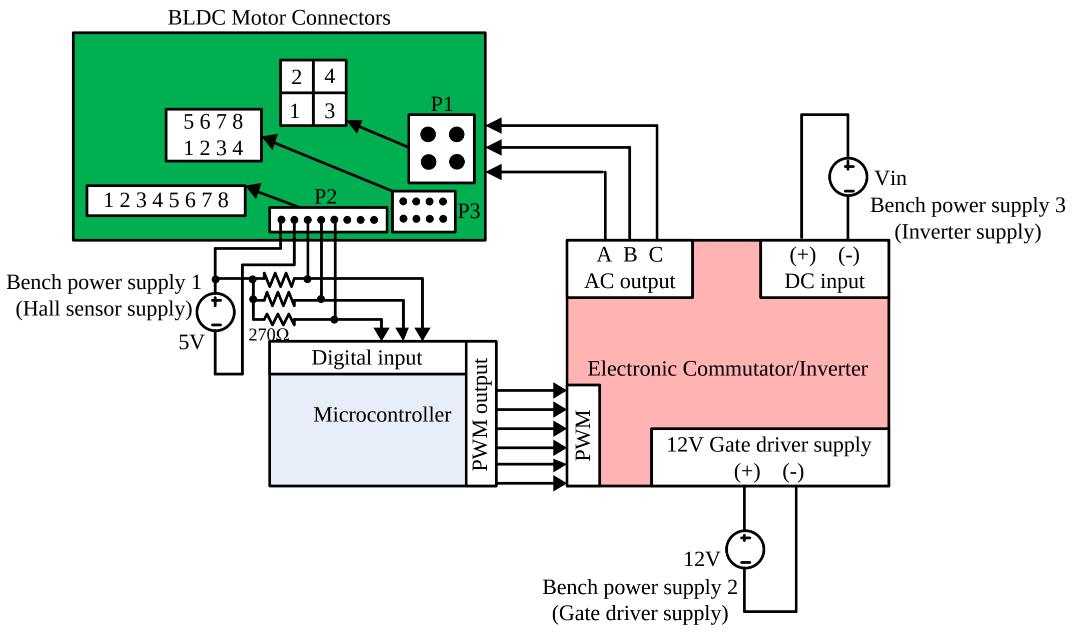


Figure 4. System connections for BLDC motor drive.

What is your observation on motor speed when the supply voltage is increased gradually? Please explain your observation.

4. Capture the waveforms of Hall sensor signals and complete the diagram in Figure 5.

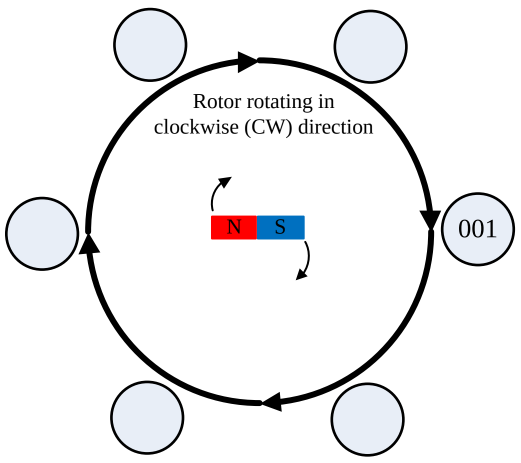


Figure 5. States of Hall sensor signals (H1 H2 H3) for CW direction.

---

4 Optional Additional Work

Marks will only be awarded for this section if you have already completed all of Section 3 to an excellent standard and with excellent understanding.

With the supply V_{in} fixed at 12V, improve your controller to achieve following feature:

- The rotating direction can be selected by a dip switch.

Note: When the motor is running, do not switch to the opposite direction. Motor should always start from stationary.

5 Appendix A: Operating Principles and Control of Three-Phase Brushless DC (BLDC) Motor

A BLDC motor is constructed with a permanent magnet rotor and wire wound stator poles. It is a type of synchronous motor. This means the magnetic field generated by the stator and the magnetic field generated by the rotor rotate at the same frequency. Electrical energy is converted to mechanical energy by the magnetic attractive forces between the permanent magnet rotor and a rotating magnetic field induced in the wound stator poles. Figure A1 shows the diagram of a BLDC motor. The current flowing in each phase winding induces flux and the sum of them determine the resultant stator flux vector.

Unlike a brushed DC motor, the commutation of a BLDC motor is controlled electronically. To rotate the BLDC motor, the stator windings should be energized in a sequence. The BLDC motor's electronic commutator sequentially energizes the stator coils generating a rotating electric field that 'drags' the rotor around with it. N "electrical revolutions" equates to one mechanical revolution, where N is the number of magnet pairs. It is important to know the rotor position in order to understand which winding will be energized following the energizing sequence. Rotor position is sensed using Hall sensors embedded into the stator. Most BLDC motors have three Hall sensors embedded into the stator on the non-driving end of the motor.

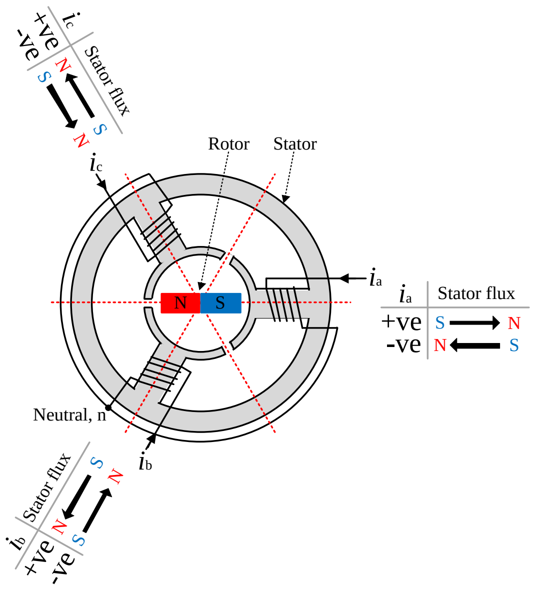
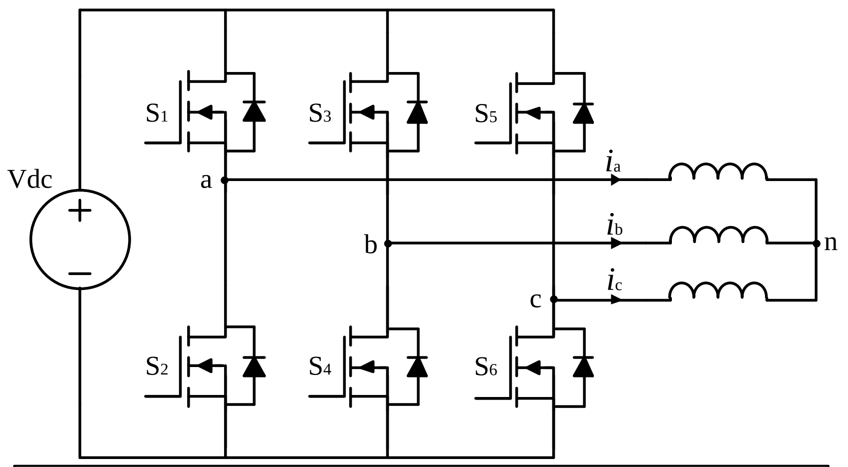


Figure A1. BLDC motor diagram.



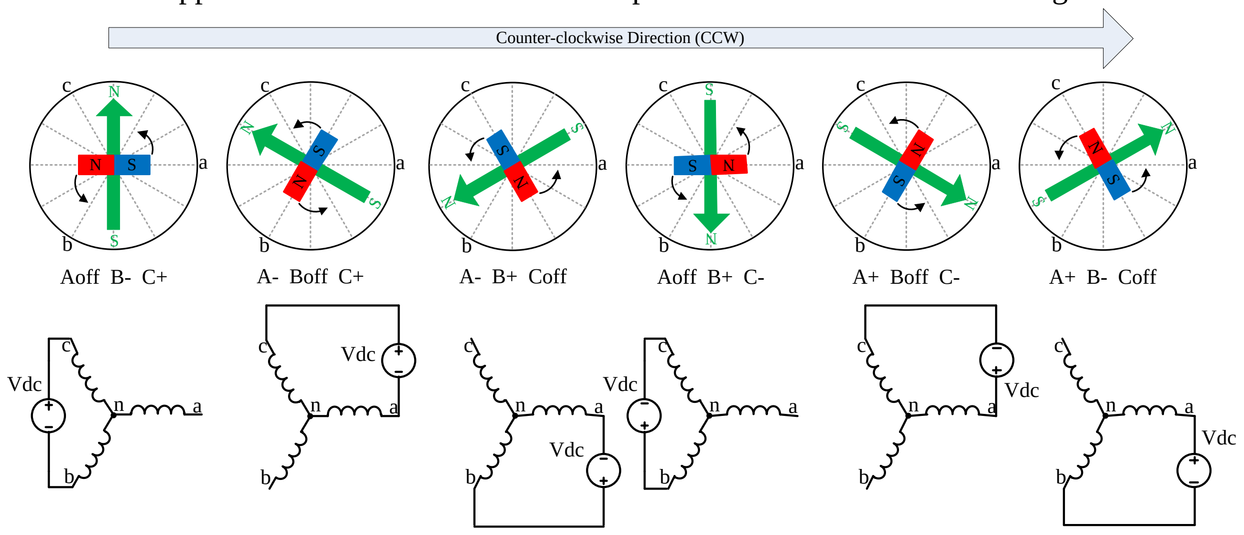
Note: For each phase, current flows in positive direction when upper switch is ON and negative direction when lower switch is ON. Short circuit (“shoot through”) when both upper and lower switches are ON together.

Figure A2. Three-phase inverter/electronic commutator.

The phase commutation depends on the Hall sensor values that indicate the rotor positions. When motor coils are correctly supplied, a magnetic field is created and the rotor moves. The most elementary commutation driving method used for BLDC motors is an on-off scheme: a coil is either conducting or not conducting. Only two windings are supplied at the same time and the third winding is floating. Connecting the coils to the power (+) and neutral (-) bus induces the current flow. This is referred to as trapezoidal commutation or block commutation. To command BLDC motors, a power stage made of 3 half bridges as shown in Figure A2 is used. A correct commutation pattern creates a rotating field.

There are six commutation sequences in each cycle, as shown in Figure A3, for counter-clockwise (CCW) direction. Each commutation sequence has one of the windings energized to positive power (“+”: current enters into the winding), the second winding is negative (“-”: current exits the winding) and the third is in a non-energized (“off”) condition. Torque is produced because of the interaction between the magnetic field generated by the stator coils and the permanent magnets. Ideally, the peak torque occurs when these two fields are at 90° to each other and falls off as the fields move together. In order to keep the motor running, the magnetic field produced by the windings should shift position, as the rotor moves to catch up with the stator field. What is known as “Six-Step Commutation” defines the sequence of energizing the windings.

Suppose that the rotor position is aligned with Phase A axis before the motor is energized as shown in the leftmost diagram. The controller should identify the rotor position through Hall sensor signals ($H1_1$ $H2_1$ $H3_1$). To produce maximum torque that attracts the rotor to spin in CCW direction, stator field pointing upward should be produced by connecting Phase C to the positive DC bus voltage, Phase B to the negativel DC bus voltage while Phase A is unpowered. This can be done by turning on S4 and S5 of the electronic commutator. The sum of the two flux vectors produced by Phase B and C lead to the resultant stator flux vector (green). Then, the rotor tries to follow the stator flux. As soon as the rotor reaches a given position, the state of Hall sensors changes its value from “$H1_1$ $H2_1$ $H3_1$” to “$H1_2$ $H2_2$ $H3_2$”. A new voltage pattern is selected and applied to the BLDC motor and this process continues as shown in Figure A3.



Note:
X+ means phase X is connected to positive terminal of input DC supply (upper switch ON)
X- means phase X is connected to negative terminal of input DC supply (lower switch ON)
Xoff means phase X is OFF (upper and lower switches OFF)

Figure A3. BLDC motor rotation sequence in counter-clockwise (CCW) direction.

For each rotor position, only one out of the six possible connections between three-phase motor terminals and electronic commutator outputs is correct for both direction of rotation while the remaining 5 are wrong. Hence, the commutation table which maps the states of Hall sensor signals (rotor position) to the states of electronic commutator is crucial. By referring to Figure A4 and performing experimental testing on BLDC motor, the commutation table can be constructed.

Step 1: Set the current limit of a DC power supply to 20% of the motor rated current.

Step 2: Connect motor winding terminals to positive or neutral of DC power supply e.g. for “A+ B- C-”, connect Phase A to positive terminal of DC power supply, Phase B and C to neutral of DC power supply then turn on the DC power supply.

Step 3: Once the rotor becomes stationary, measure the three Hall sensor signals H1, H2 and H3 e.g. “$H1_1$ $H2_1$ $H3_1$” for rotor aligned with Phase A axis (“A+ B- C-”).

Step 4: Turn off the DC power supply and repeat experiment (step 2-4) to determine all the 6 states of Hall sensor signals for 6 different rotor positions aligned with red stator flux vectors depicted in Figure A4.

Having determined the combinations of Hall sensor signals for each rotor position, the commutation table can be constructed by referring to blue vectors illustrating the stator fields produced by electronic commutator. The commutation table for CCW rotation is given in Table A1.

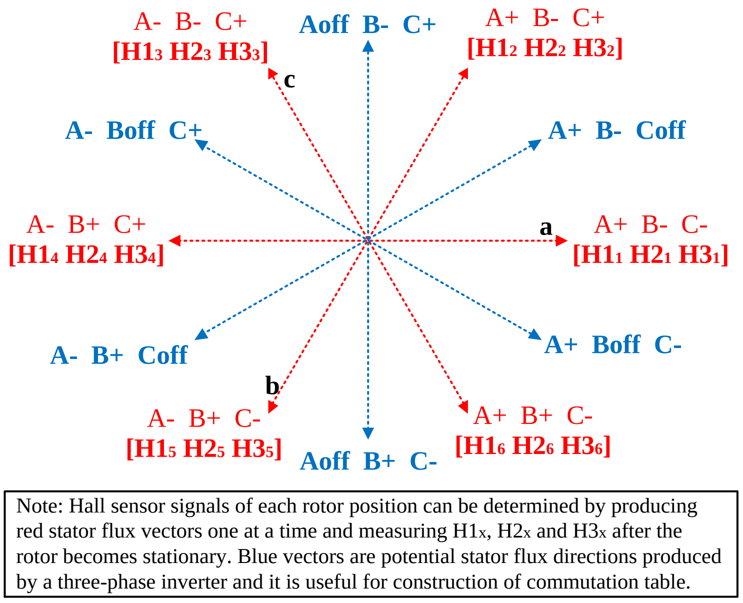

A- B- C+
[H1₃ H2₃ H3₃]
Aoff B- C+
A+ B- C+
[H1₂ H2₂ H3₂]
c
A- Boff C+
A+ B- Coff
A- B+ C+
[H1₄ H2₄ H3₄]
a
A+ B- C-
[H1₁ H2₁ H3₁]
A- B+ Coff
A+ Boff C-
b
A- B+ C-
[H1₅ H2₅ H3₅]
Aoff B+ C-
A+ B+ C-
[H1₆ H2₆ H3₆]
Note: Hall sensor signals of each rotor position can be determined by producing red stator flux vectors one at a time and measuring H1x, H2x and H3x after the rotor becomes stationary. Blue vectors are potential stator flux directions produced by a three-phase inverter and it is useful for construction of commutation table.

Figure A4. BLDC motor stator flux vectors.

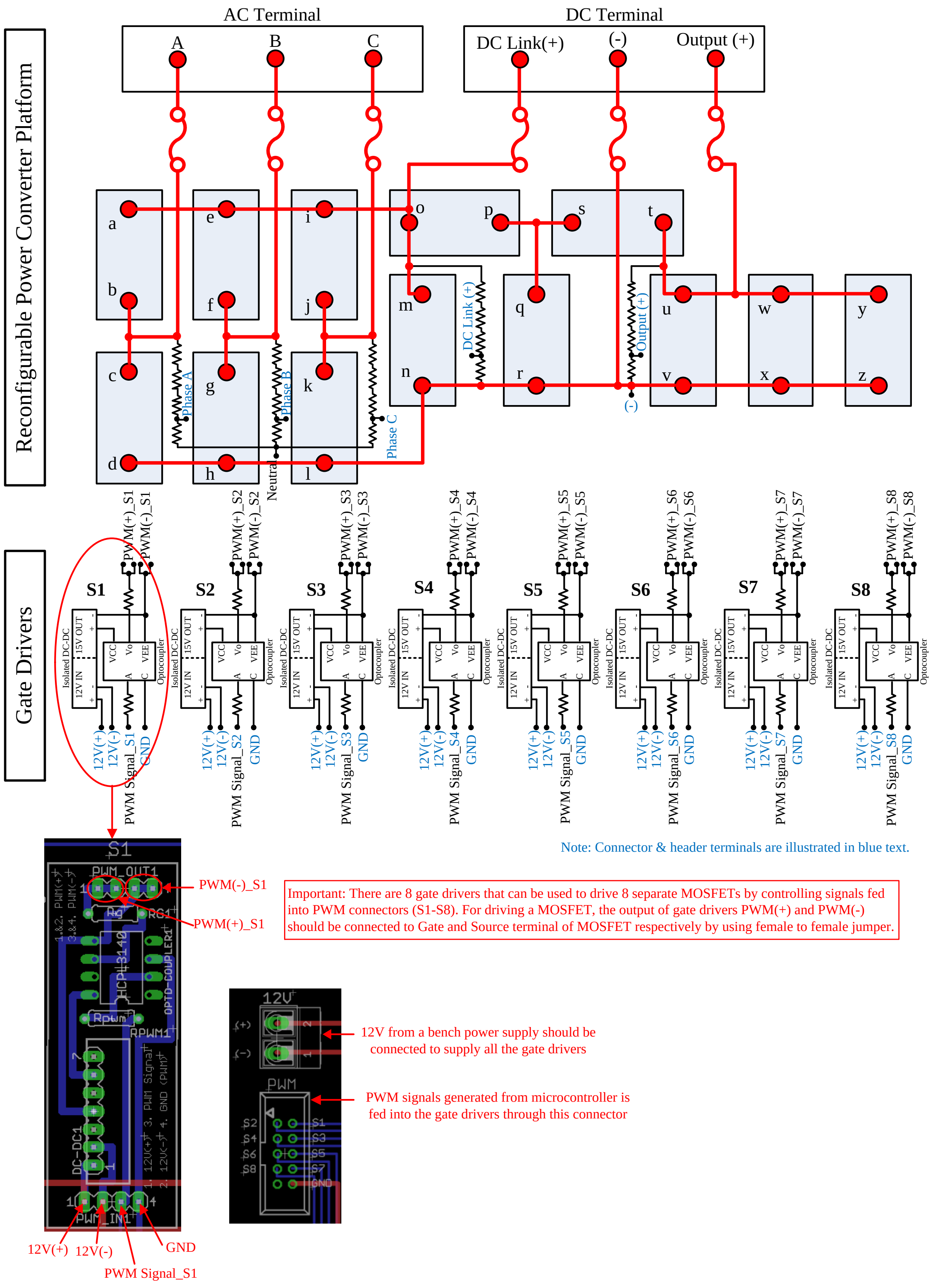
Table A1. Commutation for counter-clockwise (CCW) direction

Hall Sensor Signal			**Phase**		
H1	**H2**	**H3**	**A**	**B**	**C**
$H1_1$	$H2_1$	$H3_1$	off	-	+
$H1_2$	$H2_2$	$H3_2$	-	off	+
$H1_3$	$H2_3$	$H3_3$	-	+	off
$H1_4$	$H2_4$	$H3_4$	off	+	-
$H1_5$	$H2_5$	$H3_5$	+	off	-
$H1_6$	$H2_6$	$H3_6$	+	-	off

References for further reading:

1. W. Brown "Brushless DC Motor Control Made Easy," Microchip Application Note **AN857**, 2002.
2. P. Yedamale "Brushless DC (BLDC) Motor Fundamentals," *Microchip Application Note **AN885***, 2003.
3. P. Yedamale "Brushless DC Motor Control Using PIC18FXX31 MCUs," *Microchip Application Note **AN899***, 2004.
4. S. D'Souza "Sensored BLDC Motor Control Using dsPIC30F2010," *Microchip Application Note **AN957***, 2004.
5. L. N. Elevich "3-Phase BLDC Motor Control with Hall Sensors Using 56800/E Digital Signal Controllers," Freescale Semiconductor Application Note **AN1916**, 2005.
6. –"Hall Effect Sensor," http://www.electronics-tutorials.ws/electromagnetism/hall-effect.html
7. BLDC Motor Datasheet: http://www.hurst-motors.com/documents/Dynamo_BLDC.pdf

6 Appendix B: Experimental Platform



Note: Connector & header terminals are illustrated in blue text.

Important: There are 8 gate drivers that can be used to drive 8 separate MOSFETs by controlling signals fed into PWM connectors (S1-S8). For driving a MOSFET, the output of gate drivers PWM(+) and PWM(-) should be connected to Gate and Source terminal of MOSFET respectively by using female to female jumper.

Figure B1. Schematic of the experimental platform.

7 Appendix C: Motor Connections

BLDC Motor Connectors

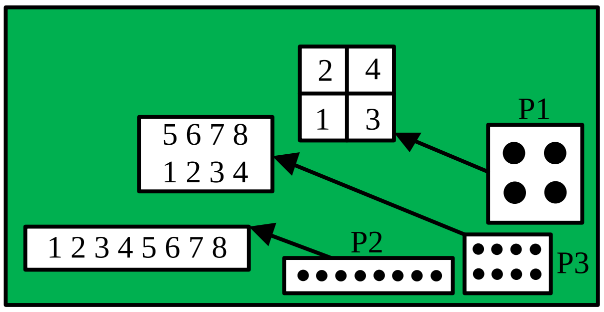


Figure C1. BLDC motor connectors.

Table C1. Description of BLDC motor connectors.

Power Connections (P1)		
Pin#	Description	Wire Color
1	Phase C	Red
2	Phase B	Black
3	Phase A	White
4	Frame GND	Green
Hall Sensor Connections (P2)		
1	Vs (+ve terminal)	Red
2	Vs (-ve/return terminal)	Black
3	Hall B (H2)	Chocolate
4	Hall A (H1)	White
5	Hall C (H3)	Green
6	Blank	Blank
7	Blank	Blank
8	Blank	Blank
Encoder Connections (P3)		
1	+5V (+ve terminal)	Red
2	Encoder A	White
3	Encoder B	Blue
4	Encoder I	Gray
5	+5V (return terminal)	Black
6	Encoder /A	Orange
7	Encoder /B	Chocolate
8	Encoder /I	Green

8 Appendix D: Code for generating PWM signals

```
/*
 *  Simple program to generate six 50Hz PWM signals with 50% duty cycle and ~1uS
deadtime
 *
 *  PORTB is used as the main output for PWM. PIN mapping is listed below
 *
 *  ilmatto          BLDC Testbed
 *  PIN A0     --       S1
 *  PIN A1     --       S2
 *  PIN A2     --       S3
 *  PIN A3     --       S4
 *  PIN A4     --       S5
 *  PIN A5     --       S6
 */

#ifndef TIMSK1
#define TIMSK1 TIMSK1
#endif

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define F_PWM 50                  // Set the PWM frequency to 50Hz
#define DEAD_TIME 0.001           // Set the deadtime for mosfets to 1uS

#define DRIVE_PORT PORTA          // Port definition
#define DRIVE_DDR DDRA            // Port direction
#define PRESCALER 8               // Timer prescaler
#define F_CPU 12000000UL          // MCU running frequency

#define toggleS1 PINA |=_BV(0)
#define toggleS2 PINA |=_BV(1)
#define toggleS3 PINA |=_BV(2)
#define toggleS4 PINA |=_BV(3)
#define toggleS5 PINA |=_BV(4)
#define toggleS6 PINA |=_BV(5)

int x, n, d1, d2, d3;

void initialize();
void setFrequency();
void enableInterrupts();

/* Toggles the phases according to Figure 1 */
ISR(TIMER1_COMPA_vect){

    n = x%3;

    if(n==0){
        if(d1==1){
            toggleS1;
            _delay_ms(DEAD_TIME);
            toggleS2;
            d1 = 0;
        }

        else{
            toggleS2;
            _delay_ms(DEAD_TIME);
            toggleS1;
            d1 = 1;
        }

        x = 1;
    }


    if(n==2){
        if(d2==1){
            toggleS3;
            _delay_ms(DEAD_TIME);
            toggleS4;
            d2 = 0;
        }

        else{
            toggleS4;
            _delay_ms(DEAD_TIME);
            toggleS3;
            d2 = 1;
        }
        x = 3;
    }


    if(n==1){
        if(d3==1){
            toggleS6;
            _delay_ms(DEAD_TIME);
            toggleS5;
            d3 = 0;
        }

        else{
            toggleS5;
            _delay_ms(DEAD_TIME);
            toggleS6;
            d3 = 1;
        }
        x = 2;
    }
}

int main(){

    d1 = 1; d2 = 1; d3 = 1; x = 0;

    // Configure the ports
    initialize();

    // Set the PWM frequency
    setFrequency();

    //Enable interrupts
    enableInterrupts();

    // call UART - use only for testing
    // init_debug_uart0();

    while(1){
    // ISR handles everything
    }

return(0);
}

/* port and timer configuration */
void initialize(){

    // set ports B to output
    DRIVE_DDR = 0xFF;

    // configure the timer
    TCCR1A = _BV( COM1A0 ) | _BV(WGM10);
    TCCR1B |=_BV(WGM13)|_BV(CS11);

    // set initial PWM output sequence
    DRIVE_PORT |= _BV(0) | _BV(2) | _BV(5);
    DRIVE_PORT &= ~_BV(1) & ~_BV(3) & ~_BV(4);
}
```

```
/* calculate the OCR1A value according to frequency */
void setFrequency(){

    // OCR1A value = 12000000/2*2*8/50*3 for 50Hz frequency
    OCR1A = (uint16_t)(F_CPU/(2*2*PRESCALER)/(F_PWM*3));
}

/* interrupt enable */
void enableInterrupts(){

    //enable interrupt on compare match
    TIMSK1 |= _BV(OCIE1A);

    //enable global interrupts
    sei();
}
```

9 Appendix E: Skeleton code for driving BLDC motor, CCW rotation

NOTE: Phase commutation section needs to be completed in order to drive the motor

```
/*  Simple program for BLDC motor control.
 *
 *  Students are required to finish the phase commutation section in order to spin the
BLDC drive
 *
 *  Pin connections listed below:
 *
 *  ilmatto          BLDC Testbed
 *  PIN A0     --       S1
 *  PIN A1     --       S2
 *  PIN A2     --       S3
 *  PIN A3     --       S4
 *  PIN A4     --       S5
 *  PIN A5     --       S6
 *
 *  PIN C0     --       Hall A
 *  PIN C1     --       Hall B
 *  PIN A7     --       Hall C
 *
 */

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
//#include "debug.h"

// switch configuration
#define S1H PORTA |=  _BV(0)
#define S1L PORTA &= ~_BV(0)
#define S2H PORTA |=  _BV(1)
#define S2L PORTA &= ~_BV(1)
#define S3H PORTA |=  _BV(2)
#define S3L PORTA &= ~_BV(2)
#define S4H PORTA |=  _BV(3)
#define S4L PORTA &= ~_BV(3)
#define S5H PORTA |=  _BV(4)
#define S5L PORTA &= ~_BV(4)
#define S6H PORTA |=  _BV(5)
#define S6L PORTA &= ~_BV(5)

void initialize();
int readHall();

int hallA, hallB, hallC, phaseA = 0, phaseB = 0, phaseC = 0;

int main(){

    /* set up ports */
    initialize();

    while(1){

        /* read hall sensor values */
        readHall();

        /***************************  PHASE COMMUTATION  ************************/

        /* Compare the read hall values with all possible ones and set the correct
phase output.
         *
         * -1 means phase is set to low
         *  0 means phase is off
         *  1 means phase is set to high
         *
         * Phase commutation is given for the first two hall sensor values 001 and 011
(for CCW rotation)
         *
```

```
          *  hallA   hallB   hallC    phaseA     phaseB      phaseC
          *    0       0       1         0         -1           1
          *    0       1       1        -1          0           1
          *   ...     ...
          */


        // hall sensor ABC value = 001
        if ((hallA == 0) && (hallB == 0) && (hallC != 0)){

            phaseA =  0;
            phaseB = -1;
            phaseC =  1;
        }

        // hall sensor ABC value = 011
        else if ((hallA == 0) && (hallB != 0) && (hallC != 0)){

            phaseA = -1;
            phaseB =  0;
            phaseC =  1;
        }

        /*  Student TODO list:
         *
         *  1)  Complete the phase commutation table for the other 4 hall sensor
values (CCW rotation)
         *      to drive the BLDC motor. Test your code.
         *
         *  2)  Adjust the code to include CW rotation. Do not forget to include a
small delay when
         *      changing the direction of rotation while BLDC is operational.
Otherwise it might damage
         *      the motor.
         */




/*************************************************************************************/


        /* Phase A configuration */
        switch (phaseA){

                // Phase A set to High
                case 1:
                    S1H;
                    S2L;
                break;

                // Phase A disabled
                case 0:
                    S1L;
                    S2L;
                break;

                // Phase A set to Low
                case -1:
                    S1L;
                    S2H;
                break;
        }

        /* Phase B configuration */
        switch (phaseB){

                // Phase B set to High
                case 1:
                    S3H;
                    S4L;
                break;

                // Phase B disabled
                case 0:
                    S3L;
                    S4L;
                break;

                // Phase B set to Low
                case -1:
                    S3L;
                    S4H;
                break;
        }

        /* Phase C configuration */
        switch (phaseC){

                // Phase C set to High
                case 1:
                    S5H;
                    S6L;
                break;

                // Phase C disabled
                case 0:
                    S5L;
                    S6L;
                break;

                // Phase C set to Low
                case -1:
                    S5L;
                    S6H;
                break;
        }
    }
return(0);
}


/****** Port configuration ******/
void initialize(){

    /* Set PORTA0-5 to output (for BLDC drive) */

    DDRA |= _BV(PA0);       // S1
    DDRA |= _BV(PA1);       // S2
    DDRA |= _BV(PA2);       // S3
    DDRA |= _BV(PA3);       // S4
    DDRA |= _BV(PA4);       // S5
    DDRA |= _BV(PA5);       // S6

    /* Set PC0, PC1, PA7 for hall sensor input*/

    DDRC &= ~_BV(PC0);      // hall sensor A input
    PORTC |= _BV(PC0);      // Enable pull-up on PC0 (might not be needed if external
pull-up is used)

    DDRC &= ~_BV(PC1);      // hall sensor B input
    PORTC |= _BV(PC1);      // Enable pull-up on PC1 (might not be needed if external
pull-up is used)

    DDRA &= ~_BV(PA7);      // hall sensor C input
    PORTA |= _BV(PA7);      // Enable pull-up on PA7 (might not be needed if external
pull-up is used)

    //init_debug_uart0();         // initialize UART for testing, requires debug.h
header file to work

}
```

```
/****** Sample the hall sensor ******/
int readHall(){

    hallA   = (PINC & _BV(PC0));
    hallB   = (PINC & _BV(PC1));
    hallC   = (PINA & _BV(PA7));

    return (hallA & hallB & hallC);
}
```