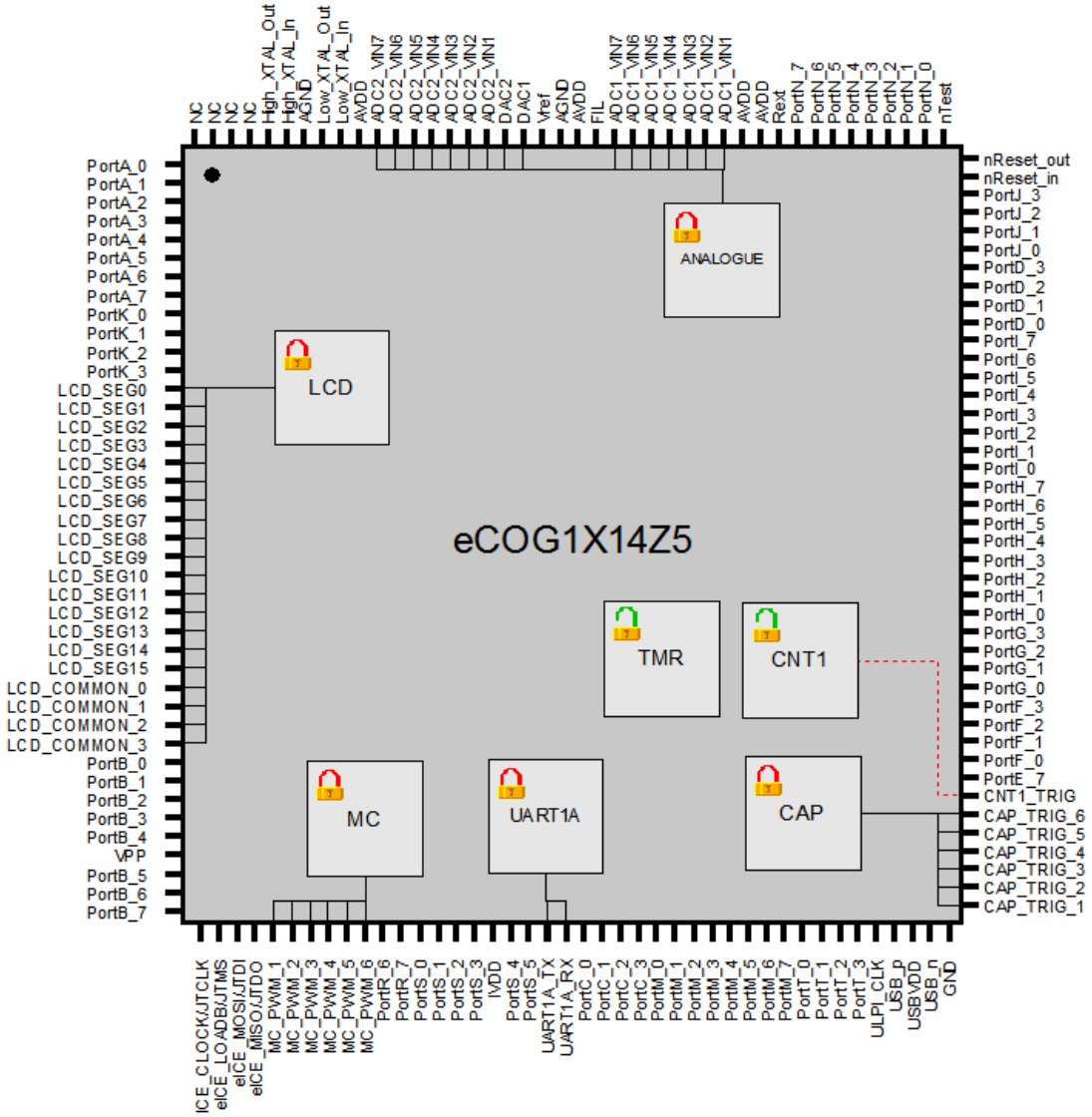
AN061 – eCOG1X 3-Phase BLDC Motor Control with Sensor Feedback

Version 1.0

This application note describes using the MCPWM peripheral in the eCOG1X to control a 3-phase brushless DC motor using a trapezoidal commutation scheme and Hall effect position sensors.



Confidential and Proprietary Information

Revision History

Version	Date	Notes
V1.0	16/08/2007	First release.

Contents

1 Introduction

This application note describes using the MCPWM peripheral in the eCOG1X to control a 3-phase brushless DC (BLDC) motor, using a trapezoidal commutation scheme and three Hall effect position sensors. An external motor control power module provides high current switching for the motor windings.

The eCOG1X features a high performance 16-bit processor core coupled with a flexible multi-channel PWM timer block and a rich set of peripherals, making the microcontroller suitable for motor control applications.

2 Glossary

A table of abbreviations used in this document.

ADC	Analogue to Digital Converter
Back-EMF	Back-Electromotive Force
BLDC	Brushless DC
CPU	Central Processing Unit
eCOG1X	Cyan Technology target microcontroller
GPIO	General Purpose Input/Output
ISR	Interrupt Service Routine
LCD	Liquid Crystal Display
MCPWM	Multi-Channel Pulse-Width Modulation
PID	Proportional-Integral-Derivative
PWM	Pulse-Width Modulation
RPM	Revolutions per Minute
SPM	Smart Power Module

3 Electric Motor

There is a wide range of different types of motors. The choice of a specific motor type for a particular application generally is determined by performance and cost. Of these, the brushless DC (BLDC) motor and the permanent magnet synchronous motor (PMSM) are gaining widespread use in various consumer and industrial applications. Both the BLDC motor and the PMSM feature high efficiency and good controllability due to their linear speed/torque characteristics, giving predictable speed regulation.

Emphasis is placed on these two popular motor types in demonstrating the motor control capability of the eCOG1X. Figure 1 indicates how they are related to other types of motor.

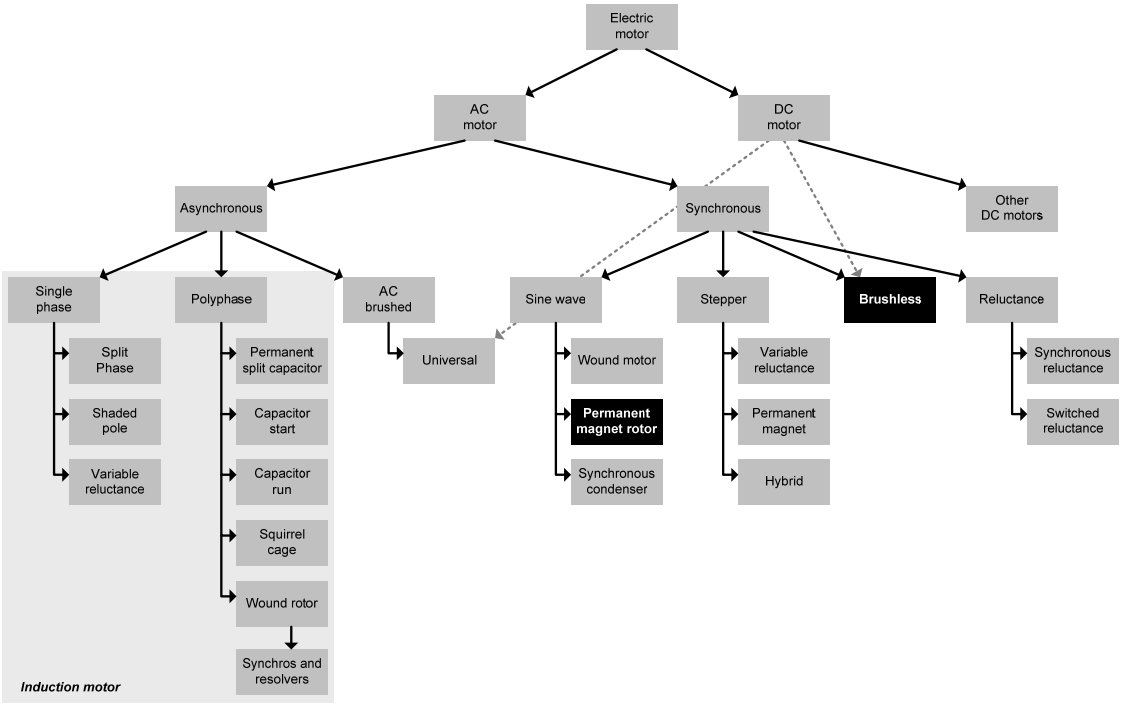


Figure 1. Different types of electric motors.

3.1 Permanent Magnet Motor Technology

The permanent magnet motor has two primary parts; the non-moving part is called the stator and the moving part, typically inside the stator, is called the rotor. Figure 2 illustrates these components for a 3-phase synchronous motor with having a rotor with one permanent magnet pole pair.

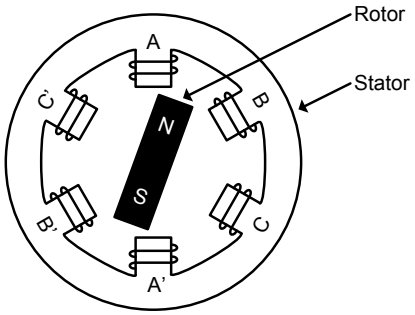


Figure 2. 3-phase synchronous motor.

In order to enable a motor to rotate, two magnetic fluxes are required, one from the stator and the other from the rotor. For this process, several motor configurations are possible. Normally the stator has a number of coil windings, driven by an electrical supply, operating as a set of electromagnets to generate the required flux. The rotor magnetic flux may be generated from windings or from permanent magnets providing a steady-state magnetic field. The latter is preferred due to the simpler construction involved.

Two common configurations of permanent magnet motor drives are widely used, described as trapezoidal and sinusoidal commutation, according to the shape of the current waveforms applied to the stator coils. Different control strategies and control hardware are implemented depending on the configuration.

All brushless motor control systems require information about the position of the rotor relative to the stator. Often a separate position sensor attached to the rotor or motor shaft is used, such as a resolver or a set of Hall effect sensors. The control system uses the measured shaft position to determine the currents required in each stator winding to achieve the motor torque or speed. The currents are controlled such that the combination of all the stator windings generates a rotating magnetic field, and the rotor then moves to align its fixed magnetic field with this rotating field. As the shaft rotates, so the control system adjusts the currents in the stator windings to keep the rotating magnetic field ahead of the rotor position, generating a torque.

When a permanent magnet motor rotates, the interaction of the rotor magnetic field with the stator coils generates a voltage on the stator coils (back-EMF) that opposes the input voltage supplied to the windings. Some control systems use this back-EMF voltage to infer the position of the rotor without using any additional position sensors, in so-called sensorless feedback.

3.2 BLDC Motor Specifics

A BLDC motor is a rotating electric machine where the stator is a classic 3-phase wound stator, like that of an induction motor, and the rotor has surface-mounted permanent magnets. When the wound stator is energised by a 3-phase alternating current, it creates a rotating magnetic flux that causes the rotor to rotate synchronously with it.

In a BLDC motor, the position of the rotor (and hence its permanent magnetic field) is sensed with respect to the stator coils (phases), and the supply current is switched electronically (commutated) to the appropriate phases. BLDC motor control systems often incorporate either internal or external position sensors to sense the actual rotor position. Alternatively, the rotor position can be detected without sensors by measuring the back-EMF in each stator winding.

The BLDC motor is driven by trapezoidal currents coupled with the given rotor position.

3.3 PMSM Specifics

Similar to the BLDC motor, the rotor of a PMSM consists of permanent magnets. The stator of a PMSM has its 3-phase windings distributed sinusoidally, as opposed to the trapezoidal salient-pole distribution found in a BLDC motor. A PMSM operates in the same way as a BLDC motor, when the wound stator is energised by a 3-phase alternating current, it creates a rotating magnetic flux that causes the rotor to rotate synchronously with it.

The PMSM is driven by sinusoidal currents coupled with the given rotor position.

3.4 Differences between BLDC Motor and PMSM

The term “BLDC” is a marketing driven label to promote the idea that a BLDC motor with the appropriate drive is a suitable drop-in replacement for a brushed DC motor and its drive. Technically, a BLDC motor is still an AC motor; that is, AC inputs are required in order for the motor to move.

BLDC motors usually are trapezoidally commutated (also known as six-step commutation for a 3-phase motor), using Hall effect position sensors for simplicity. The motors are typically trapezoidally wound; when driven, the back-EMF profile is also roughly trapezoidal. The generated torque is almost constant, but it has significant torque ripple which occurs at each step of the six-step commutation sequence.

In contrast, PMSM employs sinusoidal commutation typically working from a high-resolution position feedback sensor such as a resolver, and tends to be better controlled with a more sophisticated sinusoidal commutation algorithm. PMSM has sinusoidally distributed windings, which then create a sinusoidal back-EMF when the rotor spins. The drive signals to the stator windings are controlled to generate current waveforms that are sinusoidal. The resulting torque is smoother with less ripple than a similar BLDC motor, although the peak torque available from a PMSM is lower.

Both BLDC motor and PMSM are quite similar, with the shape of the optimal input stator currents being the main difference. A BLDC motor can be driven by either trapezoidal or sinusoidal currents, although trapezoidal currents provide a better match for the stator windings. The sinusoidal excitation of the stator windings in a PMSM generates a much smoother resultant torque than that of a BLDC motor.

A summary of the differences between both types of motor is given in Table 1.

Parameter	3-Phase BLDC Motor	3-Phase PMSM
Windings distribution	Trapezoidal	Sinusoidal
Energised phase	Two phases on at any time	Three phases on at any time
Stator current	Square/trapezoidal wave	Sinusoidal wave
Back-EMF	Trapezoidal wave	Sinusoidal wave
Electromagnetic torque	Almost constant	Constant
Torque strength	Stronger	Weaker

Table 1. Comparison of BLDC motor and PMSM.

4 Motor Control

4.1 Overview

Both 3-phase BLDC motor and PMSM share a similar motor drive topology, as shown in Figure 3. On the input side, the diode bridge rectifies the AC line voltage; this rectified voltage is then filtered by the input capacitor, forming a simple AC to DC converter.

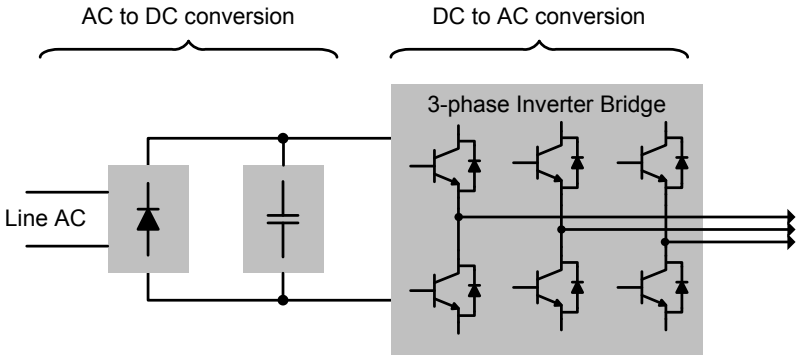


Figure 3. Motor drive topology.

Conversely, the 3-phase inverter bridge performs DC to AC conversion. One pair of transistors in this bridge circuit is connected to each motor phase. The high-side transistor is used to apply positive voltage to a motor phase, while the low-side transistor in turn applies negative voltage to a motor phase. Digital drives directly control the switching of all six transistors using pulse width modulation (PWM) techniques, enabling current to be directed into or out of any combination of the three motor phases.

An overview of the major component blocks comprising a typical motor control system is shown in Figure 4.

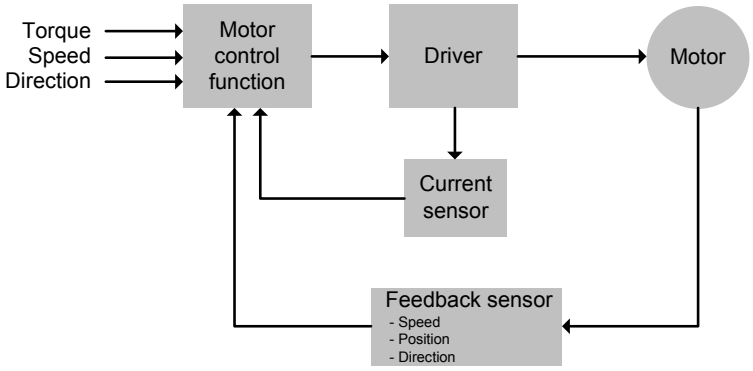


Figure 4. Motor control system block diagram.

Torque generation in a BLDC motor and PMSM is a function of current. The motor control function measures (either directly or indirectly) the amount of current flowing and adjusts the PWM switch on time or duty cycle to maintain the required average level of current flow. The average level of current flow can be considered as the torque producing current.

4.2 Feedback Sensors

In most motor control systems, several sensors are used to provide feedback information on the motor. These sensors are used in the control loop and to improve the reliability by detecting fault conditions that may damage the motor. Figure 5 lists some sensors that can be used to feed back information to the microcontroller that is executing the motor control function.

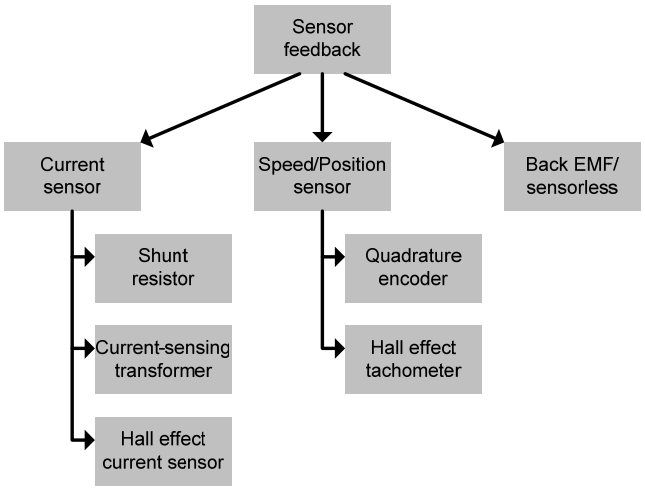


Figure 5. Motor control feedback options.

4.3 Sensor-Based Commutation

The absolute position of the rotor must be known in order that the controller can perform correct commutation of the motor. This position information can be derived from the three Hall effect sensors distributed along the stator. As the rotor completes one electrical cycle, the three Hall effect sensors generate six different signal combinations or states.

The relationship between the stator winding energising sequence, the commutation sequence for the BLDC motor and PMSM, and the Hall effect position sensor signal states is illustrated in Figure 6 and Figure 7.

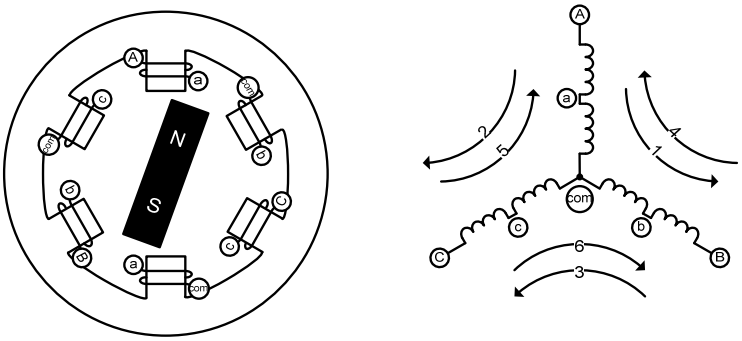


Figure 6. Winding energising sequence.

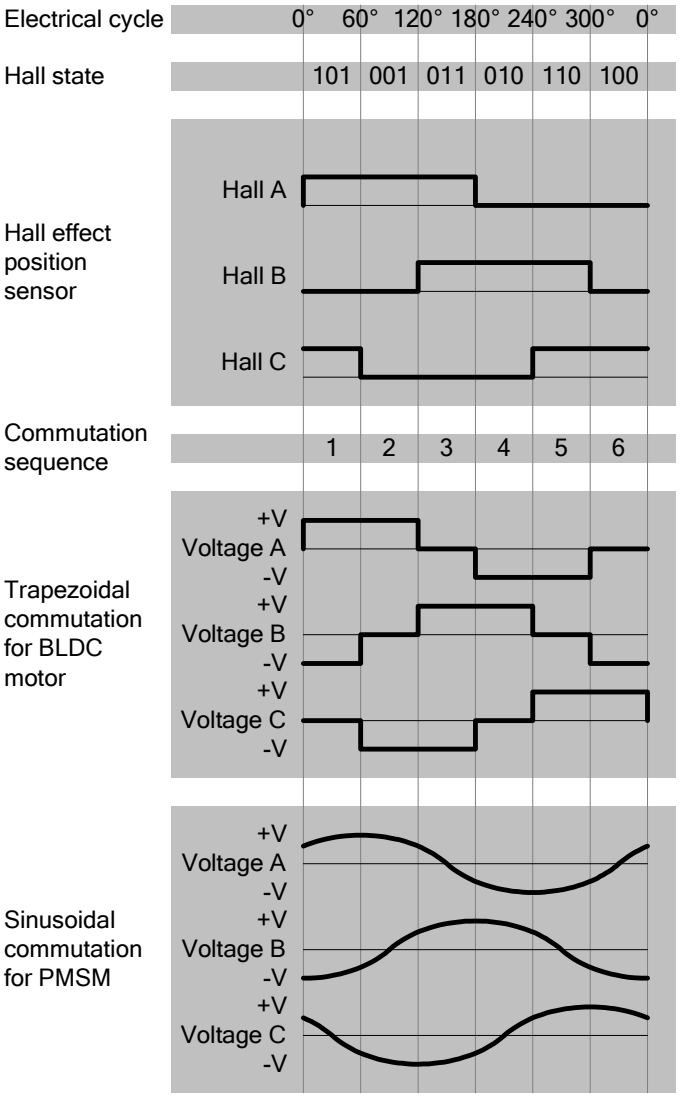


Figure 7. Sensor commutation sequence.

5 3-Phase BLDC Motor Control with Hall Sensors

The eCOG1X is a peripheral-rich, low power 16-bit microcontroller that naturally lends itself to embedded motor control applications. It hosts a number of motor control specific peripherals that aid and simplify the motor control application software executed by the microcontroller processor core.

These peripherals are as summarised in Figure 8.

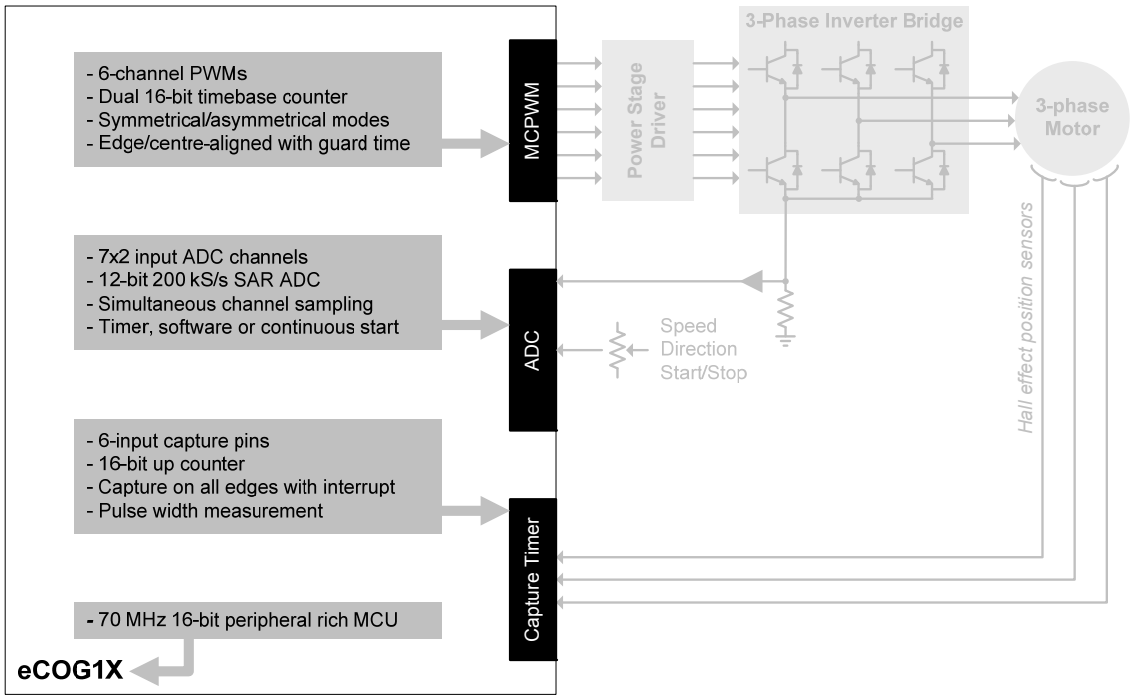


Figure 8. eCOG1X motor control specific peripherals.

The six-channel MCPWM digital outputs can control directly the switching transistors of the power driver stage, which in turn applies the appropriate phase signals to the motor phases via the inverter bridge. Full four-quadrant drive operation is possible, providing acceleration and deceleration torque with the motor running in either direction.

In applications where a potentiometer is used to input the speed demand and motor running direction, this can be monitored by one of the eCOG1X ADC channels. Another ADC channel can be used to observe the total stator current and check for overcurrent fault conditions.

Information on the rotor position and speed feedback can be obtained by capturing changes in the motor Hall effect sensor outputs using the eCOG1X input capture timer.

6 System Implementation

A system overview for a complete 3-phase BLDC motor control implementation, including the main software blocks, is shown in Figure 9.

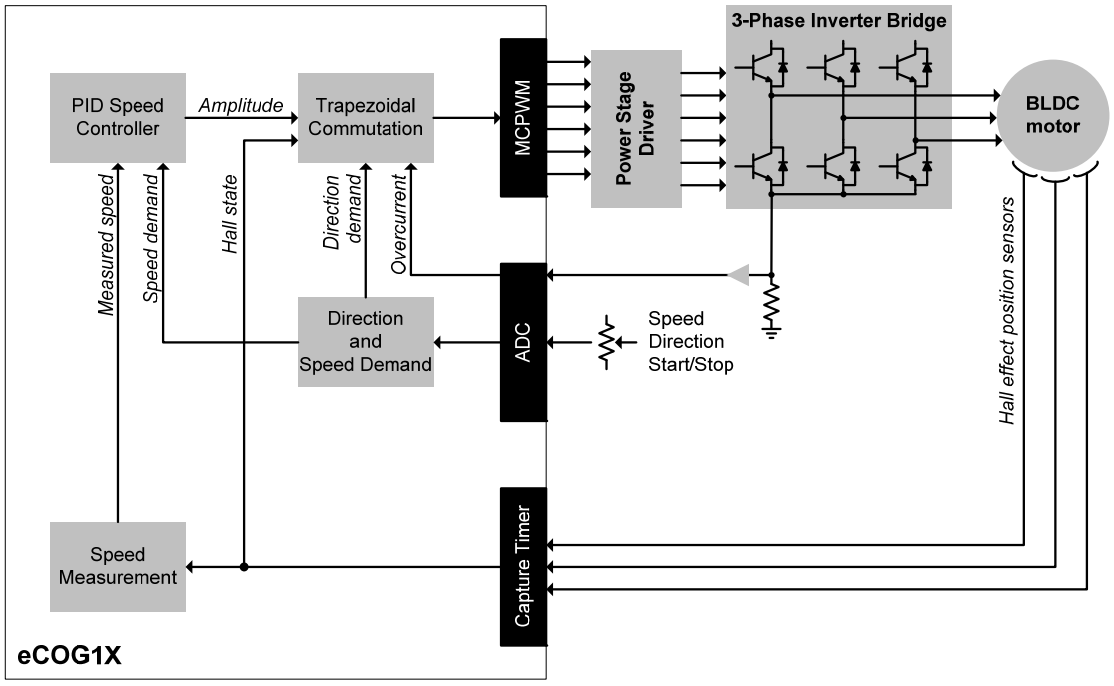
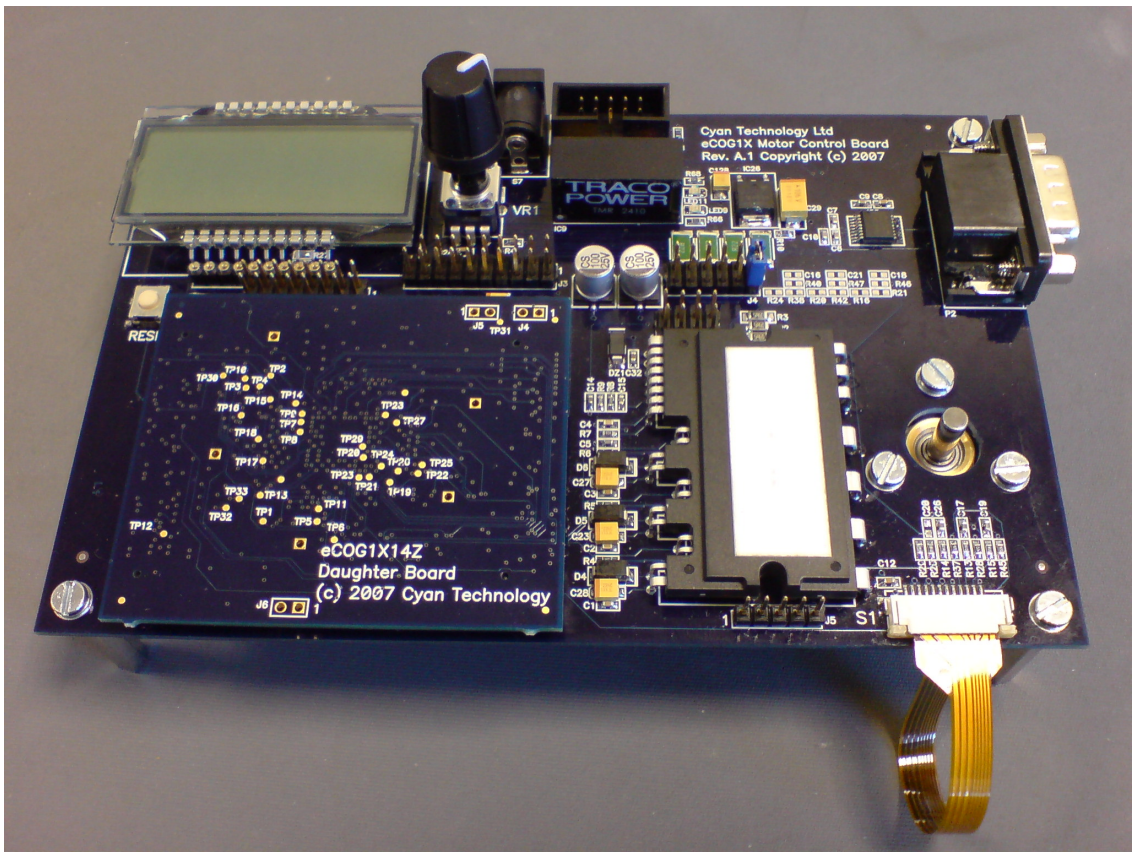


Figure 9. Overview of 3-phase BLDC motor control system with Hall sensors.

6.1 Hardware Implementation

An eCOG1X14Z5 based motor control demonstration board has been designed and built to demonstrate the motor control capability of the microcontroller. The circuit diagram for this motor control board is given in Appendix B.



In order to simplify the motor control board design, a Smart Power Module (SPM) is used. The SPM integrates all the discrete components of the power stage driver and 3-phase inverter bridge into a single compact module. The device chosen for this example is a Fairchild Semiconductor FCBS0650, a 500V 6A rated 3-phase MOSFET inverter bridge with integrated controllers for gate driving and protection. The FCBS0650 SPM typically is used in home appliance applications such as refrigerators.

The motor used for this example is the Maxon 267121 EC 32 15W BLDC motor. This is a 3-phase motor with a four pole pair rotor and three Hall effect position sensors. The Maxon EC 32 has a nominal voltage of 24V and a maximum no load speed of 4390rpm.

A single potentiometer is provided for the user to set the required motor speed and direction. The centre position of the potentiometer corresponds to zero speed, allowing both forward and reverse operation.

The eCOG1X ADC input channels are used to monitor the potentiometer input, the stator current, and to observe the individual stator phase voltages and currents applied to the motor. This flexibility provides support for future development of other motor control schemes.

The motor running speed and any error messages are displayed via the on-board LCD, while additional debug information is output through the RS-232 serial port interface.

6.2 Software Implementation

The core application software essentially involves obtaining the required speed and measured speed values, which are then passed to the speed control loop. The speed control loop in turn adjusts the PWM switching to reduce the difference between the required and measured speeds.

The details of the system implementation are somewhat more complex, as shown in the flow diagram of Figure 10.

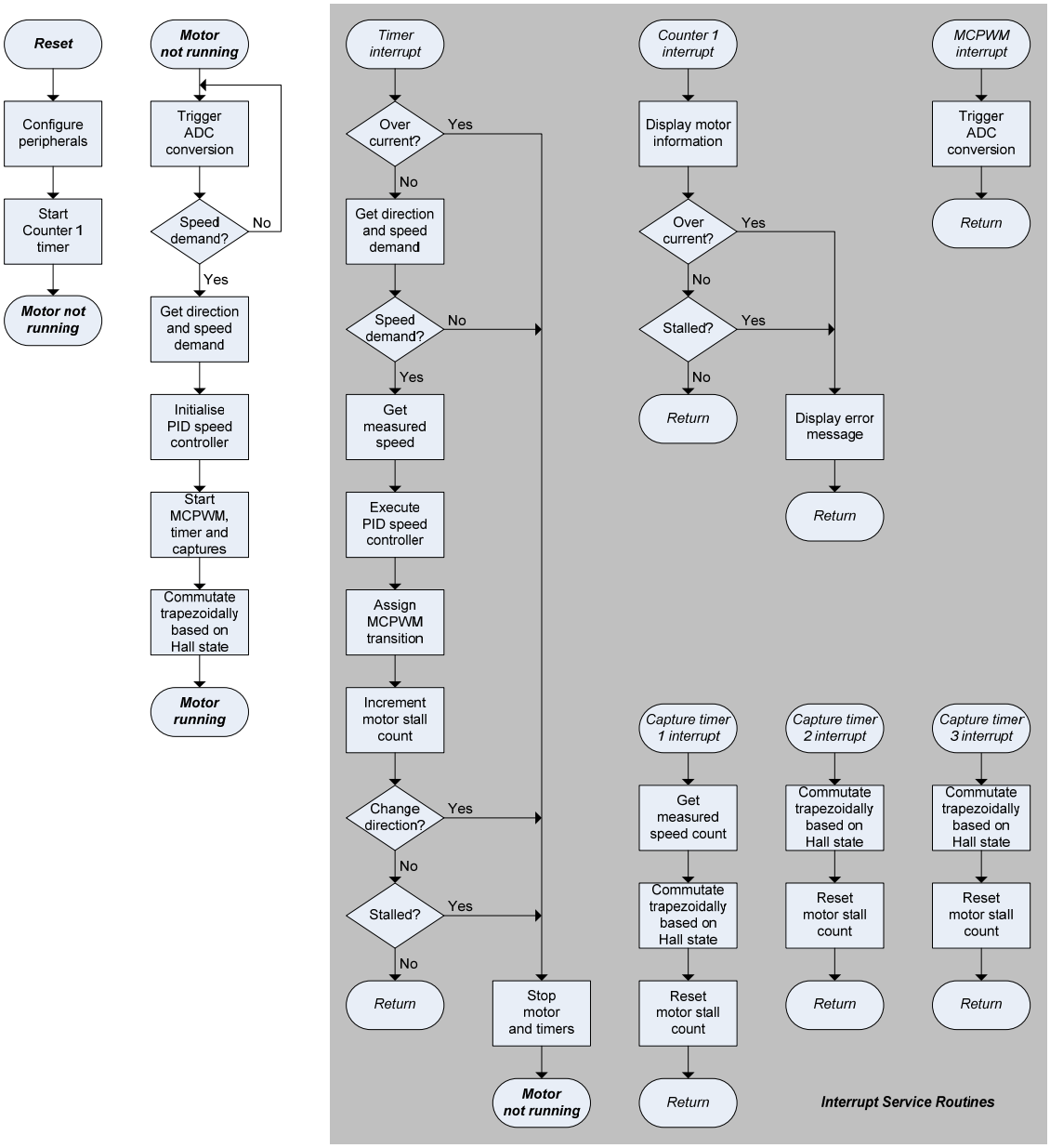


Figure 10. Motor control software flow diagram.

Following a *reset*, the necessary peripherals are set up and the software flow terminates at the *Motor not running* state. In this state, the ADC is continuously polled to check for any speed demand via the potentiometer. The input speed needs to exceed a set threshold for the speed demand to be deemed valid. A valid speed demand initialises the PID speed controller and starts the MCPWM and relevant timers. The motor then starts running at this stage.

In the *Motor running* state, all activities are interrupt driven. These interrupts can be grouped into four event types, according to their frequency and the function performed. Table 2 provides a summary of these events.

Event type	Interrupt source	Executed function
Fast event	MCPWM interrupt ▪ Triggers every 50us (20kHz)	▪ Starts ADC conversion
Medium event	Timer interrupt ▪ Triggers every 10ms (100Hz)	▪ Obtains direction and speed demand ▪ Calculates motor running speed ▪ Executes PID speed control ▪ Checks for motor overcurrent and stall condition
Capture event	Capture timers 1-3 ▪ Triggers on any change on Hall sensors A-C	▪ Commutates trapezoidally based on Hall state ▪ Captures transition time on Hall sensor A
Slow event	Counter 1 ▪ Triggers every 1s (1Hz)	▪ Display motor information on LCD ▪ Output debug information to RS-232 interface

Table 2. Interrupt driven events.

When there is no valid speed demand or a fault condition is detected, the motor is stopped and the software flow returns to the *Motor not running* state.

6.2.1 Motor Speed Measurement

The actual motor speed is determined from the elapsed time between transitions on one of the Hall effect position sensors. This transition is monitored by an input capture timer, with two counts for the two Hall sensor transitions that occur every electrical cycle as shown in Figure 7. The motor speed can be derived as follows:

$$\text{Motor speed}\,(rpm) = \frac{60}{M \times T}$$

$$= \frac{60}{M \times C \times P \times N}$$

where, M = Number of motor pole pairs

T = Time taken for one electrical cycle

C = Capture timer count value

P = Capture timer clock period

N = Number of capture events in each electrical cycle $(= 2)$

For a four pole pair motor and a capture timer clock period of 2us, the effective motor speed is calculated as:

$$\text{Motor speed}\,(rpm) = \frac{60}{4 \times C \times (2 \times 10^{-6}) \times 2}$$

$$= \frac{3750000}{C}$$

At low motor speeds, this calculation fails (with a divide by zero) if it executes before any transition occurs on the signal from the Hall effect position sensor, or if the 16-bit capture timer maximum limit is reached giving an incorrect value for the time between transitions. How often this calculation is performed and the clock rate of the capture timer are both dependent on the supported motor speed range.

6.2.2 Discrete PID Speed Controller

A proportional-integral-derivative (PID) controller is employed to maintain the motor at the correct speed. This is a control loop feedback mechanism that attempts to correct the error between a measured process variable and a desired setpoint by calculating and then outputting a corrective action that can adjust the process accordingly; the measured process variable in this instance being the motor current speed while the desired setpoint is the speed demand.

Output from the PID algorithm is defined as:

$$Output(t) = K_p\varepsilon(t) + K_i\int_0^t \varepsilon(\tau)d\tau + K_d\frac{d\varepsilon}{dt}$$

where, K_p = Proportional gain

K_i = Integral gain

K_d = Derivative gain

ε = Error (= Setpoint - Measured process value)

t = Time, present

τ = Time, past

Table 3 provides a summary of the PID parameters.

PID parameter	Function
Proportional term	▪ Determines the reaction to the current error. ▪ Larger gain permits faster response to error. ▪ Have remaining small steady state error.
Integral term	▪ Determines the reaction based on the sum of recent errors. ▪ Eliminates steady state error. ▪ Larger gain eliminates steady state error quicker. ▪ Excessive gain slows system response and causes oscillation.
Derivative term	▪ Determines the reaction to the rate at which the error has been changing. ▪ Speed up system response. ▪ Larger gain decreases overshoot but slows down transient response.

Table 3. Summary of PID parameters.

The discrete PID speed controller is triggered at a fixed time interval. This sampling time and the PID gain terms can be tuned to obtain the desired speed response from the motor of interest, in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the setpoint, and the degree of system oscillation.

A complete discussion of PID control strategies, performance and implementation is outside the scope of this document. PID control is a common topic in industrial control systems and there are many sources of reference information available.

6.2.3 Independent PWM Trapezoidal Commutation

The eCOG1X MCPWM peripheral supports a number of different PWM strategies or operating modes. For independent PWM trapezoidal commutation, the MCPWM is configured in *centre-aligned* mode.

The MCPWM peripheral input clock must be initialised before the block can be used. A suitable clock source from one of the external reference oscillators or internal PLL multipliers is first assigned for timer group 1 or 2 via register ***ssm.clk_src1***. The MCPWM block is then allocated to the selected timer group in register ***ssm.tmr_src***. Next, a divider tap is selected for the MCPWM peripheral via register ***ssm.clk_div4***. The resulting input clock to both the MCPWM timebase period counters is determined finally by the MCPWM 16-bit clock prescaler division factor, set in register ***mcpwm.prescaler***.

In *centre-aligned* mode, the PWM output waveforms are always aligned at the centre of the total PWM period, which is now two timebase counter periods. If the timebase period register value is ***m*** and the transition value is ***n***, then the output waveforms are as shown in Figure 11.

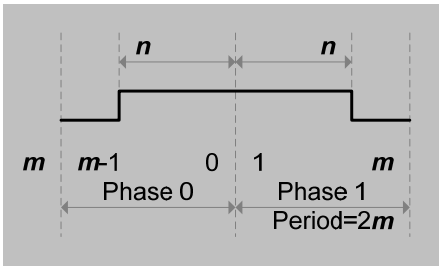


Figure 11. MCPWM centre-aligned mode.

The six PWM channel outputs are set to driven active-high mode via register ***mcpwm.cfg_op*** and their timer channels assigned to the same period counter in register ***mcpwm.cfg_period***. This period counter is then set to *symmetrical* mode via register ***mcpwm.cfg_period*** and the outputs are set to *toggle* mode in register ***mcpwm.cfg_pwm***. The PWM channels *transition* values are set via the ***mcpwm.pwm**** registers.

The period counter runs from zero to ***m*** and back to zero. In the first half of the PWM cycle (phase 0), the timebase period counter counts down from ***m*** -1 to zero; in the second half of the cycle (phase 1), it counts up from 1 to ***m***, such that the actual PWM cycle time is 2***m*** clocks. When the period counter value matches the *transition* value ***n*** in timebase phase 0, the PWM output signal is set; when it matches the transition value in timebase phase 1, the PWM output signal is cleared.

The six PWM channel transition values are updated during a *medium event* and the relevant PWM channels activated during a *capture event*, as described in Section 6.2. The trapezoidal commutation follows the sequence shown in Figure 7.

7 Example Application Software

The application software demonstrates controlling a 3-phase BLDC motor control with Hall sensor feedback, running on a custom eCOG1X motor control test board driving the Maxon EC 32 motor (as described in Section 5).

7.1 Project Files

The motor control example application software is contained in the associated compressed archived file <AN061SW.zip>. This file is available for download from the application notes page on the Cyan website at www.cyantechnology.com. Extract the example application files from the zip file to the CyanIDE projects directory.

7.1.1 Terminal Application Configuration

A terminal application on the host PC can be used to monitor the motor control debug messages output from the target system via the eCOG1X UART serial port. The terminal application software should be configured to use the correct COM port with settings of 19,200 baud, eight data bits, no parity, one stop bit and no flow control.

7.1.2 Building and Downloading to the Target System

Launch CyanIDE V1.4 and select ***Project->Open…*** from the main menu to open the project file *sensoredBLDC.cyp*. The zip file contains only the source code and no object code. Compile and build the project by selecting ***Build->Rebuild All*** from the main menu. Download the generated object code file to the target system by selecting ***Debug->Download***. Reset the target system and the example program starts executing.

7.2 Waveforms

The diagrams below are oscilloscope traces that show the PWM outputs and Hall effect position sensor inputs together with the voltages on two of the motor phases during trapezoidal commutation. These signals, measured on the motor control hardware, show the motor running at 2500rpm in both directions.

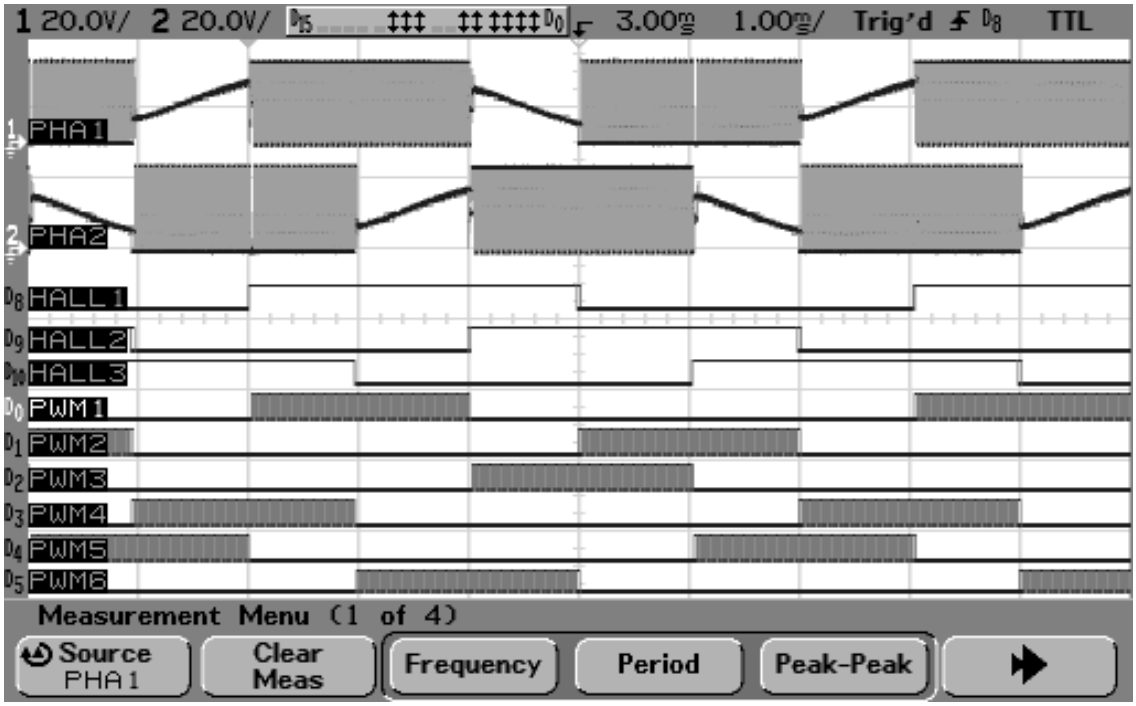


Figure 12. 3-phase BLDC motor running clockwise at 2500rpm.

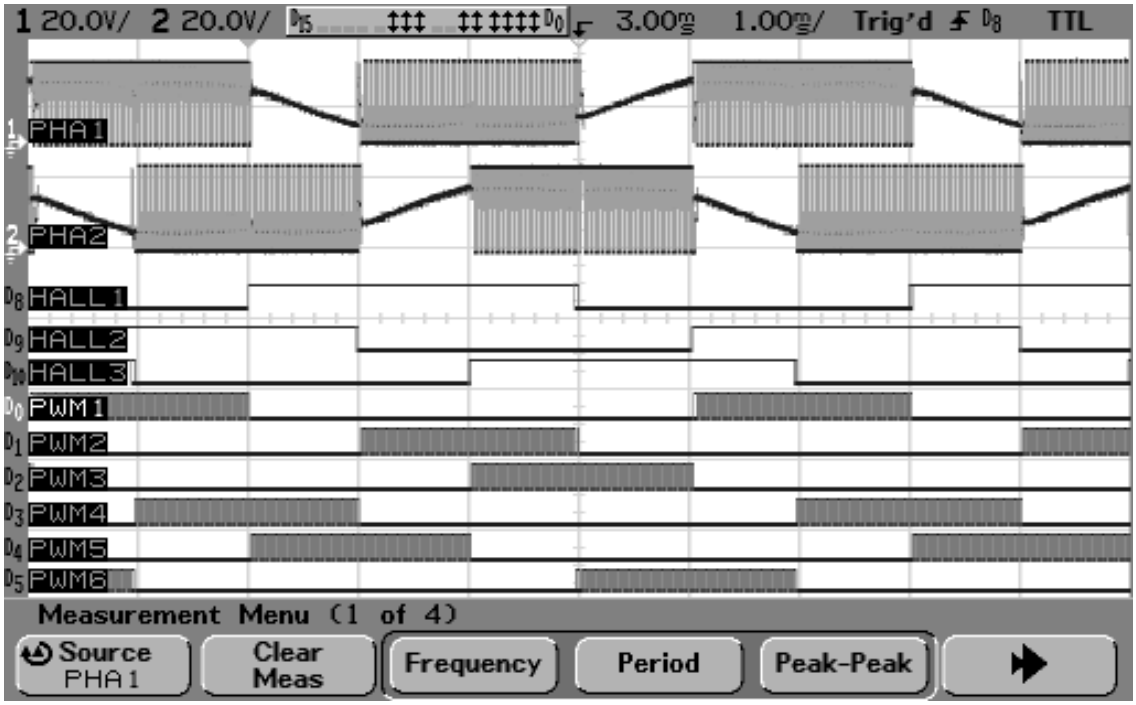


Figure 13. 3-phase BLDC motor running anti-clockwise at 2500rpm.

7.3 Motor Control Performance

A measure of the eCOG1X performance in this 3-phase BLDC motor control application can be estimated by calculating the processor loading on the eCOG1X CPU core. Most of the CPU processing is centred on the individual *fast event*, *medium event* and *capture event* routines. The CPU loading due to the *fast event* routine is dominant as it occurs most often.

With the motor speed maintained at an arbitrary 2500rpm, empirical measurements were made of the time taken to service the interrupt for each event. From these time measurements, the equivalent number of CPU clock cycles was calculated. The total number of CPU clock cycles required for all three event routines in one second were added together to get a total number of CPU clock cycles used per second.

The results of these calculations are shown in Table 4.

Parameter	Value
eCOG1X CPU clock rate	64MHz
Fast event interrupt rate	20kHz
CPU clock cycles to service *fast event* (20kHz)	12
Medium event interrupt rate	100Hz
CPU clock cycles taken to service *medium event* (100Hz)	1843
Capture event interrupt rate (at 2500rpm)	1kHz
CPU clock cycles to service *capture event* (at 2500rpm)	82
Total CPU clock cycles per second for *fast event*	240,000
Total CPU clock cycles per second for *medium event*	184,300
Total CPU clock cycles per second for *capture event*	82,000
Total CPU clock cycles available in one second	64,000,000
Effective CPU loading **= (240,000 + 184,300 + 82,000) / 64,000,000 x 100%**	**0.79%**

Table 4. eCOG1X CPU loading measurement results.

With an effective CPU loading of under 1% for the steady-state interrupt service routines performing the low-level real-time control of the BLDC motor, the eCOG1X more than adequately supports the motor control algorithm, while leaving ample CPU processing time available for other tasks.

8 Conclusion

The eCOG1X serves as a suitable enabling technology to support high performance motor control applications. This is due to the combination of a high performance 16-bit processor core and a rich set of peripheral functions, including a flexible multi-channel PWM timer block designed specifically for controlling 3-phase motors.

The eCOG1X has been demonstrated to be more than capable in driving a 3-phase BLDC motor with trapezoidal current control, using Hall effect sensors for position feedback. The hardware external to the microcontroller can be scaled up easily to support larger motors, requiring only minor changes to some software parameters. The core motor control software routines are proven and remain the same, independent of the motor size.

Appendix A Application Program Interface

Description of the primary API function calls in the relevant source files is as listed.

A.1 sensoredBLDC.c

`void mc_get_adc(void);`

Transfers the ADC conversion results from the relevant peripheral registers to the ADC data variables for later processing. This routine reads the input speed demand, the motor shunt current and bus voltage.

Parameters

Returns

None.

Example

```
// Get ADC conversion data
mc_get_adc();
```

`unsigned int mc_get_spd_demand(unsigned int spdDemand);`

Calculates the required direction and speed demand based on the ADC conversion result of the potentiometer input voltage. The centre position on the potentiometer represents zero speed, allowing both positive and negative speed demands.

Parameters

spdDemand	ADC conversion result of a potentiometer input

Returns

The interpreted speed demand.

Example

```
// Get user speed demand input
adc.spdDemand = mc_get_spd_demand(adc.spdDemand);
```

`unsigned int mc_get_rpm(void);`

Calculates the current motor speed based on the elapsed time count in capture timer 1, which triggers whenever the input for Hall sensor A changes state.

Parameters

Returns

The motor speed in units of rpm.

Example

```
// Get motor speed
measSpeed = mc_get_rpm();
```

`void mc_commutate(void);`

Reads the Hall sensor state and performs the required trapezoidal commutation depending on the rotor position and direction.

Parameters

Returns

None.

Example

```
// Execute commutation
mc_commutate();
```

`void mc_stop_motor(void);`

Disables the MCPWM outputs, the MCPWM counter and the timer interrupts. Resets all the MCPWM transition values and the measured speed value to zero. This stops the motor.

Parameters

Returns

None.

Example

```
// Stop motor
mc_stop_motor();
```

A.2 pid.c

`void pid_init(int pGain, int iGain, int dGain, struct pidData *pid);`

Initialises the PID controller parameters. The controller parameters can be tuned by changing the PID gains to get the desired closed-loop control scheme output result.

Parameters

pGain	Proportional gain
iGain	Intergral gain
dGain	Derivative gain
pid	Pointer to a structure containing the PID controller parameters

Returns

None.

Example

```
#include "pid.h"

// Initialise PID controller
pid_init(K_PROPORTIONAL, K_INTEGRAL, K_DERIVATIVE, &spdPidData);
```

`int pid_controller(int setPoint, int processValue,`
`struct pidData *pidSt);`

Calculates the closed-loop control scheme output value from the desired and actual reference level, using the previously set PID gain parameters. The PID controller maintains the required motor speed.

Parameters

setPoint	Desired reference level
processValue	Actual reference level
pid	Pointer to a structure containing the PID controller parameters

Returns

PID controller output result.

Example

```
#include "pid.h"

// Run speed PID controller
spdPidOutput = pid_controller((int)adc.spdDemand, (int)measSpeed, &spdPidData);
```

A.3 trapez_comm.c

```
void trapez_comm(unsigned int hallState,
      unsigned int motorDirClkwise);
```

Implements the next trapezoidal commutation state according to the current state of the Hall effect sensor inputs. The commutation state is stepped through one of two sequences, one for clockwise and one for anti-clockwise rotation.

Parameters

hallState	Hall effect sensors state
motorDirClkwise	Flag to indicate that motor is running in the default clockwise direction

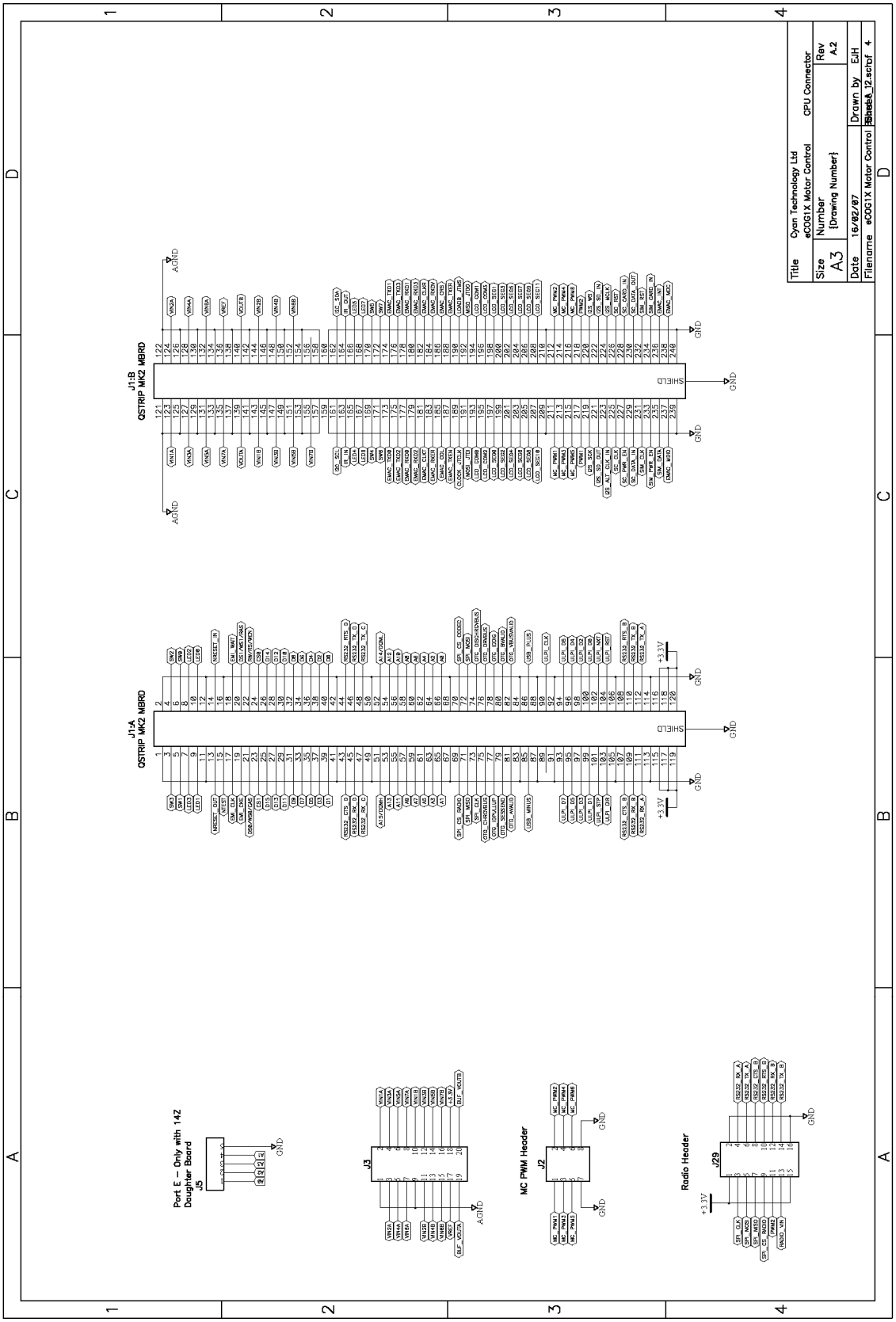
Returns

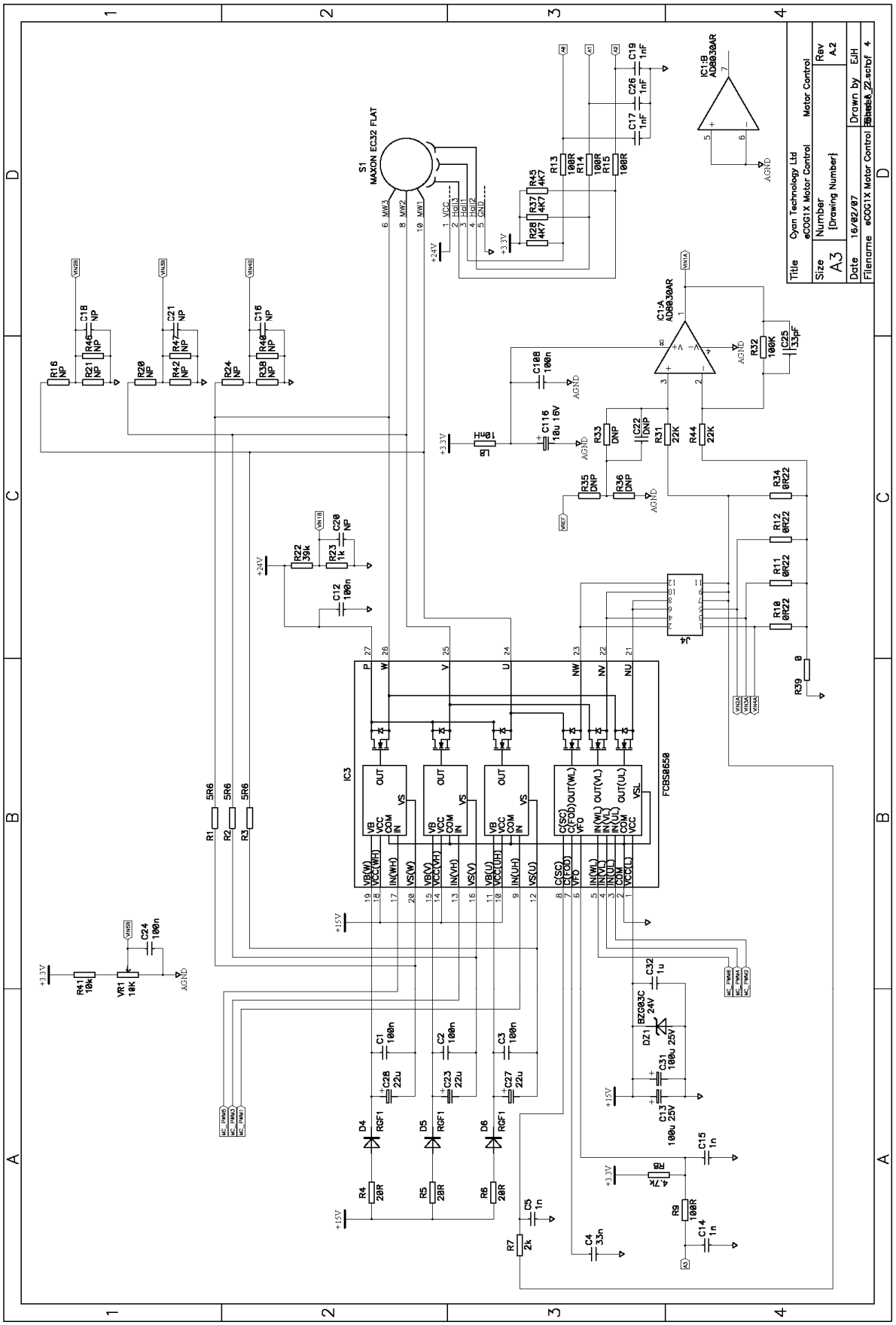
None.

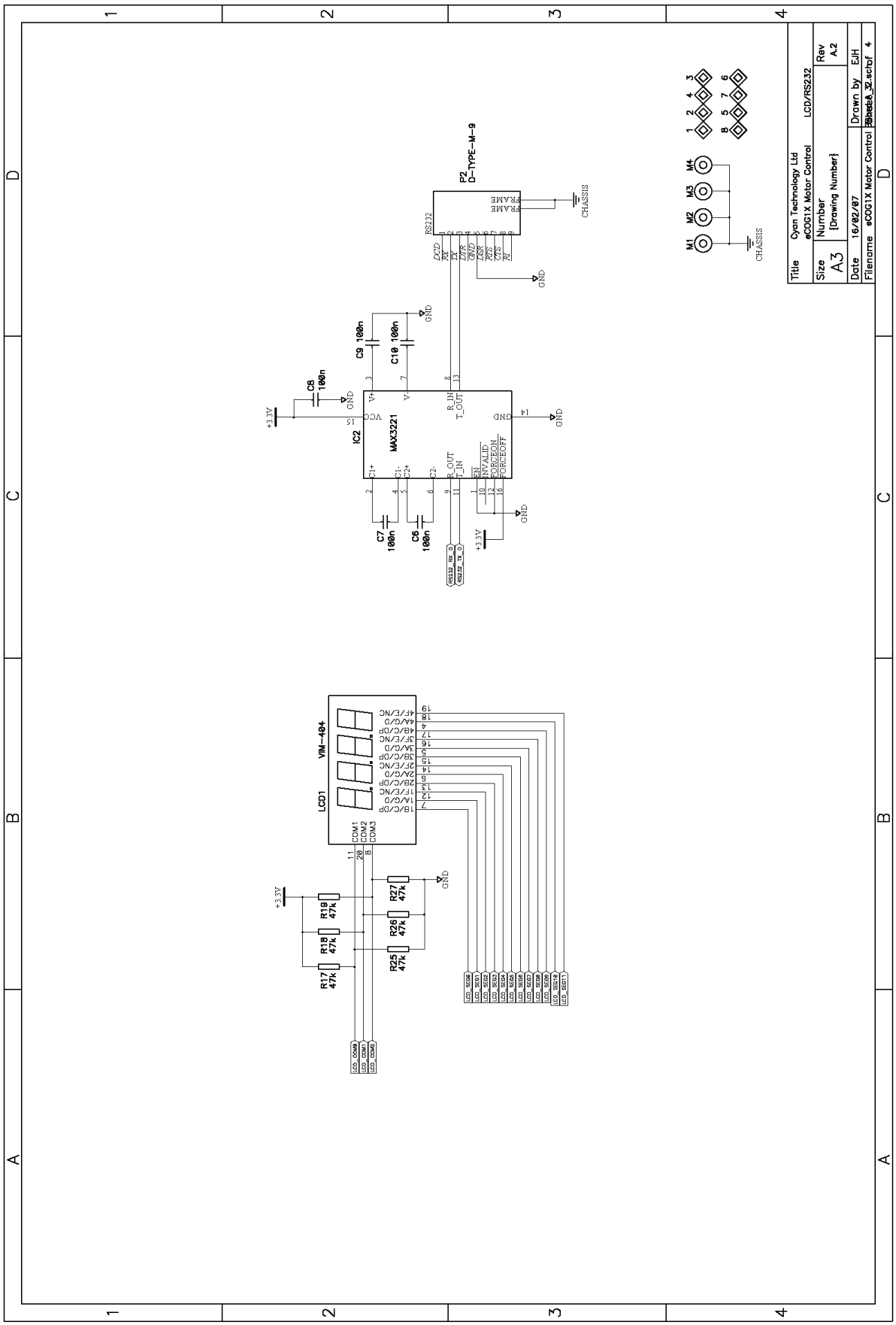
Example

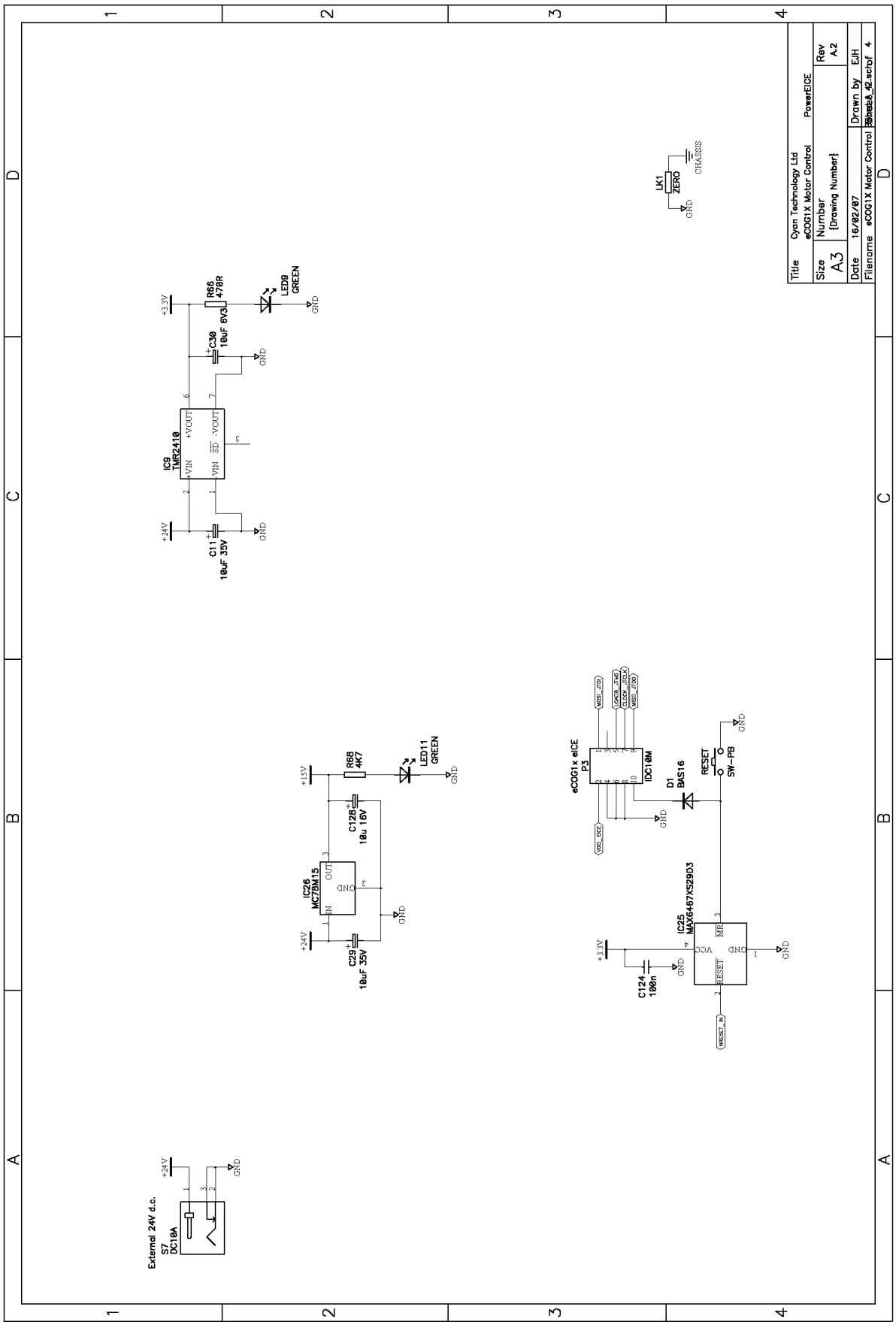
```
#include "trapez_comm.h"

// Commutate trapezoidally according to rotor position and direction
trapez_comm(hallState, flag.motorDirClkwise);
```

Appendix B Hardware Schematics




Title Cyan Technology Ltd eCOG1X Motor Control Motor Control
Size A3
Number {Drawing Number}
Rev A.2
Date 16/02/07
Drawn by EJH
Filename eCOG1X Motor Control Sheet4_22.schdoc 4
S1 MAXON EC32 FLAT
IC3 FCBS8650
IC1A AD8630AR
IC1B AD8630AR
J4
+3.3V
+15V
+24V
AGND


Title Cyan Technology Ltd eCOG1X Motor Control LCD/RS232
Size A3
Number {Drawing Number}
Rev A.2
Date 16/02/07
Drawn by EJH
Filename eCOG1X Motor Control
Sheet 4 of 4
P2 D-TYPE-M-9
RS232
CHASSIS
M1 M2 M3 M4
IC2 MAX3221
C8 100n
C9 100n
C10 100n
C7 100n
C6 100n
LCD1 VIM-404
COM1 COM2 COM3
R17 47k
R18 47k
R19 47k
R25 47k
R26 47k
R27 47k
+3.3V
GND


Title Cyan Technology Ltd
eCOG1X Motor Control PowerEICE
Size A3
Number {Drawing Number}
Rev A.2
Date 16/02/07
Drawn by EJH
Filename eCOG1X Motor Control Sheet4_A2.schpf 4
LK1 ZERO
CHASSIS
GND
R66 470R
LED9 GREEN
+3.3V
C30 10uF 6V3
IC9 TMR2410
+VOUT
-VOUT
+VIN
-VIN
SD
+24V
C11 10uF 35V
R68 4K7
LED11 GREEN
+15V
C128 10u 16V
IC26 MC78M15
OUT
IN
C29 10uF 35V
eCOG1x eICE P3
IDC10M
MRS_JTD
LOADB_JTMS
CLOCK_JTCLK
MRS_JTDO
VRS_RSE
D1 BAS16
RESET SW-PB
IC25 MAX6457XS29D3
MR
VCC
RESET
C124 100n
NRESET_IN
External 24V d.c.
S7 DC10A