

## Field Oriented Control (FOC) of a Three-Phase BLDC Motor Using a Rotary Inductive Position Sensor

*Author: Maria Loida Canada  
Microchip Technology Inc.*

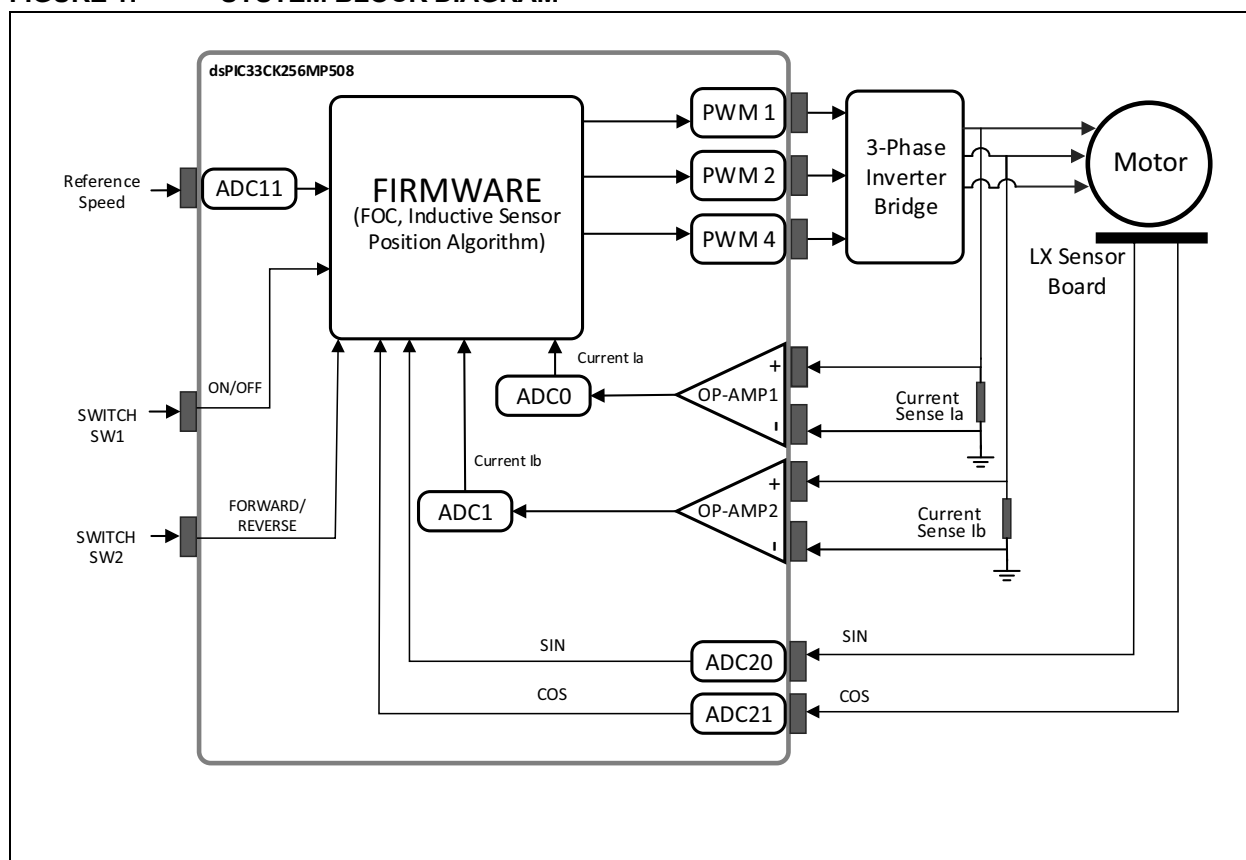
### INTRODUCTION

The need for highly accurate position sensors, like inductive position sensors, becomes prevalent due to the rise in demand in automotive, industrial and commercial applications. Inductive position sensors are ideal replacements for magnet-based position and speed sensors like Hall Effect sensors, optical encoders and magnetic resolvers. They excel with improved accuracy, magnetic stray field immunity and do not require a magnet as a target. They offer design flexibility because the sensor is not integrated with the Integrated Circuit (IC) or the motor.

This application note is created to highlight the use of the LX34050 inductive position sensor in providing an accurate rotor position for the FOC drive technique. FOC is useful for applications that require a fast dynamic response for speed and torque changes. The LX34050 achieves ASIL-B automotive safety integrity level. The control unit used is Microchip Technology's 16-bit DSC's dsPIC33CK, with a DSP engine for fast mathematical calculations and on-chip peripherals, like PWM, ADCs, op amps, that simplify the implementation and reduce the overall system components.

This document discusses the inductive position sensor, the algorithm used for the position and speed measurement and the results of the testing. This solution can efficiently drive the motor bidirectionally. The system block diagram is shown in the following figure.

**FIGURE 1: SYSTEM BLOCK DIAGRAM**



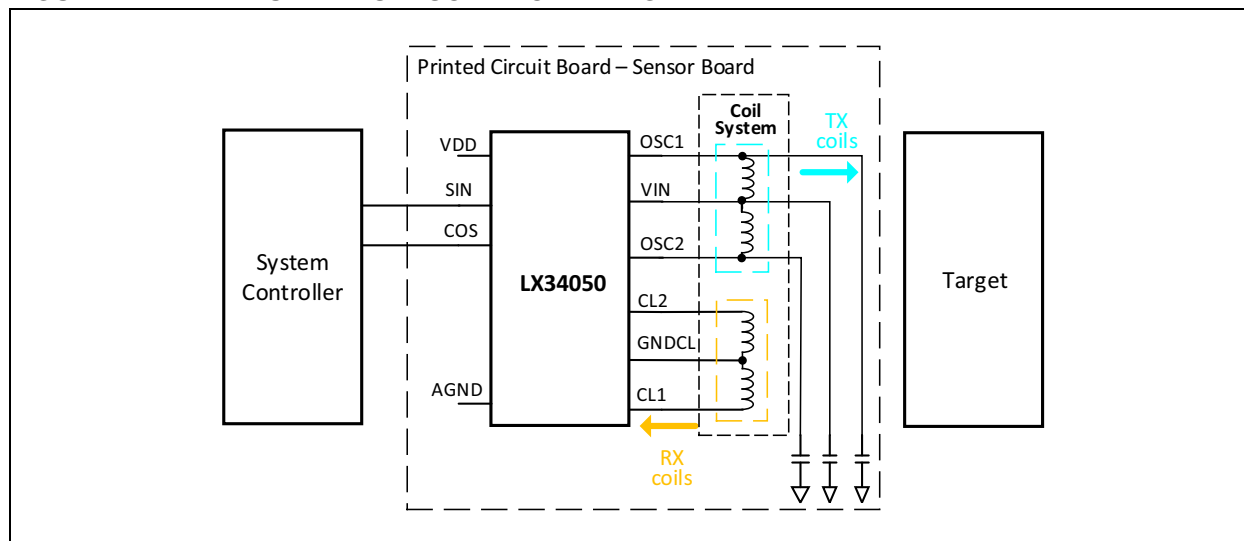
## INDUCTIVE POSITION SENSOR

An inductive sensor is used to measure position. It can detect metal targets approaching the sensor without physical contact with the target. Proximity Sensors are roughly classified into the following three types according to the operating principle: the high-frequency oscillation type using electromagnetic induction, the magnetic type using a magnet and the capacitance type using the change in capacitance. This application is developed using the high-frequency oscillation type with electromagnetic induction because of its metal-sensing versatility and reliability.

## Block Diagram of Inductive Sensor

The figure below shows the typical inductive position sensor setup. It is consisting of the connection to the system controller, the interconnection of the integrated circuit with the coil system (TX and RX coils) on the PCB, and the target. The target is mounted on the motor shaft and rotates along with the rotor, and it has no physical contact with the stationary sensor.

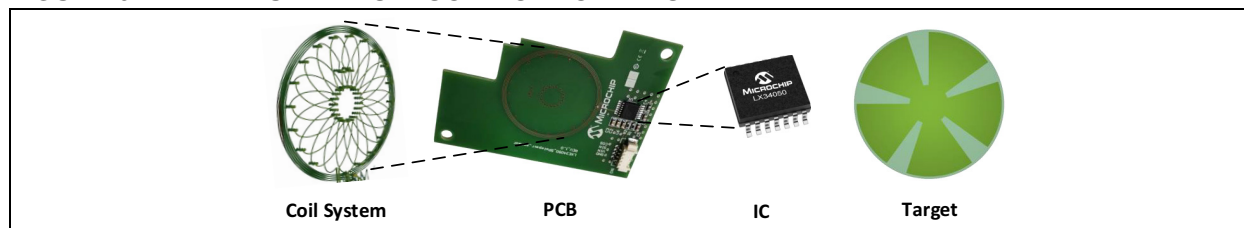
**FIGURE 2: INDUCTIVE SENSOR BLOCK DIAGRAM**



## Components of an Inductive Sensor

The inductive position sensor module has four critical components shown in the figure below: the coil system, the target, the IC and the Printed Circuit Board (PCB).

**FIGURE 3: INDUCTIVE SENSOR COMPONENTS**

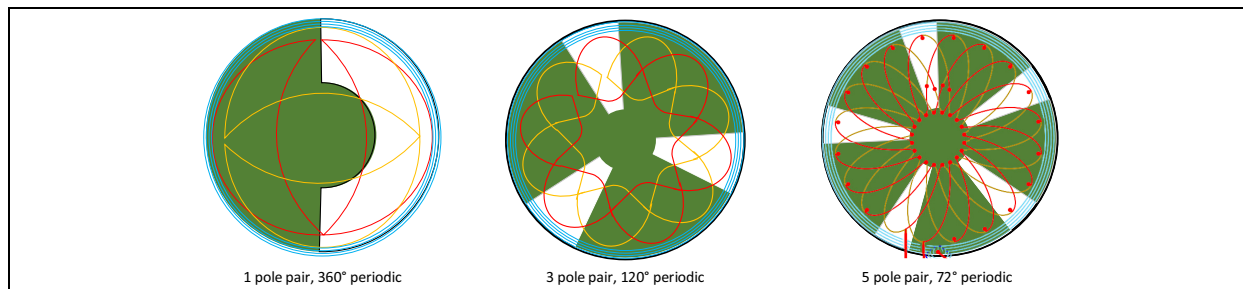


## COIL SYSTEM

The coils are created as traces on the PCB, with primary (TX coils) and secondary (RX coils) windings placed to detect the movement of a metal target. The coils are the transducer of the system. It is not integrated into the LX34050 chip; therefore, inductive sensors offer great flexibility for the coil design. The coil

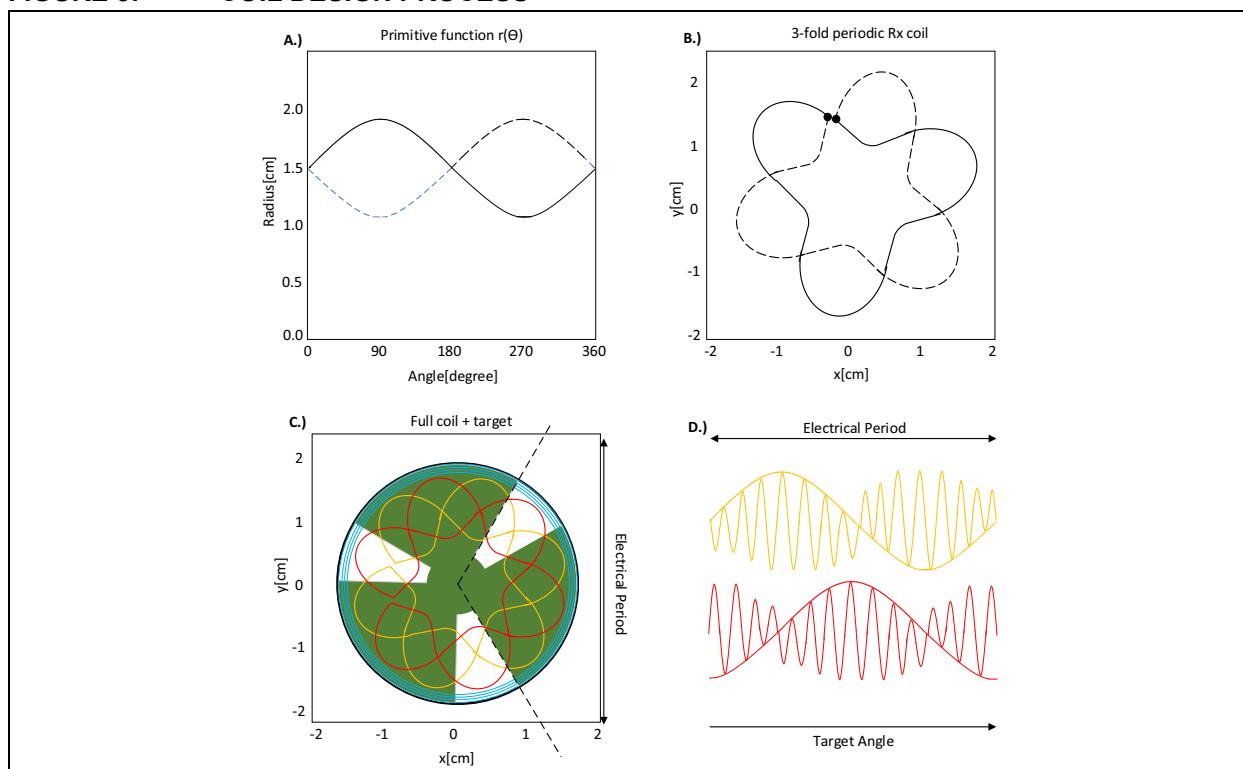
arrangement determines the periodicity, it can be tailored to the application, and be adapted to the number of pole pairs of the motor.

The variable periodicity matches the sensor range to the motor electrical period, which maximizes the sensor accuracy. The following figure shows the examples of variable periodicity depending on the number of motor pole pairs.

**FIGURE 4: INDUCTIVE SENSOR WITH VARIABLE PERIODICITY**

The step-by-step coil design process is shown in the following figure. The RX coils are derived from a sinusoidal primitive function  $r(\theta) = R_0 + \Delta R \sin(\theta)$  and its complementary  $r'(\theta) = R_0 - \Delta R \sin(\theta)$ . The RX coil pattern is, then, obtained using a polar coordinate transformation,  $x = r(N)\cos(\theta)$ ;  $y = r(N)\sin(\theta)$ , to fit N periods in a full turn, N being the number of pole pairs of the motor.

The full coil set is, then, completed by adding another RX coil, shifted by  $90^\circ/N$ , a circular TX coil around the RX coils, and a target (C in the figure below). This configuration will provide a two-phase signal set as sketched in D in the figure below.

**FIGURE 5: COIL DESIGN PROCESS**

## THE TARGET

The target is a circular PCB attached to the rotor and has lobes equal to the number of motor pole pairs. Energy from the TX coils induces eddy currents on the target, which, in turn, generate a magnetic field directed back and received by the RX coils.

Eddy currents are induced by changing the magnetic field and flow in closed loops within the conductor, perpendicular to the plane of the magnetic field. Like any current flowing through a conductor, an eddy current

will produce its magnetic field. Lenz's Law states that the direction of a magnetically-induced current, like an eddy current, will be such that the magnetic field produced will oppose the change of the magnetic field which created it. The next result is that the metal target negates the magnetic field over the receive coil.

## THE IC

The LX34050 is an integrated programmable data conversion IC designed for interfacing to and managing PCB-based inductive position sensors.

The device comprises an integrated oscillator circuit for driving the TX coil of the inductive sensor, along with two independent analog conversion paths for conditioning, converting and processing sine and cosine signals from the RX coils of the sensor. Each path incorporates an EMI filter, demodulator and programmable amplifier.

The LX34050 output interface includes two single-ended sin and cos signals that change as the metal target is moved over the sensor. The propagation delay and bandwidth of the analog paths are optimized for high-speed automotive, industrial and medical sensor applications. The LX34050 is an absolute position sensor, and, hence, the sensor phase angle must be aligned to the motor electrical angle for the sensor to work properly. This is addressed by correctly setting the theta offset between the sensor angle and rotor angle in the firmware.

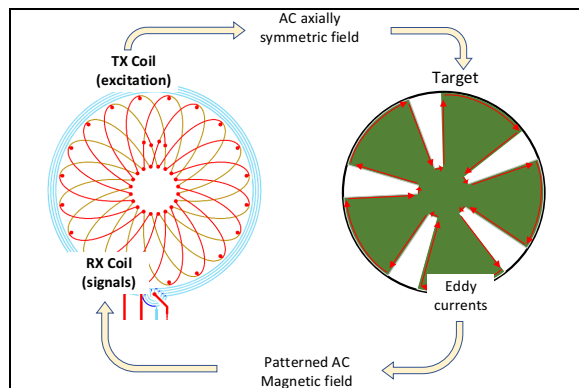
## THE PCB

The PCB designs are provided on the Microchip website.

## Inductive Position Sensor Operation

Inductive position sensors measure the position of a metallic target placed in front of a set of inductive coils. The target is attached to the motor rotor. It is designed with several lobes matching the number of pole pairs of the motor. The coils are created as traces on the PCB, which also has the LX34050 chip mounted on it. These coils are connected to the integrated circuit interface. The LX34050 chip, whose function is to drive and read voltages from the coils, computes the “target” angle from the received signals and outputs the calculated value in the form of single-ended sin/cos signals.

**FIGURE 6: PROCESS OF CONVERTING TARGET POSITION TO SIN/ COS SIGNALS**



As depicted in the figure above, the external circular coil (cyan colored coil) is the TX coil. It is controlled by the sensor IC to produce an axially symmetric AC magnetic field. The TX coil field excites eddy currents in the target, which generates a secondary field named the

target field. The target is composed of several lobes; therefore, the target field does not have cylindrical symmetry but has an angular dependency. The target rotates together with the rotor of the motor and negates the magnetic field over the coils, depending on its position. The target field period in the angular domain corresponds to the angular sector encompassed by two adjacent target lobes. This angular span is also referred to as an electrical period and is defined to be equal to  $360^\circ$  electrical ( $1^\circ \text{ el.} = 1^\circ / N \text{ mech.}$ ).

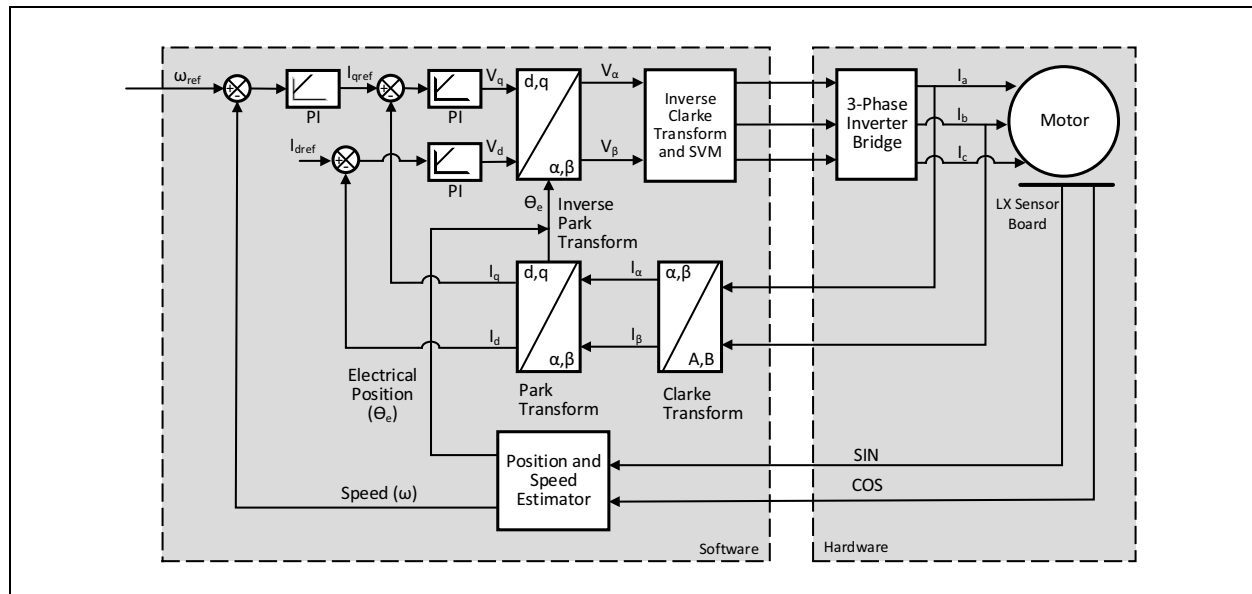
The RX coils (red and brown coil like a Spirograph) receive the net field generated by the transmitter and negated by the target and generate a set of two AC signals. The receiving (RX) coils are designed to capture the target field while being insensitive to the TX field. This is accomplished by winding each RX coil in such a way that every electrical period contains two loops connected in series, the first turning in the clockwise direction and the second in the counterclockwise direction. The cylindrically symmetric TX field is, thus, rejected as it induces opposite voltages in the two loops, while the target field is captured due to the matching periodicity. As the target rotates, the RX coil generates an AC voltage at the carrier frequency characterized by an amplitude modulation envelope that is approximately sinusoidal over an electrical period (see Figure 5D). By optimizing the RX coil shape, the envelope profile can be made closer to an ideal sinusoid.

The position sensor interface IC converts the input signals from the RX coils into single-ended sine/cosine output signals. These signals are passed to the controller unit that computes the angle using the arc tangent function and estimates the speed based on the theta difference in each period.

The inductive position sensor’s improved accuracy is attributed to its very low-temperature drift, ratiometric measurement and dynamic signal compensation. The secondary RX channels use synchronous demodulation to eliminate differences due to temperature fluctuation. The propagation delay and bandwidth of the analog paths are optimized for high-speed sensor applications. Each path includes an EMI filter, demodulator, anti-alias filter and programmable amplifier. The output is an accurate sine and cosine signals that represents the movement of the metal target over the PCB-based sensor.

## BLOCK DIAGRAM OF SENSORED FOC

FIGURE 7: FOC BLOCK DIAGRAM



The control process in the figure above can be summarized as follows:

- The three-phase stator currents are measured. For a motor with balanced three-phase windings, only two-phase currents are sufficient to be measured. The third current is calculated using the following equation:  $i_a + i_b + i_c = 0$ .
- Clarke Transform: The three-phase currents are converted to a stationary two-axis system. The conversion provides the variables  $i_\alpha$  and  $i_\beta$  from the measured  $i_a$  and  $i_b$  values. The values  $i_\alpha$  and  $i_\beta$  are time-varying quadrature current values as viewed from the perspective of the stator.
- Park Transform: The stationary two-axis coordinate system is rotated to align with the rotor flux using a transformation based on the angle measured at the last iteration of the control loop. This conversion provides the  $i_d$  and  $i_q$  variables from  $i_\alpha$  and  $i_\beta$ . The variables  $i_d$  and  $i_q$  are the quadrature currents transformed to the rotating coordinate system. For steady state conditions,  $i_d$  and  $i_q$  are constant.
- Reference values of currents are defined as follows:
  - $i_d$  reference: Controls rotor magnetizing flux
  - $i_q$  reference: Controls the torque output of the motor
- The error signals are fed to PI controllers. The output of the controllers provides  $v_d$  and  $v_q$ , which are the voltage vectors that will be applied to the motor.
- A new transformation angle is measured and calculated based on the LX34050 inductive sensor target position. The new angle guides the FOC

algorithm as to where to place the next voltage vector.

- Inverse Park Transform: The  $v_d$  and  $v_q$  output values from the PI controllers are rotated back to the stationary reference frame using the new angle. This calculation provides the succeeding quadrature voltage values  $v_\alpha$  and  $v_\beta$ .
- Inverse Clarke Transform and SVM: The  $v_\alpha$  and  $v_\beta$  values are used to calculate the new PWM duty cycle values, which will generate the desired voltage vector.
- The speed ( $\omega$ ) is calculated from several samples of theta, which happens every PWM cycle. The FOC firmware is implemented in the ADC Interrupt Service Routine (ISR), which runs at the same rate as the PWM switching frequency.

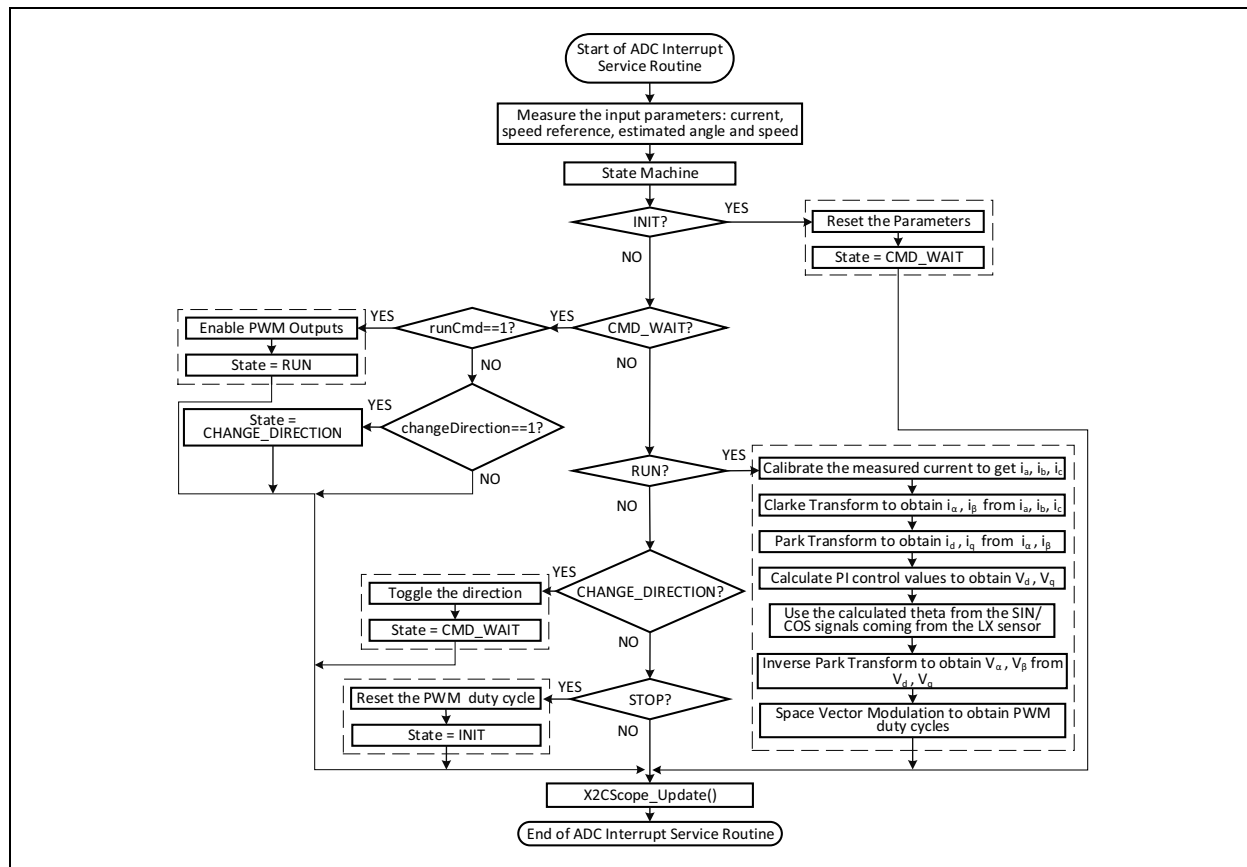
This application note did not explain the FOC operation in detail. Refer to AN1292, "Sensorless Field Oriented Control (FOC) for a Permanent Magnet Synchronous Motor (PMSM) Using a PLL Estimator and Field Weakening" for an in-depth FOC discussion.

## FLOWCHART OF SENSORED FOC AND LX SENSOR ALGORITHM

This chapter shows the flowchart of the system, the position estimator and the speed estimator block.

## Sensored FOC Flowchart

**FIGURE 8: FIRMWARE FLOWCHART**



The flowchart in the figure above illustrates the program flow in the ADC interrupt service routine, which is executed every PWM cycle. It starts with the measurements of the needed parameters such as current, speed reference for the desired speed, calculated angle and measured speed, followed by the state machine that supervises the motor operation. This state machine is composed of seven states, briefly described below:

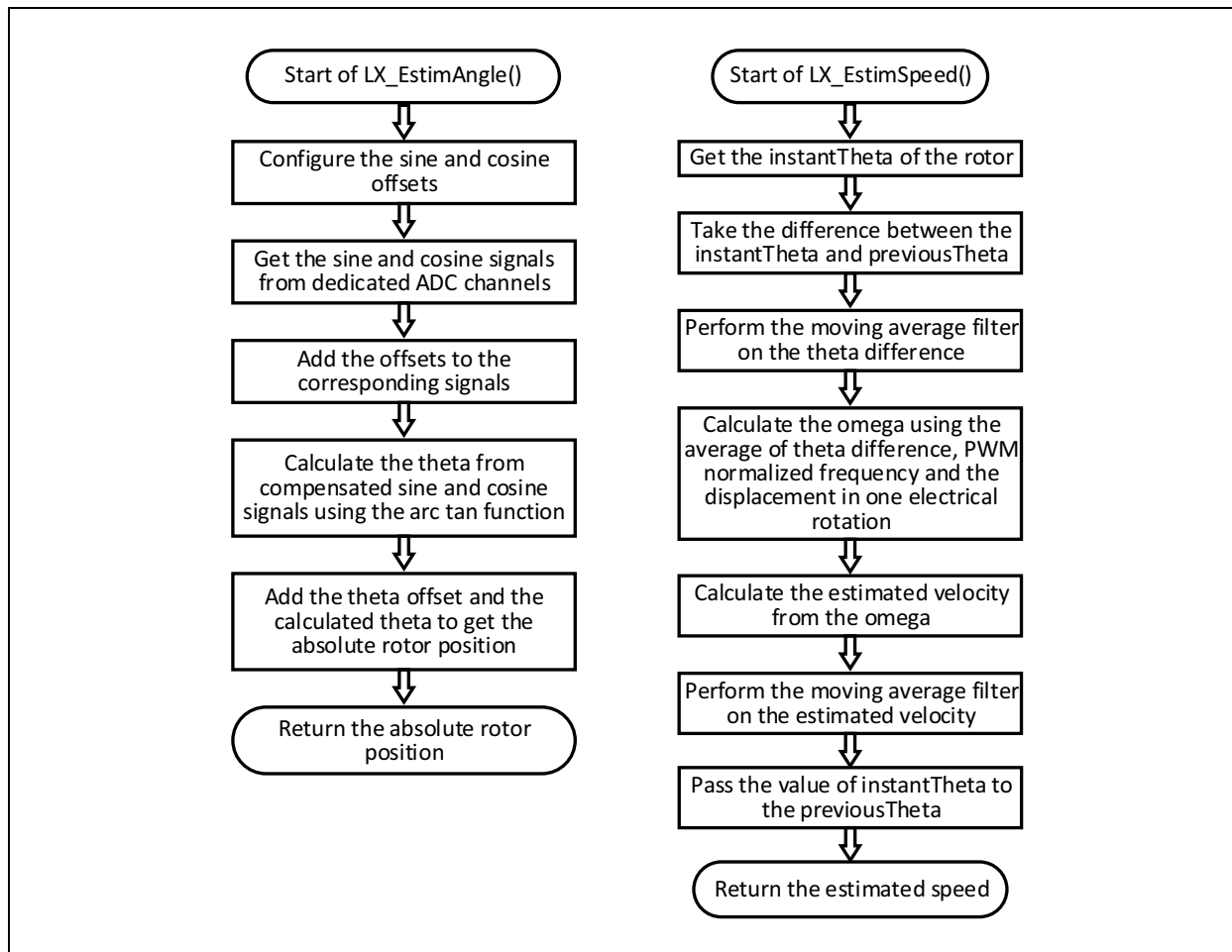
- **INIT** – Initial state where all the parameters and variables are reset. It is entered whenever the system powers up or after the motor is stopped.
- **CMD\_WAIT** – This state monitors the runCmd and changeDirection variables. The runCmd is set when its initial state is clear and the switch SW1 is pressed. Hence, its value is toggled whenever switch SW1 is pressed. In the same manner, the changeDirection value is toggled by pressing switch SW2.
- **RUN** – The motor runs using the FOC algorithm.
- **CHANGE\_DIRECTION** – The default motor direction is forward. A press in switch SW2 toggles the direction from forward to reverse. The firmware is designed to switch the direction, only when the motor is stopped.

- **STOP** – The PWM duty cycle is cleared, and the motor is stopped.

The `X2CScope_Update()` is the routine used for updating the values of the parameters being watched on the X2CScope Scope window and watch window. X2CScope is a virtual oscilloscope tool that allows runtime debugging or monitoring of this embedded application. It is a third-party plugin tool available in the MPLAB X IDE Plugins and is configurable through the MPLAB Code Configurator (MCC).

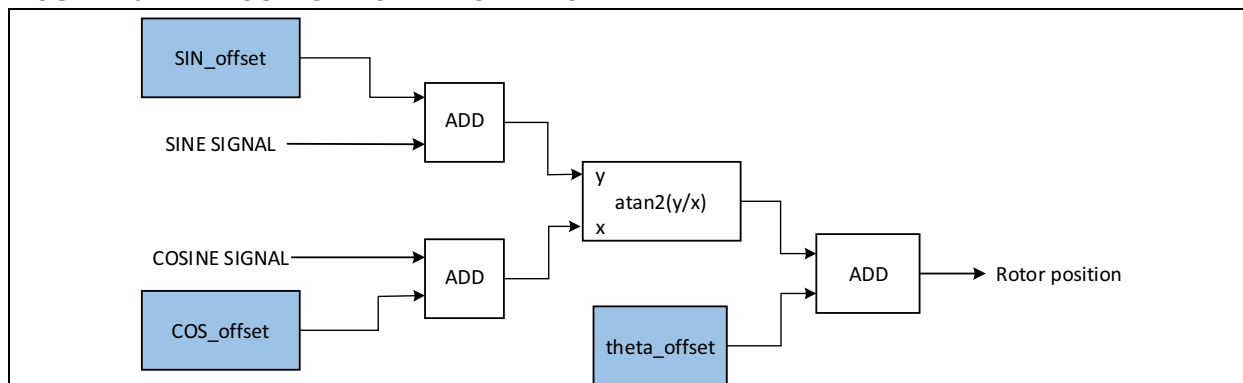
### Flowchart of Position and Speed Estimator Based on LX Sensor

The flowcharts in the following figure summarize the process of obtaining the position and speed from the raw sin/cos signals coming into the microcontroller through the ADC channels. It is a sequential procedure to calculate the angular position first, followed by the speed, because the latter is based on the angular difference taken every PWM cycle.

**FIGURE 9: POSITION AND SPEED ESTIMATOR**

## LX34050 INDUCTIVE POSITION SENSOR ALGORITHM

This section discusses the offsets that are recommended to be added to the analog signals, the arc tangent function that calculates the angle from the given ratio of sin/cos digitized values and the speed estimation.

**FIGURE 10: POSITION ESTIMATOR DIAGRAM**

The figure above shows the software blocks and the interconnections for getting the absolute rotor position. The `SIN_offset` and `COS_offset` are the values used to compensate for the DC offsets coming into the microcontroller. These values are obtained manually by using the X2Cscope. The example of acquiring the `SIN_offset` and `COS_offset` is discussed in the next chapter. The sine and cosine signals are measured from the corresponding ADC channels, ADC20 and ADC21. The adjusted sine and cosine signals are fed to the `atan2(y/x)` function.

## ATAN2(Y, X) Function

The `atan2(y/x)` function is a “two-argument arctangent” defined as the angle, given in radians, between the positive x-axis and the ray from the origin to the point (x, y). It is intended to use for returning a correct and explicit value for the angle  $\theta$  in converting from cartesian coordinates(x,y) to polar coordinates(r,θ). It is like calculating the arc tangent of y/x, except that the signs of both arguments are used to determine the quadrant of the result.

### EXAMPLE 1: QUADRANT OFFSET MACROS

```
#define ATAN2_OFFSET_Q2      0x4000
#define ATAN2_OFFSET_Q34    0x8000
```

### EXAMPLE 2: INITIAL THETA SETTINGS

$$\theta = \begin{cases} 0, & \text{if } x > 0 \text{ and } y > 0 \\ \text{ATAN2\_OFFSET\_Q2}, & \text{if } x < 0 \text{ and } y > 0 \\ \text{ATAN2\_OFFSET\_Q34}, & \text{if } x < 0 \text{ and } y < 0 \\ \text{ATAN2\_OFFSET\_Q34} + \text{ATAN2\_OFFSET\_Q2}, & \text{if } x > 0 \text{ and } y < 0 \end{cases}$$

The initial setting of theta gives its base value in the specified quadrant. In the first case, the theta is 0, which means there is no need for an offset because the theta is in quadrant 1. In the second case, the theta is in quadrant 2; therefore, the x value is negated to make it positive. The reversal of the sign causes the addition of `ATAN2_OFFSET_Q2`. For the third case, because the theta is in quadrant 3, both the x and y values are negated; hence, the theta is initialized to `ATAN2_OFFSET_Q34`. The fourth case requires the combined `ATAN2_OFFSET_Q2` and `ATAN2_OFFSET_Q34` to assign the y value with the opposite sign because the theta is in quadrant 4.

After setting the initial theta value, it is tested if y is greater than x; if it is true, swap the value of x and y. The ratio of y and x is calculated, and the result is used as an index for locating the correct value from the look-up table. The look-up table is composed of an array with 257 elements. These values, when plotted, resemble a quarter of a sinusoidal signal. The value where the index is pointing is obtained and added to the initial theta value. The result is the rotor angle.

In terms of the standard arctan function whose range is  $(-\pi/2, \pi/2]$ , it can be expressed as follows:

### EQUATION 1: ARC TANGENT OF TWO NUMBERS

$$\text{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0 \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0 \end{cases}$$

The algorithm is created based on the principle discussed in the preceding paragraph. The quadrant of the angle is determined depending on the magnitude and sign of x and y. To simplify the code execution, there are offset values added on the angle based on the quadrant where it is located. These offset values are defined in the macros in the following example.

The placement of the rotary target is not in absolute position of the motor; therefore, the calculated theta is added with the `theta.offset` to get the absolute rotor position. The example in the next chapter shows how the `theta.offset` is obtained.

## Speed Measurement

The speed is derived from several samples of the difference between subsequent rotor angle(θ). The current angle captured is named `instantTheta`, while the preceding angle captured is termed `previousTheta`. The `instantTheta` is obtained and compared to the `previousTheta`; then, the difference between the two is computed. The resulting difference is applied with the moving average filter. The moving average filter is a simple Low Pass Finite Impulse Response filter used for regulating an array of sampled data. It takes M samples of input at a time and takes the average of those to produce a single output point. The output is the filtered theta difference. It is used for omega calculation. The omega is evaluated using the following equation.



**EQUATION 2: OMEGA CALCULATION**

$$\omega = \frac{(\text{Filtered Theta Diff} * \text{PWM Normalized Freq})}{\text{Normalized Displacement}}$$

The PWM Normalized Freq is the simplified value of the PWM frequency, and the Normalized Displacement is the total angular distance rotated by the rotor in one PWM cycle. The estimated speed is, then, calculated using the following equation.

**EQUATION 3: ESTIMATED SPEED**

$$\text{Speed} = \omega * 60$$

The speed is applied with another level of filtering using the moving average filter to further reduce the random noises. The filtered speed is the measured speed fed into the PI controller.

**SAMPLE APPLICATION OF LX34050 SENSOR**

The firmware is tested using the following boards: dsPIC33CK Low Voltage Motor Control Dev Board (LVMC), a custom LX34050 sensor board created for 5-pole pairs hurst motor, an adapter board for interfacing the sensor board to the LVMC dev board, and a target conductor fabricated in the PCB. The air gap between the sensor board and the target is recommended to be between 1 mm to 3 mm. The schematics of the sensor board and the target conductor are located in [Appendix A: "Circuit Schematics"](#).

Tuning or setting the offsets is the first thing that is recommended to be done before running the motor using the LX34050 sensor as feedback. This section contains the step-by-step process of setting the offsets (sin, cos and theta).

This solution offers two methods of setting the offsets. One approach is using the PLL estimator and another technique is through manual offset setting.

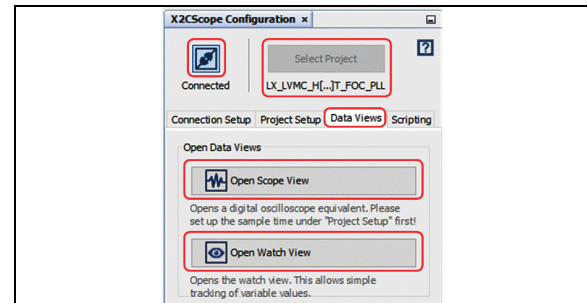
**Setting the Offsets Using PLL Estimator**

The firmware is equipped with the PLL estimator used for running the motor in the initial setting of the offsets (sin, cos and theta). In setting up the firmware initially, enable the ESTIMATOR and SETTING\_OFFSET definitions in the userparams.h file as shown in the following figure.

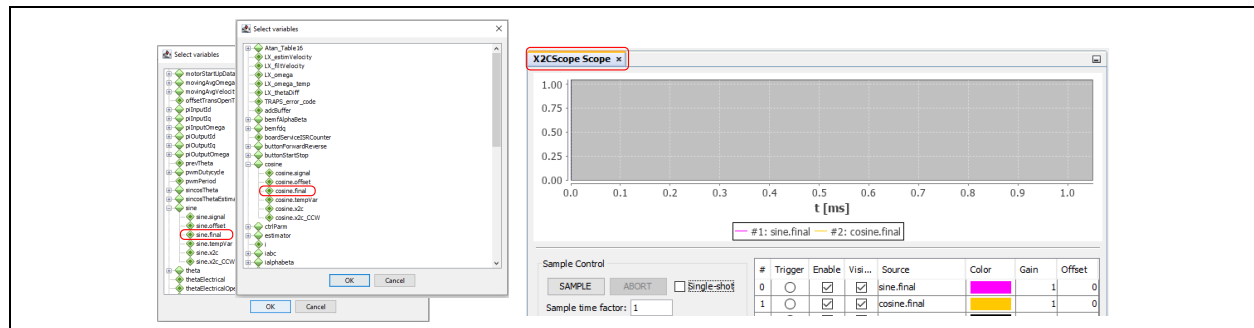
**FIGURE 11: USERPARAMS.H DEFINITIONS**

```
#define ESTIMATOR
#define SETTING_OFFSET
```

Load the program to the device. Connect the X2CScope, then open Data Views. Click Open Scope View and Open Watch View as illustrated in the following figure. For the installation guide of X2CScope, refer to [Appendix C: "X2CScope"](#).

**FIGURE 12: OPENING SCOPE AND WATCH VIEW**

In the X2CScope window, select sine.final and cosine.final from the variables as the sources for channel 0 and 1, respectively.

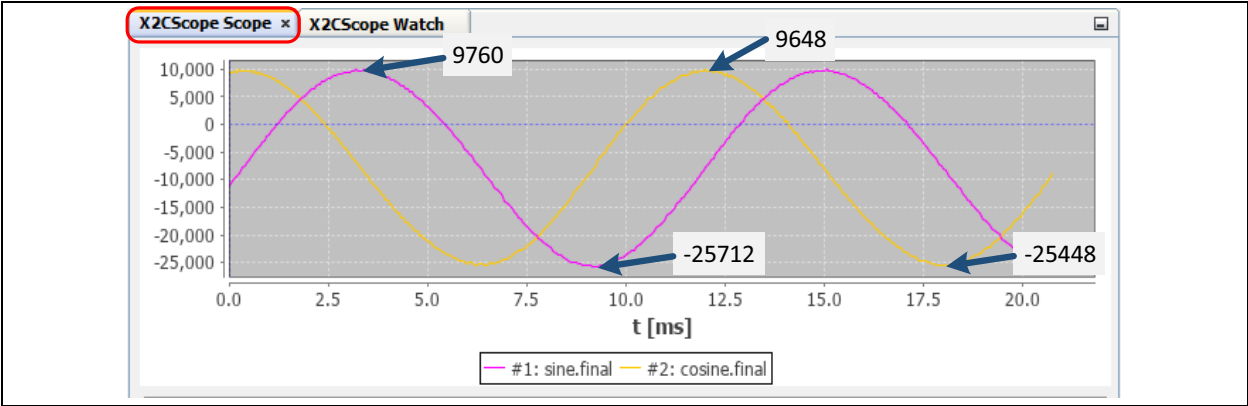
**FIGURE 13: SELECTING THE VARIABLES TO WATCH**

Run the motor and start the data capture. Observe that the sine and cosine signal in the following figure are not symmetrical around the x-axis. Calculate the offset for each signal using the formula in the following equation:

**EQUATION 4: OFFSET FOR SIN/COS SIGNALS**

$$\text{Offset} = \frac{|\text{Max} + \text{Min}|}{2}$$

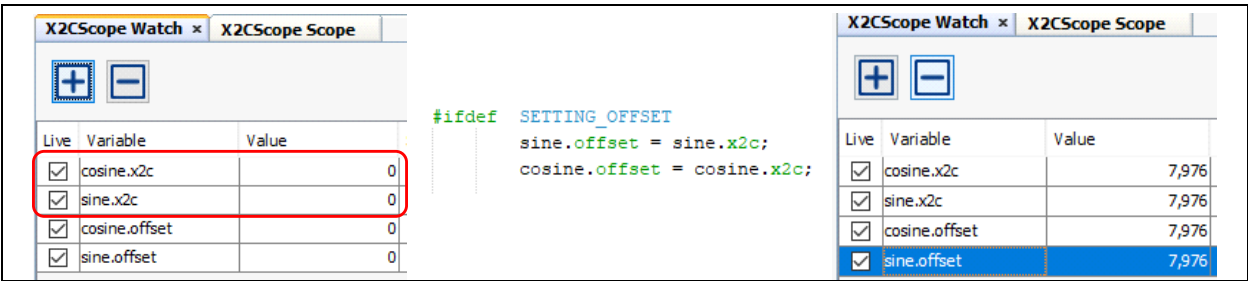
FIGURE 14: SAMPLE CAPTURE OF SIN/COS SIGNALS WITHOUT OFFSET



The following figure shows the assignment of offset values to the corresponding variables. After calculating the offsets, open the watch view, then add the x2c variables (sine.x2c and cosine.x2c). These variables are created for freely tuning the offset values through

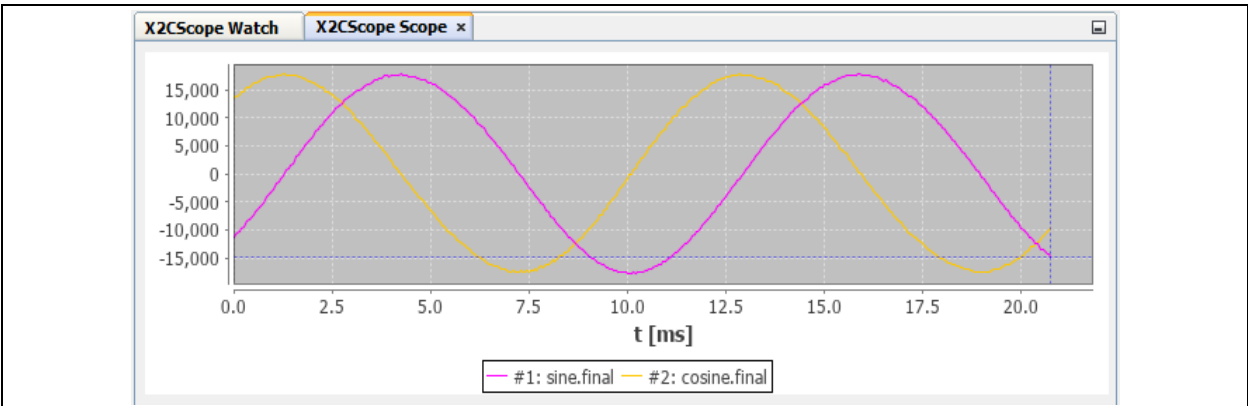
X2CScope and are assigned to the sine.offset and cosine.offset, which are also added into the watch view window for checking purposes. Place the offset values to the corresponding x2c variables.

FIGURE 15: ASSIGNING THE OFFSET TO THE CORRESPONDING VARIABLES



Run the motor and observe that the sine.final and cosine.final signals are already symmetrical around the x-axis as depicted in the following figure.

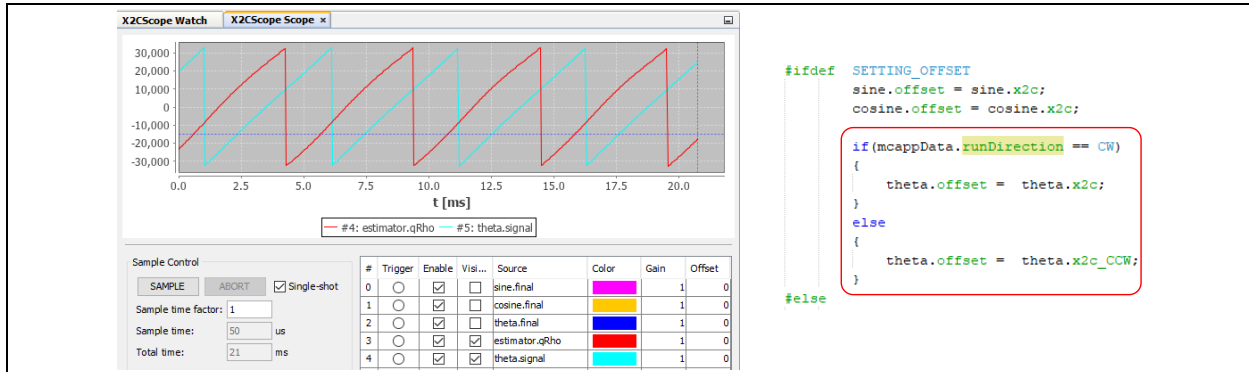
FIGURE 16: SIN/COS SIGNAL CENTERED AROUND THE X-AXIS



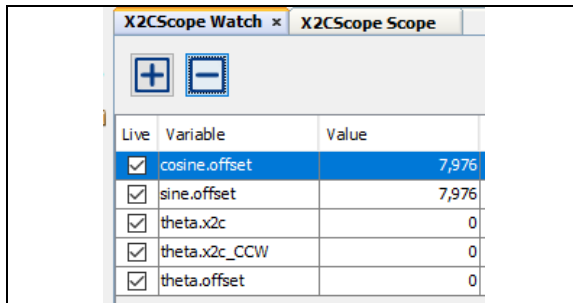
**Assigning Value to the theta.offset for Forward Direction**

Monitor the theta.signal and compare it to the absolute rotor position given by the estimator. It is recommended that the theta.offset be added to the raw calculated theta.signal to make it aligned

to the estimator.qRho. The theta.x2c and theta.x2c\_CCW are provided for real-time tuning using the X2CScope. These variables are assigned to the theta.offset.

**FIGURE 17: SETTING THE THETA.OFFSET**

Add the `theta.x2c` and `theta.x2c_CCW` variables to the watch view as shown in the following figure.

**FIGURE 18: ADDING THE THETA.X2C VARIABLES TO THE WATCH VIEW**

Use the `Theta_offset` section in the Excel sheet as a guide for faster `theta.offset` tuning. The Excel sheet is included in the code folder. The `theta.signal` is lagging behind the `estimator.qRho`; therefore, pick a large positive value around Q15(0.5) or 16383 and observe the response. Adjust the values until the `estimator.qRho` and `theta.final` overlap.

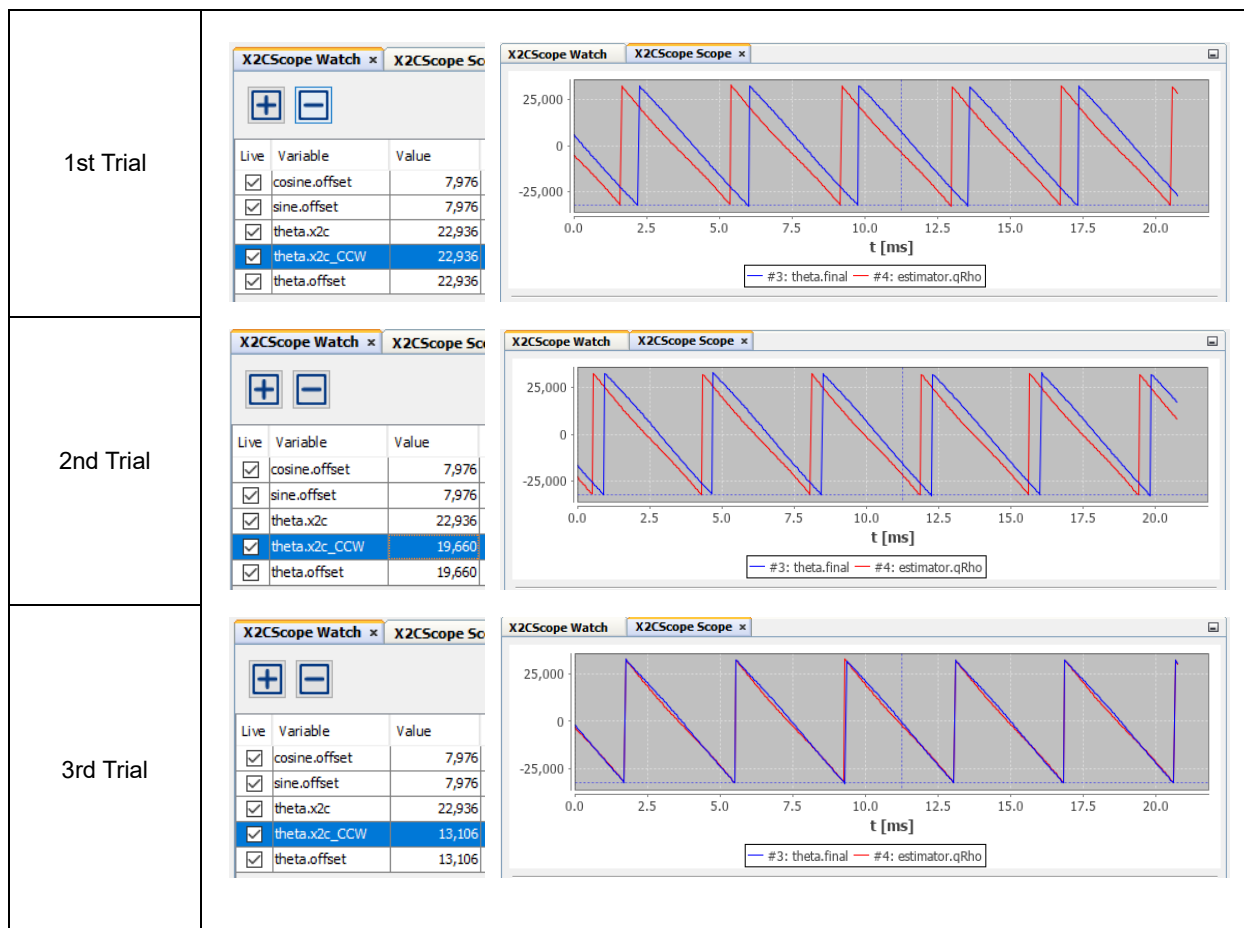
TABLE 1: THETA.X2C OFFSET ADJUSTMENT

1st Trial	<div><div>X2CScope Watch x X2CScope Scope</div><div><div><div><div>+</div><div>-</div></div><table><tr><td>Live</td><td>Variable</td><td>Value</td></tr><tr><td><input checked="" type="checkbox"/></td><td>cosine.offset</td><td>7,976</td></tr><tr><td><input checked="" type="checkbox"/></td><td>sine.offset</td><td>7,976</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.x2c</td><td>16,383</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.x2c_CCW</td><td>0</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.offset</td><td>16,383</td></tr></table></div></div><div><div>X2CScope Watch X2CScope Scope</div></div></div>	Live	Variable	Value	<input checked="" type="checkbox"/>	cosine.offset	7,976	<input checked="" type="checkbox"/>	sine.offset	7,976	<input checked="" type="checkbox"/>	theta.x2c	16,383	<input checked="" type="checkbox"/>	theta.x2c_CCW	0	<input checked="" type="checkbox"/>	theta.offset	16,383
Live	Variable	Value																	
<input checked="" type="checkbox"/>	cosine.offset	7,976																	
<input checked="" type="checkbox"/>	sine.offset	7,976																	
<input checked="" type="checkbox"/>	theta.x2c	16,383																	
<input checked="" type="checkbox"/>	theta.x2c_CCW	0																	
<input checked="" type="checkbox"/>	theta.offset	16,383																	
2nd Trial	<div><div>X2CScope Watch x X2CScope Scope</div><div><div><div><div>+</div><div>-</div></div><table><tr><td>Live</td><td>Variable</td><td>Value</td></tr><tr><td><input checked="" type="checkbox"/></td><td>cosine.offset</td><td>7,976</td></tr><tr><td><input checked="" type="checkbox"/></td><td>sine.offset</td><td>7,976</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.x2c</td><td>19,960</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.x2c_CCW</td><td>0</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.offset</td><td>19,960</td></tr></table></div></div><div><div>X2CScope Watch X2CScope Scope</div></div></div>	Live	Variable	Value	<input checked="" type="checkbox"/>	cosine.offset	7,976	<input checked="" type="checkbox"/>	sine.offset	7,976	<input checked="" type="checkbox"/>	theta.x2c	19,960	<input checked="" type="checkbox"/>	theta.x2c_CCW	0	<input checked="" type="checkbox"/>	theta.offset	19,960
Live	Variable	Value																	
<input checked="" type="checkbox"/>	cosine.offset	7,976																	
<input checked="" type="checkbox"/>	sine.offset	7,976																	
<input checked="" type="checkbox"/>	theta.x2c	19,960																	
<input checked="" type="checkbox"/>	theta.x2c_CCW	0																	
<input checked="" type="checkbox"/>	theta.offset	19,960																	
3rd Trial	<div><div>X2CScope Watch x X2CScope Scope</div><div><div><div><div>+</div><div>-</div></div><table><tr><td>Live</td><td>Variable</td><td>Value</td></tr><tr><td><input checked="" type="checkbox"/></td><td>cosine.offset</td><td>7,976</td></tr><tr><td><input checked="" type="checkbox"/></td><td>sine.offset</td><td>7,976</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.x2c</td><td>22,936</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.x2c_CCW</td><td>0</td></tr><tr><td><input checked="" type="checkbox"/></td><td>theta.offset</td><td>22,936</td></tr></table></div></div><div><div>X2CScope Watch X2CScope Scope</div></div></div>	Live	Variable	Value	<input checked="" type="checkbox"/>	cosine.offset	7,976	<input checked="" type="checkbox"/>	sine.offset	7,976	<input checked="" type="checkbox"/>	theta.x2c	22,936	<input checked="" type="checkbox"/>	theta.x2c_CCW	0	<input checked="" type="checkbox"/>	theta.offset	22,936
Live	Variable	Value																	
<input checked="" type="checkbox"/>	cosine.offset	7,976																	
<input checked="" type="checkbox"/>	sine.offset	7,976																	
<input checked="" type="checkbox"/>	theta.x2c	22,936																	
<input checked="" type="checkbox"/>	theta.x2c_CCW	0																	
<input checked="" type="checkbox"/>	theta.offset	22,936																	

## Assigning Value to the theta.offset for Reverse Direction

The same process used in adjusting the offset for the forward direction is followed to get the correct `theta.offset` for the reverse direction.

**TABLE 2: THETA.X2C.CCW OFFSET ADJUSTMENT**



## Offset Values Assignment

After obtaining all the offset values, the `ESTIMATOR` and `SETTING_OFFSET` definitions are disabled. The chosen values are assigned to their corresponding offset variables in `lx_sensor.c` file as shown in the following figure.

**FIGURE 19: OFFSET VALUES ASSIGNMENT**

```

sine.offset = 7976;
cosine.offset = 7976;

if(mcappData.runDirection == CW)
{
    theta.offset = 22936;
}
else
{
    theta.offset = 13106;
}

```

The position and velocity estimation are processed now by the inductive position algorithm. Program the device and start running the motor with the LX34050 inductive position algorithm.

## Setting the Offset Using Manual Method

For setting up the firmware initially, enable the `OPEN_LOOP_FUNCTIONING` and `SETTING_OFFSET` defines in the `userparams.h` file as shown in the figure below. When `OPEN_LOOP_FUNCTIONING` define is active, the motor is driven in force commutation mode with fixed speed.

**FIGURE 20: USERPARAMS.H DEFINITIONS FOR SETTING THE SIN AND COS OFFSET**

```
#define OPEN_LOOP_FUNCTIONING
#define SETTING_OFFSET
```

Refer to the previous section where PLL estimator is used, on how to calculate and obtain the sine and cosine offsets. Similar process is performed to get the offsets, the only difference is the open loop motor drive using force commutation. After finding the sine and cosine offsets, assign these values to the corresponding variables (`sin.x2c` and `cos.x2c`). The next step is finding the `theta.offset` for the clockwise direction first, before finding for the counterclockwise.

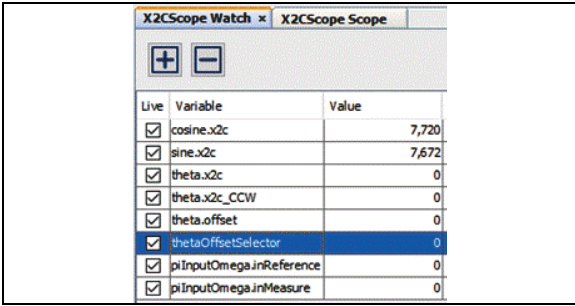
Disable the `OPEN_LOOP_FUNCTIONING` mode but retain the `SETTING_OFFSET` active as shown in the following figure. Build the code and load to the device.

**FIGURE 21: USERPARAMS.H DEFINITIONS FOR SETTING THE THETA OFFSET**

```
//#define OPEN_LOOP_FUNCTIONING
#define SETTING_OFFSET
```

Disabling the open loop mode will allow the drive to vary the speed using the potentiometer on the board. In the X2CScope watch window, add the `thetaOffsetSelector`, `theta.x2c`, `theta.x2c_CCW`, `piInputOmega.inReference` and `piInputOmega.inMeasure` variables as shown in the figure above. The value of `thetaOffsetSelector` is used as an index of `Theta_Offset[]` lookup table, which contains the predefined theta offset values ranging from -32768 to 32767 and subdivided into 42. An excel sheet file used as a guide to easily locate the correct theta offset value is included with the code. The `theta.x2c` is used for tuning the theta offset in clockwise direction while `theta.x2c_CCW` is used in counterclockwise. The correct theta offset will give a smooth drive across the speed range. The `piInputOmega.inReference` and `piInputOmega.inMeasure` are used for confirming if the motor spins correctly across the speed range.

**FIGURE 22: VARIABLES FOR THETA OFFSET SETTING**



Live	Variable	Value
<input checked="" type="checkbox"/>	<code>cosine.x2c</code>	7,720
<input checked="" type="checkbox"/>	<code>sine.x2c</code>	7,672
<input checked="" type="checkbox"/>	<code>theta.x2c</code>	0
<input checked="" type="checkbox"/>	<code>theta.x2c_CCW</code>	0
<input checked="" type="checkbox"/>	<code>theta.offset</code>	0
<input checked="" type="checkbox"/>	<code>thetaOffsetSelector</code>	0
<input checked="" type="checkbox"/>	<code>piInputOmega.inReference</code>	0
<input checked="" type="checkbox"/>	<code>piInputOmega.inMeasure</code>	0

Implement the following steps in finding for the `theta.offset`:

- Assign a value to the `thetaOffsetSelector` from the attached Excel sheet file before running the motor. Make sure that the potentiometer is at the minimum.
- Run the motor and observe. If the corresponding theta offset selected caused the motor to spin at a speed far from the reference or the motor stalled, then the theta offset is wrong. Stop the motor by toggling SW1.
- Enter another value of `thetaOffsetSelector` before running the motor again. Try other values of `thetaOffsetSelector` in step of 5 to get the correct theta offset faster.
- The correct theta offset will spin the motor at its minimum speed because the potentiometer is also at the minimum. Sweep the value of the potentiometer from minimum to maximum by rotating.
- The appropriate theta offset will give a smooth drive across the speed range. To finely tune the theta offset, the user may try two offset values before and after the theta offset value where the motor spins at the minimum reference speed.
- Monitor the `theta.x2c` to confirm the value of theta offset in the clockwise direction and `theta.x2c_CCW` in the counterclockwise direction.
- The theta offset in the counterclockwise direction is usually seven steps lower in magnitude than the offset in the clockwise direction. For example, the correct theta offset for clockwise is -11468, obtained when `thetaOffsetSelector` value is 29, then the theta offset in CCW is -22937 with `thetaOffsetSelector` value of 36.
- After successfully obtaining the correct theta offset, assign the values in the corresponding variables within the `LX_EstimAngle()` function written in the `lx_sensor.c` source file. Disable both the `OPEN_LOOP_FUNCTIONING` and `SETTING_OFFSET` defines in the `userparams.h` file. Build the project, load to the device, and run the system using the LX sensor feedback. Theta offset setting is recommended every time the con-

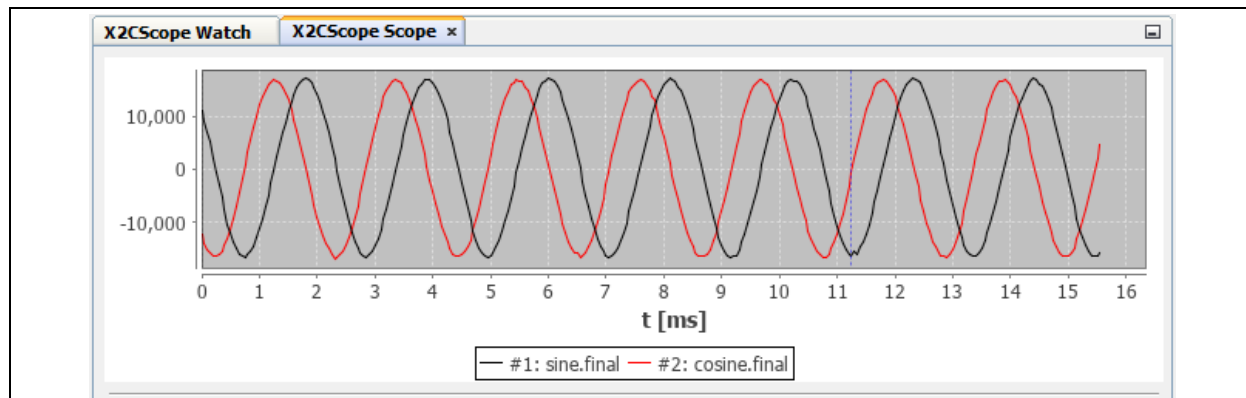
ductor target placement in the rotor is changed.

## RESULTS

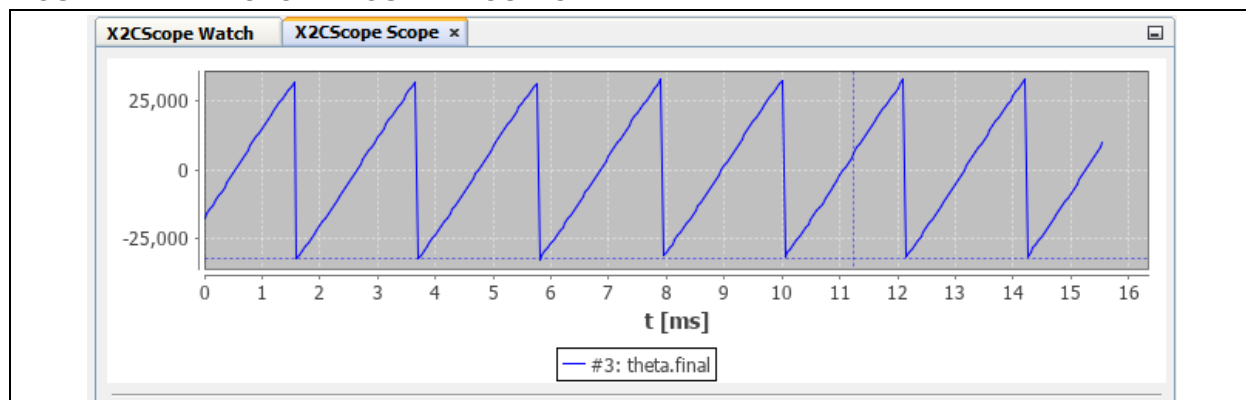
This section presents the results of testing the LX34050 sensored FOC drive at no load condition. This illustrates the sine and cosine signals, the rotor

angular position and the SVM line-to-ground voltages. The diagrams and graphs are captured using X2CScope.

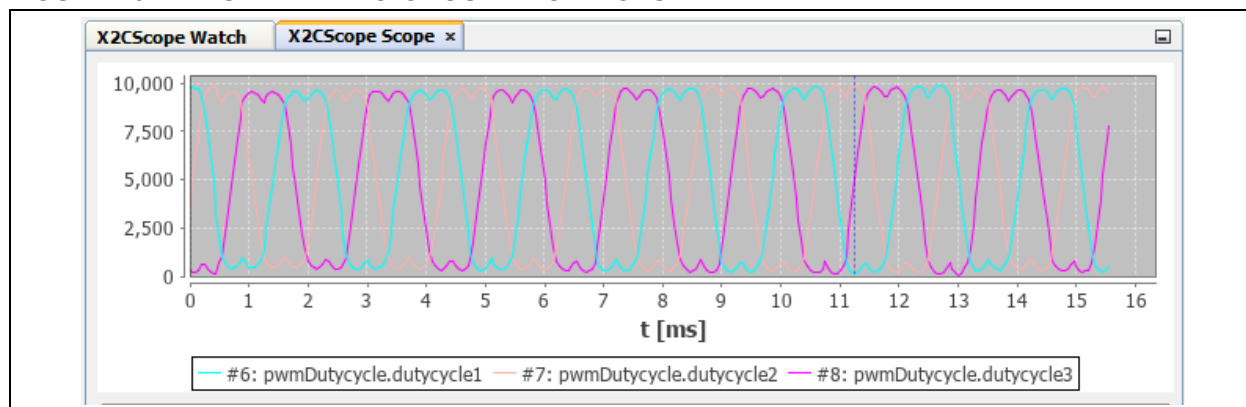
**FIGURE 23: SINE AND COSINE SIGNALS**



**FIGURE 24: ROTOR ANGULAR POSITION**



**FIGURE 25: SVM LINE-TO-GROUND VOLTAGES**



## CONCLUSION

This document describes the use of an inductive position sensor as feedback for providing accurate rotor position in the FOC drive technique. The LX34050 is proven effective for interfacing to and managing printed circuit board (PCB)-based inductive position sensors. The 16-bit Microchip dsPIC33CK DSC efficiently used its DSP for executing the FOC algorithm and its on-chip peripherals for a cost-effective solution. The X2CScope is very useful for visualizing and debugging the signals coming into and going out of the DSC. Overall, this application can be used as a pattern for the ease of adapting the use of the LX34050 inductive position sensor to other motors.



## APPENDIX A: CIRCUIT SCHEMATICS

FIGURE A-1: LX34050 SENSOR BOARD SCHEMATICS

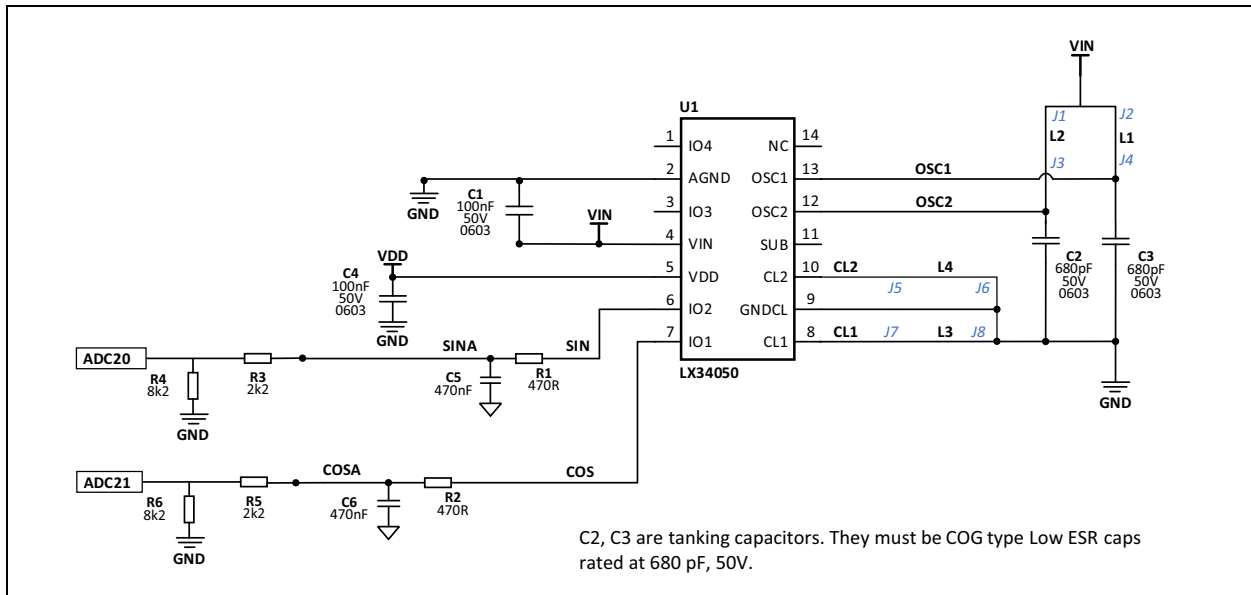
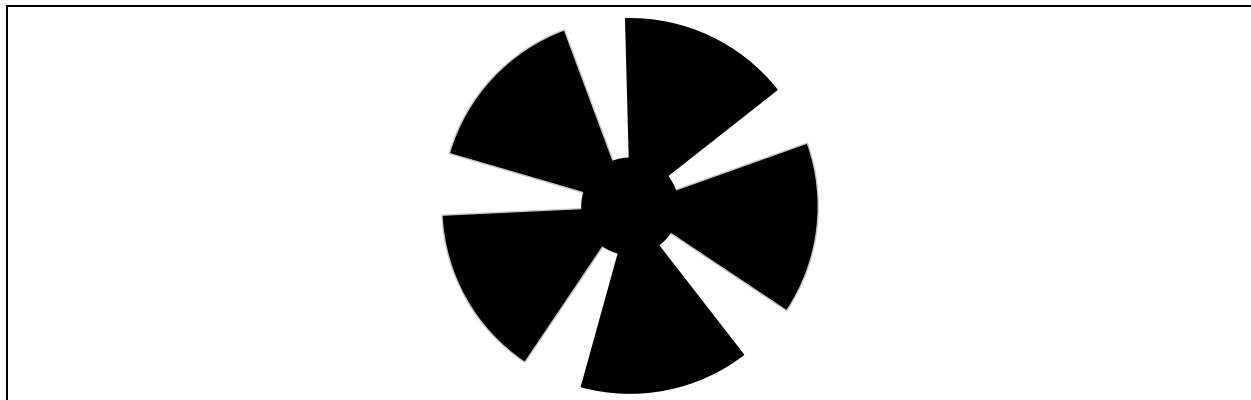
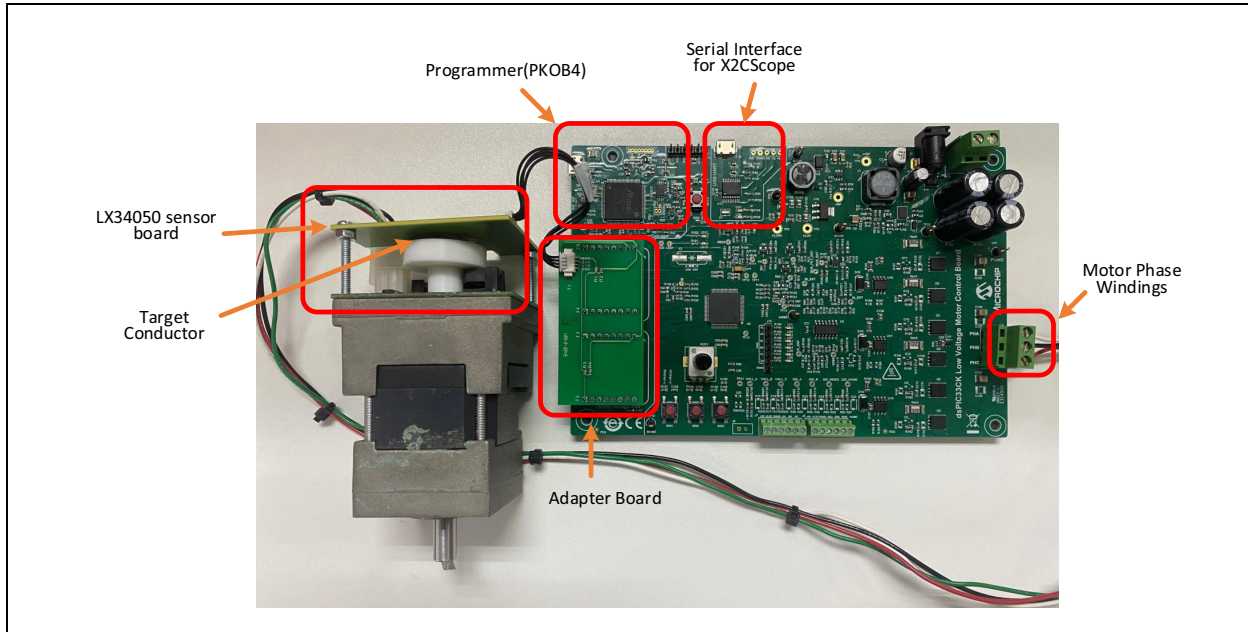


FIGURE A-2: TARGET CONDUCTOR PCB LAYOUT



**FIGURE A-3: ACTUAL SETUP**



## APPENDIX B: MCC SYSTEM MODULE AND PERIPHERAL CONFIGURATION

MPLAB® Code Configurator is an easy-to-use plugin tool for MPLAB X IDE that generates codes for controlling the peripherals of Microchip microcontrollers and DSCs, based on the settings made in its Graphical User Interface (GUI). MPLAB Code Configurator is used to easily configure the peripherals used in this motor control application. Refer to the MPLAB Code Configurator User's Guide (DS40001725) for further information on how to install and set up the MCC in MPLAB X IDE.

The step-by-step process of using MCC in this motor control application is enumerated in this section.

### B.1 MCC Initialization

1. In the System Module tab, set the clock to FRC Oscillator with 8 MHz frequency. Disable the FRC Postscaler and enable the PLL with Prescaler of 1:1, Feedback of 1:150, Postscaler1 of 1:3 and Postscaler2 of 1:1. The resulting frequency (FOSC) is 200 MHz with instruction cycle (FOSC/2) of 100 MHz.
2. Set the ADC1 module clock source to FOSC/2, with the target shared core sampling time of 250 ns. Select the channels and configure as shown in the following figure.

FIGURE B-1: ADC MODULE

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Core0	<input checked="" type="checkbox"/>	AN0	RA0	CH_AN0_IA	PWM1 Trigger1	None	<input type="checkbox"/>
Core1	<input checked="" type="checkbox"/>	AN1	RB2	CH_AN1_IB	PWM1 Trigger1	None	<input checked="" type="checkbox"/>
Shared	<input checked="" type="checkbox"/>	AN11	RB9	CH_AN11_POT	PWM1 Trigger1	None	<input type="checkbox"/>
Shared	<input checked="" type="checkbox"/>	AN20	RE0	CH_AN20_SIN	PWM1 Trigger1	None	<input type="checkbox"/>
Shared	<input checked="" type="checkbox"/>	AN21	RE1	CH_AN21_COS	PWM1 Trigger1	None	<input type="checkbox"/>

3. In the PWM Generator module, select the Master Clock as FOSC - System Clock with Clock Divided Frequency of 100 MHz. Select the required generators: PG1, PG2 and PG4. Configure the PWM Master settings with PWM Input Clock Selection of 200 MHz, PWM Operation mode: Center-Aligned and complementary PWM output mode. The master PWM period is set to 10 kHz. The PWMs are initialized to 0% duty cycle. PG1 is configured with a self-trigger every start of cycle and trigger output at EOC event. The update triggers every time the duty cycle is written, and the update mode is every SOC. PG2 and PG4 are configured with the trigger coming from the PWM1 and trigger output selection at the EOC event. The update triggers every time the duty cycle is written, and the update mode is Slaved SOC. The dead time is set to 500 ns for all the PWM generators.
4. The internal OP-AMPs are initialized as illustrated in the following figure.

FIGURE B-2: OP-AMP CONFIGURATION

5. Configure the UART settings like shown in the following figure.

FIGURE B-3: UART SETTINGS

6. In the PIN MANAGER configuration, set up the input/output pins of all the peripherals as shown in the following figure.

\_\_\_\_\_

- DS00004764A-page 20 © 2022 Microchip Technology Inc. and its subsidiaries

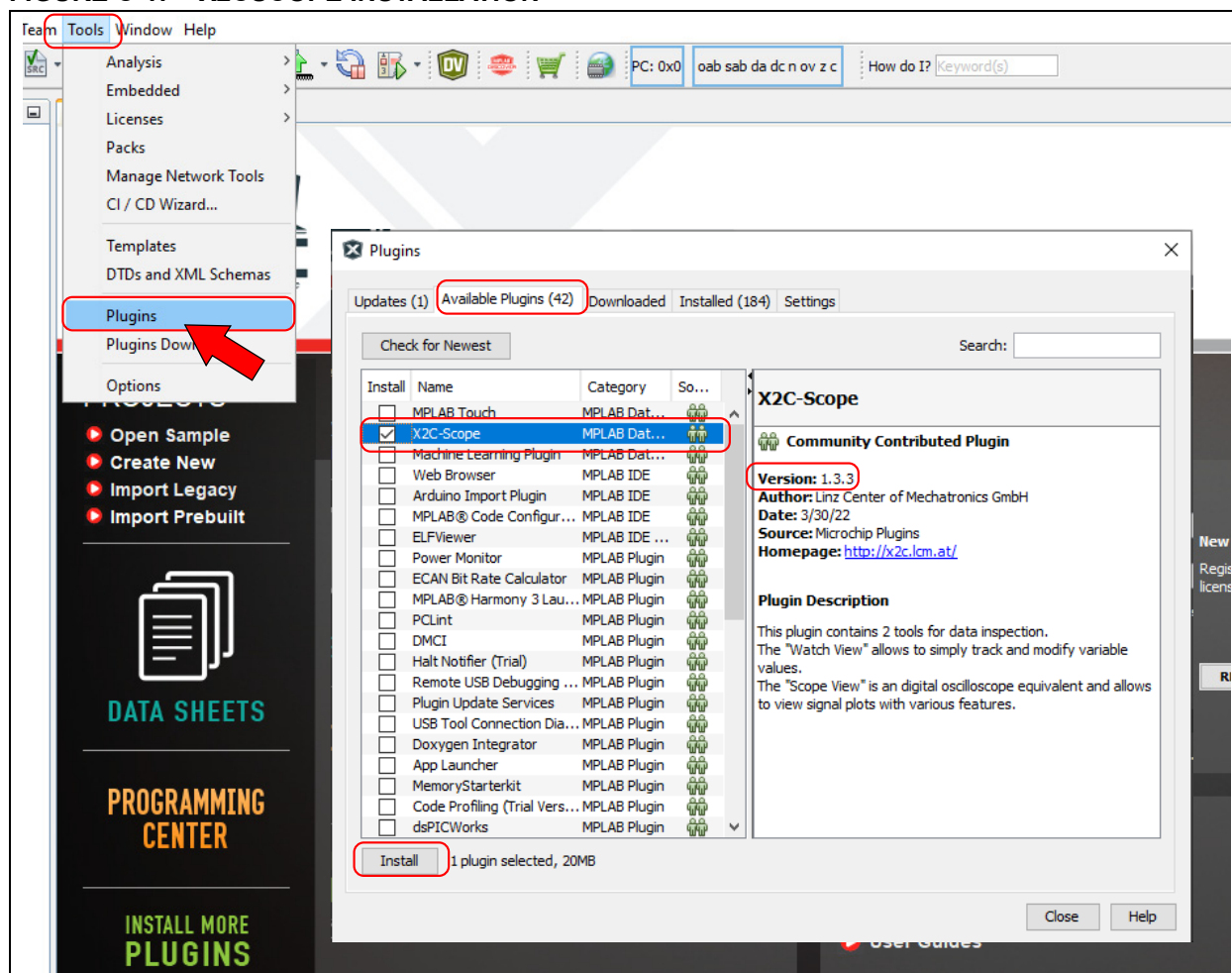
## APPENDIX C: X2CScope

X2C Scope is an MPLAB X IDE plug-in that allows a developer to interact with an application while the application program is running. X2C Scope enables the user to watch, update and plot global variables (for motor control) in real time. It communicates with the target DSC using the UART. To use X2C, the plug-in must be installed by following the instructions:

- In MPLAB X IDE, go to Tools>Plugins, then click on the Available Plugins tab.
- Select X2C Scope plug-in by ticking its check box, then click Install.

The X2CScope window is opened through clicking the Tools>Embedded>X2CScope.

**FIGURE C-1: X2CScope INSTALLATION**



## APPENDIX D: SOURCE CODE LISTING

The latest software version can be downloaded from the Microchip MPLAB Discover website (<https://mplab-discover.microchip.com/>). The user will find the source code appended to the IDE example of this application note. The latest version is v1.0.

## THE MICROCHIP WEBSITE

Microchip provides online support via our WWW site at [www.microchip.com](http://www.microchip.com). This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at [www.microchip.com](http://www.microchip.com). Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the website at: <http://microchip.com/support>**

NOTES:



---

**Note the following details of the code protection feature on Microchip products:**

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
  - Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
  - Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
  - Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable" Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.
- 

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at <https://www.microchip.com/en-us/support/design-help/client-support-services>.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

For information regarding Microchip's Quality Management Systems, please visit [www.microchip.com/quality](http://www.microchip.com/quality).

**Trademarks**

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Klear, LANCheck, LinkMD, maxStylus, maxTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, GridTime, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, KoD, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2022, Microchip Technology Incorporated and its subsidiaries.

All Rights Reserved.

ISBN: 978-1-6683-1291-9

## Worldwide Sales and Service

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://www.microchip.com/support>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Austin, TX**  
Tel: 512-257-3370

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Novi, MI  
Tel: 248-848-4000

**Houston, TX**  
Tel: 281-894-5983

**Indianapolis**  
Noblesville, IN  
Tel: 317-773-8323  
Fax: 317-773-5453  
Tel: 317-536-2380

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608  
Tel: 951-273-7800

**Raleigh, NC**  
Tel: 919-844-7510

**New York, NY**  
Tel: 631-435-6000

**San Jose, CA**  
Tel: 408-735-9110  
Tel: 408-436-4270

**Canada - Toronto**  
Tel: 905-695-1980  
Fax: 905-695-2078

### ASIA/PACIFIC

**Australia - Sydney**  
Tel: 61-2-9868-6733

**China - Beijing**  
Tel: 86-10-8569-7000

**China - Chengdu**  
Tel: 86-28-8665-5511

**China - Chongqing**  
Tel: 86-23-8980-9588

**China - Dongguan**  
Tel: 86-769-8702-9880

**China - Guangzhou**  
Tel: 86-20-8755-8029

**China - Hangzhou**  
Tel: 86-571-8792-8115

**China - Hong Kong SAR**  
Tel: 852-2943-5100

**China - Nanjing**  
Tel: 86-25-8473-2460

**China - Qingdao**  
Tel: 86-532-8502-7355

**China - Shanghai**  
Tel: 86-21-3326-8000

**China - Shenyang**  
Tel: 86-24-2334-2829

**China - Shenzhen**  
Tel: 86-755-8864-2200

**China - Suzhou**  
Tel: 86-186-6233-1526

**China - Wuhan**  
Tel: 86-27-5980-5300

**China - Xian**  
Tel: 86-29-8833-7252

**China - Xiamen**  
Tel: 86-592-2388138

**China - Zhuhai**  
Tel: 86-756-3210040

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444

**India - New Delhi**  
Tel: 91-11-4160-8631

**India - Pune**  
Tel: 91-20-4121-0141

**Japan - Osaka**  
Tel: 81-6-6152-7160

**Japan - Tokyo**  
Tel: 81-3-6880-3770

**Korea - Daegu**  
Tel: 82-53-744-4301

**Korea - Seoul**  
Tel: 82-2-554-7200

**Malaysia - Kuala Lumpur**  
Tel: 60-3-7651-7906

**Malaysia - Penang**  
Tel: 60-4-227-8870

**Philippines - Manila**  
Tel: 63-2-634-9065

**Singapore**  
Tel: 65-6334-8870

**Taiwan - Hsin Chu**  
Tel: 886-3-577-8366

**Taiwan - Kaohsiung**  
Tel: 886-7-213-7830

**Taiwan - Taipei**  
Tel: 886-2-2508-8600

**Thailand - Bangkok**  
Tel: 66-2-694-1351

**Vietnam - Ho Chi Minh**  
Tel: 84-28-5448-2100

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4485-5910  
Fax: 45-4485-2829

**Finland - Espoo**  
Tel: 358-9-4520-820

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Garching**  
Tel: 49-8931-9700

**Germany - Haan**  
Tel: 49-2129-3766400

**Germany - Heilbronn**  
Tel: 49-7131-72400

**Germany - Karlsruhe**  
Tel: 49-721-625370

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Germany - Rosenheim**  
Tel: 49-8031-354-560

**Israel - Ra'anana**  
Tel: 972-9-744-7705

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Italy - Padova**  
Tel: 39-049-7625286

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Norway - Trondheim**  
Tel: 47-7288-4388

**Poland - Warsaw**  
Tel: 48-22-3325737

**Romania - Bucharest**  
Tel: 40-21-407-87-50

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**Sweden - Gothenberg**  
Tel: 46-31-704-60-40

**Sweden - Stockholm**  
Tel: 46-8-5090-4654

**UK - Wokingham**  
Tel: 44-118-921-5800  
Fax: 44-118-921-5820