
AN1078 Tuning Procedure ReadMe

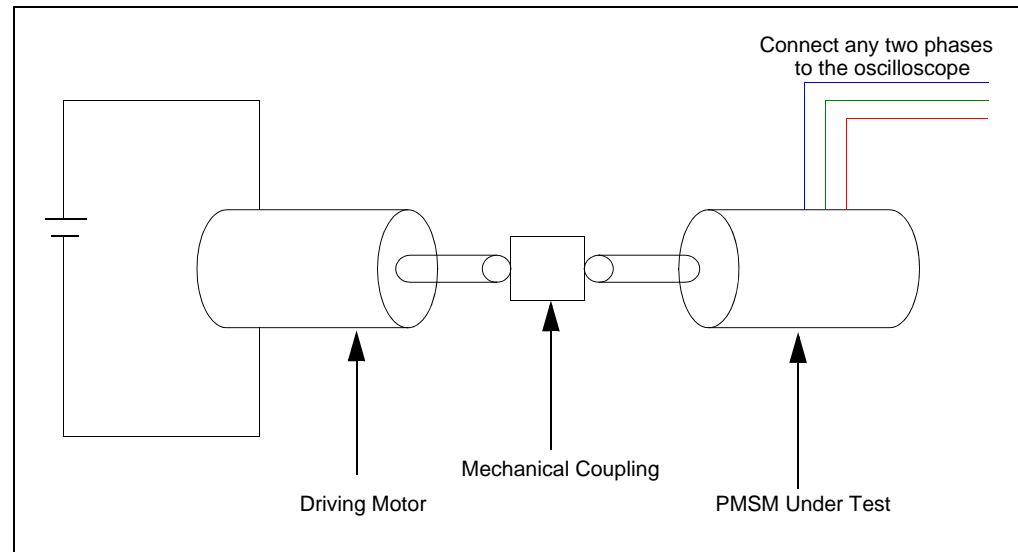
This document describes the procedure and setup necessary for tuning a PMSM motor using the FOC algorithm described in AN1078 “*Sensorless Field Oriented Control of PMSM*” (DS01078). Due to differences between various motors, this algorithm needs to be tuned to every new motor model.

1.1 KNOW YOUR PMSM

Before running a motor with the FOC algorithm, the user must determine whether their motor can be supported. The FOC algorithm is designed to run only on a PMSM with sinusoidal shape back-EMF.

Figure 1-1 shows the graphical representation of the setup for checking the back-EMF of a PMSM. It consists of a PMSM under test coupled to another driving motor. To observe the shape of back-EMF, run the driving motor up to a fixed speed (for example, 2000 RPM), and check the waveform of back-EMF across the two phases on the oscilloscope.

FIGURE 1-1: CHECKING BACK-EMF OF A MOTOR



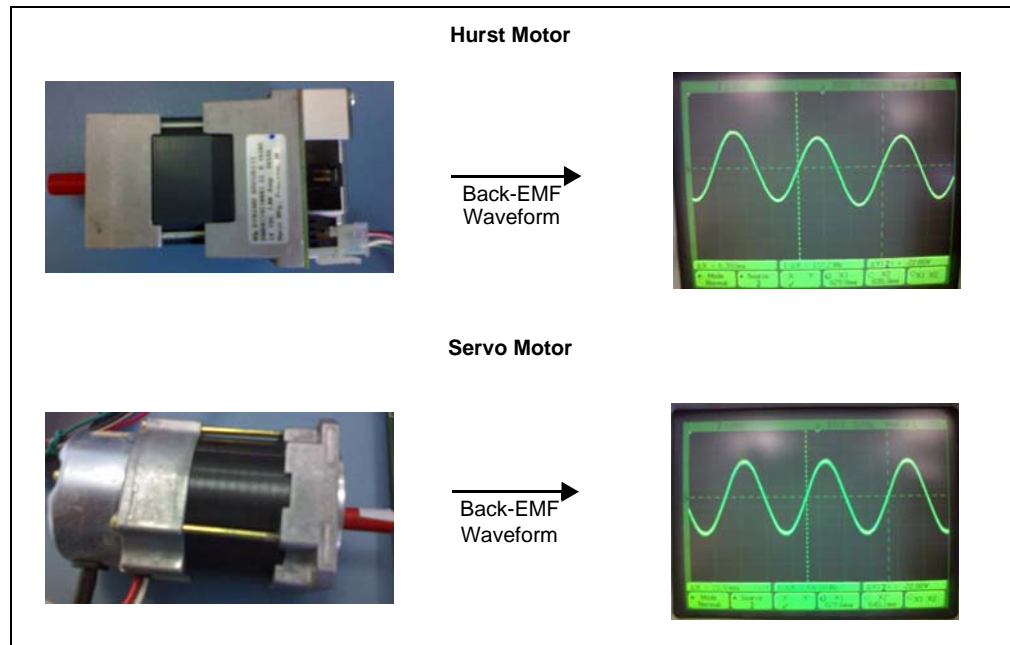
1.2 MOTORS TESTED SUCCESSFULLY ON THE dsPICDEM™ MCLV DEVELOPMENT BOARD

The following motors were tested successfully on the dsPICDEM MCLV Development Board:

- Hurst motor (Part number: AC300020 - available at www.microchipdirect.com)
The Hurst motor along with its back-EMF waveform is shown in Figure 1-2. The rating of the motor is 24V, 5-pole pair, and 2500 RPM. For additional technical specifications of this motor, visit www.myhurst.com.
- Servo motor
The Servo motor along with its back-EMF waveform is shown in Figure 1-2. The rating of the motor is 24V, 2-pole pair, and 3000 RPM. For additional technical specifications of this motor, visit www.shinano.com.

These motors, which were tested successfully with the FOC algorithm on the dsPICDEM MCLV Development Board are PMSM with sinusoidal back-EMF.

FIGURE 1-2: HURST AND SERVO MOTOR BEMF WAVEFORMS



AN1078 Tuning Procedure ReadMe

1.3 MOTORS TESTED SUCCESSFULLY ON THE dsPICDEM™ MCHV DEVELOPMENT BOARD

The following motors were tested successfully on the dsPICDEM MCHV Development Board:

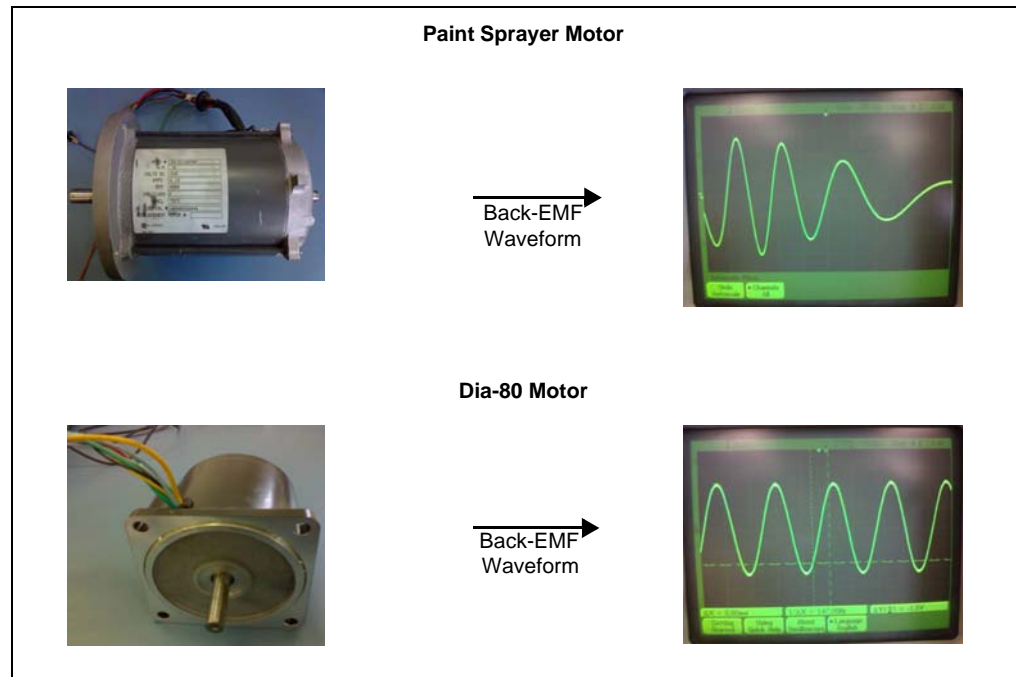
- Paint sprayer motor (custom motor)

The motor of a paint spray application along with its back-EMF waveform is shown in Figure 1-3. The rating of the motor is 160 VDC, 3-pole pair, and 4000 RPM.

- Dia-80 motor

The Dia-80 motor along with its back-EMF waveform is shown in Figure 1-3. The rating of the motor is 220V, 2-pole pair, and 3500 RPM. For additional technical specifications of this motor, visit www.eletechnic.com.

FIGURE 1-3: PAINT SPRAYER AND DIA-80 MOTOR BEMF WAVEFORMS



- Dia-YS50 motor

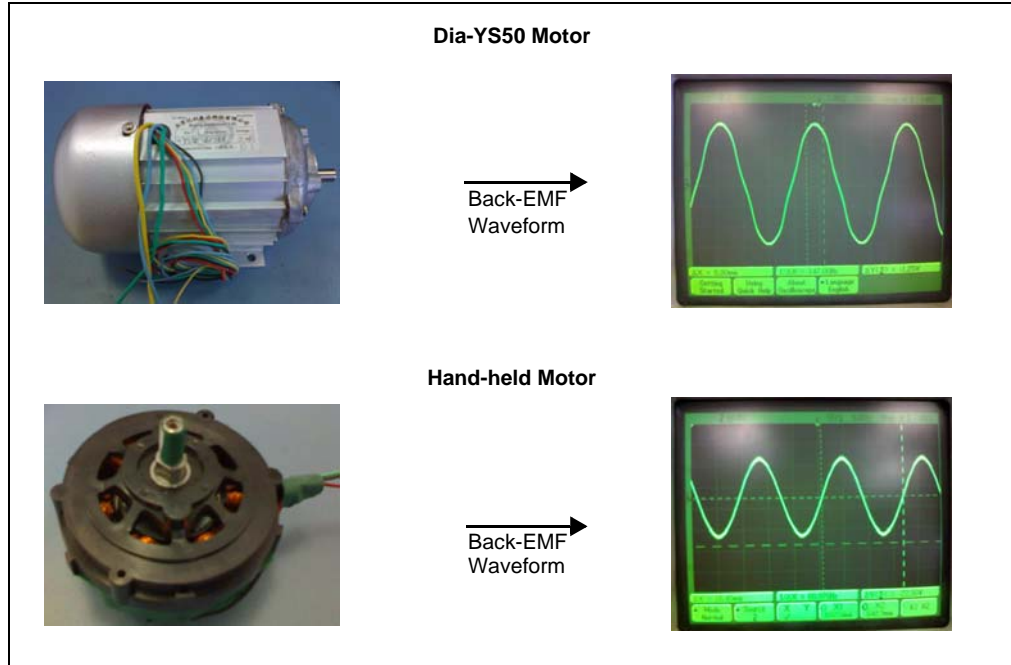
The Dia-YS50 motor along with its back-EMF waveform is shown in Figure 1-4. The rating of the motor is 220V RMS, 2-pole pair, and 4000 RPM.

- Professional hand-held tool motor

The second motor shown in Figure 1-4 is of a hand-held motor application along with its back-EMF waveform. The rating of the motor is 120V RMS, 2-pole pair, and 17,000 RPM.

Each of these motors, which were tested successfully with the FOC algorithm on the dsPICDEM MCHV Development Board, are PMSMs with sinusoidal back-EMF.

FIGURE 1-4: DIA-YS50 AND HAND-HELD MOTOR BEMF WAVEFORMS



1.4 MOTORS NOT RUNNING SATISFACTORILY WITH THE FOC ALGORITHM

This section describes motors which did not run satisfactorily with the FOC algorithm.

Figure 1-5 shows the waveforms of motors that are trapezoidal in shape. The Brushless Direct Current (BLDC) motors with trapezoidal waveform may not run satisfactorily with the FOC algorithm. These motors do not have sinusoidal back-EMF and therefore, they may not run up to the rated speed or may not lock for closed loop operation.

FIGURE 1-5: TRAPEZOIDAL BACK-EMF WAVEFORMS OF MOTORS

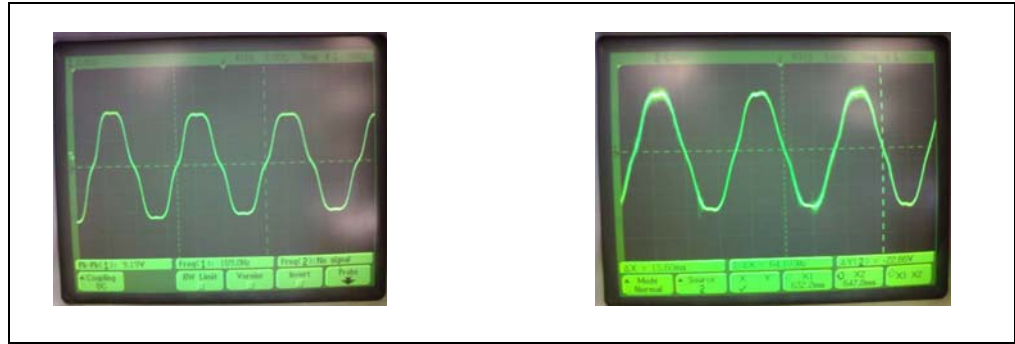
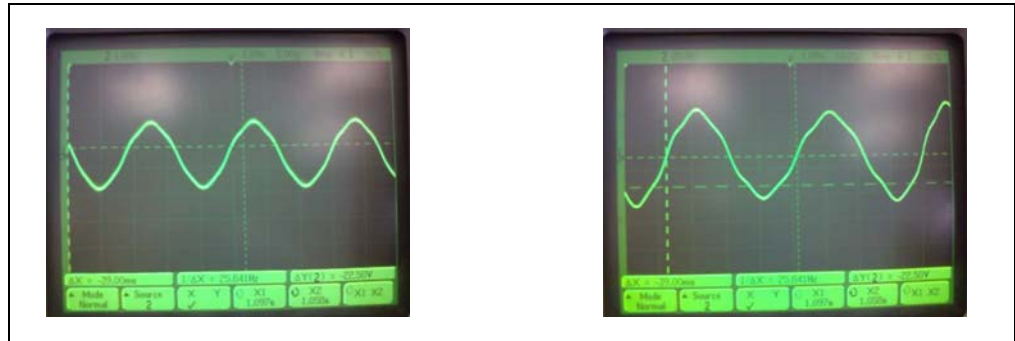


Figure 1-6 shows the waveforms of motors that are non-sinusoidal in shape. The BLDC motors with non-sinusoidal waveform may not run satisfactorily with the FOC algorithm.

FIGURE 1-6: NON-SINUSOIDAL BACK-EMF WAVEFORMS OF MOTORS



1.5 SETTING HARDWARE PARAMETERS

The hardware parameters: RSHUNT, DIFFAMPGAIN and VDD are located in the UserParms.h file. The userParms.h file contains the parameters that change based on the hardware design. Example 1-1 shows the hardware parameter settings for the dsPICDEM MCLV and MCHV Development Boards.

EXAMPLE 1-1: SETTING HARDWARE PARAMETERS

dsPICDEM™ MCLV Board

```

//***** Hardware Parameters *****/
#define RSHUNT          0.005 // Value in Ohms of shunt resistors used.
#define DIFFAMPGAIN     75    // Gain of differential amplifier.
#define VDD              3.3  // VDD voltage, only used to convert torque
                               // reference from Amps to internal variables
    
```

dsPICDEM™ MCHV Board

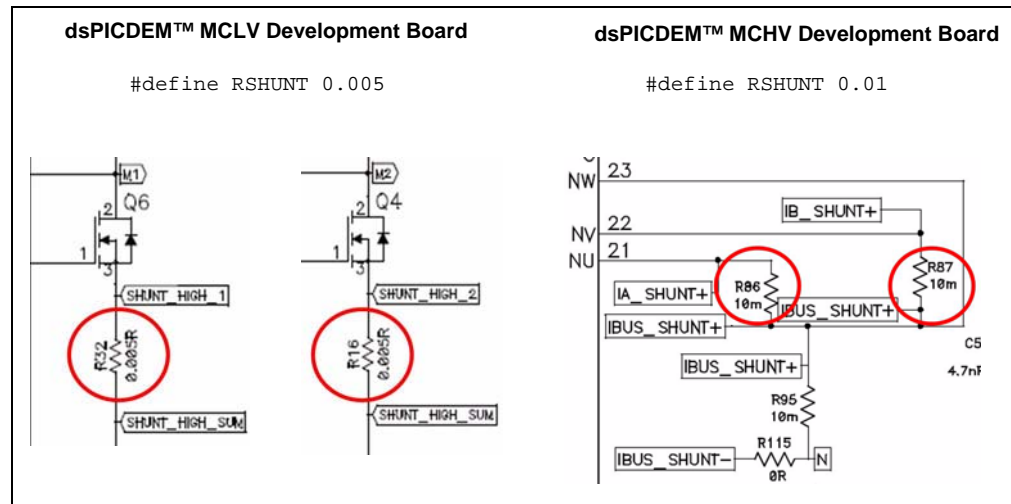
```

//***** Hardware Parameters *****/

#define RSHUNT          0.01  // Value in Ohms of shunt resistors used.
#define DIFFAMPGAIN     10    // Gain of differential amplifier.
#define VDD              3.3  // VDD voltage, only used to convert torque
                               // reference from Amps to internal variables
    
```

Figure 1-7 shows the shunt connections in the dsPICDEM MCLV and MCHV Development Boards, and the values used in the FOC algorithm. For dsPICDEM MCLV and MCHV Development Board schematics, refer to the “dsPICDEM™ MCLV Development Board User’s Guide” (DS70331) and the “dsPICDEM™ MCHV Development System User’s Guide” (DS70605), respectively.

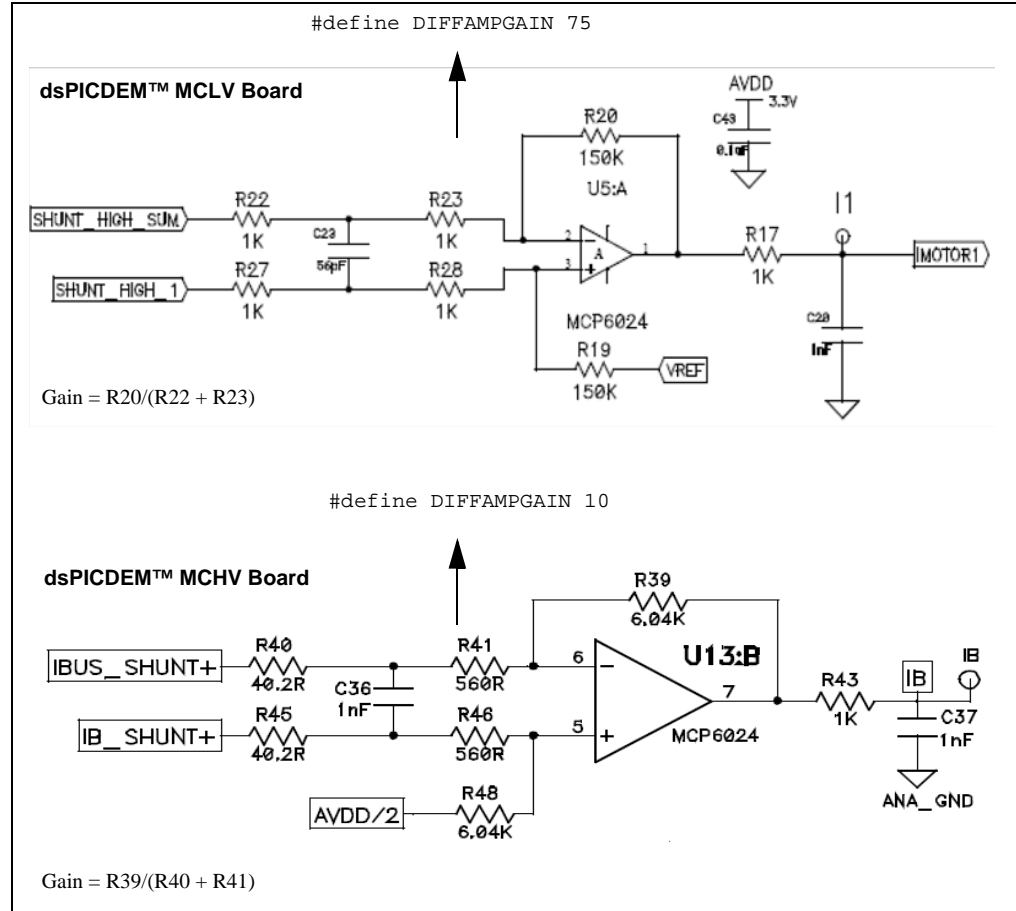
FIGURE 1-7: SHUNT CONNECTIONS



AN1078 Tuning Procedure ReadMe

The operational amplifier is used to amplify the current signal from the shunt. Based on the value of the amplifier gain set in the hardware, the correct gain values should be entered for the `DIFFAMPAIN` parameter in the `UserParms.h` file, as shown in Figure 1-8.

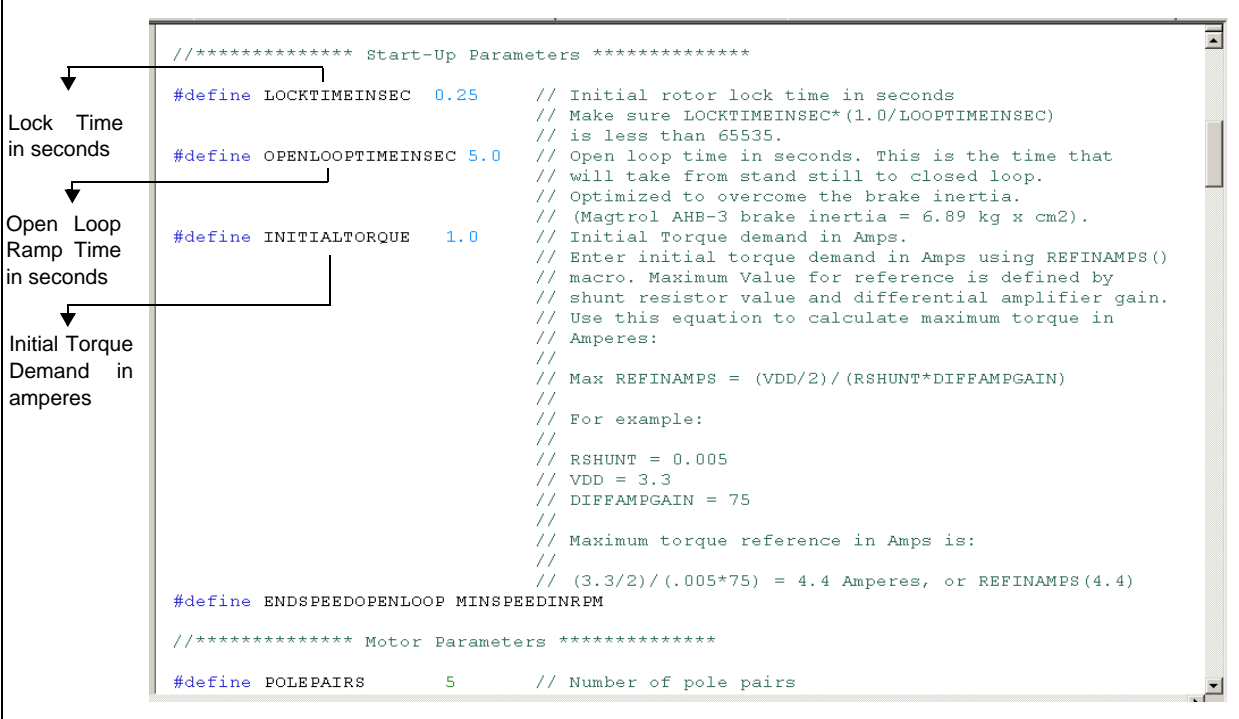
FIGURE 1-8: GAIN CALCULATION FOR DIFFERENTIAL AMPLIFIER



1.6 SETTING START-UP PARAMETERS

The start-up parameter values are motor specific and are dependent on the inertia of motor, friction, and load torque. The user must fine-tune these values to run the motor satisfactorily. Example 1-2 shows the start-up parameter settings.

EXAMPLE 1-2: SETTING START-UP PARAMETERS



The image shows a code editor window with a light gray background and a dark border. The code is in C preprocessor style, defining various parameters for a motor's start-up. On the left side of the editor, there are three text labels with arrows pointing to specific lines of code: 'Lock Time in seconds' points to the LOCKTIMEINSEC definition, 'Open Loop Ramp Time in seconds' points to the OPENLOOPTIMEINSEC definition, and 'Initial Torque Demand in amperes' points to the INITIALTORQUE definition. The code itself is as follows:

```

//***** Start-Up Parameters *****
#define LOCKTIMEINSEC 0.25 // Initial rotor lock time in seconds
// Make sure LOCKTIMEINSEC*(1.0/LOOPTIMEINSEC)
// is less than 65535.
#define OPENLOOPTIMEINSEC 5.0 // Open loop time in seconds. This is the time that
// will take from stand still to closed loop.
// Optimized to overcome the brake inertia.
// (Magtrol AHB-3 brake inertia = 6.89 kg x cm2).
#define INITIALTORQUE 1.0 // Initial Torque demand in Amps.
// Enter initial torque demand in Amps using REFINAMPS()
// macro. Maximum Value for reference is defined by
// shunt resistor value and differential amplifier gain.
// Use this equation to calculate maximum torque in
// Amperes:
//
// Max REFINAMPS = (VDD/2)/(RSHUNT*DIFFAMPAIN)
//
// For example:
//
// RSHUNT = 0.005
// VDD = 3.3
// DIFFAMPAIN = 75
//
// Maximum torque reference in Amps is:
//
// (3.3/2)/(0.005*75) = 4.4 Amperes, or REFINAMPS(4.4)
#define ENDSPEEDOPENLOOP MINSPEEDINRPM

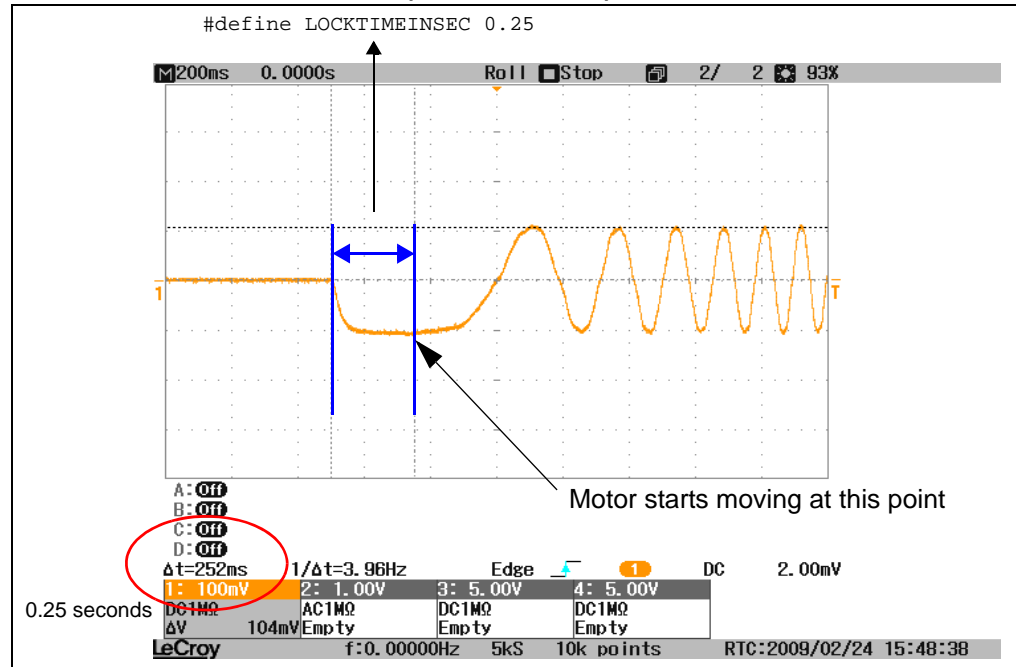
//***** Motor Parameters *****
#define POLEPAIRS 5 // Number of pole pairs

```


AN1078 Tuning Procedure ReadMe

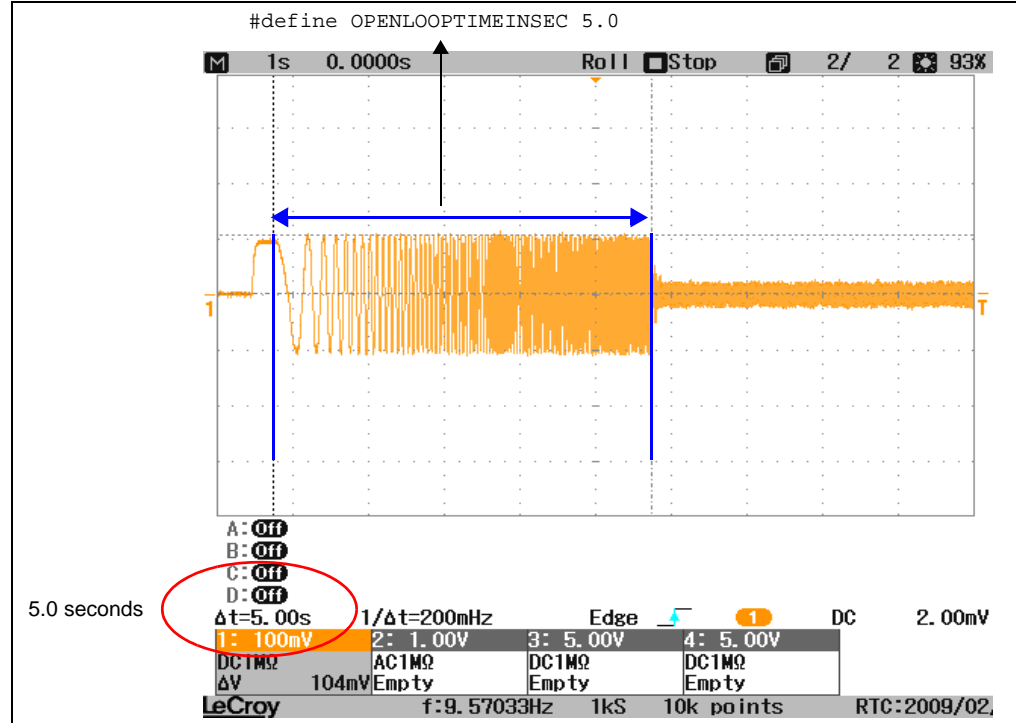
Figure 1-9 through Figure 1-11 show the oscilloscope capture of start-up parameters.

FIGURE 1-9: LOCK TIME (0.25 SECONDS)



The lock time should be sufficient for the motor to lock and stabilize. The rotor should not oscillate at the end of the lock time; if it oscillates, increase the lock time.

FIGURE 1-10: OPEN LOOP TIME (5.0 SECONDS)



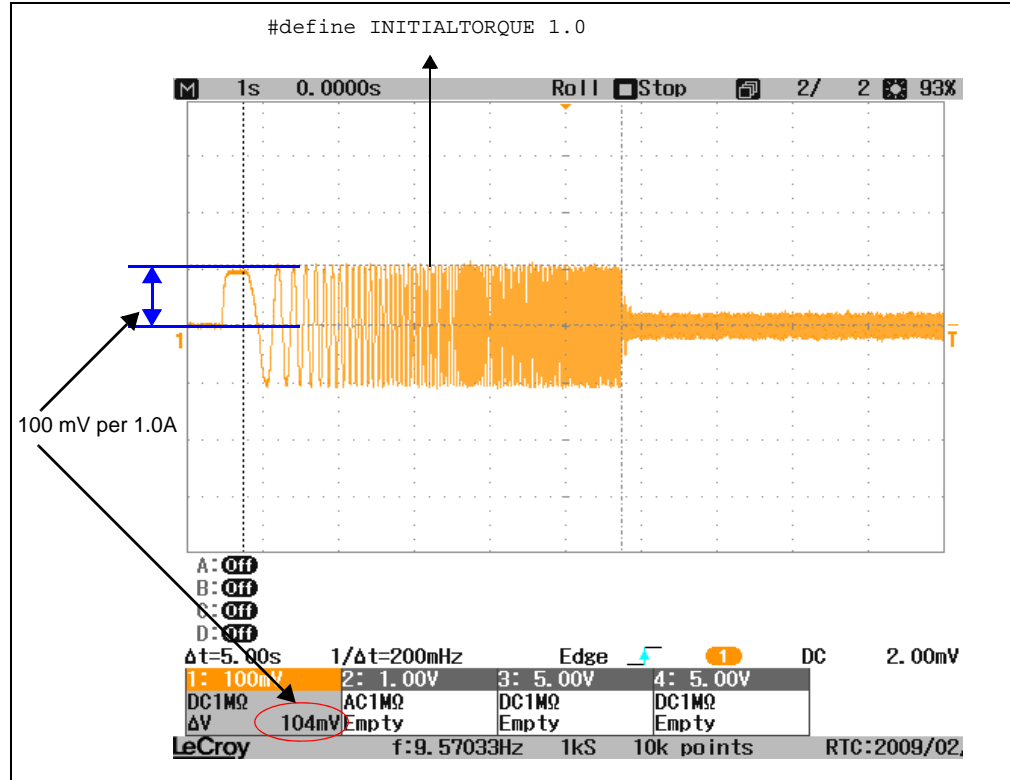
The open loop time should be long enough, so that the rotor follows the stator commutation until the end speed in open loop (MINSPEEDINRPM) is reached. If it is not reached, increase the open loop time.

Set the ramp time to a greater value, so that the rotor can catch-up with the rotating stator flux. The ramp time needs to be adjusted while running the motor under a loaded condition.

Setting the initial torque demand value lower than the required value will stop the motor while ramping beyond a certain speed. In such a case, increase the torque demand value. Setting the torque demand higher than the required value results in a stepped rotation of the motor. In such a case, reduce the torque demand value. Setting a very high torque demand value might damage the board.

Figure 1-11 shows the initial torque demand value set at 1 ampere.

FIGURE 1-11: INITIAL TORQUE DEMAND OF 1 AMPERE



The initial torque demand value should be sufficiently high to move the load as the motor starts operating. Make sure the hardware supports the required torque settings. Start with a value of 1.0 for initial torque demand, and double it each try until the rotor catches up with the stator field by the end of the ramp.

AN1078 Tuning Procedure ReadMe

1.7 SETTING MOTOR PARAMETERS

The motor parameters: POLEPAIRS, PHASERES, PHASEIND, NOMINALSPEEDINRPM, and MINSPEEDINRPM are located in the UserParms .h file. The motor parameters are based on the motor specification, and the values need to be updated when a different motor is tested. Example 1-3 shows the motor parameter settings.

EXAMPLE 1-3: MOTOR PARAMETERS SETTING

```
Number of Pole Pairs ← //***** Motor Parameters *****
Phase Resistance ← #define POLEPAIRS 5 // Number of pole pairs
Phase Inductance ← #define PHASERES ((float)2.67) // Phase resistance in Ohms.
Nominal Speed in RPM ← #define PHASEIND ((float)0.00192) // Phase inductance in Henrys.
// Make sure NOMINALSPEEDINRPM generates a MAXOMEGA < 1.0
// Use this formula:
// MAXOMEGA = NOMINALSPEEDINRPM*SPEEDLOOPTIME*POLEPAIRS*2/6
// If MAXOMEGA > 1.0, reduce NOMINALSPEEDINRPM or execute
// speed loop faster by reducing SpeedLoopTime.
// Maximum position of POT will set a reference of
// NOMINALSPEEDINRPM.
Minimum Required Speed in RPM ← #define MINSPEEDINRPM 500 // Minimum speed in RPM. Closed loop will operate at this
// speed. Open loop will transition to closed loop at
// this minimum speed. Minimum POT position (CCW) will set
// a speed reference of MINSPEEDINRPM
Maximum Required Speed in RPM After Field Weakening ← #define FIELDWEAKSPEEDRPM 5500 // Make sure FIELDWEAKSPEEDRPM generates a MAXOMEGA < 1.0
// Use this formula:
// MAXOMEGA = FIELDWEAKSPEEDRPM*SPEEDLOOPTIME*POLEPAIRS*2/6
// If MAXOMEGA > 1.0, reduce FIELDWEAKSPEEDRPM or execute
// speed loop faster by reducing SpeedLoopTime.
// Maximum position of POT will set a reference of
// FIELDWEAKSPEEDRPM.
```

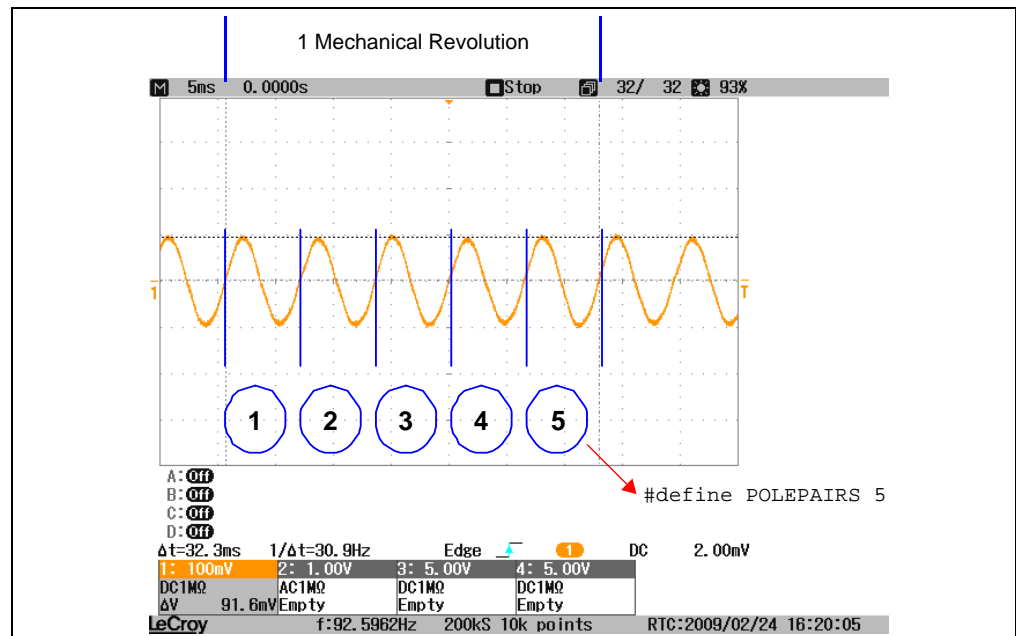
The number of pole pairs can be obtained from the motor specification sheet. It can also be obtained by driving the motor at a constant speed (i.e., using another motor), and by measuring the frequency of back-EMF. Using the measured frequency value, the pole pair is calculated using Equation 1-1.

EQUATION 1-1:

$$PolePair = (60 \cdot Frequency\ in\ Hertz) / (Speed\ in\ RPM)$$

Figure 1-12 shows the relationship between the mechanical revolution and the number of poles.

FIGURE 1-12: RELATION BETWEEN MECHANICAL REVOLUTION AND NUMBER OF POLES



The phase resistance and phase inductance of the motor are measured as follows:

- Phase Resistance – Use a multimeter and measure the DC-resistance across the two phase wires of PMSM. Substitute the measured resistance value in the following equation:

$$PHASERES = \text{Measured Resistance}/2$$

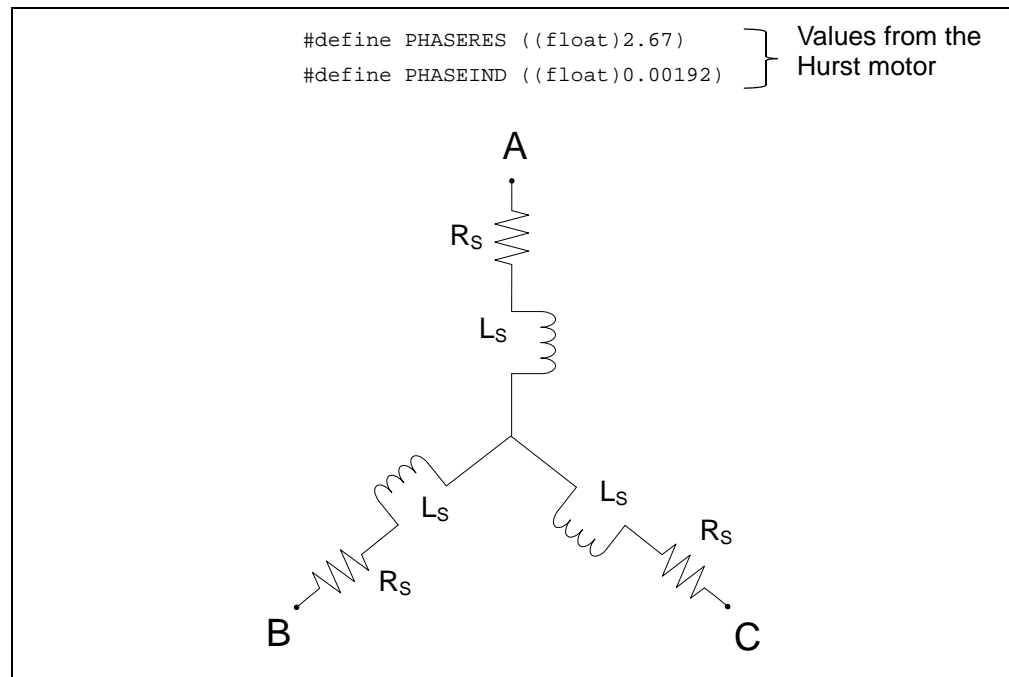
- Phase Inductance – Use a LCR meter and measure the inductance at 1 kHz across the two phase wires of PMSM. Substitute the measured inductance value in the following equation:

$$PHASEIND = \text{Measured Inductance}/2$$

These values can also be found in the manufacturer's motor specifications.

Figure 1-13 shows the measurement points.

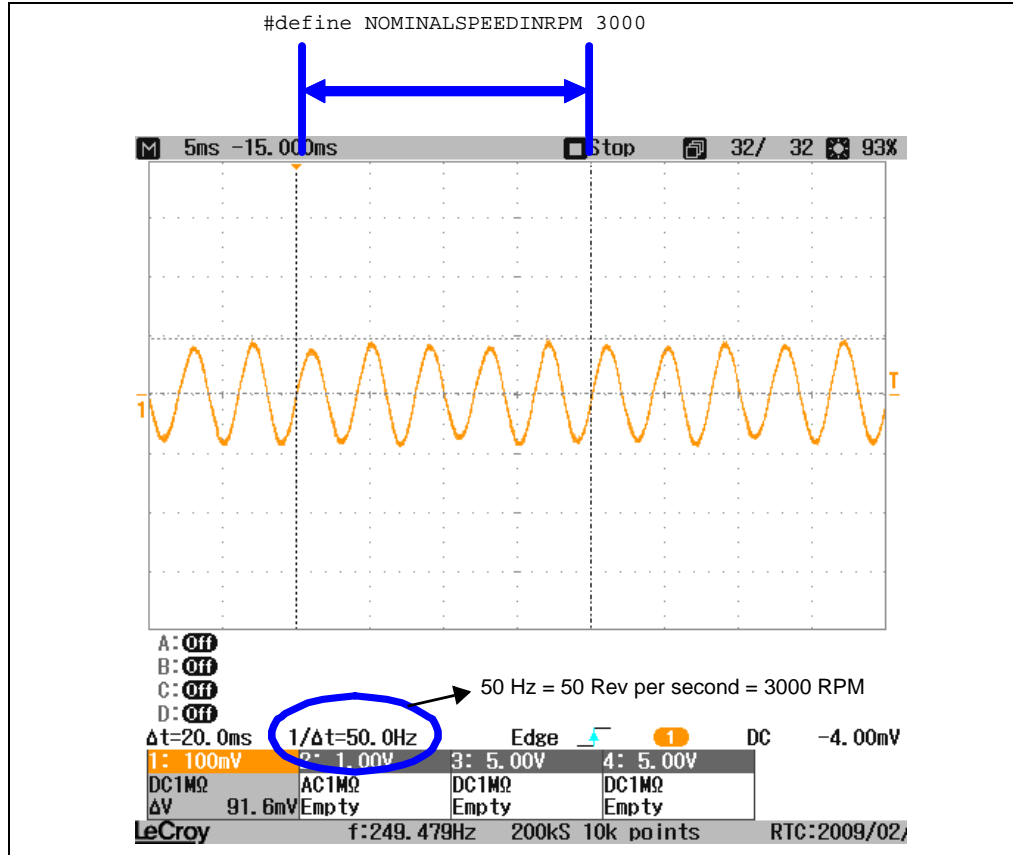
FIGURE 1-13: MEASUREMENT OF PHASE VALUES FROM LINE-TO-LINE VALUES



The nominal speed of a motor can be obtained from the motor specification sheet. Figure 1-14 shows the waveform of a motor running at a nominal speed of 3000 RPM.

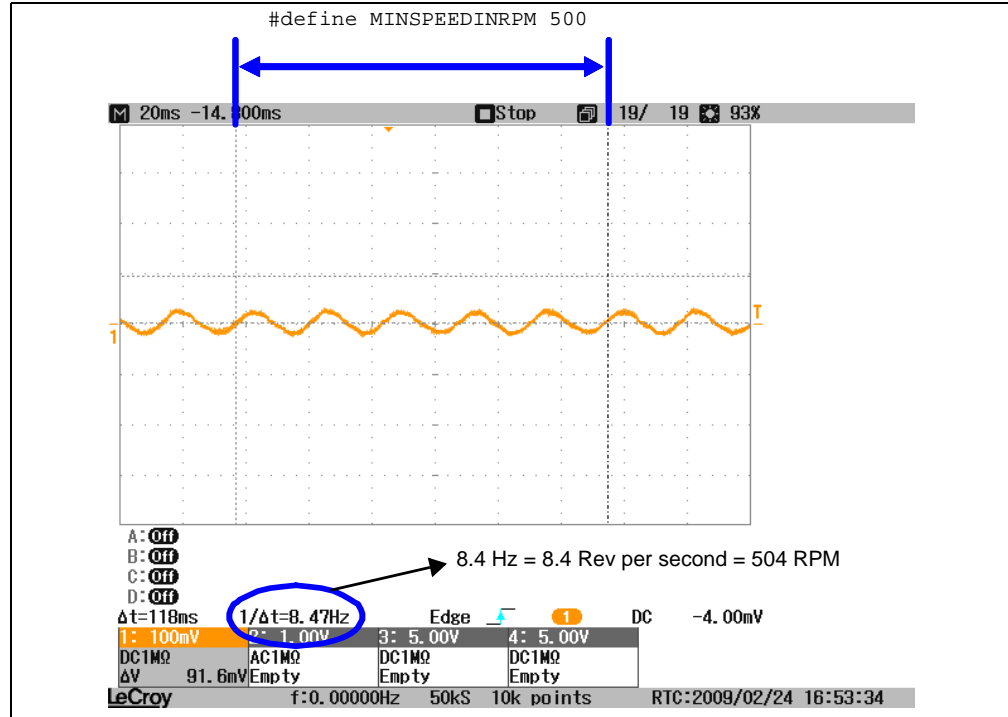
AN1078 Tuning Procedure ReadMe

FIGURE 1-14: MOTOR RUNNING AT 3000 RPM



The MINSPEEDINRPM is the minimum speed at which the motor runs satisfactorily with FOC. This parameter may change depending on the motor and the load torque. Figure 1-15 shows the waveform of a motor running at a minimum speed of 500 RPM. Initially, you can set this value between 10% and 15% of the rated motor speed, and fine tune it later.

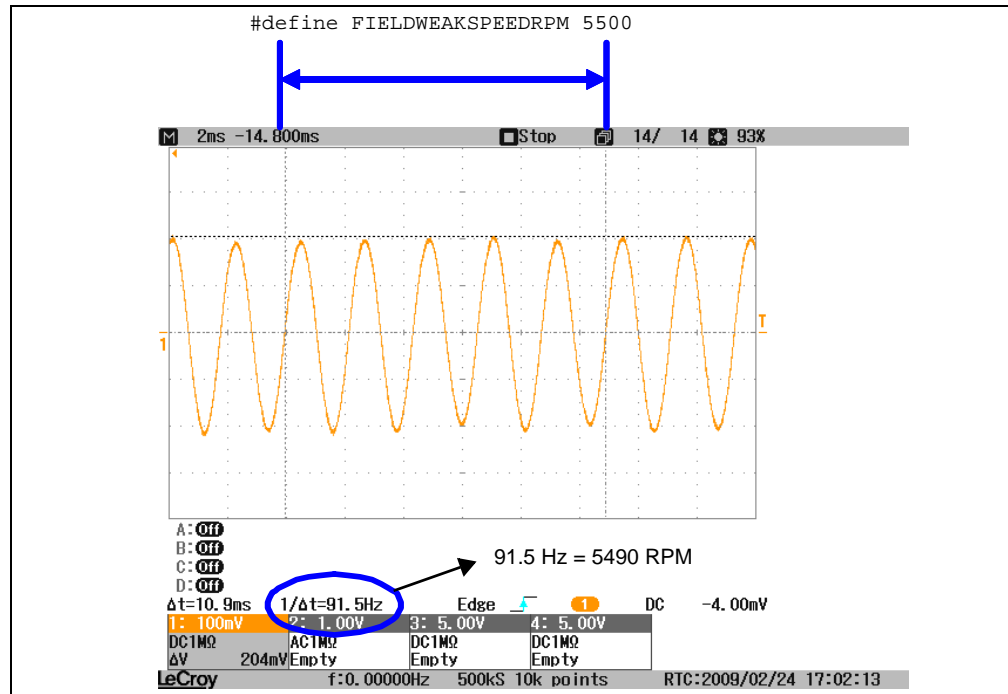
FIGURE 1-15: MOTOR RUNNING AT 500 RPM



The FIELDWEAKSPEED is the maximum desired speed at which the motor should run in Field Weakening mode. If Field Weakening mode is not required, set the FIELDWEAKSPEED with the same value as NOMINALSPEEDINRPM.

Figure 1-16 shows the waveform of a motor running at a speed of 5500 RPM in Field Weakening mode.

FIGURE 1-16: MOTOR RUNNING AT 5500 RPM IN FIELD WEAKENING MODE



1.8 PMSM FOC TUNING STEPS (OPEN LOOP)

The first step is to disable the transition from open loop to closed loop, so that the user can monitor the current consumed by the motor using an oscilloscope or the Data Monitor and Control Interface (DMCI).

By commenting the lines highlighted in Example 1-4, the motor remains in open loop and allows the user to analyze the ramping parameters.

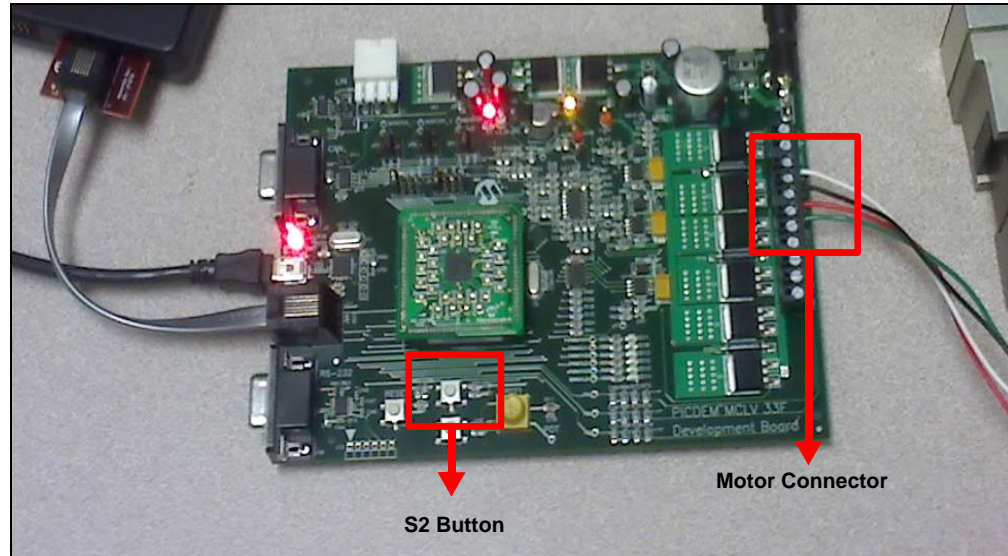
EXAMPLE 1-4: MOTOR SETTINGS IN OPEN LOOP

```
}  
  
void CalculateParkAngle(void)  
{  
    smc1.Ialpha = ParkParm.qIalpha;  
    smc1.Ibeta  = ParkParm.qIbeta;  
    smc1.Valpha = ParkParm.qValpha;  
    smc1.Vbeta  = ParkParm.qVbeta;  
  
    SMC_Position_Estimation(&smc1);  
  
    if(uGF.bit.OpenLoop)  
    {  
        if (Startup_Lock < MotorParm.LockTime)  
            Startup_Lock += 1; // This variable is incremented until  
                               // lock time expires, then the open loop  
                               // ramp begins  
        else if (Startup_Ramp < MotorParm.EndSpeed)  
            // Ramp starts, and increases linearly until EndSpeed is reached.  
            // After ramp, estimated theta is used to commutate motor.  
            Startup_Ramp += DELTA_STARTUP_RAMP;  
        else  
        {  
            // This section enables closed loop, right after open loop ramp.  
            // uGF.bit.ChangeMode = 1;  
            // uGF.bit.OpenLoop = 0;  
            // Difference between force angle and estimated theta is saved,  
            // so a soft transition is made when entering closed loop.  
            Theta_error = ParkParm.qAngle - smc1.Theta;  
        }  
    }  
}
```

1.8.1 Starting the dsPICDEM MCLV Development Board in Open Loop

1. Connect the motor phases to the dsPICDEM MCLV Development Board as shown in Figure 1-17

FIGURE 1-17: dsPICDEM MCLV Development Board with Motor Connection

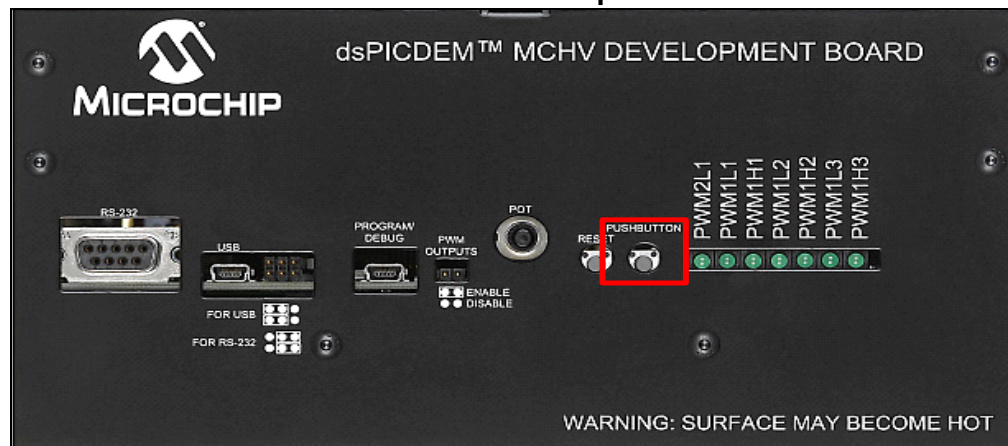


2. Program the dsPIC[®] DSC (Digital Signal Controller) using AN1078.
3. Press the S2 button to run the motor in open loop.

1.8.2 Starting dsPICDEM MCHV Development Board in Open Loop

1. Connect the motor phases to dsPICDEM MCHV Development board.
2. Program the dsPIC DSC using AN1078.
3. Press the PUSHBUTTON to run the motor in open loop, as shown in Figure 1-18.

FIGURE 1-18: dsPICDEM MCHV Development Board

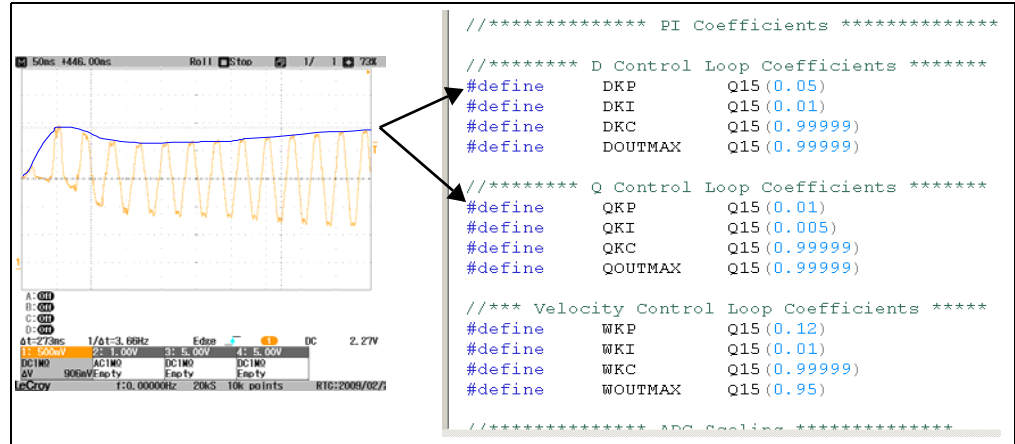


The motor should stay in open loop and at a fixed current amplitude.

If start-up current oscillates in open loop (see Figure 1-19), adjust the PI coefficients of the Id and Iq controllers to remove the oscillations. Reduce the Proportional Gain (Kp) and make sure that the Integral Gain (Ki) is 5 to 10 times smaller than Kp. Figure 1-19 shows the oscillation and the corresponding PI coefficient values.

AN1078 Tuning Procedure ReadMe

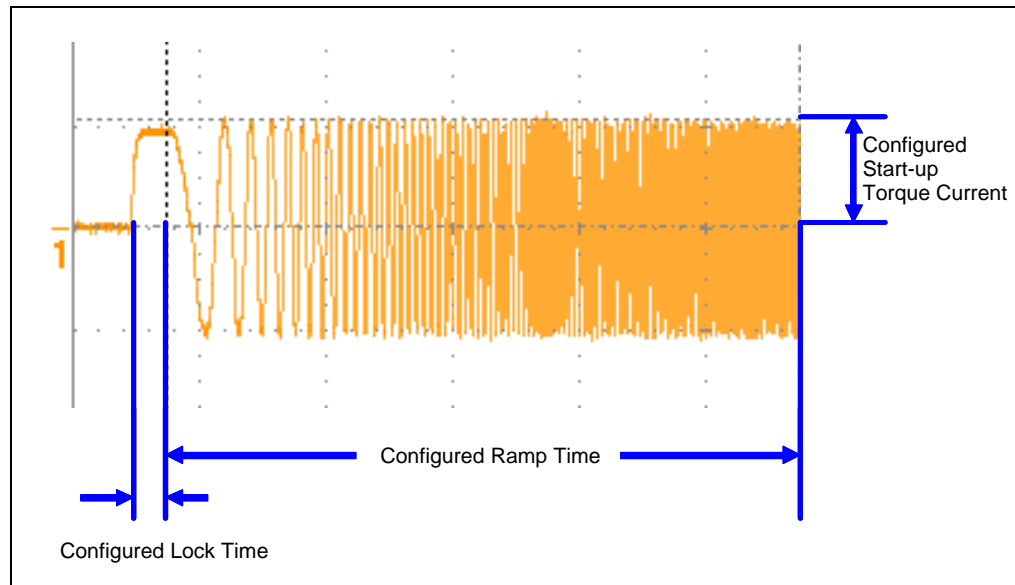
FIGURE 1-19: OSCILLATION IN CURRENT WAVEFORM



If the motor stops during the open loop ramp, the user should increase the ramp time. Once the motor starts running to the end of the ramp, slightly increase the initial torque current and reduce the ramp time until the motor operation meets the start-up requirements. If the rotor oscillates when the motor is energized and causes the motor to rotate in the opposite direction, increase the lock time.

Figure 1-20 shows the current waveform of a motor, while running in open loop condition. The lock time, ramp time, and torque demand values are shown in terms of current waveform.

FIGURE 1-20: CURRENT WAVEFORM OF A WELL-TUNED MOTOR IN OPEN LOOP



Enable the Data Monitor and Control Interface (DMCI)/Real-Time Data Monitoring (RTDM) variables. Enable the plots for alpha, Estimated Ialpha, Ibeta, and Estimated Ibeta to ensure that the Slide Mode Controller (SMC) is tracking the measured currents. Example 1-5 shows the code setting for viewing variables in RTDM/DMCI.

EXAMPLE 1-5: CODE SETTING FOR VIEWING VARIABLES ON RTDM

```

//***** Real Time Data Monitor, RTDM *****

#define RTDM // This definition enabled Real Time
// to handle RTDM protocol, and arra
// information in real time

#ifdef RTDM
#define DATA_BUFFER_SIZE 150 //Size in 16-bit Words
#define SNAPDELAY 10 // In number of PWM Interrupts
#define SNAP1 smc1.Ialpha
#define SNAP2 smc1.Ibeta
#define SNAP3 smc1.EstIalpha
#define SNAP4 smc1.EstIbeta
#endif

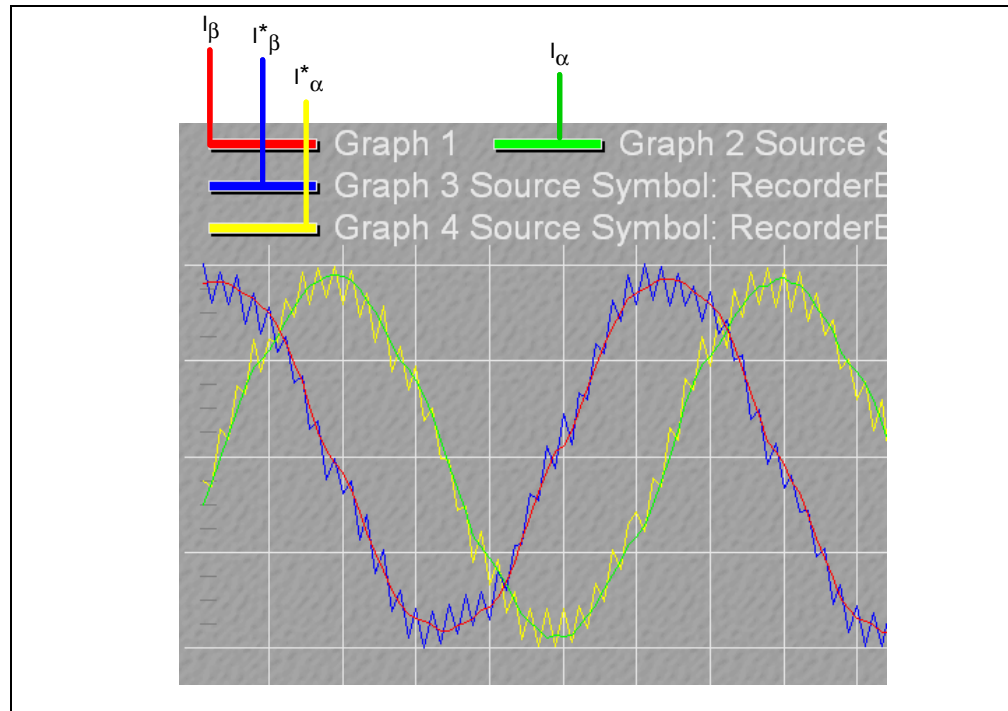
```

} Enter variable names here

Run the motor and capture the data with DMCI. The estimated current must track the measured current, and the estimated current ripple should be between 10% to 30% of the measured current peak-to-peak

Figure 1-21 shows the waveforms of actual current (red and green lines) and the estimated current (blue and yellow lines). The ripple of the estimated current should be between 10% to 30% of the measured current. Otherwise, tune the PI gains for the D and Q axes.

FIGURE 1-21: ACTUAL AND ESTIMATED CURRENT WAVEFORM



Enable plots for Ialpha, Ealpha, EalphaFinal and Theta to check the position estimation results. Example 1-6 shows the code setting for viewing different plots using the RTDM tool.

AN1078 Tuning Procedure ReadMe

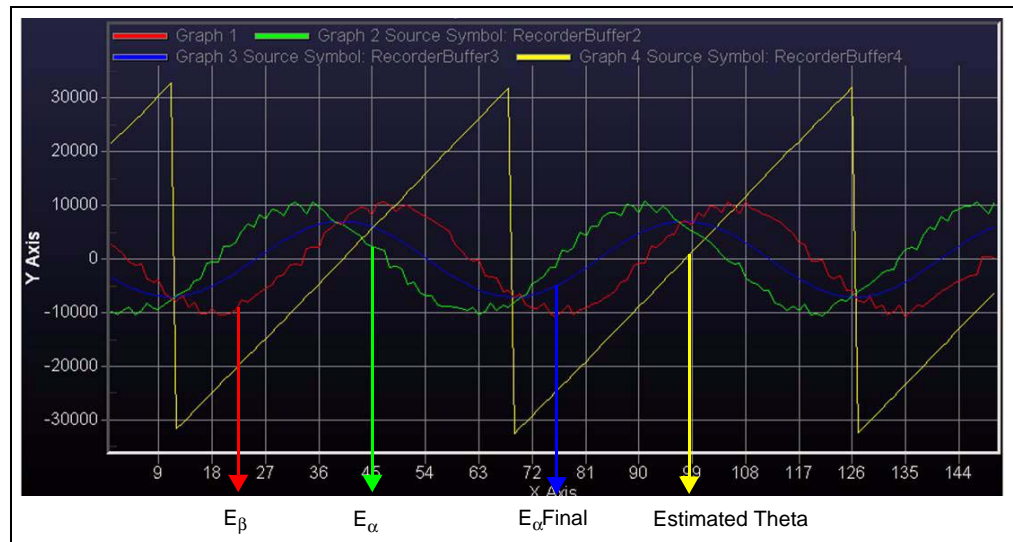
EXAMPLE 1-6: CODE SETTING FOR VIEWING PLOTS ON RTDM

```
/****** Real Time Data Monitor, RTDM *****/  
  
#define RTDM // This definition enabled Real Time Data Monitor, UART interrupts  
// to handle RTDM protocol, and array declarations for buffering  
// information in real time  
  
#ifndef RTDM  
#define DATA_BUFFER_SIZE 150 //Size in 16-bit Words  
#define SNAPDELAY 10 // In number of PWM Interrupts  
#define SNAP1 smcl.Ialpha  
#define SNAP2 smcl.Ealpha  
#define SNAP3 smcl.EalphaFinal  
#define SNAP4 smcl.Theta  
#endif
```

Figure 1-22 shows the relationship between the four different waveforms. The phase difference is due to the quadrature properties of each signal or due to the filter phase delay. The different waveforms are as follows:

- The green and red waveforms are the E_{α} and E_{β} , respectively, which are 90° apart.
- The blue waveform is the E_{α} Final. The E_{α} Final and E_{β} Final (not shown in figure) are 90° apart. The E_{α} and E_{α} Final are 45° apart.
- The yellow waveform is the Estimated Theta.

FIGURE 1-22: RELATIONSHIP BETWEEN DIFFERENT WAVEFORMS



Make sure the BackEMF Final does not have noise and a DC offset. The Estimated Theta is calculated from E_{α} Final and E_{β} Final by using the CORDIC function. The waveforms of E_{α} Final and E_{β} Final should be relatively noise free to estimate a good waveform of Estimated Theta.

Next, we will modify the SMC parameters. All of the controller parameters are located in the `UserParms.h` file. The SMC gain and linear region settings are shown in Example 1-7.

EXAMPLE 1-7: SLIDE-MODE CONTROLLER SETTINGS

```

//***** Slide Mode Controller Parameters *****
#define SMCGAIN 0.85 // Slide Mode Controller Gain (0.0 to 0.9999)
#define MAXLINEARSMC 0.01 // If measured current - estimated current
// is less than MAXLINEARSMC, the slide mode
// Controller will have a linear behavior
// instead of ON/OFF. Value from (0.0 to 0.9999)

```

Slide-Mode Controller Gain → 0.85
 Linear SMC Window → 0.01

Figure 1-23 shows the block diagram of a SMC with the gain and linear region set at 0.85 and 0.01, respectively.

FIGURE 1-23: BLOCK DIAGRAM OF SLIDE MODE CONTROLLER

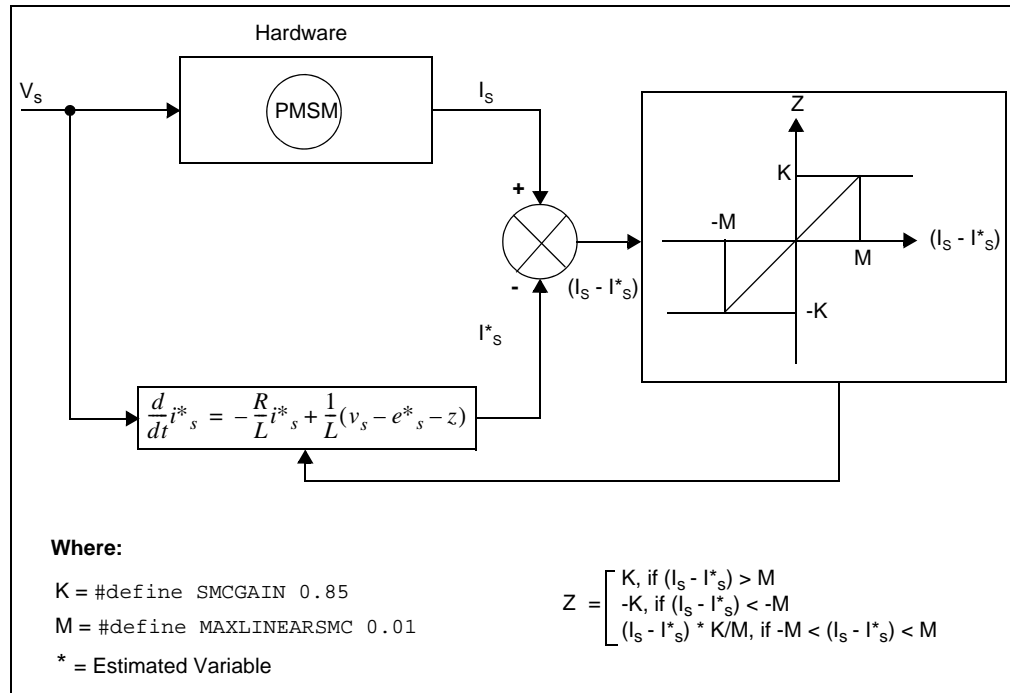
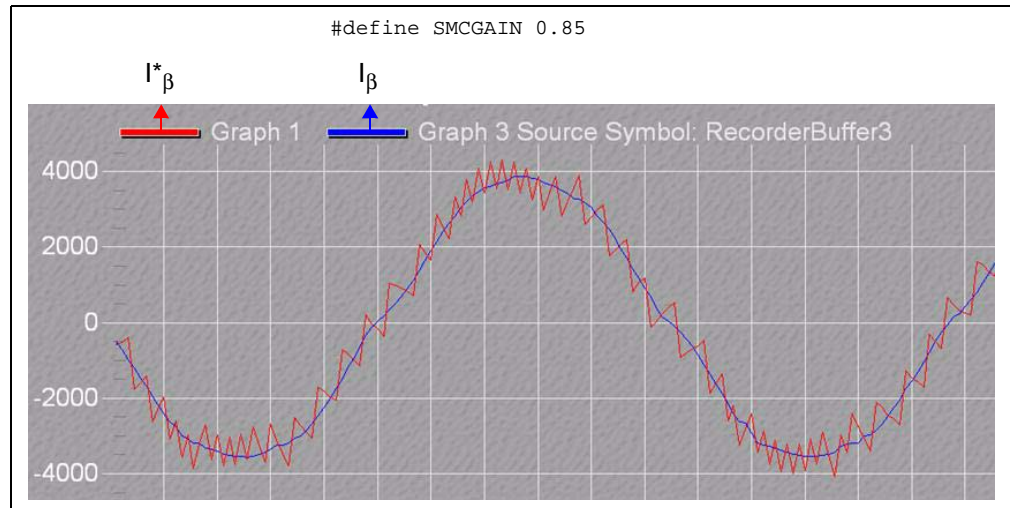


Figure 1-24 shows the estimated current waveform versus the actual waveform.

FIGURE 1-24: ESTIMATED CURRENT vs. ACTUAL CURRENT



AN1078 Tuning Procedure ReadMe

The estimated current must track the measured current. The estimated current ripple should be tuned between 10% and 30% of the measured current peak-to-peak.

The `MAXLINEARSMC` value of 0.010 provides smoother tracking with the same peak-to-peak value of estimated ripple. Figure 1-25 shows the estimated current waveforms for different values of `MAXLINEARSMC`. An optimal value of `MAXLINEARSMC` will significantly reduce the peak ripple of the estimated current.

FIGURE 1-25: SLIDE-MODE ESTIMATOR OUTPUT

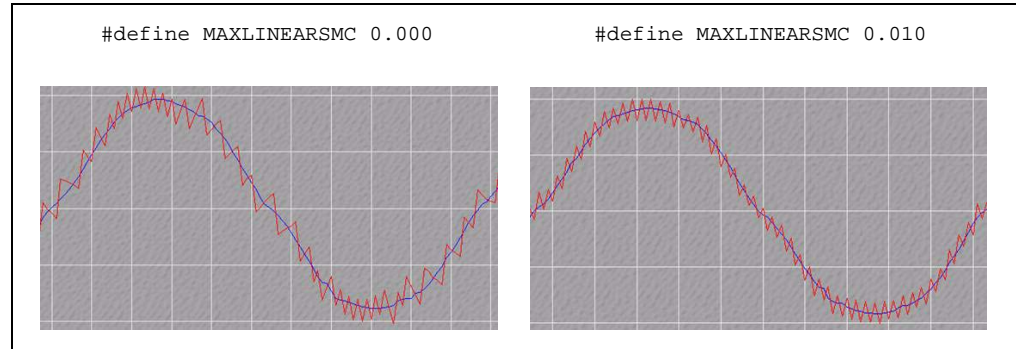
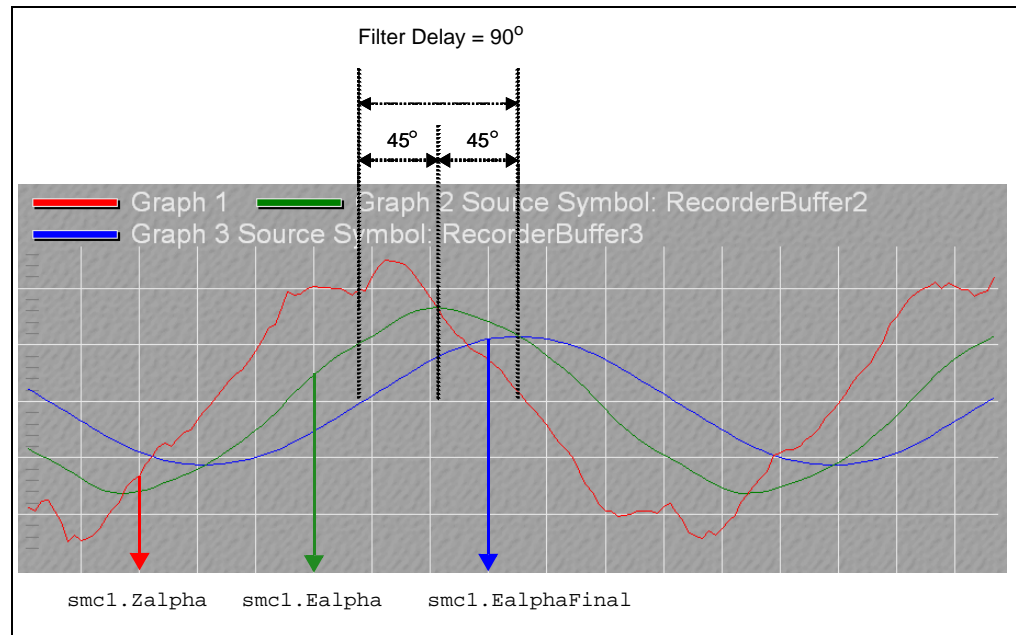


Figure 1-26 shows the phase delay due to filtering. The description of different waveforms are as follows:

- `smc1.Zalpha` is the actual signal
- `smc1.Ealpha` is the signal obtained by filtering `smc1.Zalpha` using a single-pole digital low-pass filter with a cut-off frequency equal to the input frequency. Therefore, a phase delay of 45° is present between the two signals
- `smc1.EalphaFinal` is the signal obtained by filtering `smc1.Ealpha` using a single-pole digital low-pass filter with a cut-off frequency equal to the input frequency. Therefore, a phase delay of 45° is present between the two signals
- At the end, there is a combined phase delay of 90° between the `smc1.Zalpha` and the `smc1.EalphaFinal`

FIGURE 1-26: PHASE DELAY DUE TO FILTERING



1.9 PMSM FOC TUNING STEPS (CLOSED LOOP MODE)

The closed loop operation of the motor can be enabled by uncommenting the lines highlighted in Example 1-8. The motor will be operated in closed loop mode using the Estimated Theta after the open loop ramp.

EXAMPLE 1-8: ENABLING CLOSED LOOP

```
void CalculateParkAngle(void)
{
    smc1.Ialpha = ParkParm.qIalpha;
    smc1.Ibeta  = ParkParm.qIbeta;
    smc1.Valpha = ParkParm.qValpha;
    smc1.Vbeta  = ParkParm.qVbeta;

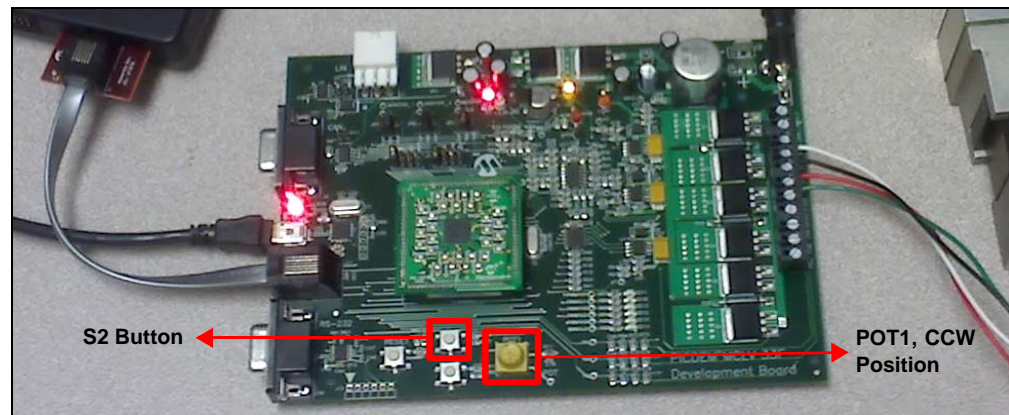
    SMC_Position_Estimation(&smc1);

    if(uGF.bit.OpenLoop)
    {
        if (Startup_Lock < MotorParm.LockTime)
            Startup_Lock += 1; // This variable is incremented until
                               // lock time expires, then the open loop
                               // ramp begins
        else if (Startup_Ramp < MotorParm.EndSpeed)
            // Ramp starts, and increases linearly until EndSpeed is reached.
            // After ramp, estimated theta is used to commutate motor.
            Startup_Ramp += DELTA_STARTUP_RAMP;
        else
        {
            // This section enables closed loop, right after open loop ramp.
            uGF.bit.ChangeMode = 1;
            uGF.bit.OpenLoop = 0;
            // Difference between force angle and estimated theta is saved,
            // so a soft transition is made when entering closed loop.
            Theta_error = ParkParm.qAngle - smc1.Theta;
        }
        ParkParm.qAngle += (int)(Startup_Ramp >> 16);
    }
    else
```

1.9.1 Starting dsPICDEM MCLV Development Board in Closed Loop

1. Move the potentiometer (POT1) to the counter-clockwise (CCW) position to ensure that the minimum speed is set.
2. Program the dsPIC DSC with the updated software program.
3. Press the S2 button to run the motor in open loop, as shown in Figure 1-27. After ramping up, the closed loop mode will be enabled automatically in the FOC algorithm

FIGURE 1-27: dsPICDEM MCLV DEVELOPMENT BOARD



The potentiometer is used as a speed reference input, and the S2 button is used to run/stop the motor.

AN1078 Tuning Procedure ReadMe

1.9.2 Starting the dsPICDEM MCHV Development Board in Closed Loop Mode

1. Move the potentiometer (POT) to the counter-clockwise (CCW) position to ensure that the minimum speed is set.
2. Program the dsPIC DSC with the updated software program.
3. Press PUSHBUTTON to run the motor in open loop mode. After ramping up, the closed loop mode will be automatically enabled in the FOC algorithm

Figure 1-28 shows the potentiometer, which is used as a speed reference input, and the push button to run/stop the motor.

FIGURE 1-28: dsPICDEM MCHV DEVELOPMENT BOARD

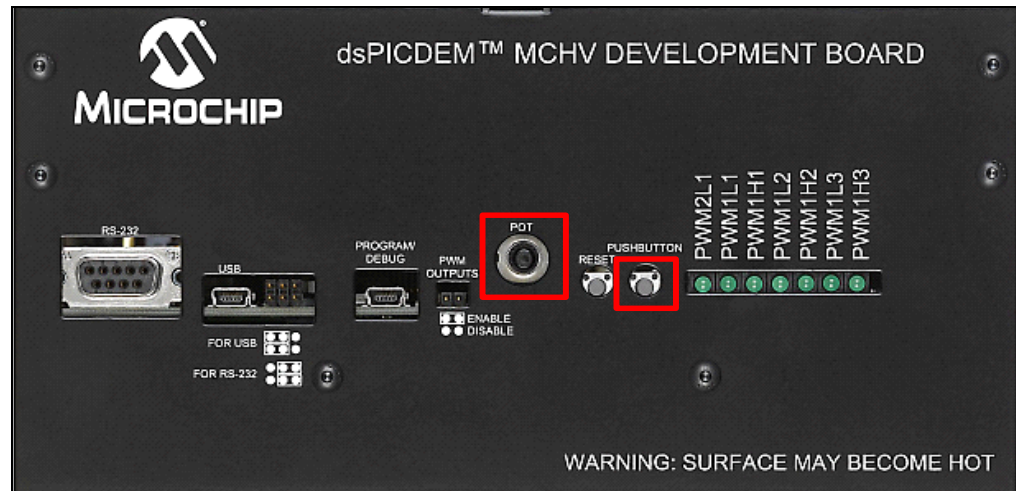
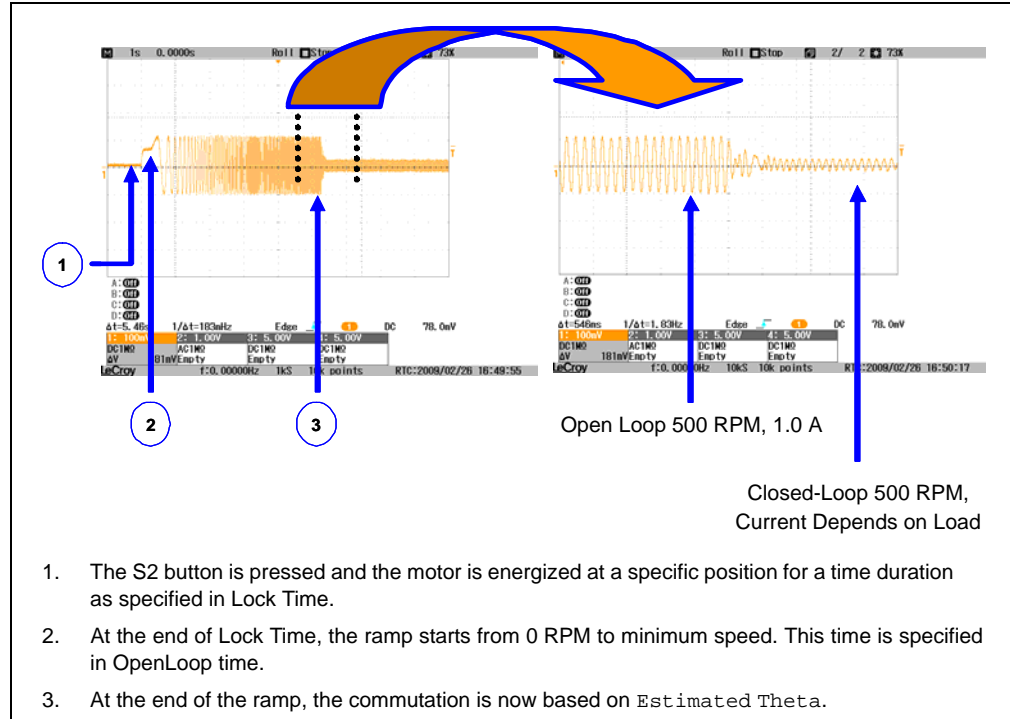


Figure 1-29 explains the steps to be followed to run the motor in closed loop mode. In event 1, the S2 button is pressed and the motor locks. In event 2, the ramp begins and the frequency increases linearly. At event 3, the ramp ends and the motor goes into closed loop operation. During the ramping, the Estimated Theta is calculated and the value is used while transitioning to the closed loop mode.

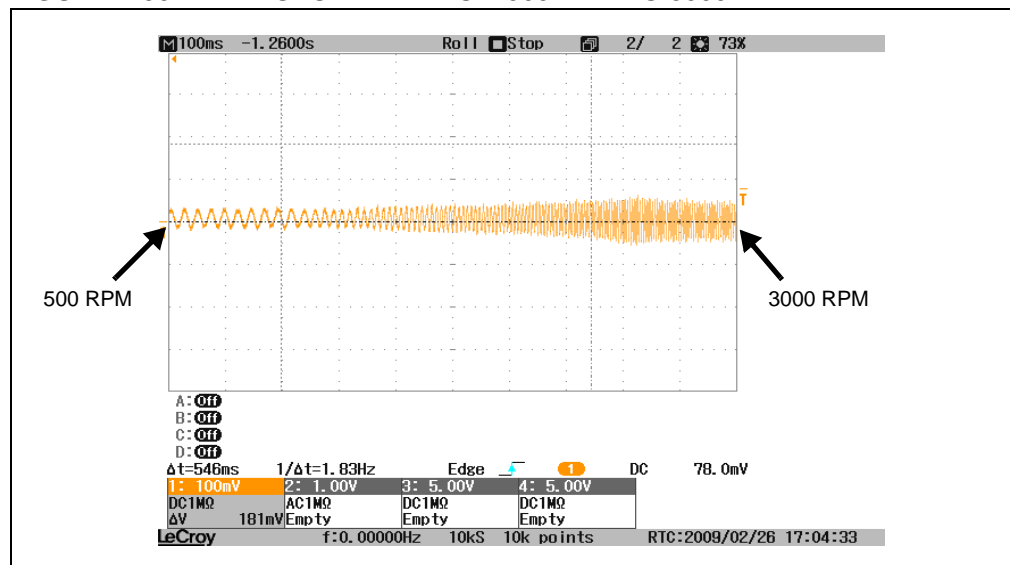
FIGURE 1-29: RUNNING A MOTOR IN CLOSED LOOP MODE



1.9.3 Adjusting I_d and I_q PI gains in Closed Loop Mode

Increase the speed reference by moving the potentiometer (POT) clockwise (CW) to verify that the current is stable. The current should be stable and if required, tune the PI gains for the I_d and I_q axes, and gain for the SMC estimator. Figure 1-30 shows the EMF of a motor driven from 500 to 3000 RPM.

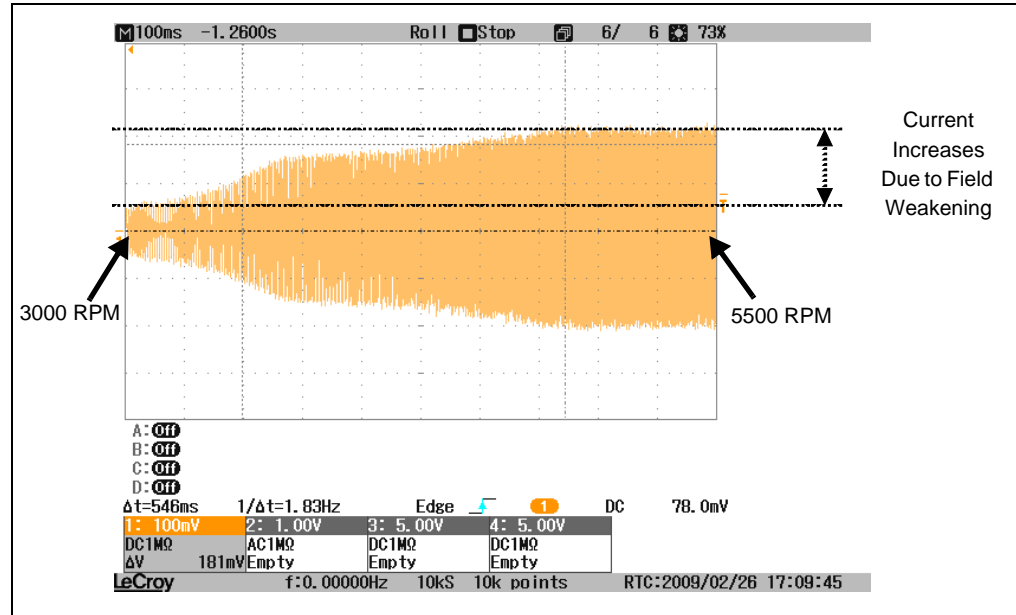
FIGURE 1-30: MOTOR EMF FROM 500 RPM TO 3000 RPM



1.9.4 Tuning Transient Response

Figure 1-31 shows how to check the transient response of the motor and FOC in the dsPICDEM MCLV Development Board. By pressing the S3 button, the motor speed command is doubled and the response of the FOC algorithm can be observed in the oscilloscope. For the dsPICDEM MCHV Development Board, this step is not applicable, as it does not have a switch to double the speed command.

FIGURE 1-31: TRANSIENT RESPONSE OF A MOTOR



1.9.5 Adjusting Software Current Gains

The current gains of the software can be adjusted based on the hardware design, as shown in Example 1-9. Modify the ADC scaling parameters (DQKA and DQKB) in the UserParms.h file based on the hardware design. The ADC result is fractional, therefore, modify the scaling parameters and the hardware to get a full range reading on the ADC of ± 0.5 at maximum current input.

EXAMPLE 1-9: SOFTWARE GAIN OF CURRENT SIGNAL

```
***** ADC Scaling *****
// Scaling constants: Determined by calibration or hardware design.
#define DQR      Q15((OMEGA10 - OMEGA1)/2.0) // POT Scaling
#define DQKA     Q15(0.5) // Current feedback software gain
#define DQKB     Q15(0.5) // Current feedback software gain
```

1.9.6 Tuning in Torque Mode

If the open loop performance of the motor is good, but if it is not locking during the closed loop operation, try running the motor in Torque mode. To run the motor in Torque mode, uncomment the TORQUEMODE define in the UserParms.h file, as shown in Example 1-10. The Torque mode bypasses the velocity PI loop and the input from the potentiometer is taken as the torque setting. Once the Torque mode fine tuning has been completed by adjusting the PI gains for I_d and I_q for smooth current in closed loop mode, run the motor in Speed mode by commenting the TORQUEMODE define.

EXAMPLE 1-10: CODE SETTING FOR RUNNING MOTOR IN TORQUE MODE

```
//***** Optional Modes *****  
#define TORQUEMODE  
//#define ENVOLTRIPPLE  
  
//***** PI Coefficients *****
```

1.9.7 Scaling Motor Resistance and Inductance

In some motors, the closed loop operation is possible only when the resistance and inductance by the maximum current sensing capability of the hardware (I_{max}/V_{rated}). The value of I_{max} for the dsPICDEM MCLV and MCHV Development Boards are 4.4A and 16.5A, respectively. The voltage rating of the motor can be obtained from the motor's specification sheet.

If the motor still does not lock under closed loop operation, scale the phase resistance and phase inductance to the maximum value of voltage and current. Equation 1-2 and Equation 1-3 show the maximum current calculation for the MCLV and MCHV boards, respectively.

EQUATION 1-2: DSPICDEM MCLV DEVELOPMENT BOARD

$$R_{shunt} = 0.005\Omega, V_{DD} = 3.3V, Gain = 75$$
$$Maximum\ Current = (3.3/2)/(0.005 * 75) = 4.4A$$

EQUATION 1-3: DSPICDEM MCHV DEVELOPMENT BOARD

$$R_{shunt} = 0.01\Omega, V_{DD} = 3.3V, Gain = 10$$
$$Maximum\ Current = (3.3/2)/(0.01 * 10) = 16.5A$$

Example 1-11 shows the scaling of phase resistance and phase inductance for the dsPICDEM MCLV Development Board and 24V motor.

For the dsPICDEM MCHV Development Board, replace the maximum current with 16.5A, and the voltage of the motor as per the application. For the AC rating of the motor, take the RMS value.

EXAMPLE 1-11: SCALING OF RESISTANCE AND INDUCTANCE

```
//***** Motor Parameters *****  
  
#define POLEPAIRS      5          // Number of pole pairs  
#define PHASERES      ((float)2.67 * (4.4/24)) // Phase resistance in Ohms.  
#define PHASEIND      ((float)0.00192 * (4.4/24)) // Phase inductance in Henrys.
```

Some motors due to their distinctive design, may not lock under closed loop operation regardless of the tuning steps followed earlier in this document. To run such motors, reduce the value of phase resistance and phase inductance to a value less than the measured value in the UserParms.h file, and check to see that the value of `Theta_error` in the PMSM.c file is reduced as a result by displaying it using DMCI/RTDM. The reduction should continue until the value of `Theta_error` becomes small enough to enable the closed loop operation of the motor. Example 1-12 shows the `Theta_error` in PMSM.c file.

AN1078 Tuning Procedure ReadMe

EXAMPLE 1-12: Theta_Error IN PMSM.C FILE

```
1
//***** Motor Parameters *****

#define POLEPAIRS      5           // Number of pole pairs
#define PHASERES      ((float)2.67) // Phase resistance in Ohms.
#define PHASEIND      ((float)0.00192) // Phase inductance in Henrys

// This section enables closed loop, right after open loop ramp.
uGF_bit.ChangeMode = 1;
uGF_bit.OpenLoop = 0;
// Difference between force angle and estimated theta is saved,
// so a soft transition is made when entering closed loop.
Theta_error = ParkParm.qAngle - smcl.Theta;
```

Diagram illustrating the location of parameters in the code:

- Parameters `POLEPAIRS`, `PHASERES`, and `PHASEIND` are located in `UserParms.h`.
- Parameters `Theta_error` and the initialization code for `uGF_bit` are located in `PMSM.c`.

1.9.8 Tuning Motors with Very Low Inductance

While running very small motors, which have a very low inductance value less than 100 μH , it is advantageous to increase the PWM switching frequency. This will allow a smoother control and will also reduce the audible noise. For low-inductance, the current waveforms appear as spikes during the PWM switching, which cannot be measured effectively by the ADC. Example 1-13 shows the PWM frequency setting in `Userparms.h` file

EXAMPLE 1-13: SETTING PWM FREQUENCY IN UserParms.h FILE

```
//***** PWM and Control Timing Parameters *****

#define PWMFREQUENCY  20000        // PWM Frequency in Hertz
#define DEADTIMESEC   0.000002    // Deadtime in seconds
#define BUTPOLLOOPTIME 0.100       // Button polling loop period in sec
#define SPEEDLOOPFREQ 1000         // Speed loop Frequency in Hertz. This value must
// be an integer to avoid pre-compiler error
```

The field weakening table as shown in Example 1-14, which is located in the `UserParms.h` file, is motor specific. The values need to be altered for different motors.

1.9.9 Adjusting for Field Weakening

EXAMPLE 1-14: FIELD WEAKENING TABLE

```
// in order to have field weakening, this reference value should be negative,  
// so maximum value in this example is -4.4, or REFINAMPS(-4.4)  
#define dqKFw0 REFINAMPS(0)  
#define dqKFw1 REFINAMPS(-0.435)  
#define dqKFw2 REFINAMPS(-0.650)  
#define dqKFw3 REFINAMPS(-0.915)  
#define dqKFw4 REFINAMPS(-1.075)  
#define dqKFw5 REFINAMPS(-1.253)  
#define dqKFw6 REFINAMPS(-1.432)  
#define dqKFw7 REFINAMPS(-1.670)  
#define dqKFw8 REFINAMPS(-1.838)  
#define dqKFw9 REFINAMPS(-1.952)  
#define dqKFw10 REFINAMPS(-2.042)  
#define dqKFw11 REFINAMPS(-2.064)  
#define dqKFw12 REFINAMPS(-2.100)  
#define dqKFw13 REFINAMPS(-2.100)  
#define dqKFw14 REFINAMPS(-2.100)  
#define dqKFw15 REFINAMPS(-2.100)
```

The maximum reference value for the dsPICDEM MCLV Development Board is 4.4A and for the dsPICDEM MCHV Development Board this value is 16.5A. The first value in the table should always be zero, which means no field weakening (for PMSM) at that speed. Enter the suitable negative values of current to match the required FW speed by experimentation.

CAUTION

Usually, the motor manufacturer indicates the maximum speed achievable by the motor without it being damaged (which could be higher than the brake point speed at rated current). If not, it is possible to run it at higher speeds but only for small periods (intermittent) assuming the risks of demagnetization or mechanical damage of the motor or of the devices attached to it.

In Field Weakening mode, if the controller becomes lost due to a miscalculation of the angle at high speed above the nominal value, the possibility of damaging the inverter is imminent. The reason is that the Back Electromotive Force (BEMF) will have a greater value than the one that would be obtained for the nominal speed, thereby exceeding the DC bus voltage value, which the inverter's power semiconductors and DC link capacitors would have to support. Since the tuning proposed implies iterative coefficient corrections until the optimum functioning is achieved, the protection of the inverter with corresponding circuitry should be modified to handle higher voltages in case of stalling at high speeds.

1.10 CONCLUSION

This document provides effective techniques for tuning the FOC algorithm described in AN1078 “*Sensorless Field Oriented Control of PMSM*” (DS01078) for running any PMSM motor.

Because many different motors were tuned in the process of developing this procedure, the AN1078 FOC algorithm and this procedure are more robust and should cover the requirements of most PMSM motors. The tuning techniques discussed in this document will help in reducing the time and effort involved in new development.

NOTES: