Atmel AV1636: Configurable PMSM Sensorless Field Oriented Control using the XMEGA

8-bit AVR Microcontrollers

Features

- 3-phase permanent magnet synchronous motor application
- Sensorless field oriented control algorithm
- Single shunt current reconstruction
- For Atmel® ATxmega16D4
- Reference design hardware and firmware
- PC based configuration utility
- Tuning guide

Introduction

This application note describes the implementation and use of a configurable 3-phase permanent magnet synchronous motor (PMSM) kit using the ATxmega16D4 microcontroller implementing sensorless field oriented control (FOC). The complete kit includes a 12V 3-phase PMSM, an ATxmega16D4 processor board, and a low voltage motor control board, a USB-to-UART bridge cable, a 12V universal power adapter, the firmware running on the ATxmega16D4, and a PC based configuration utility.
Table of Contents

1. Theory of operation ........................................................................................................ 3
   1.1 Overview .................................................................................................................. 3
   1.2 Hardware .................................................................................................................. 3
   1.3 Firmware .................................................................................................................. 3
   1.4 Configuration utility ............................................................................................... 3

2. System details ................................................................................................................. 3
   2.1 Hardware .................................................................................................................. 3
       2.1.1 3-phase PMSM ............................................................................................... 4
       2.1.2 Atmel ATxmega16D4 processor board ........................................................ 4
       2.1.3 Low voltage motor control board ................................................................. 4
       2.1.4 USB-to-UART bridge cable ............................................................................. 5
       2.1.5 Universal power adapter ............................................................................... 6
   2.2 Firmware .................................................................................................................. 6
       2.2.1 Control theory ................................................................................................. 6
       2.2.2 Code structure ............................................................................................... 9
       2.2.3 ISR routine ..................................................................................................... 10
       2.2.4 Communication .............................................................................................. 12
   2.3 Configuration utility ............................................................................................... 12
       2.3.2 Layout of utility ............................................................................................. 12
       2.3.3 Variables ........................................................................................................ 13
       2.3.4 Variable plot ................................................................................................. 13
       2.3.5 Parameters .................................................................................................... 13
       2.3.6 File functions ............................................................................................... 13

3. Running the system .................................................................................................... 13
   3.1 Hardware .................................................................................................................. 13
   3.2 Firmware .................................................................................................................. 15
   3.3 Configuration utility ............................................................................................... 15
       3.3.1 Running the configuration utility .................................................................... 16
       3.3.2 Motor variables and variable plot ................................................................. 17
       3.3.3 Motor parameters .......................................................................................... 20
       3.3.4 Test run of motor .......................................................................................... 24
       3.3.5 Oscillator calibration ..................................................................................... 25
       3.3.6 mode and input_cmd parameters ................................................................... 25
       3.3.7 Minimum bus voltage ..................................................................................... 28
       3.3.8 Volatges and currents .................................................................................... 28
       3.3.9 Back EMF calculator ....................................................................................... 29
       3.3.10 Back EMF sensing PLL ................................................................................. 33
       3.3.11 Speed and theta ............................................................................................ 35
       3.3.12 Current regulators ......................................................................................... 36
       3.3.13 Speed regulator ............................................................................................ 39

4. Conclusion .................................................................................................................... 40

Appendix A. Atmel ATmega16D4 Processor Board Schematics ........................................ 41
Appendix B. Low Voltage Motor Control Board Schematics ............................................. 42
Appendix C. 42BLS01-001 Motor Datasheet .................................................................... 43
Appendix D. Kit Parts List ............................................................................................... 44
Appendix E. Revision History .......................................................................................... 45
1. Theory of operation

1.1 Overview
This application note covers the description and use of a complete 3-phase PMSM sensorless FOC system.
After reading this document the reader should have an understanding of the system hardware, firmware, and configuration utility.

1.2 Hardware
The hardware consists of a complete motor control system. The system is composed of a 3-phase PMSM, an Atmel ATxmega16D4 processor board, an Atmel low voltage motor control board, a USB-to-UART bridge cable and a 12V universal power adapter.

1.3 Firmware
The firmware running on the ATxmega16D4 provides the control structure for PMSM sensorless FOC. The sensorless control technique uses back EMF calculation to form a back EMF phase locked loop (PLL). Speed control is provided by a proportional integral (PI) controller. The back EMF sensing PLL, the current regulators, and the speed control loop require configuration and tuning. This is covered in Chapter 3.

1.4 Configuration utility
The configuration utility is software running on a PC provided as a tool for configuring and tuning the sensorless FOC parameters in firmware.
Configuring consists of setting parameters that are known or calculated quantities. These quantities translate into parameters that are set in the EEPROM. Tuning involves an iterative process of setting and testing parameters to find the optimal value. These values are not easily calculated or involve unknown or not easily measurable quantities, so tuning provides the most accessible method of optimization.

2. System details

2.1 Hardware
The hardware consists of the 3-phase PMSM, the ATxmega16D4 processor board, the low voltage motor control board, the USB-to-UART bridge cable, and the 12V universal power adapter as shown in Figure 2-1.

Figure 2-1. Hardware block diagram.
2.1.1 3-phase PMSM
The 3-phase PMSM consists of a 3-phase 8-pole permanent magnet synchronous motor. The three motor leads, consisting of one red, one black and one yellow 20 AWG wire, are brought out for connection to the low voltage motor control board. The motor includes three Hall sensors leads plus a common sensor power and ground. These five 28 AWG wires can be left disconnected since they are not used in the sensorless algorithm.

The motor shown in Figure 2-2 is rated for 4,000 rpm at 12VDC input, the full motor specification can be found in Appendix C.

Figure 2-2. 3-phase permanent magnet synchronous motor.

2.1.2 Atmel ATxmega16D4 processor board
The ATxmega16D4 AVR® processor board supports the microcontroller, microcontroller power, programming and debugging, and serial communication connections.

The ATxmega processor board contains the following:
1. Atmel ATxmega16D4 AVR microcontroller.
2. 3.3V regulator to power to the AVR.
3. 6 pin header (2x3) PDI interface connector.
4. 6 pin header (1x6) for USB-to-UART bridge.

The complete schematics are shown in Appendix A.

2.1.3 Low voltage motor control board
The low voltage motor control board supports all the power and level shifting circuitry for 3-phase motor control for various motor types and control techniques.

The low voltage motor control board contains the following components:
1. Socket for Atmel ATxmega16D4 AVR processor board.
2. Gate drive circuitry.
3. Current sense resistor.
4. 3-phase power stage consisting of three N-channel MOSFETs as lower switches and three P-channel MOSFETs as upper switches.
5. DC bus bypass capacitor.
6. 2-pin header for 12V power adapter connector.
7. 3-pin header for motor phase connections.
The low voltage motor control board can be configured for up to 48V nominal input voltage and 12A rms output per phase depending on component selection. The configuration shown in Figure 2-3 and the schematics shown in Appendix B are for the board configured for 11 to 15VDC input range and a peak output current of 6A rms.

Figure 2-3.  Low voltage motor control board and an Atmel ATxmega16D4 processor board.

2.1.4 USB-to-UART bridge cable

The USB-to-UART bridge cable provides communication between a PC and the ATxmega16D4 processor board. This cable is a standard product from FTDI. Part numbers are listed in Appendix D.

Figure 2-4.  USB-to-UART bridge cable.
2.1.5 Universal power adapter
The universal power adapter provides 12V for powering the demo board. The adapter is rated for 2.5A at 12V. The AC power adapter uses a removable AC power connector for international use.

Figure 2-5. Universal power adapter with international plug options.

2.2 Firmware
The firmware was designed to run on the Atmel ATxmega16D4 to demonstrate a configurable motor control application driving 3-phase PMSM with sensorless FOC.

The sensorless control algorithm updates at a rate of 128μs (7812.5Hz) this can provide sinusoidal current regulation above 500Hz. The back EMF PLL is always enabled for robust operation that provides automatic motor starting or recovery of the motor speed after a stall condition.

2.2.1 Control theory
The sensorless field oriented control is based on measurement of stator current and transforming these quintiles to a reference frame rotating with the air gap flux linkage for independent control of the flux producing current (d-axis) and the torque producing current (q-axis). A block diagram of this control structure is shown in Figure 2-6. Stator current is measured using single shunt current reconstruction and field orientation is established using a back EMF phase locked loop.
The field orientation to the air gap flux linkage is established by coordinate transformation (Clark transform) and vector rotation (Park transformation) using the correct angle (theta) between a fixed reference point on the motor’s stator and the actual air gap flux represented as a vector. This can be visualized by Figure 2-7.
The sensorless field oriented control is based on calculating the back EMF from stator currents and voltages, then transforming the back EMF into the d-q reference frame that is reference by the air gap flux linkage, Figure 2-8. The back EMF phase locked loop uses d-axis back EMF as the phase error detector, since perfect lock should produce zero d-axis back EMF. This is based on the fact that back EMF is perpendicular to the flux linkage that generates it.
2.2.2 Code structure

The code structure is shown in Figure 2-9. There are two stages: initialization and then the control loop. The initialization takes care of all peripherals and reading of the EEPROM. The control loop is handled in an interrupt service routine (ISR) updated every 32µs. The ISR handles single shunt current sampling, reconstruction, all FOC operations, UART data, updating of the back EMF PLL, the speed control, and updating the pulse width modulation (PWM) outputs.
2.2.3 ISR routine

There is one main interrupt service routine (ISR) based off the PWM port C timer 0, triggered every 32µs. The timer interrupt triggers an analog to digital (ADC) conversion used for single shunt current reconstruction and the ISR. The ISR updates PWM, serial communication, and the FOC routines. Each ISR executes one fourth of the FOC routines, giving a 128µs update rate for the complete FOC control loop. Figure 2-10, shows the PWM ISR flow.

The field oriented control routines are further broken down into three functions:

1. The back EMF phase locked loop.
2. The vector control and regulator blocks.
3. The PWM control and current reconstruction.

Each function is split between the four ISRs as shown in Figure 2-11.
Figure 2-11. Field oriented control routines.
2.2.4 Communication
Serial communication to the firmware allows configuration of motor control parameters stored in the EEPROM and to allow data collection during tuning and configuring of the firmware.

A USB-to-UART bridge is used to communicate with a PC running the configuration utility.

2.3 Configuration utility
The PC based configuration utility is for configuration and tuning of the motor performance using a virtual COM port on the PC. The utility provides the following features:

1. Adjustable parameters for customizing the code to operate with different motors and sending these parameters to the EEPROM.
2. A list of variables that update every 123ms.
3. A variable plot that allows watching a single variable in a scope like plotting windows for performance testing.
4. An oscillator test to compare and adjust the Atmel ATxmega16D4 internal oscillator against the UART clock.
5. Simple file functions: a file load and save function for saving EEPROM values and a file save function to store plot data.

2.3.2 Layout of utility
The configuration utility consists of a single front panel with all the functionality displayed in one window. Figure 2-12 shows the basic layout of the configuration utility.
2.3.3 Variables
Variables from the firmware are sent to the configuration utility at the rate of approximately once per 123ms. The list of variables was chosen as the most useful for configuration and tuning of the motor.

2.3.4 Variable plot
One variable from the list can be displayed in the variable plot window. This variable is sent once every 256µs which is every other loop update. The complete plot is composed of 480 points so it is updated once every 123ms.

2.3.5 Parameters
The parameter list of values are also updated once every 123ms along with the variable list. During the configuration and tuning process, new values can be sent to the microprocessor and the new values will be reflected in the 123ms update. Updated parameters are stored in RAM so they are reset back to the EEPROM values after cycling power. To store updated values the EEPROM the program button must be pressed.

2.3.6 File functions
Additional file functions are there for configuring the utility, loading and saving parameters values, and saving variable plot data.

3. Running the system

3.1 Hardware
The system is connected as shown in Figure 3-1. Initially the Atmel ATxmega16D4 processor board can be powered directly from the USB port, this powers up the ATxmega16D4 allowing communication with the configuration utility without running the motor.
The PMSM phase leads are connected to the 3-pin connector J2. The motor phase leads are the 20 AWG black, red, and yellow wires from the motor. The five 28 AWG wires are Hall sensors connections not used in this application, so they should be left unconnected. Connecting the motor phase leads in the order shown in Figure 3-2 provides clockwise rotation for a positive command and counter clockwise rotation for a negative command. Swapping any two phases reverses the rotation direction for positive and negative commands.

Figure 3-2. PMSM phase leads phases a, b and c from top to bottom.
DC input voltage is connected to J3 shown in Figure 3-3. The DC input range is from 0 to 15V. The gate drive ICs (U1 to U3 from Appendix B) provide an under voltage lockout below approximately 10.1V so a supply of at least 12V nominal is recommended to run the motor. 12V power can be supplied from the universal power adapter or a variable lab supply can be used that is capable of supplying 12V and 2.5A.

Figure 3-3. DC bus voltage input leads +Vbus and –Vbus from top to bottom.

3.2 Firmware

The following instructions are for the Atmel ATxmega16D4 processor boards that do not have firmware preinstalled or are going to be upgraded from a previous version of firmware. This section can be skipped if firmware is already installed. Firmware needs to be installed on the ATxmega16D4 before using the configuration utility and parameters need to be stored in the EEPROM for the motor to run properly. There are provisions for downloads of firmware and fuse settings through the standard six pin header, J3 using many of the common programming tools and the latest version of Atmel Studio. Firmware can be installed using the following steps:

1. Download the AVR1636 - Firmware project.
2. Plug in USB-to-UART cable to provide 5V power to the ATxmega16D4 processor board while noting the polarity of the connector from Figure 3-1 since this connector is not keyed.

Warning: It is good practice to not supply power to the DC bus on the low voltage motor control board while updating firmware.

3. Connect the programmer to PC through USB connector and connect programmer ribbon cable to P1, match up pin 1 PCB, marked by a square pad, with pin 1 on ribbon connector since the connector is not keyed.
4. Bring up latest version of Atmel Studio, example is shown with Atmel Studio 6 at the time of this writing. If you are using a newer version of Atmel Studio, start a new C project using the GCC compiler option and select the ATxmega16D4. Add the downloaded version of AVR1636 - Firmware.c to the project and remove the project generated .c source file. This project is written for the GCC compiler as a single file: AVR1636 - Firmware.c so it is easily added into a newer project.
5. Select the GCC Compiler optimization to –O2.
6. Build the solution by pressing CTRL+SHIFT+B and check for compiling errors.
9. Download the firmware by pressing the start without debugging button.

The firmware should now be running. The EEPROM parameters can be set using the configuration utility.

3.3 Configuration utility

Once the hardware is set up and the firmware is present in the Atmel ATxmega16D4 flash memory you are ready to run the configuration utility.
3.3.1 Running the configuration utility

The configuration utility comes as a compressed zip file. The zip file should be extracted at the root directory (C:\). The folder contains the executable file *Atmel Motor Control Configuration Utility ver1_4_0.exe* along with support files. There is no Windows® installation, just run the .exe file.

A message in the variable plot window appears requesting that a COM port should be selected as shown in Figure 3-4. Use the drop down box to select the COM port.

**Figure 3-4.** COM port box before connection.

![Select COM port](image)

**Figure 3-5** shows an example of COM port four selected. If the COM port is unknown it can be found in the Windows start menu under Settings → Control Panel → System → Hardware → Device Manager → Ports (COM & LPT) you should see a USB serial port (COMx), where x is the port number. After selecting the COM port, the application configuration should be selected for AVR1636. This presents the proper variable and parameter list labels in the configuration utility.

![Select COM port and Application configuration](image)
3.3.2 Motor variables and variable plot

If the system is communicating with the configuration utility, the variables and parameters should fill up with values and the variable “counter” should be changing every 123ms. If the parameter data_watch (Figure 3-6) is set to 15, the counter variable should be displayed on the variable plot screen as a triangle waveform as shown in Figure 3-7.

If the variable plot is not displaying counter, then the counter function is selected to the variable plot screen by pressing the counter button, Figure 3-8.
The variable plot and updates of variables and parameters can be stopped using the radio buttons shown in Figure 3-9. This can be used to freeze data on the plot area, to read variables, and for capturing data to be saved to a file.

**Figure 3-9. Data updates.**

![Data updates:](image)

When selecting a variable for plotting the selected variable location in the list, name and description are displayed in the variable information window, Figure 3-10.

**Figure 3-10. Variable information displayed after selecting counter button.**

![Variable information:](image)

The complete list of 16 variables is shown in Figure 3-11. The configuration utility just displays raw 8-bit values and does not differentiate between signed and unsigned values. The first 11 variables are signed values in the firmware, but are transmitted to the utility with the zero point offset by 128, so that positive values are greater than 128 and negative values are less than 128.
Figure 3-11. The variable list.

The variables are described in Table 3-1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>v_a</td>
<td>Sinusoidal phase a PWM output.</td>
<td>Used to verify sinusoidal PWM pattern is being applied to motor on phase a.</td>
</tr>
<tr>
<td>v_b</td>
<td>Sinusoidal phase b PWM output.</td>
<td>Used to verify sinusoidal PWM pattern is being applied to motor on phase b.</td>
</tr>
<tr>
<td>i_a</td>
<td>Phase a measured current.</td>
<td>Helps verify current reconstruction of sinusoidal feedback current for phase a.</td>
</tr>
<tr>
<td>i_b</td>
<td>Phase b measured current.</td>
<td>Helps verify current reconstruction of sinusoidal feedback current for phase b.</td>
</tr>
<tr>
<td>u_a</td>
<td>Sinusoidal phase a voltage (scaled).</td>
<td>Used to verify sinusoidal voltage is being applied to motor on phase a.</td>
</tr>
<tr>
<td>ir_a</td>
<td>IR voltage drop for phase a.</td>
<td>Used to verify sinusoidal voltage drop is being produced from stator resistance on phase a.</td>
</tr>
<tr>
<td>l_didt</td>
<td>L x di/dt voltage drop for phase a.</td>
<td>Used to verify sinusoidal voltage drop is being produced from stator inductance on phase a.</td>
</tr>
<tr>
<td>e_a</td>
<td>Calculated back EMF for phase a.</td>
<td>Used to verify sinusoidal voltage is being produced from back EMF on phase a.</td>
</tr>
<tr>
<td>i_d</td>
<td>d-axis current aligned with air gap flux.</td>
<td>Used to verify current regulator is controlling flux producing component of current.</td>
</tr>
</tbody>
</table>
## Variable Description Use

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_q$</td>
<td>q-axis current perpendicular with air gap flux.</td>
<td>Used to verify current regulator is controlling torque producing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>component of current.</td>
</tr>
<tr>
<td>$e_d$</td>
<td>d-axis component of Back EMF.</td>
<td>Value driven to zero (=128) by back EMF PLL.</td>
</tr>
<tr>
<td>$v_{bus}$</td>
<td>DC input voltage.</td>
<td>Raw ADC sample of DC bus voltage. Using in calculation of $u_a$.</td>
</tr>
<tr>
<td>speed</td>
<td>Motor speed.</td>
<td>Used to watch motor speed especially useful in back EMF PLL tuning.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$speed$ is a signed number so 128 = zero speed. Positive values are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>above 128, negative numbers are below 128.</td>
</tr>
<tr>
<td>theta</td>
<td>Rotor angle</td>
<td>Integral of $speed$. Determines angle of air gap flux.</td>
</tr>
<tr>
<td>osc_error</td>
<td>Oscillator error</td>
<td>Used for testing oscillator calibration against USB-to-UART cable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$osc_error$ is a signed number so 128 = zero error. Positive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values are above 128, negative numbers are below 128. An invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value = 0 is displayed before an oscillator test has been performed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The test button must be pressed to display a valid result.</td>
</tr>
<tr>
<td>counter</td>
<td>Up/down counter.</td>
<td>Used to indicate connection between configuration utility and firmware.</td>
</tr>
</tbody>
</table>

### 3.3.3 Motor parameters

The motor parameters are EEPROM values copied into RAM on power up of the Atmel ATxmega16D4.

Figure 3-12 shows the list of parameters and the default values for the PMSM from Appendix C. If these values are not present they can be hand entered or loaded using the parameter $load$ button described later in this section.
Figure 3-12. The parameter list.

The parameters are described in Table 3-2.
Table 3-2. Parameter list description and use.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>v_bus_min</em></td>
<td>Minimum bus voltage to enable FOC.</td>
<td>Shuts off and resets all regulators below the threshold level to prevent integrator windup.</td>
</tr>
<tr>
<td><em>osc_cal_adj</em></td>
<td>Oscillator calibration adjustment.</td>
<td>Use to fine tune factor calibration. Values from 0 to 64 alter that calibration. Values above 64 are ignored using factory calibration. See section on oscillator calibration adjustment for more detail.</td>
</tr>
<tr>
<td><em>data_watch</em></td>
<td>Selected variable in variable plot window.</td>
<td>Used to display current selected variable or manually update. Variables are numbered 0 to 15 from top to bottom.</td>
</tr>
<tr>
<td><em>mode</em></td>
<td>Operation mode selection.</td>
<td>Selects between voltage, current, and speed mode.</td>
</tr>
<tr>
<td><em>input_cmd</em></td>
<td>Input command.</td>
<td>Used to manually send input command, interpretation of command depends on <em>mode</em>.</td>
</tr>
<tr>
<td><em>s_reg_rate</em></td>
<td>Speed regulator update rate in number of FOC loop updates.</td>
<td>Use to set speed regulator bandwidth below current regulator bandwidth by slowing down the update rate.</td>
</tr>
<tr>
<td><em>s_reg_scaling</em></td>
<td>Speed regulator scaling.</td>
<td>Sets the number of fractional bits used in speed regulator calculations.</td>
</tr>
<tr>
<td><em>s_error_limit</em></td>
<td>Speed error limit.</td>
<td>Clamps the speed regulator error.</td>
</tr>
<tr>
<td><em>s_k_scaling</em></td>
<td>Speed gain scaling.</td>
<td>Sets the number of fractional bits used in speed regulator gain calculations.</td>
</tr>
<tr>
<td><em>s_kp</em></td>
<td>Speed regulator proportional gain.</td>
<td>Set proportional gain of speed regulator.</td>
</tr>
<tr>
<td><em>s_ki</em></td>
<td>Speed regulator integral gain.</td>
<td>Set integral gain of speed regulator.</td>
</tr>
<tr>
<td><em>i_limit</em></td>
<td>Current limit.</td>
<td>Used to set the maximum current reference that is produced by the speed regulator output.</td>
</tr>
<tr>
<td><em>i_reg_scaling</em></td>
<td>Current regulator scaling.</td>
<td>Sets the number of fractional bits used in both Id and Iq regulator calculations.</td>
</tr>
<tr>
<td><em>i_error_limit</em></td>
<td>Current error limit.</td>
<td>Clamps the current regulator error.</td>
</tr>
<tr>
<td><em>i_k_scaling</em></td>
<td>Current gain scaling.</td>
<td>Sets the number of fractional bits used in current regulator gain calculations.</td>
</tr>
<tr>
<td><em>i_kp</em></td>
<td>Current regulator proportional gain.</td>
<td>Set proportional gain of current regulators.</td>
</tr>
<tr>
<td><em>i_ki</em></td>
<td>Current regulator integral gain.</td>
<td>Set integral gain of current regulators.</td>
</tr>
<tr>
<td><em>v_limit</em></td>
<td>Voltage limit.</td>
<td>Used to set the maximum voltage reference that is produced by the current regulator output.</td>
</tr>
<tr>
<td><em>speed_scale</em></td>
<td>Speed scaling.</td>
<td>Value sets the scaling speed and the motor RPM. Equations covered in the back EMF PLL section.</td>
</tr>
<tr>
<td><em>speed_min</em></td>
<td>Minimum speed.</td>
<td>Minimum speed of back EMF PLL. Set to reliably start up motor from rest.</td>
</tr>
<tr>
<td><em>speed_max</em></td>
<td>Maximum speed.</td>
<td>Maximum motor speed of back EMF PLL. Usually set 10% to 20% higher than no load speed of motor.</td>
</tr>
<tr>
<td><em>p_reg_scaling</em></td>
<td>Back EMF PLL regulator scaling.</td>
<td>Sets the number of fractional bits used in the back EMF PLL regulator calculations.</td>
</tr>
<tr>
<td><em>p_error_limit</em></td>
<td>Back EMF PLL error limit.</td>
<td>Clamps the back EMF PLL regulator error.</td>
</tr>
<tr>
<td><em>p_k_scaling</em></td>
<td>Back EMF PLL gain scaling.</td>
<td>Sets the number of fractional bits used in back EMF PLL regulator gain calculations.</td>
</tr>
<tr>
<td><em>p_kp</em></td>
<td>Back EMF PLL regulator proportional gain.</td>
<td>Set proportional gain of back EMF PLL regulator.</td>
</tr>
<tr>
<td><em>p_ki</em></td>
<td>Back EMF PLL regulator integral gain.</td>
<td>Set integral gain of back EMF PLL regulator.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Use</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>p_gravity</td>
<td>Back EMF PLL regulator gravity term.</td>
<td>Amount of negative error added into the back EMF PLL. Used to force the speed down to speed_min when the PLL is not locked onto the back EMF. This is used at startup or during a stall condition.</td>
</tr>
<tr>
<td>u_scaling</td>
<td>Phase voltage scaling parameter.</td>
<td>Used to set scaling of phase voltage in back EMF calculator. See section on back EMF Calculator.</td>
</tr>
<tr>
<td>r_scaling</td>
<td>Phase resistance scaling parameter.</td>
<td>Used to set scaling of phase resistance in back EMF calculator.</td>
</tr>
<tr>
<td>r_phase</td>
<td>Phase resistance.</td>
<td>Phase resistance, ohms per count determined by u_scaling and r_scaling.</td>
</tr>
<tr>
<td>l_scaling</td>
<td>Phase inductance scaling parameter.</td>
<td>Used to set scaling of phase inductance in back EMF calculator.</td>
</tr>
<tr>
<td>l_phase</td>
<td>Phase inductance.</td>
<td>Phase inductance, henries per count determined by u_scaling and l_scaling.</td>
</tr>
</tbody>
</table>

The copy all buttons shown in Figure 3-13 provides a quick method to transfer all the variables between the configuration utility and the Atmel ATxmega16D4.

**Figure 3-13. Copying all parameters.**

The white boxes on the far right column of Figure 3-12 are for editing and transferring data between the configuration utility and files. The file transfer is a simple file saves or file load (Figure 3-14) to a text file EEPROM.txt that is stored in the same directory that the configuration utility is launched. Backups of this file must be copied and renamed to prevent overwriting from the utility. The default EEPROM.txt from the download should contain the parameters values from Figure 3-12, this can be used to load the correct values.

**Figure 3-14. Parameter data load and save.**

To transfer the parameters stored in RAM back to the EEPROM the program button is pressed, this will ensure that the values are preserved and are used as the default values next time the application is powered up, Figure 3-15.
The parameter \textit{input\_cmd} is used for directly sending a voltage, current, or speed command. The \textit{input\_cmd} parameter is shown in Figure 3-16. The command can be directly modified using the input box on the right and then pressing the arrow key in the middle.

\textbf{Note:} This only transfers the value to the RAM of the Atmel ATxmega16D4 and the program button still needs to be pressed to commit the RAM values to EEPROM.

When the arrow key is pressed a description of the parameter shows up in the parameter information box, Figure 3-17.

\section*{3.3.4 Test run of motor}

Before running the motor the following check list should be completed:

1. Firmware is installed and running.
2. EEPROM parameters are the same as Figure 3-12.
3. The USB-to-UART cable is connected to the processor board and PC and communication is established with the configuration utility.
4. The motor leads are connected of J2.
5. The universal power supply (or a lab adjustable power supply) is connected to J3.

Now the universal power supply is ready to be plugged or the lab supply can be turned on and brought up to 12V. At this point the \textit{input\_cmd} is centered on 128 (zero command for a signed value). To start the motor input 158 (equal to +30) into the white box next to the \textit{input\_cmd} parameters and press the arrow button to update the parameter. The motor should start up in a clockwise direction. The motor can be reversed by entering 98 (equal to -30) and the arrow key pressed. The motor should reverse and run the same speed in a counter clockwise direction.
3.3.5 Oscillator calibration

The oscillator calibration can be checked by pressing the test button as shown in Figure 3-18. The internal oscillator is compared with the USB-to-UART cable oscillator which has an oscillator accuracy of better than 0.2%. See the Atmel ATxmega16D4 datasheet for further details about oscillator calibration.

Figure 3-18. Oscillator test button.

Once the calibration button is pressed the osc_error, Figure 3-19, variable is updated. The error is centered around 128. This means a value of 128 = zero error. A number above 128 means the oscillator is running slower than the reference value and a number below 128 means the oscillator is running faster than the reference value. When first launching the configuration utility, the osc_error value starts out with an invalid value of 0 so a test must be performed to get a valid reading.

Figure 3-19. Oscillator calibration error.

The oscillator calibration can be adjusted by updating the osc_cal_adj parameter, Figure 3-20, and programming the EEPROM. osc_cal_adj is a correction factor to the original factory calibration. The range of values is from 0 to 64, where 32 correspond to 0 correction. If the osc_error indicate that the oscillator is running too fast you want to adjust it down, by specifying an osc_cal_adj that is less than 32. Values from any value above 64 are ignored and only the factory calibration is used.

The equation for Oscillator cal adjustment for osc_cal_adj \(< 65:

\[
\text{adjusted osc_cal} = \text{osc_cal} + \text{osc_cal_adj} - 32
\]

Equation 3-1

Figure 3-20. Oscillator calibration adjustment parameter, example for -15, adjustment to factory calibration.

3.3.6 mode and input_cmd parameters

The mode parameter selects how the input_cmd parameter is used, Figure 3-21. Table 3-3 shows three modes of operation.

Figure 3-21. Mode and input_cmd parameter.
Table 3-3. Mode options.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Control mode</th>
<th>Variable affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Voltage mode (open loop PWM)</td>
<td>$v_d = 0, v_q = \text{input_cmd}$</td>
</tr>
<tr>
<td>1</td>
<td>Current control mode</td>
<td>$i_d = 0, i_q = \text{input_cmd}$</td>
</tr>
<tr>
<td>2</td>
<td>Speed control mode</td>
<td>$\text{speed_cmd} = \text{input_cmd}$</td>
</tr>
</tbody>
</table>

For mode = 0, voltage mode, the control blocks used are shown in Figure 3-22. Voltage mode directly maps into the $\text{input\_cmd}$ into the $v_q$ variable. This mode is useful for initial tuning of the back EMF calculator and the back EMF PLL, by eliminating any interaction with sensing and the speed and current regulators.

Figure 3-22. Mode = 0, voltage mode (open loop).

For mode = 1, current control mode, the control blocks used are shown in Figure 3-23. Current control mode set the current reference $i_q\_\text{ref}$ to the input command and sets $i_d\_\text{ref}$ to 0. The setting of $i_q\_\text{ref}$ produces a torque proportional to current, by regulation the torque producing component of stator current. The current regulators are proportional integral (PI) controllers that can be tuned after the back EMF calculator and back EMF PLL are tuned.
For **mode = 2**, speed control mode, the control blocks used are shown in **Figure 3-24**. Speed control mode sets the speed reference to the input command. The output of the speed regulator sets the motor torque by changing the $i_{q\_ref}$ based on speed error. The speed regulator is a proportional integral (PI) controller that can be tuned after the current regulators are tuned.
### 3.3.7 Minimum bus voltage

The $v_{bus\_min}$ sets a threshold for the DC input voltage ($v_{bus}$) above which the control loops are enabled. This prevents the speed and current regulators from winding up when the voltage is too low to properly control the motor (due to lack of back EMF feedback signal). The scaling for $v_{bus}$ and $v_{bus\_min}$ is based on the $v_{bus}$ divider and the ADC scaling:

$$v_{bus} = V_{DC} \times K_{bus}[\text{count}]$$  \hspace{1cm} \text{Equation 3-2}

Where:

- $V_{DC}$ = DC input voltage [volt]
- $K_{bus}$ = Scaling constant for variable $v_{bus}$ [count/volt]

$$K_{bus} = \frac{\text{Divider}_{bus} \times \text{Resolution}_{\text{ADC}}}{V_{ref}} \left[ \frac{\text{count}}{\text{volt}} \right]$$  \hspace{1cm} \text{Equation 3-3}

Where:

- $\text{Divider}_{bus}$ = Resistive divider on low voltage motor control board, Appendix B, R9 and R27.
- $\text{Resolution}_{\text{ADC}}$ = ADC resolution used for calculation [count]
- $V_{ref}$ = Voltage reference used for ADC [volt]

Using the values on the low voltage motor control board in Appendix B:

$$K_{bus} = \left( \frac{1k\Omega + 33.2k\Omega}{1V} \right) \times 256 \left[ \frac{\text{count}}{\text{volt}} \right]$$  \hspace{1cm} \text{Equation 3-4}

$$K_{bus} = 7.48 \left[ \frac{\text{count}}{\text{volt}} \right]$$  \hspace{1cm} \text{Equation 3-5}

An example of a 10.5V (78 count) threshold is shown in Figure 3-25.

**Figure 3-25. Enable voltage threshold set to 10.5V (78 count).**

### 3.3.8 Voltages and currents

The first two variables $v_{a}$ and $v_{b}$, see Figure 3-26, represent the output PWM values for phase a and phase b. Full PWM duty cycle is set at 255 and zero percent duty cycle is set at 0. Zero output voltage is when all three phases (a, b and c) are at 50% duty cycle or all three PWM values are set at 128. The actual output voltage is dependent on the DC input voltage so $v_{a}$ and $v_{b}$ are in counts. Variable $v_{c}$ is not brought out since it is the negative sum of $v_{a}$ and $v_{b}$.

$$v_{a}[\text{count}]$$  \hspace{1cm} \text{Equation 3-6}

**Figure 3-26. Variables $v_{a}$ and $v_{b}$.**
The variables $i_a$ and $i_b$, Figure 3-27, represent the measure phase current from single shunt current reconstruction for phase a and phase b. The scaling for current is based on the current shunt value, external divider, and ADC settings.

$$i_a = I_a \times K_{\text{current}} \text{[count]}$$  \hspace{1cm} \text{Equation 3-7}

Where:
- $I_a = \text{Phase a current [ampere]}$
- $K_{\text{current}} = \text{Scaling constant for variable } i_a, i_b \text{ and } i_c \text{ [counts/ampere]}

$$K_{\text{current}} = \frac{R_{\text{shunt}} \times \text{Divider}_{\text{current}} \times \text{Resolution}_{\text{ADC}}}{V_{\text{ref}}} \text{[count/ampere]}$$  \hspace{1cm} \text{Equation 3-8}

Where:
- $R_{\text{shunt}} = \text{Current sensing resistor on low voltage motor control board, Appendix B, R1 [Ω]}
- \text{Divider}_{\text{current}} = \text{Resistive divider used for biasing ground referenced } R_{\text{shunt}} \text{ voltage above ground on low voltage motor control board, Appendix B, R29 and R14}
- \text{Resolution}_{\text{ADC}} = \text{ADC resolution used for calculation [count]}

For the kit values:

$$K_{\text{current}} = \frac{0.05Ω \times \left(\frac{10kΩ}{1kΩ + 10kΩ}\right) \times 256}{V_{\text{ref}}} \text{[count/ampere]}$$  \hspace{1cm} \text{Equation 3-9}

$$K_{\text{current}} = 11.6 \text{ [count/ampere]}$$  \hspace{1cm} \text{Equation 3-10}

Figure 3-27. Variables $i_a$ and $i_b$.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_a$</td>
<td>128</td>
</tr>
<tr>
<td>$i_b$</td>
<td>128</td>
</tr>
</tbody>
</table>

### 3.3.9 Back EMF calculator

The instantaneous value of motor back EMF is calculated for each phase using stator quantities of voltage, current, phase resistance, and phase inductance. The calculations in firmware uses scaled values in counts:

$$e_a = u_a - i_a \times r_a - l_a \times \frac{di_a}{dt} \text{[count]}$$  \hspace{1cm} \text{Equation 3-11}

The physical equation for back EMF in SI units is:

$$E_a = U_a - I_a \times R_{\text{phase}} - L_{\text{phase}} \times \frac{di_a}{dt} \text{[volt]}$$  \hspace{1cm} \text{Equation 3-12}

Where:
- $E_a = \text{Phase a back emf [volt]}
- U_a = \text{Phase a voltage [volt]}
- I_a = \text{Phase a current [ampere]}
- R_{\text{phase}} = \text{Phase resistance [ohm]}
- L_{\text{phase}} = \text{Phase inductance [Henry]}

The back EMF calculator for one of three phases with parameters is shown in Figure 3-28.
The parameters in the back EMF calculator are shown in 0. These values need proper adjustment and scaling for calculating the back EMF.

The scaled phase voltage, $u_a$, is determined by the following equation:

$$u_a = \left( \frac{v_a \times v_{bus}}{2u_{scaling}} \right) [\text{count}]$$  \hspace{1cm} \text{Equation 3-13}

$$u_a = U_a \times K_u [\text{count}]$$  \hspace{1cm} \text{Equation 3-14}

Where:

- $K_u = \text{Scaling constant for variable } u_a [\text{count/volt}]$
- $U_a = \text{Phase a voltage [volt]}$

$$K_u = \frac{K_{bus} \times PWM_{max}}{2u_{scaling}} [\text{count}]$$  \hspace{1cm} \text{Equation 3-15}

Where:

- $K_{bus} = \text{Scaling constant for variable } v_{bus} [\text{count/volts}]$
- $PWM_{max} = \text{Maximum count for PWM timer [count]}$
- $2u_{scaling} = \text{Scaling factor for voltage } u_a, u_b, \text{ and } u_c [\text{count}]$

Using the example values:

$$K_u = \frac{7.48 \times 256}{27} \text{ [count/volt]}$$  \hspace{1cm} \text{Equation 3-16}

$$K_u = 14.96 \text{ [count/volt]}$$  \hspace{1cm} \text{Equation 3-17}
The scaled resistive drop, $ir_a$, is determined by the following equation:

$$ir_a = \frac{(i_a \times r_{phase})}{2r_{scaling}}[\text{count}]$$  \hspace{1cm} \text{Equation 3-18}

$$ir_a = I_a \times R_{phase} \times K_r[\text{count}]$$  \hspace{1cm} \text{Equation 3-19}

$$r_{phase} = R_{phase}[\text{ohm}] \times K_r[\text{count}]$$  \hspace{1cm} \text{Equation 3-20}

Where:

$K_r$ = Scaling constant for variable $r_{phase}$ [count/ohm]

$$K_r = \frac{K_u \times 2^r_{scaling}}{K_{current}}[\text{count}]$$  \hspace{1cm} \text{Equation 3-21}

Where:

$2^{r_{scaling}} = Scaling \ factor \ for \ r_{phase}$ [count]

Using the example values:

$$K_r = \frac{14.96 \times 2^7}{11.6}[\text{count}]$$  \hspace{1cm} \text{Equation 3-22}

$$K_r = 165[\text{count}]$$  \hspace{1cm} \text{Equation 3-23}

The motor datasheet value from Appendix C for line to line resistance is divided by 2:

$$R_{phase} = \frac{0.44}{2}[\text{ohm}]$$  \hspace{1cm} \text{Equation 3-24}

$$r_{phase} = 0.22[\text{ohm}] \times 165[\text{count}]$$  \hspace{1cm} \text{Equation 3-25}

$$r_{phase} = 36[\text{count}]$$  \hspace{1cm} \text{Equation 3-26}

The scaled inductive drop, $l_{didt_a}$, is determined by the following equation:

$$l_{didt_a} = \frac{l_{phase} \times (i_a - i_{a \_last})}{2^{l_{scaling}}}[\text{count}]$$  \hspace{1cm} \text{Equation 3-27}

$$l_{didt_a} = L_{phase} \times \frac{di_a}{dt} \times K_u[\text{count}]$$  \hspace{1cm} \text{Equation 3-28}

$$l_{phase} = L_{phase}[\text{Henry}] \times K_l[\text{Henry}]$$  \hspace{1cm} \text{Equation 3-29}

Where:

$K_l$ = Scaling constant for variable $l_{phase}$ [count/Henry]
\[ K_i = \frac{K_u \times 2^{\text{r_scaling}} \times \text{freq}_{\text{loop}} \text{[count]}}{K_{\text{current}} \text{[Henry]}} \]  
\text{Equation 3-30}

Where:
- \(2^{\text{r_scaling}} = \text{Scaling factor for r_phase [count]}\)
- \(\text{freq}_{\text{loop}} = \text{Frequency of FOC control loop [hertz]}\)

Using the example values:

\[ K_i = \frac{14.96 \times 2^4 \times 7812.5\text{Hz} \text{[count]}}{11.6 \text{[Henry]}} \]  
\text{Equation 3-31}

\[ K_i = 161200 \text{[count/ Henry]} \]  
\text{Equation 3-32}

The motor datasheet value from Appendix C for line to line inductance is divided by 2:

\[ L_{\text{phase}} = \frac{0.000510}{2} \text{[Henry]} \]  
\text{Equation 3-33}

\[ L_{\text{phase}} = 0.000255 \text{[Henrys]} \times 161200 \text{[count/ Henry]} \]  
\text{Equation 3-34}

\[ L_{\text{phase}} = 41 \text{[count]} \]  
\text{Equation 3-35}

Figure 3-29. Back EMF calculation parameters.

<table>
<thead>
<tr>
<th>u_scaling</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_scaling</td>
<td>7</td>
</tr>
<tr>
<td>l_phase</td>
<td>36</td>
</tr>
<tr>
<td>L_scaling</td>
<td>4</td>
</tr>
<tr>
<td>L_phase</td>
<td>41</td>
</tr>
</tbody>
</table>

Phase a back EMF values are brought out as display variables to aid in tuning the parameters, Figure 3-30. These values are signed numbers, so they are centered on the value 128 for display on the configuration utility.

Figure 3-30. Back EMF calculation variables for phase a.

<table>
<thead>
<tr>
<th>u_a</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_r_a</td>
<td>128</td>
</tr>
<tr>
<td>L_didt</td>
<td>128</td>
</tr>
<tr>
<td>e_a</td>
<td>128</td>
</tr>
</tbody>
</table>

The three phase back EMF goes through a Clarke and Park transform, Figure 3-31, for feedback to the back EMF phase locked loop.
The d-axis component of back EMF, \( e_d \) is driven to zero by the back EMF PLL; this aligns the air gap flux with the d-axis and all the back EMF is generated on the q-axis. Since \( e_d \) is a signed value it is centered on 128 for displaying on the configuration utility. Figure 3-32 shows \( e_d \) centered at 128.

### 3.3.10 Back EMF sensing PLL

The back EMF phase locked loop is regulated by a PI controller where \( e_d \) is the feedback and speed is the output. The block diagram of the calculator is shown in Figure 3-33.
Figure 3-33. Back EMF PLL block diagram with parameters.

The parameters from the block diagram are listed in Figure 3-34. The parameters for the PLL are as follows:

1. **speed_min** and **speed_max** – speed range of the back EMF PLL.
2. **p_reg_scaling** – sets the number of fractional bits used in calculation, this can range from 0 to 7, where 5 is a good starting point.
3. **p_error_limit** – clamps the amount of error that goes into the PI controller. Typical range is 8 to 32.
4. **p_k_scaling** – sets the number of fractional bits used in calculation of the PI gains.
5. **p_kp** – the proportional gain.
6. **p_ki** – the integral gain.
7. **p_gravity** – allows a small offset to force the PLL frequency to **speed_min** if PLL is not locked onto the back EMF. This helps with startup of the motor. Typical values are 0 to 2.

Figure 3-34. Back EMF PLL parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed_min</td>
<td>1</td>
</tr>
<tr>
<td>speed_max</td>
<td>255</td>
</tr>
<tr>
<td>p_reg_scaling</td>
<td>5</td>
</tr>
<tr>
<td>p_error_limit</td>
<td>8</td>
</tr>
<tr>
<td>p_k_scaling</td>
<td>4</td>
</tr>
<tr>
<td>p_kp</td>
<td>160</td>
</tr>
<tr>
<td>p_ki</td>
<td>16</td>
</tr>
<tr>
<td>p_gravity</td>
<td>1</td>
</tr>
</tbody>
</table>
Tuning a PLL regulator typically is done in \textit{mode} = 0, Voltage mode, with an \textit{input_cmd} fairly low so that excessive current does not flow in the motor at stall. The \textit{input_cmd} value used on this motor is set to 158 (30 above the zero point of 128). Also the back EMF calculator must first have its values calculated and entered. Tuning the PLL is finding the \textit{p\_kp} and \textit{p\_ki} values that give the motor quick startup and smooth steady state performance. This is done by watching the \textit{speed} variable in the Variable plot window and changing the gain values. It is best to start with setting typical values for scaling and limits and setting the integral gain \textit{i\_ki} to a low value such as 1 or 2. Then increase the \textit{i\_kp} gain so that the PLL tracks the motor speed.

### 3.3.11 Speed and theta

Speed is integrated into the angle of the air-gap flux. Integration is achieved by accumulating speed every FOC loop update of 128µs, Figure 3-35.

**Figure 3-35. The speed to theta integrator.**

The result of the accumulation is scaled by the parameter \textit{speed\_scale} Figure 3-36. This scaling sets the frequency range of the sine wave output voltages \textit{v\_a}, \textit{v\_b}, and \textit{v\_c} and as a result sets the speed range of the motor.

**Figure 3-36. Speed scaling parameter.**

<table>
<thead>
<tr>
<th>speed_scale</th>
<th>4</th>
</tr>
</thead>
</table>

The relationship between \textit{speed\_scale} and frequency is:

\[
rpm = speed \times K_{rpm}[R.P.M.]\]

Equation 3-36

\[
K_{rpm} = \left( \frac{freq\_loop[\text{hertz}]}{theta_{max}[\text{count}] \times 2^{\text{speed\_scale}}} \right) \left( \frac{60[\text{sec}]\times 2}{N_{poles}\text{[count]}} \right) [R.P.M.]
\]

Equation 3-37

Where:

\begin{align*}
\text{Speed} & = \text{speed\_min or speed\_max [unitless] (between 1 and 255)} \\
\text{freq\_loop} & = \text{Frequency of FOC control loop [hertz]} \\
\text{theta\_max} & = \text{Maximum count for theta equal to 360 degrees [count]} \\
2^{\text{speed\_scale}} & = \text{Scaling factor for speed [counts]} \\
N_{poles} & = \text{Number of motor poles from motor datasheet, Appendix C}
\end{align*}
Atmel AVR1636: Configurable PMSM Sensorless Field Oriented Control using the XMEGA

\[ K_{rpm} = \left( \frac{7812.5 \text{ hertz}}{256 \times 2^4} \right) \left( \frac{60 \text{ sec}}{\text{min}} \right) \left( \frac{2}{8} \right) \left( \frac{\text{R.P.M.}}{\text{count}} \right) \]  
\hspace{1cm} \text{Equation 3-38}

\[ K_{rpm} = 28.6 \left( \frac{\text{R.P.M.}}{\text{count}} \right) \]  
\hspace{1cm} \text{Equation 3-39}

The minimum and maximum speed can be calculated replacing in \textit{speed\_min} and \textit{speed\_max}:

\[ \text{rpm}_{\text{min}} = \text{speed\_min} \times K_{rpm}[\text{R.P.M.}] \]  
\hspace{1cm} \text{Equation 3-40}

\[ \text{rpm}_{\text{min}} = 28.6 \times [\text{R.P.M.}] \]  
\hspace{1cm} \text{Equation 3-41}

\[ \text{rpm}_{\text{max}} = \text{speed\_max} \times K_{rpm}[\text{R.P.M.}] \]  
\hspace{1cm} \text{Equation 3-42}

\[ \text{rpm}_{\text{max}} = 7293 \times [\text{R.P.M.}] \]  
\hspace{1cm} \text{Equation 3-43}

It is important to choose a \textit{speed\_scale} value that ensures that the maximum speed of the motor can be reached. For the motor included with the kit, the range is well within the no-load speed of the motor rated at 4000 [R.P.M] from the motor datasheet in Appendix C.

3.3.12 Current regulators

There are two current regulators, one for the d-axis current and one for the q-axis current. The d-axis is aligned with the air gap flux produced by the rotor magnets. Current along this axis can add to or subtract from the air gap flux, this is typically used for field weakening to extend the speed range of the motor. Field weakening is not implemented so the command is set to zero. \textit{Figure 3-37} shows the current regulator for the d-axis current, \textit{i\_d} and the reference set to zero.
The q-axis is perpendicular to the air gap flux produced by the rotor magnets. Current along this axis produces torque. Figure 3-38 shows the current regulator for the q-axis current, $i_q$, and the reference set to $input_{cmd}$ for $mode = 1$. For $mode = 2$ the $i_q_{ref}$ is set by the output of the speed regulator.
The two current regulators share the same gain settings. The parameters from the block diagrams are listed in Figure 3-39. The parameters for the current regulators are as follows:

1. \textit{i\_reg\_scaling} – sets the number of fractional bits used in calculation, this can range from 0 to 7, where 5 is a good starting point.
2. \textit{i\_error\_limit} – clamps the amount of error that goes into the PI controller. Typical range is 8 to 32.
3. \textit{i\_k\_scaling} – sets the number of fractional bits used in calculation of the PI gains.
4. \textit{i\_kp} – the proportional gain.
5. \textit{i\_ki} – the integral gain.
6. \textit{v\_limit} – set the maximum value for \(v_d\) and \(v_q\), the outputs of the regulators.

Tuning a current regulator typically starts with setting typical values for scaling and limits and setting the \textit{i\_ki} = 0 to turn off the integral gain. Then increasing the \textit{i\_kp} gain so that the current tracks the reference, but with a DC offset. Then a small amount of integral gains is added in until the DC offset is eliminated.
3.3.13 Speed regulator

The speed regulator is used to modulate the torque command until the speed error is reduced to zero. The speed regulator is a PI regulator using the same structure as the back EMF PLL and the current regulators. When $\text{mode} = 2$ the $\text{input.cmd}$ sets the speed reference as shown in Figure 3-40.

Figure 3-40. Speed regulator with parameters.

The speed regulator parameters from the block diagrams are listed in Figure 3-41. The parameters for the speed regulator are as follows:

1. $\text{s.reg.rate}$ – sets the number of current control loop updates before the speed regulator is updated, this is due to the much lower bandwidth of the speed loop vs. the current loop. A good rule of thumb is that they should be an order of magnitude apart. Choosing a value in the range from 10 to 50 is a good starting point.
2. $\text{s.reg.scaling}$ – sets the number of fractional bits used in calculation, this can range from 0 to 7, where 5 is a good starting point.
3. $\text{s.error.limit}$ – clamps the amount of error that goes into the PI controller. Typical range is 8 to 32.
4. $\text{s.k.scaling}$ – sets the number of fractional bits used in calculation of the PI gains.
5. $\text{s.kp}$ – the proportional gain.
6. $\text{s.ki}$ – the integral gain.
7. $\text{i.limit}$ – set the maximum value $i.q.ref$, the outputs of the regulator.
Tuning a speed regulator typically starts with setting typical values for scaling and limits and setting the $i_{ki} = 0$ to turn off the integral gain. Then increasing the $i_{kp}$ gain so that the speed tracks the reference, but with a DC offset. Then a small amount of integral gains is added in until the DC offset is eliminated.

4. Conclusion

This application note described the implementation and use of a configurable 3-phase permanent magnet synchronous motor (PMSM) under sensorless field oriented control using the Atmel ATxmega16D4 microcontroller. The complete hardware system includes a 12V 3-phase PMSM, an ATxmega16D4 processor board, a low voltage motor control board, a USB-to-UART bridge cable, an AC power adapter, the firmware running on the ATxmega16D4, and a PC based configuration utility for customizing the sensorless FOC for the motor.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_reg_rate</td>
<td>20</td>
</tr>
<tr>
<td>s_reg_scaling</td>
<td>5</td>
</tr>
<tr>
<td>s_error_limit</td>
<td>16</td>
</tr>
<tr>
<td>s_k_scaling</td>
<td>5</td>
</tr>
<tr>
<td>s_kp</td>
<td>200</td>
</tr>
<tr>
<td>s_ki</td>
<td>8</td>
</tr>
<tr>
<td>i_limit</td>
<td>30</td>
</tr>
</tbody>
</table>
Appendix A. Atmel ATmega16D4 Processor Board Schematics
Appendix B. Low Voltage Motor Control Board Schematics
Appendix C. 42BLS01-001 Motor Datasheet
### Appendix D. Kit Parts List

<table>
<thead>
<tr>
<th>Description</th>
<th>Manufacturer</th>
<th>Part number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Voltage Motor Control Board</td>
<td>Atmel Corporation</td>
<td></td>
</tr>
<tr>
<td>ATxmega16D4 Processor Board</td>
<td>Atmel Corporation</td>
<td></td>
</tr>
<tr>
<td>12V 4000RPM</td>
<td>Tecmotion</td>
<td>42BLS01-001</td>
</tr>
<tr>
<td>USB-to-UART Bridge Cable</td>
<td>FTDI, Future Technology Devices International Ltd</td>
<td>TTL-232R-5V</td>
</tr>
</tbody>
</table>
## Appendix E. Revision History

<table>
<thead>
<tr>
<th>Doc. Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>42061A</td>
<td>01/2013</td>
<td>Initial document release</td>
</tr>
</tbody>
</table>