APPLICATION NOTE

Atmel AVR1636: 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 Voltages 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.

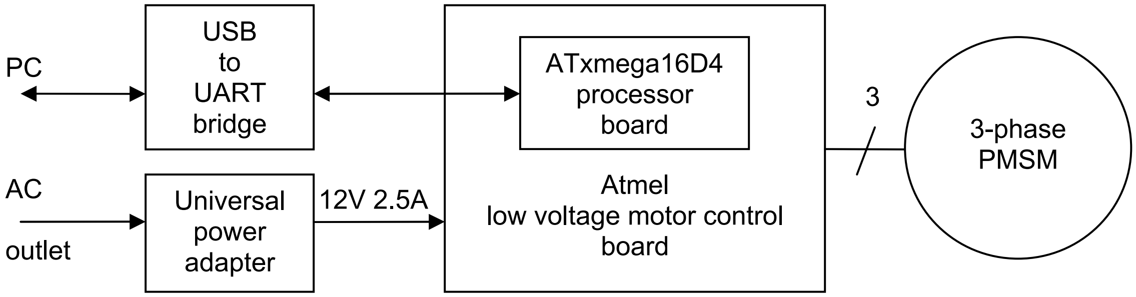
Configuring consists of setting parameters that are known or calculated quantities. These quantities translate into parameters that are set in the EEPROM. Tuning involves an iterative process of setting and testing parameters to find the optimal value. These values are not easily calculated or involve 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 if Figure 2-1.

Figure 2-1. Hardware block diagram.

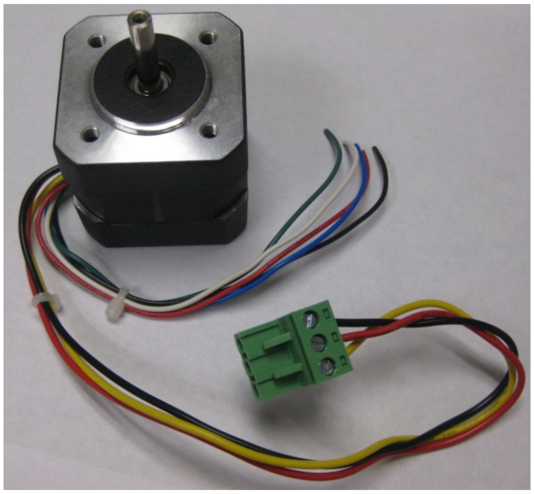


2.1.1 3-phase PMSM

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

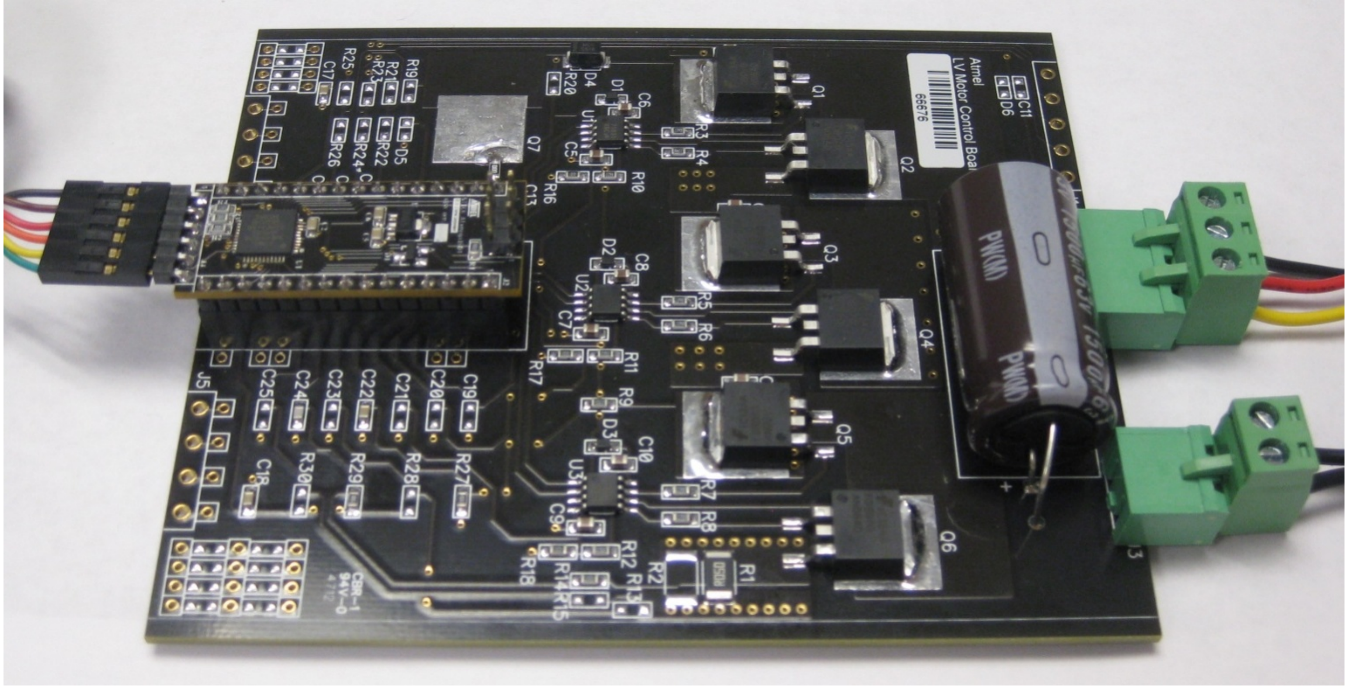
The low voltage motor control board supports all the power and level shifting circuitry for 3-phase motor control for various motor types and control techniques.

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

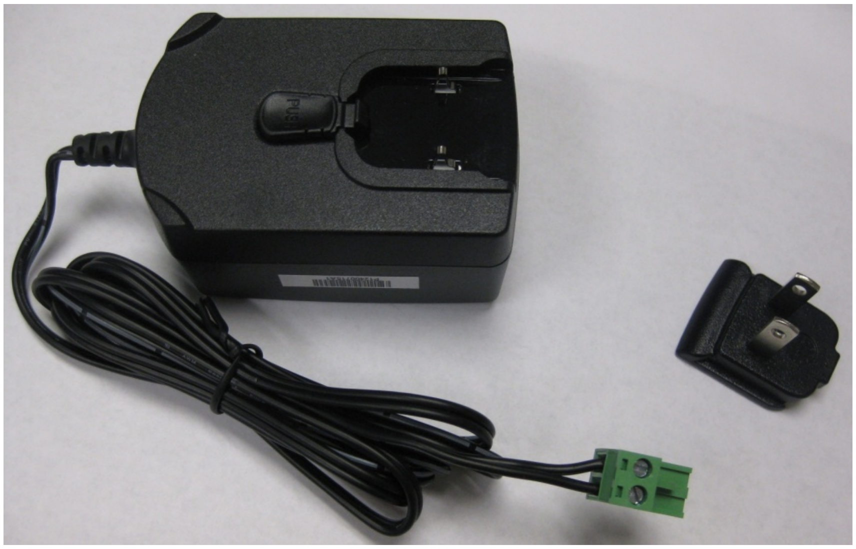
The USB-to-UART bridge cable provides communication between a PC and the ATxmega16D4 processor board. This cable is a standard product from FTDI. Part numbers are listed in Appendix D.

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

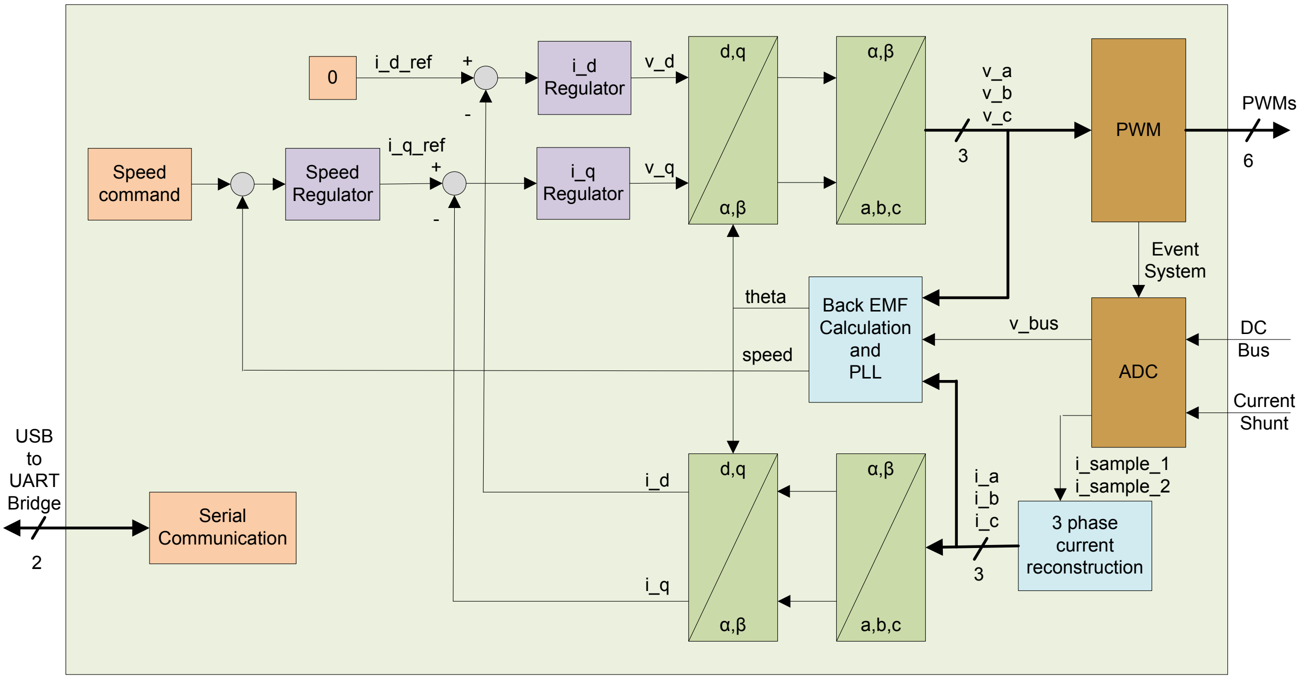
The firmware was designed to run on the Atmel ATxmega16D4 to demonstrate a configurable motor control application driving 3-phase PMSM with sensorless FOC.

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

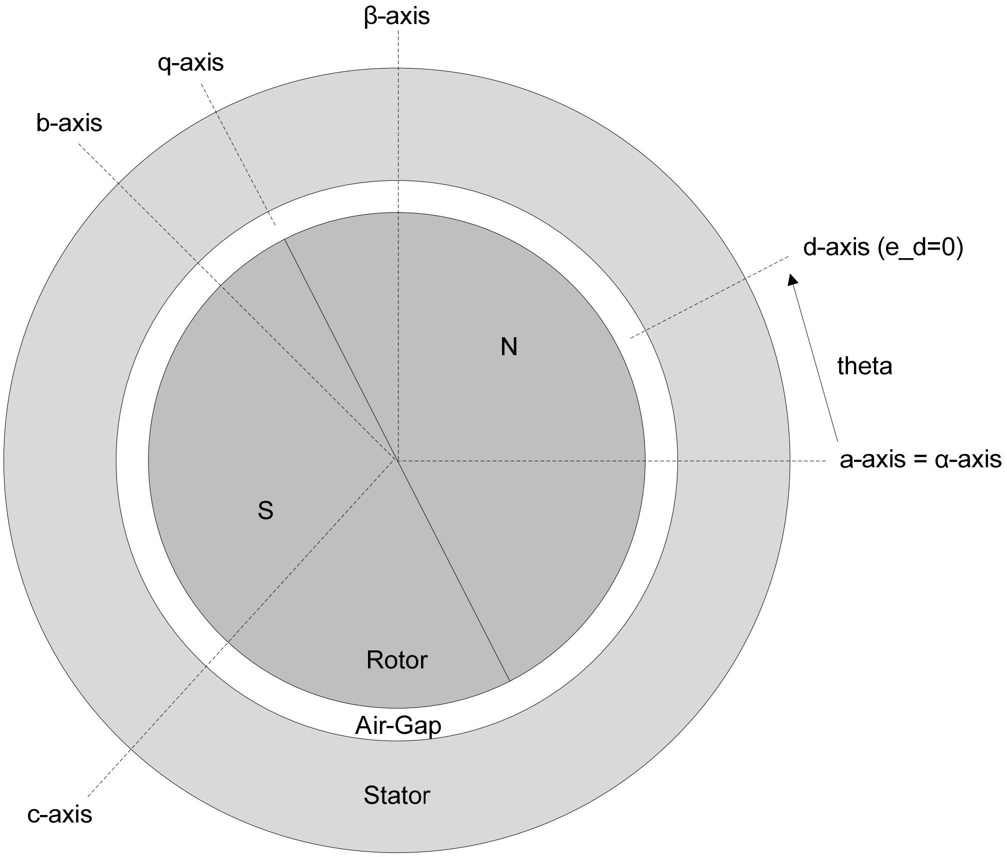
The sensorless field oriented control is based on measurement of stator current and transforming these quintiles to a reference frame rotating with the air gap flux linkage for independent control of the flux producing current (d-axis) and the torque producing current (q-axis). A block diagram of this 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.

Figure 2-6. Control structure.



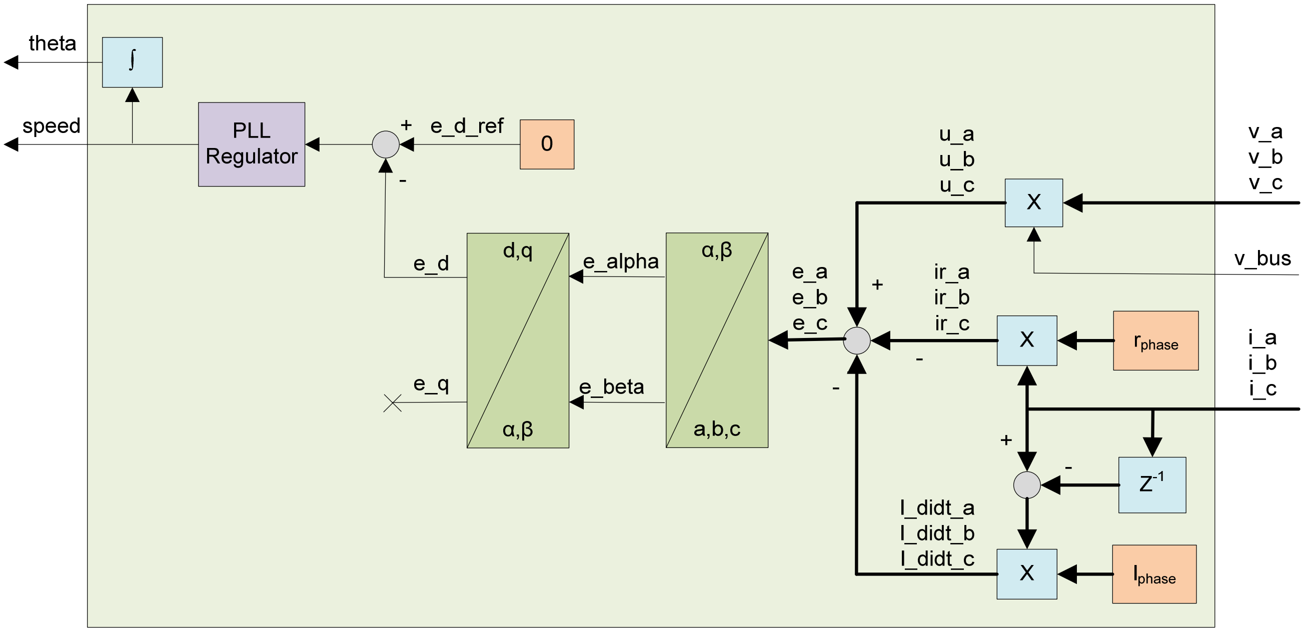
The field orientation to the air gap flux linkage is established by coordinate transformation (Clark transform) and vector rotation (Park transformation) using the correct angle (theta) between a fixed reference point on the motor's stator and the actual air gap flux represented as a vector. This can be visualized by Figure 2-7.

Figure 2-7. Air-gap flux linkage angle, theta.



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.

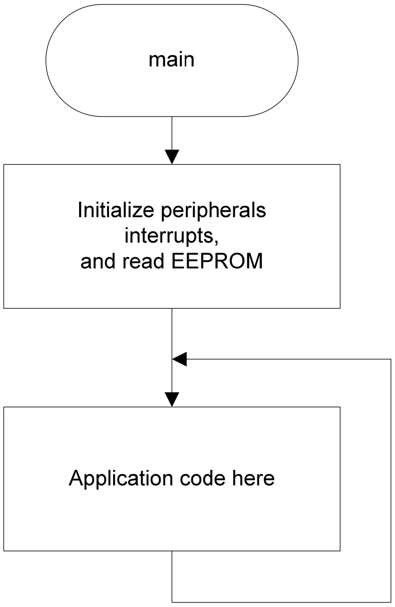
Figure 2-8. Back EMF calculation and PLL.



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.

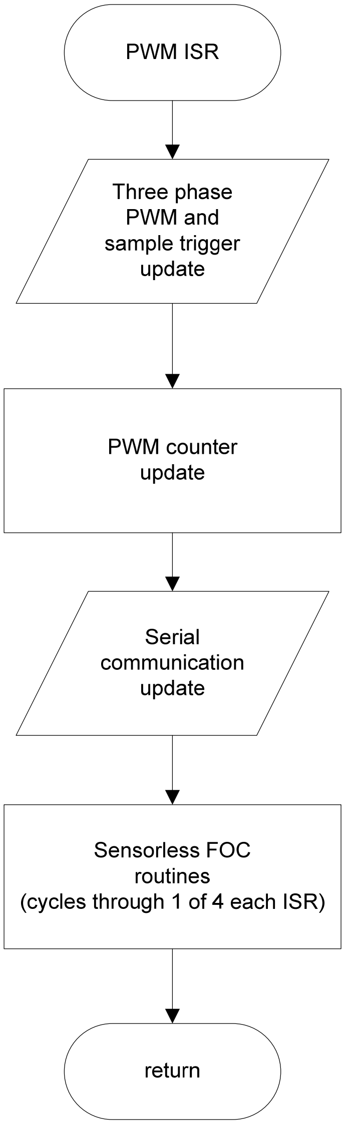
Figure 2-9. Firmware main loop.



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.

Figure 2-10. PWM interrupt service routine.

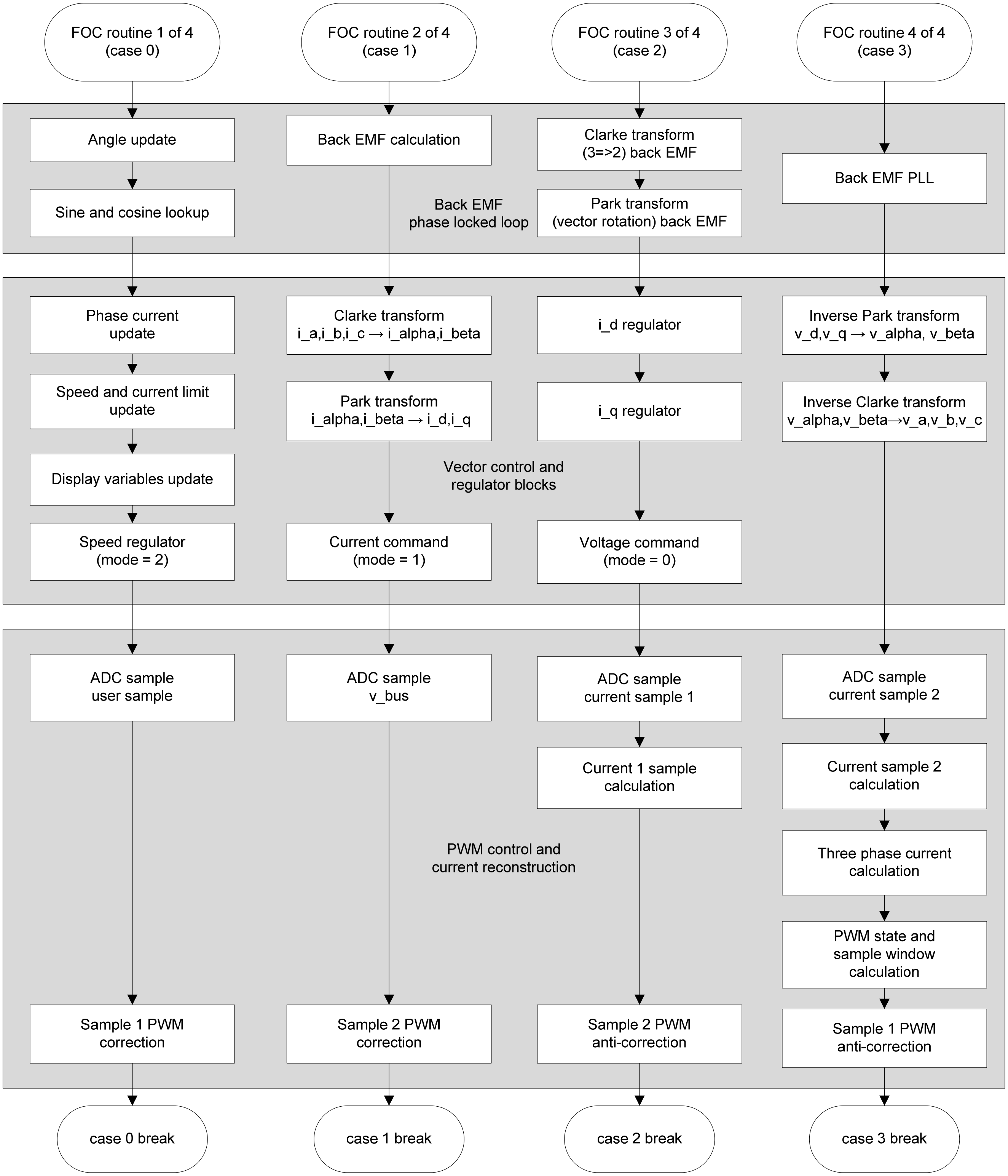

PWM ISR
Three phase PWM and sample trigger update
PWM counter update
Serial communication update
Sensorless FOC routines (cycles through 1 of 4 each ISR)
return

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

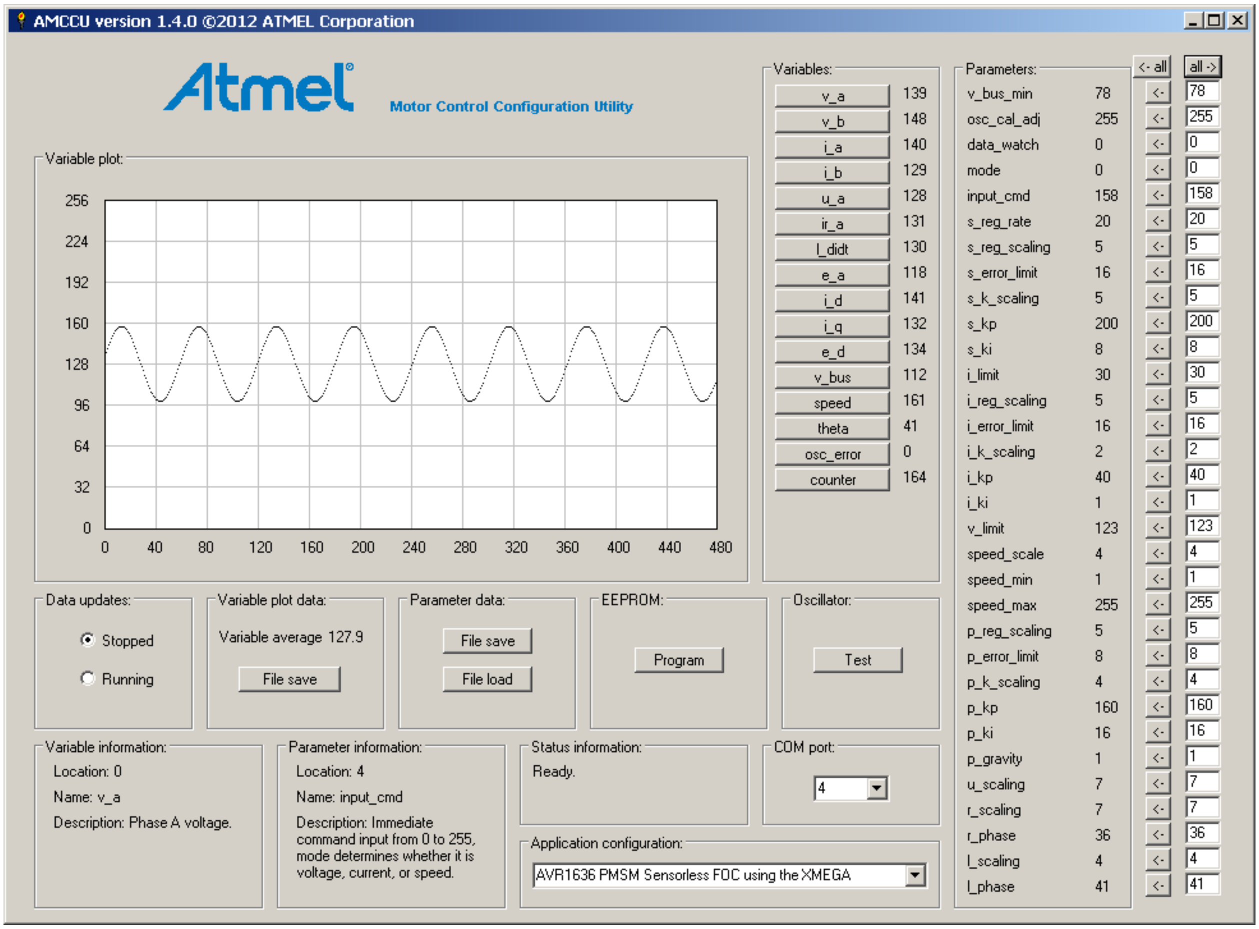
The PC based configuration utility is for configuration and tuning of the motor performance using a virtual COM port on the PC. The utility provides the following features:

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 window 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.

Figure 2-12. Configuration utility layout.



Data updates: ● Stopped ○ Running

Variable plot data: Variable average 127.9 [File save]

Variables:

v_a	139
v_b	148
i_a	140
i_b	129
u_a	128
ir_a	131
I_didt	130
e_a	118
i_d	141
i_q	132
e_d	134
v_bus	112
speed	161
theta	41
osc_error	0
counter	164

Parameters:

v_bus_min	78	<-	78
osc_cal_adj	255	<-	255
data_watch	0	<-	0
mode	0	<-	0
input_cmd	158	<-	158
s_reg_rate	20	<-	20
s_reg_scaling	5	<-	5
s_error_limit	16	<-	16
s_k_scaling	5	<-	5
s_kp	200	<-	200
s_ki	8	<-	8
i_limit	30	<-	30
i_reg_scaling	5	<-	5
i_error_limit	16	<-	16
i_k_scaling	2	<-	2
i_kp	40	<-	40
i_ki	1	<-	1
v_limit	123	<-	123
speed_scale	4	<-	4
speed_min	1	<-	1
speed_max	255	<-	255
p_reg_scaling	5	<-	5
p_error_limit	8	<-	8
p_k_scaling	4	<-	4
p_kp	160	<-	160
p_ki	16	<-	16
p_gravity	1	<-	1
u_scaling	7	<-	7
r_scaling	7	<-	7
r_phase	36	<-	36
l_scaling	4	<-	4
l_phase	41	<-	41

Variable information:
Location: 0
Name: v_a
Description: Phase A voltage.

Parameter information:
Location: 4
Name: input_cmd
Description: Immediate command input from 0 to 255, mode determines whether it is voltage, current, or speed.

Status information:
Ready.

COM port: 4

Application configuration: AVR1636 PMSM Sensorless FOC using the XMEGA

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

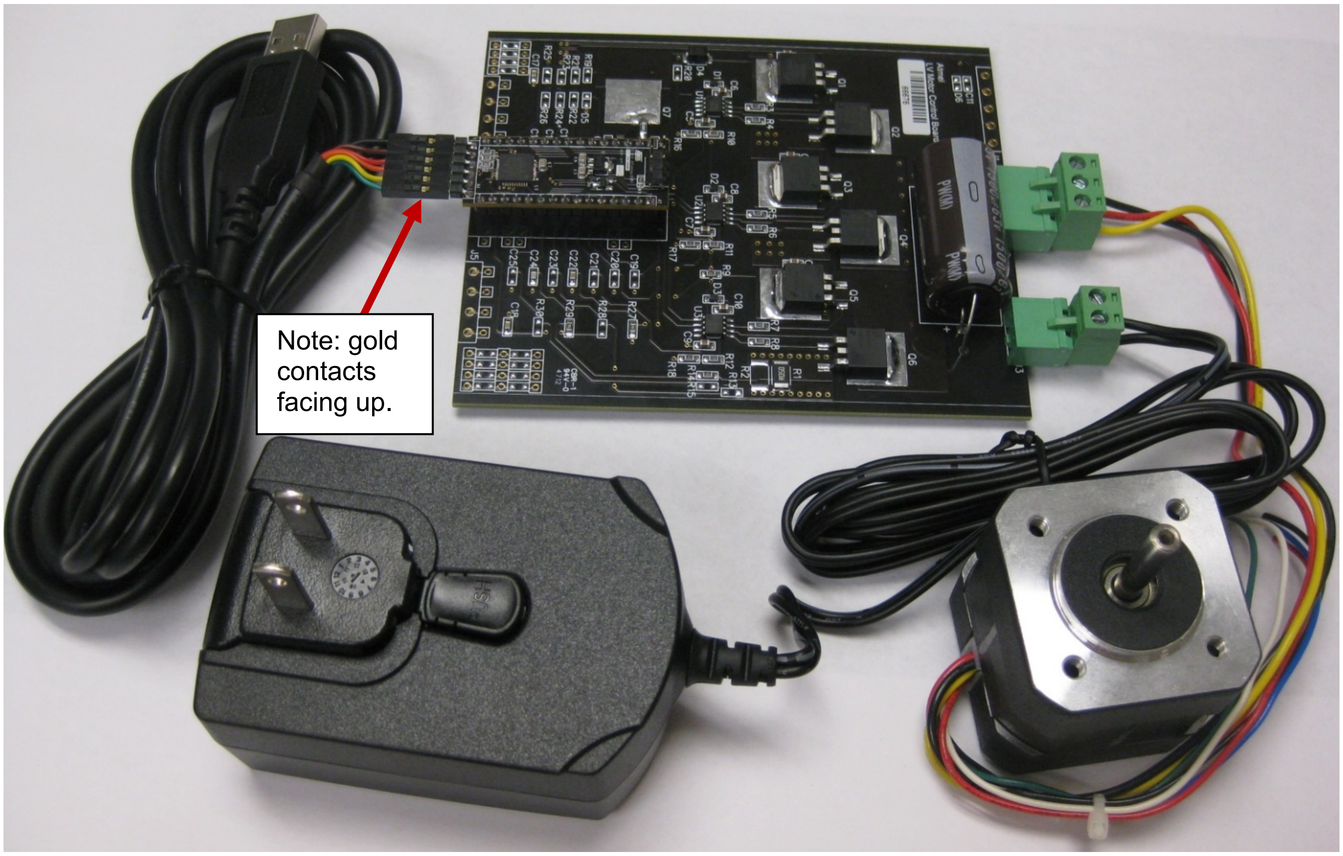
Additional file functions are there for configuring the utility, loading and saving parameters values, and saving variable plot data.

3. Running the system

3.1 Hardware

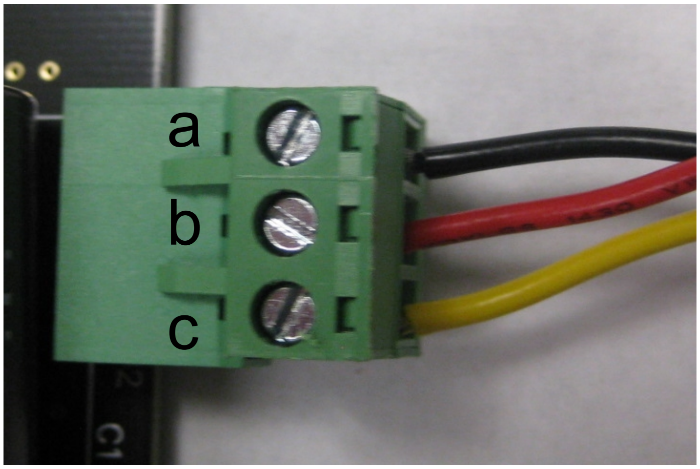
The system is connected as shown in Figure 3-1. Initially the Atmel ATxmega16D4 processor board can be powered directly from the USB port, this powers up the ATxmega16D4 allowing communication with the configuration utility without running the motor.

Figure 3-1. Configurable 3-phase PMSM with a USB-to-UART bridge and universal power adapter connected.



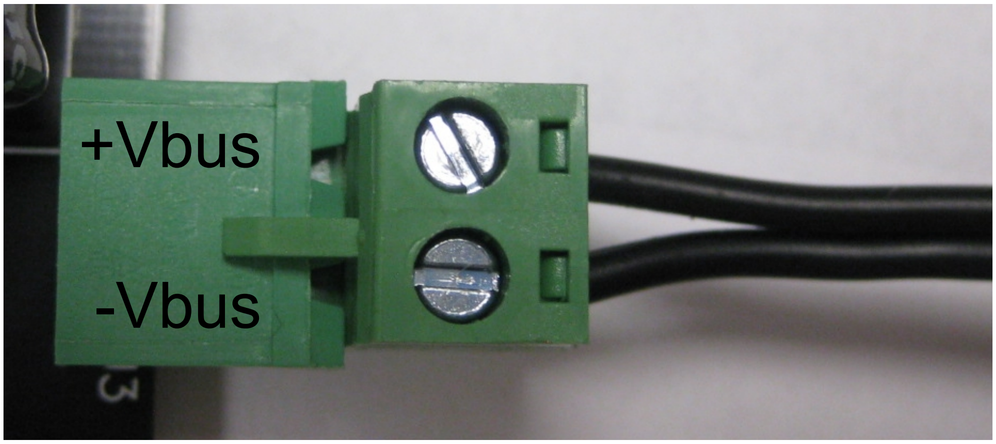
The PMSM phase leads are connected to the 3-pin connector J2. The motor phase leads are the 20 AWG black, red, and yellow wires from the motor. The five 28 AWG wires are Hall sensors connections not used in this application, so they should be left unconnected. Connecting the motor phase leads in the 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

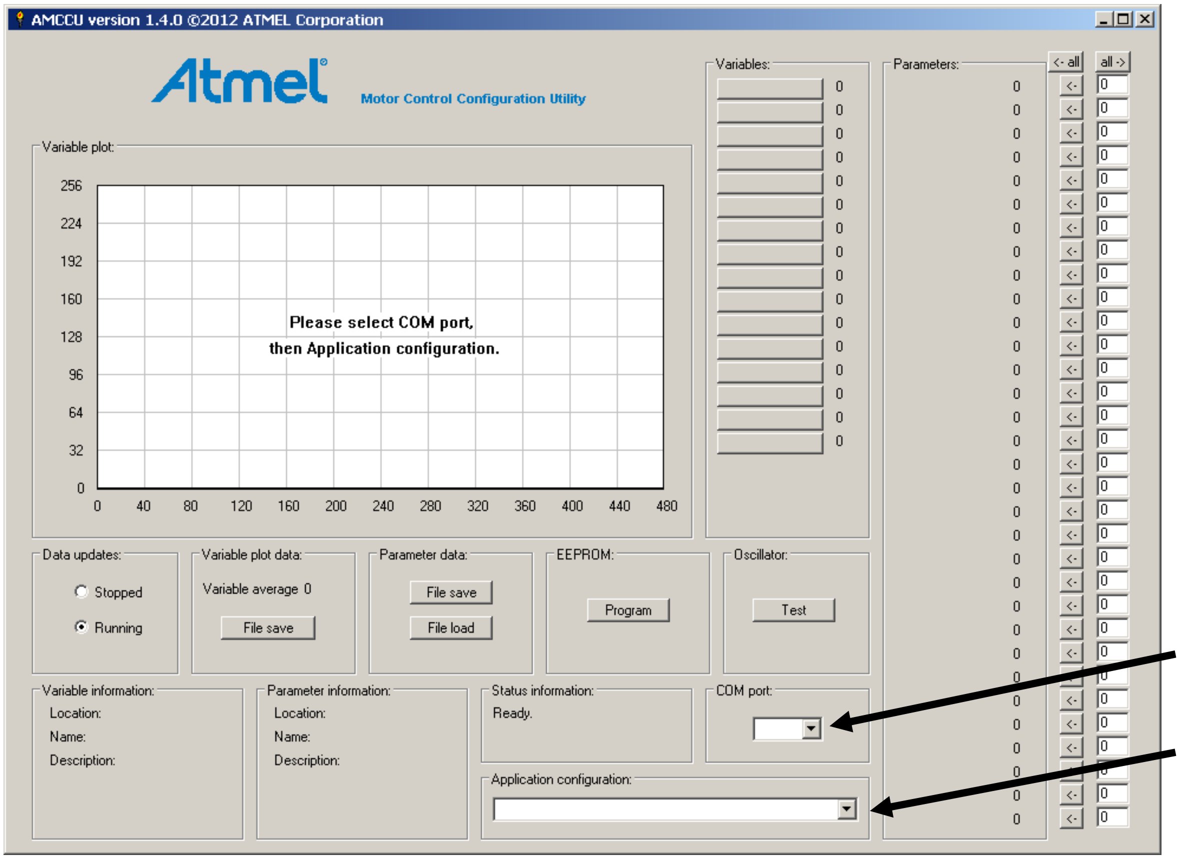
Once the hardware is set up and the firmware is present in the Atmel ATxmega16D4 flash memory you are ready to run the configuration utility.

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.



Please select COM port,
then Application configuration.

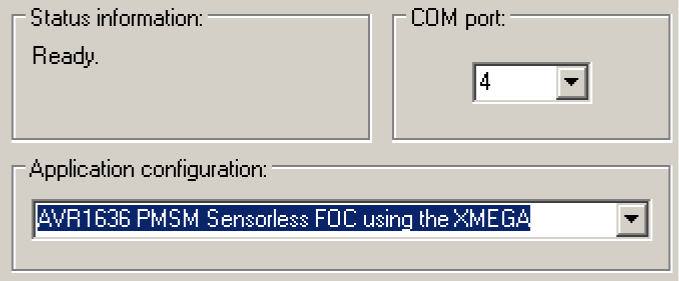
Ready.

Select COM port.

Select Application configuration.

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.

Figure 3-5. COM port and application configuration boxes after selection.



Status information:
Ready.

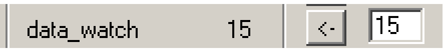
COM port: 4

Application configuration: AVR1636 PMSM Sensorless FOC using the XMEGA

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.

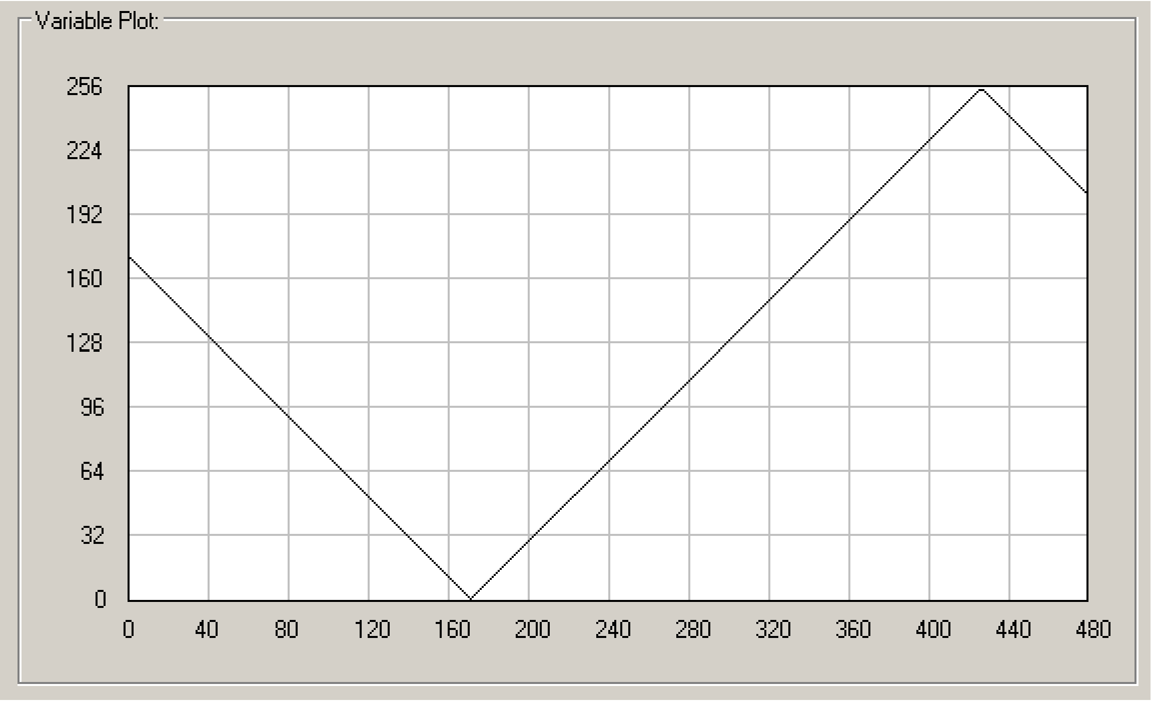
Figure 3-6. data_watch parameter.



data_watch 15

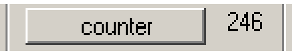
15

Figure 3-7. Variable plot screen displaying counter variable.



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.

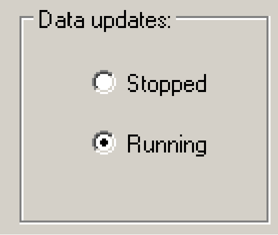
Figure 3-8. Counter button.



246

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.



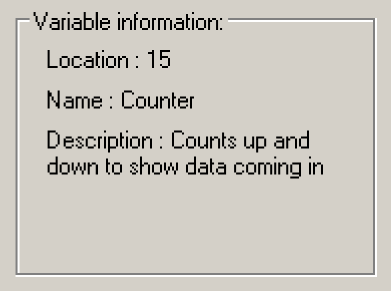
Data updates:

○ Stopped

● Running

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:

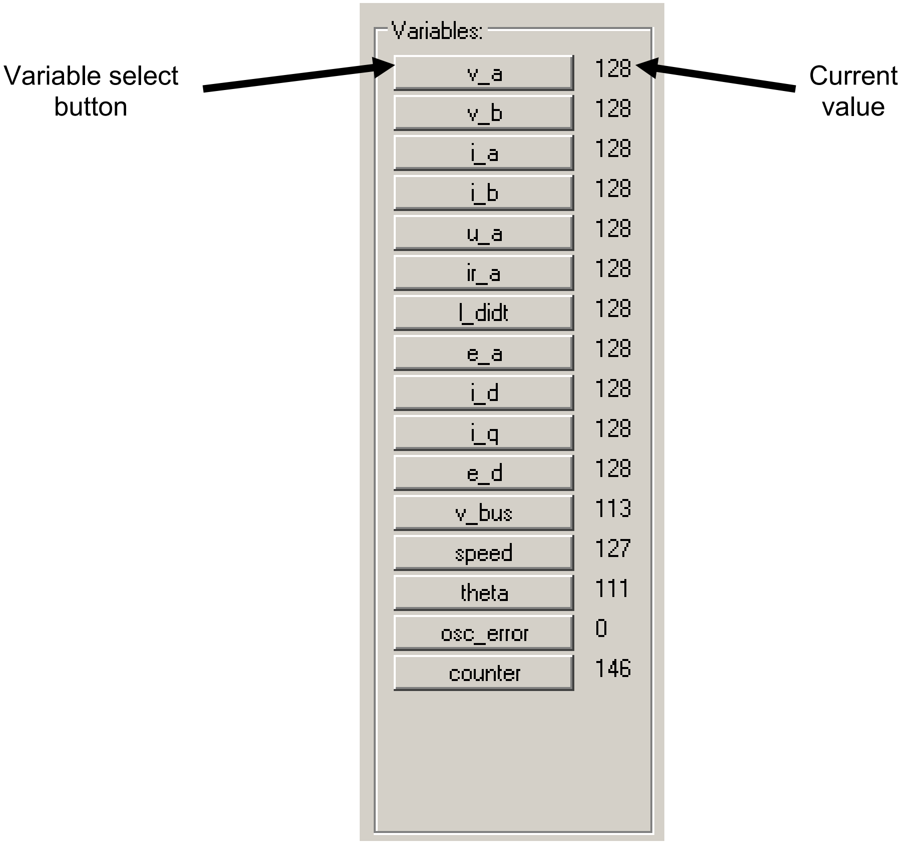
Location : 15

Name : Counter

Description : Counts up and down to show data coming in

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.



Variables:

v_a	128
v_b	128
i_a	128
i_b	128
u_a	128
ir_a	128
l_didt	128
e_a	128
i_d	128
i_q	128
e_d	128
v_bus	113
speed	127
theta	111
osc_error	0
counter	146

The variables are described in Table 3-1.

Table 3-1. Variable list description and use.

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

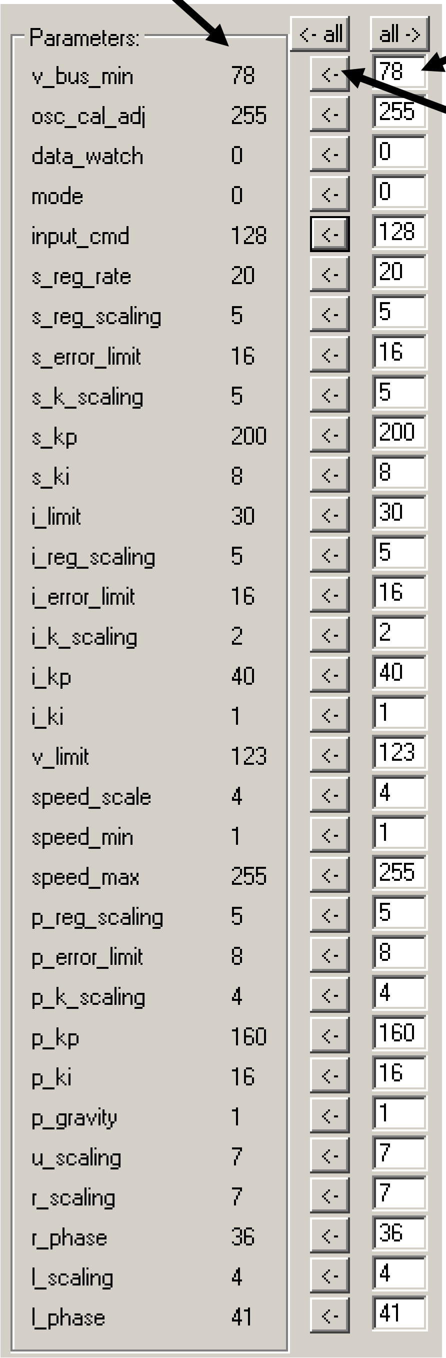
Variable	Description	Use
i_q	q-axis current perpendicular with air gap flux.	Used to verify current regulator is controlling torque producing component of current.
e_d	d-axis component of Back EMF.	Value driven to zero (=128) by back EMF PLL.
v_bus	DC input voltage.	Raw ADC sample of DC bus voltage.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

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.



Parameters:		<- all	all ->
v_bus_min	78	<-	78
osc_cal_adj	255	<-	255
data_watch	0	<-	0
mode	0	<-	0
input_cmd	128	<-	128
s_reg_rate	20	<-	20
s_reg_scaling	5	<-	5
s_error_limit	16	<-	16
s_k_scaling	5	<-	5
s_kp	200	<-	200
s_ki	8	<-	8
i_limit	30	<-	30
i_reg_scaling	5	<-	5
i_error_limit	16	<-	16
i_k_scaling	2	<-	2
i_kp	40	<-	40
i_ki	1	<-	1
v_limit	123	<-	123
speed_scale	4	<-	4
speed_min	1	<-	1
speed_max	255	<-	255
p_reg_scaling	5	<-	5
p_error_limit	8	<-	8
p_k_scaling	4	<-	4
p_kp	160	<-	160
p_ki	16	<-	16
p_gravity	1	<-	1
u_scaling	7	<-	7
r_scaling	7	<-	7
r_phase	36	<-	36
l_scaling	4	<-	4
l_phase	41	<-	41

EEPROM parameter in RAM

User edited value

Button transfers value to RAM. Values are not transferred to EEPROM until program button is pressed.

The parameters are described in Table 3-2.

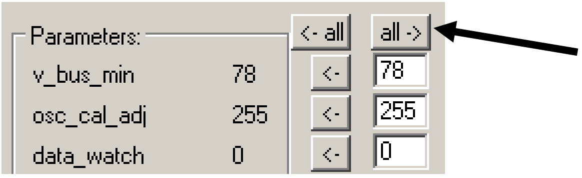
Table 3-2. Parameter list description and use.

Parameter	Description	Use
v_bus_min	Minimum bus voltage to enable FOC.	Shuts off and resets all regulators below the threshold level to prevent integrator windup.
osc_cal_adj	Oscillator calibration adjustment.	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.
data_watch	Selected variable in ***variable plot*** window.	Used to display current selected variable or manually update. Variables are numbered 0 to 15 from top to bottom.
mode	Operation mode selection.	Selects between voltage, current, and speed mode.
input_cmd	Input command.	Used to manually send input command, interpretation of command depends on ***mode***.
s_reg_rate	Speed regulator update rate in number of FOC loop updates.	Use to set speed regulator bandwidth below current regulator bandwidth by slowing down the update rate.
s_reg_scaling	Speed regulator scaling.	Sets the number of fractional bits used in speed regulator calculations.
s_error_limit	Speed error limit.	Clamps the speed regulator error.
s_k_scaling	Speed gain scaling.	Sets the number of fractional bits used in speed regulator gain calculations.
s_kp	Speed regulator proportional gain.	Set proportional gain of speed regulator.
s_ki	Speed regulator integral gain.	Set integral gain of speed regulator.
i_limit	Current limit.	Used to set the maximum current reference that is produced by the speed regulator output.
i_reg_scaling	Current regulator scaling.	Sets the number of fractional bits used in both Id and Iq regulator calculations.
i_error_limit	Current error limit.	Clamps the current regulator error.
i_k_scaling	Current gain scaling.	Sets the number of fractional bits used in current regulator gain calculations.
i_kp	Current regulator proportional gain.	Set proportional gain of current regulators.
i_ki	Current regulator integral gain.	Set integral gain of current regulators.
v_limit	Voltage limit.	Used to set the maximum voltage reference that is produced by the current regulator output.
speed_scale	Speed scaling.	Value sets the scaling speed and the motor RPM. Equations covered in the back EMF PLL section.
speed_min	Minimum speed.	Minimum speed of back EMF PLL. Set to reliably start up motor from rest.
speed_max	Maximum speed.	Maximum motor speed of back EMF PLL. Usually set 10% to 20% higher than no load speed of motor.
p_reg_scaling	Back EMF PLL regulator scaling.	Sets the number of fractional bits used in the back EMF PLL regulator calculations.
p_error_limit	Back EMF PLL error limit.	Clamps the back EMF PLL regulator error.
p_k_scaling	Back EMF PLL gain scaling.	Sets the number of fractional bits used in back EMF PLL regulator gain calculations.
p_kp	Back EMF PLL regulator proportional gain.	Set proportional gain of back EMF PLL regulator.
p_ki	Back EMF PLL regulator integral gain.	Set integral gain of back EMF PLL regulator.

Parameter	Description	Use
p_gravity	Back EMF PLL regulator gravity term.	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.
u_scaling	Phase voltage scaling parameter.	Used to set scaling of phase voltage in back EMF calculator. See section on back EMF Calculator.
r_scaling	Phase resistance scaling parameter.	Used to set scaling of phase resistance in back EMF calculator.
r_phase	Phase resistance.	Phase resistance, ohms per count determined by u_scaling and r_scaling.
l_scaling	Phase inductance scaling parameter.	Used to set scaling of phase inductance in back EMF calculator.
l_phase	Phase inductance.	Phase inductance, henries per count determined by u_scaling and l_scaling.

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.



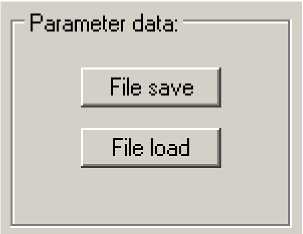
Parameters:

		<- all	all ->
v_bus_min	78	<-	78
osc_cal_adj	255	<-	255
data_watch	0	<-	0

Copy all button

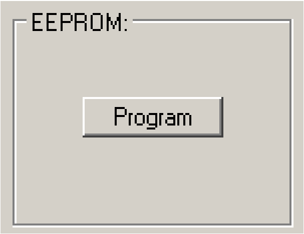
The white boxes on the far right column of Figure 3-12 are for editing and transferring data between the configuration utility and files. The file transfer is a simple file saves or file load (Figure 3-14) to a text file EEPROM.txt that is stored in the same directory that the configuration utility is 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.

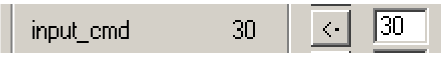
Figure 3-15. Program parameters in EEPROM.



The parameter ***input_cmd*** is used for directly sending a voltage, current, or speed command. The ***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.

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.

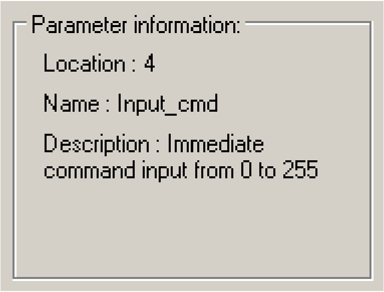
Figure 3-16. input_cmd direct input.



input_cmd 30 <- 30

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

Figure 3-17. Parameter information.



Location : 4

Name : Input_cmd

Description : Immediate command input from 0 to 255

3.3.4 Test run of motor

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

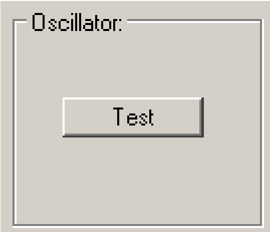
1. Firmware is installed and running.
2. EEPROM parameters are the same as Figure 3-12.
3. The USB-to-UART cable is connected to the processor board and PC and communication is established with the configuration utility.
4. The motor leads are connected of J2.
5. The universal power supply (or a lab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

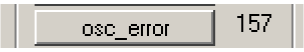
The oscillator calibration can be checked by pressing the test button as shown is in Figure 3-18. The internal oscillator is compared with the USB-to-UART cable oscillator which has an oscillator accuracy of better than 0.2%. See the Atmel ATxmega16D4 datasheet for further details about oscillator calibration.

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.



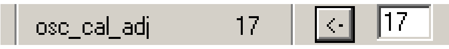
osc_error 157

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:

$$adjusted\ osc\_cal = osc\_cal + osc\_cal\_adj - 32 \qquad \text{Equation 3-1}$$

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

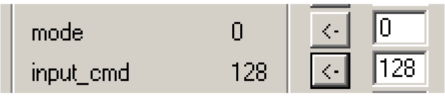


osc_cal_adj 17 <- 17

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.



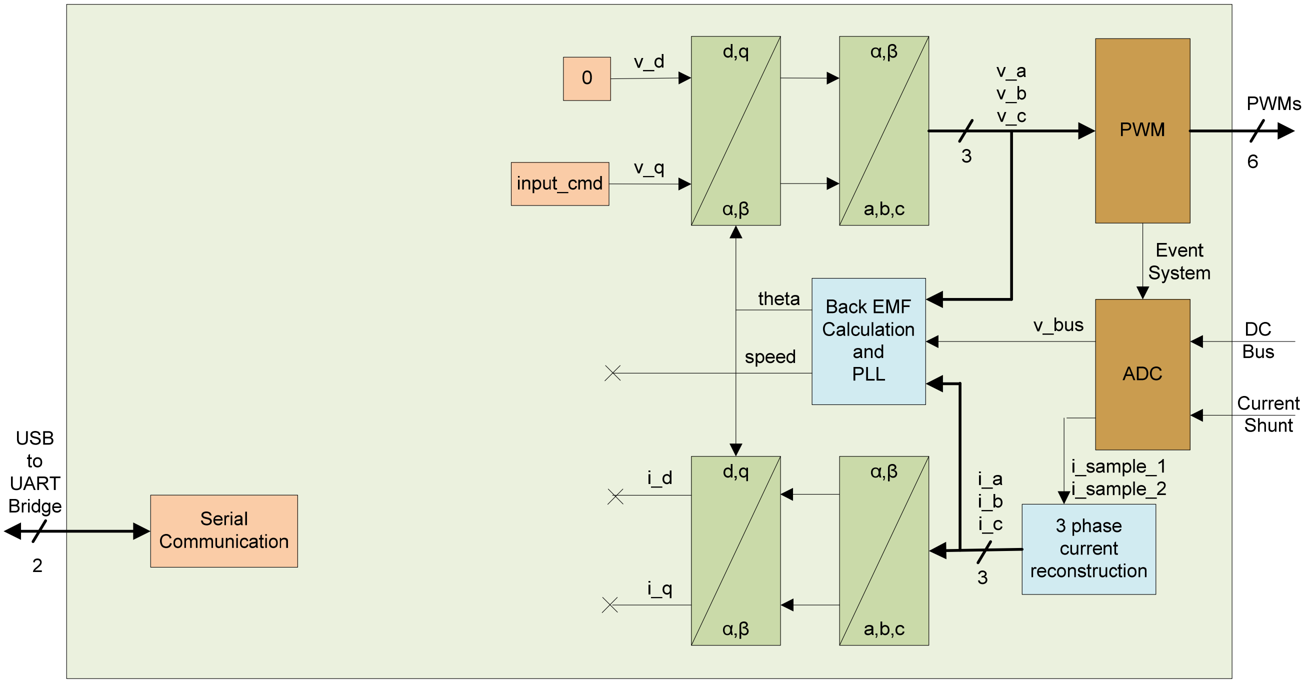
mode 0 <- 0
input_cmd 128 <- 128

Table 3-3. Mode options.

Mode	Control mode	Variable affected
0	Voltage mode (open loop PWM)	***v_d*** = 0, ***v_q*** = ***input_cmd***
1	Current control mode	***i_d*** = 0, ***i_q*** = ***input_cmd***
2	Speed control mode	***speed_cmd*** = ***input_cmd***

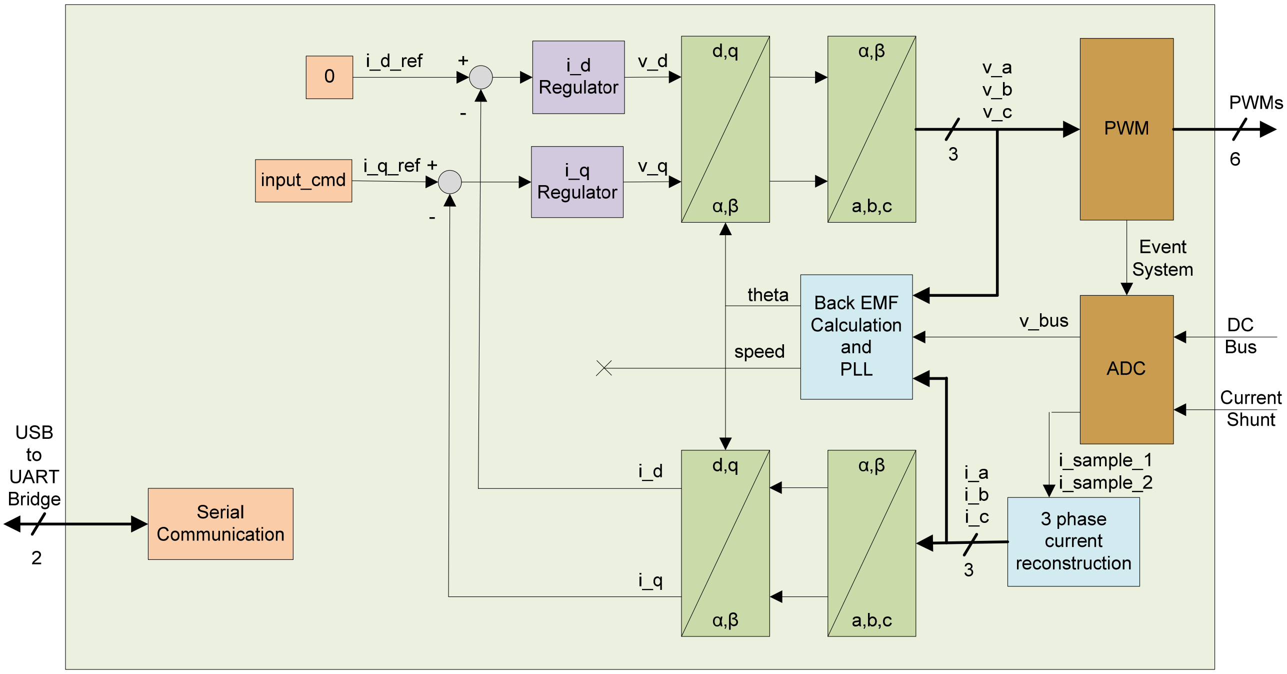
For ***mode*** = 0, voltage mode, the control blocks used are shown in Figure 3-22. Voltage mode directly maps into the input_cmd into the ***v_q*** variable. This mode is useful for initial tuning of the back EMF calculator and the back EMF PLL, by eliminating any interaction with sensing and the speed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

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



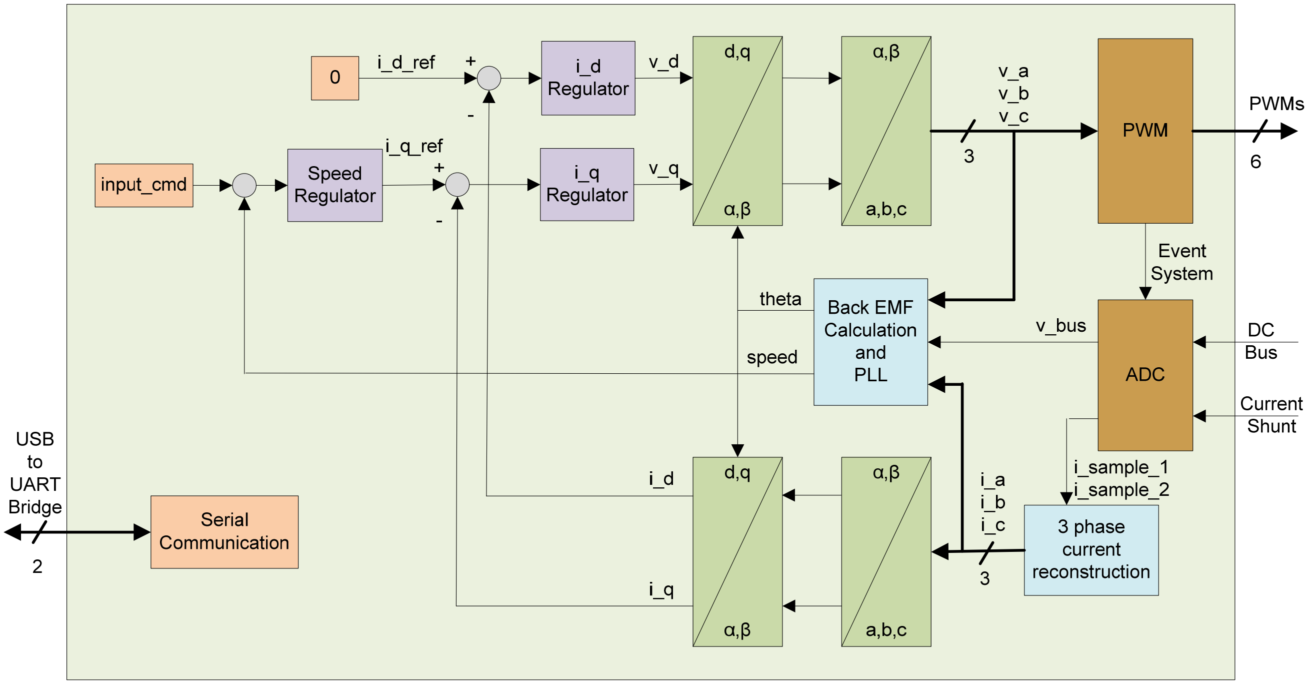
For ***mode*** = 1, current control mode, the control blocks used are shown in Figure 3-23. Current control mode set the current reference ***i_q_ref*** to the input command and sets ***i_d_ref*** to 0. The setting of ***i_q_ref*** produces a torque proportional to current, by regulation the torque producing 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.

Figure 3-23. Mode = 1, current control mode (torque control).



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 toque 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.

Figure 3-24. Mode = 2, speed control mode.



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:

$$v\_bus = V_{DC} \times K_{bus}[count] \qquad \text{Equation 3-2}$$

Where:

V_{DC} = *DC input voltage [volt]*

K_{bus} = *Scaling constant for variable **v_bus** [count/volt]*

$$K_{bus} = \frac{Divider_{bus} \times Resolution_{ADC}}{V_{ref}} \left[\frac{count}{volt}\right] \qquad \text{Equation 3-3}$$

Where:

$Divider_{bus}$ = *Resistive divider on low voltage motor control board, Appendix B, R9 and R27.*

$Resolution_{ADC}$ = *ADC resolution used for calculation [count]*

V_{ref} = *Voltage reference used for ADC [volt]*

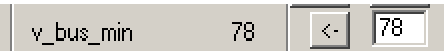
Using the values on the low voltage motor control board in Appendix B:

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

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

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).



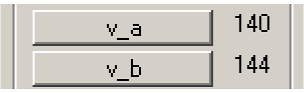
v_bus_min 78 <- 78

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***.

$$v\_a[count] \qquad \text{Equation 3-6}$$

Figure 3-26. Variables v_a and v_b.



v_a 140

v_b 144

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.

$$i\_a = I_a \times K_{current}[count] \qquad \text{Equation 3-7}$$

Where:

I_a = *Phase a current [ampere]*

$K_{current}$ = *Scaling constant for variable i_a, i_b and i_c [counts/ampere]*

$$K_{current} = \frac{R_{shunt} \times Divider_{current} \times Resolution_{ADC}}{V_{ref}} \left[\frac{count}{ampere}\right] \qquad \text{Equation 3-8}$$

Where:

R_{shunt} = *Current sensing resistor on low voltage motor control board, Appendix B, R1 [Ω]*

$Divider_{current}$ = *Resistive divider used for biasing ground referenced R_{shunt} voltage above ground on low voltage motor control board, Appendix B, R29 and R14*

$Resolution_{ADC}$ = *ADC resolution used for calculation [count]*

For the kit values:

$$K_{current} = \frac{0.05\Omega \times \left(\frac{10k\Omega}{1k\Omega + 10k\Omega}\right) \times 256}{1V} \left[\frac{count}{ampere}\right] \qquad \text{Equation 3-9}$$

$$K_{current} = 11.6 \left[\frac{count}{ampere}\right] \qquad \text{Equation 3-10}$$

Figure 3-27. Variables i_a and i_b.

i_a 128

i_b 128

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:

$$e\_a = u\_a - ir\_a - l\_didt\_a[count] \qquad \text{Equation 3-11}$$

The physical equation for back EMF in SI units is.

$$E_a = U_a - I_a \times R_{phase} - L_{phase} \times \frac{dI_a}{dt}[volt] \qquad \text{Equation 3-12}$$

Where:

E_a = *Phase a back emf [volt]*

U_a = *Phase a voltage [volt]*

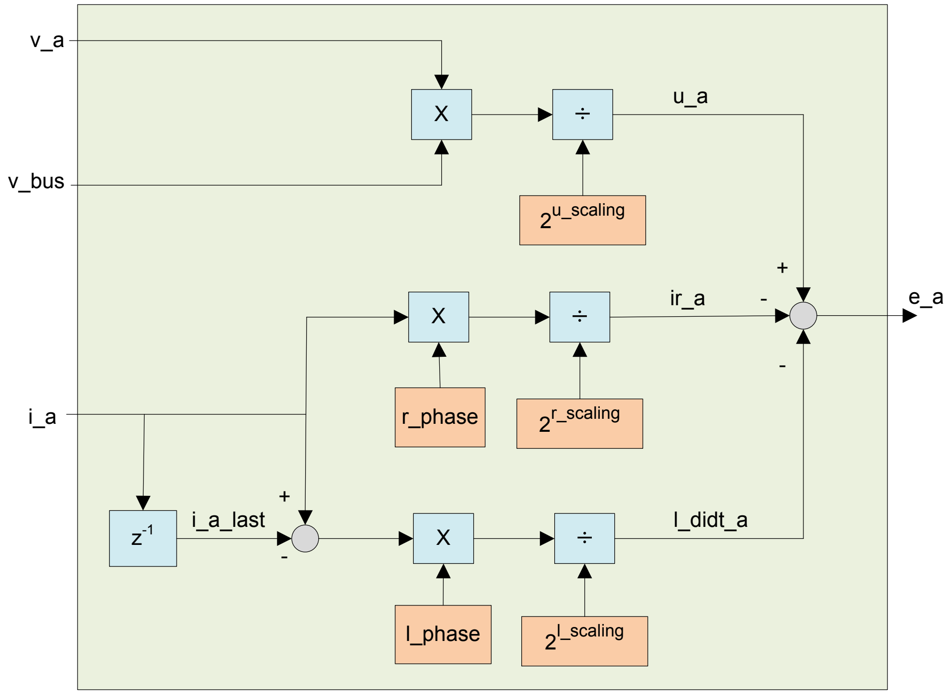
I_a = *Phase a current [ampere]*

R_{phase} = *Phase resistance [ohm]*

L_{phase} = *Phase inductance [Henry]*

The back EMF calculator for one of three phases with parameters is shown in Figure 3-28.

Figure 3-28. Back EMF calculator one of three phases with parameters.



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:

$$u\_a \;=\; \frac{(v\_a\ x\ v\_bus)}{2^{u\_scaling}}[count] \qquad \text{Equation 3-13}$$

$$u\_a \;=\; U_a \times K_u[count] \qquad \text{Equation 3-14}$$

Where:

K_u = *Scaling constant for variable **u_a** [count/volt]*

U_a = *Phase a voltage [volt]*

$$K_u = \frac{K_{bus} \times PWM_{max}}{2^{u\_scaling}}\left[\frac{count}{volt}\right] \qquad \text{Equation 3-15}$$

Where:

K_{bus} = *Scaling constant for variable **v_bus** [count/volts]*

PWM_{max} = *Maximum count for PWM timer [count]*

$2^{u\_scaling}$ = *Scaling factor for voltage **u_a**, **u_b**, and **u_c** [count]*

Using the example values:

$$K_u = \frac{7.48 \times 256}{2^7}\left[\frac{count}{volt}\right] \qquad \text{Equation 3-16}$$

$$K_u = 14.96\left[\frac{count}{volt}\right] \qquad \text{Equation 3-17}$$

The scaled resistive drop, ***ir_a***, is determined by the following equation:

$$ir\_a \; = \; \frac{(\, i\_a \times r\_phase)}{2^{r\_scaling}} [count] \qquad \text{Equation 3-18}$$

$$ir\_a \; = \; I_a \times R_{phase} \times K_u [count] \qquad \text{Equation 3-19}$$

$$r\_phase = R_{phase}[\mathrm{ohm}] \times K_r \left[\frac{count}{\mathrm{ohm}}\right] \qquad \text{Equation 3-20}$$

Where:

K_r = *Scaling constant for variable **r_phase** [count/ohm]*

$$K_r = \frac{K_u \times 2^{r\_scaling}}{K_{current}} \left[\frac{count}{\mathrm{ohm}}\right] \qquad \text{Equation 3-21}$$

Where:

$2^{r\_scaling}$ = *Scaling factor for **r_phase** [count]*

Using the example values:

$$K_r = \frac{14.96 \times 2^7}{11.6} \left[\frac{count}{\mathrm{ohm}}\right] \qquad \text{Equation 3-22}$$

$$K_r = 165 \left[\frac{count}{\mathrm{ohm}}\right] \qquad \text{Equation 3-23}$$

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

$$R_{phase} = \frac{0.44}{2} [\mathrm{ohm}] \qquad \text{Equation 3-24}$$

$$r\_phase = 0.22[\mathrm{ohm}] \times 165 \left[\frac{count}{\mathrm{ohm}}\right] \qquad \text{Equation 3-25}$$

$$r\_phase = 36[count] \qquad \text{Equation 3-26}$$

The scaled inductive drop, ***l_didt_a***, is determined by the following equation:

$$l\_didt\_a \; = \; \frac{l\_phase \times (\, i\_a - i\_a\_last)}{2^{l\_scaling}} [count] \qquad \text{Equation 3-27}$$

$$l\_didt\_a \; = \; L_{phase} \times \frac{dI_a}{dt} \times K_u [count] \qquad \text{Equation 3-28}$$

$$l\_phase = L_{phase}[\mathrm{Henry}] \times K_l \left[\frac{count}{\mathrm{Henry}}\right] \qquad \text{Equation 3-29}$$

Where:

K_l = *Scaling constant for variable l_phase [count/Henry]*

$$K_l = \frac{K_u \times 2^{l\_scaling} \times freq_{loop}}{K_{current}} \left[\frac{count}{\text{Henry}}\right] \qquad \text{Equation 3-30}$$

Where:

$2^{r\_scaling}$ = *Scaling factor for r_phase [count]*

$freq_{loop}$ = *Frequency of FOC control loop [hertz]*

Using the example values:

$$K_l = \frac{14.96 \times 2^4 \times 7812.5Hz}{11.6} \left[\frac{count}{\text{Henry}}\right] \qquad \text{Equation 3-31}$$

$$K_l = 161200 \left[\frac{count}{\text{Henry}}\right] \qquad \text{Equation 3-32}$$

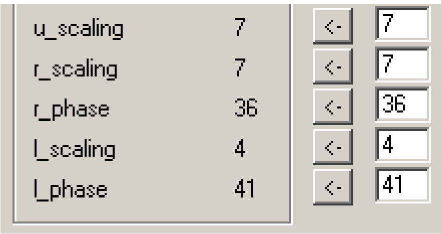
The motor datasheet value from Appendix C for line to line inductance is divided by 2:

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

$$l\_phase = 0.000255[\text{Henrys}] \times 161200 \left[\frac{count}{\text{Henry}}\right] \qquad \text{Equation 3-34}$$

$$l\_phase = 41[count] \qquad \text{Equation 3-35}$$

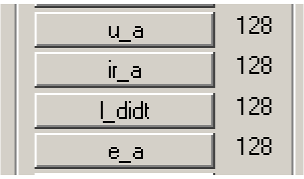
Figure 3-29. Back EMF calculation parameters.



u_scaling	7	<-	7
r_scaling	7	<-	7
r_phase	36	<-	36
l_scaling	4	<-	4
l_phase	41	<-	41

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.

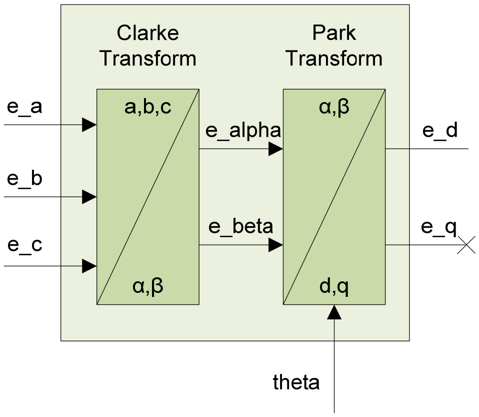
Figure 3-30. Back EMF calculation variables for phase a.



u_a	128
ir_a	128
l_didt	128
e_a	128

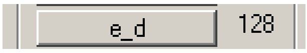
The three phase back EMF goes through a Clarke and Park transform, Figure 3-31, for feedback to the back EMF phase locked loop.

Figure 3-31. Back EMF Clarke and Park transforms.



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.

Figure 3-32. Back EMF d-axis value.

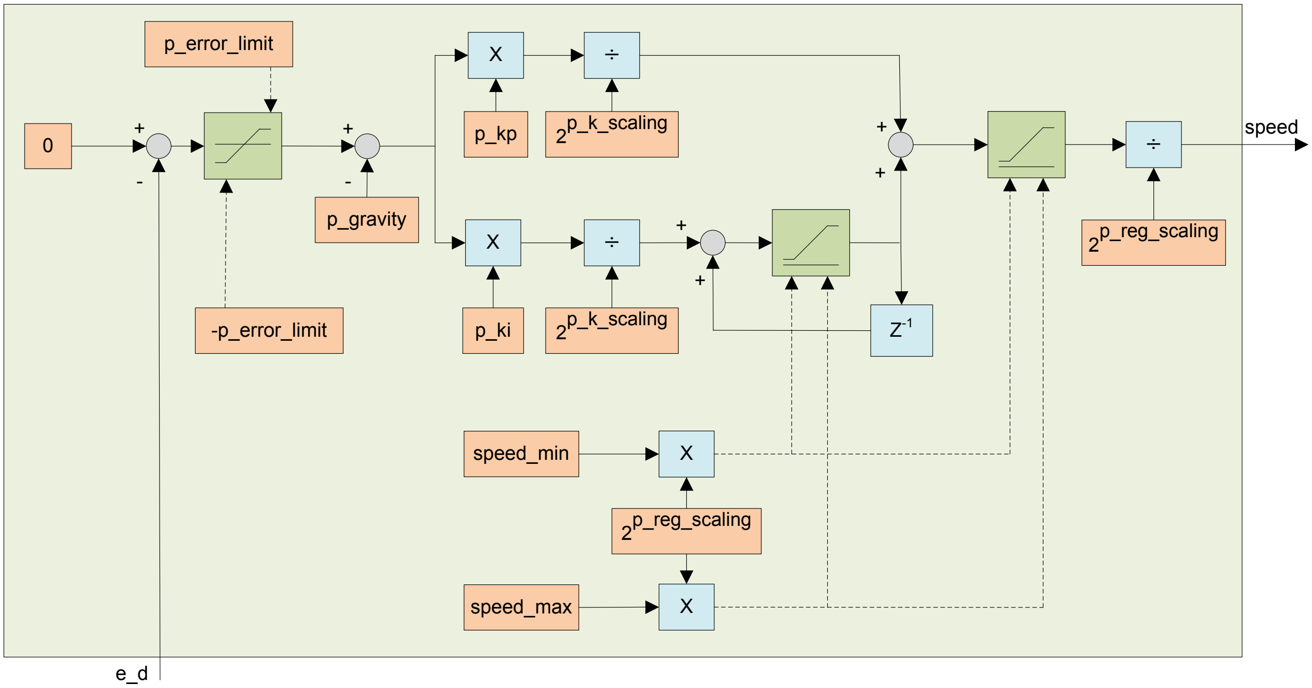


128

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.

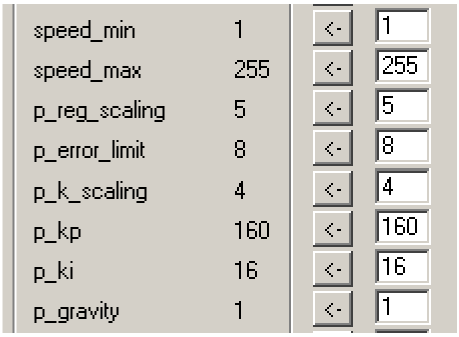
Figure 3-33. Back EMF PLL block diagram with parameters.



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.



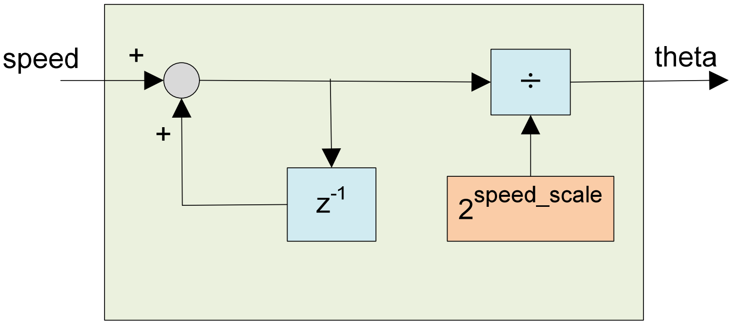
speed_min	1	<-	1
speed_max	255	<-	255
p_reg_scaling	5	<-	5
p_error_limit	8	<-	8
p_k_scaling	4	<-	4
p_kp	160	<-	160
p_ki	16	<-	16
p_gravity	1	<-	1

Tuning a PLL regulator typically is done in ***mode*** = 0, Voltage mode, with an ***input_cmd*** fairly low so that excessive current does not flow in the motor at stall. The ***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 ***p_kp*** and ***p_ki*** values that give the motor quick startup and smooth steady state performance. This is done by watching the ***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 ***i_ki*** to a low value such as 1 or 2. Then increase the ***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 ***speed_scale*** Figure 3-36. This scaling sets the frequency range of the sine wave output voltages ***v_a***, ***v_b***, and ***v_c*** and as a result sets the speed range of the motor.

Figure 3-36. Speed scaling parameter.

speed_scale 4

4

The relationship between ***speed_scale*** and frequency is:

$$rpm = speed \times K_{rpm}[R.P.M.] \qquad \text{Equation 3-36}$$

$$K_{rpm} = \left(\frac{freq_{loop}\left[\frac{hertz}{count}\right]}{theta_{max} \times 2^{speed\_scale}}\right)\left(60\left[\frac{sec}{min}\right]\right)\left(\frac{2}{N_{poles}}\right)\left[\frac{R.P.M.}{count}\right] \qquad \text{Equation 3-37}$$

Where:

Speed = *speed_min or speed_max [unitless] (between 1 and 255)*

$freq_{loop}$ = *Frequency of FOC control loop [hertz]*

$theta_{max}$ = *Maximum count for theta equal to 360 degrees [count]*

2^{speed_scale} = *Scaling factor for speed [counts]*

N_{poles} = *Number of motor poles from motor datasheet, Appendix C*

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

$$K_{rpm} = 28.6\left[\frac{R.P.M.}{count}\right]$$ Equation 3-39

The minimum and maximum speed can be calculated replacing in ***speed_min*** and ***speed_max***:

$$rpm_{min} = speed\_min \times K_{rpm}[R.P.M.]$$ Equation 3-40

$$rpm_{min} = 28.6[R.P.M.]$$ Equation 3-41

$$rpm_{max} = speed\_max \times K_{rpm}[R.P.M.]$$ Equation 3-42

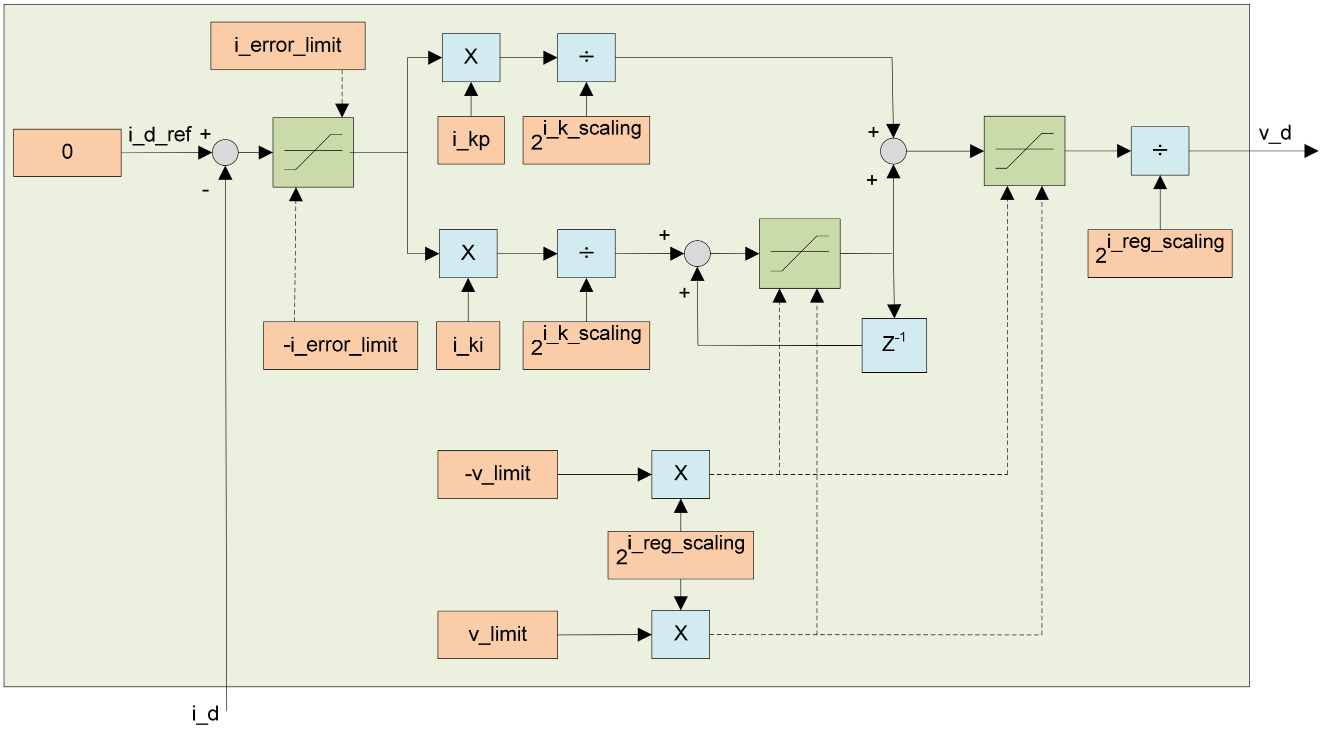
$$rpm_{max} = 7293[R.P.M.]$$ Equation 3-43

It is important to choose a ***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

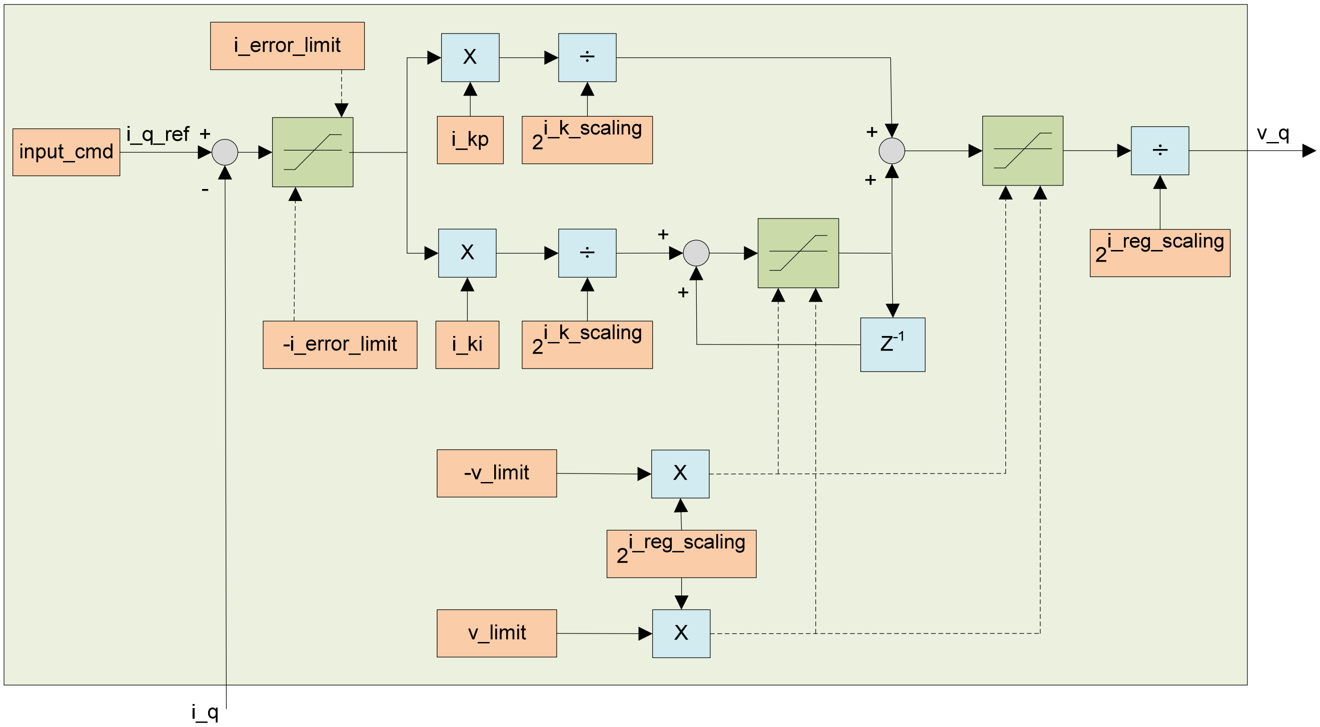
There are two current regulators, one for the d-axis current and one for the q-axis current. The d-axis is aligned with the air gap flux produced by the rotor magnets. Current along this axis can add to or subtract from the air gap flux, this is typically used for field weakening to extend the speed range of the motor. Field weakening is not implemented so the command is set to zero. Figure 3-37 shows the current regulator for the d-axis current, ***i_d*** and the reference set to zero.

Figure 3-37. i_d current regulator with parameters.



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.

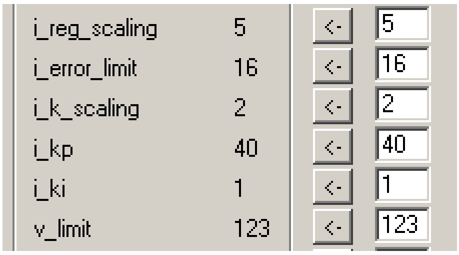
Figure 3-38. i_q current regulator with parameters.



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. ***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. ***i_error_limit*** – clamps the amount of error that goes into the PI controller. Typical range is 8 to 32.
3. ***i_k_scaling*** – sets the number of fractional bits used in calculation of the PI gains.
4. ***i_kp*** – the proportional gain.
5. ***i_ki*** – the integral gain.
6. ***v_limit*** – set the maximum value for ***v_d*** and ***v_q***, the outputs of the regulators.

Figure 3-39. Current regulator parameters.



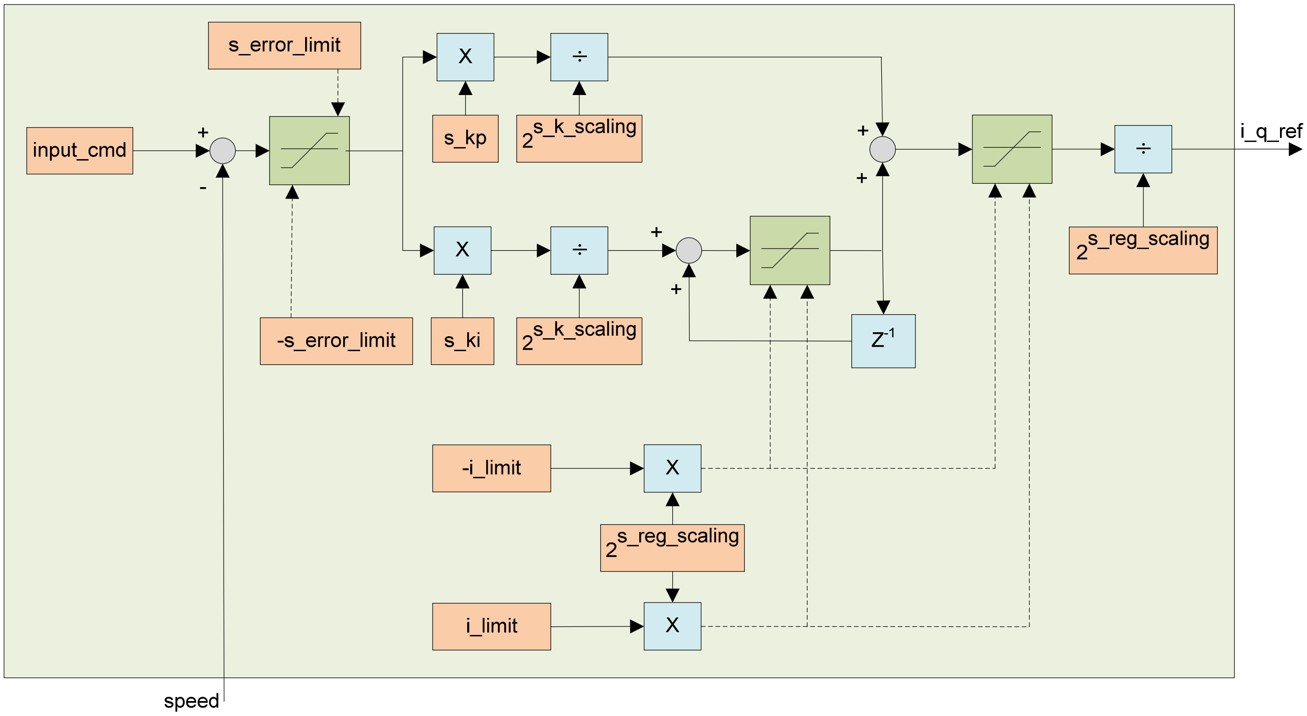
i_reg_scaling	5	<-	5
i_error_limit	16	<-	16
i_k_scaling	2	<-	2
i_kp	40	<-	40
i_ki	1	<-	1
v_limit	123	<-	123

Tuning a current 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 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 ***mode*** =2 the ***input_cmd*** sets the speed reference as shown in Figure 3-40.

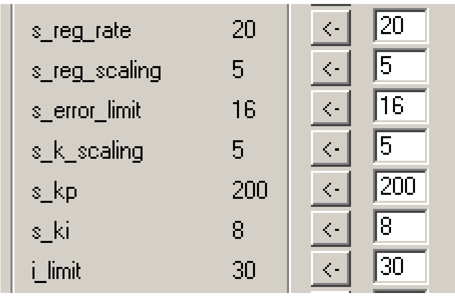
Figure 3-40. Speed regulator with parameters.



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

1. ***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. ***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. ***s_error_limit*** – clamps the amount of error that goes into the PI controller. Typical range is 8 to 32.
4. ***s_k_scaling*** – sets the number of fractional bits used in calculation of the PI gains.
5. ***s_kp*** – the proportional gain.
6. ***s_ki*** – the integral gain.
7. ***i_limit*** – set the maximum value ***i_q_ref***, the outputs of the regulator.

Figure 3-41. Speed control parameters.



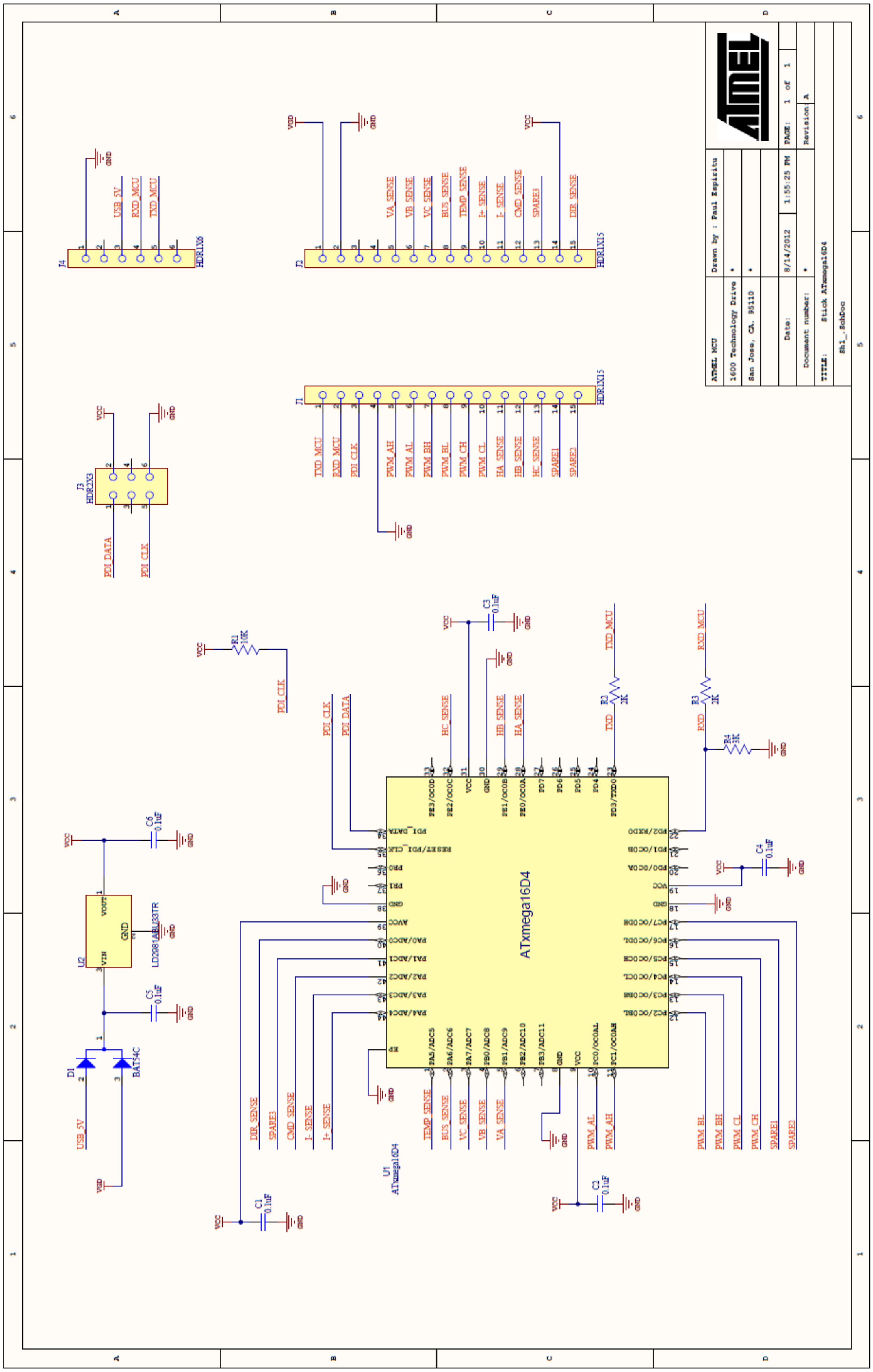
s_reg_rate	20	<-	20
s_reg_scaling	5	<-	5
s_error_limit	16	<-	16
s_k_scaling	5	<-	5
s_kp	200	<-	200
s_ki	8	<-	8
i_limit	30	<-	30

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.

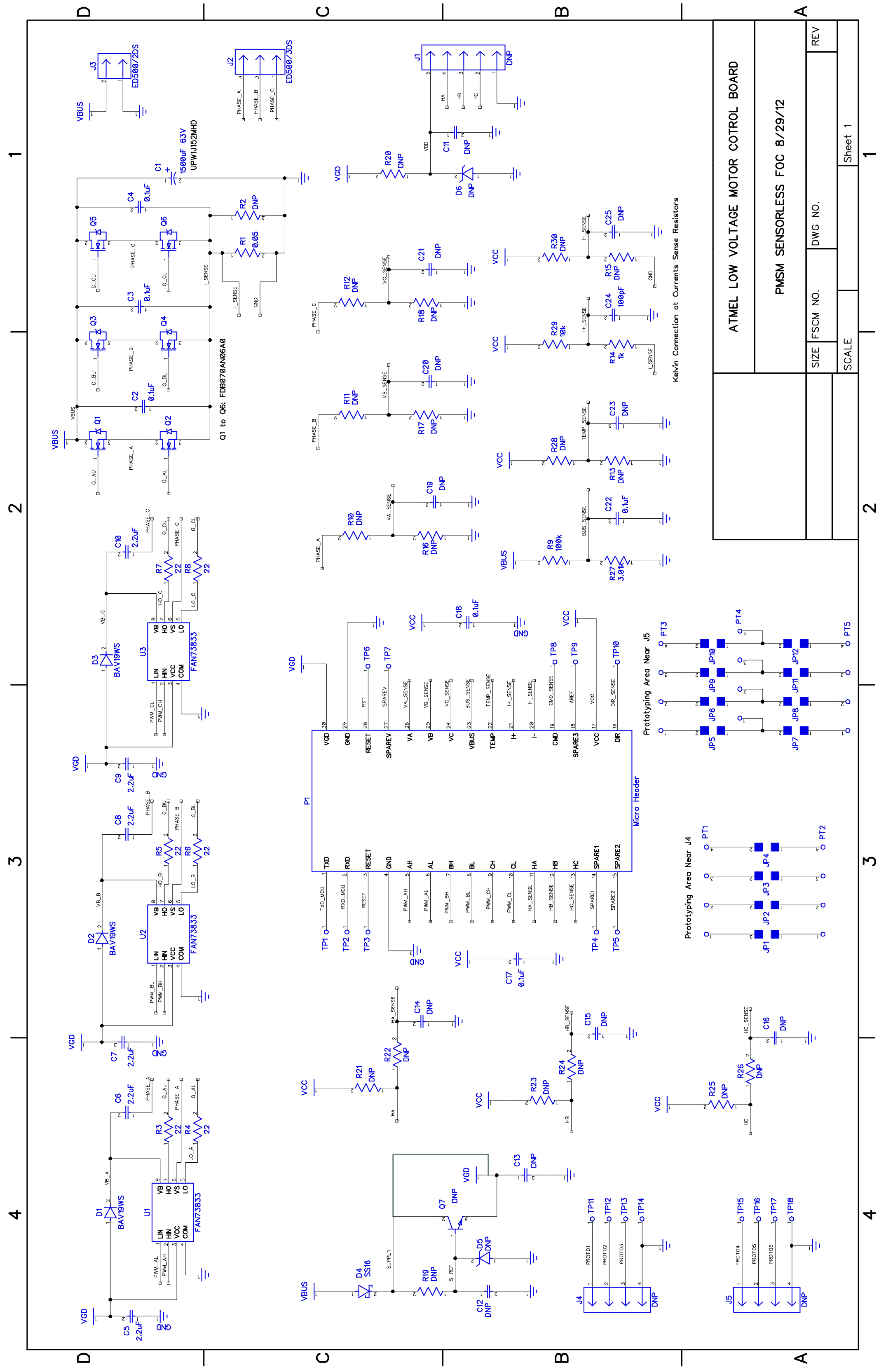
4. Conclusion

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.

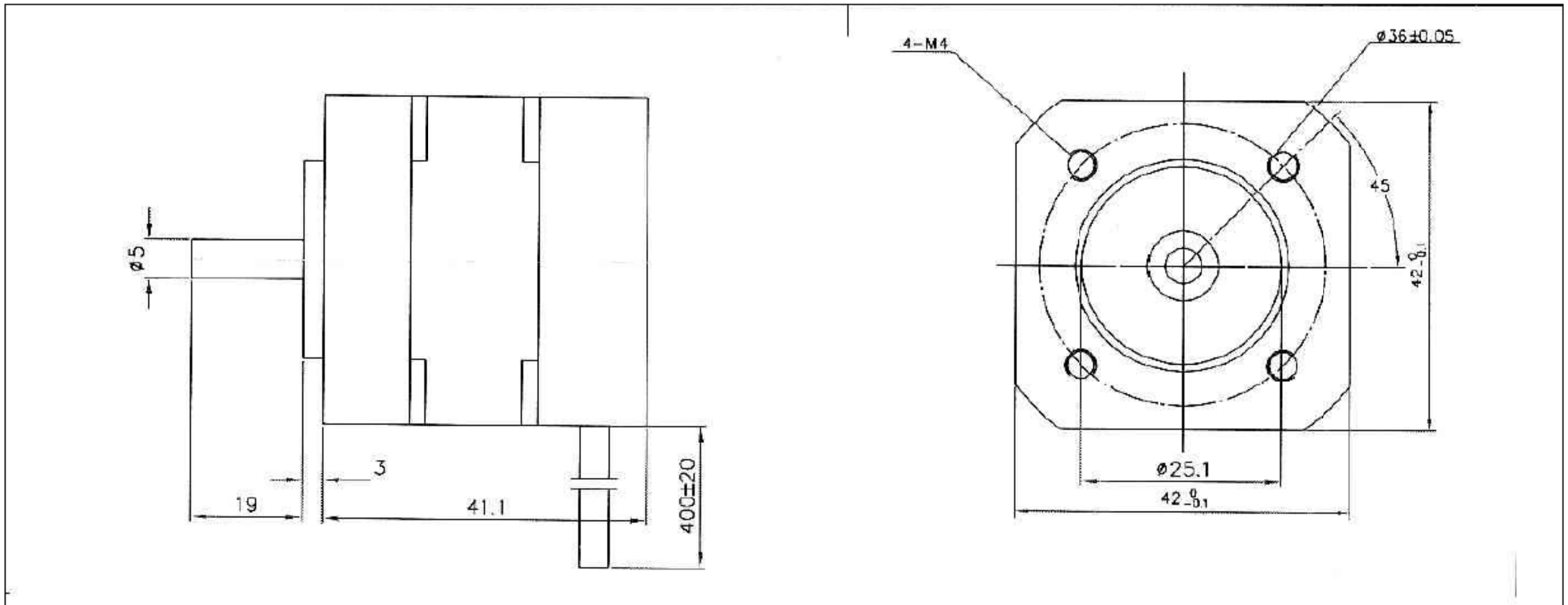
Appendix A. Atmel ATmega16D4 Processor Board Schematics



Appendix B. Low Voltage Motor Control Board Schematics



Appendix C. 42BLS01-001 Motor Datasheet



Specifications		
Item	Specifications	Conditions
N° phase	3	
No of poles	8	
Rated voltage	12 VDC	
No load speed	6200 RPM	
No load current	0,5A	
Rated speed	4000 RPM +/-5%	
Rated torque	0,0625 N.m	
Torque constant	0,0168 N.m/A	
Resistance line to line	0,44 ohms +/-10%	at 20°C
Inductance line to line	0,51 mH +/-20%	
Peak torque	0.19 N.m	
Rotor inertia	24 g.cm²	
Mass	0,3 Kg	
Dielelectric strength	500 VDC	1 minute
Insulation class	Class B	
Insulation resistance	100 Mohm min,	
Winding type	Delta	
Hall effect angle	3 x 120°	
Radial force max	28N	20 mm flange
Axial force max	10N	

Wiring diagram		
Lead color	Function	Lead gauge
YEL	Phase U	UL1007 AWG20
RED	Phase V	
BLK	Phase W	
RED	Hall +5V	UL1007 AWG26
BLU	Hall sensor 1	
GRN	Hall sensor 2	
WHT	Hall sensor 3	
BLK	Hall GND	

				Tolérances générales: +/-0.1	42BLS01-001
				Matière:	
				Echelle:	
				TECMOTION 147, rue Marcel MERIEUX 69007 LYON Tél: 04 78 25 28 09 Fax: 04 78 72 28 77	
stg		création			
Ope	Indice	Modifications	Date		

2

Appendix D. Kit Parts List

Description	Manufacturer	Part number
Low Voltage Motor Control Board	Atmel Corporation	
ATxmega16D4 Processor Board	Atmel Corporation	
12V 4000RPM	Tecmotion	42BLS01-001
USB-to-UART Bridge Cable	FTDI, Future Technology Devices International Ltd	TTL-232R-5V

Appendix E. Revision History

Doc. Rev.	Date	Comments
42061A	01/2013	Initial document release

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Building
1-6-4 Osaki
Shinagawa-ku, Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.