APPLICATION NOTE

Atmel AVR675: Configurable Three Phase BLDC Fan

8-bit AVR Microcontrollers

Features

- 3-phase BLDC fan application
- Sensorless back EMF control algorithm
- For Atmel® ATtiny461A or Atmel ATtiny861A
- Reference Design Hardware and Firmware
- PC based Atmel Motor Control Configuration Utility
- Tuning Guide

Introduction

This application note describes the implementation and use of a configurable 3-phase brushless DC (BLDC) fan using the ATtiny461A microcontroller implementing sensorless control. The complete hardware system includes a 12V three-phase BLDC fan, a Configurable BLDC Fan board, a USB to UART bridge cable, an AC power adapter, the firmware running on the ATtiny461A, and a PC based Motor Control Configuration Utility for customizing the fan for a particular application.

Table of Contents

1. Theory of Operation

1.1 Overview

This application note covers the description and use of a complete 3-phase BLDC fan control system.

After reading this document the reader should have an understanding of the system hardware, firmware, and the Motor Control Configuration Utility. Also the user should be able to configure and tune a BLDC fan using the Motor Control Configuration Utility.

1.2 Hardware

The Hardware consists of the complete fan control system. The system is composed of a 3-phase BLDC fan, Configurable BLDC Fan board, USB to UART bridge cable, AC power adapter provide all the hardware tools along with a user provided PC for configuring and tuning a fan for an end application.

1.3 Firmware

The firmware running on the Atmel ATtiny461A provides the control structure for sensorless BLDC motor control. The sensorless control technique uses back electromotive force (back EMF) sampling to form a back EMF sensing phase locked loop (PLL). Speed control is provided by a proportional integral (PI) controller. Both the back EMF sensing phase locked loop and the speed control loop require configuration and tuning covered in Chapter 0.

1.4 Motor Control Configuration Utility

The Atmel Motor Control Configuration Utility is software running on a PC provided as a tool for configuring and tuning the fan application through setting parameters that are stored in the ATtiny's EEPROM without the need to modify the firmware.

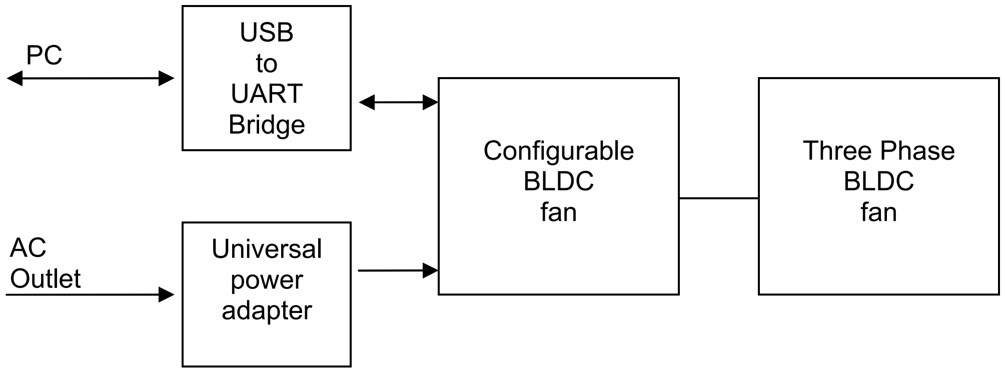
Configuring consists of setting parameters that are known or calculated quantities. These quantities translate into parameters that are set in the ATtiny's EEPROM. Tuning involves an iterative process of setting and testing parameters to find the optimal value. These values are not easily calculated 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 BLDC fan, the Configurable BLDC Fan Board, the USB to UART bridge cable, and the AC power adapter as shown if Figure 2-1.

Figure 2-1. Hardware Block Diagram.



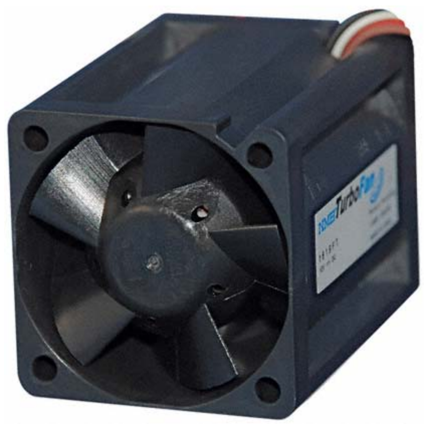
2.1.1 Three Phase BLDC Fan

The 3-phase BLDC fan consists of the following:

1. 3-phase, 8-pole, permanent magnet BLDC motor.
2. Plastic housing.
3. Fan prop.
4. Three motor leads brought out for connection to the Configurable BLDC Fan Board.

The fan shown in Figure 2-2 is rated for 12VDC input and 1.2A at 23,000rpm.

Figure 2-2. Three phase BLDC Fan.



2.1.2 Configurable BLDC Fan Board

The Configurable BLDC Fan Board controls power flow between an AC adapter and the 3-phase BLDC fan, as well as command translation from the outside world to the microcontroller.

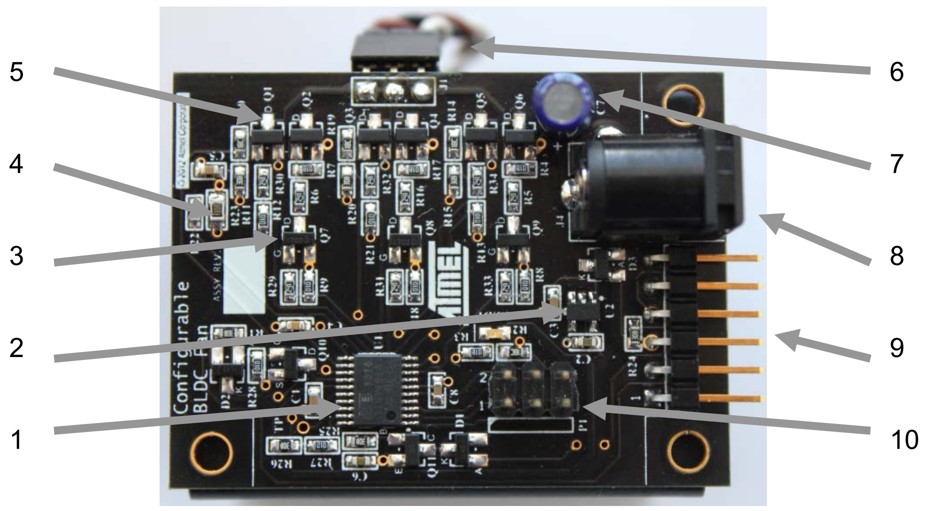
The Configurable BLDC Fan Board contains the following components:

1. Atmel ATtiny461A AVR® microcontroller.
2. 3.3V regulator to power the AVR.
3. Gate Drive circuitry.
4. Current Sense resistor.
5. 3-phase power stage consisting of three N-Channel MOSFETs as lower switches and three P-Channel MOSFETs as upper switches.
6. 3-pin header for motor phase connections.
7. DC Bus Bypass Capacitor.
8. 12V power adapter connector.
9. 6-pin header for USB to UART Bridge.
10. ISP/Debug connector.

The Configurable BLDC Fan Board is rated for 12V nominal input voltage and 1.2A output per phase. Rough component placement is shown in Figure 2-3.

The complete schematics are shown in Appendix A.

Figure 2-3. Configurable BLDC Fan Board.



2.1.3 USB to UART Bridge Cable

The USB to UART bridge cable provides communication between a PC and the Configurable BLDC Fan board. This cable is a standard product from FTDI. Part numbers are listed in Appendix B.

Figure 2-4. A USB to UART bridge cable.

2.1.4 Universal Power Adapter

The Universal adapter provides 12V for powering the DC bus for the Configurable Fan board. The adapter is for 2.5A at 12V. The AC Power Adapter uses a removable AC power connector. See Figure 2-5 for international use.

Figure 2-5. Universal Power Adapter with international plug options.

2.2 Firmware

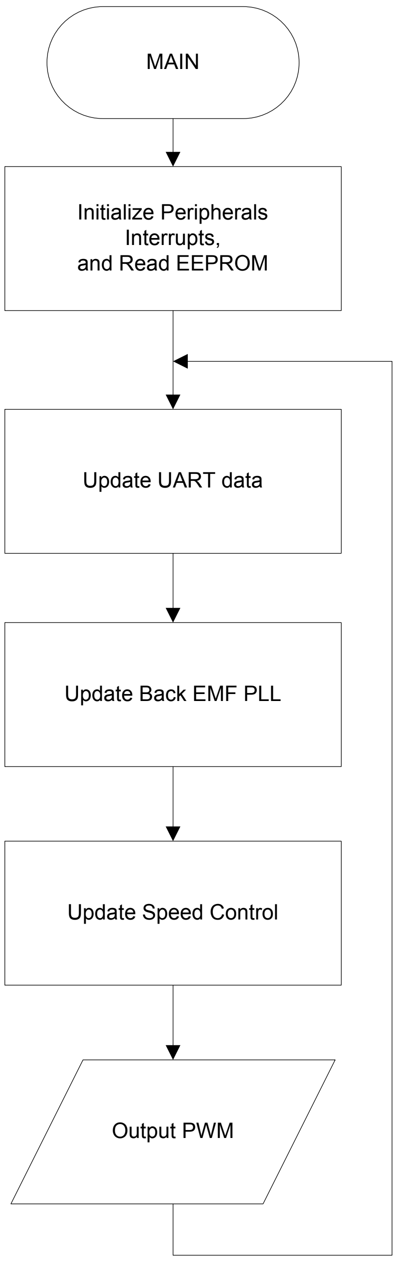
The firmware was designed to run on the Atmel ATtiny461A or Atmel ATtiny861A to demonstrate a configurable motor control application for driving 3-phase BLDC fans.

The sensorless control algorithm provides high speed operation above 50,000rpm for a 4-pole motor. The PLL is always enabled for robust operation that provides automatic recovery of the motor speed after a stall condition.

2.2.1 Control structure

The control structure is shown in Figure 2-6. 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 an infinite loop that handles UART data, updating of the back EMF PLL, the speed control, and updating the PWM output.

Figure 2-6. Firmware main loop.



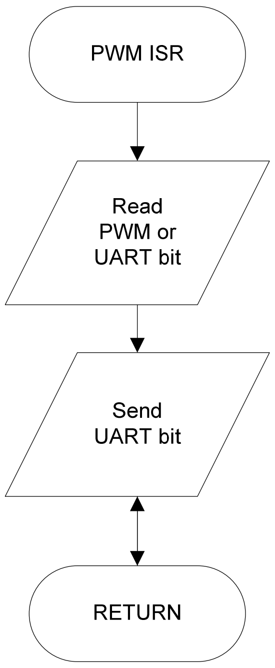
2.2.2 ISR routines

There are three Interrupt Service Routines (ISRs). Two are based off the PWM timer 1 and one is based off timer 0. The three ISRs are:

1. PWM ISR.
2. ADC ISR.
3. Commutation ISR.

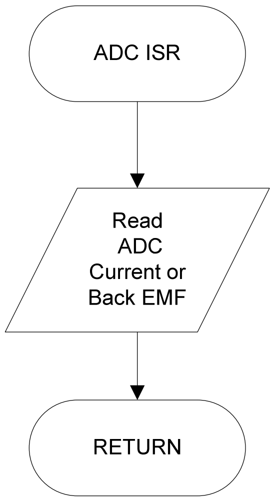
The PWM ISR serves three functions: timing for the main loop, reading the PWM speed command, and timing for UART communication. Because of the UART function, the PWM period is set to 19,200Hz to conform to standard PC UART speeds. Figure 2-7 shows the PWM ISR flow.

Figure 2-7. PWM ISR.



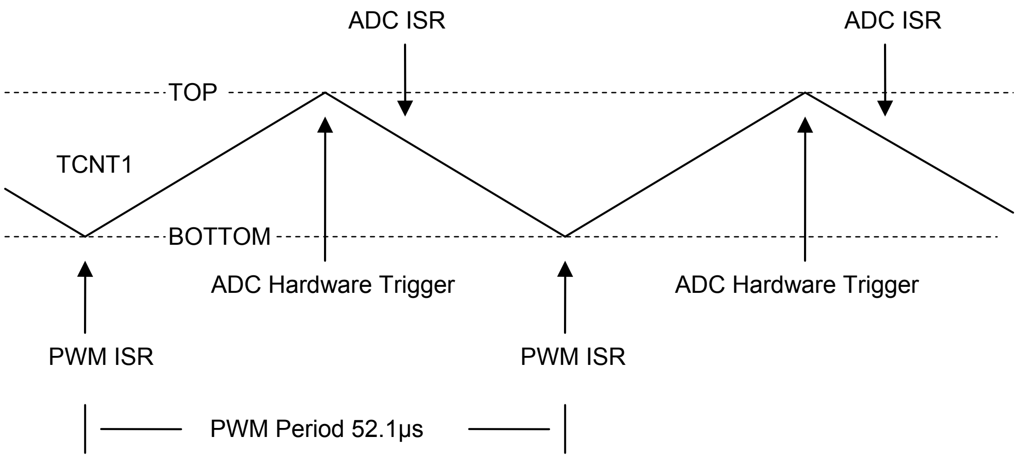
The ADC ISR, Figure 2-8, is based off the "end of conversion" interrupt. The ADC starts a conversion off a TOP count from timer 1. After the conversion time of 13µs the ADC ISR is triggered. The ADC ISR is used to read current or back EMF. The routine is set up to alternate every other ISR. This allows both values to be available for their respective routines in the main loop. The back EMF samples are also averaged in the ADC ISR to calculate the average value as half of the DC bus voltage reference.

Figure 2-8. ADC ISR.



The relationship between the PWM ISR, the ADC ISR, and the up/down count sequence of timer 1 is depicted in Figure 2-9.

Figure 2-9. Timer 1, phase and frequency correct PWM mode with ISRs.



The Commutation ISR, Figure 2-10, is based on timer 0 in 16-bit Compare mode. The timer is loaded with a compare value that interrupts at a frequency proportional to the motor speed. This frequency generation forms the basis of the PLL used for back EMF sensing. The Commutation interrupt alternates between updating the commutation state and triggering a back EMF sample to be used in the PLL update routine of the main loop. Timer 0 is varying its frequency proportional to speed so the Commutation ISR forms an asynchronous interrupt relative to the PWM ISR and ADC ISR. This is illustrated in Figure 2-11.

Figure 2-10. Commutation ISR.

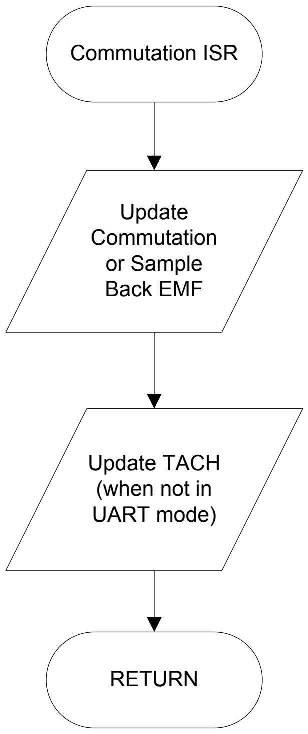
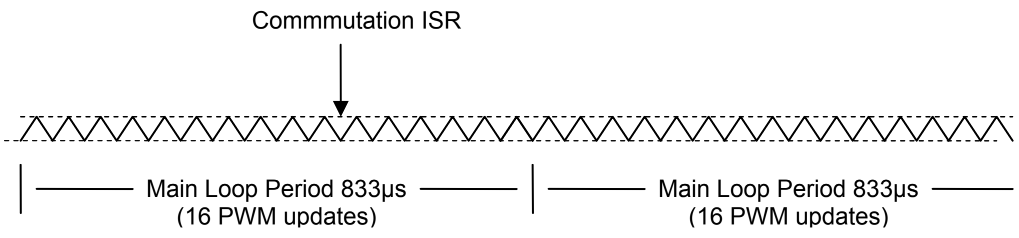

Commutation ISR
Update Commutation or Sample Back EMF
Update TACH (when not in UART mode)
RETURN

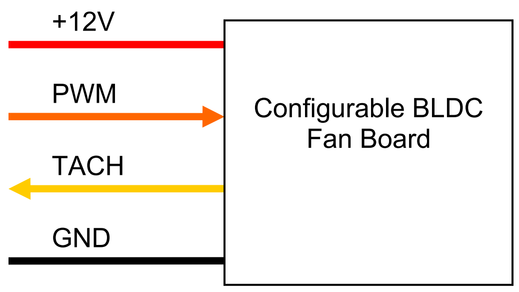
Figure 2-11. Main loop with asynchronous commutation ISR.



2.2.3 Communication

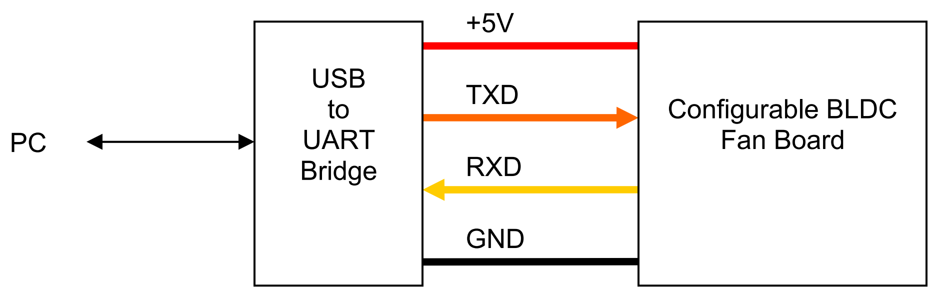
Communication is established by repurposing the single PWM input and single TACH output found in some fan applications. The PWM input is typically a speed command where duty cycle determines fan speed. The TACH output typically represents a square wave output with a frequency that is proportional to speed. This configuration is shown in Figure 2-12.

Figure 2-12. Normal fan operation.



The PWM speed command is repurposed as a UART transmit (receive for the fan) from a USB to UART bridge. TACH is repurposed as a UART receive (transmit for the fan) from a USB to UART Bridge, Figure 2-13. These conventions are used to be consistent with the markings typically found on a USB to UART Bridge.

Figure 2-13. Configuration and tuning.



2.3 Motor Control Configuration Utility

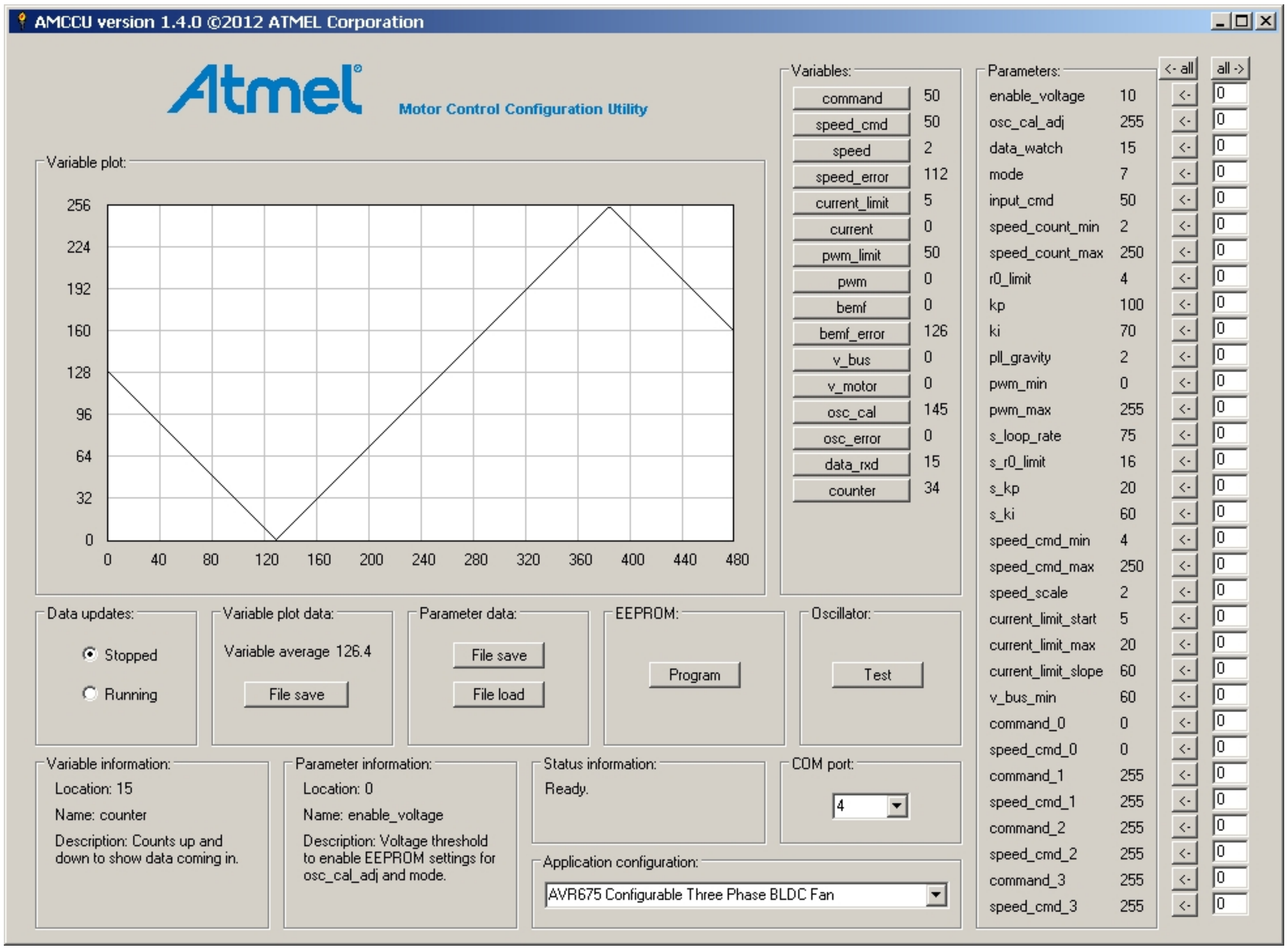
The PC based Motor Control Configuration Utility is for configuration and tuning of the motor performance using a virtual COM port on the PC. The Utility provides the following features:

1. Adjustable Parameters for customizing the code to operate with different fans and sending these parameters to the EEPROM.
2. A list of variables that update every 800ms.
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 ATtiny461A 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.

2.3.1 Layout of utility

The Motor Control Configuration Utility consists of a single front panel with all the functionality displayed in one window. Figure 2-14 shows the basic layout of the fan configuration utility.

Figure 2-14. Motor Control Configuration Utility layout.



Data updates: [✓ Stopped] Running

Variable plot data: Variable average 126.4

Variables:

command	50
speed_cmd	50
speed	2
speed_error	112
current_limit	5
current	0
pwm_limit	50
pwm	0
bemf	0
bemf_error	126
v_bus	0
v_motor	0
osc_cal	145
osc_error	0
data_rxd	15
counter	34

Parameters:

enable_voltage	10	0
osc_cal_adj	255	0
data_watch	15	0
mode	7	0
input_cmd	50	0
speed_count_min	2	0
speed_count_max	250	0
r0_limit	4	0
kp	100	0
ki	70	0
pll_gravity	2	0
pwm_min	0	0
pwm_max	255	0
s_loop_rate	75	0
s_r0_limit	16	0
s_kp	20	0
s_ki	60	0
speed_cmd_min	4	0
speed_cmd_max	250	0
speed_scale	2	0
current_limit_start	5	0
current_limit_max	20	0
current_limit_slope	60	0
v_bus_min	60	0
command_0	0	0
speed_cmd_0	0	0
command_1	255	0
speed_cmd_1	255	0
command_2	255	0
speed_cmd_2	255	0
command_3	255	0
speed_cmd_3	255	0

Variable information:
Location: 15
Name: counter
Description: Counts up and down to show data coming in.

Parameter information:
Location: 0
Name: enable_voltage
Description: Voltage threshold to enable EEPROM settings for osc_cal_adj and mode.

Status information:
Ready.

COM port: 4

Application configuration: AVR675 Configurable Three Phase BLDC Fan

2.3.2 Variables

Variables from the firmware are sent to the configuration utility at the rate of approximately once per 800ms. The list of variables was chosen as the most useful for configuration and tuning of the fan.

2.3.3 Variable plot

One variable from the list can be displayed in the variable plot window. This variable is sent very other loop update or approximately once every 1.6ms. The complete plot is updated once every 800ms with the variable list.

2.3.4 Parameters

The parameter list of values are also updated once every 800ms along with the variable list. During the configuration and tuning process, new values can be sent to the Configurable Fan and the new value will be reflected in the 800ms 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 “Program” button must be pressed.

2.3.5 File functions

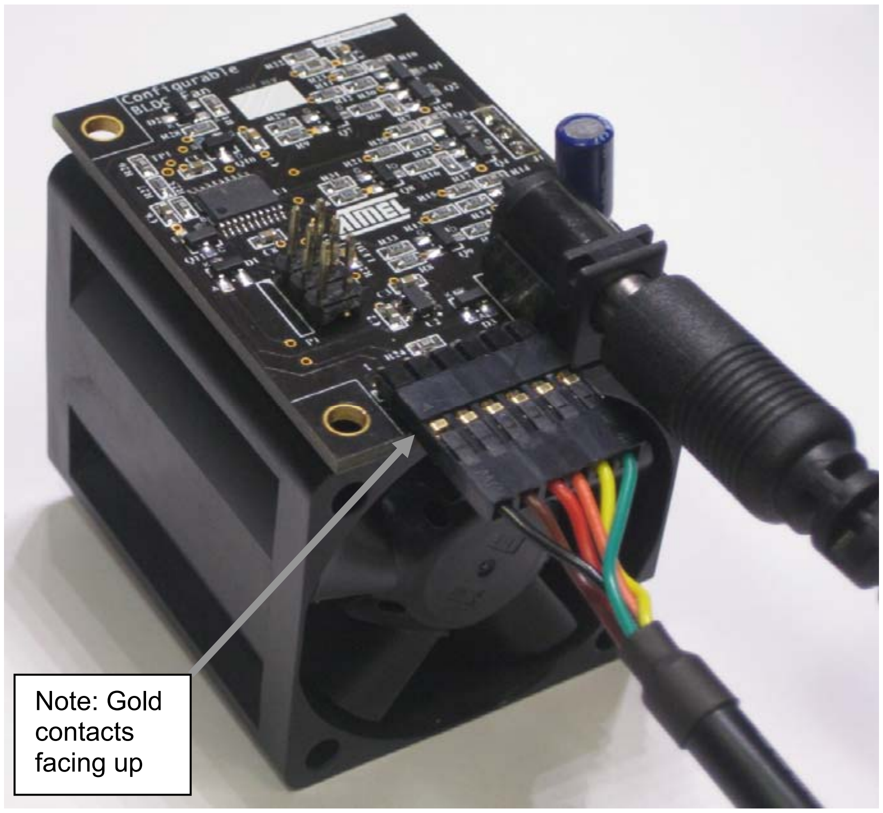
Additional file functions are there for configuring the utility, loading and saving parameters values, and saving variable plot data.

3. Running the System

3.1 Hardware

The system is connected as shown in Figure 3-1. Initially the Configurable BLDC fan PCB can be powered directly from the USB port, this powers up the Atmel ATtiny461A allowing communication with the Motor Control Configuration Utility without running the fan.

Figure 3-1. Configurable Three Phase BLDC Fan with USB to UART Bridge and Universal Power Adapter connected.



3.2 Firmware

For first time use firmware needs to be installed on the ATtiny461A before using the Motor Control Configuration Utility. There are provisions for ISP downloads of firmware and fuse settings through the standard six pin header, P1 using many of the common programming tools and the latest version of Atmel Studio. Firmware can be installed using the following steps:

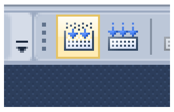
1. Download the AVR Configurable 3-phase BLDC fan project.
2. Plug in USB to UART cable to provide 5V power to the Configurable BLDC Fan PCB while noting the polarity of the connector from Figure 3-1 since this connector is not keyed.

Warning: Do not supply 12V power to the board as the PWM pins are shared the ISP pins this can damage the MOSFETs during programming.

3. Connect ISP programmer to PC through USB connector and connect programmer ribbon cable to P1, match up pin 1 PCB 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 Atmel ATtiny461A. Replace the main.c with the downloaded version. This project is written for the GCC compiler as a single file: main.c so it is easily converted to a newer version of Atmel Studio.
5. Select the GCC Compiler optimization to -Os for size.
6. Build the solution to check for compiling errors, Figure 3-2.

Figure 3-2. Build solution button.

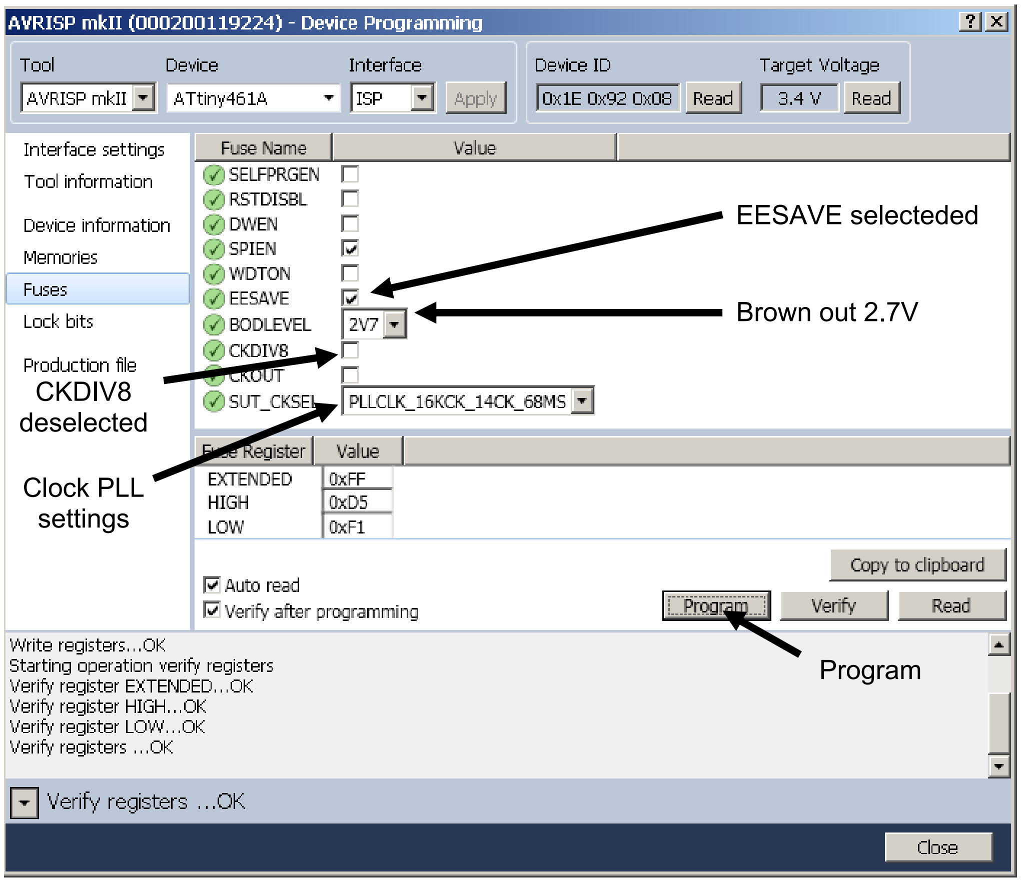


7. Select the Atmel AVR programming button for fuse settings, Figure 3-2.

Figure 3-3. AVR programming button.

8. Connect to the ATtiny461A and set the fuses to the configuration shown in Figure 3-3. The CKDIV8 should be de-selected and the SUT_CKSEL should be selected as PLLCK_16KCK_14CK_68MS. The BODLEVEL should be set to 2V7 for a brownout level of 2.7V. This set the system clock to internal 16MHz operation. EESAVE is optional; this allows firmware downloads without erasing the EEPROM.

Figure 3-4. AVR programming fuse settings.



AVRISP mkII (000200119224) - Device Programming

Tool	Device	Interface	Device ID	Target Voltage
AVRISP mkII	ATtiny461A	ISP	0x1E 0x92 0x08	3.4 V

Fuse Name	Value
SELFPRGEN	☐
RSTDISBL	☐
DWEN	☐
SPIEN	☑
WDTON	☐
EESAVE	☑
BODLEVEL	2V7
CKDIV8	☐
CKOUT	☐
SUT_CKSEL	PLLCLK_16KCK_14CK_68MS

Fuse Register	Value
EXTENDED	0xFF
HIGH	0xD5
LOW	0xF1

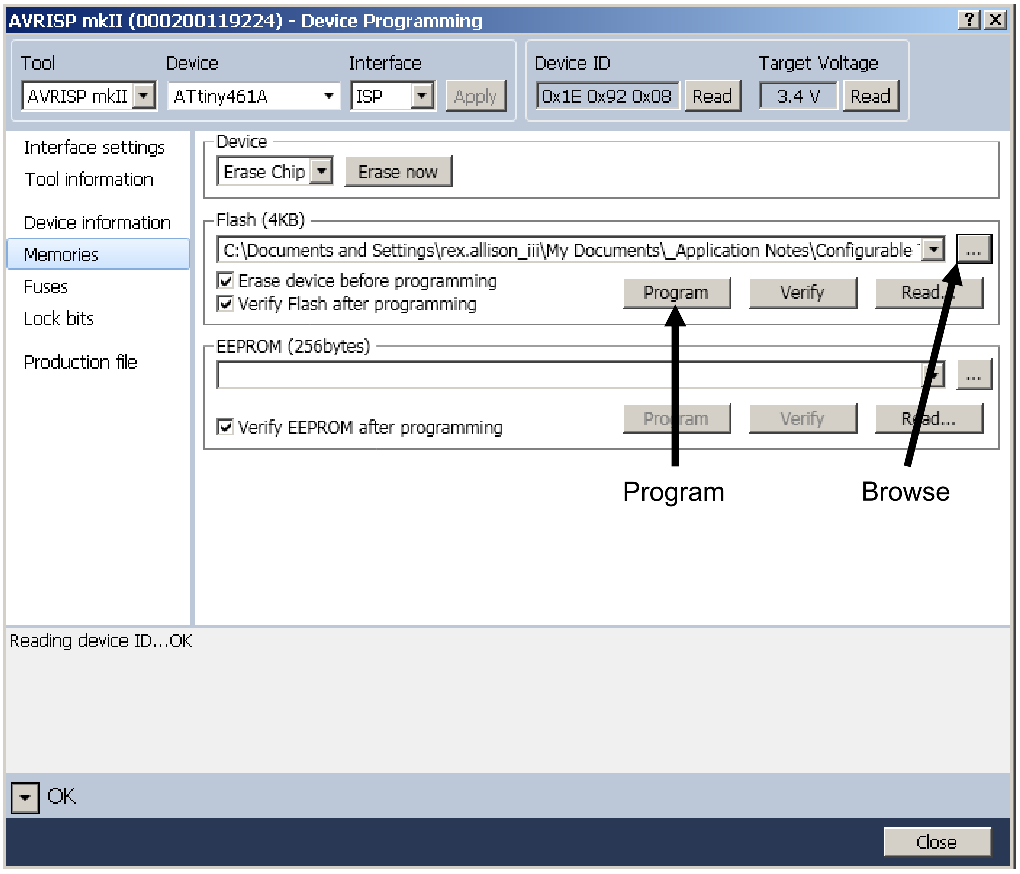
☑ Auto read
☑ Verify after programming

```
Write registers...OK
Starting operation verify registers
Verify register EXTENDED...OK
Verify register HIGH...OK
Verify register LOW...OK
Verify registers ...OK
```

Verify registers ...OK

9. To download the firmware navigate to the Memories section, browse to the .HEX file and press the program button.

Figure 3-5. Start without debugging button.



Device ID: 0x1E 0x92 0x08

Target Voltage: 3.4 V

Flash (4KB): C:\Documents and Settings\rex.allison_iii\My Documents_Application Notes\Configurable

☑ Erase device before programming

☑ Verify Flash after programming

☑ Verify EEPROM after programming

Reading device ID...OK

OK

The blue LED should blink as an indicator that the firmware is ready to use.

3.3 Motor Control Configuration Utility

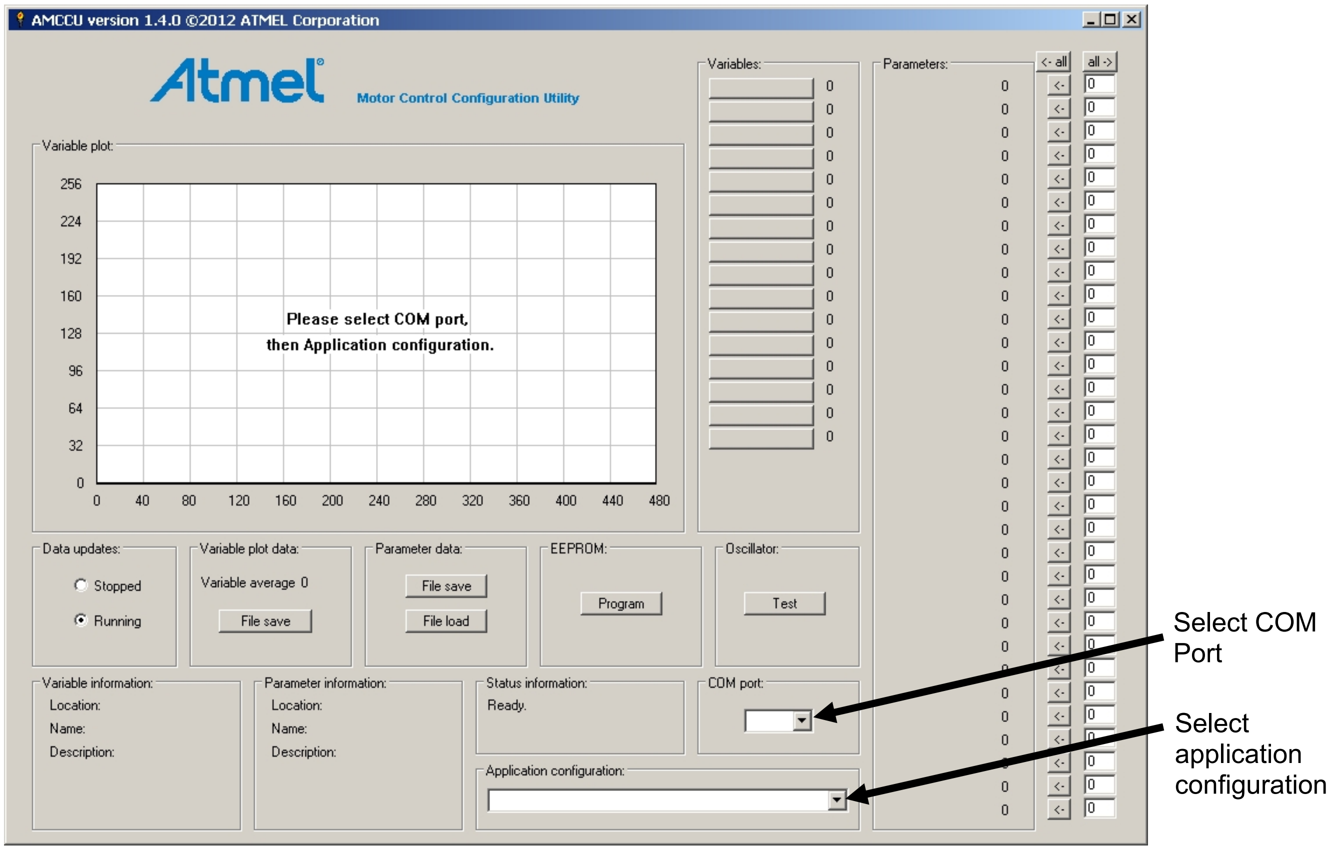
Once the hardware is set up and the firmware is present in the Atmel ATtiny461A flash you are ready to run the Motor Control Configuration Utility.

3.3.1 Running the Motor Control Configuration Utility

The Atmel Motor Control Configuration Utility as a compressed zip file for Windows® XP. The zip file should be extracted at the root directory (C:/) as the folder Atmel Motor Control Configuration Utility. The folder contains the executable file "Atmel Motor Control Configuration Utility ver1_1_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 3-6. Use the drop down box to select the COM port.

Figure 3-6. Atmel Motor Control Configuration Utility when first running.



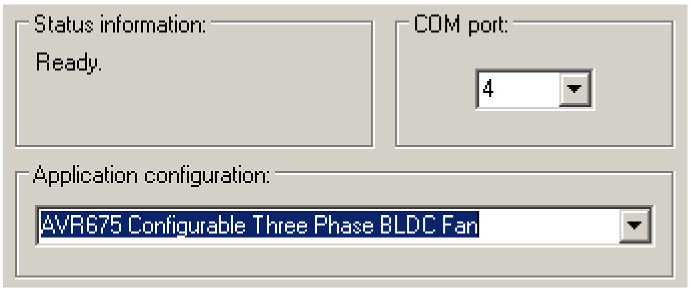
Please select COM port,
then Application configuration.

Data updates: [✓ Running]

Status information: Ready.

[Figure 3-7](#) shows an example of COM port four selected. If the COM port is unknown it can be found in the Windows XP 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 AVR675. This presented the proper variable and parameter list labels in the Motor Control Configuration Utility.

Figure 3-7. COM Port and application configuration boxes after selection.



Status information: Ready.

COM port: 4

Application configuration: AVR675 Configurable Three Phase BLDC Fan

3.3.2 Motor variables and variable plot

If the system is communicating with the Motor Control Configuration Utility, the variables and parameters should fill up with values and the variable "Counter" should be changing every 800ms. If the parameter data_watch (Figure 3-8) is set to 15, the counter variable should be displayed on the Variable Plot screen as a triangle waveform as shown in Figure 3-9.

Figure 3-8. data_watch parameter.

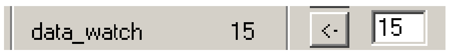
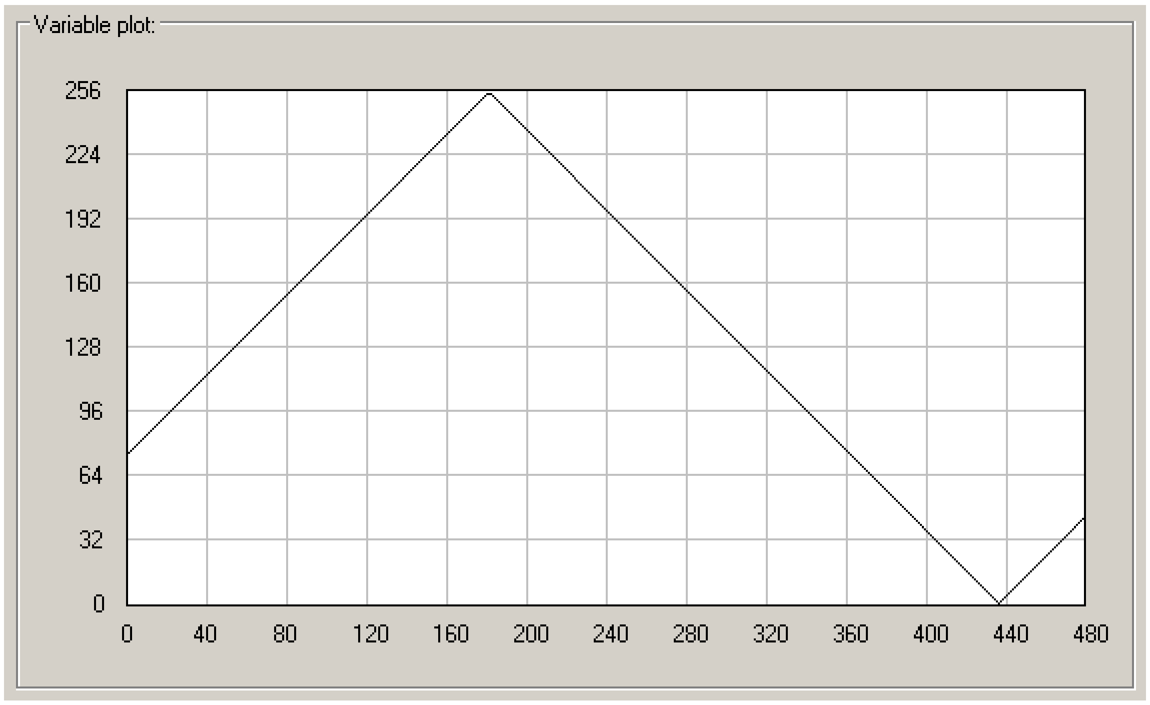


Figure 3-9. Variable plot screen displaying "counter" variable.

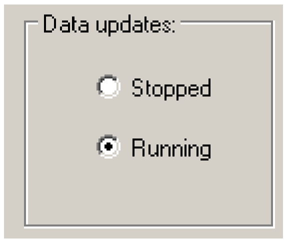


If the Variable plot is not displaying counter, then the counter function is selected to the variable plot screen by pressing the counter button, Figure 3-10.

Figure 3-10. Counter button.

The variable plot and updates of variables and parameters can be stopped using the radio buttons shown in Figure 3-11. This can be used to freeze data on the plot area, to read variables, and for capturing data to be saved to a file.

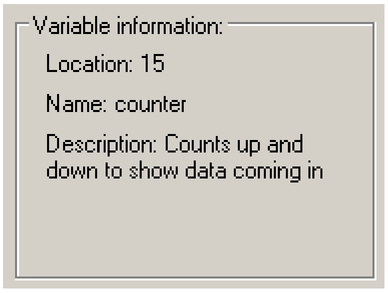
Figure 3-11. Data updates.


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 3-12.

Figure 3-12. Variable information displayed after selecting Counter button.



Variable information:

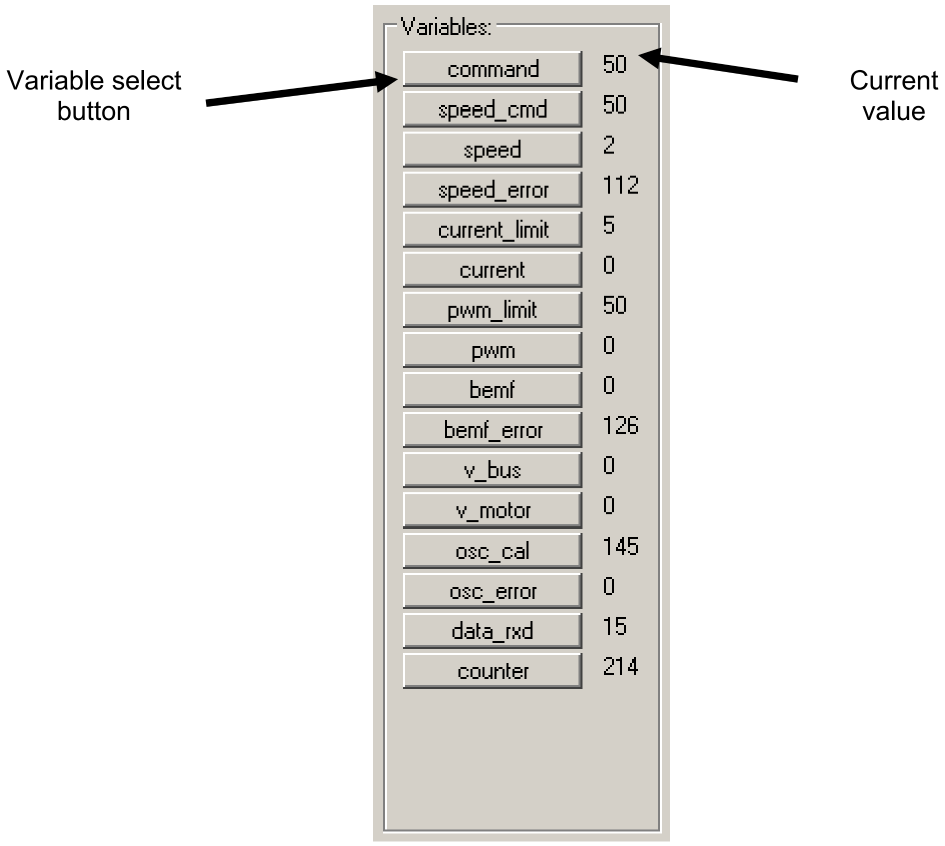
Location: 15

Name: counter

Description: Counts up and down to show data coming in

The complete list of 16 variables is shown in Figure 3-13.

Figure 3-13. The variable list.



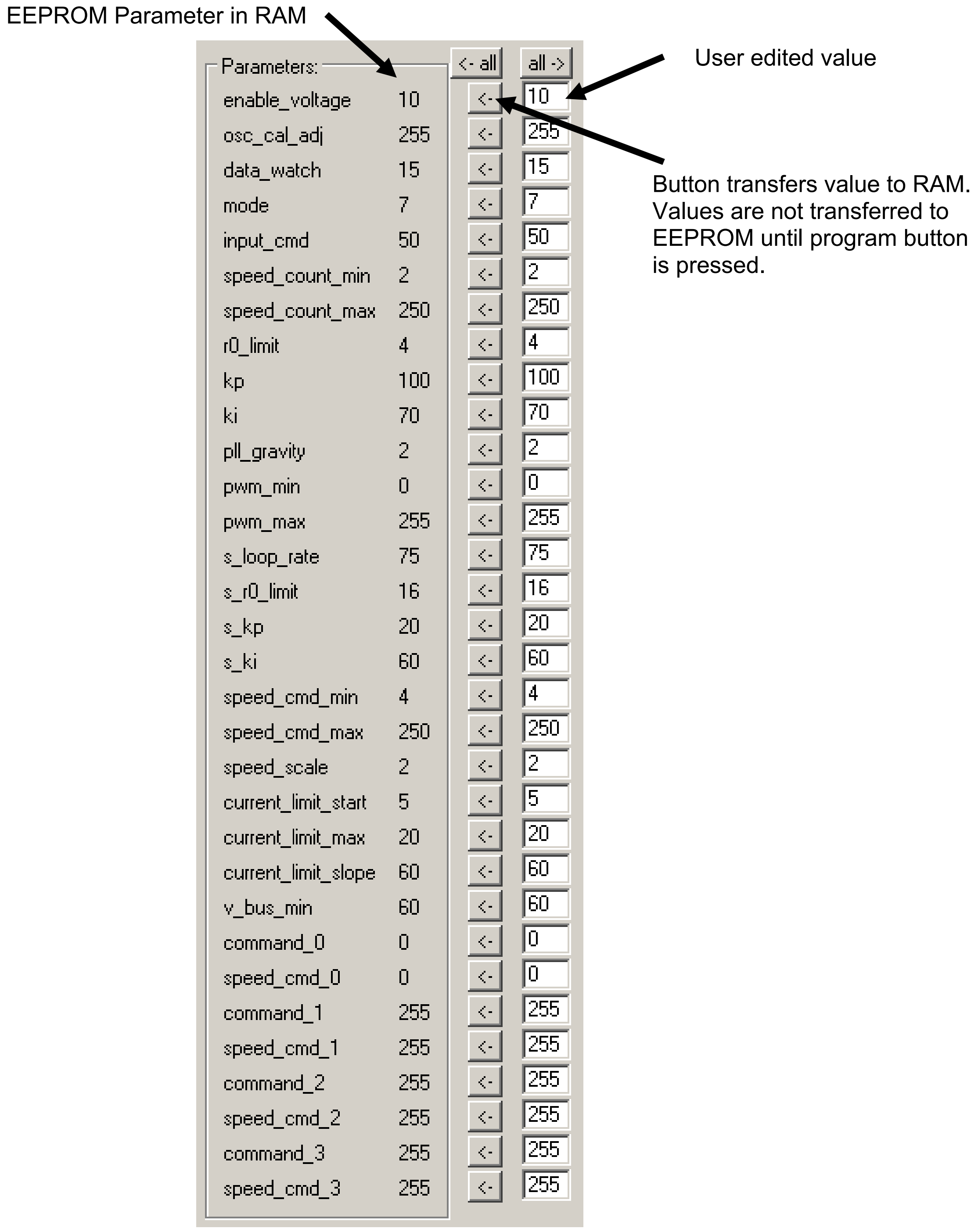
command	50
speed_cmd	50
speed	2
speed_error	112
current_limit	5
current	0
pwm_limit	50
pwm	0
bemf	0
bemf_error	126
v_bus	0
v_motor	0
osc_cal	145
osc_error	0
data_rxd	15
counter	214

3.3.3 Motor parameters

The motor parameters are EEPROM values copied into RAM on power up of the Atmel ATtiny461A.

Figure 3-14 shows the list of parameters.

Figure 3-14. The parameter list.

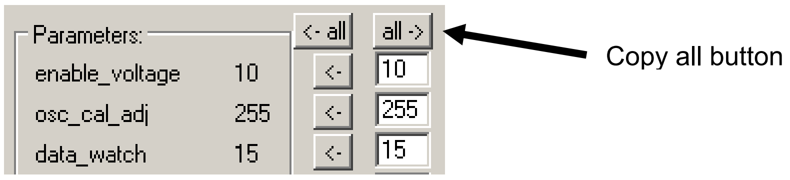

EEPROM Parameter in RAM
User edited value
Button transfers value to RAM. Values are not transferred to EEPROM until program button is pressed.
<- all
all ->

Parameters:

enable_voltage	10	<-	10
osc_cal_adj	255	<-	255
data_watch	15	<-	15
mode	7	<-	7
input_cmd	50	<-	50
speed_count_min	2	<-	2
speed_count_max	250	<-	250
r0_limit	4	<-	4
kp	100	<-	100
ki	70	<-	70
pll_gravity	2	<-	2
pwm_min	0	<-	0
pwm_max	255	<-	255
s_loop_rate	75	<-	75
s_r0_limit	16	<-	16
s_kp	20	<-	20
s_ki	60	<-	60
speed_cmd_min	4	<-	4
speed_cmd_max	250	<-	250
speed_scale	2	<-	2
current_limit_start	5	<-	5
current_limit_max	20	<-	20
current_limit_slope	60	<-	60
v_bus_min	60	<-	60
command_0	0	<-	0
speed_cmd_0	0	<-	0
command_1	255	<-	255
speed_cmd_1	255	<-	255
command_2	255	<-	255
speed_cmd_2	255	<-	255
command_3	255	<-	255
speed_cmd_3	255	<-	255

The copy all buttons shown in Figure 3-15 provides a quick method to transfer all the variables between the Motor Control Configuration Utility and the ATtiny461A.

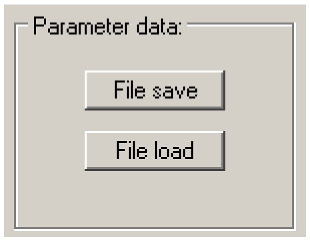
Figure 3-15. Copying all parameters.



Parameters:			
enable_voltage	10	<-	10
osc_cal_adj	255	<-	255
data_watch	15	<-	15

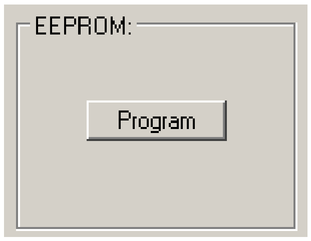
The white boxes on the far right column of Figure 3-14 are for editing and transferring data between the configuration utility and files. The file transfer is a simple file saves or file load (Figure 3-16) to a text file EEPROM.txt that is stored in the same directory that the Motor Control Configuration Utility is launched. Backups of this file must be copied and renamed to prevent overwriting from the utility.

Figure 3-16. Parameter data load and save.



To transfer the parameters stored in RAM back to the EEPROM the program button is pressed, Figure 3-17.

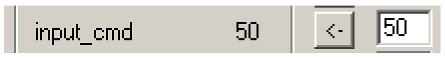
Figure 3-17. Program Parameters in EEPROM.



The parameter input_cmd is used for directly sending a PWM or speed command during testing. This simulates the PWM command that would be sent to the fan if it were not in communication mode. The input command parameter is shown in Figure 3-18. 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 ATtiny461A and the program button still needs to be pressed to commit the RAM values to EEPROM.

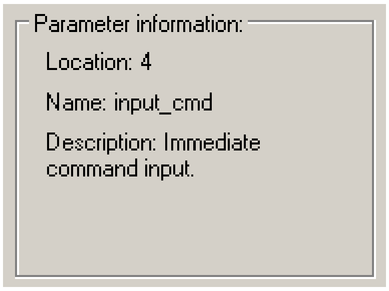
Figure 3-18. input_cmd direct input.



input_cmd	50	<-	50

When the arrow key is pressed a description of the parameter shows up in the parameter information box, Figure 3-19.

Figure 3-19. Parameter information.


Parameter information:
Location: 4
Name: input_cmd
Description: Immediate command input.

3.3.4 Enable voltage

The enable_voltage sets a threshold for the DC input voltage (v_bus) above which the oscillator adjustment and mode settings are enabled. These settings are described in the next two sections. The scaling for enable_voltage is based on the voltage dividers for back EMF sensing and the ADC on the ATtiny461A. For this application the scaling is based on the following equation:

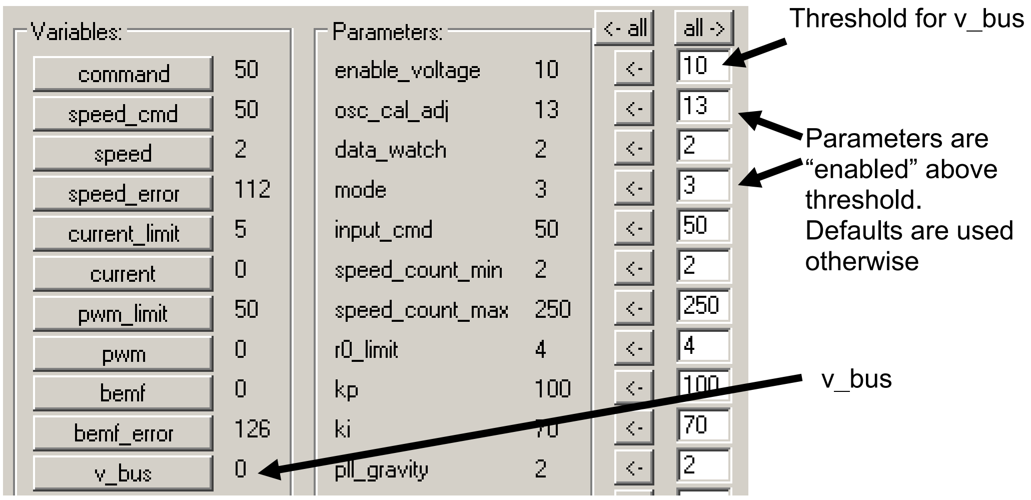
$$v\_bus = Vcc\_12 \text{ x } Resistive\ divider / ADC\ full\ scale\ voltage * ADC\ resolution \quad \textbf{(1)}$$

At an input of 12Vdc:

$$v\_bus = 12V \text{ x } 2k / (2k + 15k) /5V * 256 = 72 \quad \textbf{(2)}$$

Approximately six counts per volt, an example of a 1.7V threshold is shown in Figure 3-20.

Figure 3-20. Enable voltage threshold set to 1.7V (10 counts).



Variables:
Parameters:
<- all
all ->
Threshold for v_bus
Parameters are "enabled" above threshold. Defaults are used otherwise
v_bus

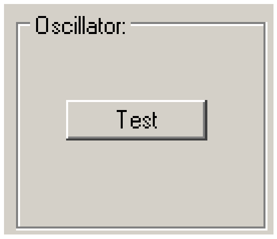
Variables		Parameters		
command	50	enable_voltage	10	10
speed_cmd	50	osc_cal_adj	13	13
speed	2	data_watch	2	2
speed_error	112	mode	3	3
current_limit	5	input_cmd	50	50
current	0	speed_count_min	2	2
pwm_limit	50	speed_count_max	250	250
pwm	0	r0_limit	4	4
bemf	0	kp	100	100
bemf_error	126	ki	70	70
v_bus	0	pll_gravity	2	2

The default configuration of factory oscillator calibration and UART transmit, receive are enabled and open loop PWM (mode = 7). This feature allows changes that loose PC communication such as PWM input mode to be reestablished by powering the board only from the USB to UART bridge cable. The 5V from the USB to UART cable only powers up the Atmel ATtiny461A microcontroller and not the DC input bus.

3.3.5 Oscillator calibration

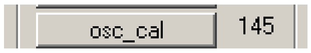
The oscillator calibration can be checked by pressing the Test button as shown is Figure 3-21. The internal oscillator is compared with the USB to UART cable oscillator which has an oscillator accuracy of better than 0.2%. See ATtiny461A Data Sheet for further details about oscillator calibration.

Figure 3-21. Oscillator test button.



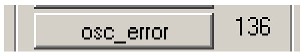
The factor oscillator calibration value is displayed under the variable osc_cal, Figure 3-22. This represents the factory calibration value that is copied into the oscillator calibration register (OSCCAL).

Figure 3-22. Oscillator calibration value.


osc_cal 145

Once the calibration button is pressed the osc_error, Figure 3-23, 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 Motor Control 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 3-23. Oscillator calibration error.

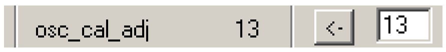

osc_error 136

The osc_cal can be adjusted by updating the osc_cal_adj parameter, Figure 3-24, and programming the EEPROM. osc_cal_adj is a correction factor to the original factory calibration. The range of values is from 0 to 32, where 16 correspond to 0 corrections. Values from any value above 32 are ignored and only the factory calibration is used.

The equation for Oscillator cal adjustment for osc_cal_adj < 33:

$$\textit{Adjusted osc\_cal} = \textit{osc\_cal} + \textit{osc\_cal\_adj} - 16 \qquad \textbf{(3)}$$

Figure 3-24. Oscillator calibration adjustment parameter, example for -3 adjustment to factory calibration.


osc_cal_adj 13 <- 13

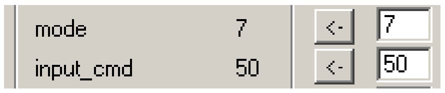
Please note that this value is not enabled until the DC input voltage has exceeded the enable_voltage threshold.

3.3.6 mode and input_cmd

The mode parameter determines the behavior of the fan I/O and how the input_cmd parameter is interpreted. Changing modes helps with the configuration and tuning of the fan.

There are two control loops to tune, the back EMF sensing PLL and the speed control loop. The back EMF sensing PLL should be tuned first, to get rotor position sensing and start-up working properly then the speed control loop can be tuned for proper speed response. The first step in tuning the back EMF sensing PLL is to disable the speed control loop and use open loop PWM duty cycle control by setting mode to seven, Figure 3-25.

Figure 3-25. mode for Open Loop PWM.



mode	7	<-	7
input_cmd	50	<-	50

As can be seen in Table 3-1, there eight possible Mode settings, the default for tuning is mode = 7. When the motor is tuned and configured, mode = 0 gives typical fan behavior of closed loop speed control with the speed command coming from the PWM input and speed output as a tachometer. The other modes one through six allows different combinations of I/O and control.

Table 3-1. mode options.

mode	Control state (bit 2)	Output state (bit 1)	Input state (bit 0)
0	0 = Speed control	0 = Tachometer output	0 = PWM command in
1	0 = Speed control	0 = Tachometer output	1 = UART receive
2	0 = Speed control	1 = UART transmit	0 = PWM command in
3	0 = Speed control	1 = UART transmit	1 = UART receive
4	1 = Open Loop PWM	0 = Tachometer output	0 = PWM command in
5	1 = Open Loop PWM	0 = Tachometer output	1 = UART receive
6	1 = Open Loop PWM	1 = UART transmit	0 = PWM command in
7	1 = Open Loop PWM	1 = UART transmit	1 = UART receive

Note: An Input State = 0 the PWM duty cycle overrides the input_cmd parameter and for Input State = 1, the input_cmd parameter is used directly.

3.3.7 Tuning the back EMF sensing PLL

Tuning the back EMF sensing PLL has four goals:

1. Reliable start-up.
2. Dynamic response - tracking of speed changes.
3. Steady state - stable operation at constant speed.
4. Recovery from a stall condition.

The basis for tuning the PLL is setting the frequency range: speed_count_min to speed_count_max, Figure 3-26. The relationship between speed count and motor RPM is:

Motor speed in revolutions per minute (rpm):

$$\omega_m = \left(\frac{f_{clk}}{prescaler}\right)\left(\frac{Timer\_scale}{Timer\_period}\right)\left(\frac{Speed\_count}{timer\_periods}\right)\left(\frac{2}{N_{poles}}\right)\left(\frac{60\sec}{1\min}\right) \quad (4)$$

where:

fclk = Main Oscillator frequency = 16MHz

prescaler = timer 0 prescaler = 8

timer_scale = timer software prescaler = 2 x speed_scale

timer_period = 0xFFFF

speed_count = numeric speed value = 0 to 255

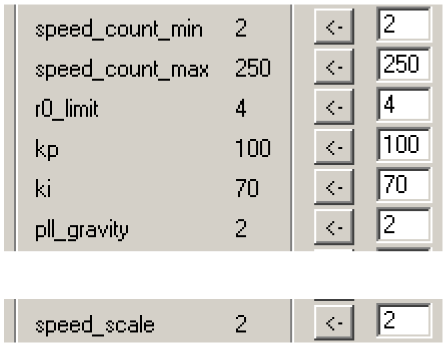
timer_periods = 12 timer periods per electrical cycle (2 periods per state x 6 states)

Npoles = number of magnetic poles (Npoles/2 = pole pairs and 2/Npoles = 1/pole pairs)

60sec/min = conversion factor from 1/sec to 1/minutes

$$\omega_m = \frac{speed \times 305 \times 2^{speed\_scale}}{N_{poles}} \tag{5}$$

Figure 3-26. PLL parameters.



speed_count_min	2	<-	2
speed_count_max	250	<-	250
r0_limit	4	<-	4
kp	100	<-	100
ki	70	<-	70
pll_gravity	2	<-	2
speed_scale	2	<-	2

The back EMF sensing PLL is compensated with a proportional integral (PI) controller. See Application Note AVR221: Discrete PID Controller for further details about PI and PID controllers. kp is the proportional gain and ki is the integral gain and r0_limit is an error clamp that limits the amount of error that is processed by the PI controller. Tuning the PLL requires some iteration of kp and ki. The goal is to find a combination of kp and ki that runs the motor in open loop mode. The operation doesn't have to be particularly smooth or reliable, but these values become the starting point for tuning. The first rule of thumb is to start with small gains for both kp and ki. Then increase kp while keeping ki constant. If no results are found increase ki and start search through kp values again. Repeating this process should yield a combination that runs the motor.

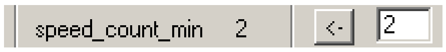
Once working values are found for kp and ki, they must be tested over the speed range to check for stable operation at steady state and dynamic load changes. This can be tested by putting in step commands into the input_cmd parameter when in mode 7. For example step from a command of 50 to 100 by entering 100 into the input box for input_cmd and pressing the arrow key to send the updated command to the fan. Larger steps should be tested as well as steps in both directions (50 to 100 and 100 to 50). Fine adjustment of the kp and ki parameters can be done by watching the speed variable on the Variable plot window to minimize ripple on the speed variable at steady state. There is always a trade off of smoothness of speed at steady state and dynamic response that should be determined by the application requirements.

Adjustment of the error limiter r0_limit is mostly for out of lock conditions, stall or start-up. When the back EMF sensing PLL is running the error tends to be small so the error is not clamped. The parameter is implemented as another fine tuning adjustment if a particular motor is difficult to start.

3.3.8 Fine tuning startup

For starting the motor the first parameter to set is the speed_count_min. This is the minimum frequency of the back EMF sensing PLL. The frequency should be set so that the motor starts stepping through the states. During start up the motor should rotate slowly, but distinctly. Figure 3-27 shows the speed_count_min for the example fan, the value is usually in the range of 1 to 10.

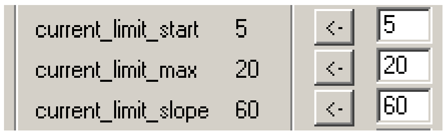
Figure 3-27. Minimum speed count.



speed_count_min	2	<-	2

The parameter that works with speed_count_min is current_limit_start, is the current_limit_start parameter, shown in Figure 3-28, that sets the current limit threshold when the back EMF sensing PLL is running at the minimum frequency.

Figure 3-28. Current limiting parameters.



current_limit_start	5	<-	5
current_limit_max	20	<-	20
current_limit_slope	60	<-	60

The current limit parameters allow a settable current limit that gives good startup, but also allows a higher current for high speed operation. This is especially important in fan applications where low inertia and lack of mechanical damping require very little starting current and high speeds where power is the highest to have a higher current threshold. An indication of too high of a current_limit_start is during startup the motor oscillates while stepping through the states. The oscillation is from too much torque causing the motor to overshoot the step position. Too low of current can produce too little current to get the motor to rotate. This can be heard as a buzzing from the motor with very little or no movement.

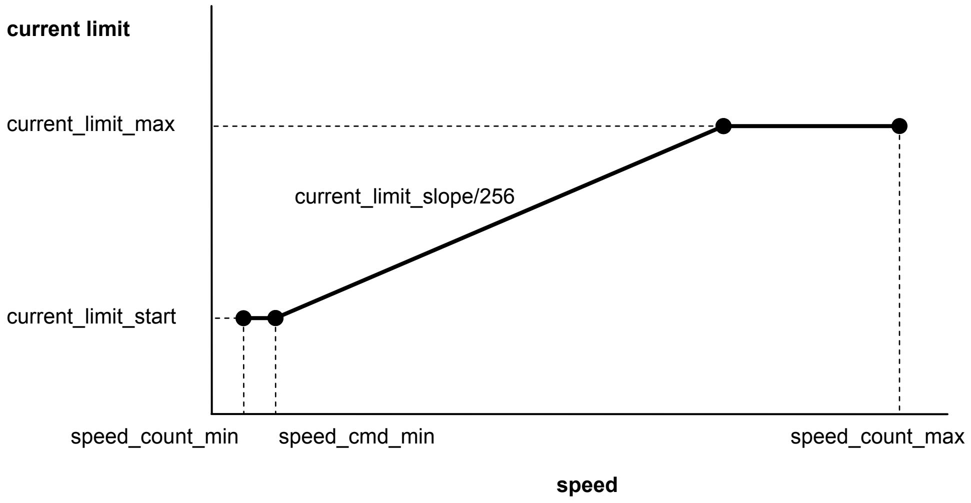
The scaling for current limiting comes from the current sense resistor (R23 from Appendix A) of 0.1Ohms, the ADC reference (3.3V), and the ADC resolution used (8 bits).

current_limit scaling = 0.1Ohm X 256 ADC counts / 5V reference **(6)**

current_limit scaling = 5.12 counts/Amp **(7)**

In this example just 1A is required for startup and 4A for running at full speed without tripping overcurrent. The relationship between current_limit_start, current_limit_max, and current_limit_slope is shown in Figure 3-29.

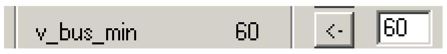
Figure 3-29. Current limiting parameters mapping.



Startup is also controlled by the term v_bus_min. This is an undervoltage lockout for PWM preventing startup until the DC input voltage has reached the threshold. In the example of Figure 3-30 the start up voltage is set to approximately 6V.

*v_bus = 10V x 2k / (2k + 15k) /5V * 256 = 60* **(8)**

Figure 3-30. v_bus_min, minimum start up voltage.


v_bus_min 60 <- 60

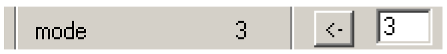
3.3.9 Recovery from stall

If the motor is stalled, due to a mechanical obstruction or an interruption in power, the motor does not produce a back EMF signal. The back EMF sensing PLL does not have a signal to lock onto. To restart the motor the back EMF sensing PLL must go back down to its minimum frequency to restart the motor. The parameter pll_gravity term from Figure 3-26 is used to bias the PLL towards its minimum speed. Without this term the PLL can stay at a high frequency and not restart the motor. The typical value is usually 1 or 2 since only a small bias is required to get the frequency to fall towards its minimum value.

3.3.10 Speed control tuning

The goal of the speed control loop is to maintain control of the PWM dutycyle so that the motor speed tracks the speed_cmd. Speed is controlled by a proportional integral regulator with the output a PWM command. To enable the regulator, the mode is set to three as shown in Figure 3-31. From Table 3-1 it can be seen that mode 3 enables speed control while preserving UART communication (Transmit and Receive) with the Motor Control Configuration Utility.

Figure 3-31. Input mode for Speed Control.

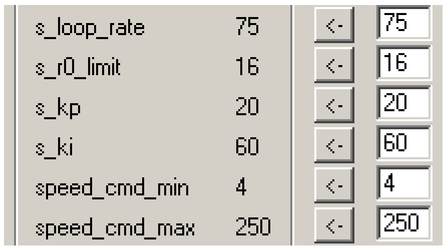

mode 3 <- 3

Since there are two control loops - the back EMF sensing PLL and the speed control loop- the ideal situation is to not have the two loops interact. The back EMF sensing PLL is tracking the rotor position so it must be able to respond to sudden changes in speed, this represents the fastest control loop in the system. To prevent the control loops from interacting, the Speed control loop should look like it is in steady state relative to the back EMF sensing PLL. This is accomplished by running the speed loop at a slower rate by using the parameter s_loop_rate. s_loop_rate determines the number of 800ms updates between the update of the speed control loop.

The s_r0_limit parameter is the speed error clamp; this limits the amount of speed error that is used by the PI control loop. s_kp is the proportional gain and s_ki is the integral gain. s_kp and s_ki can be tuned using traditional PI control tuning methods, Figure 3-32:

1. First set s_ki = 0 to turn off the integral term
2. Select the variable plot to be speed.
3. Start with a low value of s_kp and start the motor, with proportional control there will be a steady state error. s_kp can be increased to lower the steady state error. At some point, large values of s_kp will amplify the noise in the system. If increasing s_kp causes noisy operation, the value should be backed off to reduce this effect.
4. Once the highest s_kp is set, s_ki can be increased from zero and the motor restarted. As s_ki is increased, the steady state error decreases faster on start-up. The trade off is that high values of s_ki can cause wind up in the control loop that will cause the motor to overshoot the speed reference. The optimal value for s_ki for the application is balancing speed of response and overshoot. Generally fans do not require fast speed transitions and overshoot is undesirable so low values of s_ki give the best performance.
5. Last of all the speed command range can be set with speed_cmd_min and speed_cmd_max. The minimum speed command should be set above the speed_count_min so that the fan is not being commanded to a speed below or equal to the startup frequency. speed_cmd_max is limit to the 8 bit value of 255.

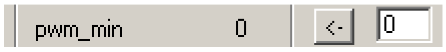
Figure 3-32. Speed control parameters.



s_loop_rate	75	<-	75
s_r0_limit	16	<-	16
s_kp	20	<-	20
s_ki	60	<-	60
speed_cmd_min	4	<-	4
speed_cmd_max	250	<-	250

A minimum PWM value can be set so that the motor always starts or it can be set to zero to allow stopping the motor for a zero input_cmd, see Figure 3-33.

Figure 3-33. Minimum PWM value.

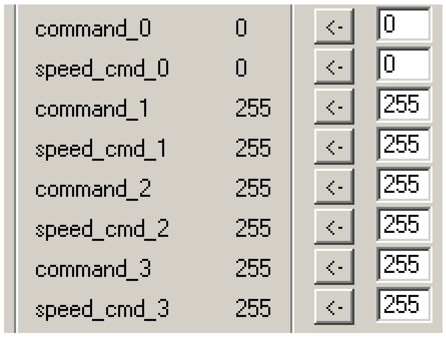


pwm_min	0	<-	0

3.3.11 Mapping Speed Command

A nonlinear mapping of PWM input command to speed_cmd can be set by the parameters command_0 to 3 and speed_cmd_0 to 3 shown in Figure 3-34.

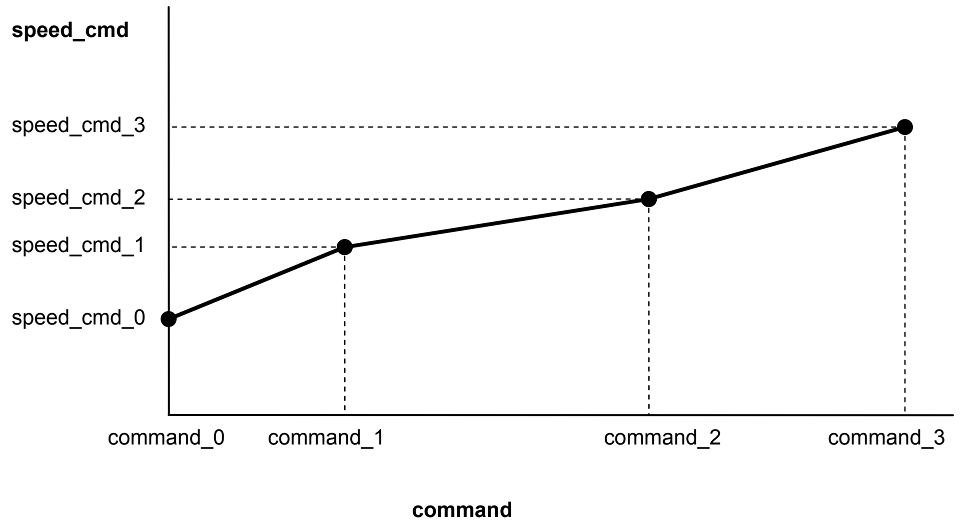
Figure 3-34. Mapping parameters.



command_0	0	<-	0
speed_cmd_0	0	<-	0
command_1	255	<-	255
speed_cmd_1	255	<-	255
command_2	255	<-	255
speed_cmd_2	255	<-	255
command_3	255	<-	255
speed_cmd_3	255	<-	255

An example of nonlinear mapping is shown in Figure 3-35. For linear mapping simple set command_0 = speed_cmd_0 = 0 and command_1 = speed_cmd_1 = 255 and set all remaining values to 255.

Figure 3-35. Example of Nonlinear Mapping of Command to speed_cmd.

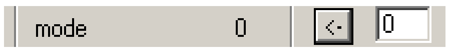


3.3.12 Stand Alone Fan Operation

After configuration and tuning is complete, the EEPROM update should be disabled so the fan operates in stand alone operation mode with PWM input for the speed command and a tachometer output. The steps for configuring fan operation are:

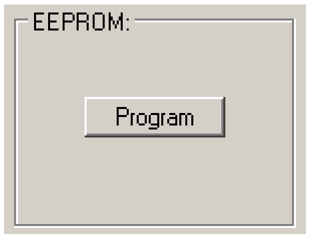
1. Make sure the enable_voltage is set below the ATtiny461A operating voltage of 1.8V such as setting Enable_voltage to 9 for a 1V threshold as previously shown in Figure 3-20.
2. Setting up Mode 0 is shown in Figure 3-36 which is the default for stand alone fan operation from Table 3-1.

Figure 3-36. Mode 0, PWM speed in, Tachometer out and Speed Control.


mode 0 <- 0

3. Program the EEPROM with these settings, Figure 3-37. This requires the processor to be powered from the USB to UART cables 5V and to have the DC input voltage at zero volts to allow UART communication.

Figure 3-37. Program Parameter in EEPROM.



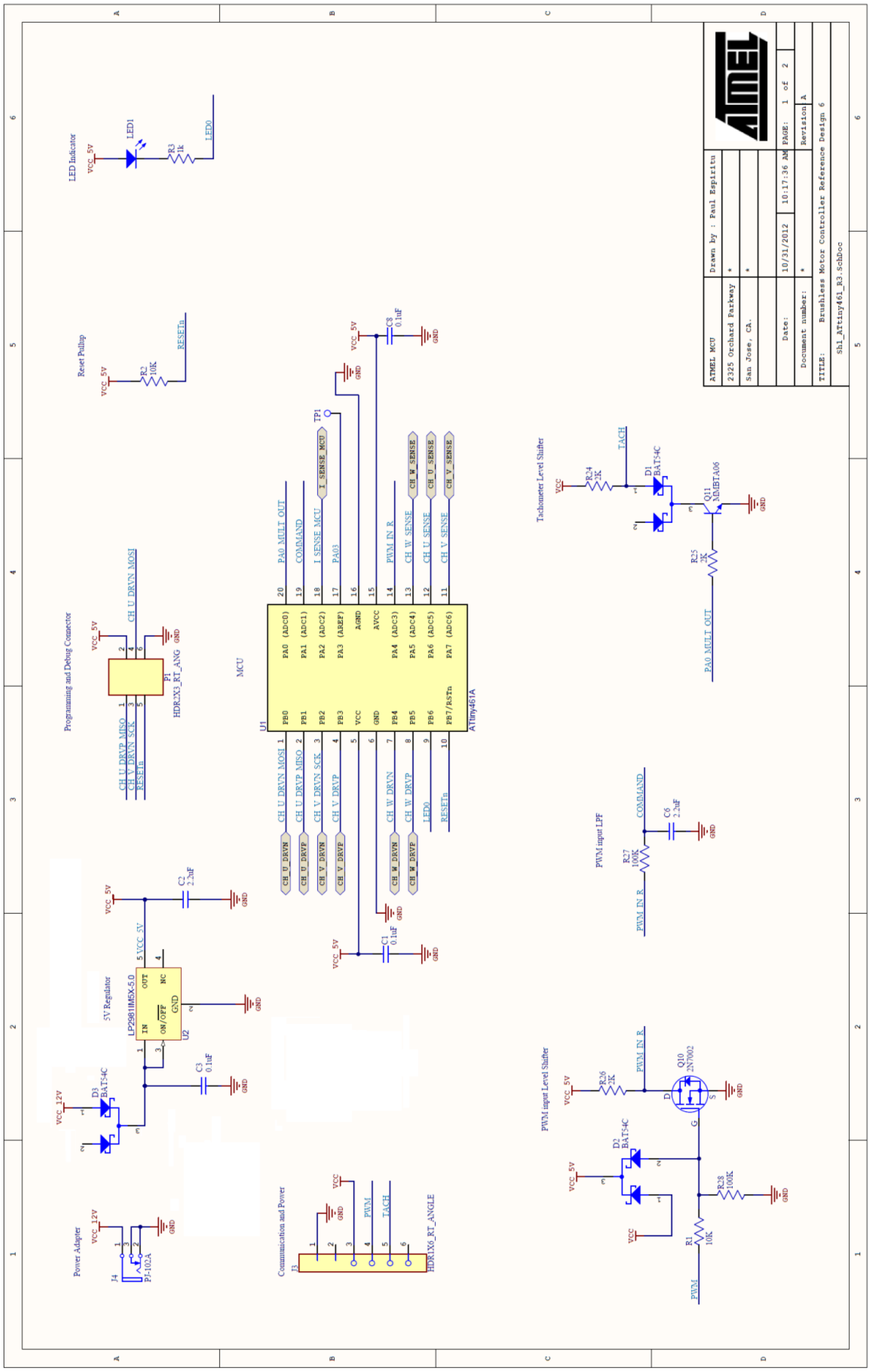
The EEPROM is now programmed. Next time the power is cycled the fan will power up with these settings. As long as power is supplied by the 12V input, serial communication will not be established and the Atmel ATtiny461 only accepts a PWM command. If this operation is done accidentally communication can always be reestablished by powering the ATtiny461A from the USB to UART cable and leaving the DC input voltage at zero.

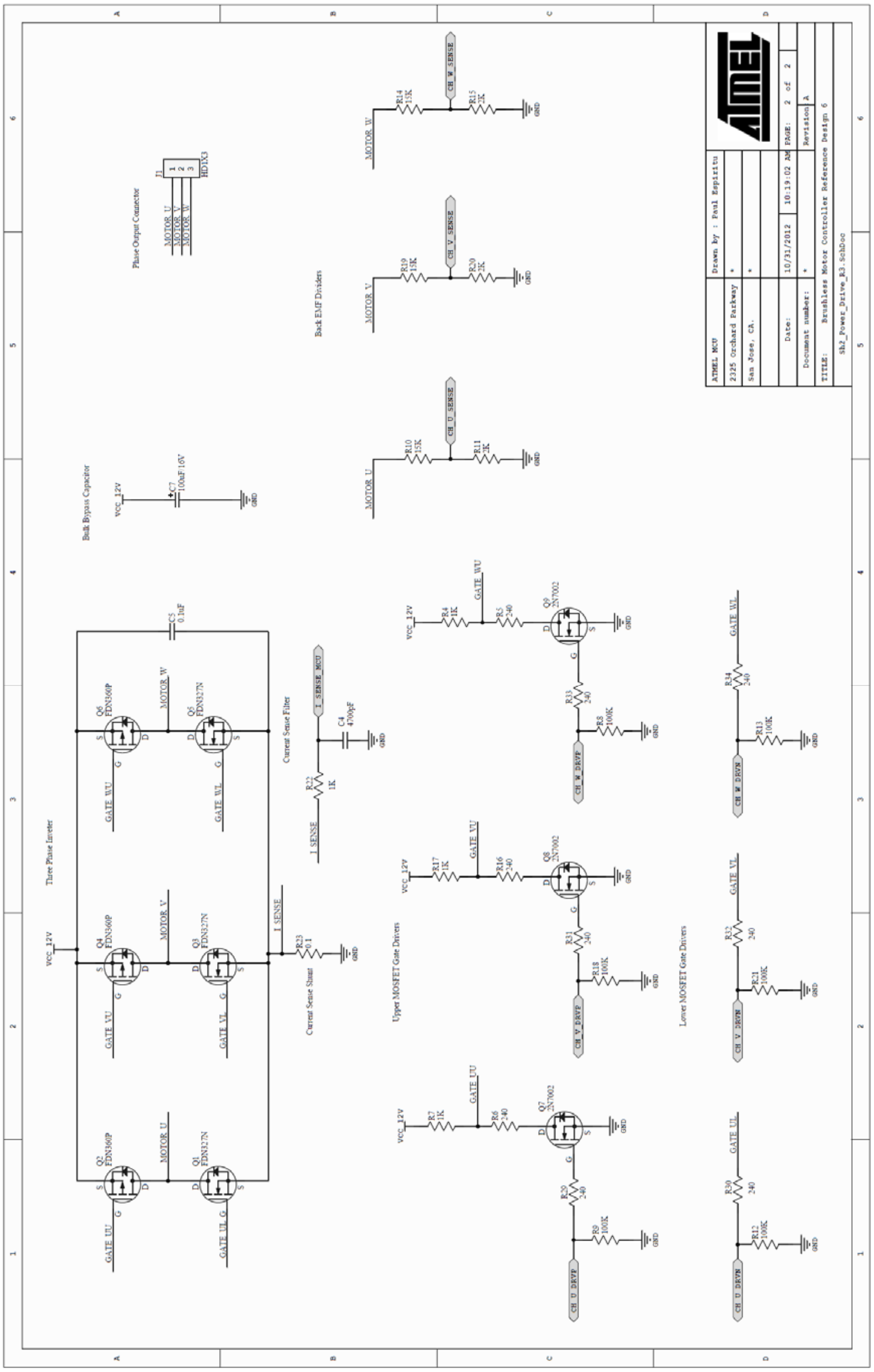
4. Conclusion

This application note described the implementation and use of a configurable three-phase brushless DC (BLDC) fan using the Atmel ATtiny461A microcontroller implementing sensorless control. The complete hardware system includes a 12V three-phase BLDC fan, an Configurable BLDC Fan board, a USB to UART bridge cable, an AC power adapter, the firmware running on the ATtiny461A, and a PC based Motor Control Configuration Utility for customizing the fan for the application.

Appendix A. Schematics

Figure 4-1. Schematics.




Three Phase Inverter
Phase Output Connector
Bulk Bypass Capacitor
Back EMF Dividers
Current Sense Filter
Current Sense Shunt
Upper MOSFET Gate Drivers
Lower MOSFET Gate Drivers
VCC_12V
MOTOR_U
MOTOR_V
MOTOR_W
I_SENSE
I_SENSE_MCU
CH_U_SENSE
CH_V_SENSE
CH_W_SENSE
CH_U_DRVP
CH_V_DRVP
CH_W_DRVP
CH_U_DRVN
CH_V_DRVN
CH_W_DRVN
GATE_UU
GATE_VU
GATE_WU
GATE_UL
GATE_VL
GATE_WL
GND
Q1 FDN327N
Q2 FDN360P
Q3 FDN327N
Q4 FDN360P
Q5 FDN327N
Q6 FDN360P
Q7 2N7002
Q8 2N7002
Q9 2N7002
J1 HDR1X3
C5 0.1uF
C7 100uF/16V
C4 4700pF
R23 0.1
R22 1K
R10 15K
R11 2K
R19 15K
R20 2K
R14 15K
R15 2K
ATMEL MCU
2325 Orchard Parkway
San Jose, CA.
Drawn by : Paul Espiritu
Date: 10/31/2012 10:19:02 AM
PAGE: 2 of 2
Document number:
Revision A
TITLE: Brushless Motor Controller Reference Design 6
Sh2_Power_Drive_R3.SchDoc

Appendix B. Kit Parts List

Description	Manufacturer	Part number
Configurable 3-phase BLDC Fan Board	Atmel Corporation	Configurable 3-phase BLDC Fan Board
Fan 12V 40mm x 48mm 23000RPM	NMB Technologies Corporation	1619FT-04W-B86-B50
USB to UART Bridge Cable	FTDI, Future Technology Devices International Ltd	TTL-232R-5V
Adapter wall R-series 30W 12V	PHIHONG USA	PSAA30R-120
Adapter wall R-series clip US	PHIHONG USA	RPA

Appendix C. Revision History

Doc. Rev.	Date	Comments
42016B	12/2012	Revision history added and some other updates/corrections have been performed
42016A	07/2012	Initial document release

Atmel | 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 Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370