
AVR172: Sensorless Commutation of Brushless DC Motor (BLDC) using ATmega32M1 and ATAVRMC320



Features

- Robust sensorless commutation control
- Ramp-up sequence

References

- [1] ATmega32M1 Data sheet
- [2] AVR194: Brushless DC Motor Control using ATmega32M1
- [3] AVR430: MC300 Hardware User Guide
- [4] AVR470: MC310 User Guide
- [5] AVR471: MC320 Getting Started Guide
- [6] AVR928: Sensorless methods to drive BLDC motors

1 Introduction

This application note describes how to implement a sensorless commutation of BLDC motors with the ATAVRMC320 development kit.

The ATmega32M1 is equipped with integrated peripherals that reduce the number of external components required in a BLDC application. The ATmega32M1 is suitable for sensorless commutation and for commutation with Hall sensors as well, but this application note focuses on the sensorless commutation.

The AVR928 Application Note describes the theory of the sensorless control method and must be carefully read first.

8-bit **AVR**[®]
Microcontrollers

Application Note

Rev. 8306B-AVR-05/10





2 Hardware

The hardware includes the ATAVRMC310 and ATAVRMC300 boards which are the two parts of the ATAVRMC320 Starter kit.

Please refer to the ATAVRMC300 and ATAVRMC310 user guides :

- AVR430: MC300 Hardware User Guide
- AVR470: MC310 Hardware User Guide

2.1 MC310 jumpers setting

The AVR172 firmware has been developed with the following jumper settings:

Table 2-1.ATAVRMC310 jumpers setting for sensorless control

Designator	Setting	Function
J5	Vm	connect PB4 to Vm' (motor voltage measurement if necessary)
J6	PFC OC	Connect to overcurrent signal
J7	none	used by CAN applications
J8	ShCo	connect PC5 to ShCo for current measurement
J9	GNDm	connect PC4 to GNDm for current measurement
J12	TxD	connect PD3 to the RS232 driver
	MOSI A	Connect PD3 to ISP connector (for ISP use)
	RxDUSB	Connect PD3 to RxD1 (for USB interface use)
J13	RxD	connect PD4 to the RS232 driver
	SCK	Connect PD3 to ISP connector (for ISP use)
	TxDUSB	Connect PD3 to RxD1 (for USB interface use)
J15	none	used by CAN application to add a termination resistor
J21	Cmp-	connect ACMP0- to V+W bemf conditioning
J22	Cmp+	connect ACMP0+ to U bemf conditioning
J23	Cmp-	connect ACMP1- to U+W bemf conditioning
J24	Cmp+	connect ACMP1+ to V bemf conditioning
J25	Cmp-	connect ACMP2- to U+V bemf conditioning
J26	Cmp+	connect ACMP2+ to W bemf conditioning
J28	VCC	supply the on board USB dongle from the board power supply

See also following picture of MC310 Jumpers configurations :



Line to Line Resistance : 1.8 ohm = **R**

Back EMF : 3.66 V/Krpm = **k_e**

Peak current : 5.4A

As $V_m=12V$, the rated speed will be 2000 rpm.

2.5 ATmega32M1 Configuration

ATmega32M1 must be programmed to run at 16MHz using PLL (set corresponding Fuse bits).

The CKDIV8 fuse must be disabled.

Extended/High/Low Fuses configurations are : FF/DF/F3

2.6 Technical Advices

2.6.1 Disconnecting the BLDC Motor

The BLDC motor must not be disconnected while it is running or while its coils carry current. It is allowed to disconnect a BLDC motor if the PWM duty cycle is 0% and the rotor is at rest so that no current is driven through the coils. Be careful, when stopping the power supply or PWM, a BLDC motor with a high moment of inertia is able to run for a relatively long time.

2.6.2 Ground and Power Wirings

One design its own board has to take care of the ground wiring and power wiring. The power supply of the processor and additional signal conditioning components (e.g. additional fast comparators, operational amplifiers, ...) has to be decoupled from the motor power supply. The ground connection has to be of low resistance and low inductance to prevent against voltage drop and noise due to high currents. A ground plane within a multi layer PCB is recommended for proper operation.

3 Firmware

The example firmware is based on the Sensorless method described in AVR928 Application Note.

It is operating in sensorless mode using the ATmega32M1 internal comparators. Hall sensor wires of the BLDC motor of the kit can remain unconnected.

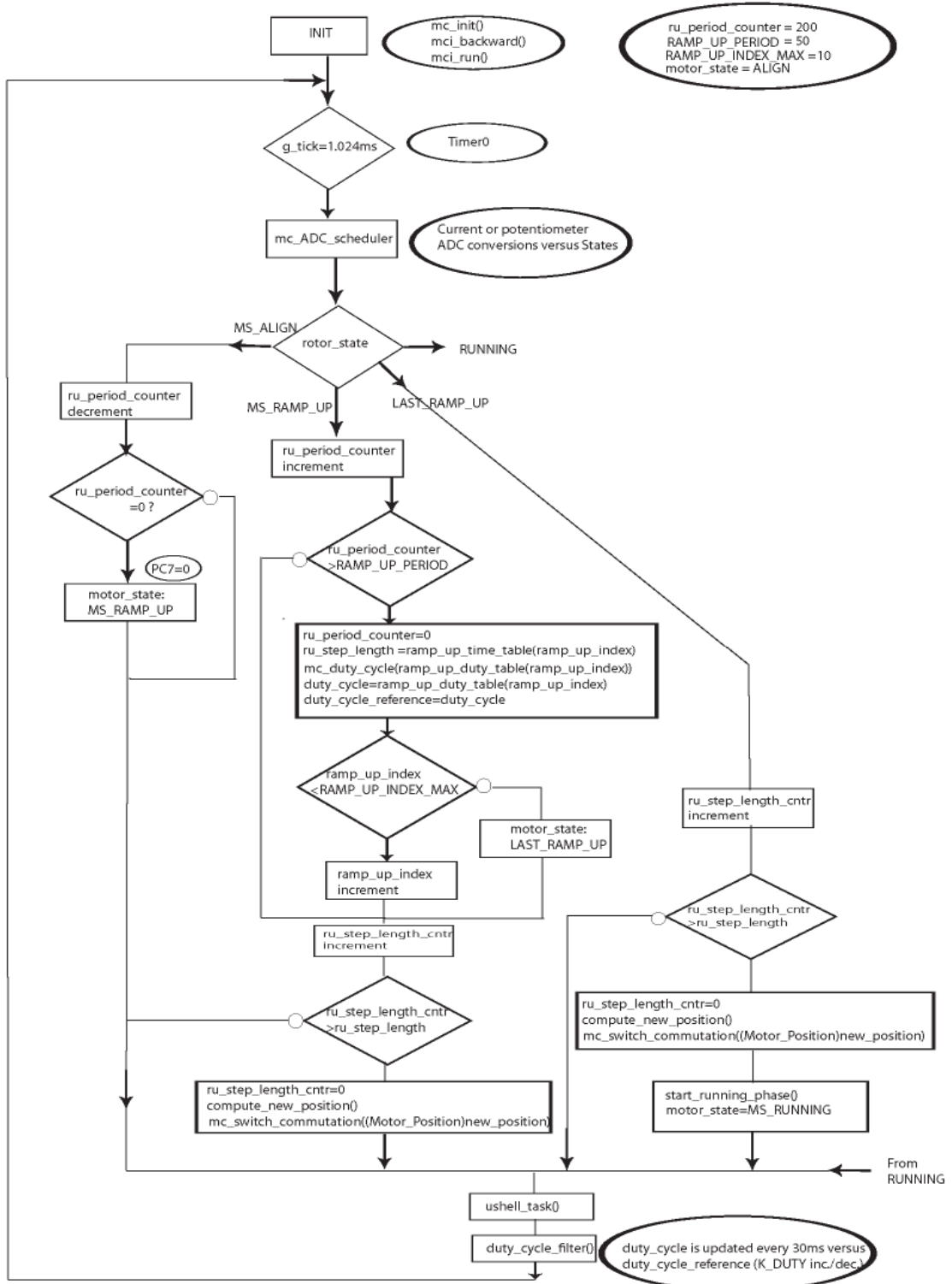
The source file directory embeds an html documentation which can be opened through the readme.html file.

The theory of the different tasks has been detailed in AVR928. The application to ATmega32M1 is detailed in following sections.

3.1 Main Flow chart

The firmware main flowchart is described below :

Figure 2. Main flow chart



The tasks are scheduled thanks to the g_tick produced each 1.024ms with Timer0.



3.2 MS_ALIGN phase

The ALIGN phase forces the motor at a specific position. The time of this phase is controlled with ALIGN_TIME constant which is the ru_period_counter initial value (200 for MC310 motor).

3.3 RAMP_UP phase

The ramp-up characteristics (duty-cycles and times) are stored in two tables:

- ramp_up_duty_table[] : which provides the duty_cycle of the step
- ramp_up_time_table[] : which provides the length of the step (ru_step_length)

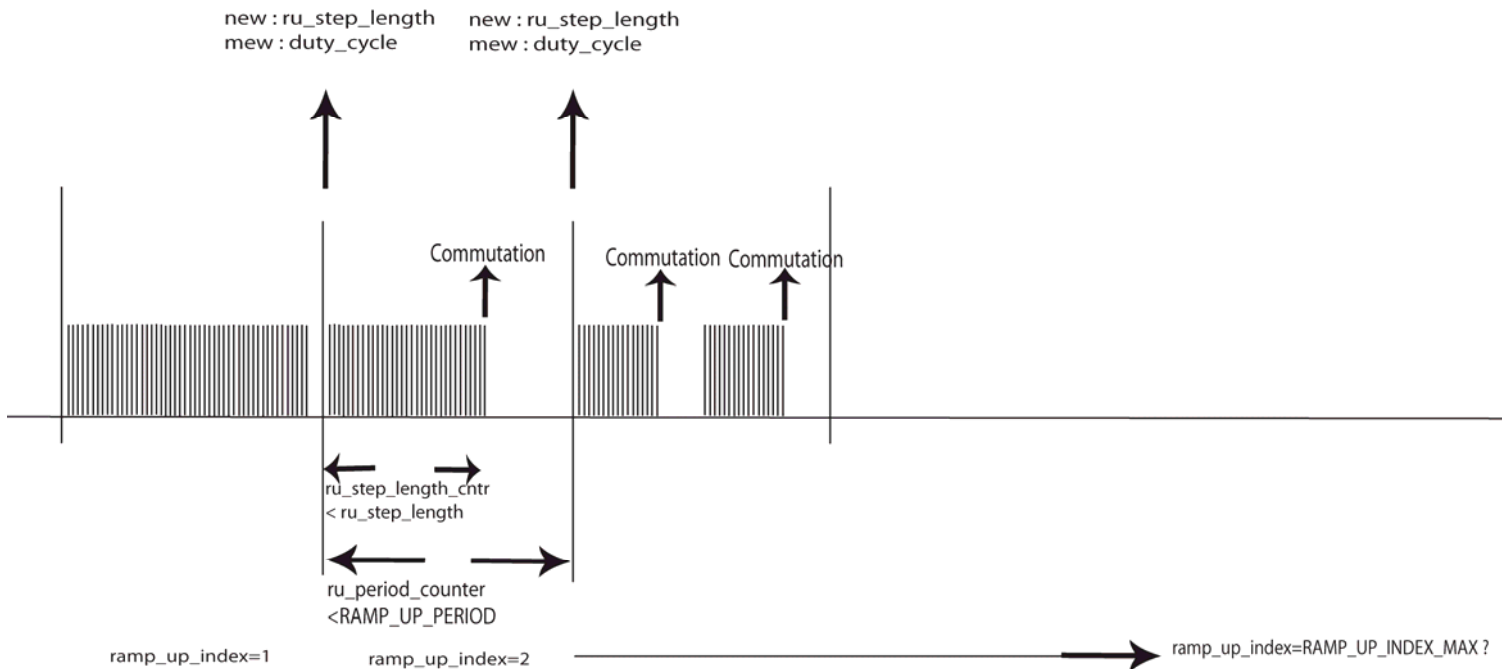
These two tables are specific to the motor and the application.

The scanning of the step sequences and the monitoring of the step length are achieved thanks to three independant counters :

- ru_step_length_cntr : which counts the commutation time (up to ru_step_length variable)
- ru_period_counter : which counts the step length (up to RAMP_UP_PERIOD constant)
- ramp_up_index : which counts the step numbers (up to RAMP_UP_INDEX_MAX constant)

The figure below provides a waveform of steps timing :

Figure 3. Steps timing



3.3.1 Time of steps

The step time is $RAMP_UP_PERIOD = 50ms$.

3.3.2 Number of steps

The parameter : $RAMP_UP_INDEX_MAX = 9$, defines 10 steps ramp up.

3.3.3 Parameters tables

In firmware example, the tables have been defined according to the characteristics of the motor provided in the kit (see parameters in 2.4 Motor section) :

```
ramp_up_time_table[] = {26,23,20,17,14,11,8,5,3,2,2};
```

```
ramp_up_duty_table[] = {122,124,126,129,131,133,135,137,140,143,145};
```

3.3.4 Sp1/pwm1

The usual parameters described in AVR928 Application Note are:

- $Pwm1 = 50\%$
- $Sp1 = Sp_max/60$

The parameters defined with MC310 Tecmotion motor are:

- $Pwm1 = 48\% (= 122/256)$
- $Sp1 :$

$Sp1$ is defined thanks to the initialization value of ru_step_length :

```
ru_step_length = RAMP_UP_STEP_MAX = 40
```

This variable determines one commutation each 40ms.

So an electrical rotation time is 120ms. As the motor has 4 pairs of poles, the mechanical rotation time is 480ms. So the rotation speed is $60/0.48 = 125$ rpm.

So $Sp1 = Sp_max/32$.

The second value of ru_step_length is 26 in the time table. It defines the following commutation time.

3.3.5 Sp2/pwm2

The theoretical parameters described in AVR928 Application Note are:

- $Pwm2 = 60\%$
- $Sp2 = Sp_max/6 = Sp1 / 10$

The parameters defined with Tecmotion motor are:

- $Pwm2 = 57\% (= 145/256)$
- $Sp2 :$

$Sp2$ is defined thanks to the last value of ru_step_length : 2

This variable determines one commutation each 4ms.

So an electrical rotation time is 12ms. As the motor has 4 pairs of poles, the mechanical rotation time is 48ms. So the rotation speed is $60/0.048 = 1250$ rpm.

So $Sp2 = Sp_max/3.2$.



This confirms also the usual ratio = 10 between Sp1 and Sp2 which is defined in AVR498 Application Note.

3.4 LAST_RAMP_UP phase

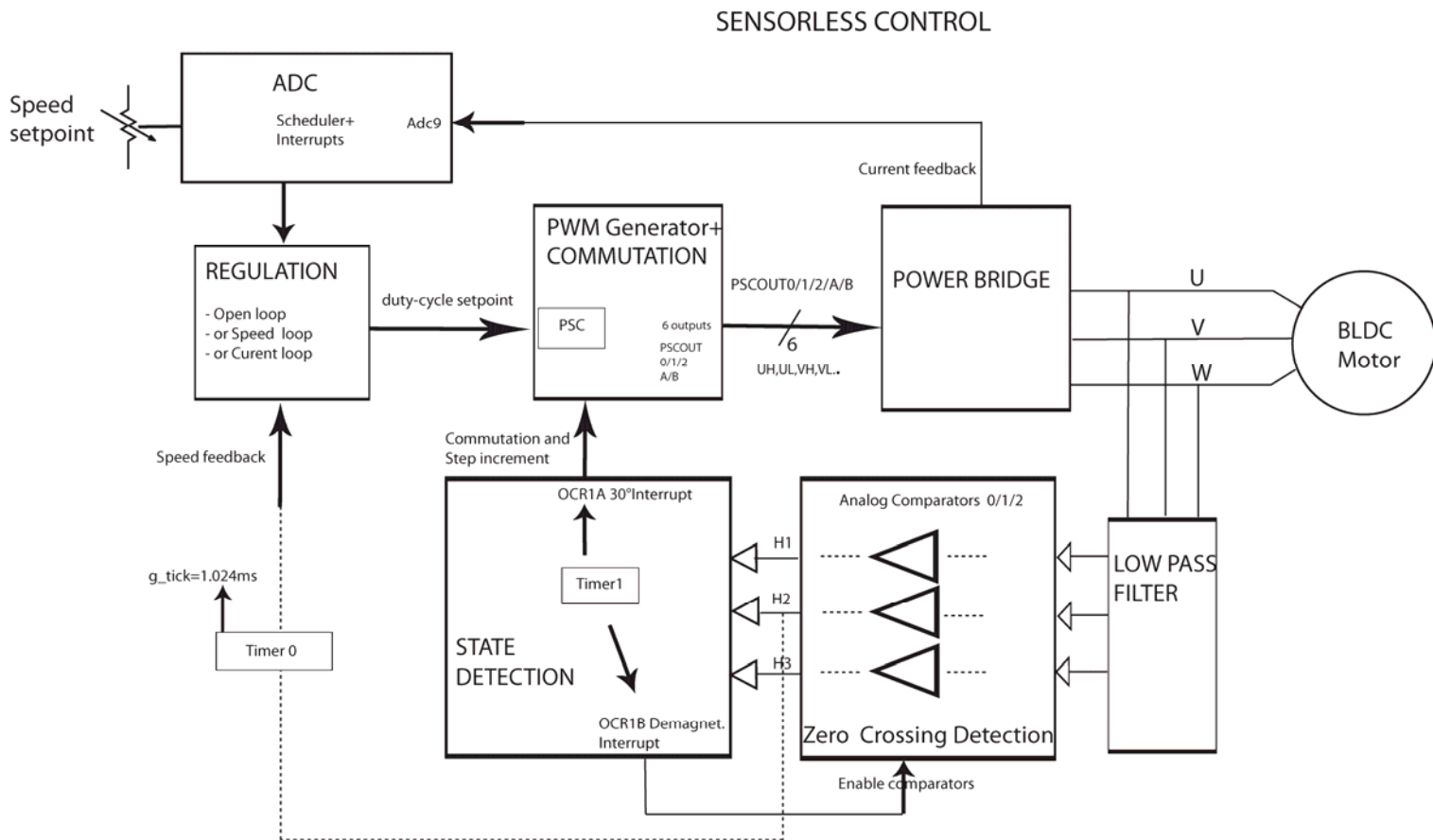
To avoid a shorten last step, this phase monitors the last ramp-up step to guarantee it is ended properly before running in closed loop.

3.5 RUNNING Phase

3.5.1 Closed-loop block diagram

The Running phase is a sensorless closed loop which block diagram is following :

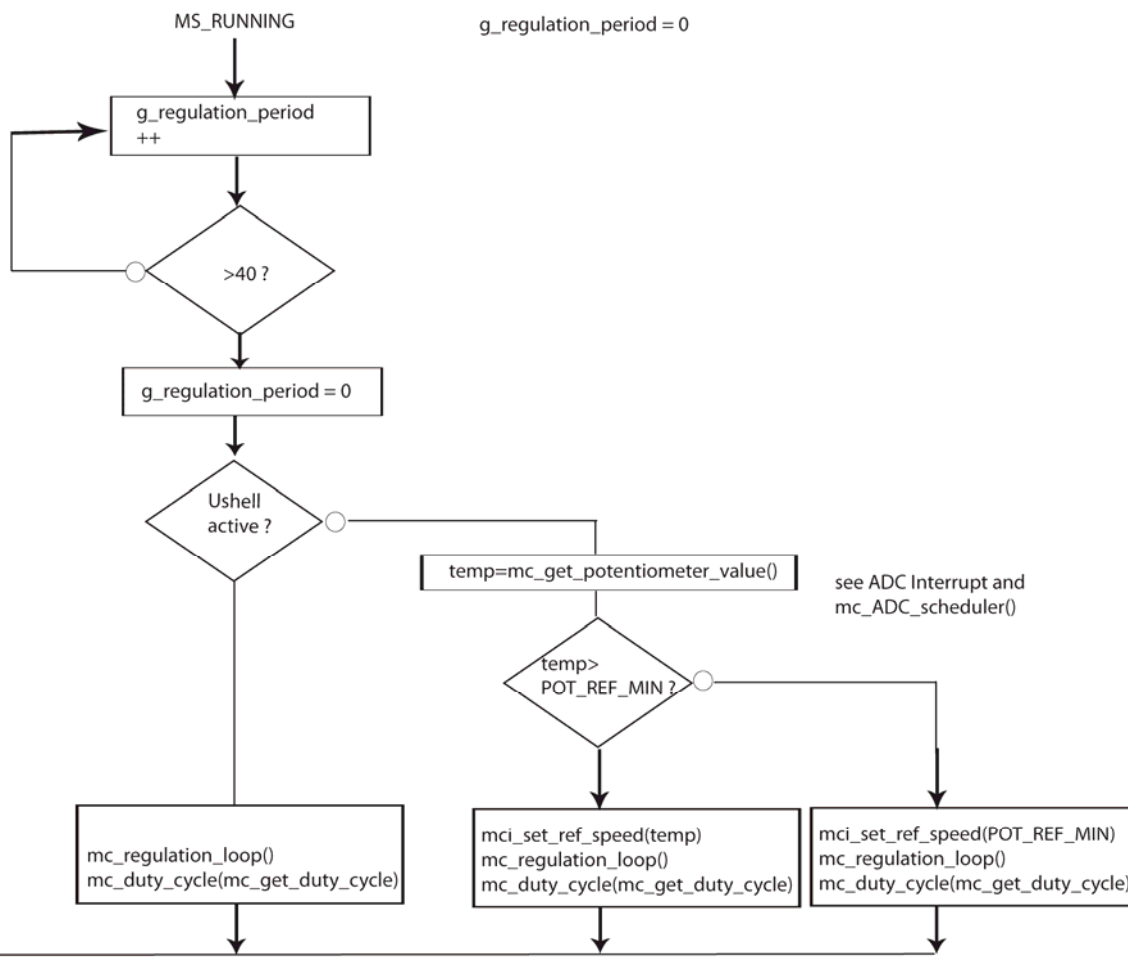
Figure 4. Closed-loop block diagram



3.5.2 Running flowchart

The flowchart is following :

Figure 5. Closed-loop flowchart



Motor_state is kept equal to MS_RUNNING

mci_set_ref_speed() function updates the speed setpoint according to the potentiometer adjustment or the speed command received on serial transmission.

In *mc_regulation_loop()* function, *duty_cycle_reference* is the *duty_cycle* variable which controls the PWM generator. This variable is the result of following functions :

- In OPEN_LOOP:
mci_set_ref_speed() function
- In SPEED_LOOP:





```
mc_control_speed(2*mcj_get_ref_speed())
```

duty-cycle_reference is calculated from *ref_speed* and from monitored *mcj_get_measured_speed()*

```
measured_speed = (KSPEED * 4) / mcj_measured_period
```

with *mcj_measured_period* calculated in the Interrupt vector of Analog Comparator 1. This interrupt uses Timer 0 to compute the period.

- In CURRENT_LOOP :

```
mc_control_current(mc_get_potentiometer_value())
```

3.5.3 Sensorless Detection and Commutation Management

The analog comparators 0, 1 and 2 are used to detect the zero crossing of the U, V and W phases.

The timer 1 is used to monitor the time between two consecutive zero crossings. This time corresponds to one sector of the electrical rotation of the motor. It equals 60° of the entire electrical period of the motor.

When a zero crossing event occurs, the timer 1 value is stored. Then this value is divided by 2 (providing the 30° time) and loaded into the Compare A register of timer 1. Then this value is added to the half of itself to provide the 45° time and loaded into the Compare B register of timer 1.

The timer 1 compare A event occurs 30° after the zero crossing. It activates the next commutation state and masks the zero crossing to avoid the discharge of the inductance (demagnetization) pulse generated at the end of a step when the active switches are released.

Due to the inductance of the motor coils, a voltage equals to $-Ldi/dt$ is generated, the demagnetization is done through the diodes of the power bridge.

The timer 1 compare B event releases the zero crossing mask : enables the comparator n interrupt according to the *motor_step* variable. This Timer1 interrupt provides the demagnetization mask delay.

4 RS232 Communication with firmware

4.1 Connecting ATAVRMC310 to use the RS232 interface

Connect PC com port to the ATAVRMC310 RS232 connector through a direct cable.

The serial configuration is:

- 38400 bauds,
- 8 bit data bit,
- 1 stop bit,
- no handshake,

4.2 PC applications

User can communicate with firmware through RS232 with usual PC serial communication applications (i.e. Hyperterminal) or the Atmel "Motor Control Center" application which can be downloaded from Atmel web at url : <http://www.atmel.com>

4.2.1 PC Terminal : RS232 Messages and Commands

At power up the following welcome message is received on terminal :
"ATMEL Motor Control Interface".

The following commands can be sent to the firmware:

Table 2-1. List of commands

Command	Action
ru	Run motor
st	Stop Motor
help	Gives help
fw	Set direction to Forward
bw	Set direction to Backward
ss	Set Speed (followed with speed value)
gi	Get ID
g0	Get Status 0
g1	Get Status 1

4.2.2 Motor Control Center

The User Guide is available in Install directory at URL :
C:\Program Files\Atmel\Motor Control Center\help\Overview.htm

The AVR172 Target must be selected first to get the right configuration :


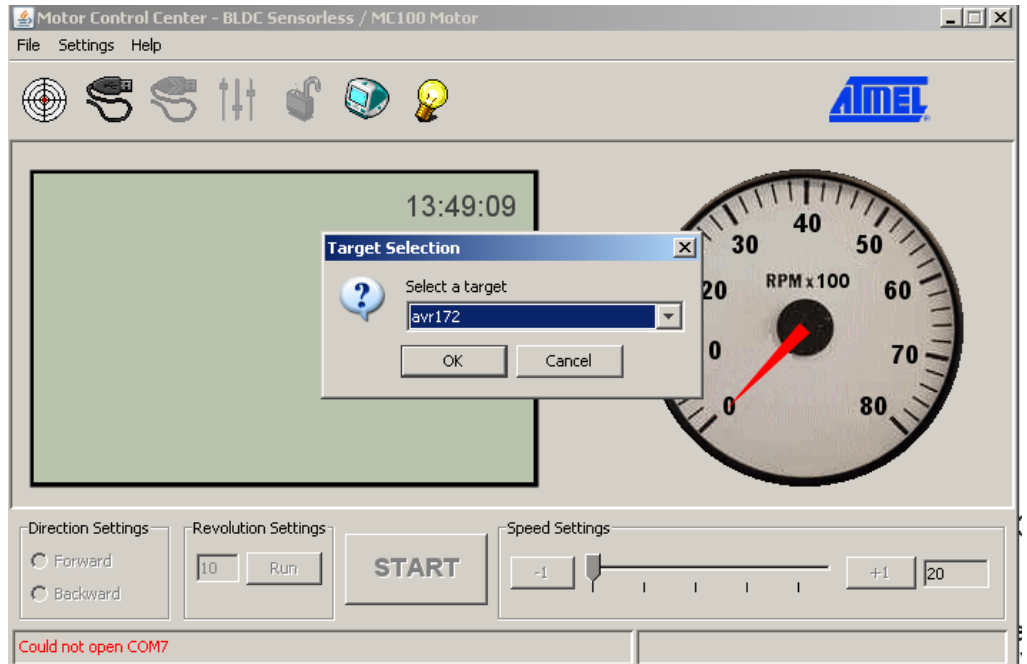
To select a target, execute the **File > Select Target** command or click the  button in the toolbar. The following dialog pops up:





Figure 6. Motor Control Center Interface



5 USB communication

Communication can be achieved from PC to USB connector of MC310 board.

The AVR470, MC310 Hardware User Guide details the configuration to be achieved.

Communication port becomes a Virtual Com port. Same tools as described in section 4 (RS232 Communication with firmware), can be used through this Virtual Com port.



Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
<http://www.atmel.com/>

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

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 ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, 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 OF 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 product 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's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2010 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, AVR®, AVR® logo and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.