AVR492: Brushless DC Motor Control using AT90PWM3

AVR Microcontrollers

Application Note

Features

- **BLDC Motor Basics**
- **PSC use**
- **Hardware Implementation**
- **Code Example**

Introduction

This application note describes how to implement a brushless DC motor control in sensor mode using AT90PWM3 AVR microcontroller.

The high performance of AVR core fitted with Power Stage Controller module of AT90PWM3 enable the design of high speed brushless DC motor applications.

In this document, we give a short description of brushless DC motor theory of operation, we detail how to control brushless DC motor in sensor mode and also give the hardware description of ATAVRMC100 reference design used in this application note.

Software implementation is also discussed with software control loop using PID filter.

This application note deals only with BLDC motor control application using Hall effect position sensors to control commutation sequence.

7518A–AVR–07/05

Theory of Operation

Brushless DC motors are used in a growing number of motor applications as they have many advantages:

1. They have no brushes so they required little or no maintenance.
2. They generate less acoustic and electrical noise than universal brushed DC motors.
3. They can be use in hazardous operation environment (with flammable products).
4. They have a good weight/size to power ratio...

Such types of motor have a little rotor inertia, the coils are attached to the stator. The commutation is controlled by electronics. Commutation times are provided either by position sensors or by coils Back Electromotive Force measuring.

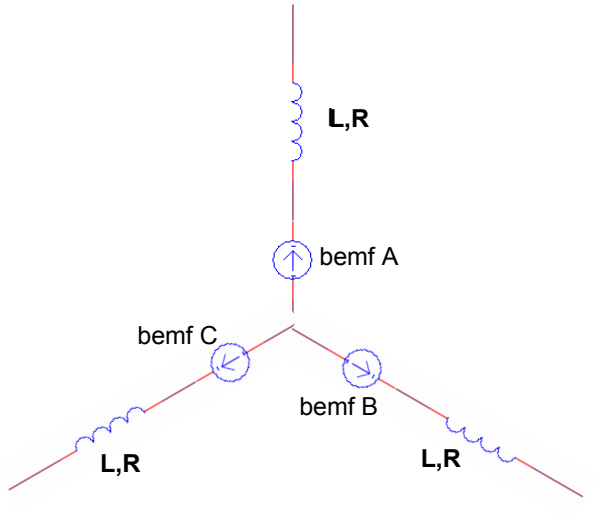
In sensor mode, Brushless DC motor usually consists of three main parts. A Stator, a Rotor and Hall Sensors.

Stator

A basic three phases BLDC motor Stator has three coils. In many motors the number of coils is replicated to have a smaller torque ripple.

Figure 1 shows the electrical schematic of the stator. It consists of three coils each including three elements in series, an inductance, a resistance and one back electromotive force.

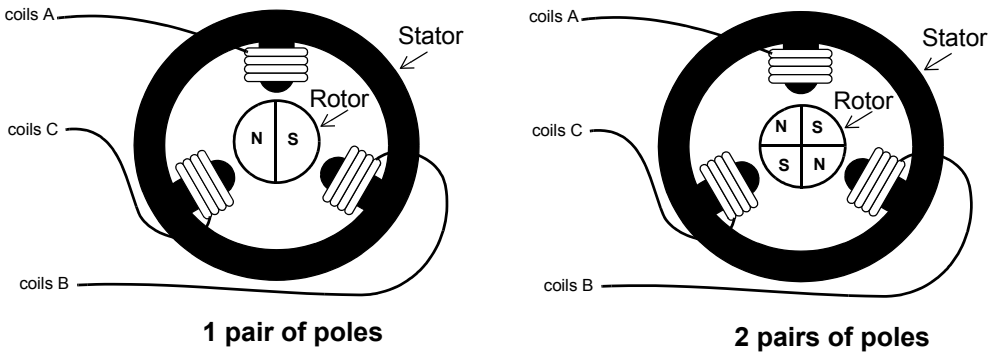
Figure 1. Stator Electrical Configuration (Three phases, three coils)



Rotor

The rotor in a BLDC motor consists of an even number of permanent magnets. The number of magnetic poles in the rotor also affects the step size and torque ripple of the motor. More poles give smaller steps and less torque ripple. The permanent magnets go from 1 to 5 pairs of poles. In certain cases it can goes up to 8 pairs of poles.(Figure 2).

Figure 2. Three phases, three coil BLDC motor stator and rotor



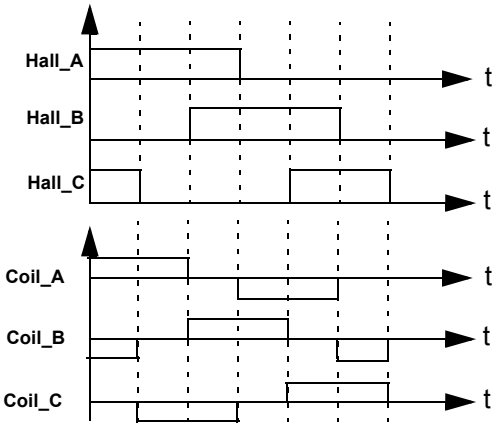
The coils are stationary while the magnet is rotating. The rotor of the BLDC motor is lighter than the rotor of a conventional universal DC motor where the coils are placed on the rotor.

Hall Sensor

For estimating the rotor position, three hall sensors are positioned. These hall sensors are separated by 120° each. With these sensors, 6 different commutations are possible. Phase commutation depends on hall sensor values.

Power supply to the coils change when hall sensor values change. With right synchronized commutations, the torque remain nearly constant and high.

Figure 3. Hall Sensors signals for CW rotation

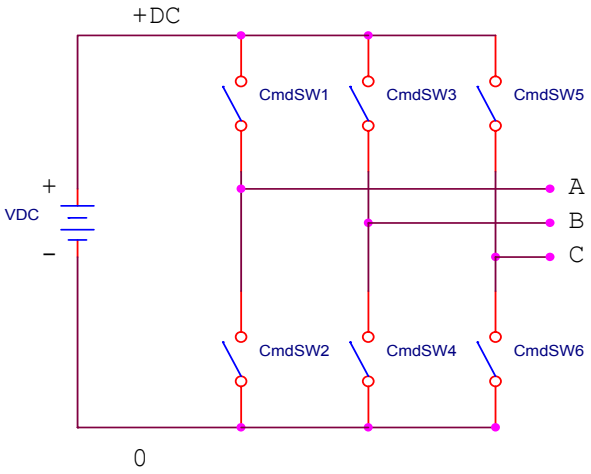


Phases Commutation

To simplify the explanation of how to operate a three phases BLDC motor a typical BLDC motor with only three coils is considered. As previously shown, phases commutation depends on the hall sensor values. When motor coils are correctly supplied, a magnetic field is created and rotor moves. The most elementary commutation driving method used for BLDC motors is an on-off scheme: a coil is either conducting or not conducting. Only two windings are supplied at the same time the third winding is floating. Connecting the coils to the power and neutral bus induces the current flow. This is referred to as trapezoidal commutation or block commutation.

To command brushless DC motors, a power stage made of 3 half bridges is used. Figure 4 below shows 3 half bridges schematic.

Figure 4. Power Stage



Reading hall sensor value, indicate which switch should be closed.

Table 1. Switches commutation for CW rotation

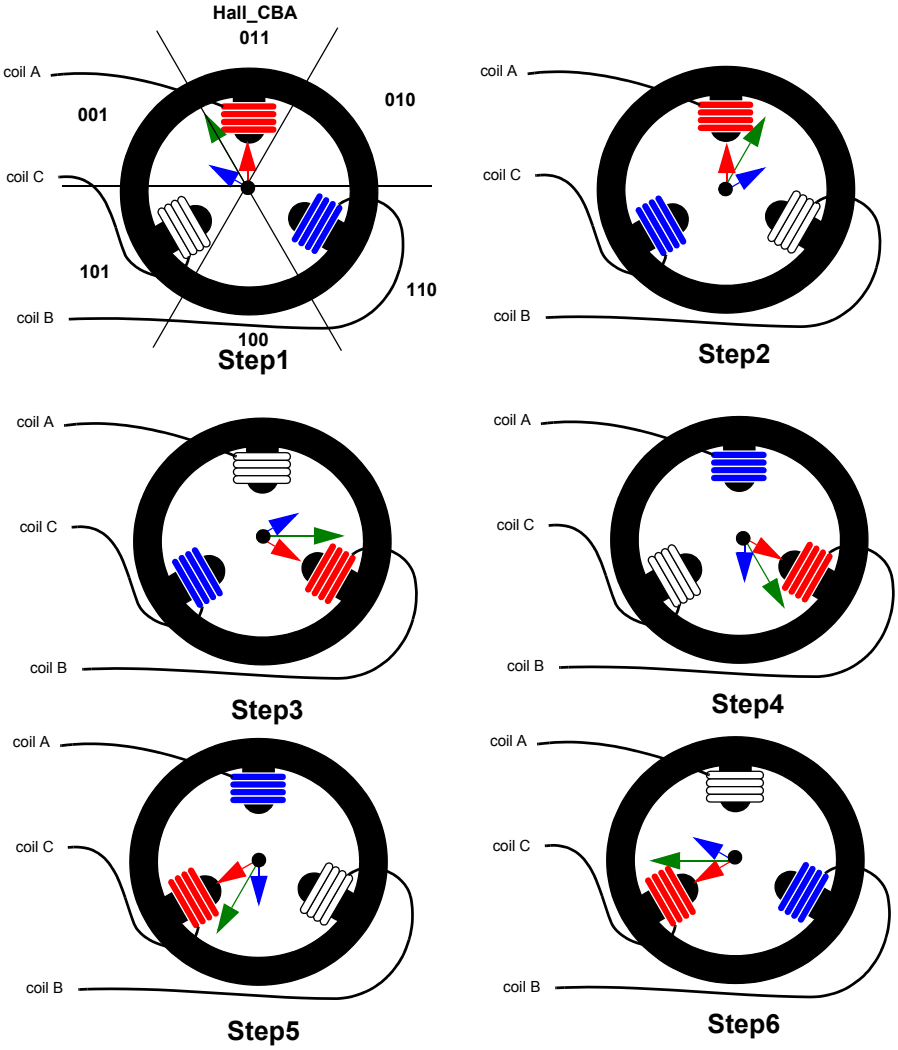
Hall Sensors Value (Hall_CBA)	Phase	Switches
101	A-B	SW1 ; SW4
001	A-C	SW1 ; SW6
011	B-C	SW3 ; SW6
010	B-A	SW3 ; SW2
110	C-A	SW5 ; SW2
100	C-B	SW1 ; SW4

For motors with multiple poles the electrical rotation does not correspond to a mechanical rotation. A four pole BLDC motor uses four electrical rotation cycles to have one mechanical rotation.

The strength of the magnetic field determines the force and speed of the motor. By varying the current flow through the coils the speed and torque of the motor can be adjusted. The most common way to control the current flow is to control the average current flow through the coils. PWM(Pulse Width Modulation) is used to adjust the average voltage and thereby the average current, including the speed. The speed can be adjusted with a high frequency from 20 khz to 60 khz.

For a three phase, three coils BLDC motor, the rotating field is described in Figure 5.

Figure 5. Commutation steps and rotating field



Commutation creates a rotating field. Step1 Phase A is connected to the positive DC bus voltage by SW1 and Phase B is connected to ground by SW4, Phase C is unpowered. Two flux vectors are generated by phase A (red arrow) and phase B (blue arrow) the sum of the two vectors give the stator flux vector (green arrow). Then the rotor tries to follow stator flux. As soon as the rotor reach a certain position when the hall sensors state changes its value from “010” to “011” a new voltage pattern is selected and applied to the BLDC motor. Then Phase B is unpowered and Phase C is connected to the ground. It results in a new stator flux vector ‘Step2’.

By following the commutation schematic Figure 3 and Table 1, we obtain six differents stator flux vectors corresponding to the six commutation steps. The six steps provide one rotor revolution.

Starter KIT ATAVRMC100

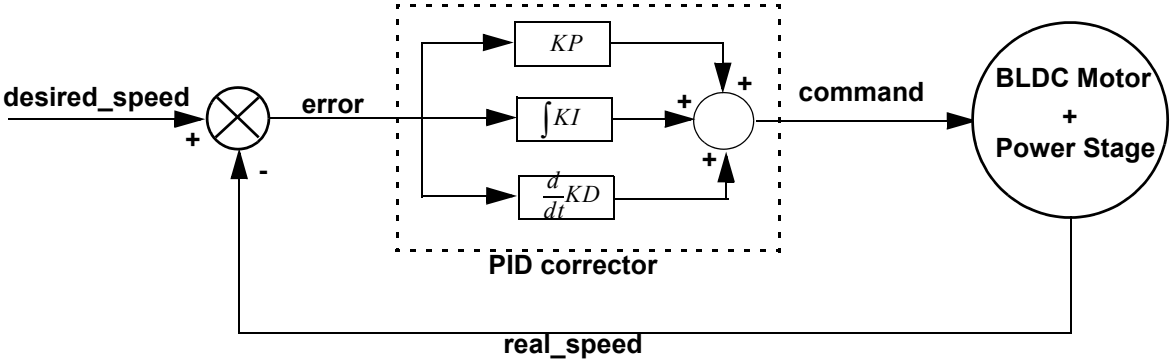
After this brief theoretical presentation of Brushless DC Motor Control, the practical implementation will be introduced with the help of an example. The next part of this application note will deal with the hardware and the software implementation based on the starter kit ATAVRMC100 running with a microcontroller AT90PWM3.

Electrical schematics are available Figure 21, Figure 22, Figure 23 and Figure 24 at the end of the document. They describe the reference design for the Starter Kit ATAVRMC100.

The software includes the control of the speed through a PID corrector. Such a corrector is composed of three main coefficients : KP, KI, and KD.

KP is the proportionnal gain coefficient, KI is the integral gain coefficient and KD is the derivative gain coefficient. The error between the desired speed and the real speed (called error in the Figure 6) is multiplied by each gain. Then, the sum of the three thermes give the command to apply to the motor to get the right speed (Figure 6).

Figure 6. PID diagram



$$command(t) = KP \times eror(t) + KI\int error(t)d(t) + KD\frac{d}{dt}error(t)$$

$$error(t) = desired\_speed(t) - \text{real}\_speed(t)$$

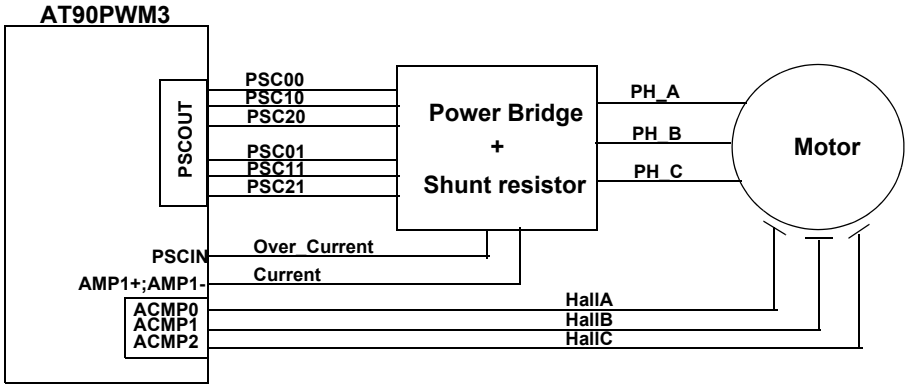
The KP coefficient fixes the motor response time, the KI coefficient is used to cancel the static error and the KD is used in particular for position regulation(refer to regulation loop in software description for tuning the coefficients).

Hardware Description

As shown in Figure 7 the microcontroller contains 3 Power Stage Controller (PSC). Each PSC can be seen as a Pulse Width Modulator with two output signals. To avoid cross conduction a Dead Time control is integrated (see AT90PWM3 datasheet for more information about PSC or Figure 9 below).

A fault input (Over_Current) is linked to PSCIN. This fault input enables the microcontroller to disable all PSC outputs.

Figure 7. Hardware implementation



PSCij : Power Stage Controller i (i = 0,1,2) output j (j = 0,1)
ACMPi : Analog Comparator Positive input (i = 0,1,2)
AMPi+/- : Analog Differential Amplified channel Positive/Negative inputs (i = 0,1,2)

It's possible to measure the current with two differential amplified channels with programmable 5, 10, 20 and 40 gain stage. The Shunt resistor has to be adjusted to the amplifier conversion range.

The Over_Current signal results from an external comparator. The comparator's reference can be adjusted by the internal DAC.

The phase commutation has to be done according to hall sensors value. HallA, HallB and HallC are connected to external interrupt sources or to the three internal comparators. The comparators generate the same type of interrupts as the external interrupts. Figure 8 shows how the microcontroller I/O ports are used in the starter kit.

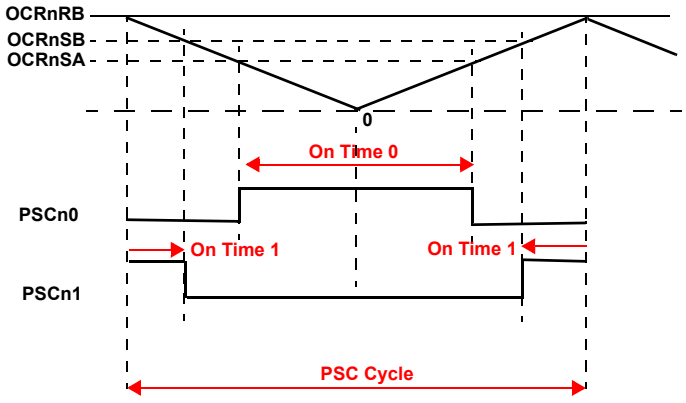
Figure 8. Microcontroller I/O Ports use (SO32 package)

			Operating Mode
PORTB	PIN	Port Function	SENSOR
PB0/MISO/PSCOUT20	12	H_C	H_C
PB1/MOSI/PSCOUT21	13	L_C	L_C
PB2/ADC5/INT1	20	VMOT	VMOT
PB3/AMP0-	27	*EXT1*	Depend on the communication mode
PB4/AMP0+	28	*EXT2*	
PB5/ADC6/INT2	30	*EXT5*	
PB6/ADC7/PSCOUT11/ICP1B	31	L_B	L_B
PB7/ADC4/PSCOUT01/SCK	32	L_A	L_A
PORTC			
PC0/INT3/PSCOUT10	2	H_B	H_B
PC1/PSCIN1/OC1B	7	*EXT3*	Depend on the communication mode
PC2/T0/PSCOUT22	10	*EXT4*	
PC3/T1/PSCOUT23	11	LIN_NSLP	
PC4/ADC8/AMP1-	21	V_Shunt-	V_Shunt-
PC5/ADC9/AMP1+	22	V_Shunt+	V_Shunt+
PC6/ADC10/ACMP1	26	HallB / BEMF_B	HallB
PC7/D2A	29	DAC_OUT	DAC_OUT
PORTD			
PD0/PSCOUT00/XCK/SS_A	1	H_A	H_A
PD1/PSCIN0/CLKO	4	Over_Current	Over_Current
PD2/PSCIN2/OC1A/MISO_A	5	MISO / *EXT10*	Depend on the communication mode
PD3/TXD/DALI/OC0A/SS/MOSI_A	6	MOSI / LIN TxD / TxD / *EXT7*	
PD4/ADC1/RXD/DALI/ICP1A/SCK_A	16	SCK / LIN RxD / RxD / POT / *EXT8*	
PD5/ADC2/ACMP2	17	HallC / BEMF_C	HallC
PD6ADC3/ACMPM/INT0	18	VMOT_Half	VMOT_Half
PD7/ACMP0	19	HallA / BEMF_A	HallA
PORTE			
PE0/RESET/OCD	3	NRES / *EXT9*	Depend on the communication mode
PE1/OC0B/XTAL1	14	*EXT6*	
PE2/ADC0/XTAL2	15	LED	

VMOT and VMOT_Half are implemented but not used. They can be used to get the value of the motor supply.

The outputs H_x and L_x are used to control the power bridge. As previously seen, they depend on the Power Stage Control (PSC) which generate PWM signals. For such application the recommended mode is the Center Aligned Mode (see Figure 9), the register OCR0RA is used to adjust ADC synchronization for current measurement.

Figure 9. PSCn0 & PSCn1 Basic Waveforms in Center Aligned Mode



- **On Time 0 = 2 * OCRnSA * 1/Fclkpsc**
- **On Time 1 = 2* (OCRnRB - OCRnSB + 1) * 1/Fclkpsc**
- **PSC Cycle = 2 * (OCRnRB + 1) * 1/Fclkpsc**

The dead time value between PSCn0 and PSCn1 is :

- **|OCRnSB - OCRnSA| * 1/Fclkpsc**

The input clock of PSC is given by CLKPSC. Clock input from I/O clock or from PLL.

Two strategies can be employed for the PWM signals applied on the power stage. The first consists in applying the PWM signals on the high side AND the low side of the power bridge and the second in applying the PWM signals only on the high side of the power bridge.

Software Description

Atmel provides libraries to control Brushless DC motors. The first step is to configure and initialize the microcontroller.

Microcontroller Configuration and Initialization

The function to be used is named mc_init_motor(). It calls hardware and software initialization functions and initialize all motor parameters (motor direction, motor speed and stop the motor).

Software Implentation Diagram

After microcontroller configuration and initialization, the motor can be started. Only few functions are needed to control the motor. All user functions are define in mc_lib.h:

```
void mc_motor_run(void)
```

- Used to start the motor. The regulation loop function is launched to set the duty cycle. Then the first phases commutation is executed.

```
Bool mc_motor_is_running(void)
```

- Get the motor state. If ‘TRUE’ the motor is running. Else if ‘FALSE’ the motor is stopped.

```
void mc_motor_stop(void)
```

- Used to stop the motor.

```
void mc_set_motor_speed(U8 speed)
```

- Set the user requested speed.

```
U8 mc_get_motor_speed(void)
```

- Return the current user requested speed.

```
void mc_set_motor_direction(U8 direction)
```

- Set the motor direction, ‘CW’ or ‘CCW’.

```
U8 mc_get_motor_direction(void)
```

- Return the current direction of rotation of the motor.

```
U8 mc_set_motor_measured_speed(U8 measured_speed)
```

- Save the measured speed in the measured_speed variable.

```
U8 mc_get_motor_measured_speed(void)
```

- Get the measured speed.

```
void mc_set_Close_Loop(void)
void mc_set_Open_Loop(void)
```

- Set type of regulation Open Loop or Close Loop. See Figure 13 for more information.

Figure 10. AT90PWM3 Configuration

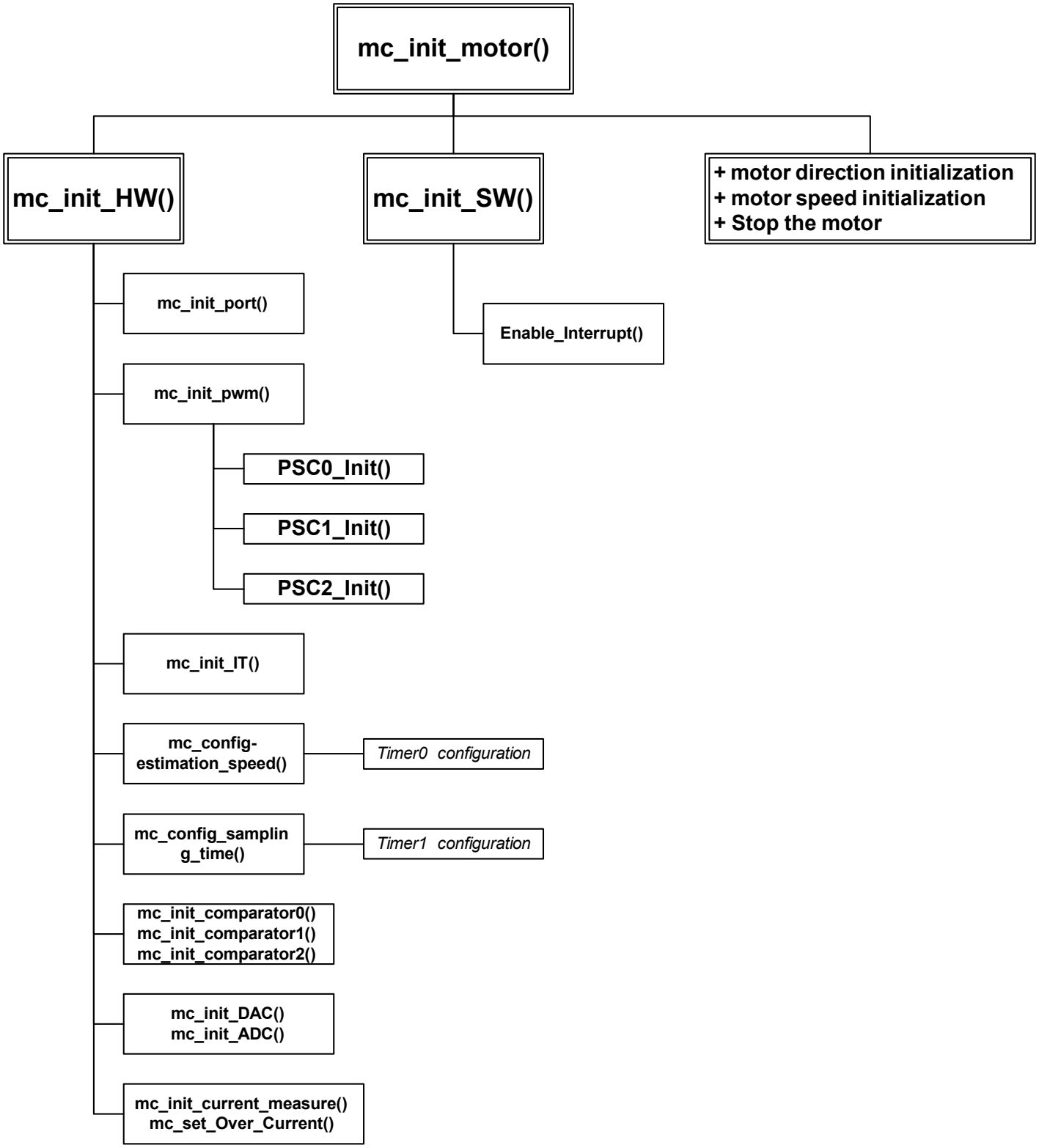


Figure 11. Software Implentation Diagram.

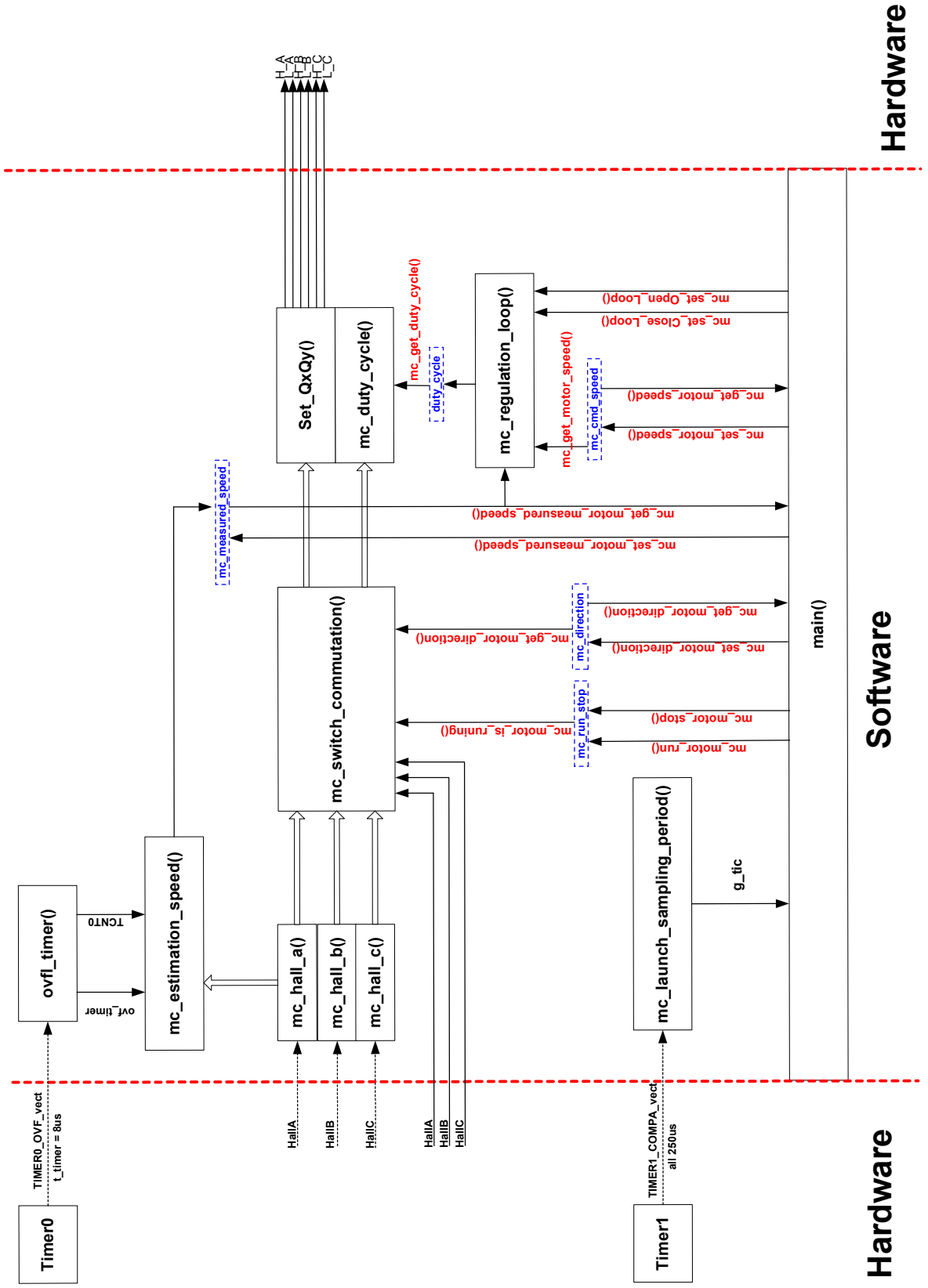
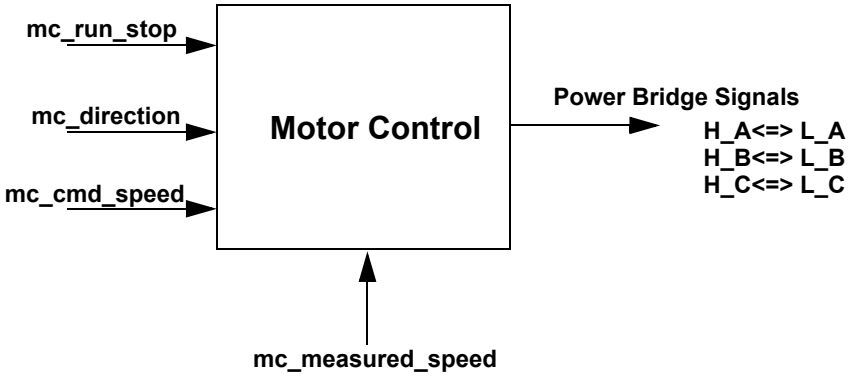


Figure 11 shows four variables mc_run_stop, mc_direction, mc_cmd_speed and mc_measured_speed. They are the main software variables which can be accessed by user functions previously described.

The software can be seen as a black box named Motor Control (Figure 12) with inputs (mc_run_stop, mc_direction, mc_cmd_speed, mc_measured_speed) and output (All power bridge signals).

Figure 12. Main Software Variables.



Most of the functions are available in mc_drv.h. Only a few depends on the motor type. Functions can be distributed in four main class :

- **Hardware initialization**

```
void mc_init_HW(void);
```

- All hardware initialization are made in this function. We can found ports initialization, interrupts initialization, timers and PSCs.

```
void mc_init_SW(void);
```

- Used to initialized software. Enable all interrupts.

```
void mc_init_port(void);
```

- Microcontroller I/O ports initialization with DDRx register initialization to choose if the pin acts as an Output or an Input and PORTx to activate or not the pull-up on Input.

```
void mc_init_pwm(void);
```

- This function start the PLL and sets all PSC registers with their initial values.

```
void mc_init_IT(void);
```

- Modifie this function to enable or disable types of interrupts.

```
void PSC0_Init ( unsigned int dt0,
                 unsigned int ot0,
                 unsigned int dt1,
                 unsigned int ot1);
void PSC1_Init ( unsigned int dt0,
                 unsigned int ot0,
                 unsigned int dt1,
                 unsigned int ot1);

void PSC2_Init (unsigned int dt0,
                   unsigned int ot0,
                   unsigned int dt1,
                   unsigned int ot1);
```

- PSCx_Init allows the user to select the configuration of the Power Stage Control (PSC) of the microcontroller.

- **Phases commutation functions**

```
U8 mc_get_hall(void);
```

- Get the Hall sensors value corresponding to the six commutations steps (HS_001, HS_010, HS_011, HS_100, HS_101, HS_110).

```
_interrupt void mc_hall_a(void);
_interrupt void mc_hall_b(void);
_interrupt void mc_hall_c(void);
```

- These functions are executed when an external interrupt is detected (Hall sensor edge). They allow phases commutation and speed calculation.

```
void mc_duty_cycle(U8 level);
```

- This function set the PSC duty cycle according to PSC configuration.

```
void mc_switch_commutation(U8 position);
```

- The phases commutation is done according to hall sensors value and only if the user start the motor.

- **Sampling time configuration**

```
void mc_config_sampling_period(void);
```

- Initialize the timer1 to generate an interrupt all 250 us.

```
_interrupt void launch_sampling_period(void);
```

- When the 250us interrupt is activated, a flag is set. It can be use for sampling time control.

- **Estimation speed**

```
void mc_config_time_estimation_speed(void);
```

- Configuration of the timer0 to be use for the speed calculation.

```
void mc_estimation_speed(void);
```

- This function calculate the motor speed with the hall sensor periode measure principle.

```
_interrupt void ovfl_timer(void);
```

- When an overflow occurs on the timer0 a 8bits variable is increased to obtain a 16bits timer with a 8bits timer.

- **Current measure**

```
_interrupt void ADC_EOC(void);
```

- As soon as the amplifier conversion is finished, the ADC_EOC function is executed to set a flag usable for the user.

```
void mc_init_current_measure(void);
```

- This function initializes the amplifier 1 for the current measure.

```
U8 mc_get_current(void);
```

- Get the current value if the conversion is finished.

```
Bool mc_conversion_is_finished(void);
```

- Indicate if the conversion is finished.

```
void mc_ack_EOC(void);
```

- Reset the End Of Conversion flag.

- **Over Current detection**

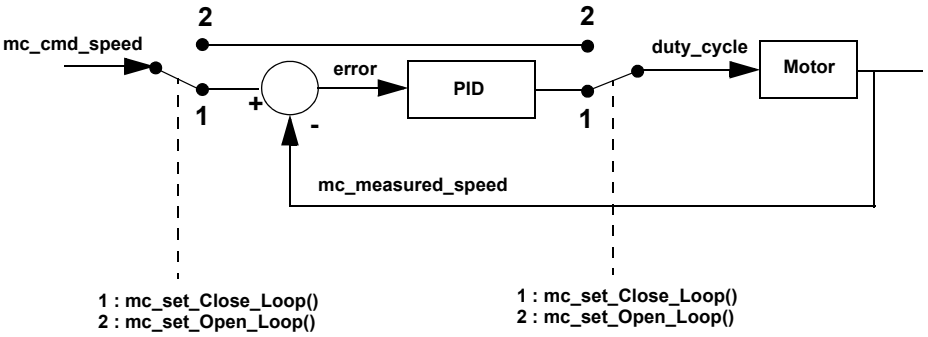
```
void mc_set_Over_Current(U8 Level);
```

- Set the level of detection of an over current. The level is output by the DAC on external comparator.

Regulation Loop

Two functions select the regulation loop. Open loop or close loop. Figure 13 shows the regulation loop implemented in the software.The two functions are mc_set_Close_Loop() and mc_set_Open_Loop().

Figure 13. Regulation Loop



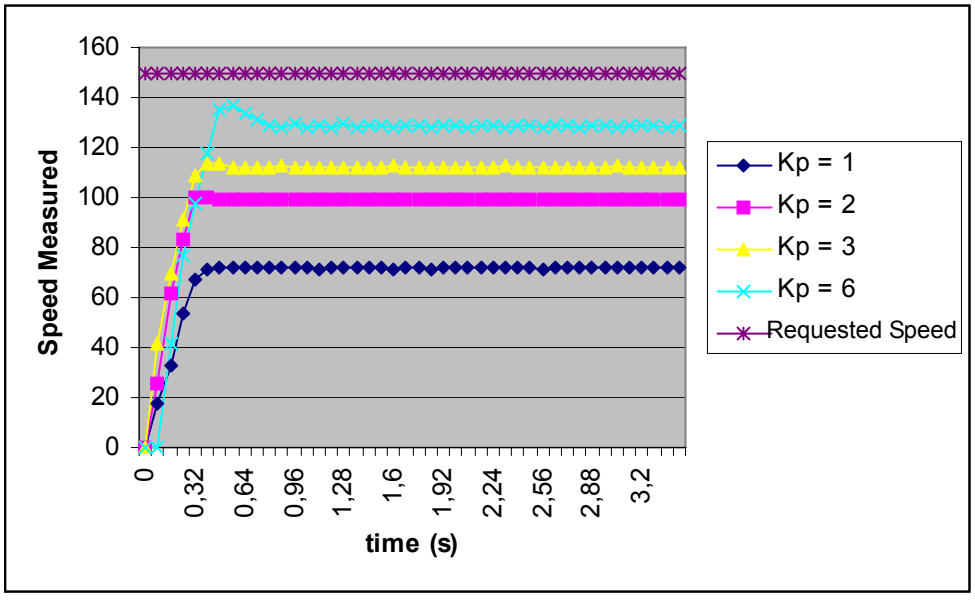
The close loop consists in a speed regulation loop with the PID corrector.

We will further explain how to adjust the coefficients KP and KI. The KD coefficient is present in the regulation loop but not used.

As previously shown, the KP coefficient is used to regulate the motor response time. In a first time set KI and KD to ‘0’.Try different value of KP to get a right motor response time.

- If the response time is too slow increase KP gain.
- If the response time is quick but unstable decrease KP gain.

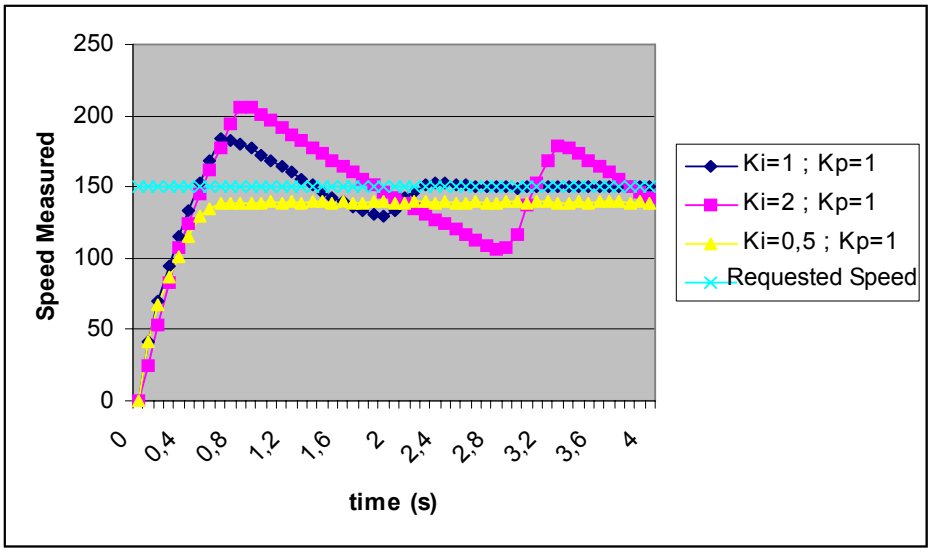
Figure 14. Kp tuning



KI parameter is used to cancel the static error. Leave the KP coefficient unchanged and set the KI parameter.

- If the error is still different from zero increase KI gain.
- If the error is cancelled but after oscillations decrease KI gain.

Figure 15. Ki tuning



In the Figure 14 and Figure 15 example, the right parameters are KP = 1, KI = 0.5 and KD = 0.

To adjust KD parameter :

- If the reponse is still low increase KD gain.
- If the reponse is still unstable decrease KD gain.

One another significant point is the sampling time. It has to be chosen according to the response time of the system. The sampling time must be at least twice smaller than the response time of the system (according to the Shannon-Nyquist criterie).

Two functions are available for the sampling time configuration (explained previously). They result in a global variable called g_tick which is set all 250us. With this variable it's possible to configure the sampling time.

CPU & Memory usage

All measurements has been realized with Fosc = 8MHz. It also depends on the motor type too (numbers of pair of poles). With a motor of 5 pair of poles, hall sensor frequency is five times slower than motor rotation.

All results in Figure 16 are obtained with a three phases BLDC motor with five pairs of poles and a maximum speed of 14000 rpm.

Figure 16. Microcontroller utilization rate

Function	Parameters	activation time	activation period	Ratio uc %
ovfl_timer()	all case	2us	2,05ms	0,1
mc_estimation_speed()	Speed = 14000 rpm	31 us	857,14 us	3,62
	Speed = 2800 rpm		4,29 ms	0,72
mc_switch_commutation()	Speed = 14000 rpm	17 us	142,86 us	11,89
	Speed = 2800 rpm		714 us	2,38
mc_regulation_loop()	Open Loop	4,5 us	80 ms	0,0056
	Close Loop	45 us	80 ms	0,056

In the worst case, the microcontroller utilization ratio is about 18% with a sampling time of 80ms at 14000 rpm.

A first estimation can be made for a faster motor and with a current regulation added. Activation time for mc_regulation_loop() grows up from 45us to 55us (we have to take into account the conversion time of the ADC about 7us). For this estimation, a BLDC motor with a current response time about 2-3ms, five pairs of poles and a maximum speed of 50000 rpm is chosen.

The maximum speed of the motor is about 50000 rpm, with a rotor of 5 pairs of poles we obtain a Hall sensors frequency of (50000rpm/60)*5 = 4167 Hz. The function mc_estimation_speed() is launched at each rising edge of hall sensor A, so every 240us with an activation time of 31us.

For mc_switch_commutation() the three hall sensors have to be considered. The function is executed at each edges of the three hall sensors (rising edge and falling edge) we have six interrupts in one Hall sensor period. It results in an activation period of 240/6us = 40us.

Finally, the sampling time of the regulation loop must to be at least twice smaller than the current reponse time of the motor (about 1ms).

All results are summarized Figure 17.

Figure 17. Microcontroller utilization rate estimated

Function	**Parameters**	**activation time**	**activation period**	**Ratio uc %**
ovfl_timer()	all case	2us	2,05 ms	0,1
mc_estimation_speed()	Speed = 50000 rpm	31 us	240 us	12,91
mc_switch_commutation()	Speed = 50000 rpm	17 us	40 us	42,5
mc_regulation_loop()	Close Loop	55 us	1 ms	5,5

In such case the utilization ratio of the microcontroller is about 61%.

All ratio measurement have been made with the same software. No communication mode are used (no UART, no LIN...).

In these conditions, the microcontroller memory usage is :
- 3175 bytes of CODE memory (Flash occupation = 38.7%).
- 285 bytes of DATA memory (SRAM occupation = 55.7%).

ATAVRMC100 Configuration and Use

Figure 18 shows us the complet schematics of the various operating mode of the starter kit ATAVRMC100.

Figure 18. Microcontroller I/O Ports use and communication modes

				Communication Mode				
			Operating Mode		STK500		JTAGICE mkII	
PORTB	PIN	Port Function	SENSOR	LIN	PC Link	Switchs/LEDs	Debug Wire	ISP
PB0/MISO/PSCOUT20	12	H_C	H_C	H_C	H_C	H_C	H_C	H_C
PB1/MOSI/PSCOUT21	13	L_C	L_C	L_C	L_C	L_C	L_C	L_C
PB2/ADC5/INT1	20	VMOT	VMOT	VMOT	VMOT	VMOT	VMOT	VMOT
PB3/AMP0-	27	*EXT1*	Depend on the communication mode	*EXT1*	*EXT1*	*EXT1*	*EXT1*	*EXT1*
PB4/AMP0+	28	*EXT2*		*EXT2*	*EXT2*	*EXT2*	*EXT2*	*EXT2*
PB5/ADC6/INT2	30	*EXT5*		*EXT5*	*EXT5*	*EXT5*	*EXT5*	*EXT5*
PB6/ADC7/PSCOUT11/ICP1B	31	L_B	L_B	L_B	L_B	L_B	L_B	L_B
PB7/ADC4/PSCOUT01/SCK	32	L_A	L_A	L_A	L_A	L_A	L_A	L_A
PORTC								
PC0/INT3/PSCOUT10	2	H_B	H_B	H_B	H_B	H_B	H_B	H_B
PC1/PSCIN1/OC1B	7	*EXT3*	Depend on the communication mode	*EXT3*	*EXT3*	*EXT3*	*EXT3*	*EXT3*
PC2/T0/PSCOUT22	10	*EXT4*		*EXT4*	*EXT4*	*EXT4*	*EXT4*	*EXT4*
PC3/T1/PSCOUT23	11	LIN_NSLP		LIN_NSLP	LIN_NSLP	LIN_NSLP	LIN_NSLP	LIN_NSLP
PC4/ADC8/AMP1-	21	V_Shunt-	V_Shunt-	V_Shunt-	V_Shunt-	V_Shunt-	V_Shunt-	V_Shunt-
PC5/ADC9/AMP1+	22	V_Shunt+	V_Shunt+	V_Shunt+	V_Shunt+	V_Shunt+	V_Shunt+	V_Shunt+
PC6/ADC10/ACMP1	26	HallB / BEMF_B	HallB	Depend on the operating mode				
PC7/D2A	29	DAC_OUT	DAC_OUT	DAC_OUT	DAC_OUT	DAC_OUT	DAC_OUT	DAC_OUT
PORTD								
PD0/PSCOUT00/XCK/SS_A	1	H_A	H_A	H_A	H_A	H_A	H_A	H_A
PD1/PSCIN0/CLKO	4	Over_Current	Over_Current	Over_Current	Over_Current	Over_Current	Over_Current	Over_Current
PD2/PSCIN2/OC1A/MISO_A	5	MISO / *EXT10*	Depend on the communication mode	*EXT10*	*EXT10*	*EXT10*	*EXT10*	MISO
PD3/TXD/DALI/OC0A/SS/MOSI_A	6	MOSI / LIN TxD / TxD / *EXT7*		LIN TxD	TxD	POT	*EXT7*	MOSI
PD4/ADC1/RXD/DALI/ICP1A/SCK_A	16	SCK / LIN RxD / RxD / POT / *EXT8*		LIN RxD	RxD	*EXT8*	*EXT8*	SCK
PD5/ADC2/ACMP2	17	HallC / BEMF_C	HallC	Depend on the operating mode				
PD6ADC3/ACMPM/INT0	18	VMOT_Half	VMOT_Half	VMOT_Half	VMOT_Half	VMOT_Half	VMOT_Half	VMOT_Half
PD7/ACMP0	19	HallA / BEMF_A	HallA	Depend on the operating mode				
PORTE								
PE0/RESET/OCD	3	NRES / *EXT9*	Depend on the communication mode	*EXT9*	*EXT9*	*EXT9*	NRES	NRES
PE1/OC0B/XTAL1	14	*EXT6*		*EXT6*	*EXT6*	*EXT6*	*EXT6*	*EXT6*
PE2/ADC0/XTAL2	15	LED		LED	LED	LED	LED	LED

Operating mode

Two differents operating modes are available, set jumpers JP1, JP2 and JP3 according to Figure 19 to select between both modes. In this application, only Sensor mode is used. Refer to the ATAVRMC100 User Guide for complete description of hardware.

Figure 19. Sensor mode selection

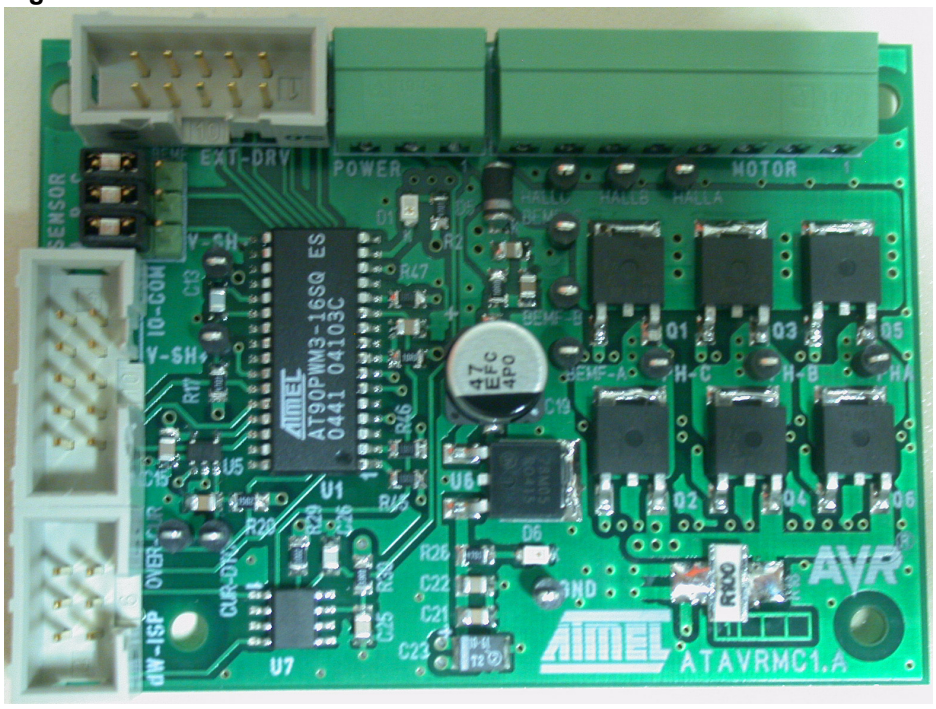
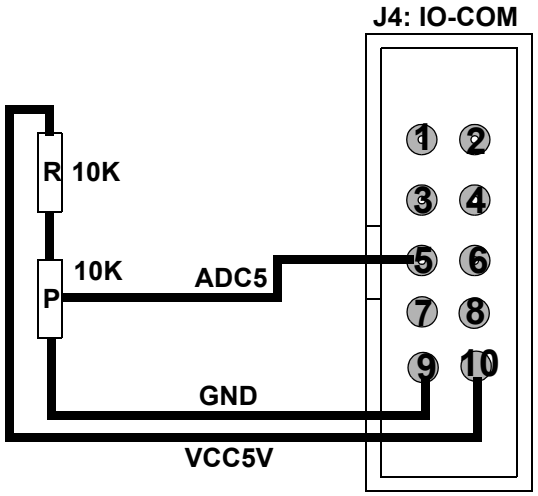


Figure 19 shows the default jumpers setting to use the software associated with this application note.

The software delivered with the ATAVRMC100 board allows two operating modes :

- No external component, the motor start with a speed of 100%.
- One external potentiometer use to adjust the speed of the motor.

Figure 20. Potentiometer connection.



Conclusion

This application note provides a software and hardware solution for sensor brushless DC motors applications. All source code is available on the Atmel web site.

The software library provides functions to start and control the speed of any sensor three phases brushless DC motors.

The hardware is based on the minimal design with minimal external components required for control sensor brushless DC motors.

The AT90PWM3 CPU and memory usage allow a developer to expand his horizons with new functionalities.

Figure 21. Schematic page 1/4

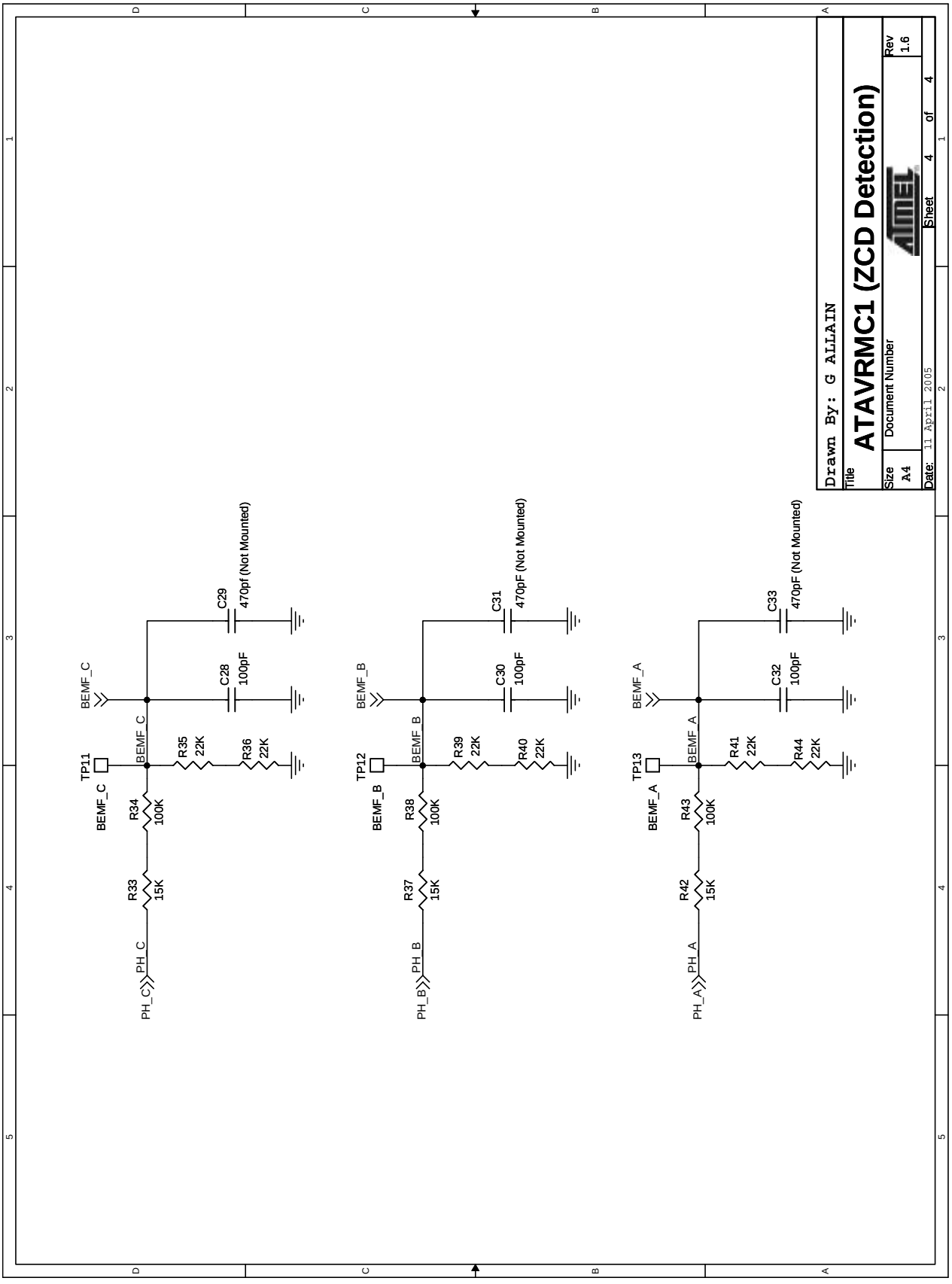


Figure 22. Schematic page 2/4

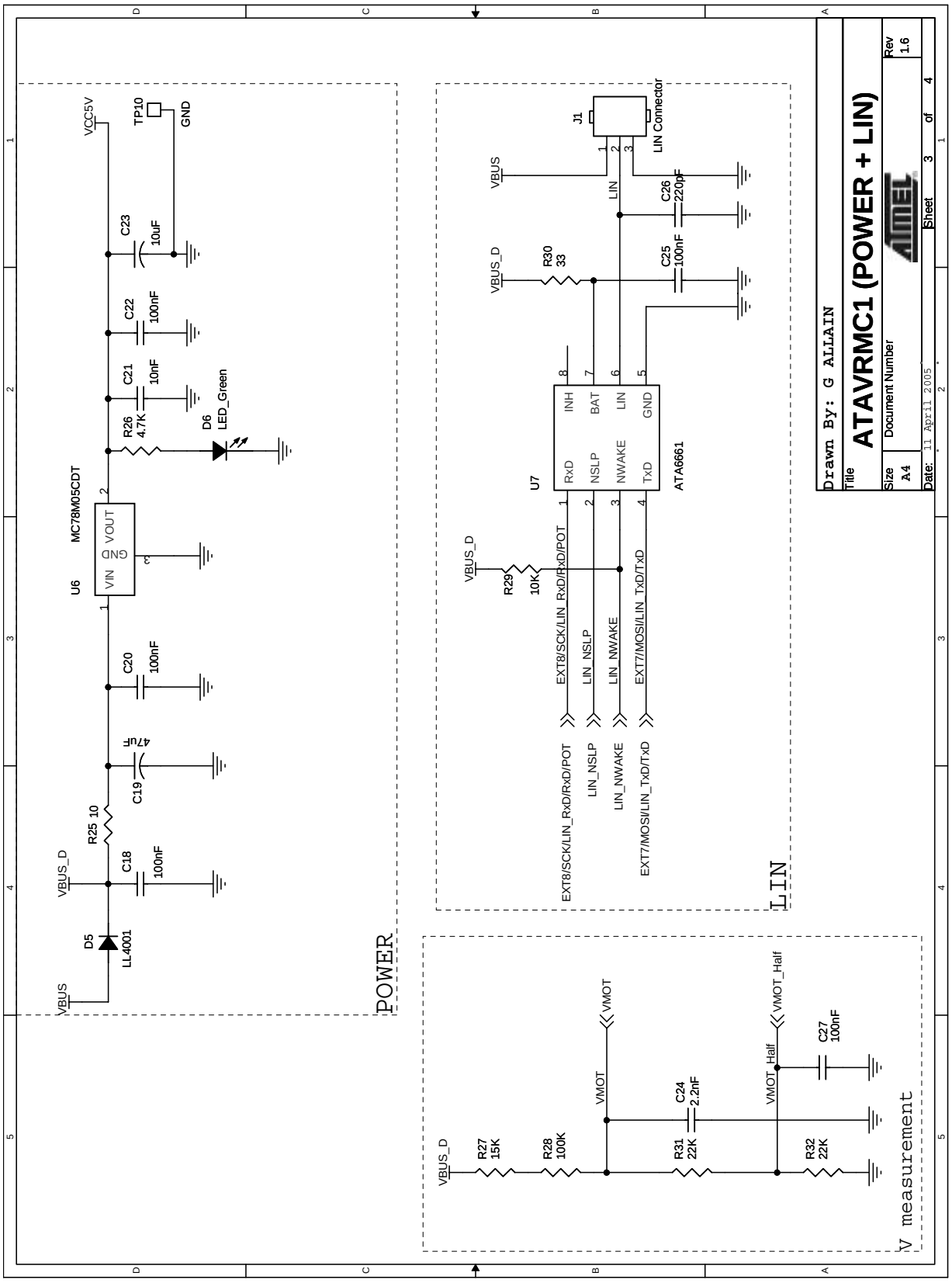


Figure 23. Schematic page 3/4

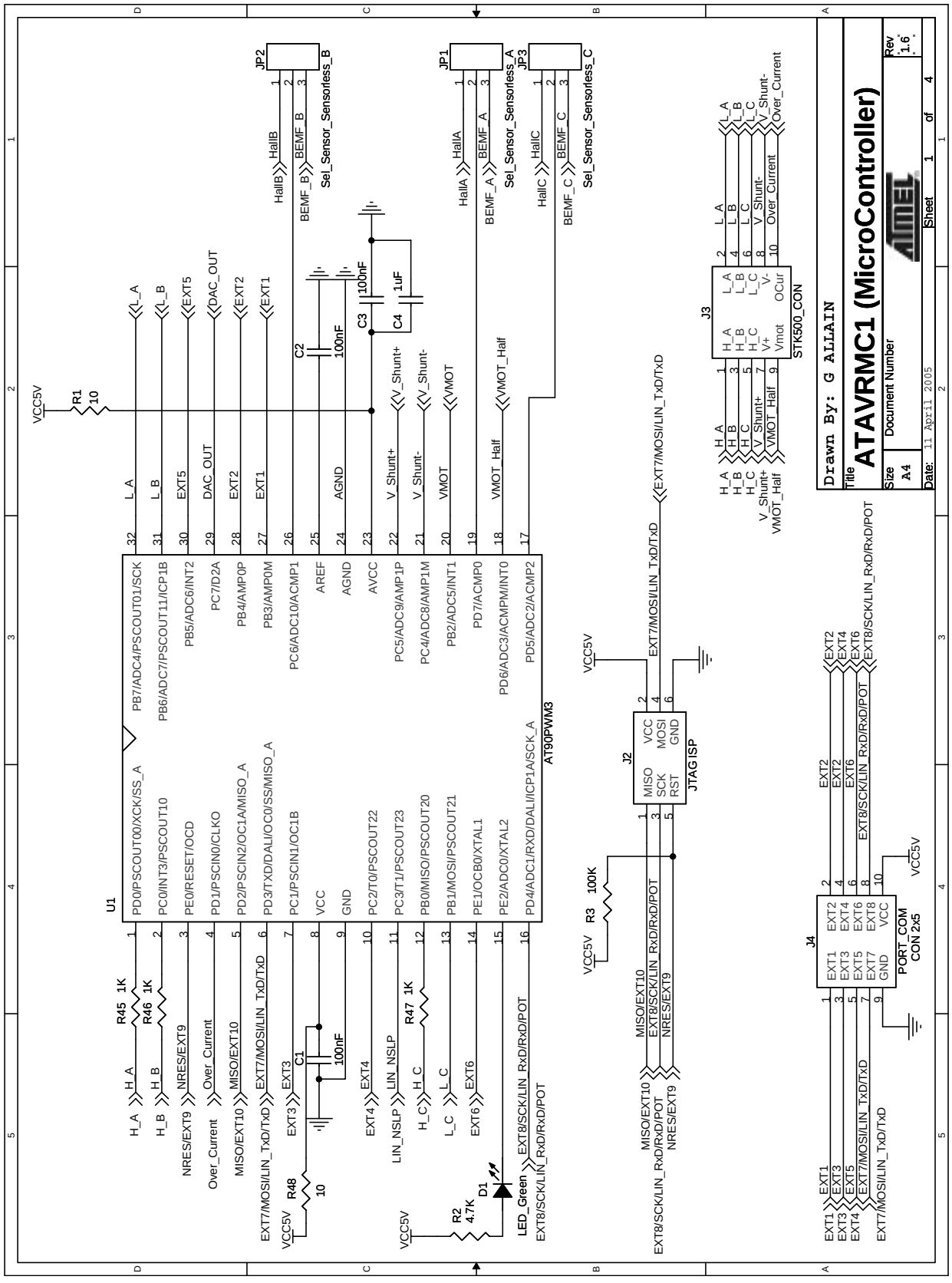
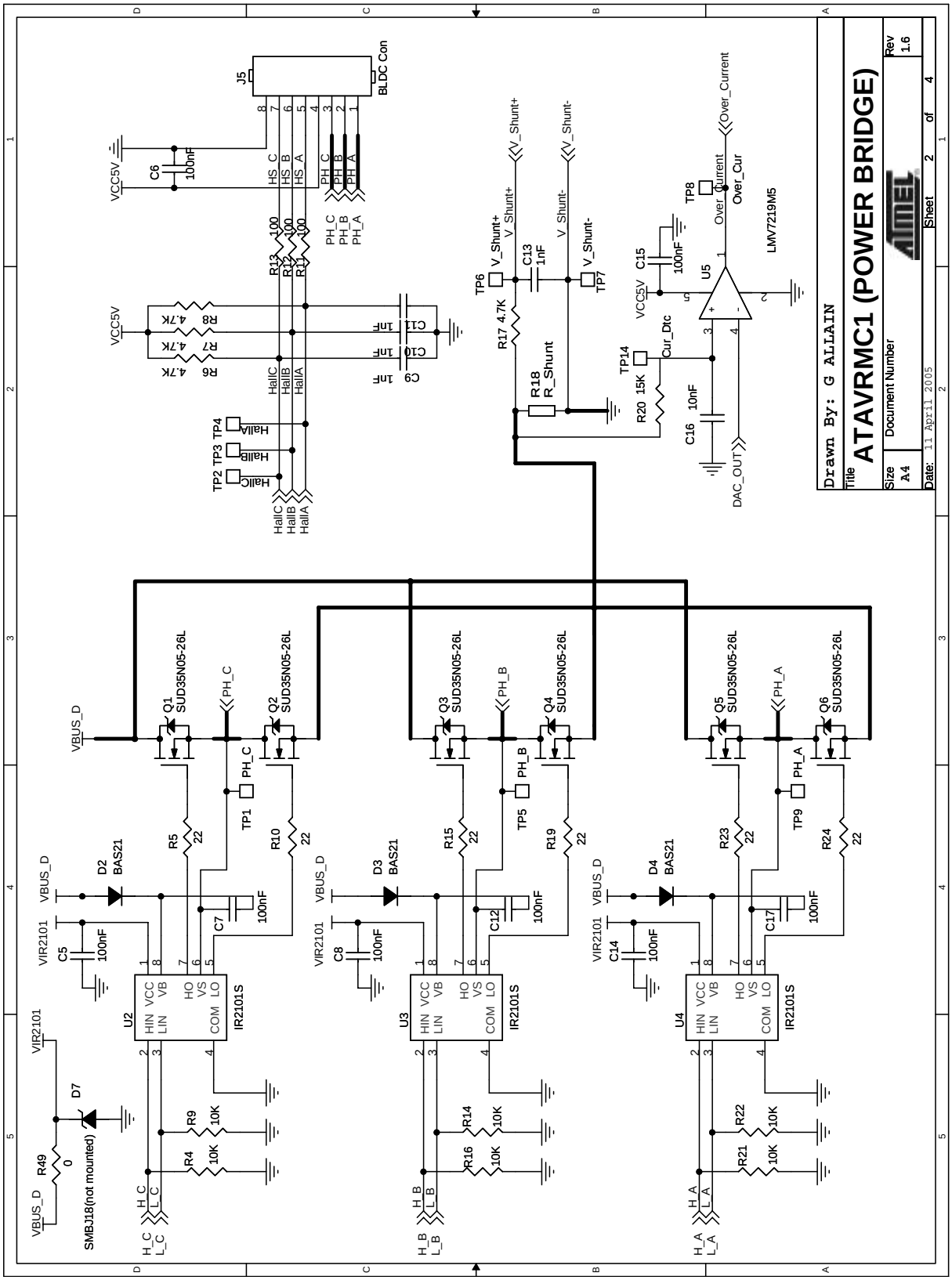


Figure 24. Schematic page 4/4



Atmel Corporation

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

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

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

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

7518A–AVR–07/05