AVR138: ATmega32M1 family PSC Cookbook

8-bit AVR® Microcontroller

Application Note

Features

- **PSC Basics**
- **Waveform Generation using Power Stage Controller**
- **Code Examples**

1. Preamble

ATMEL® is introducing a new 8 bit AVR® family with the ATmega32M1.

This family embeds an innovative PSC dedicated to motor control applications and led lighting applications.

Note: This PSC is different that the PSC used in the AT90PWMx family dedicated for lighting. As it is primarily dedicated for motor control and to simplify its use, it is a merge of the three AT90PWMx family PSC. The AVR434 'PSC cookbook' is specially dedicated for the AT90PWMx family PSC and is not applicable for the ATmega32M1 family PSC.

2. Introduction

This application note is an introduction to the use of the Power Stage Controller (PSC) available in ATmega32M1 family. The object of this document is to give a general overview of the PSC, show its various modes of operation and explain how to configure them. The code examples will make this clearer and can be used as guideline for other applications. The examples are developed and tested on ATmega32M1.

PSC description and additional information can be found in the data sheet and in application notes where the PSC is used.

This application note describes waveforms which can be generated by the PSC and how to program it. It also introduces fault modes and output enable management.

All source files are available on the Atmel website. They are written on GCC compiler, nevertheless they can easily modified to compile on other compilers.

3. General Description

The Power Stage Controller is a special timer with 3 modules dedicated to drive the power stage of an equipment or a board. The PSC is logical level compatible and is able to drive a bridge of power transistors.

The PSC can drive different kind of bridges (DC, Brushless DC, AC motors ...)

Rev. 8122A–AVR–03/08

Each of the 3 PSC modules can be seen as a PWM generator with two complementary outputs. To provide a self running PSC mode without the need of embedded software action, the PSC has 3 inputs which can stop the waveform generation. For example, in a current sensing mode, the current can be monitored by a comparator which can stop the PSC waveform when a maximum current is reach.

The PSC can be clocked by a fast clock like the output of a 64 MHz PLL. So it can generate high speed PWM with a high resolution. It can also be clocked by slower clocks such as PLL intermediate output or by CPU clock (CLKio). Moreover it includes a prescaler to generate signals with very low frequency.

4. PSC Applications

The PSC is intended to drive applications with a power stage:

- Motor Control (Waveform Generation and Speed/Torque regulation)
- Led Lighting Control (Current Regulation)

Figure 4-1. Led boost control example

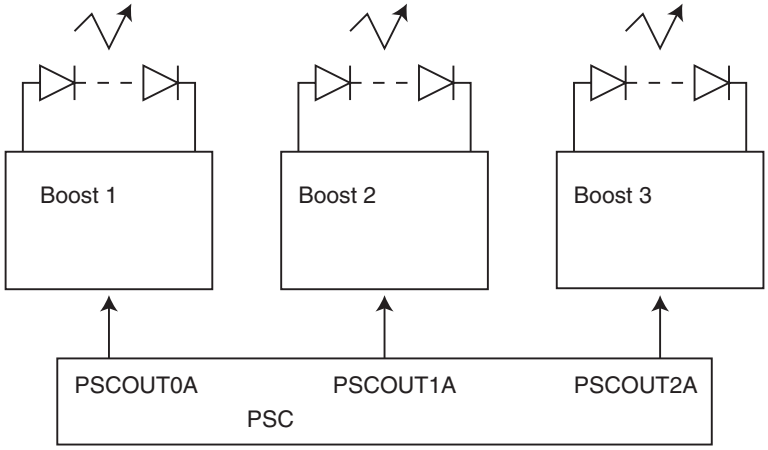
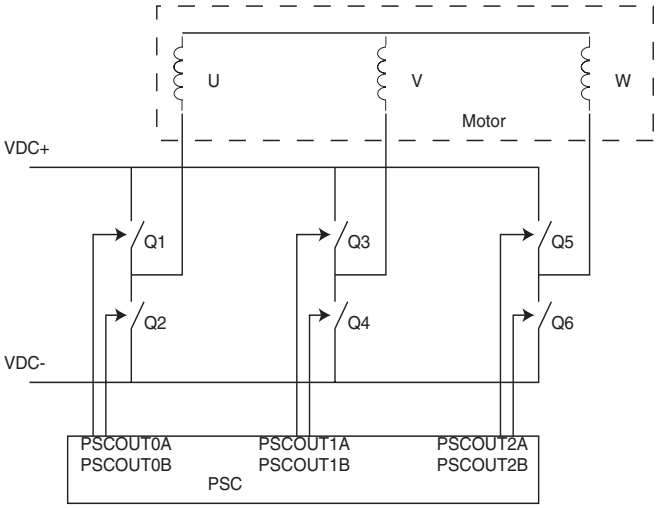


Figure 4-2. Motor control example



Note: Motor can be BLDC or asynchronous motor

5. PSC Modes

5.1 Prerequisite

We recommend reading the following chapters of the ATmega32M1 datasheet:

- Overview
- PSC Description
- Functional Description

5.2 Why Different Modes

The PSC provides 2 modes of operation:

1 ramp mode	This mode is used to generate overlapped waveform. The major risk of this mode is when driving a half bridge to have cross-conduction.
centered mode	The PSC output waveforms are symmetrical and centered. This mode is useful for space vector pwm methods to generate sinusoidal waveforms. (Overlapped waveforms are possible)

5.3 Running Modes Examples

All examples are written in C and embedded in a AVR Studio GCC project.

These examples are given to quickly start the generation of waveforms and to evaluate the two running modes.

The name of the AVR projects are OneRampMode.aps and CenteredMode.aps.

Thanks to PSC_MODE definition, we can select the running mode.

5.3.1 Mode: one ramp

The One Ramp mode can also be seen as a edge aligned mode.

To select the 1 ramp mode use the following syntax:

#define PSC_MODE PSC_MODE_ONE_RAMP

In the following example, CLOCK PSC = CLK PLL = 64 Mhz

Figure 5-1. One Ramp Mode

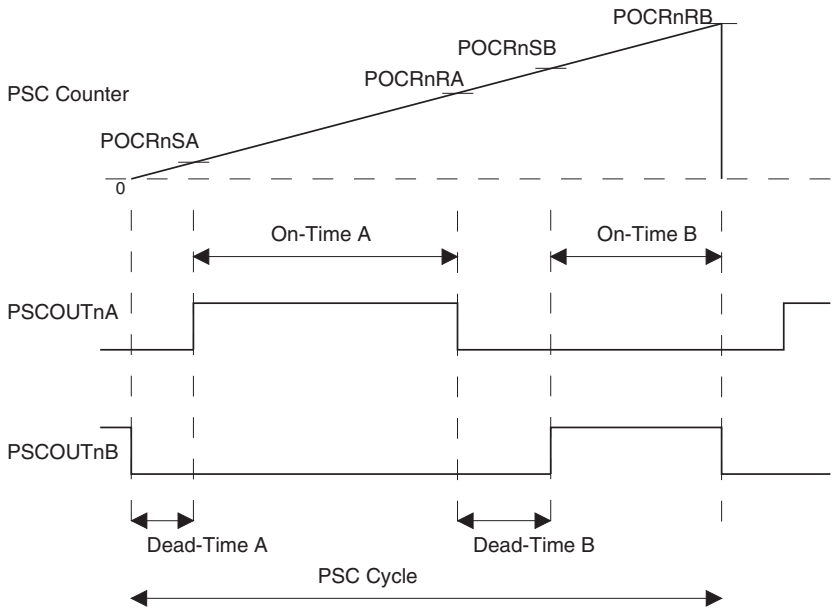
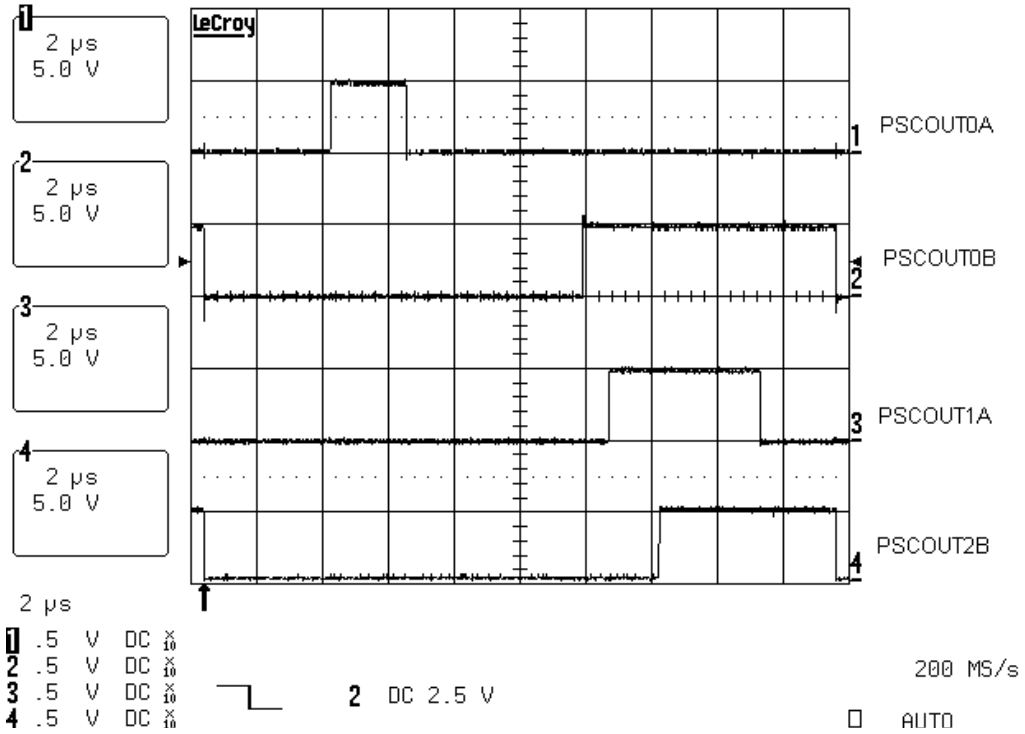


Table 5-1. Settings for Figure 5-2.

	PSC SFR	Instruction	Result in Clock Number	Result in µS
PSCOUT0A PSCOUT0B	POCR0SA	A_SA_VAL = 250	Dead Time 0A = 250 + 1	3.9µS
	POCR0RA	A_RA_VAL = 400	On Time 0A = 400 - 250	2.3µS
	POCR0SB	A_SB_VAL = 750	Dead Time 0B = 750 - 400	5.5µS
			On Time 0B = 1250 - 750	7.8µS
PSCOUT1A PSCOUT1B	POCR1SA	B_SA_VAL = 800	Dead Time 1A = 800 + 1	12.5µS
	POCR1RA	B_RA_VAL = 1100	On Time 1A = 1100 - 800	4.7µS
	POCR1SB	B_SB_VAL = 1150	Dead Time 1B = 1150 - 1100	0.8µS
			On Time 1B = 1250 - 1150	1.6µS
PSCOUT2A PSCOUT2B	POCR2SA	C_SA_VAL = 600	Dead Time 2A = 600 + 1	9.4µS
	POCR2RA	C_RA_VAL = 800	On Time 2A = 800 - 600	3.1µS
	POCR2SB	C_SB_VAL = 900	Dead Time 2B = 900 - 800	1.6µS
			On Time 2B = 1250 - 900	5.5µS
	POCR_RB	RB_VAL = 1250		

Figure 5-2. One Ramp Waveforms



One Ramp With Overlapped Waveforms

To select the one ramp mode with overlapped waveforms un-comment the following line:

#define T_OVERLAP

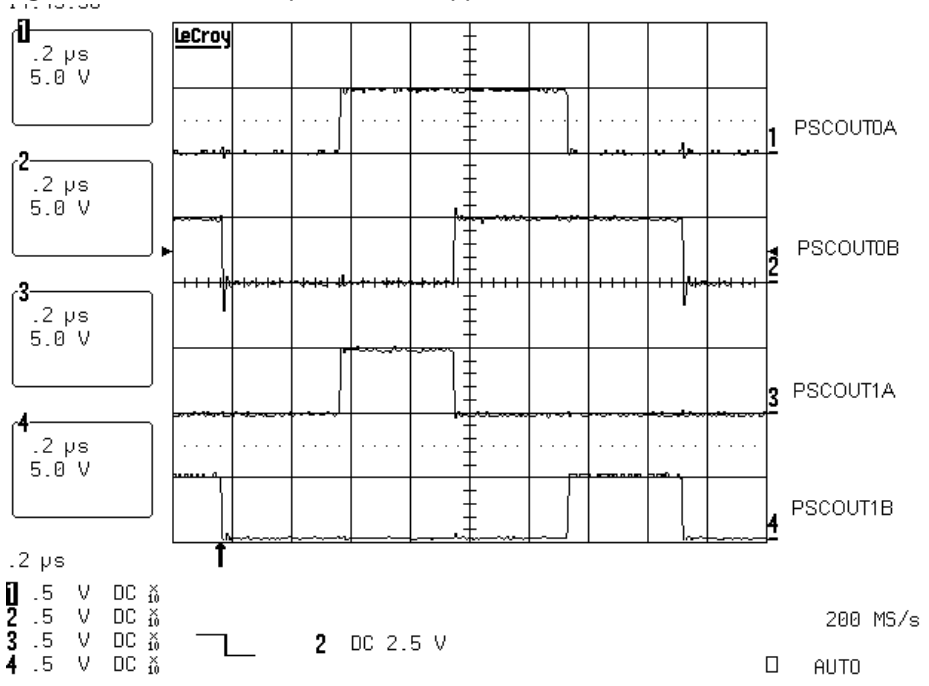
Table 5-2. Settings for Figure 5-3.

	PSC SFR	Instruction	Result in Clock Number	Result in µS
PSCOUT0A PSCOUT0B	POCR0SA	A_SA_VAL = 25	Dead Time 0A = 25 + 1	0.41µS
	POCR0RA	A_RA_VAL = 75	On Time 0A = 75 - 25	0.78µS
	POCR0SB	A_SB_VAL = 50	Dead Time 0B = 50 - 75	-0.39µS
			On Time 0B = 100 - 50	0.78µS
PSCOUT1A PSCOUT1B	POCR1SA	B_SA_VAL = 25	Dead Time 1A = 25 + 1	0.41µS
	POCR1RA	B_RA_VAL = 75	On Time 1A = 75 - 25	0.78µS
	POCR1SB	B_SB_VAL = 50	Dead Time 1B = 50 - 75	-0.39µS
			On Time 1B = 100 - 50	0.78µS
PSCOUT2A PSCOUT2B	POCR2SA	C_SA_VAL = 20	Dead Time 2A = 20 + 1	9.4µS
	POCR2RA	C_RA_VAL = 40	On Time 2A = 40 - 20	3.1µS
	POCR2SB	C_SB_VAL = 60	Dead Time 2B = 60 - 40	1.6µS
			On Time 2B = 100 - 60	5.5µS
	POCR_RB	RB_VAL = 100		

PSCOUT0A and PSCOUT0B have overlap protection disabled.

PSCOUT1A and PSCOUT1B have overlap protection enabled.

Figure 5-3. One Ramp With Overlapped Waveforms



5.3.2 Mode: Centered

To select the centered mode use the following syntax:

#define PSC_MODE PSC_MODE_CENTERED

In the following example, CLOCK PSC = CLK IO = 8Mhz

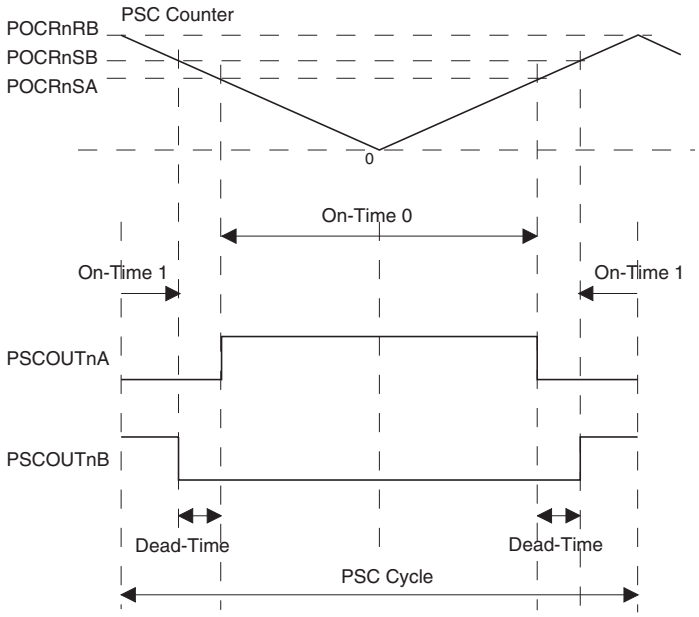
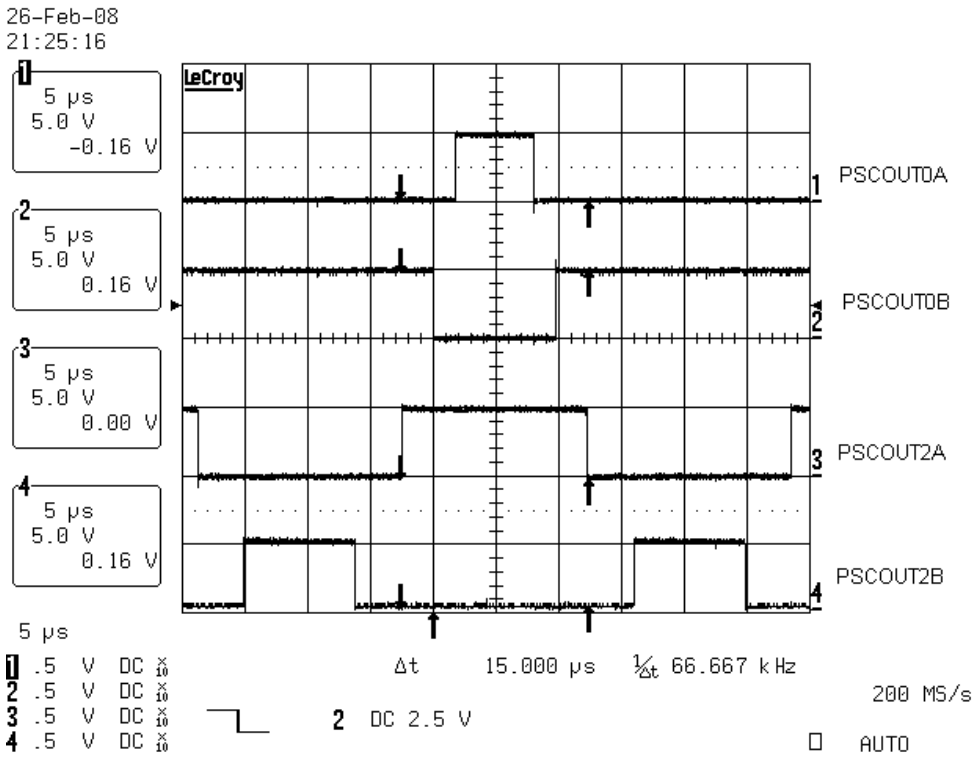


Table 5-3. Settings for Figure 5-4.

	PSC SFR	Instruction	Result in Clock Number	Result in µS
PSCOUT0A PSCOUT0B	POCR0SA	A_SA_VAL = 25	On Time 0A = 2 * 25	6.2µS
	POCR0RA	A_RA_VAL = 75	RA is used for synchr. signal	
	POCR0SB	A_SB_VAL = 40	On Time 0B = 2*(125 - 40 + 1)	21.5µS
			Dead Time 0 = 40 - 25	1.9µS
PSCOUT1A PSCOUT1B	POCR1SA	B_SA_VAL = 110	On Time 1A = 2 * 110	27.5µS
	POCR1RA	B_RA_VAL = 80	RA is used for synchr. signal	
	POCR1SB	B_SB_VAL = 115	On Time 1B = 2*(125 - 115 + 1)	2.8µS
			Dead Time 1 = 115 - 110	0.6µS
PSCOUT2A PSCOUT2B	POCR2SA	C_SA_VAL = 60	On Time 2A = 2 * 60	15µS
	POCR2RA	C_RA_VAL = 80	RA is used for synchr. signal	
	POCR2SB	C_SB_VAL = 90	On Time 2B = 2*(125 - 90 + 1)	9µS
			Dead Time 2 = 90 - 60	3.7µS
	POCR_RB	RB_VAL = 125		

Figure 5-4. Centered Mode Waveforms



6. PSC Inputs

6.1 Prerequisite

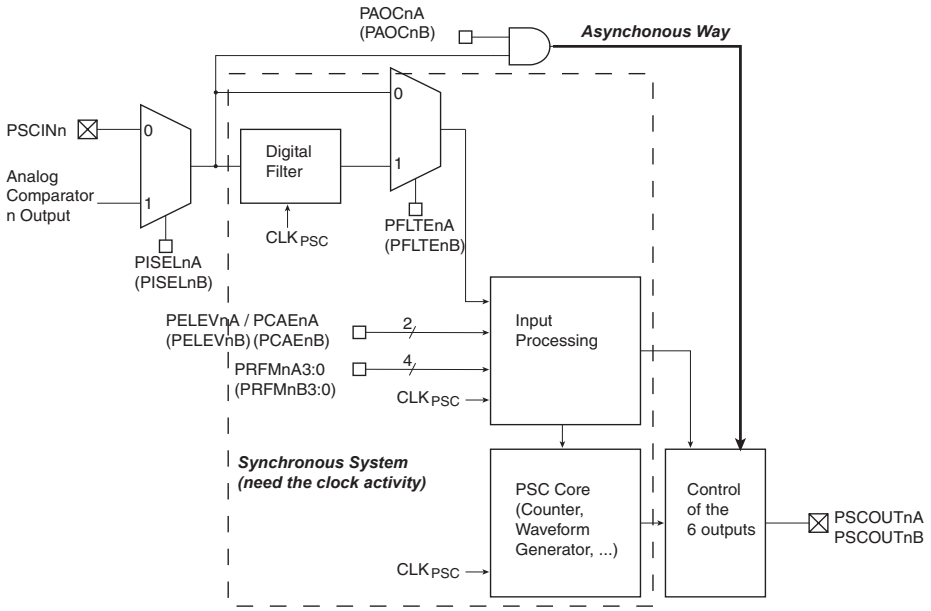
We recommend reading the following chapters of the ATmega32M1 datasheet:

- Overview
- PSC Description
- PSC Inputs

6.2 Description

The PSC has 3 identical inputs which can be programmed to quickly react on the PSC outputs. The outputs can be de-activated definitively (fault mode) or only during the active state of the selected input. The PSC inputs have a digital filter which can be bypassed to get a shorter reaction time.

Figure 6-1. PSC Input Block Diagram



The following examples are given to quickly start the use of PSC inputs and to evaluate the input modes. It is done in one ramp mode. Thanks to PSC_MODE definition, they can easily be modified to test inputs in centered mode.

These examples can be used on STK500/STK524 boards.

The name of the AVR project is test_halt.aps.

The PSCIN0 signal is generated by an AGILENT 33250A Function Waveform Generator synchronized on the PSCOUT1B output.

Synchronous mode versus asynchronous modes of inputs:

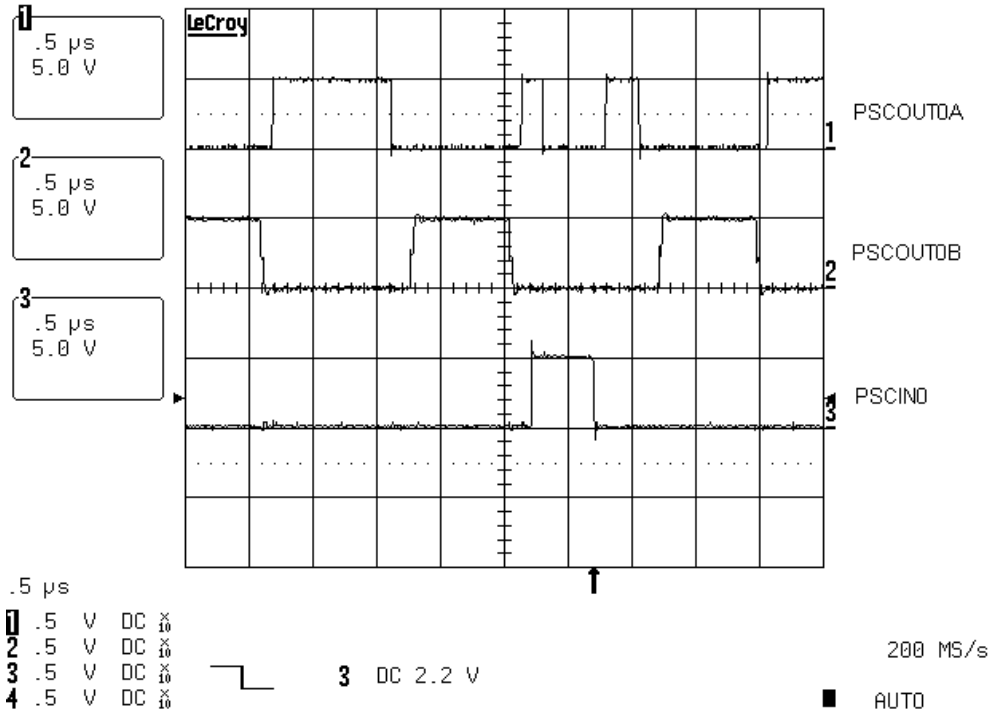
All the following examples are given in synchronous mode (PSC_SYNCHRONOUS_OUTPUT_CONTROL definition), the system clock running is necessary to propagate the PSC input to the PSC output. The user can easily test the direct propagation of the PSC input by using the PSC_ASYNCHRONOUS_OUTPUT_CONTROL definition in the source files of the examples.

6.2.1 Input Mode 1: De-activate module 0 Output A

Comment the #define PSC_TEST_HALT in the source file to get the following waveforms.

Thanks to PSC_USE_HIGH_LEVEL, the PSCIN0 input acts on PSCOUT0A when it is in a high level.

Figure 6-2. PSC Input Mode 1 on PSCIN0 input



6.2.2 Input Mode 7: Halt PSC and Wait for Software Action

PSCIN0 input can easily be activated by connected PD1 to SW0 switch on STK500 board.

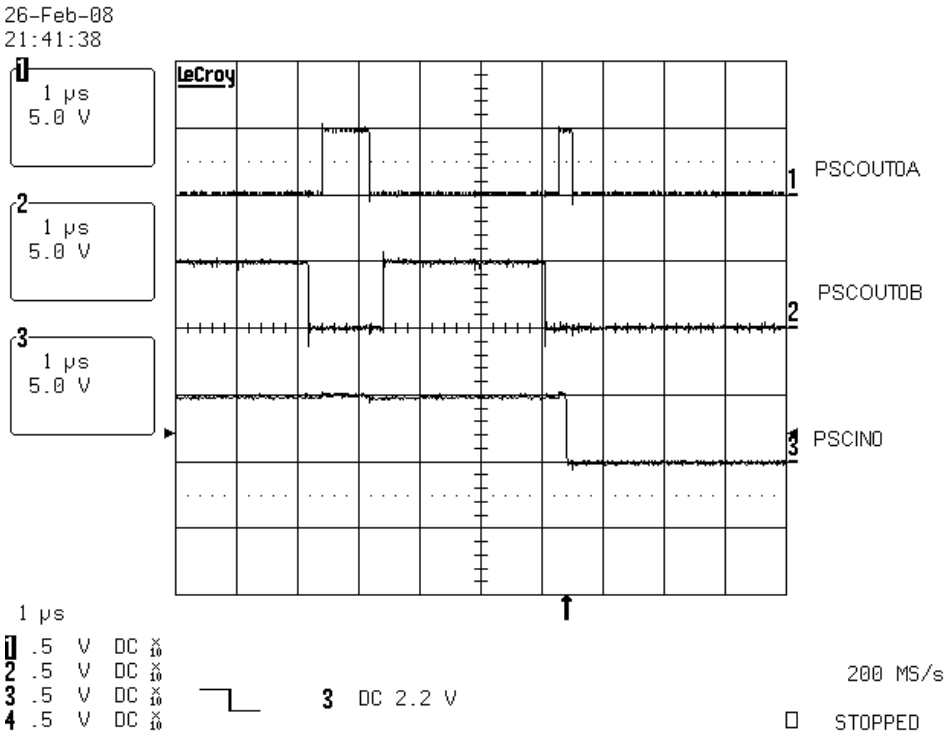
When PSC is in fault state, the software will reset it when PE2 equals 0. So PE2 can be connected to SW1 to restart the PSC.

Thanks to PSC_USE_LOW_LEVEL, the PSCIN0 input acts on PSCOUT0A when it is in a low level. The restart of the PSC is not possible while PSCIN0 remains active (low level).

Un-comment the #define PSC_TEST_HALT in the source file.

The PSCIN0 configuration uses PSC_INPUT_HALT_PSC when it is involved.

Figure 6-3. PSC Input Mode 7 on PSC0IN0 input



6.2.3 Use of the filter on the PSC inputs

In this example, PSC operates with the 64MHz output of the PLL.

Comment the #define PSC_TEST_HALT in the source file to get the following waveforms.

Un-comment the #define TEST_FILTER

Table 6-1. Settings for Figure 6-4. and Figure 6-5.

	PSC SFR	Instruction	Result in Clock Number	Result in µS
PSCOUT0A PSCOUT0B	POCR0SA	A_SA_VAL = 5	Dead Time 0A = 5 + 1	0.09µS
	POCR0RA	A_RA_VAL = 25	On Time 0A = 25 - 5	0.3µS
	POCR0SB	A_SB_VAL = 30	Dead Time 0B = 30 - 25	0.08µS
			On Time 0B = 40 - 30	0.16µS
	POCR_RB	RB_VAL = 40		

6.2.3.1 Filter Propagation Delay

On Figure 6-4. the input filter is enable, the propagation delay equals 92.5nS.

On Figure 6-5. the input filter is disabled, the propagation delay equals 30nS.
The filter delay is 4*Clock cycles, which corresponds to 62.5nS.

Figure 6-4. PSC0IN0 Configured to act on PSCOUT0A (filter enabled)

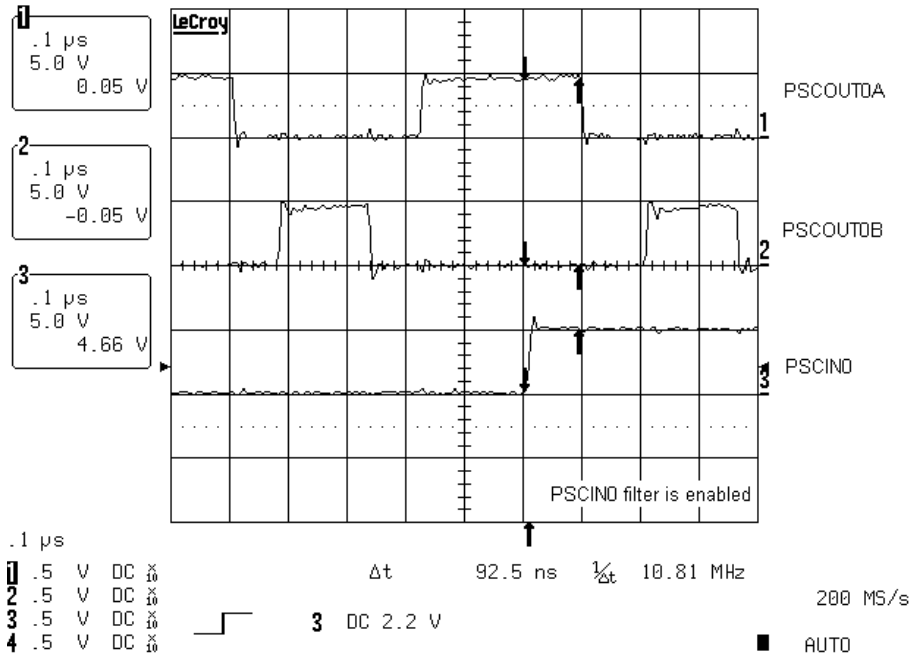
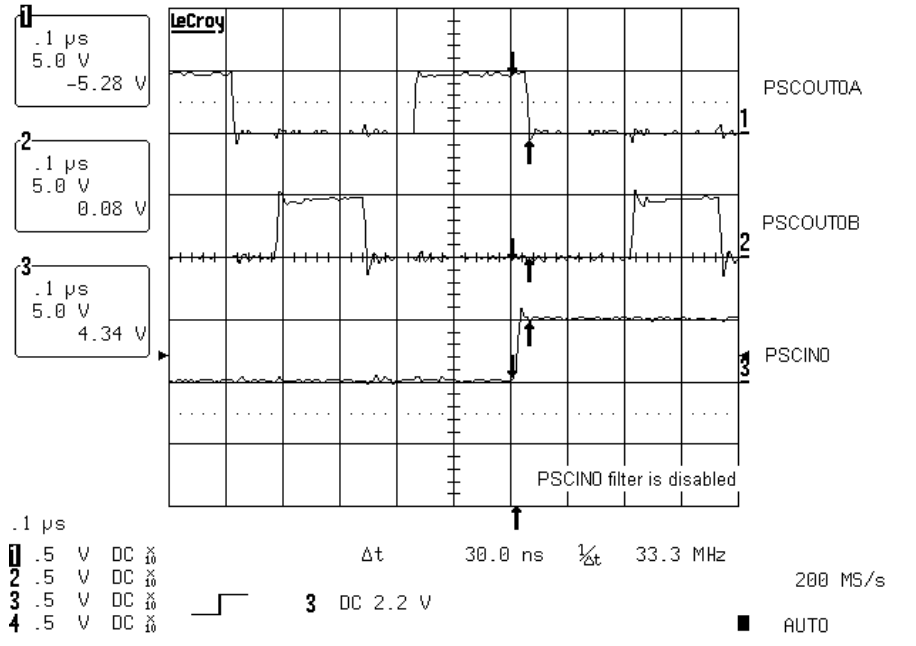


Figure 6-5. PSC0IN0 Configured to act on PSCOUT0A (filter disabled)



6.2.3.2 *Filter Performance*

On Figure 6-6. the input filter is disabled, the 50nS spike on PSCIN0 is propagated on PSCOUT0A.

On Figure 6-7. the input filter is enabled, the 50nS spike on PSCIN0 is not propagated on PSCOUT0A

Figure 6-6. PSC0IN0 Configured to act on PSCOUT0A (filter disabled)

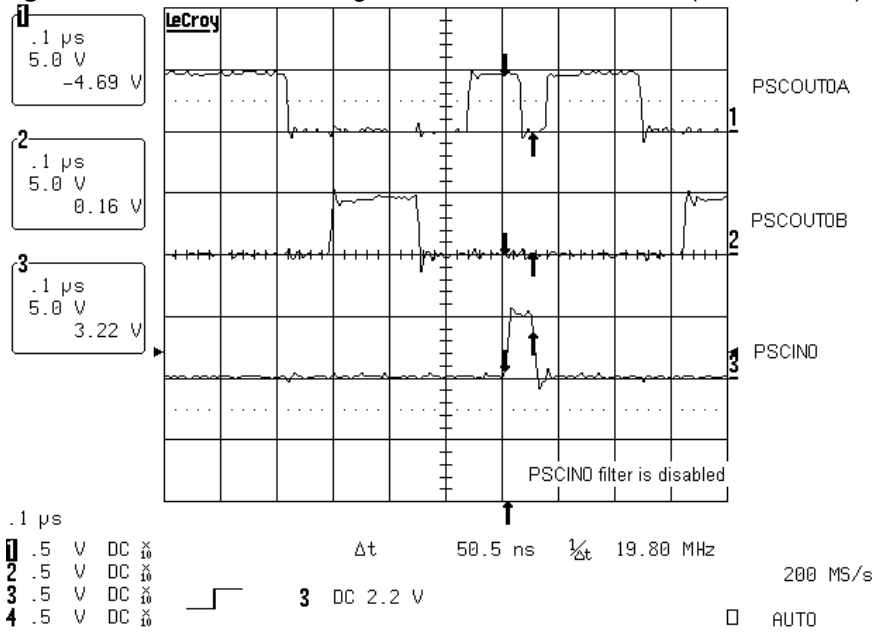
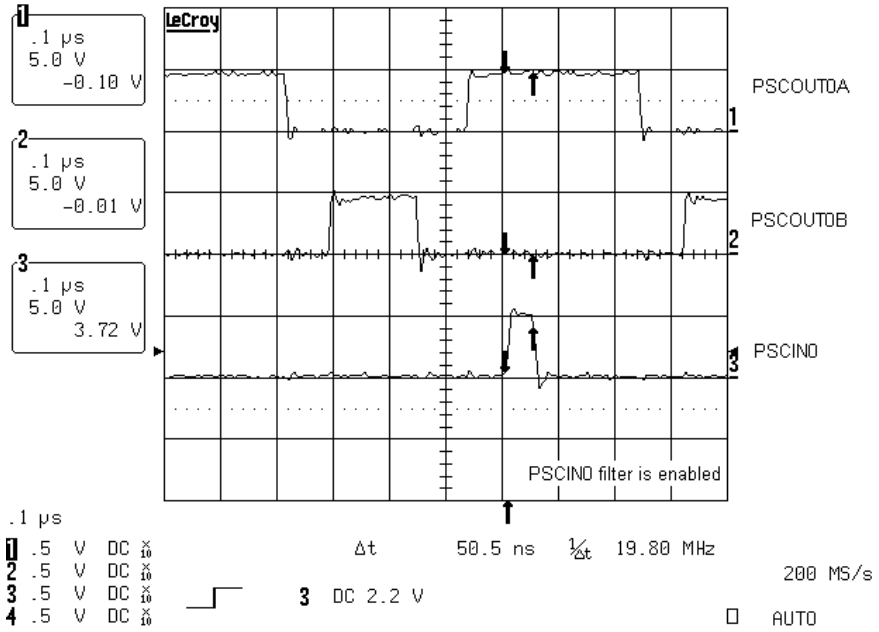


Figure 6-7. PSC0IN0 Configured to act on PSCOUT0A (filter enabled)



6.2.4 Use of the Comparator (ACMP0) as PSC input

This example demonstrates the use of comparator 0 as PSC input.

The name of the AVR project is Comparator.aps.

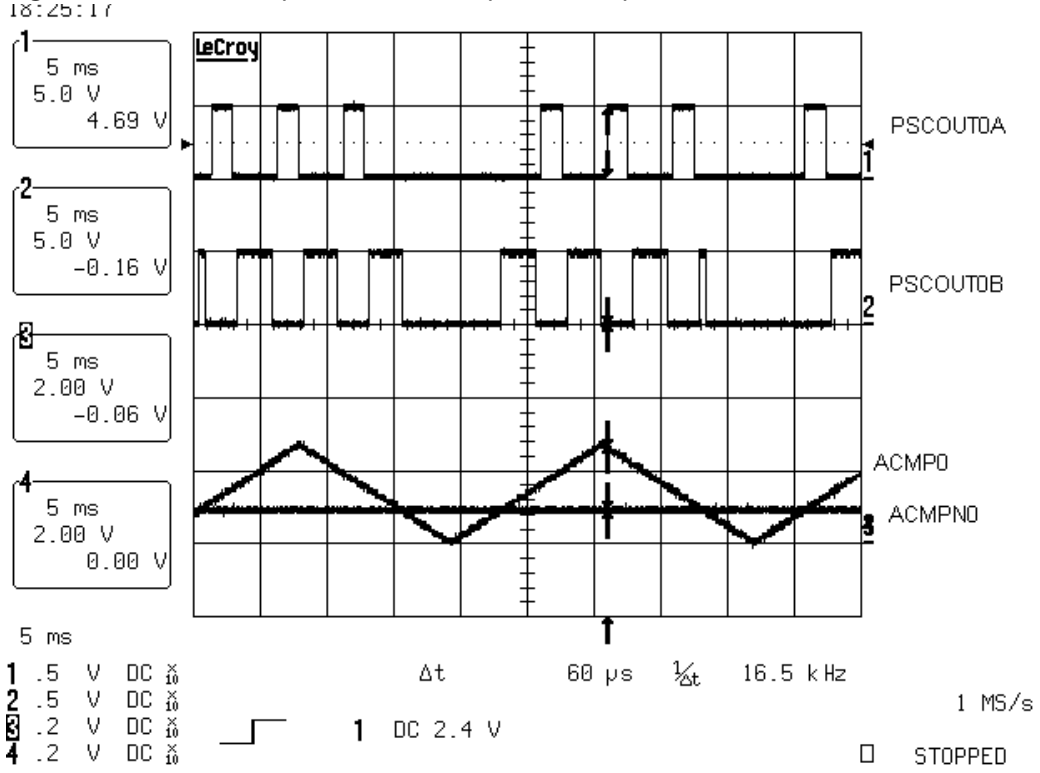
The comparator 0 is used as PSC input thanks to Psc_config_input0(..) functions.

Comparator 0 negative input is connected to a 0.9V DC source.

Comparator 0 positive input is connected to the output of an AGILENT 33250A Waveform Generator. The Waveform Generator generates a saw signal from 0 to 2.6V.

We can see that each time the positive input is below the negative input the outputs of the PSC are de-activated.

Figure 6-8. PSC Outputs versus COmparator 0 inputs



6.3 Output Modulation with POC register

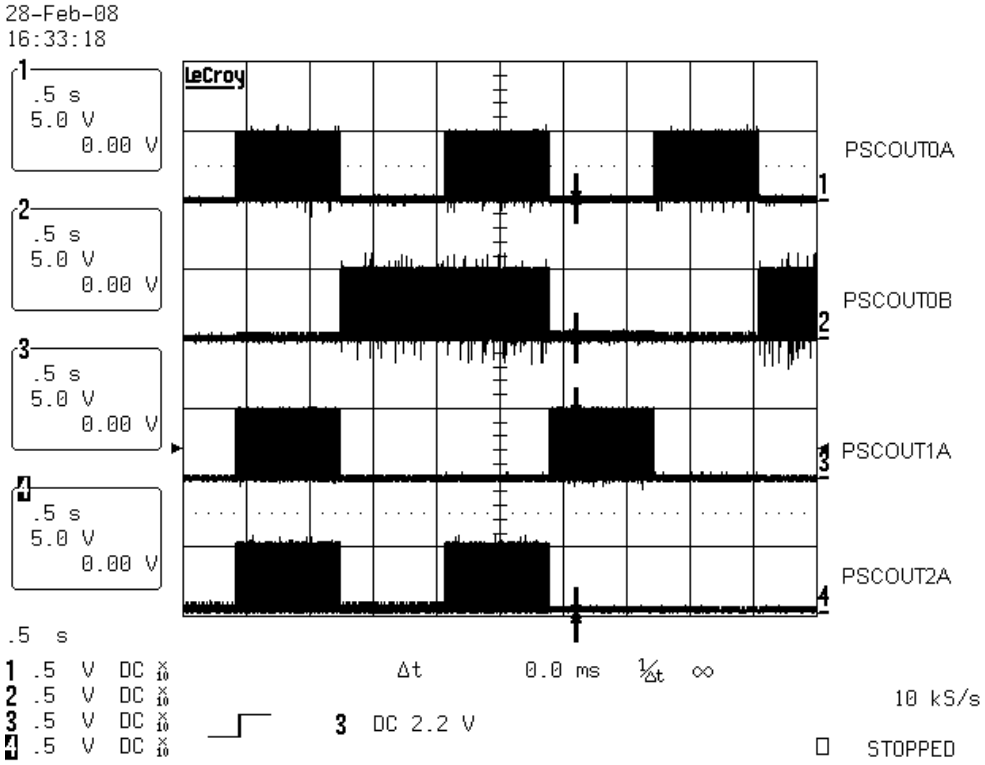
This example demonstrates the use of PSC Output Configuration (POC) register to validate or not validate the PSC outputs. This register is useful to drive BLDC motors. It can select the good outputs according to the Hall Sensor values or according to the Back-Emf detection.

The example can be used on STK500/STK524 boards.

The name of the AVR project is OutputEnable.aps.

The user can connect the PSC outputs to the STK500 LED inputs and see a kind of LED chaser where the light intensity of each LED is adjusted by the duty cycle of the corresponding output.

Figure 6-9. Output Modulation with POC register



7. PSC Interrupts

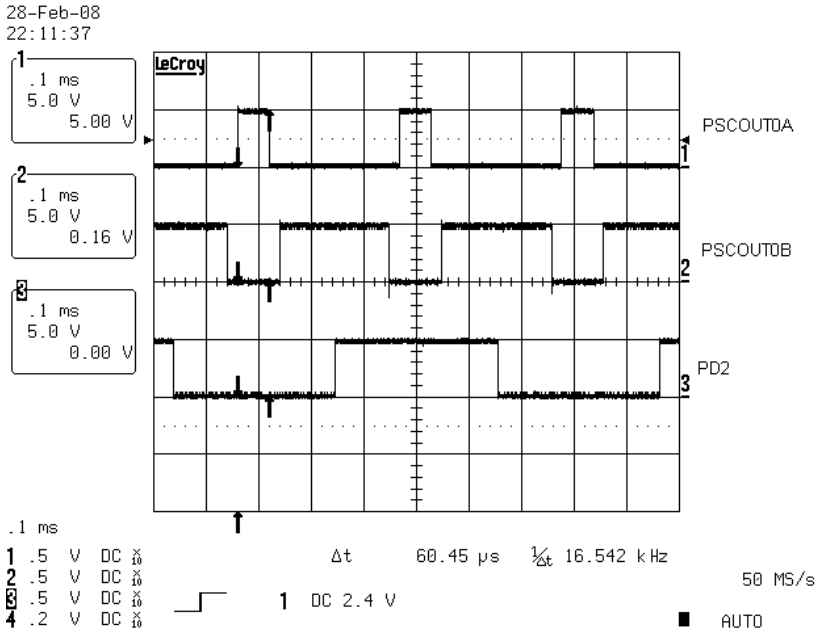
The PSC can generate interrupts when a significant event occurs on PSC inputs or at the end of the PSC cycle. This example demonstrates the use of the interrupt at the end of PSC cycle.

The name of the AVR project is Test_Interrupt.aps.

The PSC end of cycle interrupt is enabled and the interrupt routine makes toggle the PD2 pin.

The PSC uses the centered mode and the end of cycle is located in the middle of the waveforms.,

Figure 7-1. PSC interrupt toggles PD2 pin.



Headquarters

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

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
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
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
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.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.