AVR434: PSC Cookbook

8-bit AVR® Microcontroller

Application Note

Features

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

1. Introduction

This application note is an introduction to the use of the Power Stage Controllers (PSC) available in some AVR microcontrollers. The object of this document is to give a general overview of the PSC, show their various modes of operation and explain how to configure them. The code examples will make this clearer and can be used as guide for other applications. The examples are developed and tested on AT90PWM3.

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

This application note describes some possible waveforms which can be generated by the PSC and how to program it. It also introduces:

- Re-triggering and fault modes

- Synchronization with other PSC

- Output matrix

Note: All examples are made with the first revision of the AT90PWM3 part. On following revisions, the PSC has been enhanced:

- in centered mode, polarity change of the low-side PSC output for power bridge direct drive,
- added software triggering capture (see PICRnH/L registers in datasheet),
- added status bit to monitor the output activity by software (see PIFRn registers in datasheet).

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

2. General Description

The Power Stage Controllers are peripherals provided to drive the power stages of an equipment or a board. Each PSC is logical level compatible and is able to drive a half bridge of power transistors.

They can work in a mutual independent way or synchronized:

- When they are independent, the PSC can drive independent functions such as Power Factor Corrector, DC/DC converter, Power Inverter.
- When they are synchronized the PSC can drive dependent bridges (DC, BLDC, AC motors ...)

Rev. 7670A–AVR–10/06

Each PSC 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 2 inputs which can stop or retrigger waveforms. For example, in a current sensing mode, the current can be monitored by a comparator which can retrigger the PSC waveform when a maximum current is reach.

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.

3. PSC Applications

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

- Lighting Control (Power Factor Correction and Lamp Control)
- Motor Control (Waveform Generation and Speed/Torque regulation)
- Led Lighting Control (Current Regulation)
- Power Supply (PFC and Output Voltage regulation)

PSC Synchronization

The PSC provide signals for PSC synchronization. The PSC can be independent for independent functions (see Figure 3-1) or synchronized for generating consistent waveforms (see Figure 3-2).

Figure 3-1. Example with three independent PSC (HID ballast...)

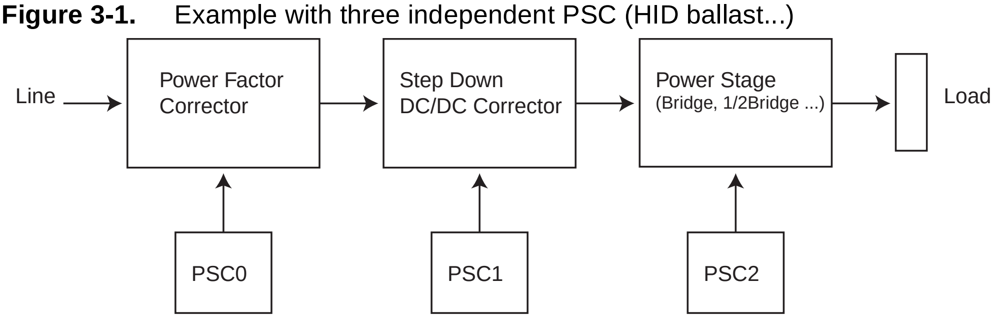
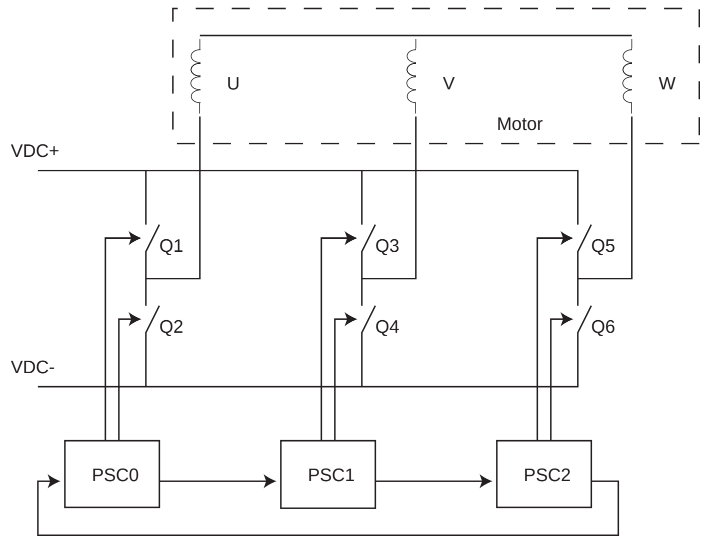


Figure 3-2. Example with three synchronized PSC (motor control ...)



Note: Motor can be BLDC or asynchronous motor

4. PSC Modes

4.1 Prerequisite

We recommend reading the following chapters of the AT90PWM3 datasheet:

- Overview
- PSC Description
- Functional Description

4.2 Why Different Modes

The PSC provides 4 modes of running:

4 ramp mode	First natural use of PSC. Each PSC outputs have 2 specific SFR to describe On-Time and Dead-Time. In this mode OCRnRAH/L and OCRnRBH/L SFR do not include Dead-Time of waveforms which are given by OCRnSAH/L and OCRnSBH/L. Overlapped waveforms are not possible, so there is no risk of cross conduction.
2 ramp mode	This mode is similar to 4 ramp mode. The main difference is that OCRnRAH/L and OCRnRBH/L are the sum of On-Time and Dead-Time for each output. OCRnSAH/L and OCRnSBH/L are Dead-Times. Overlapped waveforms are not possible, so there is no risk of cross conduction.
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 symetrical and centered. This mode is useful for space vector pwm methods to generate sinusoidal waveforms. (Overlapped waveforms are possible)

The choice between "4 ramp mode" and "2 ramp mode" depends on the architecture of the application and the number of registers which must be written at each update or during interrupt routines. So it is important to analyse the differences if optimization is necessary.

4.3 Running Modes Examples

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

Thanks to E_N_RAMPS, we can select the running mode.

PSC initialization for waveform generation with one of the 4 running modes

C Code Example

```
* E_N_RAMPS to init PSC 0 ramp number 4,2,1,0(centered) */
#define E_N_RAMPS 0

/* E_OVERLAPPED to show an example with overlapped values in one ramp mode
*/
//#define E_OVERLAPPED

#ifdef E_OVERLAPPED
#define DEAD_TIME_0 50
#define ON_TIME_0 100
#define DEAD_TIME_1 75
#define ON_TIME_1 125
#else
#define DEAD_TIME_0 50
#define ON_TIME_0 75
#define DEAD_TIME_1 100
#define ON_TIME_1 125
#endif

/*F**************************************************************************
* NAME: PSC0 Init
***************************************************************************/
void PSC0_Init (void)
{
  PSOC0 = (1<<POEN0A)|(1<<POEN0B);
OCR0SAH = HIGH(DEAD_TIME_0);
  OCR0SAL = LOW(DEAD_TIME_0);
  OCR0RAH = HIGH(ON_TIME_0);
  OCR0RAL = LOW(ON_TIME_0);
  OCR0SBH = HIGH(DEAD_TIME_1);
  OCR0SBL = LOW(DEAD_TIME_1);
  OCR0RBH = HIGH(ON_TIME_1);
  OCR0RBL = LOW(ON_TIME_1);
```

C Code Example

```
#if (E_N_RAMPS == 4)
  PCNF0 = (1<<PMODE01)|(1<<PCLKSEL0)|(1<<POP0); /* four ramps */
#else
#if (E_N_RAMPS == 2)
  PCNF0 = (1<<PMODE00)|(1<<POP0); /* two ramps */
#else
#if (E_N_RAMPS == 1)
  PCNF0 = (0<<PMODE01)|(0<<PMODE00)|(1<<POP0); /* one ramp */
#else
  PCNF0 = (1<<PMODE01)|(1<<PMODE00)|(1<<POP0); /* centered */
#endif
#endif
#endif

PFRC0A = 0;
PFRC0B = 0;
  PCTL0 = (1<<PRUN0); /* RUN !! */
}
```

4.3.1 Mode: 4 ramps

To select the 4 ramp mode use the following syntax:

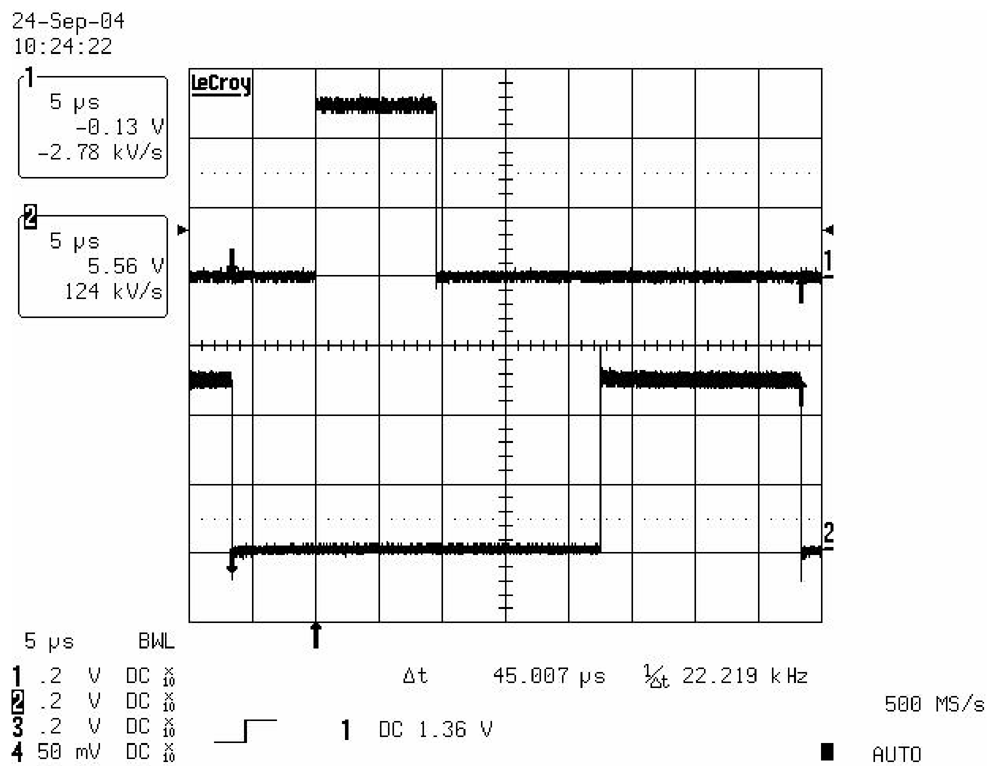
#define E_N_RAMPS 4

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

Table 4-1. Settings for Figure 4-1.

PSC SFR	Instruction	Result in Clock Number	Result in µS
OCRSAH/L	DEAD_TIME_0 = 50	Dead Time 0 = 50 + 2	Dead Time 0 = 6.5µS
OCRRAH/L	ON_TIME_0 = 75	On Time 0 = 75	On Time 0 = 9.375µS
OCRSBH/L	DEAD_TIME_1 = 100	Dead Time 1 = 100 + 2	Dead Time 1 = 12.75µS
OCRRBH/L	ON_TIME_1 = 125	On Time 1 = 125	On Time 1 = 15.625µS

Figure 4-1. Four Ramps



4.3.2 Mode: 2 ramps

To select the 2 ramp mode use the following syntax:

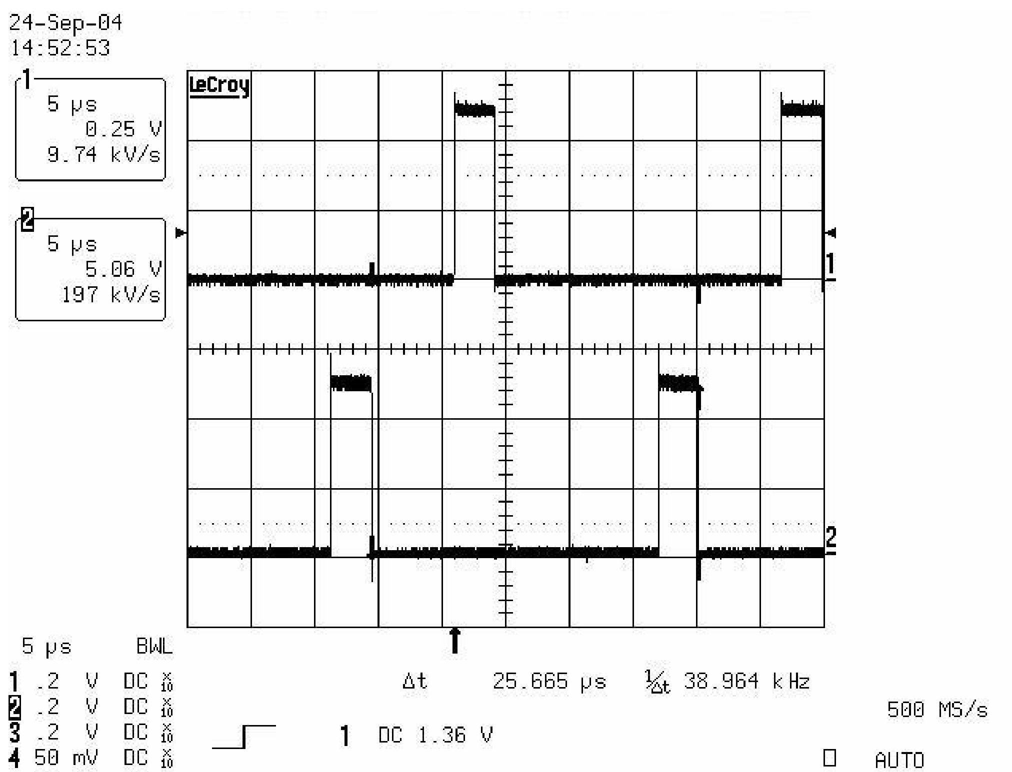
#define E_N_RAMPS 2

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

Table 4-2. Settings for Figure 4-2.

PSC SFR	Instruction	Result in Clock Number	Result in µS
OCRSAH/L	DEAD_TIME_0 = 50	Dead Time 0 = 50 + 1	Dead Time 0 = 6.375µS
OCRRAH/L	ON_TIME_0 = 75	On Time 0 = 75 - 50	On Time 0 = 3.125µS
OCRSBH/L	DEAD_TIME_1 = 100	Dead Time 1 = 100 + 1	Dead Time 1 = 12.625µS
OCRRBH/L	ON_TIME_1 = 125	On Time 1 = 125 - 100	On Time 1 = 3.125µS

Figure 4-2. Two Ramps



4.3.3 Mode: 1 ramp

To select the 1 ramp mode use the following syntax:

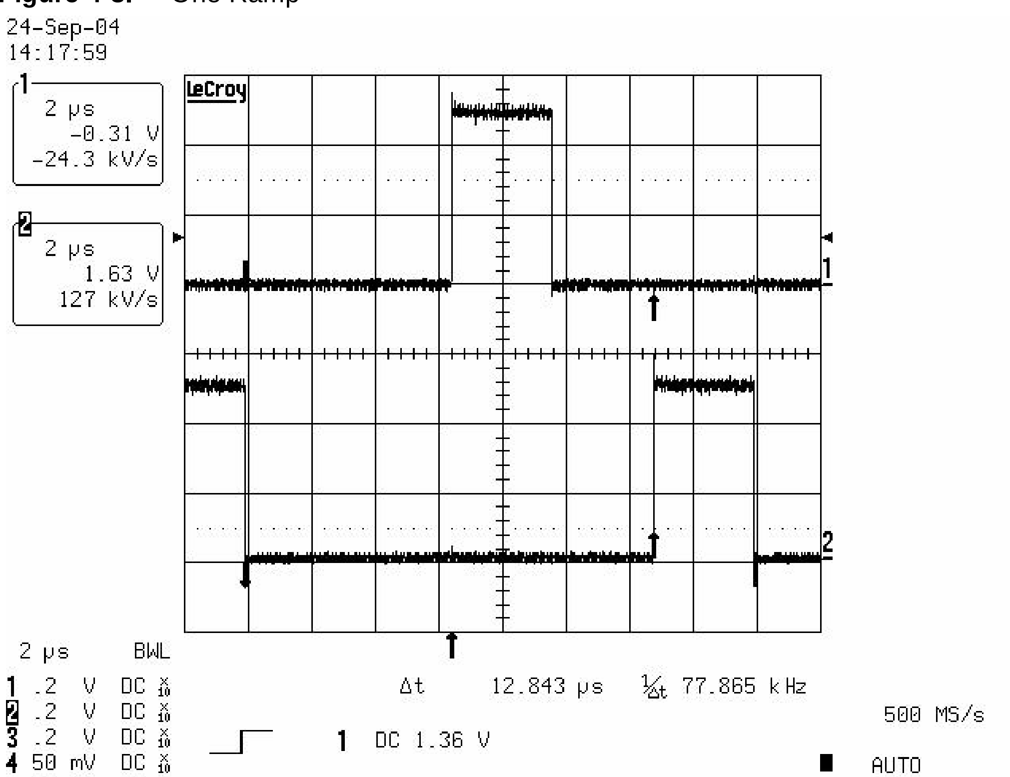
#define E_N_RAMPS 1

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

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

PSC SFR	Instruction	Result in Clock Number	Result in µS
OCRSAH/L	DEAD_TIME_0 = 50	Dead Time 0 = 50 + 1	Dead Time 0 = 6.375µS
OCRRAH/L	ON_TIME_0 = 75	On Time 0 = 75 - 50	On Time 0 = 3.125µS
OCRSBH/L	DEAD_TIME_1 = 100	Dead Time 1 = 100 - 75	Dead Time 1 = 3.125µS
OCRRBH/L	ON_TIME_1 = 125	On Time 1 = 125 - 100	On Time 1 = 3.125µS

Figure 4-3. One Ramp



One Ramps With Overlapped Waveforms

To select the 1 ramp mode with overlapped waveforms uncomment the following line:

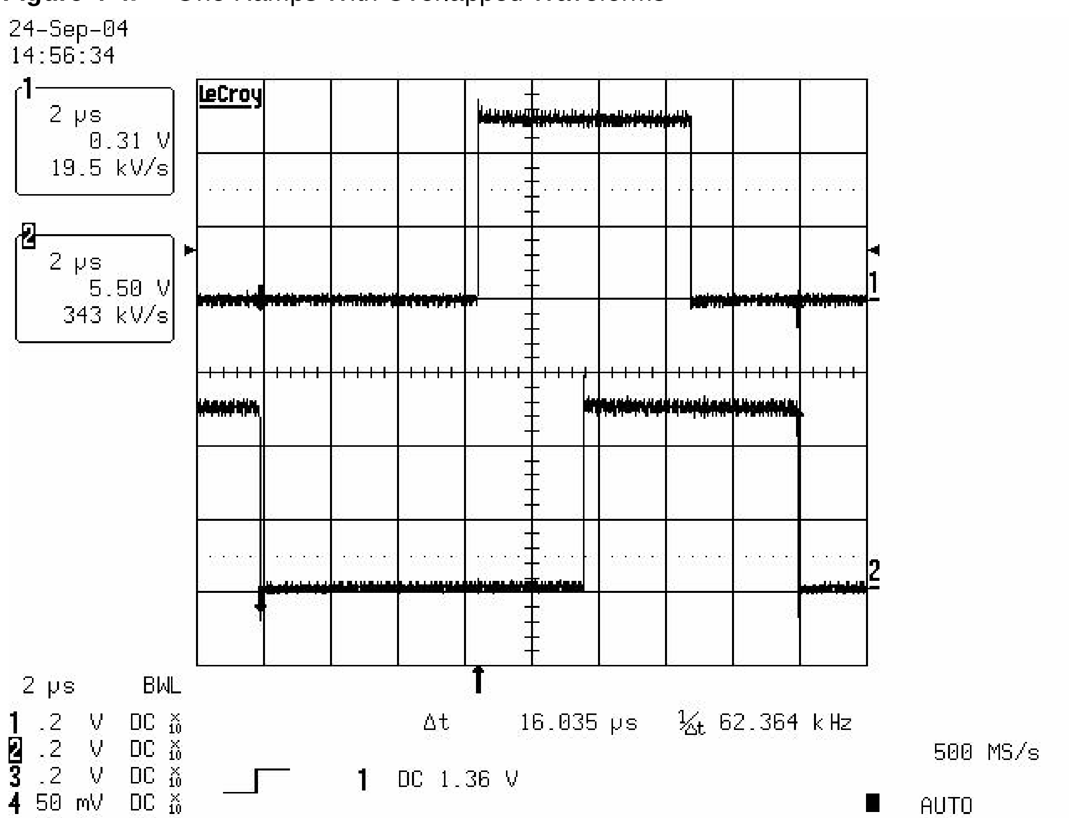
#define E_OVERLAPPED

Table 4-4. Settings for Figure 4-4.

PSC SFR	Instruction	Result in Clock Number	Result in µS
OCRSAH/L	DEAD_TIME_0 = 50	Dead Time 0 = 50 + 1	Dead Time 0 = 6.375µS
OCRRAH/L	ON_TIME_0 = 100	On Time 0 = 100 - 50	On Time 0 = 6.25µS
OCRSBH/L	DEAD_TIME_1 = 75	Dead Time 1 = 75 - 100	Dead Time 1 = -3,125µS (1)
OCRRBH/L	ON_TIME_1 = 125	On Time 1 = 125 - 75	On Time 1 = 6.25µS

Note: 1. The names of variables or constants have been defined for 4 ramp mode. In 1 ramp mode they can be not appropriate.

Figure 4-4. One Ramps With Overlapped Waveforms



4.3.4 Mode: Centered

To select the centered mode use the following syntax:

#define E_N_RAMPS 0

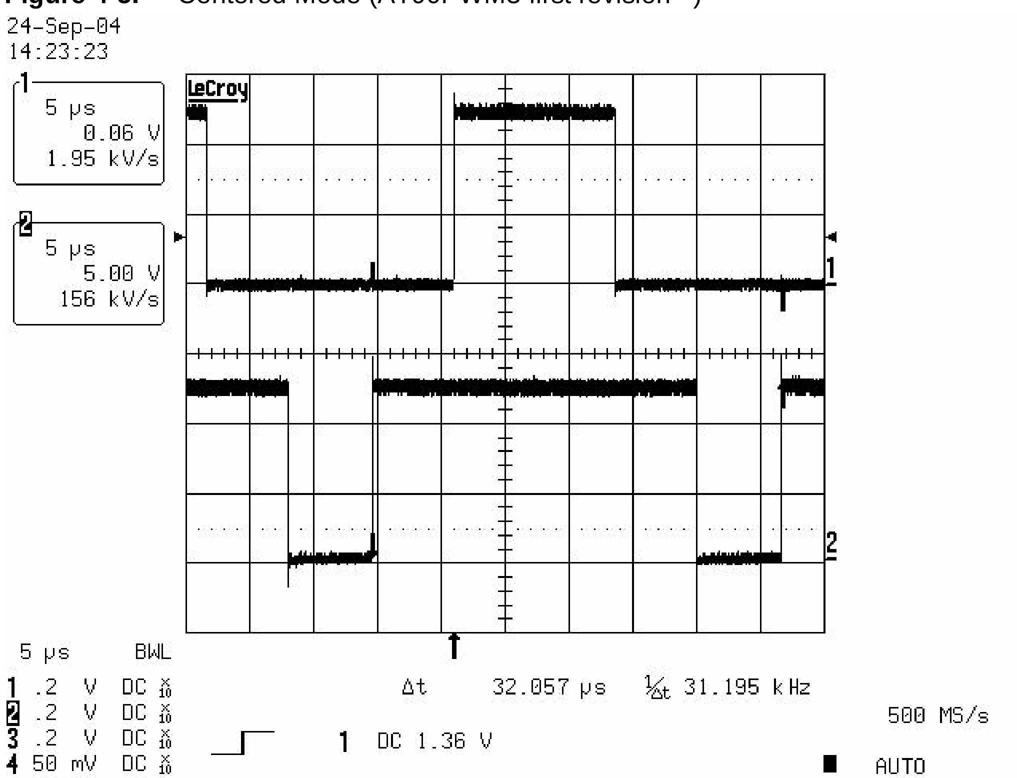
In the following example, CLOCK PSC = CLK IO = 8 Mhz

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

PSC SFR	Instruction	Result in Clock Number	Result in µS
OCRSAH/L	ON_TIME_0 = 50 (1)	On Time 0 = 2 * 50	On Time 0 = 12.5µS
OCRRAH/L	ON_TIME_0 = 75	used for ADC synchronization	-
OCRSBH/L	DEAD_TIME_1 = 100	Dead Time = 100 - 50	Dead Time = 6.25µS
OCRRBH/L	ON_TIME_1 = 125	On Time 1 = 2 * (125 - 100 + 1)	On Time 1 = 6.5µS

Note: 1. The names of variables or constants have been defined for 4 ramp mode. In 1 ramp mode they can be not appropriate.

Figure 4-5. Centered Mode (AT90PWM3 first revision[1])



Note: 1. On following AT90PWM3 revisions, the low-side output (oscilloscope channel 2) has an inverted polarity.

5. Output Matrix Use

5.1 Prerequisite

We recommend reading the following chapters of the AT90PWM3 datasheet:

- PSC2 outputs - Output Matrix

5.2 Description

Without “Output Matrix”, the PSC outputs change in a predefined way according to running mode and output polarity configuration. Thanks to “Output Matrix”, it is possible to program new output values. “Output Matrix” is a kind of lock-up table which replaces the PSC output values by a programmed table.

In the following example the PSC2 is initialized to use the Output Matrix which gives a flexibility in the choice of output values. The figures in this chapter give some examples according POM2_Init_Value. PSC2 is configured in 4 ramp mode, the following examples can be transposed in the other running modes.

In the AT90PWM3 part, only the PSC2 has an output matrix.

C Code Example

```
#define DEAD_TIME_0 1024
#define ON_TIME_0 2048
#define DEAD_TIME_1 1392
#define ON_TIME_1 1791

/*F**************************************************************************
* NAME: PSC2 Init
**************************************************************************/
void PSC2_Init (void)
{
  PSOC2 = (1<<POEN2D)|(1<<POEN2C)|(1<<POEN2B)|(1<<POEN2A);
  POM2 = POM2_Init_Value; /* see following figures */
  OCR2SAH = HIGH(DEAD_TIME_0);
  OCR2SAL = LOW(DEAD_TIME_0);
  OCR2RAH = HIGH(ON_TIME_0);
  OCR2RAL = LOW(ON_TIME_0);
  OCR2SBH = HIGH(DEAD_TIME_1);
  OCR2SBL = LOW(DEAD_TIME_1);
  OCR2RBH = HIGH(ON_TIME_1);
  OCR2RBL = LOW(ON_TIME_1);
  PCNF2 = (1<<PMODE21)|(1<<POP2)|(1<<POME2);
  PFRC2A = 0;
  PFRC2B = 0;
  PCTL2 = (1<<PRUN2); /* RUN !! */
}
```

5.3 Experiments

POM2 contains two nibbles. The most significant nibble concerns PSCOUT21 and the least significant nibble concerns PSCOUT20. High bit of each nibble concerns the Ramp4.

For example, with POM2 = “0101 0001”, “0101” is dedicated to PSCOUT21 which will be equal to 1 during ramp 1, 0 during ramp 2, 1 during ramp 3 and 0 during ramp 4. In the same manner, “0001” is dedicated to PSCOUT20 which will be equal to 1 during ramp 1, 0 during ramp 2, 0 during ramp 3 and 0 during ramp 4

Figure 5-1. POM2_Init_Value = 0101 0001

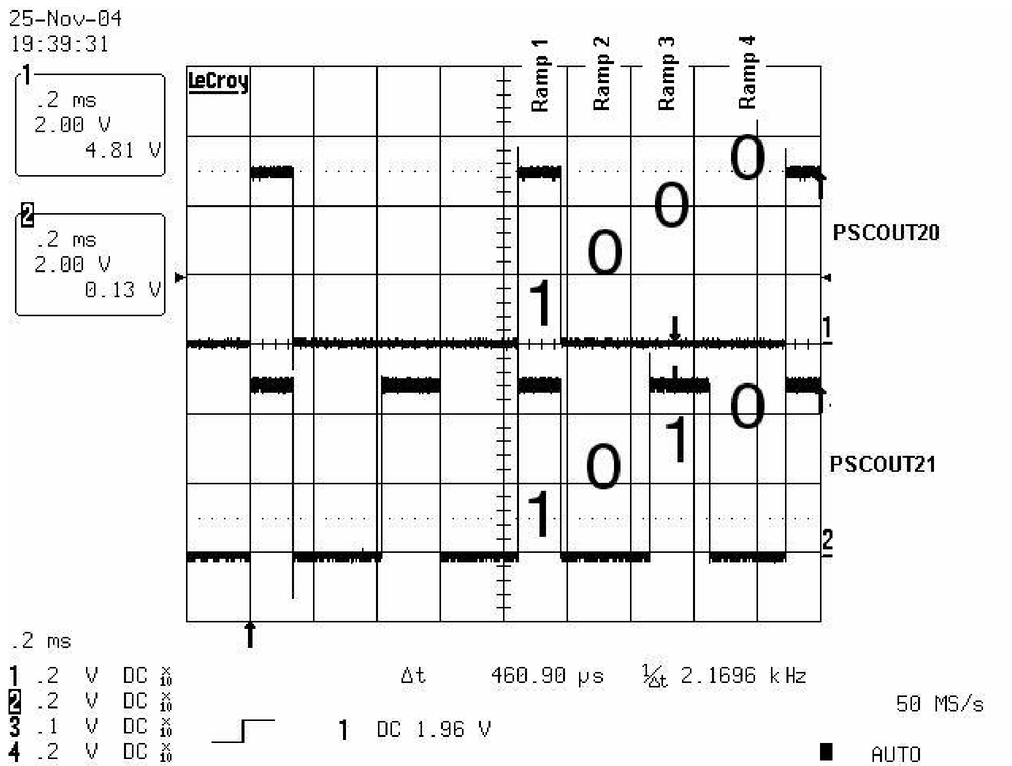


Figure 5-2. POM2_Init_Value = 1001 0001

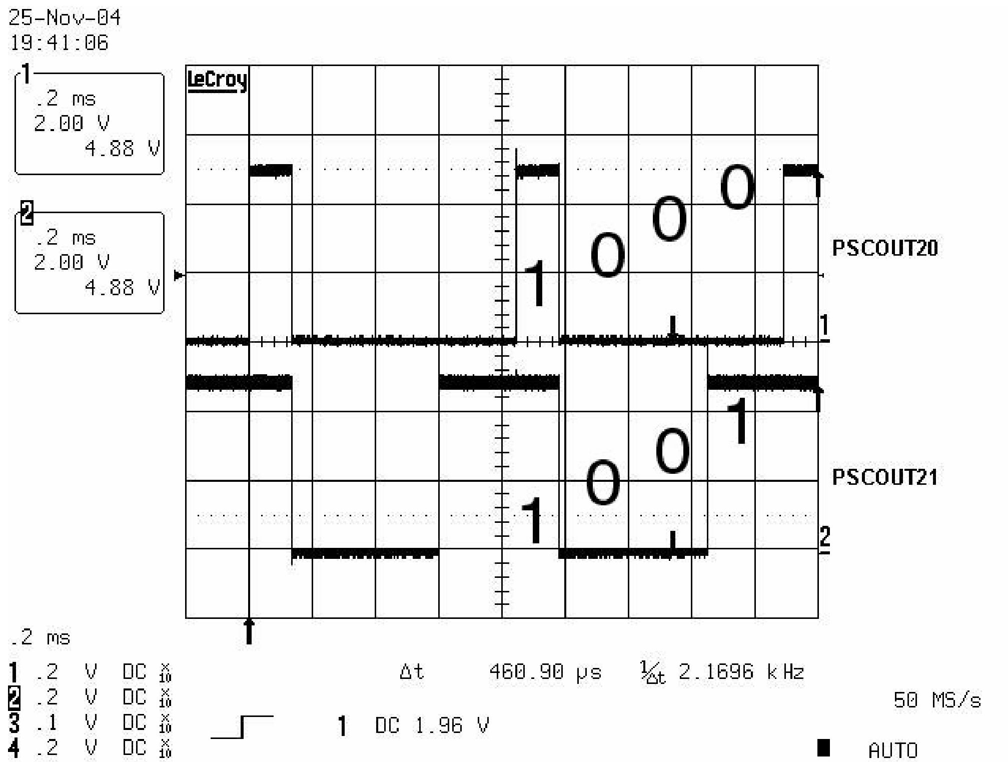
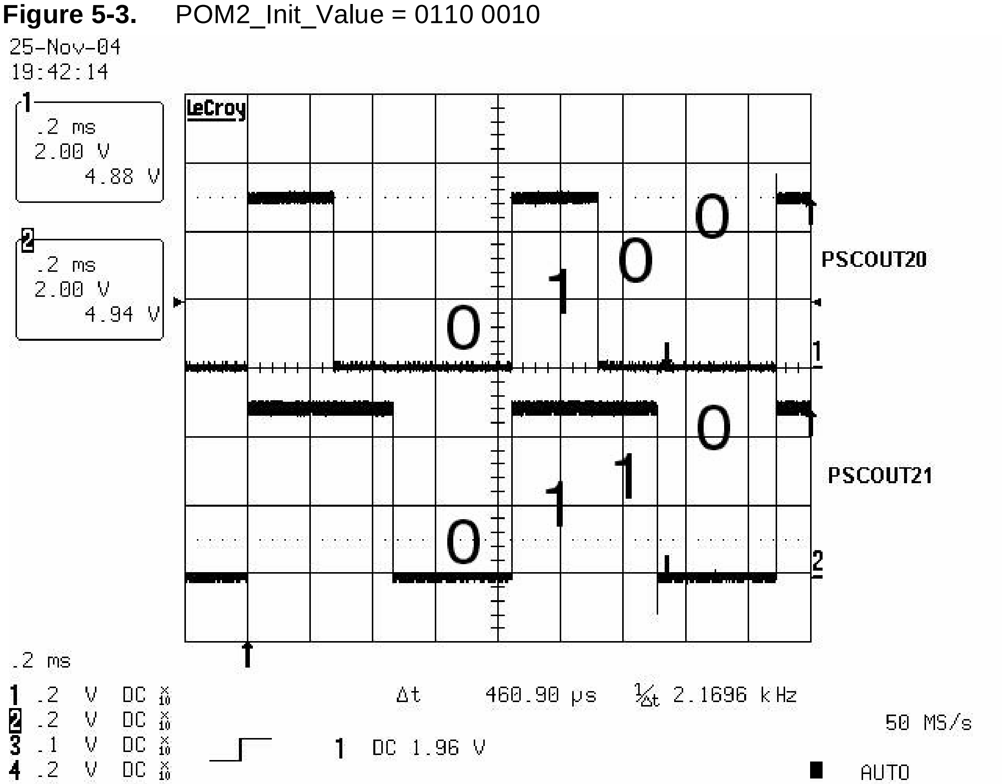


Figure 5-3. POM2_Init_Value = 0110 0010



6. PSCOUT22 & PSCOUT23 Output Selector Use

6.1 Prerequisite

We recommend reading the following chapters of the AT90PWM3 datasheet:

- PSC2 outputs - PSCOUT22 & PSCOUT23 Selectors

6.2 Description

Use the same code example as previous chapter (see "Output Matrix Use" on page 12).

6.3 Experiments

PSC2 is a initialized in such a manner that PSCOUT22 is a copy of PSCOUT20 and PSCOUT23 is a copy of PSCOUT21 (see PSOC2 initialization).

Figure 6-1. POM2_Init_Value = 0110 0010

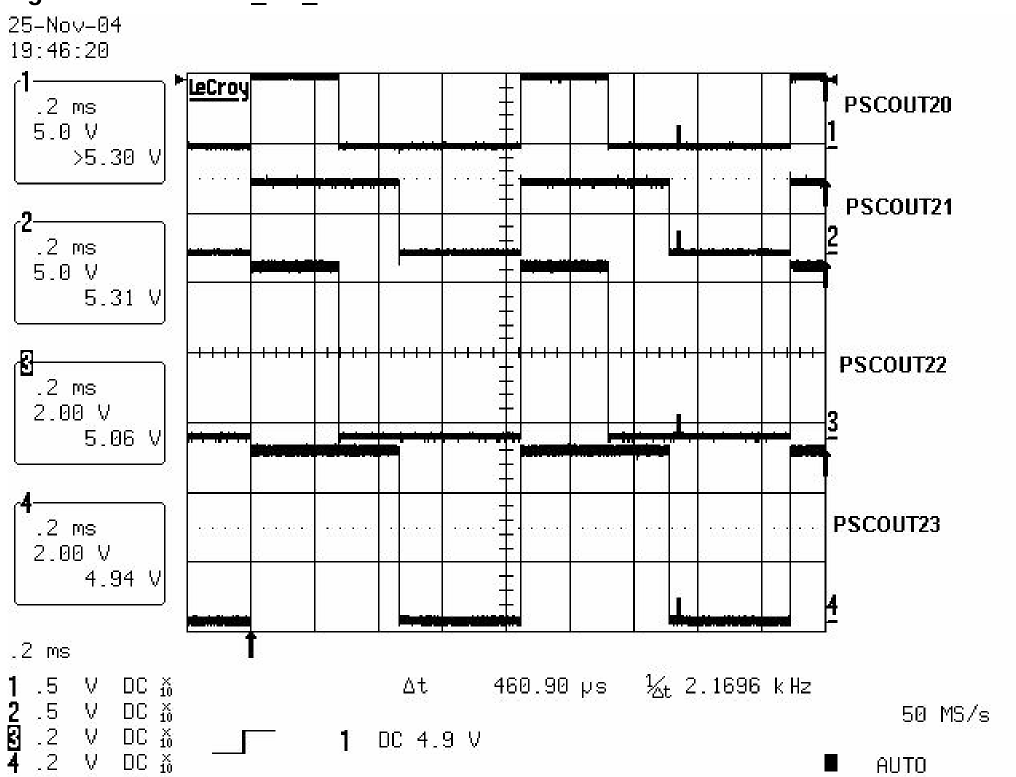
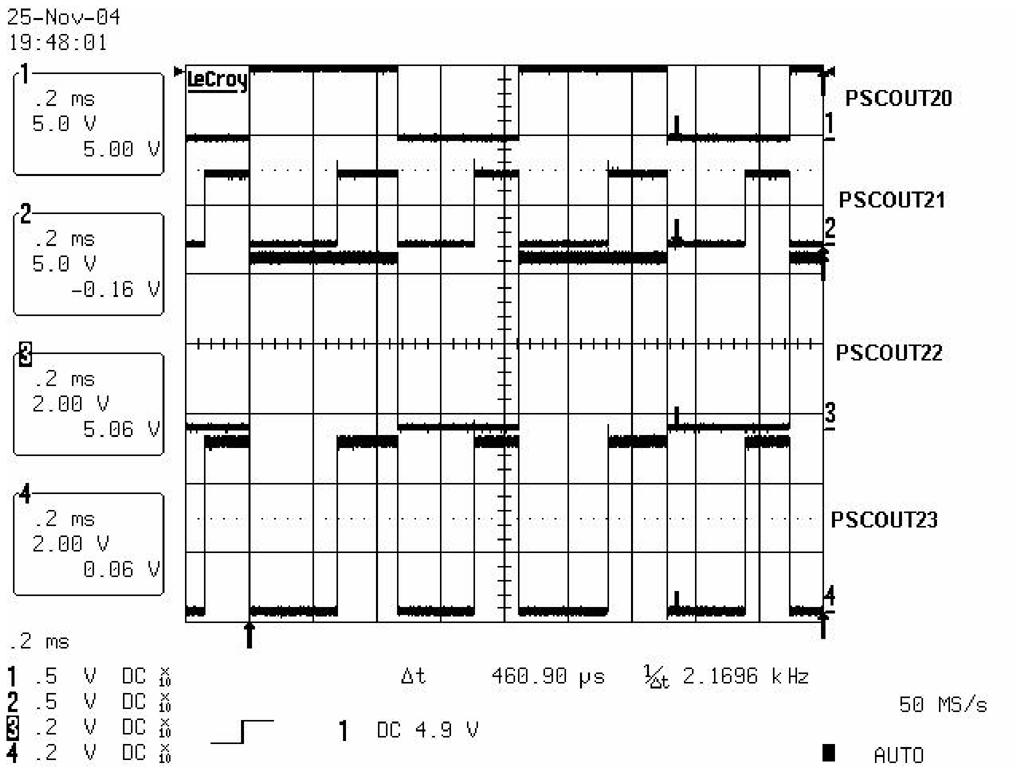


Figure 6-2. POM2_Init_Value = 0101 0110



7. PSC Inputs

7.1 Prerequisite

We recommend reading the following chapters of the AT90PWM3 datasheet:

- Overview
- PSC Description
- PSC Inputs

7.2 Description

These examples are given to quickly start the use of PSC inputs and to evaluate the input modes. They are done in 4 ramp mode. Thanks to E_N_RAMPS definiton, they can easily be modified to test inputs in other running modes.

C Code Example

```
#define DEAD_TIME_0 50
#define ON_TIME_0 75
#define DEAD_TIME_1 100
#define ON_TIME_1 125

/* E_FOUR_RAMPS to init PSC 0 ramp number 4,2,1,0(centered) */
#define E_N_RAMPS 4

/* E_RETRIG_0A to enable fault A input */
#define E_FAULT_0A

/* E_RETRIG_0B to enable fault B input */
//#define E_FAULT_0B
```

C Code Example

```
/*F***************************************************************
* NAME: PSC0_Stop
*****************************************************************/
void PSC0_Stop (void)
{
  PCTL0 = (0<<PRUN0); /* Stop !! */
}
```

```
/*F**************************************************************************
* NAME: PSC0 Init
*****************************************************************************/
void PSC0_Init (unsigned char Fault_Number)
{
  PSOC0 = (1<<POEN0A)|(1<<POEN0B);
  OCR0SAH = HIGH(DEAD_TIME_0);
  OCR0SAL = LOW(DEAD_TIME_0);
  OCR0RAH = HIGH(ON_TIME_0);
  OCR0RAL = LOW(ON_TIME_0);
  OCR0SBH = HIGH(DEAD_TIME_1);
  OCR0SBL = LOW(DEAD_TIME_1);
  OCR0RBH = HIGH(ON_TIME_1);
  OCR0RBL = LOW(ON_TIME_1);

#if (E_N_RAMPS == 4)
  PCNF0 = (1<<PMODE01)|(1<<POP0); /* four ramps */
#else
#if (E_N_RAMPS == 2)
  PCNF0 = (1<<PMODE00)|(1<<POP0); /* two ramps */
#else
#if (E_N_RAMPS == 1)
  PCNF0 = (0<<PMODE01)|(0<<PMODE00)|(1<<POP0); /* one ramp */
#else
  PCNF0 = (1<<PMODE01)|(1<<PMODE00)|(1<<POP0); /* centered */
#endif
#endif
#endif

#ifdef E_FAULT_0A
  PFRC0A = (1<<PELEV0A)|(1<<PFLTE0A)|(0<<PRFM0A3)|(Fault_Number&0x07);
#else
  PFRC0A = 0;
#endif

/* set PISEL0B to select PSC0IN1 input */
#ifdef E_FAULT_0B
  PFRC0B =
(0<<PISEL0B)|(1<<PELEV0B)|(1<<PFLTE0B)|(0<<PRFM0B3)|(Fault_Number&0x07);
#else
  PFRC0B = 0;
#endif

  PCTL0 = (1<<PRUN0); /* RUN !! */
}
```

C Code Example

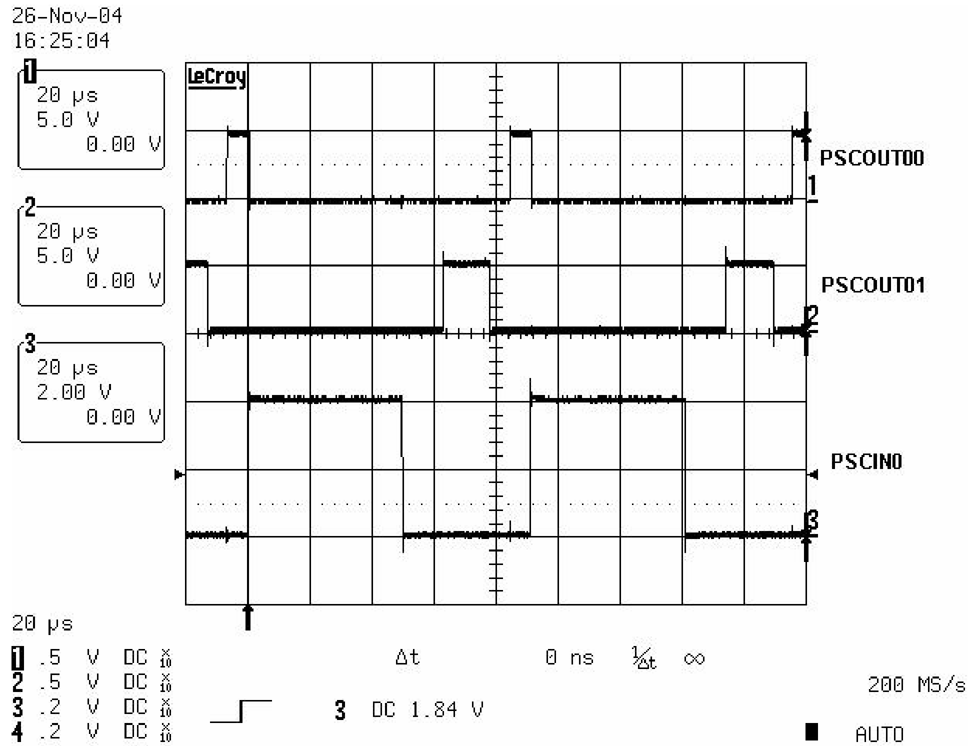
```
/*F*****************************************************************
***********
* NAME: main
******************************************************************
***********/
int main (void)
{
  unsigned char Fault = 0;

  PSC0_Init(0);
  PORTD = (1<<PORTD1); /* pull up on PSCIN0 pin */
  PORTC = 0xFF;
  DDRC = 0xFF;
  while(1)
  {
    if (bit_is_clear(PINE,PINE1))
    {
      while (bit_is_clear(PINE,PINE1));
      Fault++;
      Fault = Fault & 0x07;
      PORTC = ~(Fault);
      PSC0_Init (Fault);
    }

    if (bit_is_clear(PINE,PINE2))
    {
      while (bit_is_clear(PINE,PINE2));
      PSC0_Stop ();
    }
  }
}
```

This example can be used on a STK500 board. In this case Port E is connected to switches and Port C is connected to leds.

Thanks to SW1, we can increment the PSC input mode number. The PSC input mode number is displayed on the leds.

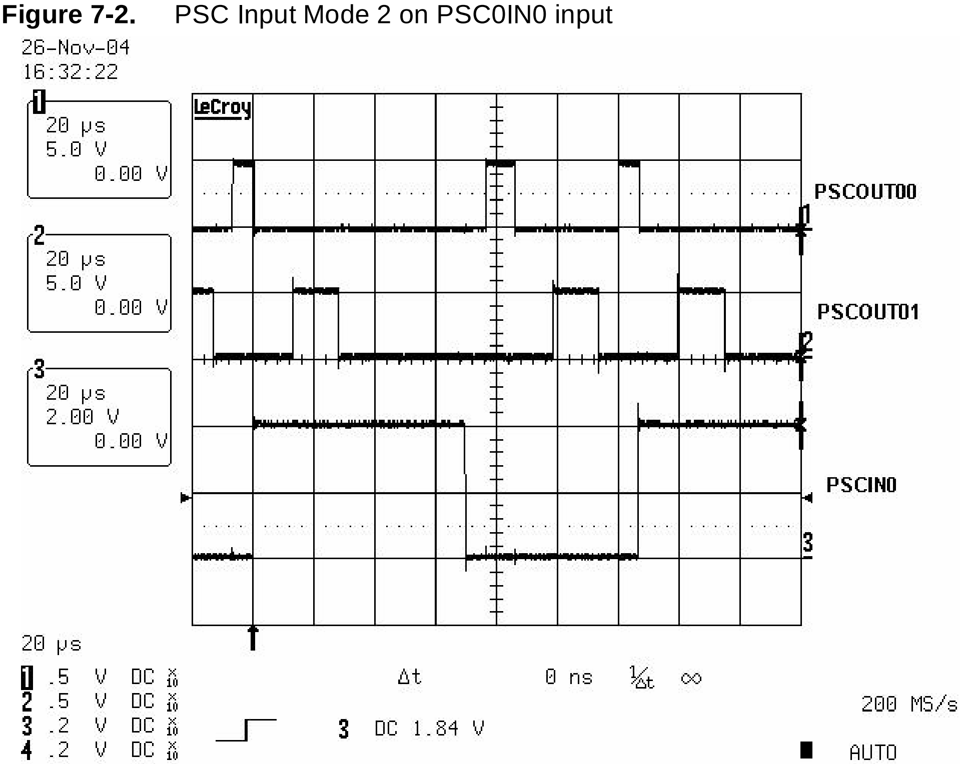
7.2.1 Input Mode 1: Stop signal, Jump to Opposite Dead-Time and Wait

Figure 7-1. PSC Input Mode 1 on PSC0IN0 input



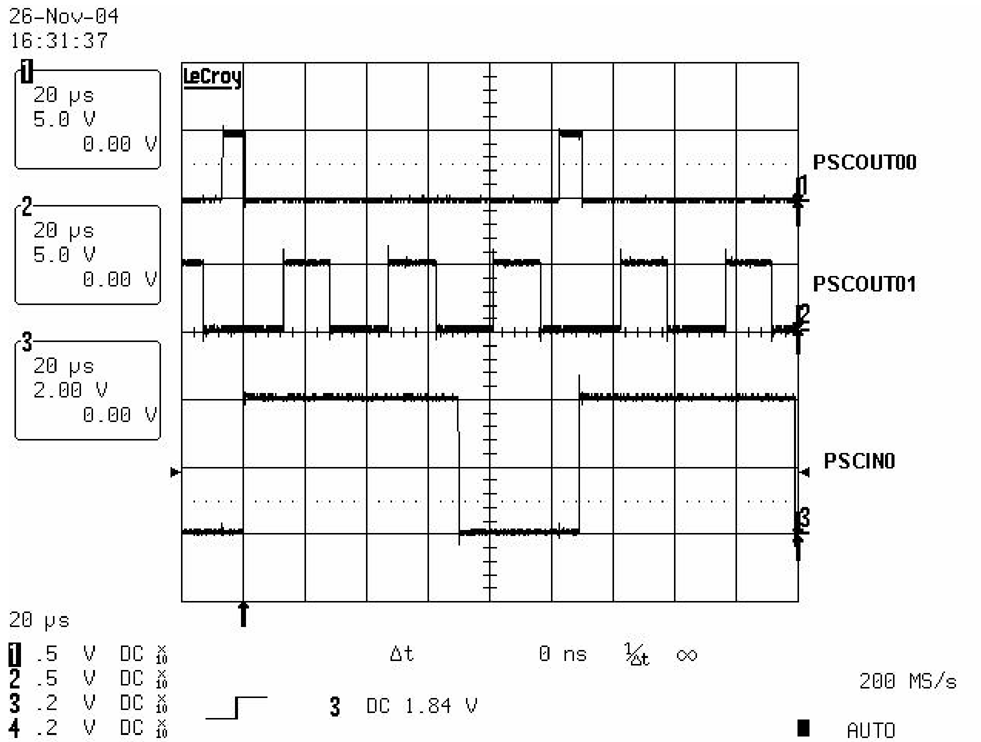
7.2.2 Input Mode 2: Stop signal, Execute Opposite Dead-Time and Wait

Figure 7-2. PSC Input Mode 2 on PSC0IN0 input



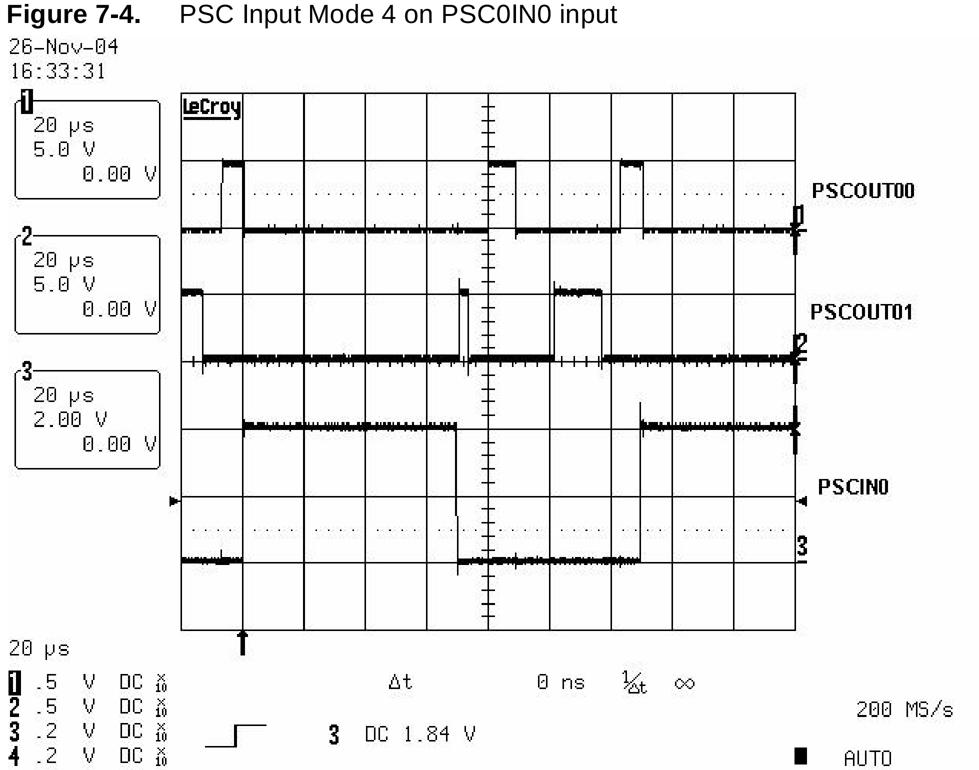
7.2.3 Input Mode 3: Stop signal, Execute Opposite while Fault active

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



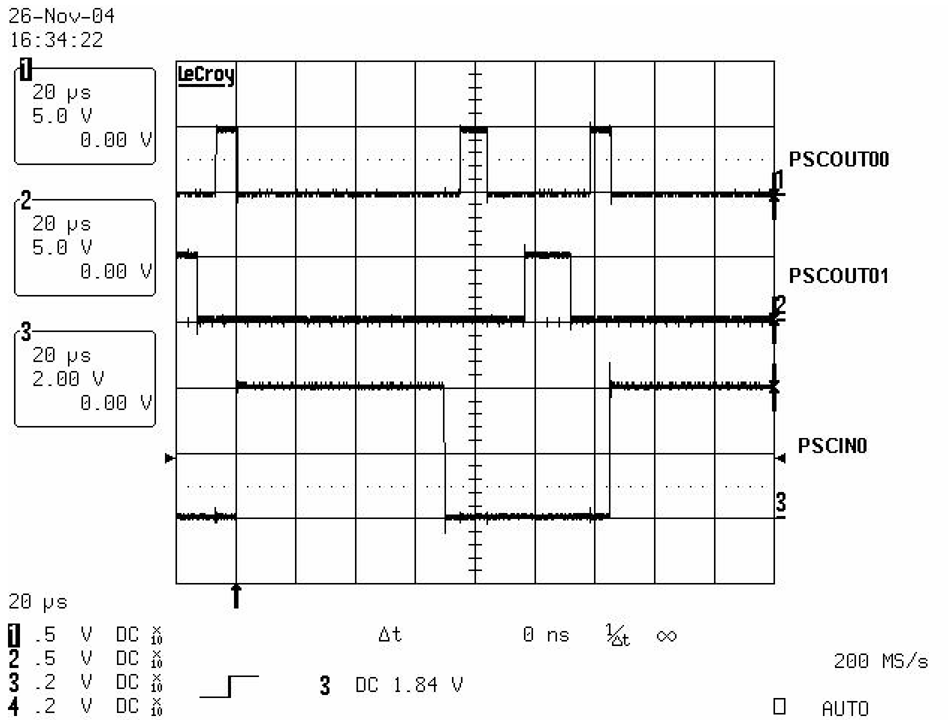
7.2.4 Input Mode 4: Deactivate outputs without changing timing

Figure 7-4. PSC Input Mode 4 on PSC0IN0 input



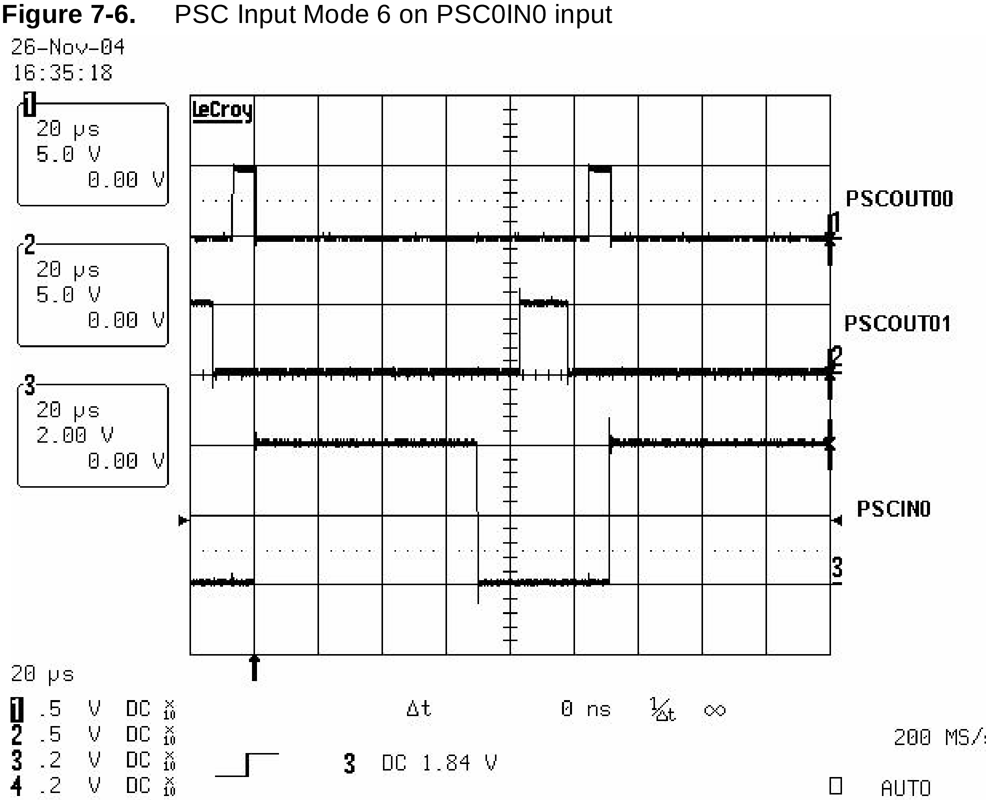
7.2.5 Input Mode 5: Stop signal and Insert Dead-Time

Figure 7-5. PSC Input Mode 5 on PSC0IN0 input



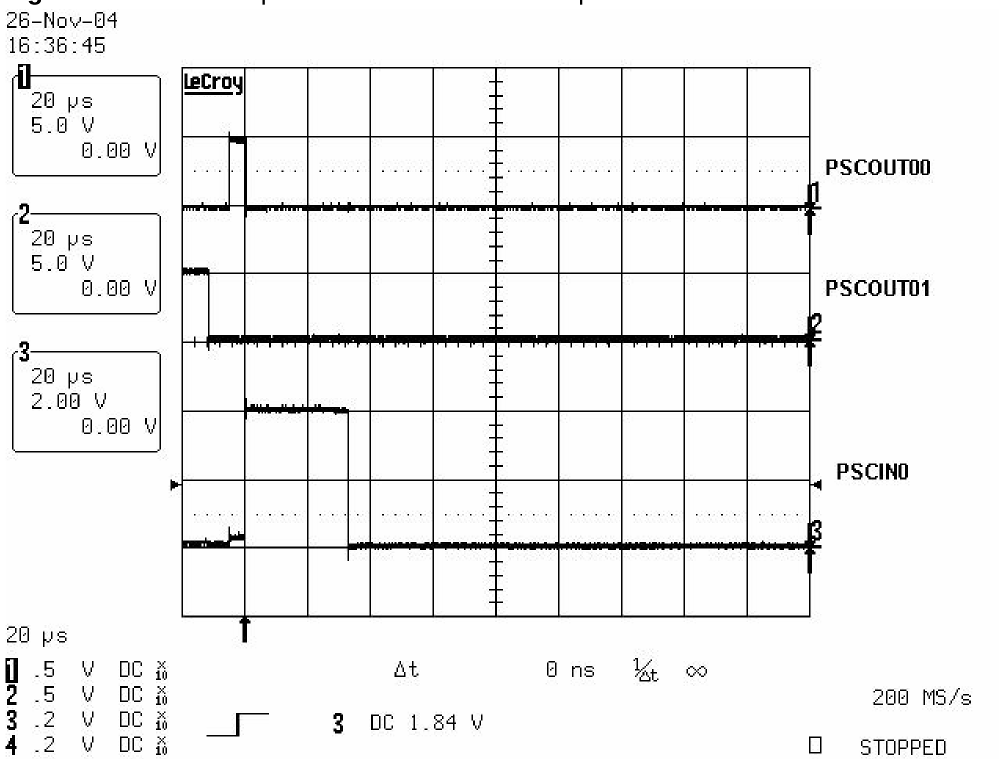
7.2.6 Input Mode 6: Stop signal, Jump to opposite Dead-Time and Wait

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



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

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



7.2.8 Input Mode 8: Edge Retrigger PSC

PSC0 is used in 4 ramps mode

C Code Example

```
/* init of PSC0 */
PSOC0 = (1<<POEN0A)|(1<<POEN0B);
PCNF0 = (1<<PMODE01)|(1<<POP0); /* four ramps */
PFRC0A = (1<<PRTGE0A)|(1<<PELEV0A)|(1<<PFLTE0A);

DEAD_TIME_0 = 50
ON_TIME_0 = 1000
DEAD_TIME_1 = 75
ON_TIME_1 = 1250
```

Figure 7-8. Retrigger Occurs During On Time 1

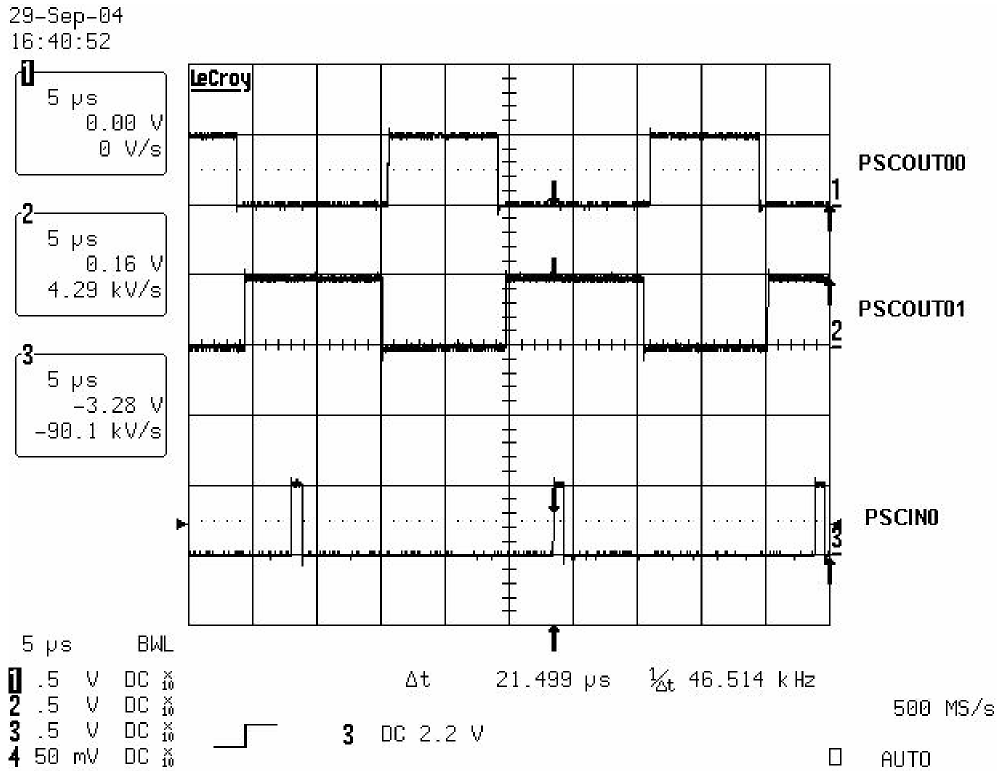


Figure 7-9. Retrigger Occurs During OnTime0

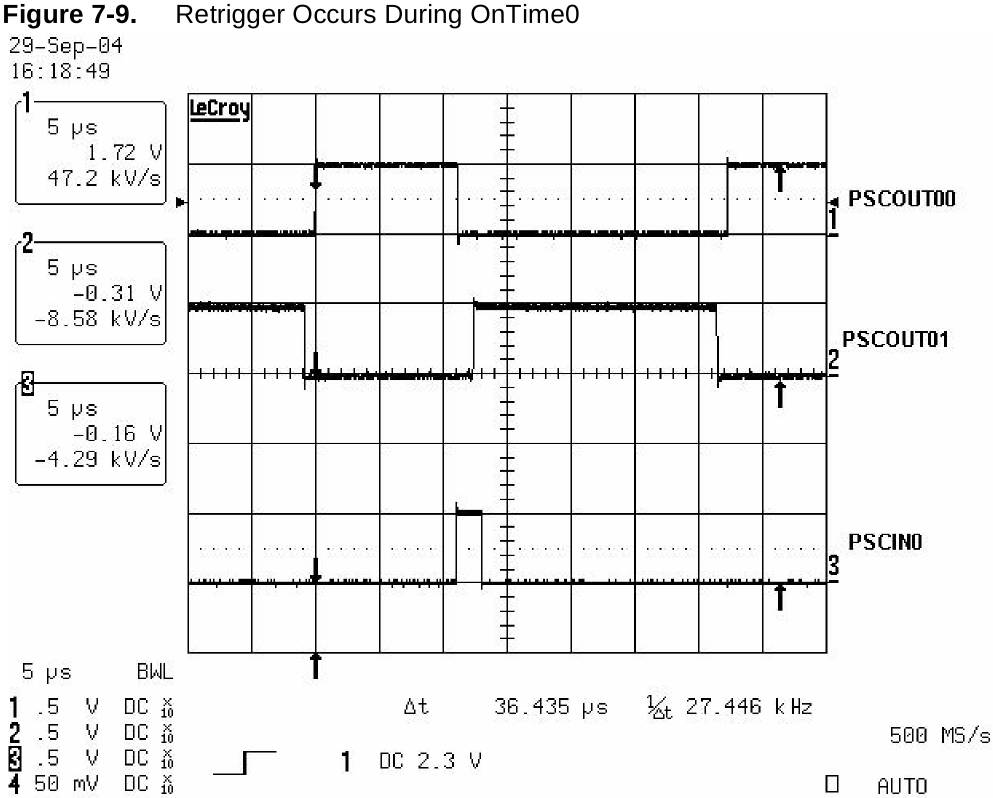
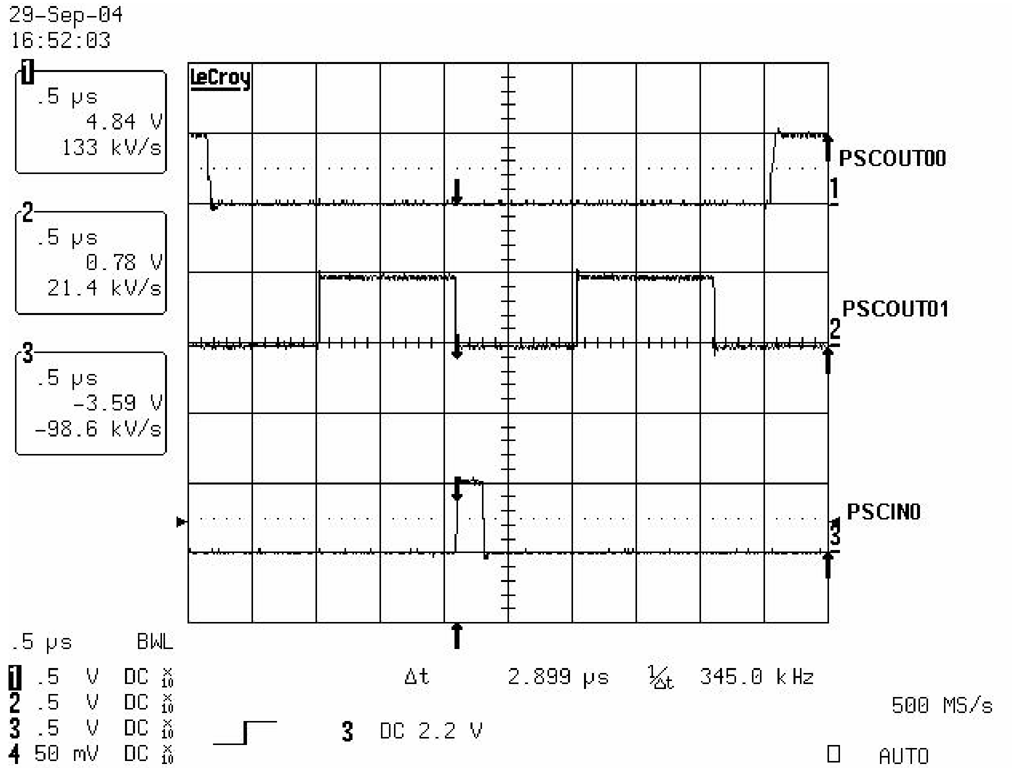


Figure 7-10. Retrigger Occurs During DeadTime0



7.2.9 Fast Timing (High Frequency Pulses)

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

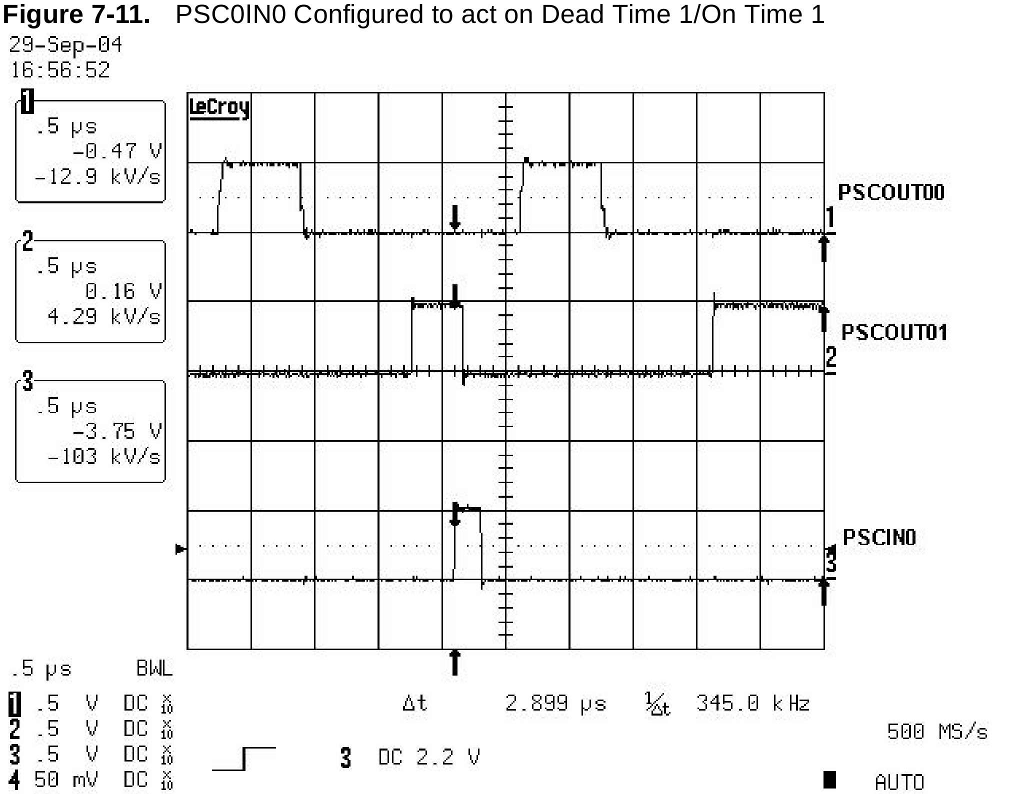
C Code Example

```
PSOC0 = (1<<POEN0A)|(1<<POEN0B);
PCNF0 = (1<<PMODE01)|(1<<PCLKSEL0)|(1<<POP0); /* four ramps */
PFRC0B = (1<<PRTGE0B)|(0<<PISEL0B)|(1<<PELEV0B)|(1<<PFLTE0B);

DEAD_TIME_0 = 30
ON_TIME_0 = 40
DEAD_TIME_1 = 50
ON_TIME_1 = 60
```

Table 7-1. Settings for Figure 7-11.

PSC SFR	Instruction	Result in Clock Number	Result in µS
OCRSAH/L	DEAD_TIME_0 = 30	Dead Time 0 = 30 + 2	Dead Time 0 = 0.5µS
OCRRAH/L	ON_TIME_0 = 40	On Time 0 = 40	On Time 0 = 0.62µS
OCRSBH/L	DEAD_TIME_1 = 50	Dead Time 1 = 50 + 2	Dead Time 1 = 0.81µS
OCRRBH/L	ON_TIME_1 = 60	On Time 1 = 60	On Time 1 = 0.93µS

Figure 7-11. PSC0IN0 Configured to act on Dead Time 1/On Time 1



The propagation delay is less than 100nS.

This propagation delay includes the propagation delay in the digital filter. This delay is 4*Clock cycles, which correspond to 62.5nS. In the following revision of the AT90PWM3 part, this filter can be bypassed (see PFRCnA and PFRCnB SFR in the datasheet).

7.2.10 Use of the Comparator (ACMP0)

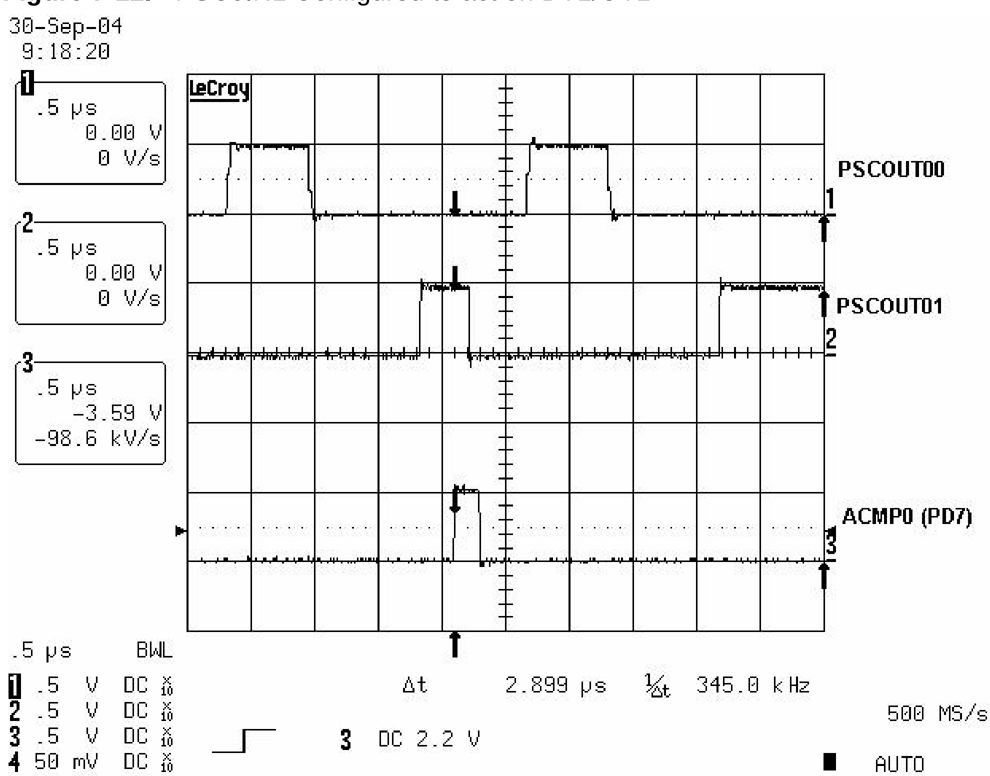
This example demonstrates the use of comparator 0 as retrigger input.

The PSC is in 4 ramp mode.

C Code Example

```
PSOC0 = (1<<POEN0A)|(1<<POEN0B);
PCNF0 = (1<<PMODE01)|(1<<PCLKSEL0)|(1<<POP0); /* four ramps */
PFRC0B = (1<<PRTGE0B)|(1<<PISEL0B)|(1<<PELEV0B)|(1<<PFLTE0B);

DEAD_TIME_0 = 30
ON_TIME_0 = 40
DEAD_TIME_1 = 50
ON_TIME_1 = 60
```

Figure 7-12. PSC0IN1 Configured to act on DT1/OT1



The propagation delay is less than 150nS.

This propagation delay includes the propagation delay in the digital filter. This delay is 4*Clock cycles, which correspond to 62.5nS. In the following revision of the AT90PWM3 part, this filter can be bypassed (see PFRCnA and PFRCnB SFR in the datasheet).

8. Autorun

8.1 Prerequisite

We recommend reading the following chapters of the AT90PWM3 datasheet:

- Overview
- PSC Description
- PSC Synchronization

8.2 Description

Thanks to Autorun, we can generate 3 pairs of centered waveforms. The purpose of this can be the generation of sinusoidal waveform for motor control.

The following example uses PSC0, PSC1 & PSC2. The 3 PSC are in Centered Mode. PSC0, PSC1 and PSC2 are synchronized thanks to PARUNn bits. PSC0 and PSC1 are slaves, PSC2 is master and starts all timings with its PRUN2 bit.

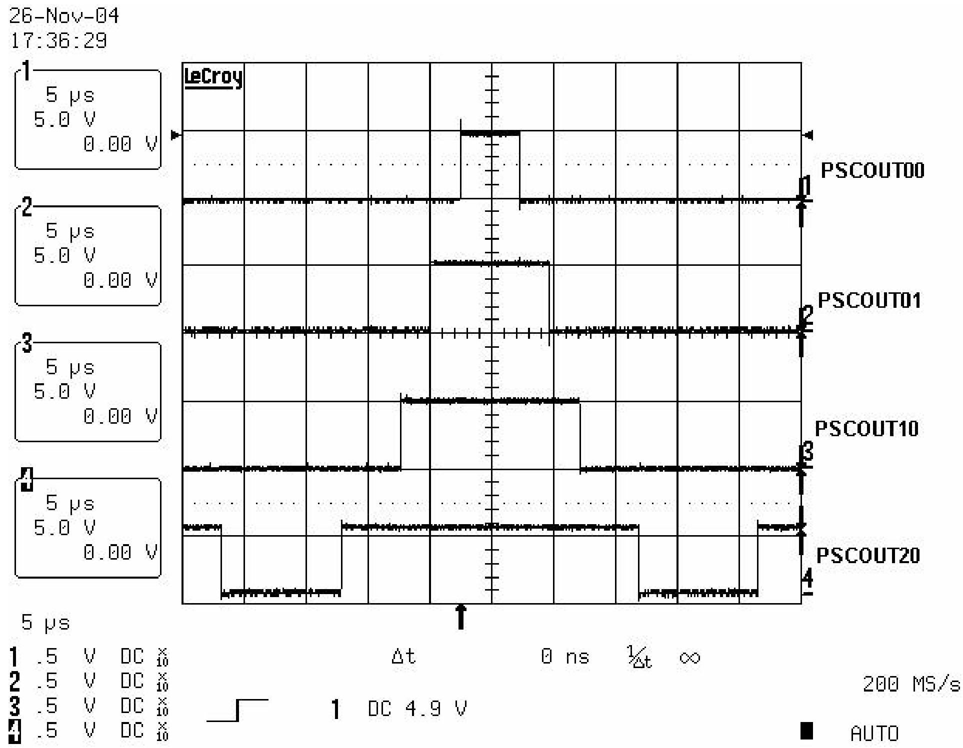
The available outputs are: PSCOUT00, PSCOUT01, PSCOUT10, PSCOUT11, PSCOUT20 & PSCOUT21. PSCOUT11 and PSCOUT21 are not displayed on Figure 8-2.

In this example, the PSC clock comes from the PLL 64MHz output.

Figure 8-1. Settings for Figure 8-2.

	DT0 **OCRnSA**	**OT0** **OCRnRA**	**DT1** **OCRnSB**	**OT1** **OCRnRB**
PSC0	157 4.9µS **(PSCOUT00)**	3 Not Used	314 9.8µS **(PSCOUT01)**	1099 34µS **(Period)**
PSC 1	471 14.7µS **(PSCOUT10)**	3 Not Used	628 19.6µS *(PSCOUT11)*	1099 34µS (Period)
PSC 2	785 24.5µS **(PSCOUT20)**	3 Not Used	942 29.4µS *(PSCOUT21)*	1099 34µS (Period)

Figure 8-2. 3 PSC Synchronized Centered Waveforms



9. Generation of a Fixed Duty Cycle or a Fixed Frequency

With the following code, we can generate a waveform on PSCOUT00.

The frequency of the signal can be modified with the PSC_update(..) function without changing the duty cycle.

In this example, the minimum output frequency min. is 8 kHz and the maximum output frequency is 125kHz to keep a 256 step duty cycle resolution.

To simplify the division operation, the duty cycle is a value in the range from 0 to 255.

PSC functions

C Code Example

```
/*F**************************************************************
* NAME: PSC0 Init
***************************************************************/
void PSC0_Init (void)
{
  PSOC0 = (1<<POEN0A)|(1<<POEN0B);

  PCNF0 = (0<<PMODE01)|(0<<PMODE00)|(1<<PCLKSEL0)|(1<<POP0); /*
one ramp on PLL */

  PFRC0A = 0;
  PFRC0B = 0;

  PCTL0 = (1<<PRUN0); /* RUN !! */
}

/*F**************************************************************
* NAME: PSC0 Update
*  period max = 4096
*  duty max = 255
***************************************************************/
void PSC0_Update (U16 period,U8 duty)
{
  U16 temp;

  PCNF0 =
(1<<PLOCK0)|(0<<PMODE01)|(0<<PMODE00)|(1<<PCLKSEL0)|(1<<POP0);

  temp = ((U32)period * duty) >> 8;
  OCR0RAH = HIGH(temp);
  OCR0RAL = LOW(temp);

  OCR0RBH = HIGH(period);
  OCR0RBL = LOW(period);
  PCNF0 =
(0<<PLOCK0)|(0<<PMODE01)|(0<<PMODE00)|(1<<PCLKSEL0)|(1<<POP0);

}
```

C Code Example

```
int main (void)
{
  U16 var_period = 4000;
  U8 var_duty = 128;

  Start_pll_32_mega();
  Wait_pll_ready();
  PORTC = 0xFF;
  DDRC = 0xFF;
  PSC0_Init();

  PSC0_Update (var_period,var_duty);
  while(1)
  {
    if (bit_is_clear(PINE,PINE1))
    {
      while (bit_is_clear(PINE,PINE1));
      var_period -= 40;
      if (var_period < 100) var_period = 4000;
      PSC0_Update (var_period,var_duty);
    }

    if (bit_is_clear(PINE,PINE2))
    {
      while (bit_is_clear(PINE,PINE2));
      var_duty += 5;
      if (var_duty > 254) var_duty = 5;
      PSC0_Update (var_period,var_duty);
    }
}
```

PE1 is connected to SW1 and PE2 is connected to SW2 on a STK500 board. So each time we press SW1 switch the frequency is increased and each time we press the SW2 switch the duty cycle changes.

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

7670A–AVR–10/06