NetMedia

Basic Express BX-24
Application Note

Programming Timer1 for Dual Pulse Width Modulation

Timer1 and pulse width modulation

The BX-24 processor includes a built-in timer called Timer1. The timer can be used for several functions, one of which is to generate dual hardware PWM outputs without CPU overhead. Pins OC1A and OC1B are used for outputs, which are pins 26 and 27 respectively. You have a choice of 8-, 9- or 10-bit PWM outputs, as well as a choice of several frequencies.

The timer is capable of operating at 5 discrete tick frequencies ranging from approximately 7.20 kHz to 7.37 MHz. After scaling, the tick frequency translates to PWM pulse rates ranging from 3.52 Hz (for 10-bit outputs) to 14.4 kHz (for 8-bit outputs).

The following is a list of the major registers required for getting access to Timer1:

Type	Name	Description
Byte	TCNT1L	Timer/Counter1 Low Byte
Byte	TCNT1H	Timer/Counter1 High Byte
Byte	TCCR1B	Timer/Counter1 Control Register B
Byte	TCCR1A	Timer/Counter1 Control Register A

See also pages 32 to 38 of file AT90_8535.pdf for more details on Timer1, which is actually called Timer/Counter1. This file is provided in the BasicX installation and documents the Atmel 8535 chip, which is used as the BX-24 processor.

Warning -- Timer1 should only be used where it does not conflict with other system resources, such as InputCapture and OutputCapture, all of which depend on Timer1.

Programming Timer1 for dual PWM

Initialization

This example illustrates how to use Timer1 to generate 8-bit PWM outputs on pins OC1A and OC1B, which are pins 26 and 27 respectively on the BX-24 system. Note that pin 27 is shared with the built-in green LED, which makes it easy to get a visual confirmation of the PWM function on that pin.

The first initialization step is to stop the timer:

```
Register.TCCR1B = 0
```

The next step is to set Timer1 to 8-bit PWM mode (9- and 10-bit modes are similar):

```
Const PWMmode8bit  As Byte = bx0000_0001
Const PWMmode9bit  As Byte = bx0000_0010
Const PWMmode10bit As Byte = bx0000_0011
Const PWMmodeOff   As Byte = bx0000_0000

Register.TCCR1A = PWMmode8bit
```

Now we need to define the duty cycles for each pin, which are loaded in registers 0CR1A and OCR1B. The duty cycle can be an 8, 9 or 10 bit number. In this example, we'll use 8-bit values for 25 % and 75 % duty cycles (64 and 191). Note that high bytes must be written first for these registers:

```
Register.OCR1AH = 0
Register.OCR1AL = 64    ' = 25 %

Register.OCR1BH = 0
Register.OCR1BL = 191   ' = 75 %
```

Starting Timer1

Now we can start the timer by writing one of 5 enumerated values to the timer control register TCCR1B. The allowable values are shown below:

TCCR1B Value	Tick Frequency (Hz)	8-bit PWM Pulse Rate (Hz)
1	7 372 800	14 456
2	921 600	1 807
3	115 200	225.9
4	28 800	56.47
5	7 200	14.12

In this example, we'll us the lowest frequency of 7.2 kHz:

```
Register.TCCR1B = 5
```

Here the pulse rate is equal to the Timer1 frequency divided by 510, which in this case is about 14.1 Hz. For 9-bit outputs, the divisor is 1022, and for 10-bit outputs the divisor is 2046.

The last step is to connect the timer to pins OC1A and OC1B:

```
Const MaskOC1A As Byte = bx1000_0000
Const MaskOC1B As Byte = bx0010_0000

' Initialize both pins to output-low.
Call PutPin(PinOC1A, bxOutputLow)
Call PutPin(PinOC1B, bxOutputLow)

' Enable PWM for both pins.
Register.TCCR1A = Register.TCCR1A Or MaskOC1A
Register.TCCR1A = Register.TCCR1A Or MaskOC1B
```

After this point you can continuously vary the PWM duty cycle by writing new values to registers OCR1A and OCR1B. The PWM outputs are generated in the background.

Example code

Source code for an example program is provided as a separate file. The filename is PWMexample.bas.

© 1998-2001 by NetMedia, Inc. All rights reserved.

Basic Express, BasicX, BX-01, BX-24 and BX-35 are trademarks of NetMedia, Inc.

All other trademarks are the property of their respective owners.

2.00.A