NetMedia

Basic Express
Application Note

Programming Timer1 for Edge Capture

Timer1 and the input capture pin

The BasicX processor includes a built-in timer called Timer1. The timer can be used for several functions, one of which is to measure the time delay until a rising or falling edge occurs on the input capture pin.

Why use Timer1 for edge capture? The main advantage is that the processor does not have to be dedicated to waiting for a pin transition.

By contrast, consider system call RCtime, which performs a similar function. RCtime allows you to set a general purpose I/O pin to input-tristate (high impedance) and measure the time delay to a transition. The problem is that RCtime dedicates the processor to the time measurement. The real time clock, task switching and network traffic are suspended during the function.

Timer1 does not have this problem. While Timer1 is waiting for an edge event, the processor can continue with other tasks.

On the other hand, there are disadvantages to this approach. Timer1 should only be used where it does not conflict with other system resources, such as the Com2 or Com3 serial ports, as well as procedures InputCapture and OutputCapture, all of which depend on Timer1.

You might wonder why we can't use procedure InputCapture for edge capture. There are two problems -- first, InputCapture is really intended for measuring pulse widths, and does not record the time delay to the start of the first transition. Second, the procedure will hang if a transition never occurs.

Timer1 capabilities

We can overcome these problems by using Timer1. The timer is capable of operating at 5 discrete tick frequencies ranging from approximately 7.20 kHz to 7.37 MHz, which means you can use Timer1 to measure time intervals at resolutions ranging from about 136 ns to 139 µs.

The following registers are used for getting access to Timer1 for edge capture:

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
Byte	TIFR	Timer/Counter Interrupt Flag register
Byte	ICR1L	T/C 1 Input Capture Register High Byte
Byte	ICR1H	T/C 1 Input Capture Register High Byte

See also the following PDF files for more details on Timer1, which is actually called Timer/Counter1. This file is provided in the BasicX installation and documents the Atmel chip used as the BasicX processor.

```
BX-01: File AT90S4414_8515.pdf, page 31.

BX-24, BX-35: File AT90S_8535.pdf, page 32.
```

Warning -- Timer1 should only be used where it does not conflict with other system resources, such as the Com2 or Com3 serial ports, as well as InputCapture and OutputCapture, all of which depend on Timer1.

Programming Timer1 for edge capture

Initialization

We need to initialize the timer before using it. The first step is to clear the Timer1 control register TCCR1A. This disconnects Timer1 from output pins OC1A and OC1B, and disables PWM operation:

```
Register.TCCR1A = 0
```

The next step is to stop the timer:

```
Register.TCCR1B = 0
```

Now we need to clear the 2 bytes of the Timer1 counter. The high byte must be written first, followed by the low byte:

```
Register.TCNT1H = 0
Register.TCNT1L = 0
```

We also need to clear the Input Capture Flag 1 (ICF1), which is done by writing 1 to a bit in register TIFR:

```
Const ICF1 As Byte = bx00001000  ' BX-01
Const ICF1 As Byte = bx00100000  ' BX-24, BX-35

Register.TIFR = ICF1
```

The Timer/Counter1 Overflow Flag (TOV1) should be cleared, again by writing to the appropriate bit of TIFR:

```
Const TOV1 As Byte = bx10000000 ' BX-01
Const TOV1 As Byte = bx00000100 ' BX-24, BX-35

Register.TIFR = TOV1
```

The last initialization step is to define whether we're looking for a rising or falling edge. The Input Capture1 Edge Select (ICES1) bit handles this job. ICES should be set to 1 for a rising edge and 0 for a falling edge:

```
Const ICES1 As Byte = bx01000000

Register.TCCR1B = ICES1             ' For rising edge.
```

Starting Timer1

Once Timer1 has been initialized, it is started by writing one of 5 enumerated values to the lower 3 bits of the register TCCR1B. The allowable values are shown below:

TCCR1B Value	Timer Resolution (µs)	Tick Frequency (Hz)	Maximum Time Range (s)
1	0.135 633 7	7 372 800	0.008 889
2	1.085 069	921 600	0.071 11
3	8.680 555	115 200	0.568 9
4	34.722 22	28 800	2.276
5	138.888 9	7 200	9.102

In this example, we'll us the 28.8 kHz tick rate:

```
Register.TCCR1B = Register.TCCR1B + 4
```

Waiting for a transition

Once the timer has been intitialized and started, the program needs to look at the input capture flag ICF1, which is automatically set when the desired pin transition occurs. As soon as the flag is set, the processor also copies the timer's value to registers ICR1L and ICR1H. If the transition doesn't occur before the timer overflows, the overflow flag TOV1 is set.

The following code waits for either an edge capture or overflow, whichever comes first:

```
Dim HasOverflowed As Boolean, TIFRcopy As Byte

HasOverflowed = False
Do
    TIFRcopy = Register.TIFR

    ' Bail out if timer overflows.
    If ((TIFRcopy And TOV1) = TOV1) Then
        HasOverflowed = True
        Exit Do
    End If

Loop Until ((TIFRcopy And ICF1) = ICF1) ' Stop if edge is detected.
```

During this loop, other tasks are allowed to run, including tasks handling network traffic, serial I/O and the real time clock. The processor does not have to be dedicated to looking for a pin transition.

Note that we make a copy of register TIFR rather than test the register directly. This is because we need to check two different bits in the register. By making a copy, we freeze the register's bit pattern, and we don't have to worry about bits changing between the capture and overflow checks.

If no overflow occurs, the timer value can be copied from registers ICR1L and ICR1H. It is important to read the low byte first, then the high byte:

```
Dim LowByte As Byte, HighByte As Byte, Count As New UnsignedInteger

LowByte = Register.ICR1L
HighByte = Register.ICR1H

Count = CuInt(HighByte) * 256 + CuInt(LowByte)
```

The end result *Count* is a value with a range of 0 to 65 535 units. The unit conversion to seconds depends on the Timer1 tick rate. In our 28.8 kHz example, the scaling factor is about 3.47×10^{-5}, which gives us a limit of about (65 535 units)(3.47×10^{-5} s/unit) = 2.27 seconds before overflow.

Example code

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

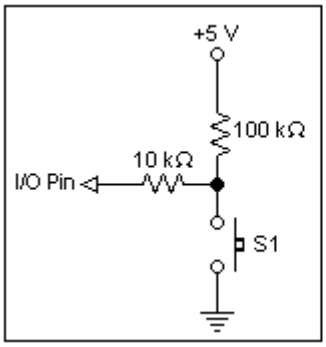

Figure 1

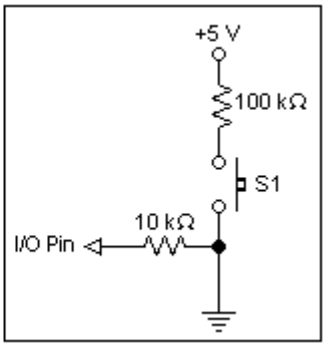

Figure 2

This example program can be used with either of the above circuits connected to the input capture pin (pin 31 on the BX-01; pin 12 on the BX-24; pin 20 on the BX-35). Depending on which circuit you use and whether the switch is normally-open or normally-closed, you can tailor the boolean variable RisingEdgeSelected to specify either a rising or falling edge for the capture.

2.00.A