

atmega8 counter bascom

Published: September 19, 2026 | Index ID: #-

Understanding the ATmega8 Counter with BASCOM: A Complete Guide

When you dive into the world of embedded systems, one of the most common tasks you'll encounter is counting pulses, measuring frequencies, or generating precise timing signals. The ATmega8 microcontroller, coupled with the BASCOM compiler, offers a powerful yet straightforward platform to accomplish these tasks. In this article, we'll walk through everything you need to know about using the ATmega8 counter in BASCOM, from basic concepts to advanced applications.

Table of Contents

1. ATmega8 Overview
2. Getting Started with BASCOM
3. Counter Fundamentals
4. ATmega8 Timer/Counter Modules
5. Configuring Counters in BASCOM
6. Sample Counter Programs
7. Practical Applications
8. Troubleshooting Common Issues
9. Optimization Tips
10. Conclusion

1. ATmega8 Overview

The ATmega8 is an 8 bit AVR microcontroller from Atmel (now Microchip Technology). It's widely used in hobbyist and industrial projects due to its:

- 32 /kB Flash memory for program storage
- 2 /kB SRAM for runtime data
- 1 /kB EEPROM for non volatile data
- 8 bit I/O ports (PORTA–PORTD)
- Three 8 bit timers/counters with PWM capabilities
- USART, SPI, I²C, and ADC peripherals
- Low power consumption modes

Its Timer/Counter modules (Timer0, Timer1, Timer2) can operate in various modes, making the ATmega8 an excellent choice for counting pulses and generating time based events.

2. Getting Started with BASCOM

BASCOM (Basic for AVR) is a high level compiler that translates BASIC like code into AVR machine code. It simplifies microcontroller programming, especially for beginners, while still offering access to low level registers.

- Installation: Download the BASCOM AVR IDE from the official website and install it on Windows.
- Project Setup: Create a new project, select "ATmega8" as the target device, and choose "AVR" as the

compiler type.

- **Compilation:** After writing code, click “Compile” to generate a .HEX file that can be uploaded to the ATmega8 via a programmer.

- **Debugging:** Use the built in simulator or connect an AVR ISP programmer (e.g., USBasp) for real time debugging.

3. Counter Fundamentals

A counter is a digital circuit that increments or decrements a value each time it receives a clock pulse. In microcontrollers, counters are typically implemented using timer/counter modules. They can be used to:

- Measure the frequency of an external signal.
- Count the number of pulses from a sensor.
- Generate precise delays or time intervals.
- Implement PWM for motor control.

Key parameters include:

- **Clock Source:** Internal oscillator or external clock.
- **Prescaler:** Divides the clock frequency to achieve desired counting speed.
- **Mode:** Normal, CTC (Clear Timer on Compare), PWM, etc.
- **Overflow:** When the counter exceeds its maximum value, it resets to zero, triggering an interrupt if enabled.

4. ATmega8 Timer/Counter Modules

The ATmega8 features three timers/counters, each with unique characteristics:

Timer0

- 8 bit timer/counter.
- Supports Normal, CTC, Fast PWM, and Phase Correct PWM modes.
- Can be used as a basic counter with an external clock source.

Timer1

- 16 bit timer/counter.
- Advanced features: Input Capture, PWM, CTC, and Normal modes.
- Ideal for high resolution counting and frequency measurement.

Timer2

- 8 bit timer/counter with a prescaler of 1, 8, 32, 64, 128, or 256.
- Supports Normal, CTC, and PWM modes.
- Often used for time keeping or simple pulse counting.

5. Configuring Counters in BASCOM

Below is a step by step guide to set up Timer1 as a counter with an external clock source.

5.1. Define the I/O Pin

Timer1's external clock input is on PB0 (pin 1). In BASCOM, we can assign this pin to a variable.

```
PB0 = INPUT
```

5.2. Set Timer1 Control Registers

We need to configure Timer/Counter Control Register B (TCCR1B) to select the external clock source and prescaler.

```
TCCR1B = &H00 ' Clear existing settings
```

```
TCCR1B.ICES1 = 1 ' Input Capture Edge Select: rising edge
```

```
TCCR1B.ICES1 = 1 ' Input Capture Edge Select: falling edge
```

```
TCCR1B.CS12 = 0
```

```
TCCR1B.CS11 = 0
```

```
TCCR1B.CS10 = 1 ' Clock Source: External Clock on T1 pin (Falling Edge)
```

In this configuration, the timer increments on every falling edge of the external signal on PB0.

5.3. Enable Timer Overflow Interrupt

To handle counter overflows, enable the overflow interrupt.

```
TIMSK = &H01 ' Enable Timer1 overflow interrupt
```

```
ICR1 = &HFFFF ' Set maximum count value for 16 bit counter
```

5.4. Write the ISR

Define an Interrupt Service Routine (ISR) to execute when Timer1 overflows.

```
On Timer1_Overflow
```

```
    ' Your code here
```

```
End On
```

Inside the ISR, you can process the count value or reset counters as needed.

6. Sample Counter Programs

Below are two example programs demonstrating common counter use cases.

6.1. Frequency Measurement Using Timer1

This program measures the frequency of an external signal on PB0 and displays the result on a serial terminal.

```
$regfile = "ATmega8.dat"
```

```
$crystal = 8000000
```

```
PB0 = INPUT
```

```
Serial_Init 9600
```

```
TCCR1B = &H00
```

```
TCCR1B.ICES1 = 0 ' Rising edge
```

```
TCCR1B.CS10 = 1 ' External clock on T1 pin (Falling Edge)
```

```
TIMSK = &H01 ' Enable Timer1 overflow interrupt
```

```
Do
```

```
    ' Main loop can perform other tasks
```

```
Loop
```

```

On Timer1_Overflow
  Dim freq As Word
  freq = (F_CPU / (TCCR1B.CS10 + 1)) / (TCNT1 + 1)
  Print freq
  TCNT1 = 0 ' Reset counter
End On

```

6.2. Pulse Counting with Overflow Handling

In this example, we count pulses on PB0 using Timer0 in Normal mode and handle overflow by incrementing a 32 bit counter.

```

$regfile = "ATmega8.dat"
$crystal = 8000000

```

```

PB0 = INPUT

```

```

TCCR0 = &H01 ' Normal mode, prescaler = 1
TIMSK = &H02 ' Enable Timer0 overflow interrupt

```

```

Dim pulseCount As Long
pulseCount = 0

```

```

Do
  ' Main loop can perform other tasks
Loop

```

```

On Timer0_Overflow
  pulseCount = pulseCount + 256 ' 8 bit timer overflow
End On

```

With these snippets, you can adapt the counter logic to various sensors and applications.

7. Practical Applications

Counting pulses and measuring frequencies are common tasks in many embedded projects. Below are some real world scenarios where the ATmega8 counter bascom can shine.

7.1. RPM Measurement for Motors

By connecting a Hall effect sensor to the external clock pin, you can count the number of pulses per minute and calculate the motor's RPM.

7.2. Light Level Meter

Use a photodiode connected to a comparator. The comparator's output can be fed into a counter to measure the frequency of light pulses, providing an indirect light intensity measurement.

7.3. Ultrasonic Distance Sensor

Many ultrasonic modules output a series of pulses whose duration correlates with distance. Counting these pulses or measuring their period allows you to calculate distance.

7.4. Digital Tachometer

By counting pulses from a gear tooth sensor, you can display the vehicle's speed or engine RPM on an LCD.

7.5. Pulse Width Modulation (PWM) Control

While PWM is usually used for output, the counter can be employed to generate precise PWM duty cycles for dimming LEDs or controlling servo motors.

8. Troubleshooting Common Issues

Even with careful configuration, counters can behave unexpectedly. Here are common pitfalls and how to resolve them.

8.1. Incorrect Clock Source Selection

Verify that the external clock pin is correctly assigned and that the signal polarity matches the configuration (rising vs. falling edge).

8.2. Overflow Not Triggering

Ensure the TIMSK register's overflow bit is set and that the global interrupt enable (sei()) is active.

8.3. Count Value Skew

If the measured frequency or pulse count differs from expected, check the prescaler settings and the actual oscillator frequency.

8.4. Signal Noise

Use a pull up resistor on the counter input or debounce the signal in software to avoid false counts.

8.5. Power Down Mode Interference

When the ATmega8 enters a low power mode, timers may stop. Make sure to keep the microcontroller awake during counting or use sleep modes that preserve timer operation.

9. Optimization Tips

Efficient counter usage can improve performance and power consumption. Consider the following strategies.

9.1. Use 16

Baca Juga (Artikel Terkait)

- [amarok](#)

URL: <http://adultvibex.com/amarok.pdf>

- [amaravati kathalu by satyam](#)

URL: <http://adultvibex.com/amaravati-kathalu-by-satyam.pdf>

- [amara la negra twerk](#)

URL: <http://adultvibex.com/amara-la-negra-twerk.pdf>

- [amar o depender descargar](#)

URL: <http://adultvibex.com/amar-o-depender-descargar.pdf>

Tags: #atmega8 #counter #bascom

