VIDEO#30: Electronic Basics #30: Microcontroller (Arduino) Timers

Timers and Their Applications in Microcontrollers

Introduction to Timers

Timers are essential components of microcontrollers that help in generating precisely timed events. They ensure accurate timing for operations such as counting seconds, blinking LEDs, moving figures on LED matrices, and generating Pulse Width Modulation (PWM) signals. Timers function independently of the main code execution, preventing delays and ensuring responsiveness.

Problems with Traditional Delay Functions

Using the delay() function in microcontroller programming creates two major issues:

1. **Blocking Code Execution:** The microcontroller is stuck in the delay function, ignoring other inputs such as button presses.
2. **Timing Inaccuracy:** The delay function may drift over time, causing errors in long-duration applications.

Introduction to Microcontroller Timers

Microcontrollers such as the Atmega328P, MSP430, and PIC series feature timers as peripheral components. These timers operate independently once configured, freeing up processing power for other tasks. The Atmega328P has three timers, and this lecture focuses on **Timer1**, which is a 16-bit timer offering multiple functionalities.

Normal Mode Operation of Timer1

In normal mode:

- The timer counts up each clock cycle until it reaches its maximum value (65,535 for a 16-bit timer).
- Once it reaches the maximum value, it resets to 0 and continues counting.

- The overflow flag is set in the **TIFR1** register when the timer resets.
- By enabling the **overflow interrupt**, a function is executed each time the timer overflows.

Configuring Timer1 in Normal Mode

To enable Timer1 in normal mode:

- Set **WGM13, WGM12, WGM11, and WGM10** to 0 in the **TCCR1A** and **TCCR1B** registers.
- Choose a **prescaler** by setting the **CS10, CS11, and CS12** bits.

Understanding Prescalers

Prescalers reduce the effective clock speed to control overflow timing. Common prescaler values:

- No prescaler (1) → Overflow time: ~4ms
- Prescaler 256 → Overflow time: ~1.04s

The formula to calculate overflow time:

Using Compare Match Mode (CTC Mode)

CTC (Clear Timer on Compare Match) mode allows for more precise timing control. Instead of waiting for an overflow, we can set custom compare values:

- The timer resets when it reaches the **OCR1A** value instead of the maximum 65,535.
- Compare match interrupts are triggered when the timer matches **OCR1A** and **OCR1B**, allowing multiple time events.

CTC Mode Configuration

- Set **WGM12** to 1.
- Define values for **OCR1A** and **OCR1B** using the formula:

Example values:

- **OCR1A = 31,249** → 0.5s interval
- **OCR1B = 15,624** → 0.25s interval

PWM Generation Using Timers

PWM (Pulse Width Modulation) is used to control devices like motors and LED brightness. Timer1 supports **Fast PWM Mode**, where:

- The timer counts up to 255 (8-bit mode) or a custom top value (ICR1 register for higher resolution).
- The output pin is set HIGH at compare match and LOW at overflow, creating a square wave.

Fast PWM Configuration

- Set **WGM12 and WGM10** to 1.
- Enable non-inverting mode by setting **COM1A1 and COM1B1**.
- Choose an appropriate prescaler for frequency control.

Variable Frequency PWM

- Use **ICR1** to set a dynamic top value.
- Adjust **OCR1A** to change duty cycle.
- A second potentiometer can control frequency up to **8MHz**.

Timers are crucial for creating precise time-based events, freeing up the main execution loop, and enabling efficient PWM generation. Understanding normal mode, CTC mode, and PWM configurations allows better control over timing operations in microcontrollers. Reading the Atmega328P datasheet provides further insights into advanced timer functionalities.