МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ТАРАСА ШЕВЧЕНКА
ФІЗИЧНИЙ ФАКУЛЬТЕТ

# КОНКУРСНА РОБОТА: ОТРИМАННЯ СИГНАЛУ НАЙВИЩОЇ ЧАСТОТИ НА ПЛАТІ ARDUINO UNO (16 MHZ)

Виконав: студент 2 курсу 5-А групи спеціальності 104 «Фізика та астрономія»
Свінтозельський Володимир Ярославович

Науковий керівник: викладач
Єрмоленко Руслан Вікторович

Київ – 2020

ЗМІСТ

# ВСТУП

Отримання сигналів, частота яких співмірна із частотою роботи тактового генератора, підключенного до мікроконтроллера Atmega, може бути досить цікавою задачею. Фізичний максимум - $\omega = \omega_{gen}/2$ пов'язаний із необхідністю почергового притягання ножки до високого та низького потенціалів, на кожний з таких переходів витрачається деякий мінімальний час $\tau \approx 1/\omega_{gen}$.

Розумний метод досягнення такої цілі - використання таймерів, що базуються на перериваннях мікроконтроллера atmega328, який використовується на платах arduino uno. Перевагою такого методу є те, що контроллер може спокійно виконувати інші завдання, а в певні моменти часу, визначені таймером, переривати виконання основної програми для генерування нашого сигналу.

# РОЗДІЛ 1
# НАЛАШТУВАННЯ ТА РОБОТА ІЗ ТАЙМЕРАМИ У КОНТРОЛЕРАХ ATMEGA328

## 1.1 Короткі відомості для використовуваних побітових операцій

Оскільки весь процес роботи із таймерами полягає у використанні реєстрів, нагадаємо деякі побітові операції, що стануть у нагоді.

- Побітове **АБО (x|y)**: Нехай задано число $x = 2$ та $= 4$. Їхні представлення у двійковій системі: $x = 10$ та $y = 100$. Результатом роботи оператора або буде $x|y = 110$

- Функція **bit(n)**: Створення байту, в якому біт номер $n = 1$, а всі інші рівні нулю. Її зручно використовувати в парі із попереднім оператором, коли потрібно в існуючому байті певному біту приписати 1.

## 1.2 Принцип роботи таймерів

Робота таймеру полягає у збільшенні певної змінної, що носить ім'я рахункового регістру. Він збільшує її на один, крок за кроком, аж поки не досягне максимального значення, що повязане із розміром регістру. У момент переповнення таймер встановлює біт відповідного прапорця.

Ми можемо вручну перевіряти стан цього прапорця, але краще використовувати таймерний перемикач, який викликає переривання автоматично, після встановлення прапорця. Звісно ми можемо привязати підпрограму переривання, в якій написати все, для чого нам власне і потрібен таймер.

## 1.3 Типи таймерів, доступні для arduino uno [1]

- **Timer0**: 8-бітний таймер (максимальне значення 255), який використовується стандартними функціями delay() та millis(). Тому краще не використовувати цей таймер без потреби.

- **Timer1**: 16-бітний таймер (максимальне значення 65535), який використовується бібліотекою Arduino Servo.

- **Timer2**: 8-бітний таймер, який використовується функцією tone().

## 1.4 Конфігурація регістрів

Для налаштування роботи таймеру використовується 2 регістри: TCCRxA та TCCRxB, де x - номер таймеру. Кожен з них містить 8 біт. [2]

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x80) | COM1A1 | COM1A0 | COM1B1 | COM1B0 | – | – | WGM11 | WGM10 | TCCR1A |
| Read/Write | R/W | R/W | R/W | R/W | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Рис. 1.1: Регістр TCCR1A.

| COM1A1/COM1B1 | COM1A0/COM1B0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC1A/OC1B disconnected. |
| 0 | 1 | Toggle OC1A/OC1B on compare match. |
| 1 | 0 | Clear OC1A/OC1B on compare match (set output to low level). |
| 1 | 1 | Set OC1A/OC1B on compare match (set output to high level). |

Рис. 1.2: Режими роботи виходів OC1A та 0C1B.

Розглянемо регістр TCCR1A (Рис. 1.1). Біти 7-6 відповідають за режим роботи каналу А (ножка 0C1A, на Arduino Uno - D9), 5-4 - режим каналу Б (0C1B). Для нашої задачі, необхідно просто періодично подавати високу напругу на вихід Тому використовується режим порівняння на каналі А. Канал Б не задіяний. Всі режими роботи каналів показані на рис. 1.3

Таймер може видавати як звичайний так і різноманітний ШИМ сигнал. Але очевидно, що ці всі витребеньки нам не потрібні, тому в даній роботі використовується звичайний CTC режим, який передбачає очистку таймера при співпадінні його значення з деяким, наперед заданим. Тобто, використовується режим 4 на рис. 1.4. Як видно із колонки TOP - значення, з яким буде порівнюватися таймер задається байтом 0CR1A.

Останні біти, потрібні нам (CS12 - CS10), відповідають за режим роботи самого таймеру. А саме, режим використання тактового генератору. Ми можемо використовувати частоту останнього ’як є’, а можемо ділити її на різні степені двійки. Тобто використовувати так званий дільник, що буде уповільнувати роботу. Очевидно, що для наших цілей, такі речі можуть бути смертельними, тому в роботі зануляються всі біти, окрім CS10, тобто використовується режим без дільника (рис. 1.5)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0x81) | ICNC1 | ICES1 | – | WGM13 | WGM12 | CS12 | CS11 | CS10 | TCCR1B |
| Read/Write | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Рис. 1.3: Регістр TCCR1B.

| Mode | WGM13 | WGM12 (CTC1) | WGM11 (PWM11) | WGM10 (PWM10) | Timer/Counter Mode of Operation | TOP | Update of OCR1x at | TOV1 Flag Set on |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Normal | 0xFFFF | Immediate | MAX |
| 1 | 0 | 0 | 0 | 1 | PWM, phase correct, 8-bit | 0x00FF | TOP | BOTTOM |
| 2 | 0 | 0 | 1 | 0 | PWM, phase correct, 9-bit | 0x01FF | TOP | BOTTOM |
| 3 | 0 | 0 | 1 | 1 | PWM, phase correct, 10-bit | 0x03FF | TOP | BOTTOM |
| 4 | 0 | 1 | 0 | 0 | CTC | OCR1A | Immediate | MAX |
| 5 | 0 | 1 | 0 | 1 | Fast PWM, 8-bit | 0x00FF | BOTTOM | TOP |
| 6 | 0 | 1 | 1 | 0 | Fast PWM, 9-bit | 0x01FF | BOTTOM | TOP |
| 7 | 0 | 1 | 1 | 1 | Fast PWM, 10-bit | 0x03FF | BOTTOM | TOP |
| 8 | 1 | 0 | 0 | 0 | PWM, phase and frequency correct | ICR1 | BOTTOM | BOTTOM |
| 9 | 1 | 0 | 0 | 1 | PWM, phase and frequency correct | OCR1A | BOTTOM | BOTTOM |
| 10 | 1 | 0 | 1 | 0 | PWM, phase correct | ICR1 | TOP | BOTTOM |
| 11 | 1 | 0 | 1 | 1 | PWM, phase correct | OCR1A | TOP | BOTTOM |
| 12 | 1 | 1 | 0 | 0 | CTC | ICR1 | Immediate | MAX |
| 13 | 1 | 1 | 0 | 1 | (Reserved) | – | – | – |
| 14 | 1 | 1 | 1 | 0 | Fast PWM | ICR1 | BOTTOM | TOP |
| 15 | 1 | 1 | 1 | 1 | Fast PWM | OCR1A | BOTTOM | TOP |

Рис. 1.4: Різноманітні налаштування форми сигналів на виході.

| CS12 | CS11 | CS10 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source (Timer/Counter stopped). |
| 0 | 0 | 1 | $clk_{I/O}$/1 (no prescaling) |
| 0 | 1 | 0 | $clk_{I/O}$/8 (from prescaler) |
| 0 | 1 | 1 | $clk_{I/O}$/64 (from prescaler) |
| 1 | 0 | 0 | $clk_{I/O}$/256 (from prescaler) |
| 1 | 0 | 1 | $clk_{I/O}$/1024 (from prescaler) |
| 1 | 1 | 0 | External clock source on T1 pin. Clock on falling edge. |
| 1 | 1 | 1 | External clock source on T1 pin. Clock on rising edge. |

Рис. 1.5: Доступні режими використання таймером тактового генератору.

# РОЗДІЛ 2
# НЕЙМОВІРНЕ ПЕРЕТВОРЕННЯ ПЛАТИ ARDUINO UNO У ГЕНЕРАТОР ІМПУЛЬСІВ ЦІКАВОЇ ФОРМИ ІЗ ЧАСТОТОЮ $\omega = 8MHZ$

## 2.1 Написання програми у середовищі Arduino IDE.

Написання програми (рис. 2.1) по суті зводиться до встановленя потрібних значень бітів регістрів, як описано у попередньому розділі. Тобто, встановлення CTC режиму шляхом присваювання біту WGM12 одиниці регістру TCCR1B, вибору режиму без дільника - присваювання одиниці біту CS10. Та подання на канал A (на arduino Uno вихід D9) високого рівня кожного разу, як встановлюється прапорець очистки регістру таймеру - біт COM1A0. Сама ж очистка відбувається як тільки значення регістру буде більшим за значення OCR1A. Тобто, таким чином можна отримати частоти вихідного сигналу 8, 8/2, 8/3 ... MHz.

```
void setup ()
{
  pinMode (9, OUTPUT);
  TCCR1A = bit (COM1A0);
  TCCR1B = bit (WGM12) | bit (CS10);
  OCR1A =  0;
}
void loop (){}
```

Рис. 2.1: Код програми.

## 2.2 Результати роботи

Після прошивки плати arduino uno відповідним скетчем, та підключення осцилографа до відповідних ножок плати, було отримано картинку, зображену на рис. 2.2. Дійсно було отримано сигнал, частота якого доходила до обіцяних $\omega = 8MHz = \omega_{gen}/2$, що є половиною від частоти вмонтованого у плату кварцового генератора. Звісно форма сигналу досить цікава, але її аналіз виходить за рамки цієї роботи.

Також, потрібно звернути увагу на пусту функцію $loop()$ - можливість для використання такого генератора у будь-яких проектах

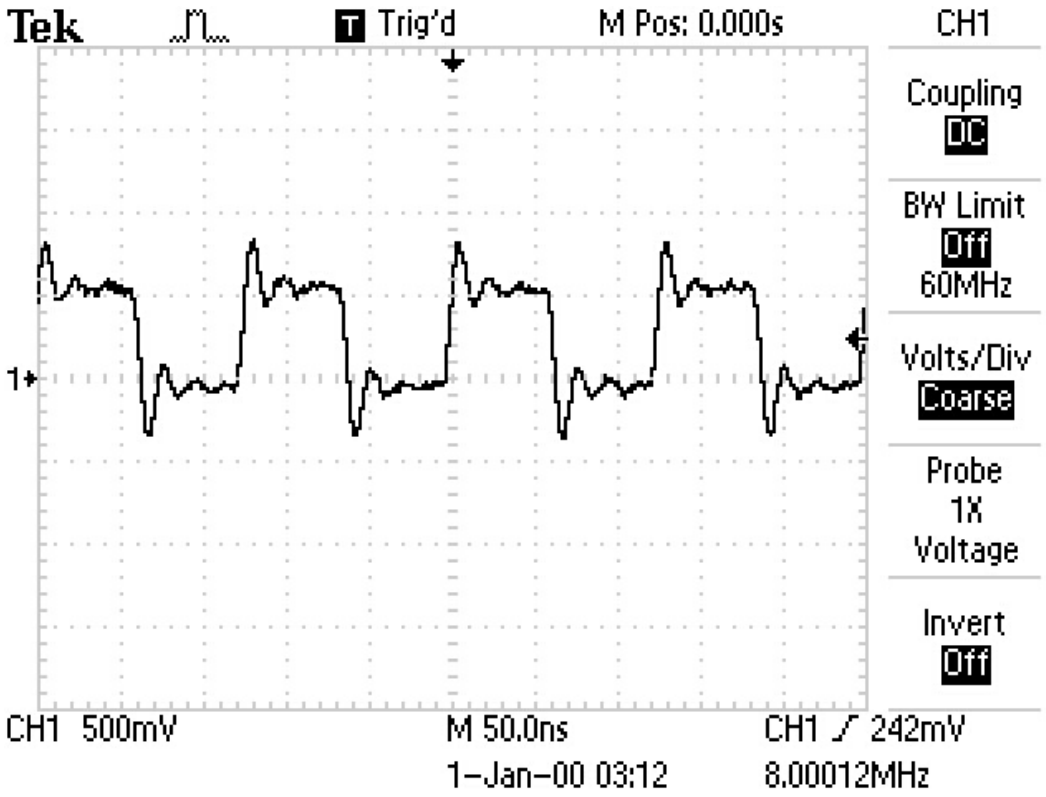


Рис. 2.2: Покази осцилографа.

# ВИСНОВКИ

В даній роботі було отримано сигнал частотою до $\omega = 8MHz$ на ножці плати Arduino Uno. Також реалізована можливість пониження цієї частоти, при потребі, у ціле число раз.

Показано можливість налаштування вмонтованих у мікроконтроллери Atmega3 таймерів, та їх використання для різноманітних задач. Перевірено на практиці та показано особливості режиму роботи CTC.

# БІБЛІОГРАФІЯ

[1] Timer interrupts. — Access mode: https://arduinodiy.wordpress.com/2012/02/28/timer-interrupts/ (online; accessed: 2019-02-19).

[2] ATmega328P, 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash, DATASHEET. — Access mode: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (online; accessed: 2019-02-19).