Technika Mikroprocesorowa 2

Projekt – Generator funkcyjny

Michał Tomacha

1. Podłączenie
    a. Klawiatura:
        i. C3 – PTB6
        ii. C4 – PTB7
        iii. R4 – GND
    b. Wyświetlacz:
        i. SCL – PTB3
        ii. SDA – PTB4
        iii. VCC – 5V
        iv. GND – GND
    c. Głośnik wpinany jest pomiędzy PTB1 a GND

2. Instrukcja obsługi
    a. Przycisk S1 powoduje cykliczną zmianę kształtu przebiegu (prostokątny -> sinus -> trójkątny).
    b. Przycisk S2 przełącza między trybem ustawiania częstotliwości a trybem ustawiania napięcia międzyszczytowego wybranego przebiegu.
    c. Pole dotykowe pozwala na regulację wybranego parametru
    d. Wyświetlacz pokazuje kształt przebiegu (sqr, sin, tri) oraz obecnie ustawiany parametr wraz z jego wartością.

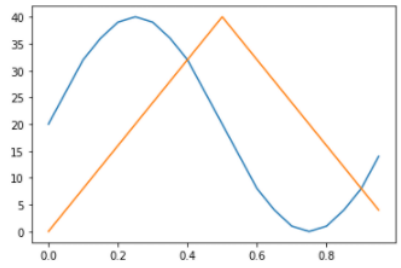
3. Parametry
    a. Regulacja częstotliwości w zakresie: 20Hz – 14460Hz
    b. Regulacja napięcia międzyszczytowego: 0.06V – 2.72V

4. Zasada działania
    Dla tablicy o ilości elementów n, PIT0 powinien generować przerwania z częstotliwością n razy większą od pożądanej częstotliwości sygnału wyjściowego (zwiększenie liczby próbek spowoduje wygładzenie przebiegu na wyjściu, lecz ograniczy jego maksymalną częstotliwość). Przerwanie od PIT0 powoduje wysłanie kolejnej próbki z wybranej tablicy do przetwornika C/A i ustalenie odpowiedniej wartości na wyjściu (PTB1). Poprzez przesuwanie palcem po polu dotykowym można wpłynąć na częstotliwość generowania przerwań (zmiana wartości rejestru LDVAL) oraz na wartość napięcia na wyjściu (przemnażanie wartości próbki).

5. Generowanie tablic
   Tablice przebiegów powstają poprzez pobranie równo oddalonych od siebie próbek z jednego okresu wybranego przebiegu. Żeby uzyskać jak najwyższą częstotliwość maksymalną przy zachowaniu kształtu przebiegu zdecydowałem się na tablice zawierające 20 próbek.

   Aby wygenerować tablice dla przebiegu sinusoidalnego i trójkątnego użyłem poniższego skryptu w języku Python.

```
In [1]: import numpy as np
        from scipy import signal as sci
        from matplotlib import pyplot as plt

        f = 1
        T = 1/f
        dt = T/20 #odstęp między próbkami dla 20 próbek
        t = np.arange(0,T,dt) #generacja punktów w których pobrana zostanie próbka

        sin = 20+20*np.sin(2*np.pi*f*t) #obliczenie wartości próbek przebiegu sinusoidalnego ze składową stałą
        sin = np.around(sin) #zaokrąglenie wartości próbek
        sin = sin.astype(int)

        tri = 20+20*sci.sawtooth(2*np.pi*f*t, 0.5) #obliczenie wartości próbek przebiegu trójkątnego ze składową stałą
        tri = np.around(tri) #zaokrąglenie wartości próbek
        tri = tri.astype(int)

        np.set_printoptions(formatter={'int':hex})

        print('sine:', sin)
        print('triangle:', tri)

        plt.plot(t, sin)
        plt.plot(t, tri)

sine: [0x14 0x1a 0x20 0x24 0x27 0x28 0x27 0x24 0x20 0x1a 0x14 0xe 0x8 0x4 0x1
 0x0 0x1 0x4 0x8 0xe]
triangle: [0x0 0x4 0x8 0xc 0x10 0x14 0x18 0x1c 0x20 0x24 0x28 0x24 0x20 0x1c 0x18
 0x14 0x10 0xc 0x8 0x4]

Out[1]: [<matplotlib.lines.Line2D at 0x1ce85670>]
```



   Generator powinien bez większych problemów działać dla tablic innych prostych przebiegów przy zachowaniu długości 20 oraz wartości próbek z zakresu 0x00 – 0x28 (0-40).

6. Prezentacja działania
   Link do repozytorium: https://github.com/tomacha/Generator-funkcyjny
   Film z prezentacją znajduje się pod adresem: https://youtu.be/td5GKXfnqgk
   Podczas prezentacji zdecydowałem się na użycie oscyloskopu, ponieważ za jego pomocą dokładniej można zaprezentować działanie projektu.