Politechnika Opolska

LABORATORIUM

Technika Mikroprocesorowa

KIERUNEK STUDIÓW:	AiR Ns	ROK STUDIÓW:	III
SEMESTR:	VI	ROK AKADEMICKI:	2019/2020

Temat ćwiczenia:
Program wykorzystujący funkcje timera/licznika

Projekt wykonali:			
Nazwisko i imię:		*Nazwisko i imię:*	
1.	Marco Tiszbierek	**2.**	Marek Szczekała
3.	Klaudiusz Tacica	**4.**	

Ocena:	*Data:*	*Uwagi:*

1. Wstęp:

Z powodu braku posiadania Płytki firmy Texas Instruments MSP-EXP430FR4133 lub innej z jakich korzystamy na laboratoriach, na której powinno zostać wykonane ćwiczenie. W celu wykonania zadania układ został zbudowany na płytce stykowej oraz zaprogramowany przy użyciu programatora ASP-USBASP.

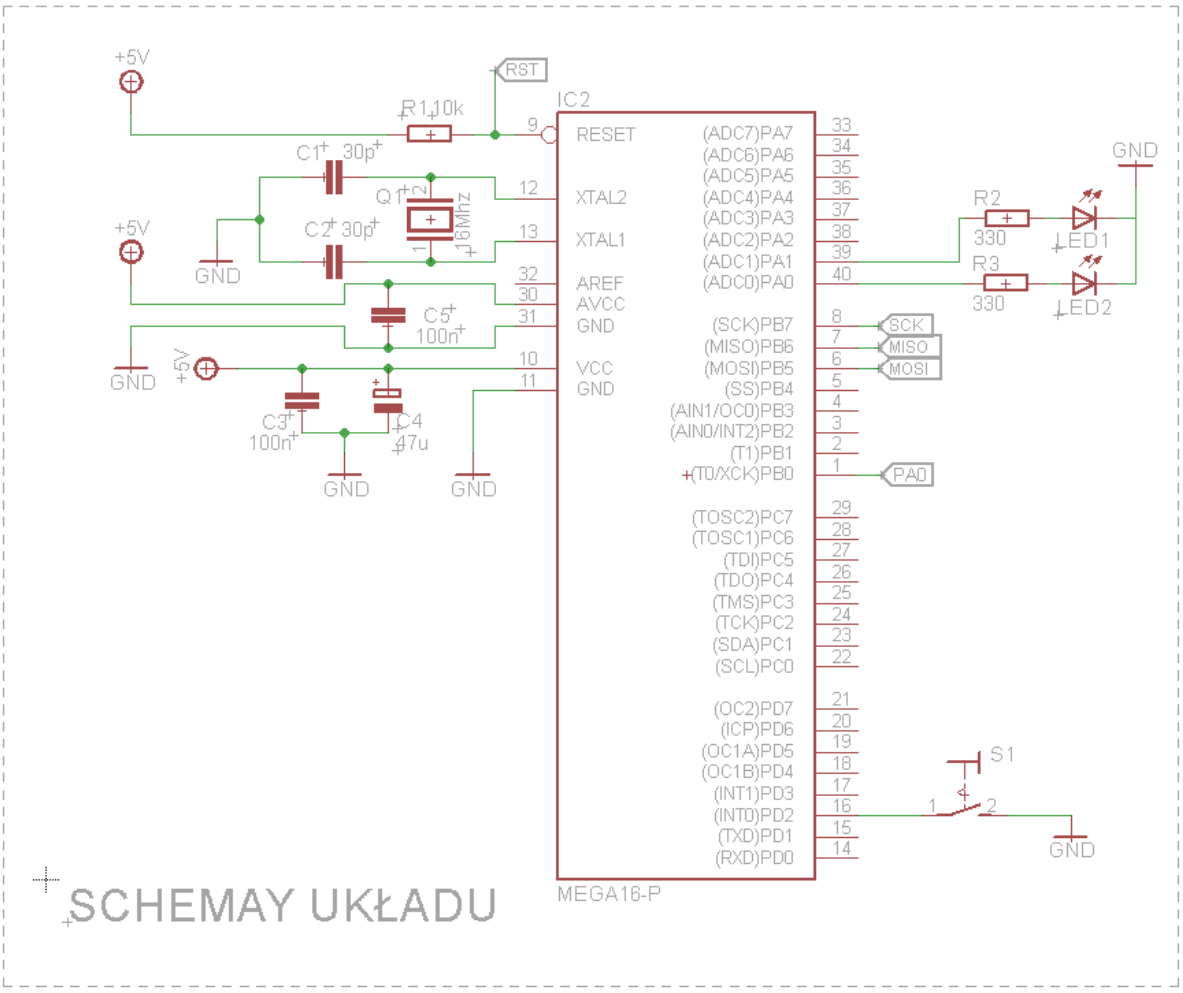
2. Cel ćwiczenia:

Głównym celem ćwiczenia było zaznajomienie się z funkcjami Timera/Licznika Do tego celu należało stworzyć program wykorzystujący właśnie te funkcje. Kod zostało napisany w języku C w programie eclipse.

3. Polecenie:

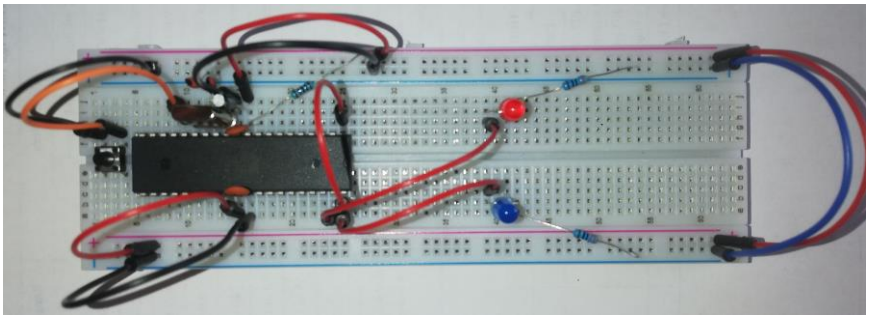
Napisać program, który będzie startował timer T1 3s po wciśnięciu przycisku, a w tym samym momencie niech uruchomi się dioda zielona. Po 3s dioda zielona gaśnie, a czerwona się zaświeca i tak na zmianę, póki licznik nie doliczy do 5, licznik inkrementuje się zawsze, kiedy T1 skończy liczyć 3s. Kiedy licznik doliczy do 5 niech uruchomi się timer T2, który odlicza 1s. Po sekundzie program wraca na początek i czeka na wciśniecie przycisku. W trakcie liczenia T2 niech dioda czerwona zamruga 2x na czerwono informując o zakończeniu programu.

4. Schemat zbudowanego układu:

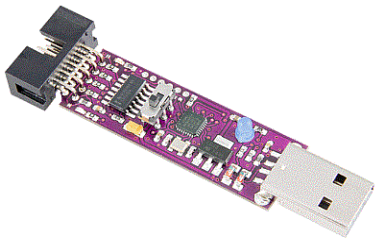


Rys.1. Schemat zbudowanego na płytce stykowej układu w celu realizacji ćwiczenia.

5. Zdjęcie zbudowanego układu oraz programatora wykorzystanych w zadaniu:



Zdj.1. Zdjęcie zbudowanego układu na mikrokontrolerze firmy Atmel model ATmega16.



Zdj.2. Zdjęcie użytego programatora ATB-USBASP.

6. Opis działania programu:

W celu wizualizacji działania zawartej w programie funkcji timera oraz licznika, program został napisany w taki sposób, aby po podłączeniu układu do zasilania oraz po rozpoczęciu działania pętli głównej układ czekał ,,bezczynnie" (chodź pętla while ciągle się wykonuje) na wciśnięcie przez użytkownika przycisku. Po wciśnięciu przycisku oraz zwolnieniu program wchodzi w instrukcję warunkową, w której zeruje licznik, zmienną t2, w której przechowywana jest ilość impulsów pozwalających odliczyć 1s przed zakończeniem programu, rejestry timera/licznika T0 oraz 11, włącza diodę LED niebieską (zamiennie za zieloną) oraz ustawia na zmiennej x wartość 1. Po zwolnieniu przycisku i wykonaniu wspomnianej wcześniej pętli timer T1 odlicza 3 sekundy, w trakcie zliczania 3s na bieżąco rejestr OCR1A, w którym zapisana jest wartość 46875 odpowiadająca 3s jest porównywany z rejestrem licznika TCNT1 w momencie zrównania się tych rejestrów wykonywana jest procedura przerwań dla kanału A, w której zmieniany jest stan na obu diodach, nadpisywana wartość licznika oraz zerowany rejestr licznika T1 TCNT1, aby kolejny raz mógł zliczyć od zera dokładnie 3s. Procedura przerwań dla kanału A wykonywana jest, dopóki do zmiennej counter nie zostanie przypisana wartość 5. W trakcie przełączania się stanu na obu diodach na zbocze opadające LED niebieskiej dokładnie po 3s, na pinie PB0(T0), timera/licznika T0 ustawionego na zewnętrze źródło zegara doliczany jest jeden impuls. Impulsy zliczane są w rejestrze TCNT0, którego wartość podczas wykonywania się procedury przerwania kanału A nadpisywana jest do zmiennej counter (naszego licznika). Cykl ten powtarza się, dopóki zmienna counter nie zostanie nadpisana wartością 5, kiedy to się stanie spełnione zostaną instrukcje warunkowe do rozpoczęcia wykonywania się procedury przerwań dla kanału B. W procedurze przerwań kanału B wykonywana jest zmiana stanu na LED czerwonej, nadpisywana zmienna t2 wartością z rejestru TCNT1, po czym wartość rejestru zostaje zerowana. Warunek wykonywania się przerwań na kanale B będzie spełniony, dopóki w zmiennej t2 nie znajdzie się wartość 15625 odpowiadająca 1s. W trakcie zliczania 1s procedura przerwania dla kanału B wykona się 3 razy. Co spowodowane jest tym, że rejestr TNCT1 na bieżąco jest porównywany z rejestrem OCR1B, w którym zapisana jest wartość 5208 odpowiadająca 1/3s. Moment zrównania się wartości tych rejestrów wywoła wykonanie się procedury przerwań kanału B dzięki czemu dioda LED czerwona w trakcie zliczania 1s zamruga 2 raz w równych odstępach czasu. Po upływie 1 sekundy program wchodzi w pętle warunkową, w której do zmiennej x zostaje przypisana wartość 0 oraz obie diody ustawione w stan niski. W takim stanie program wraca do stanu ,,bezczynności" wykonując na próżno pętle while czekając na ponowne wciśniecie przycisku.

7. Informacje potrzebne do konfiguracji rejestrów liczników/timerów:

W celu włącznie funkcji timerów/liczników w pierwszej kolejności należy odpowiednio ustawić poszczególne rejestry. Wszelkie informacje można bez problemu znaleźć w notach katalogowych poszczególnych układów. W różnych układach rejestry mogą przybierać inne nazwy wiec zawsze należy posługiwać się notami katalogowymi.

7.a) Rejestry oraz ustawienia wykorzystane do konfiguracji licznika/timera T0:

Tab.1. Rejestr kontroli licznika/timera T0

Rejestr TCCR0								
Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	SC02	CS01	CS00
Dostęp	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Tab.2. Ustawienia źródeł zegara dla licznika 0

CS02	CS01	CS00	Opis
0	0	0	Nie wybrane źródło zegara, licznik zatrzymany
0	0	1	clk_{IO} (bez preskalera)
0	1	0	clk_{IO} / 8 (preskaler)
0	1	1	clk_{IO} / 64 (preskaler)
1	0	0	clk_{IO} / 256 (preskaler)
1	0	1	clk_{IO} / 1024 (preskaler)
1	1	0	Zewnętrzne źródło zegara z pinu T0, zbocze opadające [1)]
1	1	1	Zewnętrzne źródło zegara z pinu T0, zbocze narastające [1)]

[1)] T0 może być źródłem taktowania nawet gdy jest ustawiony jako pin wyjściowy

Tab.3. Rejestr licznika/timera T0

Rejestr TCNT0								
Bit	7	6	5	4	3	2	1	0
	TCNT0[7:0]							
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Tab.4 Rejestr Output Compare

Rejestr OCR0								
Bit	7	6	5	4	3	2	1	0
	OCR0[7:0]							
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

W tabeli 1, 3, 4 znajdują się rejestry, które zostały wykorzystane do wykonania programu, natomiast w tabeli 2 przedstawione zostały ustawienia źródeł zegara dla licznika 0 jakie można wykorzystać przy ustawienie odpowiednich bitów rejestru z tabeli 1. Dokładne informacje oraz znaczenia poszczególnych rejestrów nie zostały wypisane, ponieważ zajęło by to dużo miejsca, ale bez problemu informacje te można znaleźć w notach katalogowych poszczególnych mikrokontrolerów. W zależności od modeli mikrokontrolerów nazwy rejestrów mogą się różnić.

7.b) Rejestry oraz ustawienia wykorzystane do konfiguracji licznika/timera T1:

Tab.5. Rejestr A kontroli licznika/timera T1

Rejestr TCCR1A								
Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Dostęp	R/W	R/W	R/W	R/W	W	W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Tab.6. Rejestr B kontroli licznika/timera T1

Rejestr TCCR1B								
Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
Dostęp	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Tab.7. Ustawienia źródeł zegara dla licznika 1

CS12	CS11	CS10	Opis
0	0	0	Nie wybrane źródło zegara, licznik zatrzymany
0	0	1	clk_{IO} (bez preskalera)
0	1	0	clk_{IO} / 8 (preskaler)
0	1	1	clk_{IO} / 64 (preskaler)
1	0	0	clk_{IO} / 256 (preskaler)
1	0	1	clk_{IO} / 1024 (preskaler)
1	1	0	Zewnętrzne źródło zegara z pinu T1, zbocze opadające
1	1	1	Zewnętrzne źródło zegara z pinu T1, zbocze narastające

Pin T1 może być źródłem taktowania nawet gdy jest ustawiony jako pin wyjściowy.

Tab.8. Rejestr licznika/timera T1

Rejestr TCNT1								
Bit	7	6	5	4	3	2	1	0
TCNT1H	TCNT1[15:8]							
Rejestr TCNT1								
TCNT1L	TCNT1[7:0]							
Dostęp	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Tab.9. Rejestr Output Compare 1A

Rejestr OCR1A								
Bit	7	6	5	4	3	2	1	0
OCR1AH	OCR1A[15:8]							
OCR1AL	OCR1A[7:0]							
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Tab.10. Rejestr Output Compare 1B

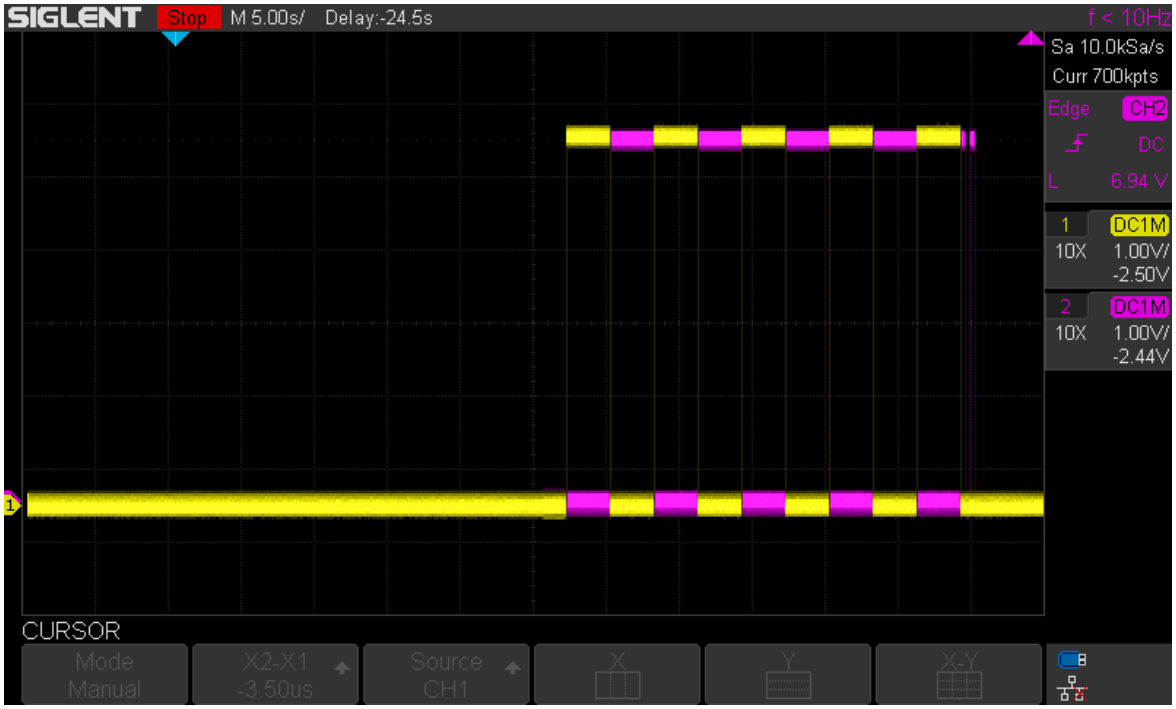
Rejestr OCR1B								
Bit	7	6	5	4	3	2	1	0
OCR1BH	OCR1B[15:8]							
OCR1BL	OCR1B[7:0]							
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Tab.11. Rejestr maskowania przerwań liczników

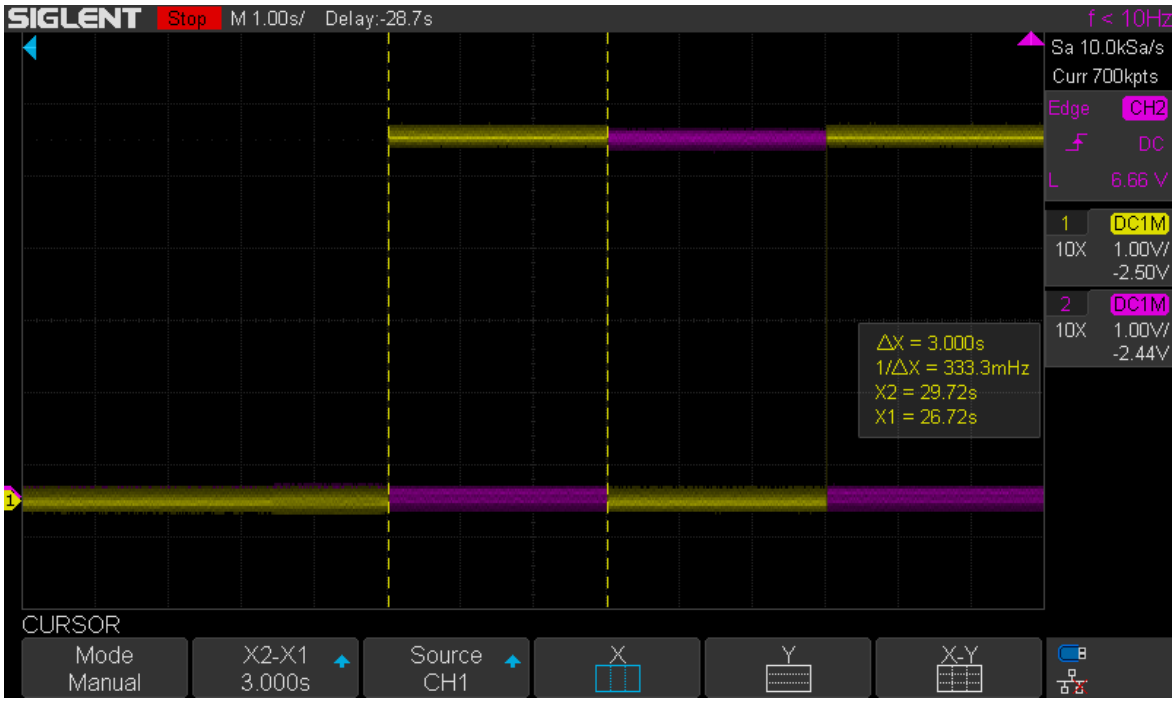
Rejestr TIMSK								
Bit	7	6	5	4	3	2	1	0
	[illegible]	[illegible]	TICIE1	OCIE1A	OCIE1B	TOIE1	[illegible]	[illegible]
Dostęp	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Wartość początkowa	0	0	0	0	0	0	0	0

Podobnie jak w ustawieniach licznika/timera 0. W tabeli 5, 6, 8, 9, 10, 11 znajdują się rejestry, które zostały wykorzystane do wykonania programu, natomiast w tabeli 7 przedstawione zostały ustawienia źródeł zegara dla licznika 1 jakie można wykorzystać przy ustawienie odpowiednich bitów rejestru z tabeli 6. Dokładne informacje oraz znaczenia poszczególnych rejestrów jak poprzednio również nie zostały wypisane, ponieważ zajęło by to dużo miejsca, ale również bez problemu informacje te można znaleźć w notach katalogowych poszczególnych mikrokontrolerów. W zależności od modeli mikrokontrolerów nazwy rejestrów mogą się różnić.

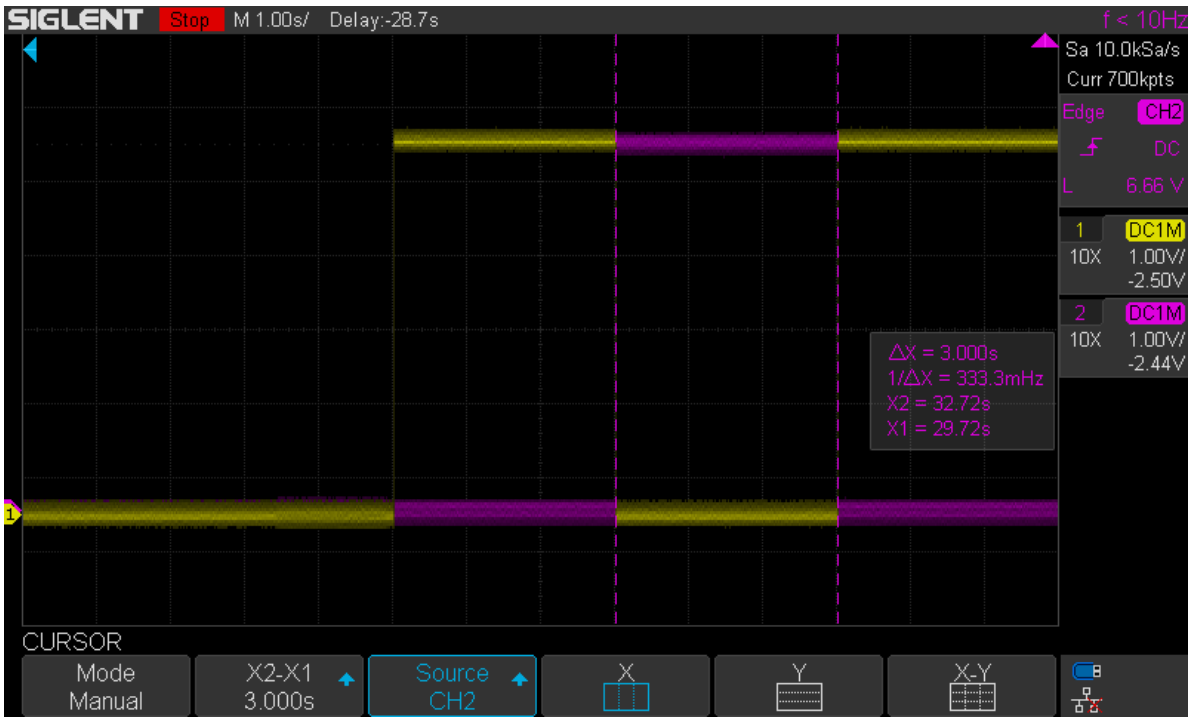
8. Charakterystyki z przebiegu działania programu:



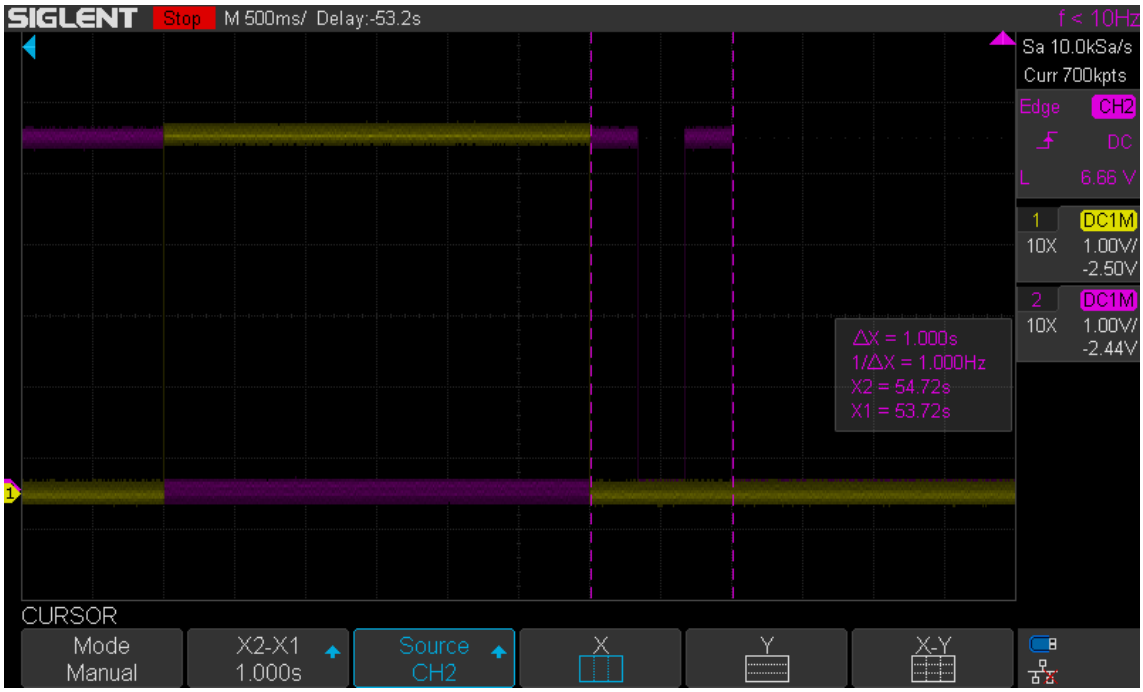
Rys.1. Rysunek przedstawia kolejno charakterystyki przebiegu napięcia na diodach. Charakterystyka żółta (LED niebieska), Charakterystyka fioletowa (LED czerwona).



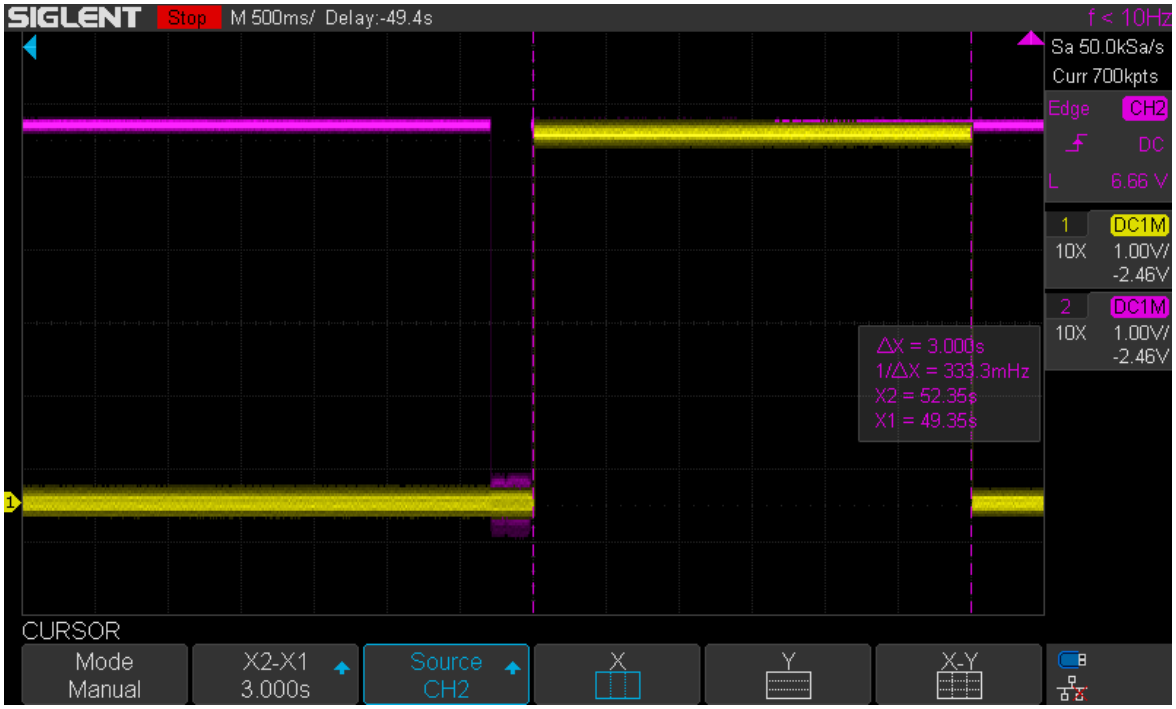
Rys.2. Przybliżonym rys.1. w celu dokładnego ukazania czasu trwania stanu wysokiego na diodzie LED niebieskiej (przebieg żółty).



Rys.3. Przybliżonym rys.1. w celu dokładnego ukazania czasu trwania stanu wysokiego na diodzie LED czerwonej (przebieg fioletowy).



Rys.4. Przybliżonym rys.1. w celu dokładnego ukazania zachowanie się diody LED czerwonej (przebieg fioletowy), po zliczeniu przez licznik 5 cykli.



Rys.5. Rysunek przedstawia moment włączania programu przez naciśniecie przycisku (przebieg fioletowy) oraz Diody LED niebieskiej (przebieg żółty).

9. Kod programu:

```
// Program z funkcja obsługi przerwań
// Wykonane przez : Tacica, Tiszbierek, Szczekała AiR ns semestr VI
//*************************************************************************************************************************************************************
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdbool.h>
//*************************************************************************************************************************************************************Definicje dla procesora
#define SW (1<<PD2)                    // Definicja pinu do którego, jest podłączony przycisk
#define SW_DOWN !(PIND & SW)           // Definicja marko sprawdzajace czy na przycisku jest stan niski
#define SW_UP (PIND & SW)              // Definicja makro sprawdzającego czy na przycisku jest stan wysoki
#define LED_1 (1<<PA1)                 // Definicja pinu do którego, jest podłączona dioda czerwona
#define LED_1_TOG PORTA ^= LED_1       // Definicja zmiany stanu na dodzie czerwonej
#define LED_2 (1<<PA0)                 // Definicja pinu do którego, jest podłączona dioda niebieska
#define LED_2_TOG PORTA ^= LED_2 // Definicja zmiany stanu na diodzie niebieskiej
//*************************************************************************************************************************************************************Tworzenie zmiennych
bool x=0;                       // Tworzenie zmiennej typu bool x dla właczania układu, wartości początkowo 0
unsigned char counter=0;        //  Tworzenie zmiennej typu unsigned char dla coounter, wartośc początkowa 0
int t2=0;                       //   Tworzenie zmennej typu int t2 dla timera t2, wartością początkową 0
//*************************************************************************************************************************************************************Procedury obsługi przerwań
ISR(TIMER1_COMPA_vect)                      // Procedura obsługi przerwań kanału A
{
   if ((x==1)&&(counter<5))                // Instrukcja if z warunkami jej wykonania, kiedy  ( x==1 po naciśnieciu i zwolnieniu przyciska) & >>
       {                                   //    >> & ( counter<5 ) licznik zwiksza sie o 1 z zboczem opadajacym LED_2 kiedy t1 konczy liczyc 3s
       LED_2_TOG;                         // Zmiana stanu na dziodzie niebieskiej //
       LED_1_TOG;                         // Zmiana stanu na dziodzie czerwonej
       counter = TCNT0;                   // Nadpisywanie wartości counter wartością z rejestru licznika/timera T0
       TCNT1 = 0;                         // Zerowanie rejestru licznika/timera T1
       }
}

ISR(TIMER1_COMPB_vect)   // Procedura obsługi przerwań kanału B
{
    if ((x==1)&&(counter>=5)&&(t2<15625)) // instrukcja if z warunkami kiedy, (( x==1 & counter>=5)podobnie jak w wcześniejszym przypadku ) & (t2<15632) >>
        {                                 // >> wartość 15632 = 1 sekundzie. czestotliość taktowania 16Mhz/preskalar 1024 = 15625 impulsów zegarowych na sekunde
        LED_1_TOG;                        // zmiana stanu na diodzie czerwonej
        t2=t2+TCNT1;                      // nadpisywanie wartosci t2 wartoscia z rejestru licznika T1, która przy kazdym przerwaniu wynosi 5208
        TCNT1 = 0;                        // zerowanie rejestru licznika T1
        }
}
//*************************************************************************************************************************************************************Petla główna main()
int main( void )
{
    DDRD &=~ SW;                // Rejestr kierunku SW PD2 - wejście
    DDRA |= LED_1;              // Rejestr kierunku LED_1 dioda czerwona na wyjscie
    DDRA |= LED_2;              // Rejest kierunku LED_2 dioda niebieska na wyjscie
    PORTD |= SW;                // Podciagniecie pinu pod VCC (wewnętrzyn rezystor)
    PORTA &=~ LED_1;            // Wyłacznie diody LED_1 dioda czerwona (podłączona anodą do pinu)
    PORTA &=~ LED_2;            // Wyłaczenie diody LED_2 dioda niebieska (podłączona anodą do pinu)

//*************************************************************************************************************************************************************Ustawienie rejestrów kontroli liczniaka T1
    TCCR1A = (1<<FOC1A) | (1<<FOC1B)  ; // Rejestr kontroli licznika/timera 1 TCCR1A, (1<<FOC1A | 1<,FOC1B) Wymyszenie trybu Output Compare dla kanału A | B
    TCCR1B = (1<<CS10) | (1<<CS12) ;   // Rejestr kontroli licznika/timera 1 TCCR1B, (1<<CS10) | (1<<CS12) Ustawienie prestkalacra na 1024

    OCR1A = 46875;                  // Rejestr OCR1A wartośc, która stale jest porównywana z rejestrem licznika TCNT1, kiedy OCR1A==TCNT1 nastepuje przerwanie
    OCR1B = 5208;                   // Rejestr OCR1B wartośc, która stale jest porównywana z rejestrem licznika TCNT1, kiedy OCR1B==TCNT1 nastepuje przerwanie

    TIMSK=(1<<OCIE1A) | (1<<OCIE1B);  // Rejestr maskowania przerwań liczników, (1<<OCIE1A) | (1<<OCIE1B) odblokowanie przerwań output compare A | B
//*************************************************************************************************************************************************************Ustawienie rejestrów kontroli liczniaka T0
    TCCR0 =  (1<<CS01) | (1<<CS02); //  Rejestr kontroli licznika/ timera 0  (1<<CS01) | (1<<CS02) właczenie zewnętrzenego źródła zegara z pinu T0 >>
                                    // zliczające impulsy na zbocze opadające. Utworzone dla licznika zliczajacego do 5
     sei();                         // Właczenie globalnych przerwań

//*************************************************************************************************************************************************************Petla główna programu
while(1)
 {
    if((x==1)&&(counter==5)&&(t2>=15625))   // instrukcja wyłaczajaca program kiedy, x==1, licznik zliczył do 5 oraz t2= 15625(1s)
           {
             PORTA &=~ LED_1;            // Wyłacznie diody LED_1 dioda czerwona (podłączona anodą do pinu)
             PORTA &=~ LED_2;            // Wyłaczenie diody LED_2 dioda niebieska (podłączona anodą do pinu)
             x=0;                        // Wyzerowanie zmiennej x wpisaniem do niej wartości 0
           }

    if (SW_DOWN)                        // instrukcja z makro sprawdzające czy na przycisku jest stan niski
       {
        _delay_ms(100);                 // odczekanie 100ms w celu upewnienia sie czy przycisk wciśniety by wyelminowac drgania styków
        if (SW_UP)                      // instrukcja z makro sprawdzające czy na przycisku jest stan wysoki aby rozpoczac program po zwolnieniu przycisku
           {                            //*************************************************************W instukcji ustawiane sa warunki poczatkowe programu
         x=1;                           //  Nadpisanie zmiennej x wartości 1 niezbedna do włacznie programu
         t2=0;                          // Wyzerowanie zmiennej t2 wpisaniem do niej wartosci 0
         counter=0;                     // Wyzerowanie zmiennej counter wpisaniem do niej wartosci 0
         TCNT0 = 0;                     // Wyzerowanie rejestru kontrolnego licznika/timera T0
         TCNT1 = 0;                     // Wyzerowanie rejestru kontrolnego licznika/timera T1
         PORTA &=~ LED_1;  // Wyłacznie diody LED_1 dioda czerwona
         PORTA |= LED_2;  // Wyłaczenie diody LED_2 dioda niebieska
           }
       }
 }                      //*************************************************************************************************************************************Koniec petli głównej programu
}                      //*************************************************************************************************************************************Koniec petli głównej main ()

```

8. Wnioski:

Podsumowując wykonane ćwiczenie można stwierdzić ze dzięki wykonaniu postawionego celu w poleceniu zadania podczas pisania programu utrwalamy sobie coraz bardziej wcześniej już nabyte umiejętności oraz uczymy się nowych, które z każdym kolejnym razem skutkują szybszym oraz czyściejszym pisaniem kody programy. Funkcja liczników oraz timer należy również do jednych z podstawowych zagadnień bez której nie obejdzie się żaden przyszły programista. Z otrzymanych charakterystyk można stwierdzić że program wykonuje swoje zadanie zgodnie z zaleceniami polecenia.