Systemy wbudowane

Lab3: Obsługa systemu wejścia - wyjścia

Bartnicki Mateusz WCY20IX1N1

Data wykonania ćwiczenia: 28.05.2022 r.

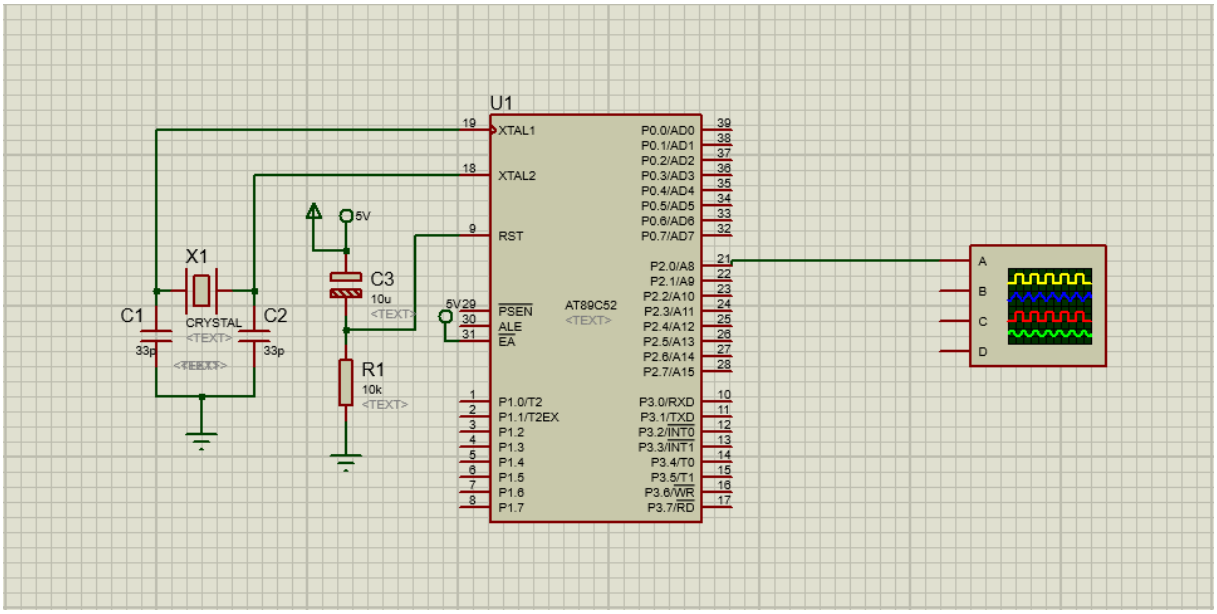
Wykonano zadanie na ocenę dst

Zadanie na dostatecznie.

Na podstawie przykładowego programu ze strony o PWM:

napisać program **zad1.c** w języku C na układ z projektu PWM_1A.pdsprj (w razie potrzeby zbudować taki układ samodzielnie w starszej wersji Proteusa) tak, aby na wyjściu P2_0 uzyskać falę prostokątną: o **okresie** równym około (**10 ms x numer_w_dzienniku**) i współczynniku wypełnienia równym **((3 x numer_w_dzienniku) +10) %**.

W sprawozdaniu należy przedstawić stosowne obliczenia, niezbędne do wykonania zadania oraz zrzuty ekranowe z programu Proteus, wykorzystujące ekran oscyloskopu tak, aby graficznie za pomocą pomiarów wykorzystujących funkcję Cursors oscyloskopu (awaryjnie podstawę czasu oscyloskopu i liczbę kratek, zajmowanych przez wykres badanego przebiegu), udowodnić poprawną realizację zadania. Obliczenia, dotyczące **okresu należy wyróżnić w edytorze zielonym kolorem tła**, a obliczenia dotyczące **współczynnika wypełnienia żółtym kolorem tła**. W podpisie rysunku muszą być zawarte wyniki pomiaru, podane jawnie (nie za pomocą zwrotu - wyniki pomiaru przedstawiono na rysunku!).



Jednym z elementów wykorzystanych w zaimplementowanym programie był Timer0. Jest on ustawiany na timer 16-bitowy, czyli przyjmuje wartości z zakresu od 0 do 65535. Przeładowanie (przerwanie) takiego timera następuje co około 2.26 ms. Dla wartości PWM i PWM_Freq_Num obliczana jest wartość TH0 i TL0 ustawiana na Timerze. Jest to realizowane za pomocą następujących fragmentów kodu w zależności od tego, czy obecnie ma być podawany stan niski lub wysoki.:

```
temp = (255-PWM)*PWM_Freq_Num;
TH0  = 0xFF - (temp>>8)&0xFF;
TL0  = 0xFF - temp&0xFF;
```

```
temp = PWM*PWM_Freq_Num;
TH0  = 0xFF - (temp>>8)&0xFF;
TL0  = 0xFF - temp&0xFF;
```

Czyli, na podstawie PWK i PWM_Freq_Num obliczana jest wartość temp. Następnie TH0 obliczane jest poprzez odjęcie od 0xFF wartości będącej przesunięciem bitowym o 8 w prawo liczby temp oraz wykonaniem bitowej koniunkcji z 0xFF na wyniku przesunięcia.

Jednocześnie w chwili wywołania przerwania Timer jest zatrzymywany, a jego praca jest wznawiana na samym końcu funkcji obsługującej przerwanie wywołane przez Timer.

Zasadniczym problemem do rozwiązania było obliczenie odpowiednich wartości wpisywanej do zmiennej PWM (oznaczającej współczynnik wypełnienia) oraz odpowiedniej wartości PWM_Freq_Num tak, aby uzyskać współczynnik wypełnienia równy 13% oraz okres powtarzania impulsów ok. 10 ms.

Wartość współczynnika wypełnienia możemy wyliczyć korzystając z wiedzy iż PWM = 127 da nam wypełnienie o wartości 0.498, Czyli

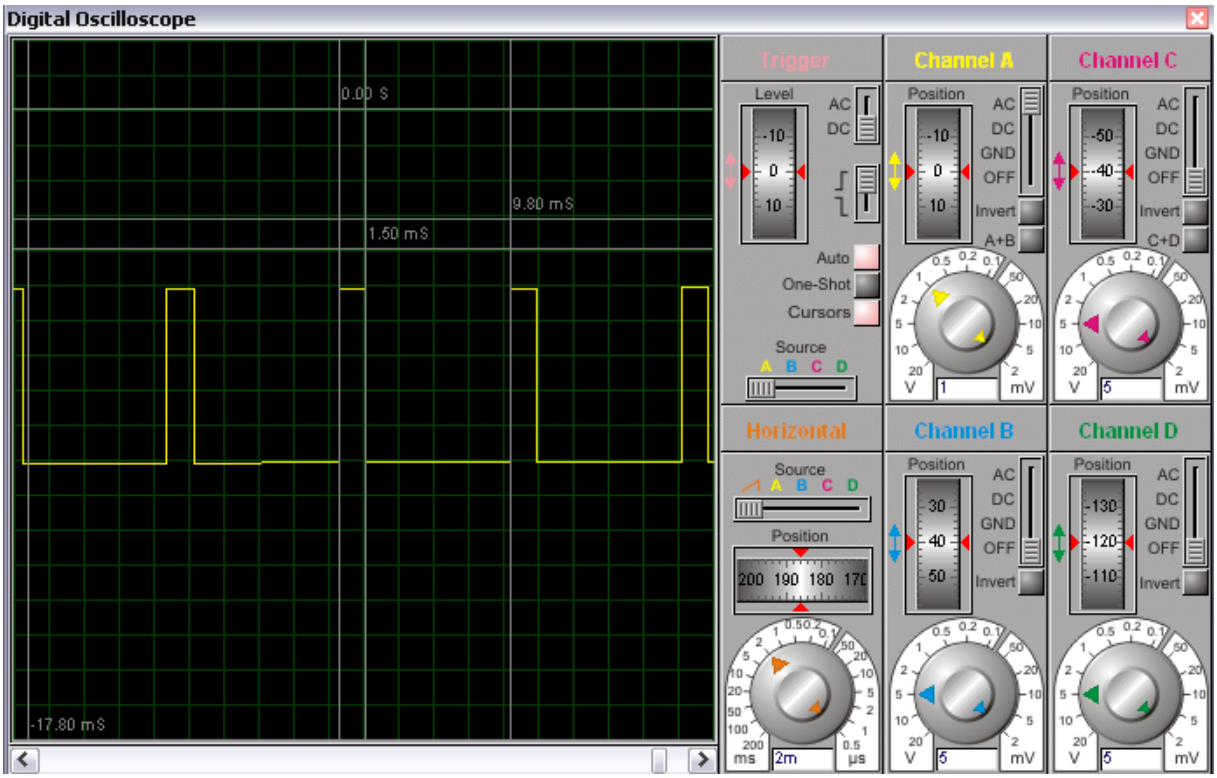
$$127 - 0.498$$
$$x - 0.13$$
$$x = \frac{127 * 0.13}{0.498} \approx 33.15 \approx 33$$

Z powyższych obliczeń wynika, że wartość PWM powinna wynosić 33, żebyśmy uzyskali współczynnik wypełnienia o wartości ok. 13%.

Następnie obliczono wartość PWM_Freq_Num:

$$\frac{65536 - 64384}{255} \approx 4{,}51$$

Zaokrąglając wartość PWM_Freq_Num do 5 uzyskano okres równy 12,2 ms, natomiast zaokrąglając w dół do wartości 4 na oscyloskopie w programie Proteus otrzymano okres równy 9,8 ms. W związku z powyższym postanowiłem wybrać wartość 4.



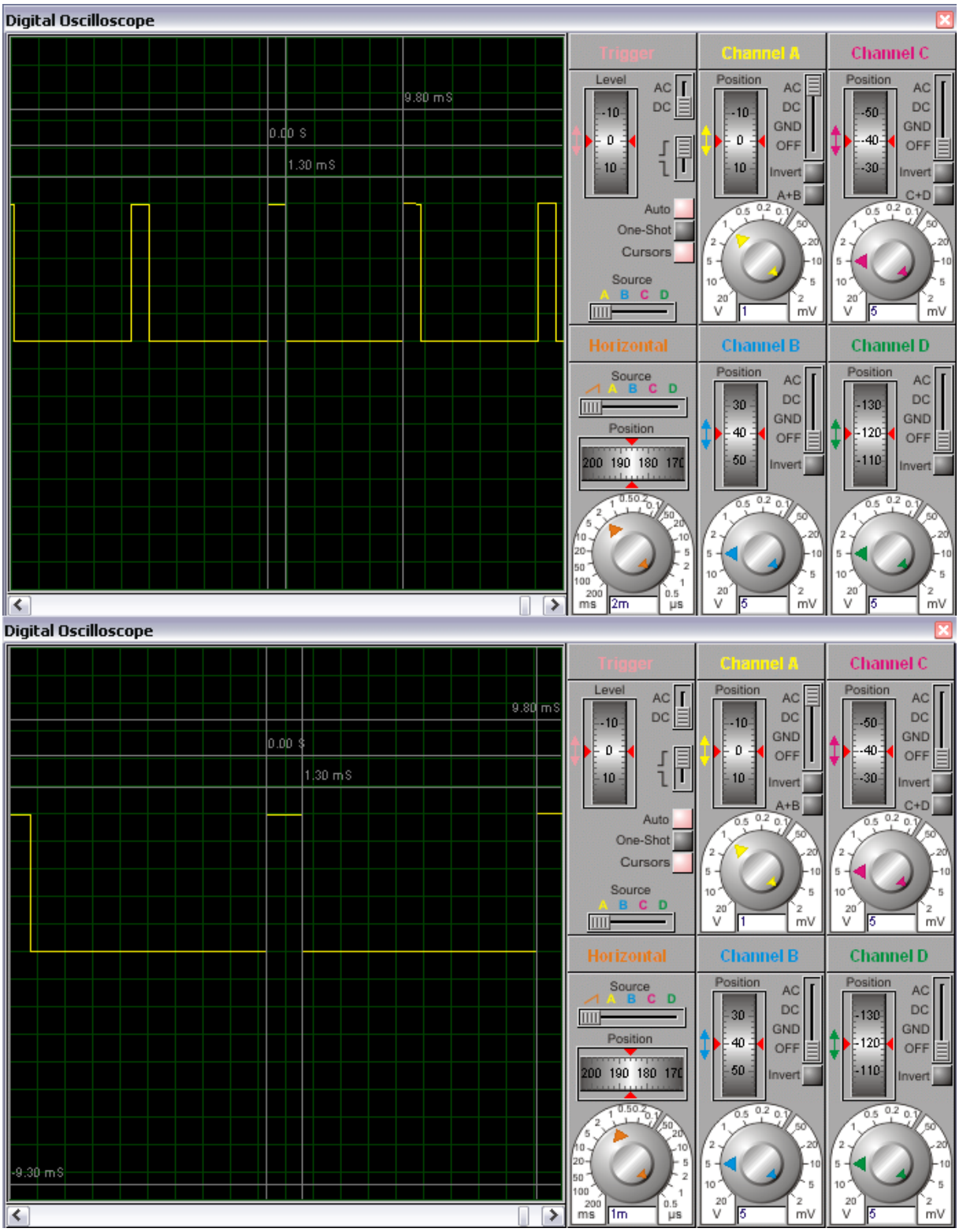
Dla wartości PWM = 33 i PWM_Freq_Num = 4 uzyskano następujące wartości:
Okres: 9.8 ms,
czas trwania stanu wysokiego: 1.5 ms, czas trwania stanu niskiego: 8.5 ms.
W związku z powyższym wartość wypełnienia wynosi:

$$\frac{1.5}{9.8} \approx 0.153$$

Uzyskana wartość jest zdecydowanie zbyt odległa od wartości współczynnika wypełnienia równego 0.13, dlatego też należało zmodyfikować wartość PWM, tak aby uzyskane wypełnienie bardziej zbliżyło się do 0.13. Wynika to z faktu, iż wartości początkowe w rejestrach TH0 i TL0 przyjmują inne wartości przy modyfikacji PWM_Freq_Num. Wynika to z następującego fragmentu kodu:

```
TH0  = 0xFF - (temp>>8)&0xFF;
TL0  = 0xFF - temp&0xFF;
```

Po wprowadzeniu modyfikacji do kodu ostateczna wartość PWM powinna wynosić 26. Potwierdza to przebieg oscyloskopu w programie Proteus.

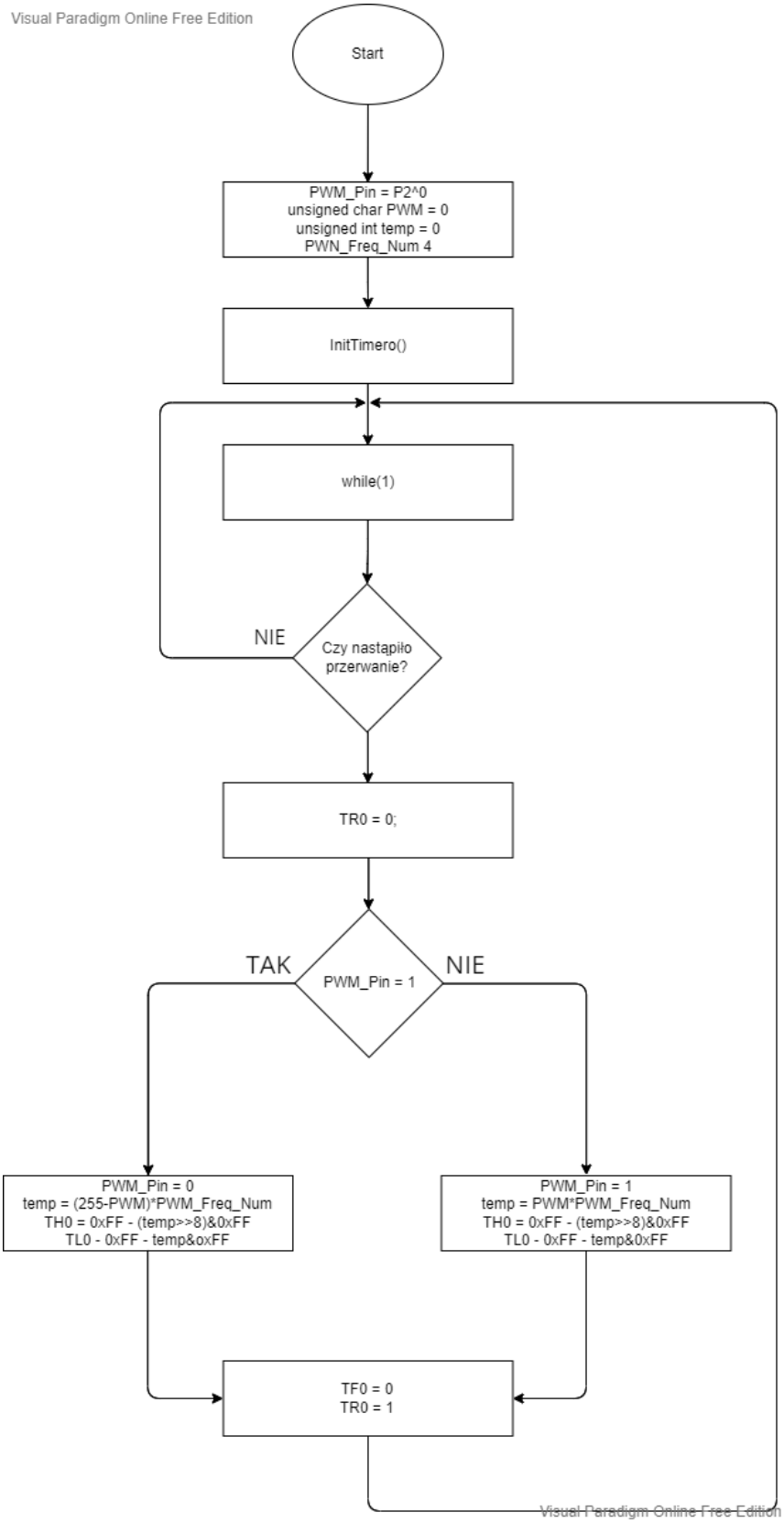


Dla wartości PWM = 26 i PWM_Freq_Num = 4 uzyskano następujące wartości:
Okres: 9.8 ms,
czas trwania stanu wysokiego: 1.3 ms, czas trwania stanu niskiego: 8.5 ms.
W związku z powyższym wartość wypełnienia wynosi:

$$\frac{1.3}{9.8} \approx 0.132$$

Uzyskany współczynnik wypełnienia równy 0.132 jest wynikiem jak najbardziej zadowalającym.

Schemat blokowy zad1.c opracowany za pomocą Visual Paradigm Online:



Treść programu:

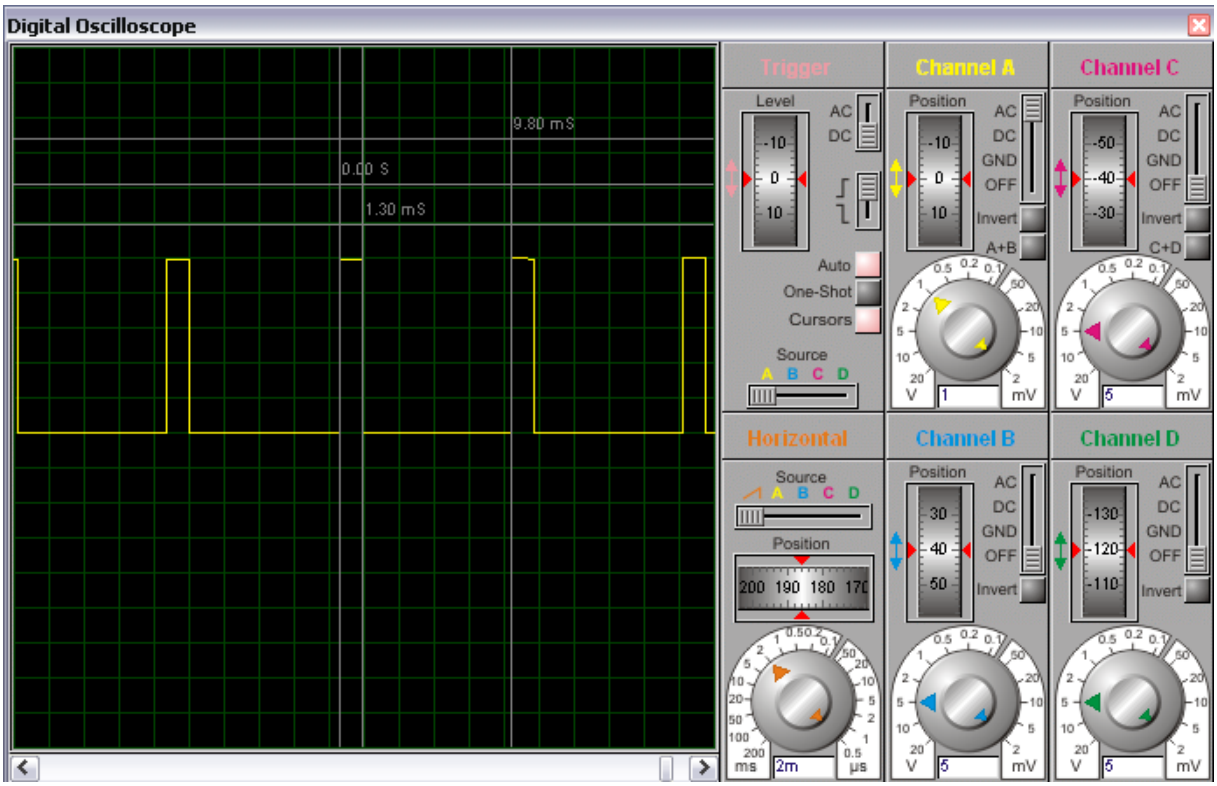
zad1.c

```c
#include <REGX52.H>
// PWM_Pin
sbit PWM_Pin = P2^0;       // Pin P2.0 to PWM_Pin
// deklaracje
void InitTimer0(void);
void InitPWM(void);
// zmienne globalne
unsigned char PWM = 0;     // wartosc od 0 (0% duty cycle) do 255 (100% duty cycle)
unsigned int temp = 0;     // zmienna robocza w procedurze obslugi przerwania Timer0
#define PWM_Freq_Num   4    //  1 = najwyzsza czestotliwosc gdy PWM_Freq_Num, zakres 1 - 255
// Main Function
int main(void)
{
   InitPWM();              // Start PWM
   PWM = 26;              // 127 = 50% wspolczynnik wypelnienia

   while(1) {;}
}
// Timer0 init
void InitTimer0(void)
{
  TMOD &= 0xF0;     // wyzeruj bity dla Timer0
  TMOD |= 0x01;     // ustaw tryb mode 1 = 16bit mode

  TH0 = 0x00;       // Pierwsze
  TL0 = 0x00;       // ustawienie

  ET0 = 1;          // Enable Timer0 interrupts
  EA  = 1;          // Enable All

  TR0 = 1;          // Start Timer 0
}

// PWM init
void InitPWM(void)
{
  PWM = 0;          // poczatkowo zero
  InitTimer0();     // Init Timer0 dla rozpoczecia generacji przerwan
}
// Timer0 ISR
void Timer0_ISR (void) interrupt 1
{
  TR0 = 0;     // Stop Timer 0

  if(PWM_Pin) // if PWM_Pin =1 wyzeruj sygnal PWM i zaladuj licznik
  {
    PWM_Pin = 0;
    temp = (255-PWM)*PWM_Freq_Num;
    TH0  = 0xFF - (temp>>8)&0xFF;
    TL0  = 0xFF - temp&0xFF;
  }
  else       // if PWM_Pin =0 ustaw pin na 1 i zaladuj licznik
  {
    PWM_Pin = 1;
    temp = PWM*PWM_Freq_Num;
    TH0  = 0xFF - (temp>>8)&0xFF;
    TL0  = 0xFF - temp&0xFF;
  }
  TF0 = 0;      // wyczysc flage
```

```c
  }
  TF0 = 0;      // wyczysc flage
  TR0 = 1;      // Start Timer 0
}
```

Zrzuty ekranu z kompilacji i linkowania pokazujące brak błędów oraz rozmiar kodu i danych:

Zrzut ekranu z symulatora Proteus:



Okres: 9.8 ms,
czas trwania stanu wysokiego: 1.3 ms, czas trwania stanu niskiego: 8.5 ms.
W związku z powyższym wartość wypełnienia wynosi:

$$\frac{1.3}{9.8} \approx 0.132$$