



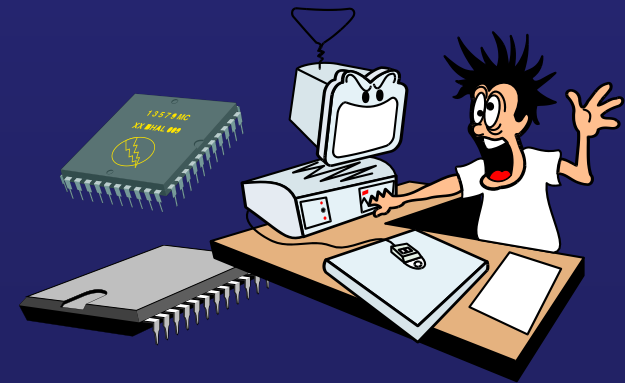
WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki



Systemy wbudowane



Witold Kozłowski



<https://std2.phys.uni.lodz.pl/mikroprocesory/>

Systemy wbudowane

Kierunek: Informatyka
PRACOWNIA DYDAKTYCZNA

Uwaga !!!

**Proszę o wyłączenie
telefonów komórkowych**

na wykładzie i laboratorium

Wykład 3.

**Zastosowanie licznika-czasomierza
Timer0 do generowania stałych
odcinków czasu**

**Przykład generowania sygnału
PWM wykorzystując Timer1**

**Jak
wygenerować
odcinek czasu o
dowolnej
długości**

???

Timer0

8 bitowy

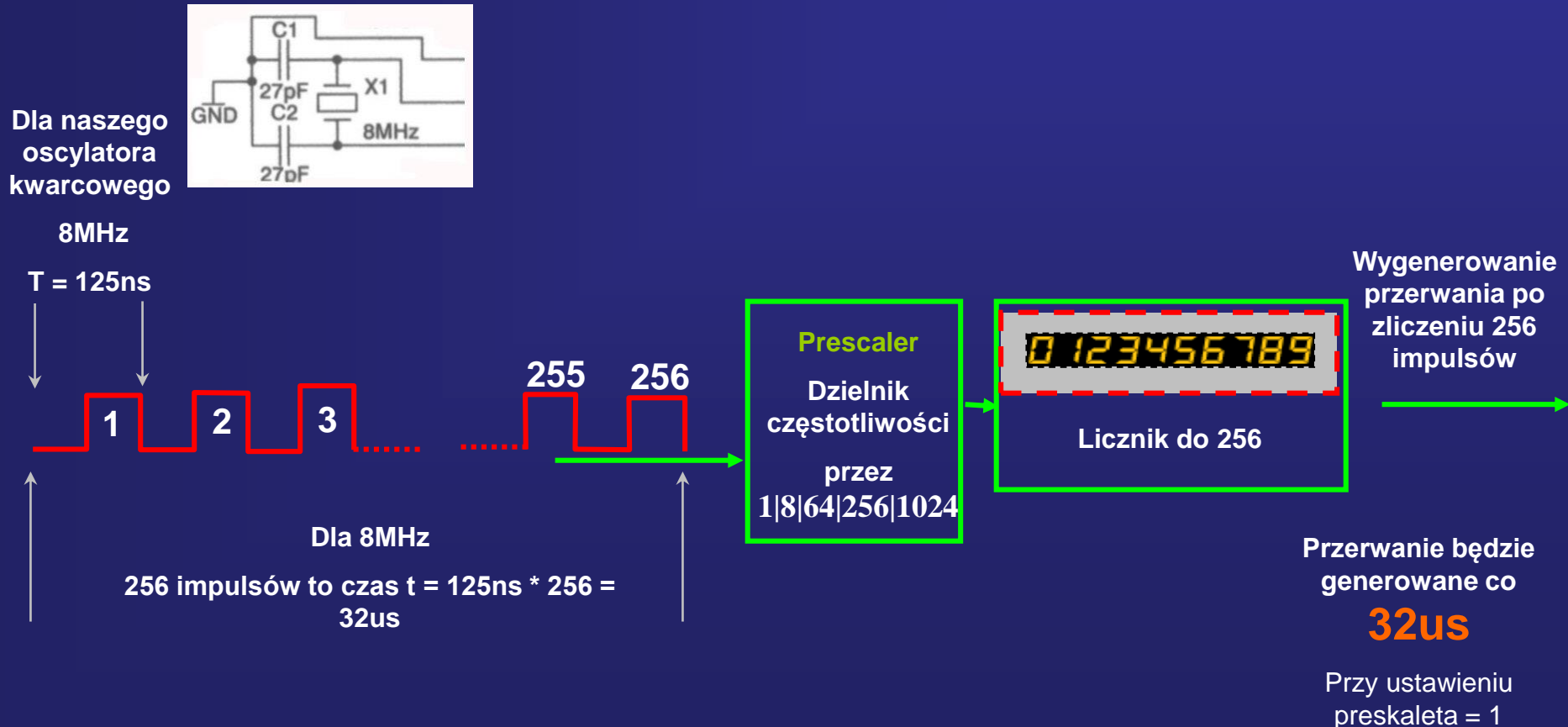
Każdy mikrokontroler jest wyposażony w co najmniej jeden timer, który można zazwyczaj skonfigurować do pracy w trybach licznika, czasomierza czy generatora PWM.

Dużą zaletą timerów jest to, że mogą pracować niezależnie od innych bloków funkcjonalnych mikrokontrolera – mikrokontroler pracuje w swoim rytmie a timer w swoim.

licznik-czasomierz Timer0

Licznik TIMER0 jest 8 bitowy- tzn może zliczyć tylko 256 impulsów .

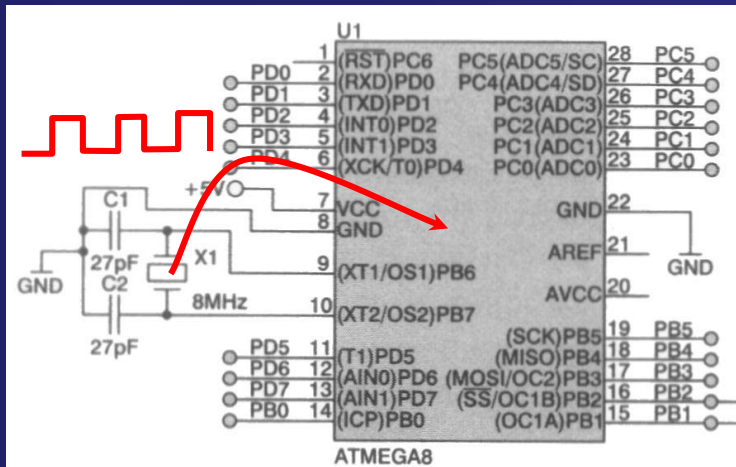
Może on zliczać impulsy zegara taktującego procesor doprowadzone do jego wejścia bezpośrednio lub przez prescaler. Zliczanie można w każdej chwili zatrzymać i wznowić oraz wpisać „dowolną” wartość do licznika.



Konfiguracja licznika-czasomierza Timer0

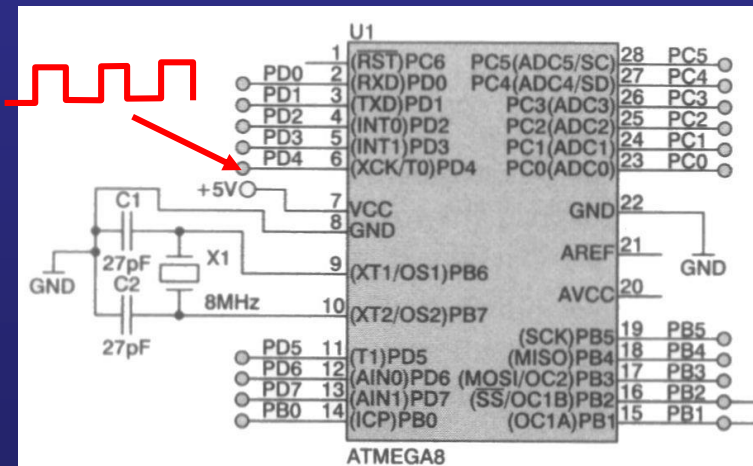
Konfiguracja jako czasomierz:

Config Timer0 = Timer , Prescale = 1|8|64|256|1024



Konfiguracja jako licznik:

Config Timer0 = Counter , Prescale = 1|8|64|256|1024
Edge = Rising|Falling, Clear Timer = 1|0



licznik-czasomierz Timer0

Licznik czasomierz TIMER0 cechuje się dużą rozdzielczością i wysoką dokładnością gdy używany jest przy małych stopniach podziału preskalera. Podobnie, przy dużym podziale preskalera licznik staje się użyteczny przy odmierzaniu dłuższych odcinków czasu.

Konfiguracją pracy licznika zajmuje się instrukcja *Config Timer0*

Do sterowania licznikiem przewidziano instrukcje *Start* oraz *Stop*

Uproszczono także dostęp do rejestrów licznika definiując w języku BASCOM BASIC specjalne instrukcje
Counter0, Load

Na przykład instrukcja:

Counter0 = 206 – spowoduje, że do licznika Timer0 zostanie wpisana wartość początkowa 206

lub:

Load - która dokonuje niezbędnego przeliczenia tej wartości, tak aby licznik przepelniał się po podanej liczbie impulsów. Jest ona szczególnie użyteczna w tych przypadkach gdy wymagane staje się “skracanie” cyklu licznika.

Na przykład instrukcja:

Load Timer0, 50 - spowoduje, że do licznika Timer0 zostanie wpisana wartość początkowa 206, a więc licznik przepelni się właśnie po 50 impulsach ($206+50=256$)

Przewidziano także stosowanie przez użytkownika przerw jakie generuje licznik. Można je łatwo obsłużyć stosując instrukcję *ON INTERRUPT* w połączeniu z odpowiednim programem obsługi. Zgłaszanie przerw przez licznik można włączać i wyłączać za pomocą instrukcji *ENABLE* i *DISABLE*

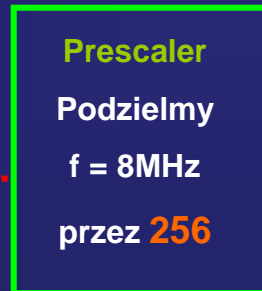
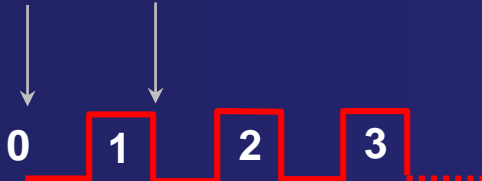
Wykorzystamy Timer0 - czasomierz

Jak odmierzyć 1s przy pomocy Timera0 ?

Dla naszego
oscylatora
kvarcowego

$f = 8\text{MHz}$

$T = 125\text{ns}$



Po podziale

$f = 31.25\text{kHz}$

$T = 32\mu\text{s}$



Uwaga:

należy ustawić licznik
tak aby liczył do

250



Wygenerowanie
przerwania po
zliczeniu 250
impulsów

Przerwanie będzie
generowane co

8ms

Wykorzystamy Timer0 - czasomierz

Jak odmierzyć 1s przy pomocy Timera0 ?

To jest naprawdę proste !!!

Należy ustawić tak licznik aby liczył do

250



Wygenerowanie przerwania po zliczeniu 250 impulsów



Zliczenie 125 kolejnych przerwani które będą występować co 8ms da nam właśnie 1s

$$125 * 8ms = 1s$$

Przerwanie będzie generowane co

8ms

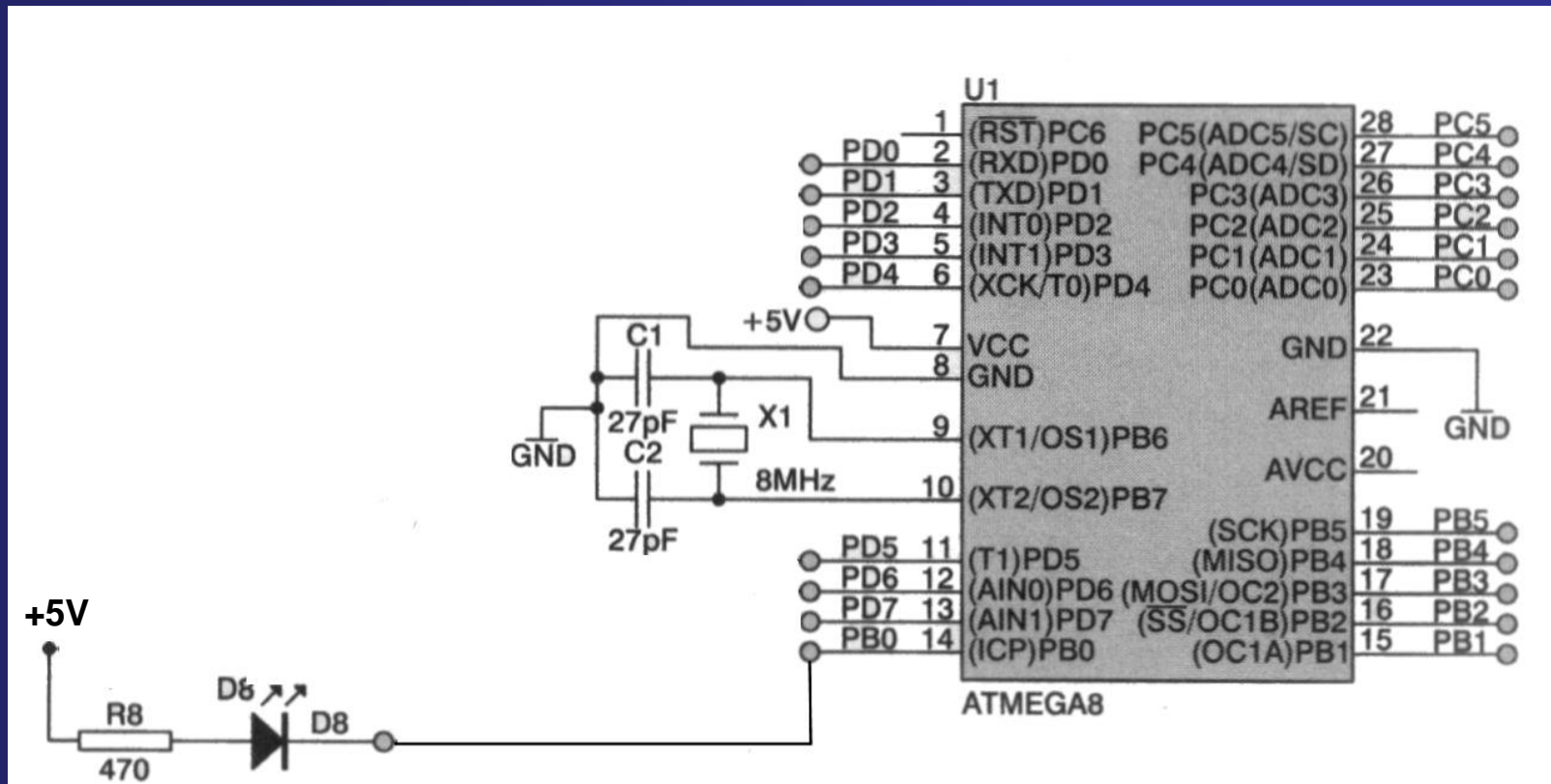
Program 3

Przykład wykorzystania Timer0 do generowania przerwań.

Odmierzanie 1 sekundowych odcinków czasu poprzez zliczenie ilości przerwań Timera0

Program 3

Schemat połączenia diody LED do linii PBO portu B mikrokontrolera



Przykład wykorzystania Timer0 do generowania przerwań.

Program 3

```
D:\Mikroprocesory\Bascom Colege\basAVR_listingi\8...
Sub
Label

$regfile = "m8def.dat"
$crystal = 8000000

Config Pinb.0 = Output
Config Timer0 = Timer, Prescale = 256

On Timer0 Odmierz 1s

Dim Licz_8ms As Byte

Enable Interrupts
Enable Timer0
Load Timer0 = 250

Do
Loop

End

Odmierz 1s:

Load Timer0 = 250
Incr Licz_8ms

If Licz_8ms = 125 Then

Licz_8ms = 0
Toggle Portb.0

End If
Return
```

informuje kompilator o pliku dyrektyw mikrokontrolera oraz częstotliwości oscylatora

konfiguracja linii PB0 jako wyjścia

konfiguracja Timer0 jako timera z podziałem preskalera przez 256

przerwanie od przepełnienia Timer0 o etykiecie Odmierz_1s

zmienna pomocnicza zliczająca odcinki czasu równe 8ms

odblokowanie globalnego systemu przerwań
odblokowanie przerwania od przepełnienia Timer0

wpisanie wartości początkowej 6 do Timer0 (256-250)

Program główny początek i koniec pętli

koniec programu

początek podprogramu obsługi przerwania od przepełnienia Timer0

wpisanie wartości początkowej 6 do Timer0 (256-250)

zwiększ o jeden wartość zmiennej pomocniczej Licz_8ms

jeżeli wartość tej zmiennej równa 125 ($125 \cdot 8 \text{ ms} = 1$) to odliczono 1 sekundę

zerowanie zmiennej licznikowej

zmień na przeciwny stan linii Pb0 portu B

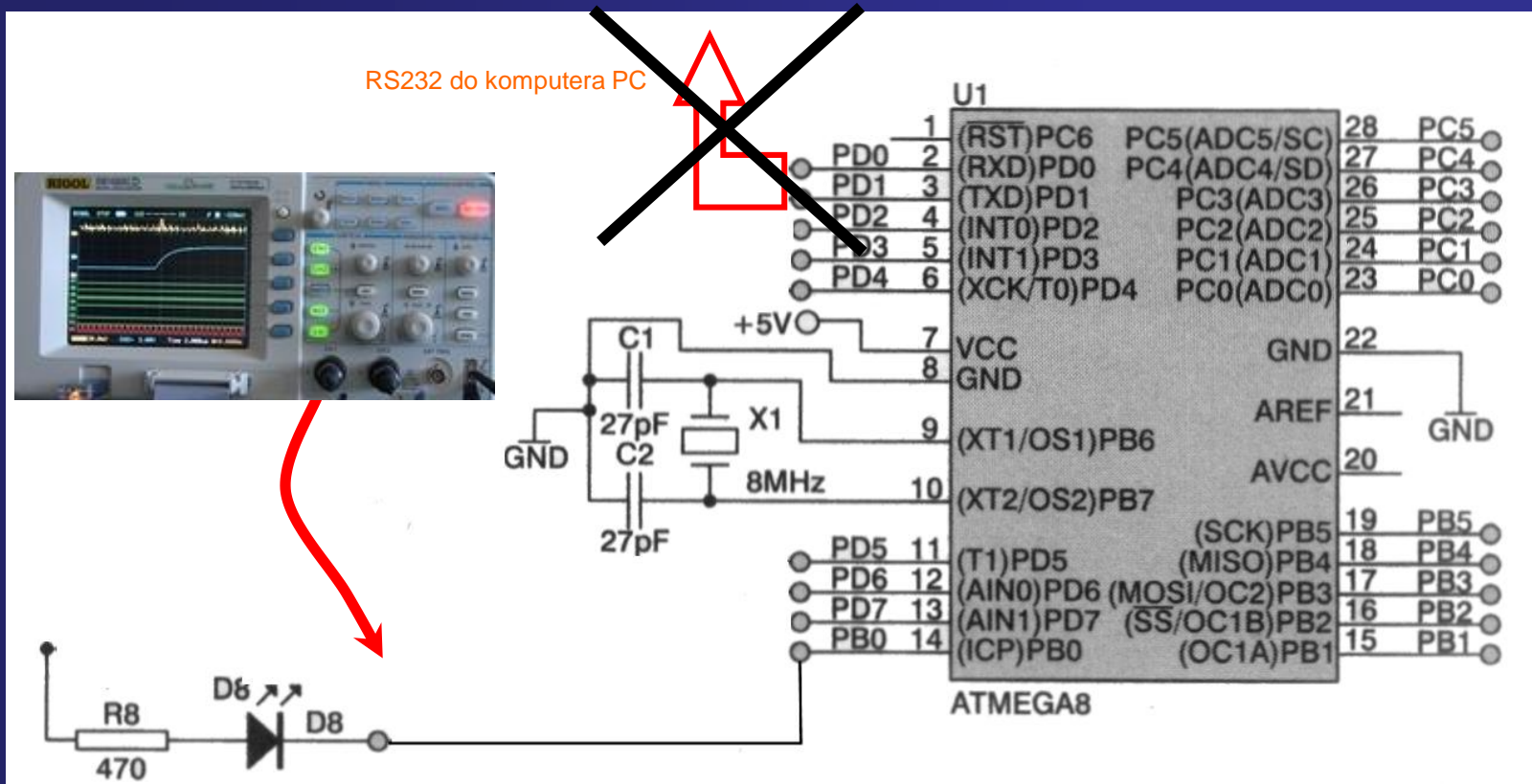
koniec warunku If...Then

powrót z przerwania

Program3

Przebadanie zależności czasowych podczas rzeczywistej pracy mikrokontrolera realizującego program3.

Odmierzanie 1 sekundowych odcinków czasu poprzez zliczenie ilości przerwań Timera0



Timer1

16 bitowy

Licznik TIMER1 jest 16 bitowy i może zliczać impulsy zegara taktującego procesor doprowadzone do jego wejścia bezpośrednio lub przez prescaler (65536 impulsów).

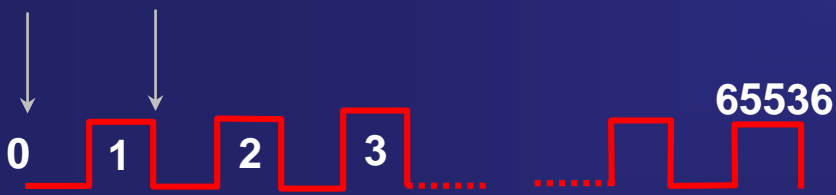
Może też z powodzeniem zliczać impulsy doprowadzone do jednej z końcówek portów

Timer1

Dla naszego
oscylatora
kvarcowego

8MHz

$T = 125\text{ns}$



Dla 8MHz

65535 impulsów to czas $t = 125\text{ns} * 65536$
 $= 8.192\text{ms}$

Można:

Wpisywać zmienne do licznika
lub je porównywać



Generowanie
przerwań

OC1A

OC1B

Port PB1

Port PB2)

W wyniku porównania
zmieniany jest stan
odpowiednich wyjść
OC1A (PB1) lub
OC1B(PB2)

Timer1

czasomierz - licznik

Konfiguracja jako czasomierz:

Config Timer1 = Timer , Prescale = 1|8|64|256|1024

Konfiguracja jako licznik:

Config Timer1 = Counter , Prescale = 1|8|64|256|1024

Edge = Rising|Falling,

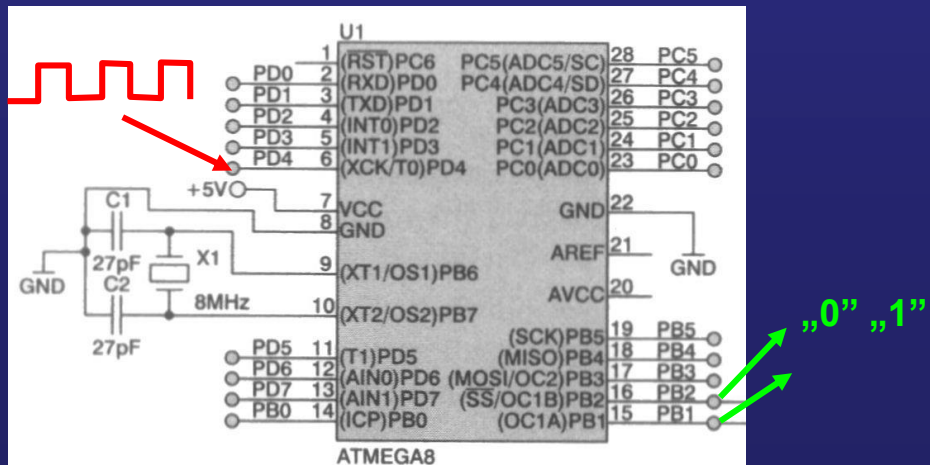
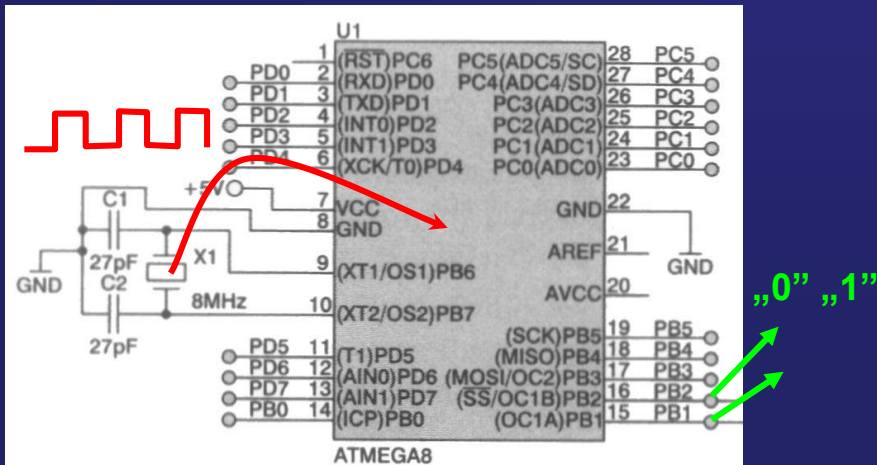
Noise Cancel=0|1

Capture Edge = Rising|Falling

Compare A = Clear|Set|Toggle|Disconnect

Compare B = Clear|Set|Toggle|Disconnect

Clear Timer = 1|0,



Timer1

Generatorów impulsów o regulowanym wypełnieniu - Pulse Width Modulation

PWM

Konfiguracja jako PWM:

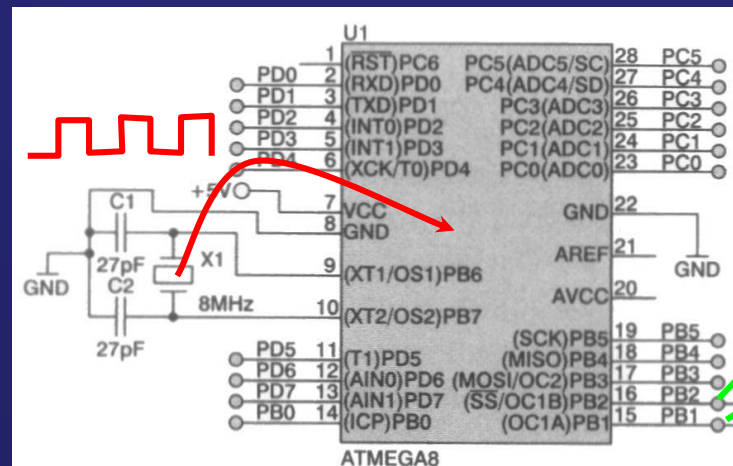
Config Timer1 = PWM ,

Prescale = 1|8|64|256|1024,

PWM = 8|9|10,

Compare A PWM = Clear Up|Clear Down|Disconnect

Compare B PWM = Clear Up|Clear Down|Disconnect

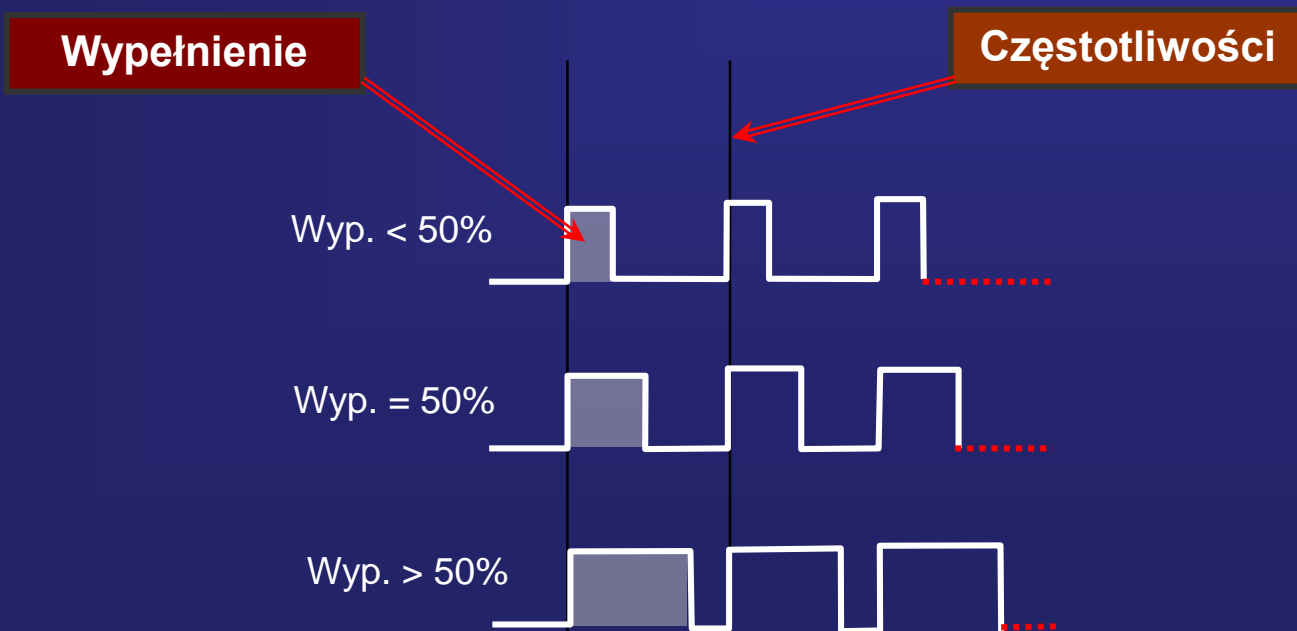


zmienne

Pwm1a, Pwm1b

Wykorzystamy Timer1 do generowania sygnału PWM

Sygnał PWM ma przebieg prostokątny o zmiennym wypełnieniu. Wypełnienie może być zmieniane od 0 do 100%. Przy wypełnieniu 50% otrzymuje się symetryczny przebieg prostokątny (poziom wysoki „1” trwa tak długo jak poziom niski „0”).



Wykorzystamy Timer1 do generowania sygnału PWM

Uwaga:

Celem ćwiczenia będzie generowanie przebiegu prostokątnego PWM o różnym wypełnieniu i częstotliwości.

Każdy w grupie będzie musiał zaprogramować Timer1 aby generował określony sygnał PWM.

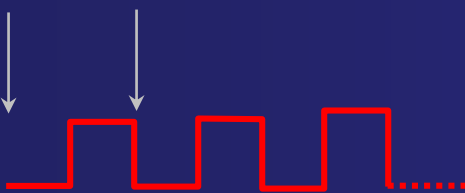
Parametry sygnału dla każdej osoby w grupie podane są w tabeli:

Timer1 PWM

Dla naszego
oscylatora
kvarcowego

8MHz

T = 125ns



Ustawiamy rozdzielczość licznika:

8,9,10 bitowy

$$PWM = 8/9/10$$



Ustawiamy kierunek zliczania licznika a
tym samym kierunek wypełnienia
impulsu:

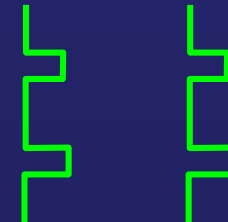
*Compare A PWM = Clear Up/Clear
Down/Disconnect*



OC1A

OC1B

W zależności od wartości zmiennej
Pwm1a, Pwm1b na
wyprowadzeniach **OC1A (port PB1)**
lub **OC1B (port PB2)** generowany
jest przebieg o różnym wypełnieniu



Timer1 PWM

Jak konfigurować Timer1 aby generował sygnał PWM

np: 15,68kHz i 20% wypełnienia

Config Timer1 = PWM , Prescale = 1|8|64|256|1024,

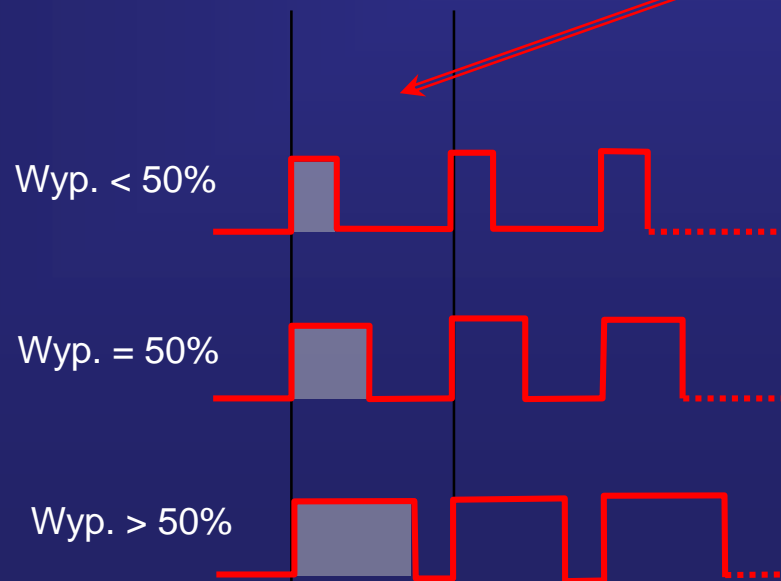
PWM = 8|9|10,

Compare A PWM = Clear Up|Clear Down|Disconnect

Compare B PWM = Clear Up|Clear Down|Disconnect

Jak regulować częstotliwość sygnału PWM

Rozdzielczość PWM	Wartość maksymalna licznika	Częstotliwość sygnału PWM
8-bitów	255	$f_c / \text{Prescaler} / 510$
9-bitów	511	$f_c / \text{Prescaler} / 1022$
10-bitów	1023	$f_c / \text{Prescaler} / 2046$



Jak regulować wypełnienie sygnału PWM

Config Timer1 = PWM , Prescale = 1, PWM = 8, Compare A PWM = Clear Down, Compare B PWM = Disconnect

Na przykład:

Ustawiamy rozdzielczość licznika jako 8 bitowy

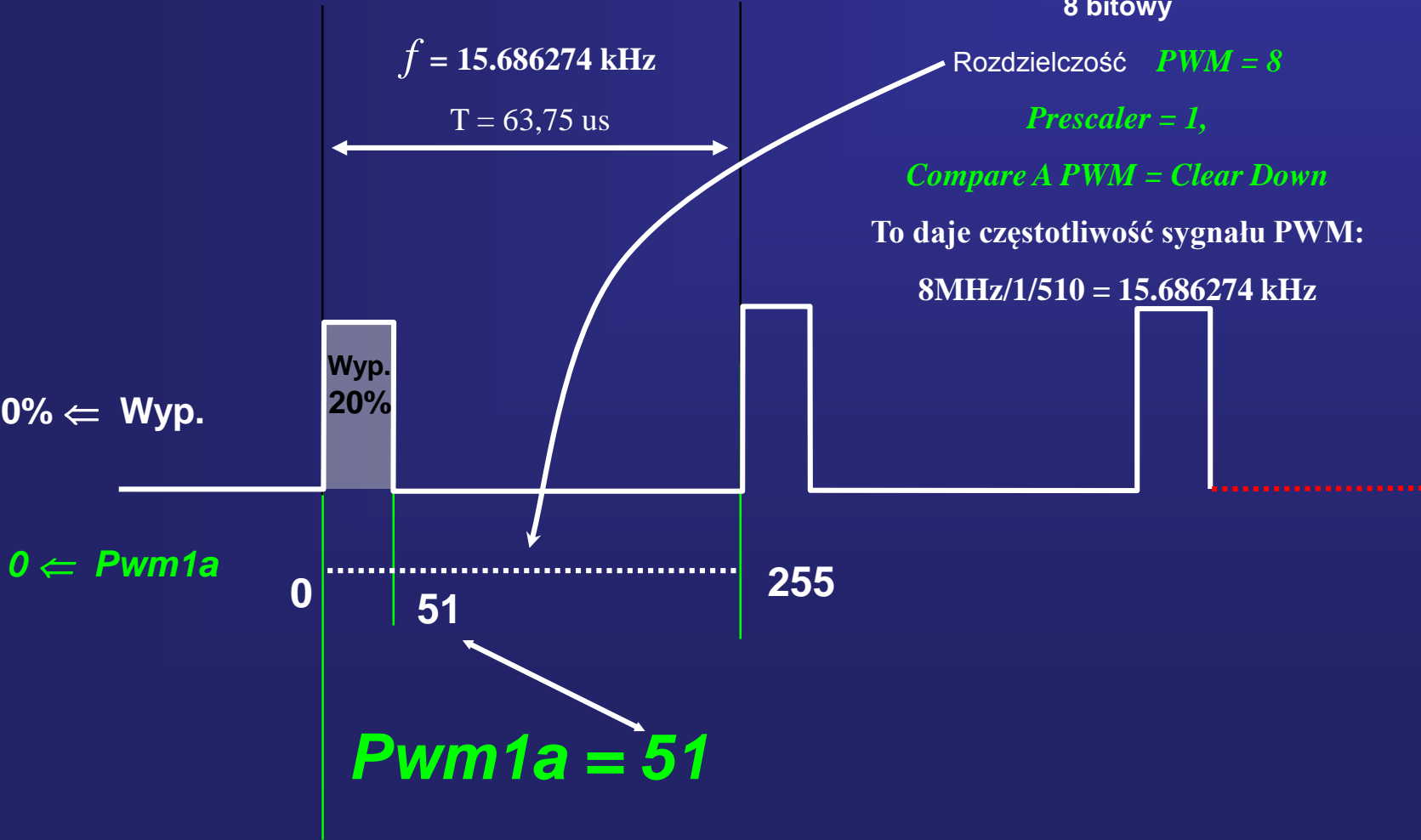
Rozdzielczość *PWM = 8*

Prescaler = 1,

Compare A PWM = Clear Down

To daje częstotliwość sygnału PWM:

$$8\text{MHz}/1/510 = 15.686274 \text{ kHz}$$



Jak regulować wypełnienie sygnału PWM

Na przykład:

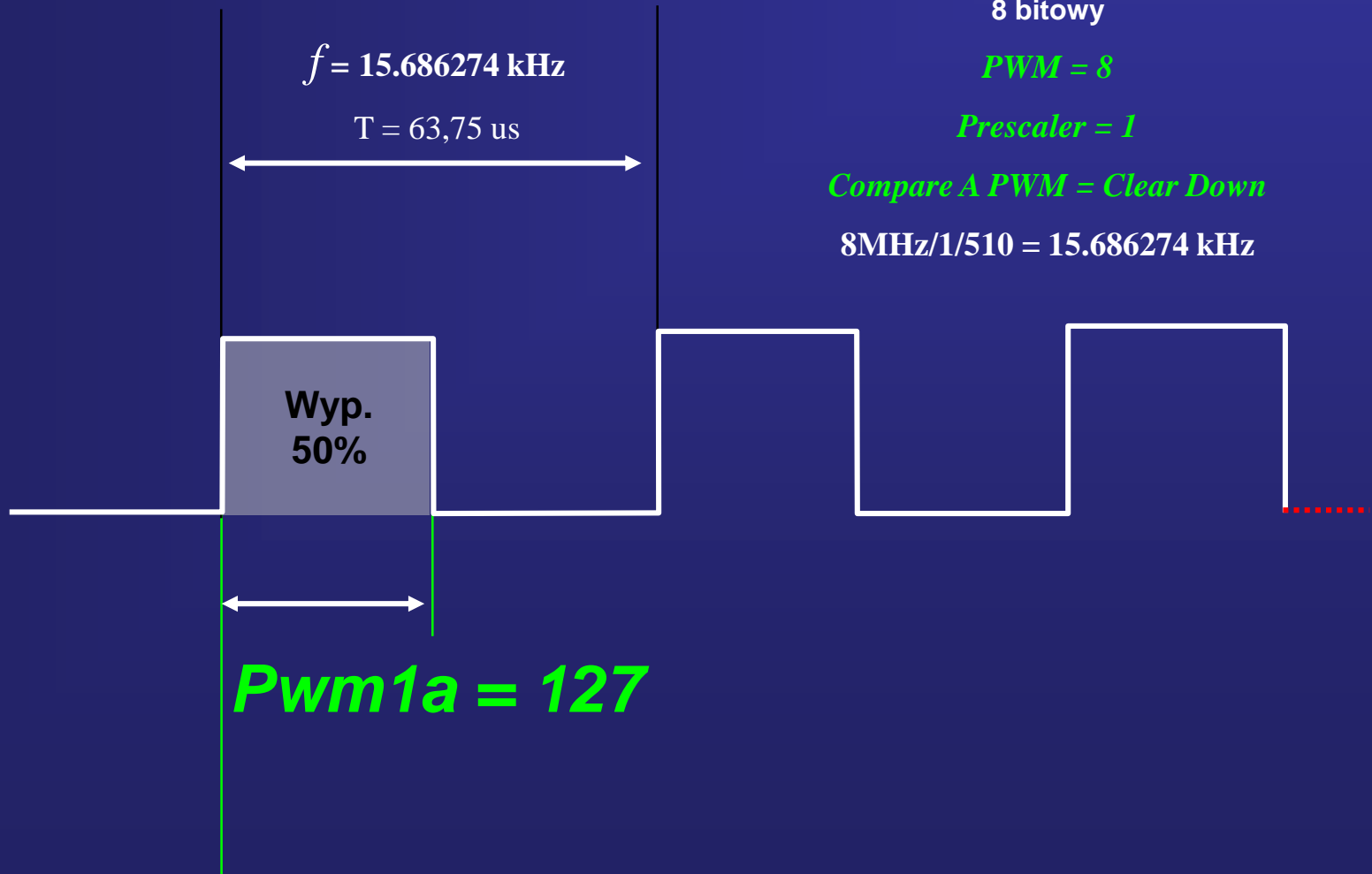
Ustawiamy rozdzielczość licznika jako
8 bitowy

$PWM = 8$

$Prescaler = 1$

$Compare\ A\ PWM = Clear\ Down$

$8MHz/1/510 = 15.686274\ kHz$



Jak regulować wypełnienie sygnału PWM

Na przykład:

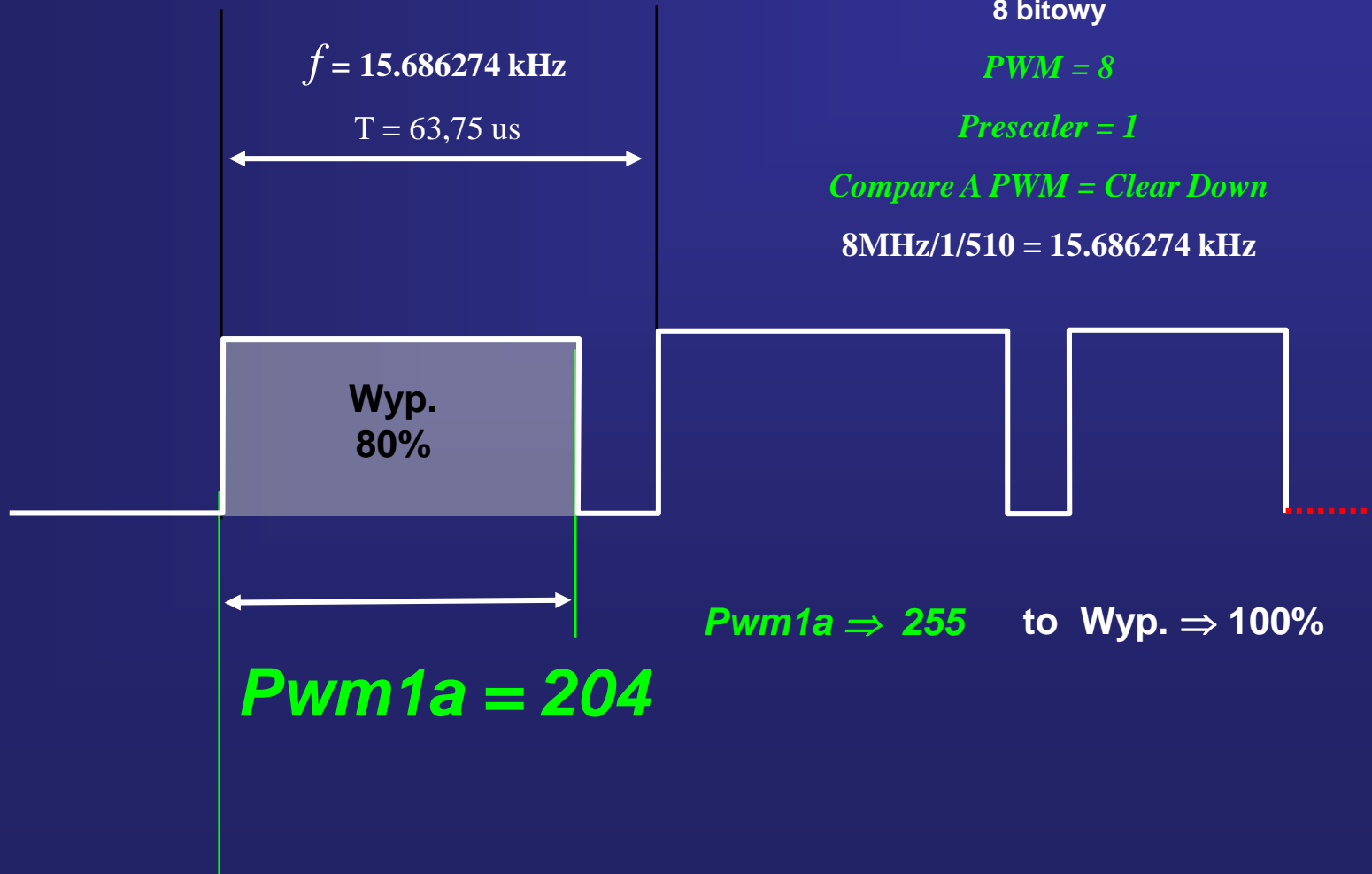
Ustawiamy rozdzielczość licznika jako
8 bitowy

$PWM = 8$

$Prescaler = 1$

$Compare\ A\ PWM = Clear\ Down$

$8MHz/1/510 = 15.686274\ kHz$



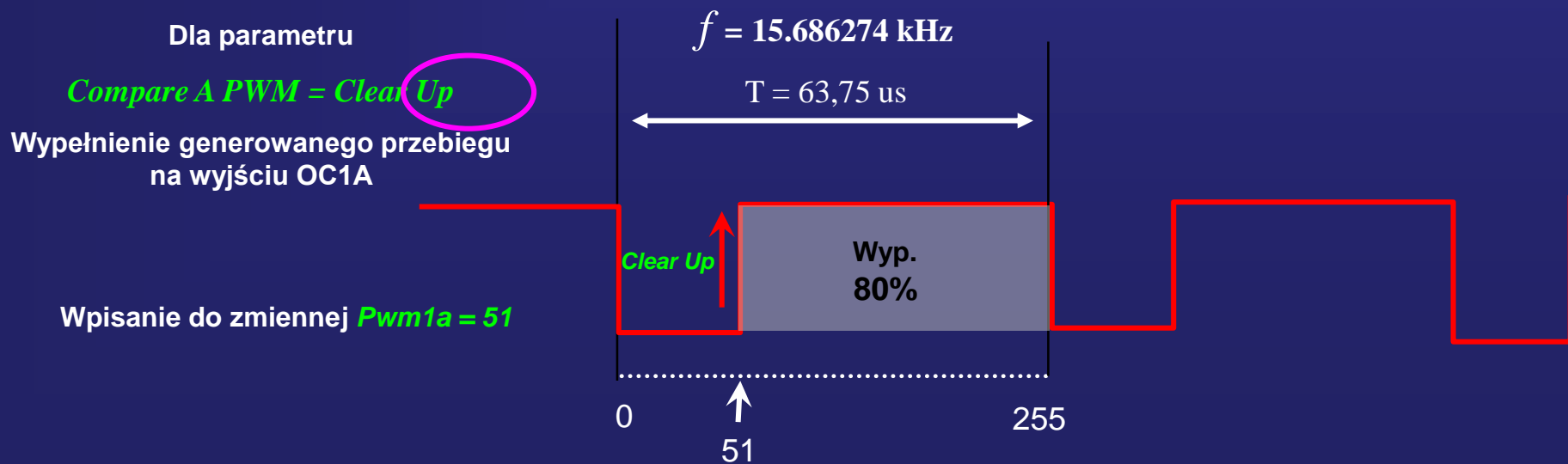
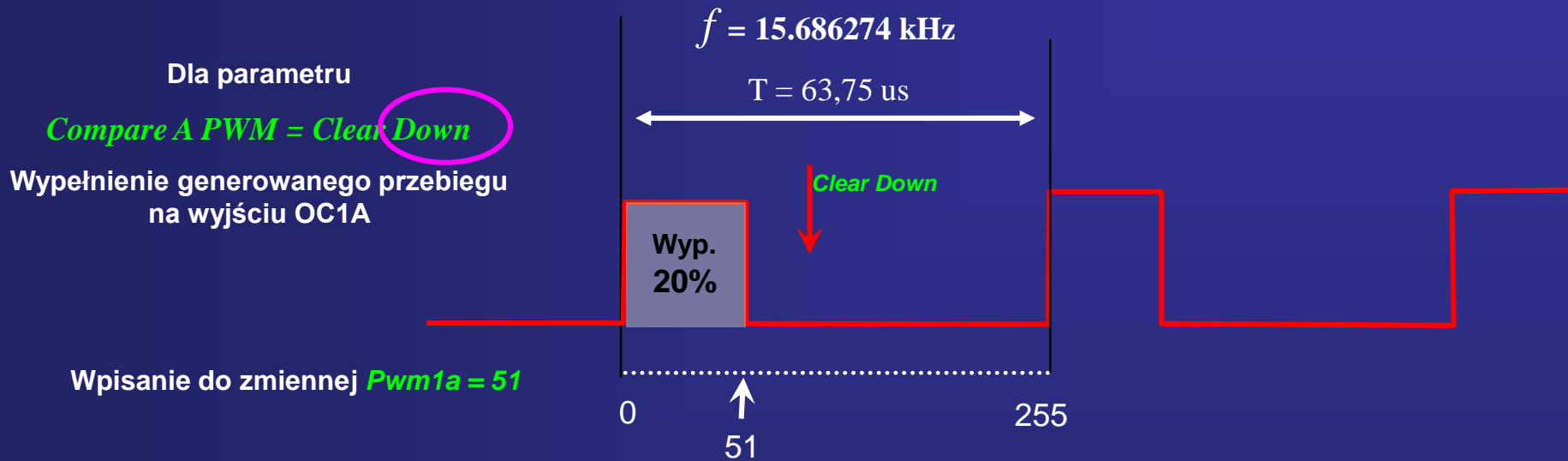
$Pwm1a \Rightarrow 255$ to Wyp. $\Rightarrow 100\%$

$Pwm1a = 204$

Jak regulować wypełnienie sygnału PWM

Rozdzielczość PWM	Wartość maksymalna licznika	Wypełnienie PWM
8-bitów	255	255 - 100% x - 10% x =
9-bitów	511	511 - 100% x - 10% x =
10-bitów	1023	1023 - 100% x - 10% x =

Jak regulować wypełnienie sygnału PWM



Jak regulować wypełnienie sygnału PWM

Dla parametru

Compare A PWM = Clear Up

Wypełnienie generowanego przebiegu na OC1A będzie tym mniejsze, im większa wartość jest wpisana do zmiennej

Pwm1a

Wpisanie do zmiennej *Pwm1a* = 10

Wyp. > 50%

Wpisanie do zmiennej *Pwm1a* = 127
(polowa 256)

Wyp. = 50%

Wpisanie do zmiennej *Pwm1a* = 250

Wyp. < 50%

Dla parametru

Compare A PWM = Clear Down

Wypełnienie generowanego przebiegu na OC1A będzie tym większe im większa wartość jest wpisana do zmiennej

Pwm1a

Wpisanie do zmiennej *Pwm1a* = 250

Wpisanie do zmiennej *Pwm1a* = 127
(polowa 256)

Wpisanie do zmiennej *Pwm1a* = 10

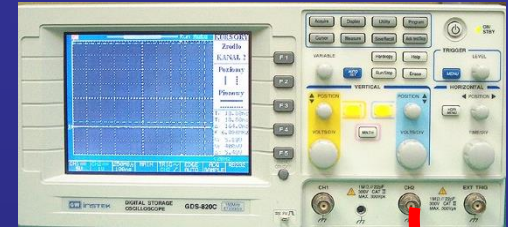
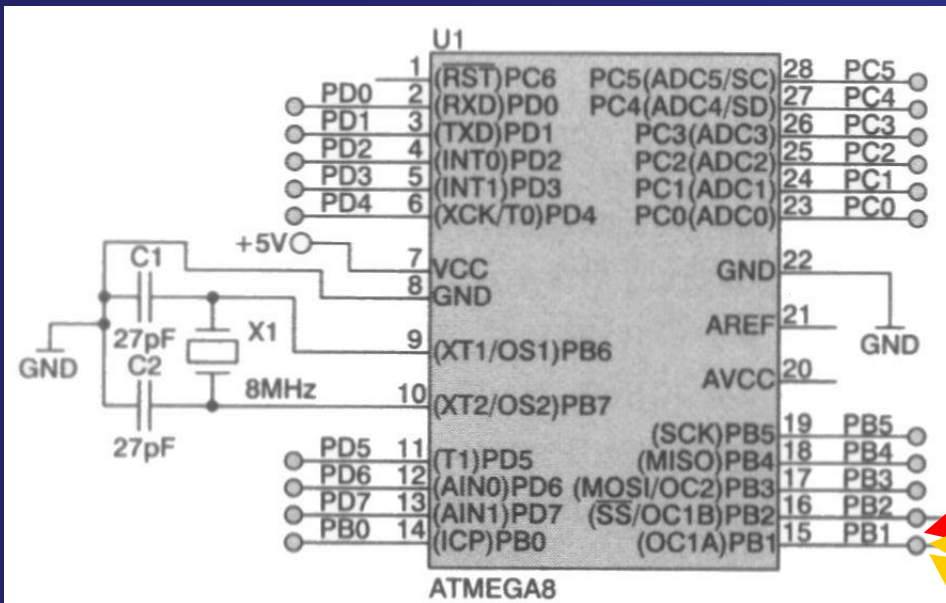
↑
Wzrost wypełnienia



Program 4

Zaprogramowanie sygnału PWM o określonej częstotliwości i wypełnieniu

Testowanie Timer1 do generowania sygnału PWM



Program 4

```
D:\Mikroprocesory\Bascom Colege\basAVR_listingi\Dziala_zaj...
Sub          Label
$regfile = "m8def.dat"
$crystal = 8000000

Config Pinb.1 = Output

Config Timer1 = Pwm , Pwm = 8 ,
                Compare A Pwm = Clear Up ,
                Compare B Pwm = Disconnect ,
                Prescale = 1

Pwm1a = 10

End
```

informuje kompilator o pliku dyrektyw mikrokontrolera

informuje kompilator o częstotliwości oscylatora taktującego mikrokontroler

linia PB1 jako wyjście

konfiguracja Timer1 jako generatora sygnału PWM na wyjściu OC1A

wpisanie do zmiennej Pwm1a wartości 10 określającej wypełnienie sygnału na wyjściu OC1A

koniec programu

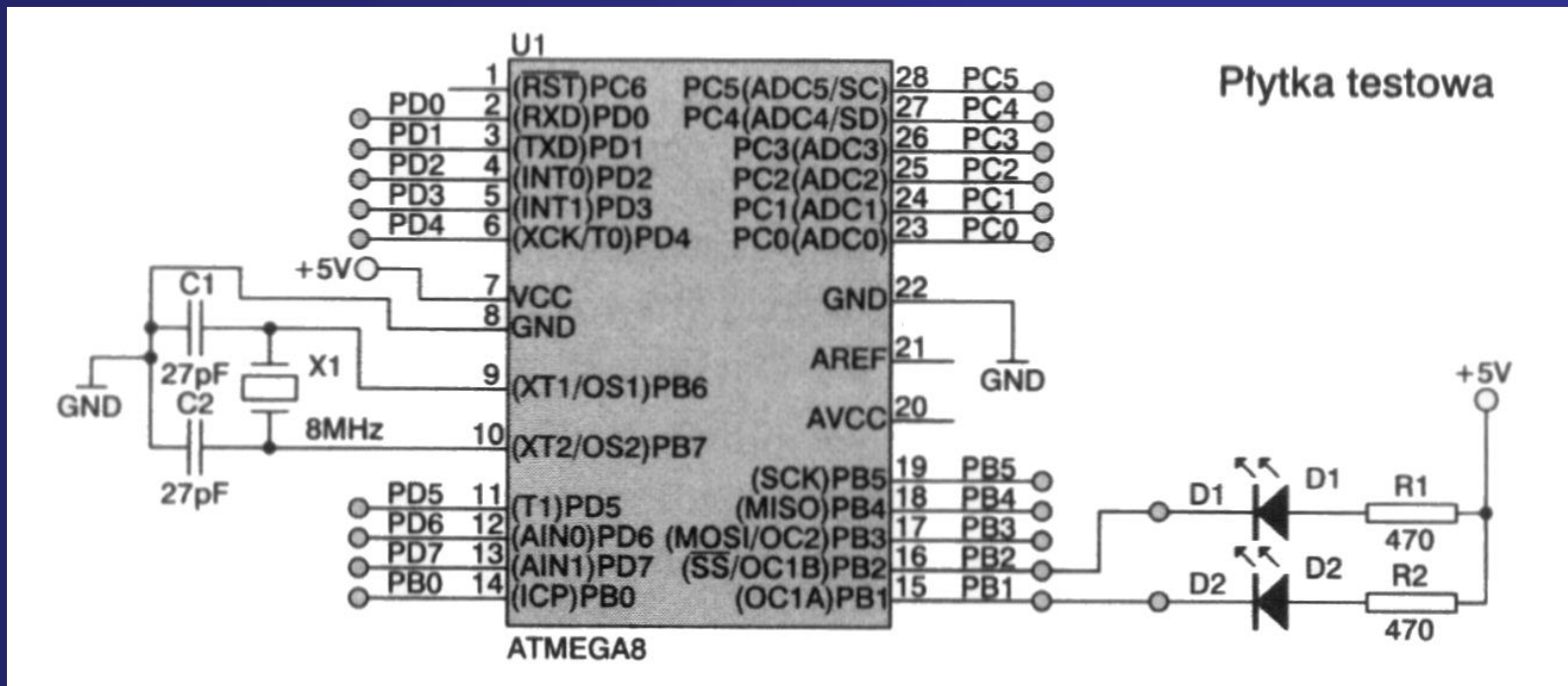
Program 5

Program sterowania jasnością dwóch diod LED
za pomocą sygnału PWM

Rozdzielczość sygnałów PWM 8 bitów
Częstotliwość sygnału PWM ok. 15,6 kHz

Program 5

Schemat połączenia diod LED do linii PB1 i PB2 portu B mikrokontrolera



Program sterowania jasnością dwóch diod LED za pomocą sygnału PWM

Program 5

```
D:\Mikroprocesory\Bascom Colege\basAVR_listingi\Dziala_z...
Sub
$regfile = "m8def.dat"
$crystal = 8000000

Config Pinb.1 = Output
Config Pinb.2 = Output

Config Timer1 = Pwm , Pwm = 8
                Compare A Pwm = Clear Up
                Compare B Pwm = Clear Down
                Prescale = 1

Dim I As Byte

Do
  For I = 0 To 255
    Pwm1a = I
    Pwm1b = I
    Waitms 4
  Next I

  For I = 255 To 0 Step -1
    Pwm1a = I
    Pwm1b = I
    Waitms 4
  Next I
Loop
End
```

informuje kompilator o pliku dyrektyw mikrokontrolera

informuje kompilator o częstotliwości oscylatora taktującego mikrokontroler

linia PB1, PB2 jako wyjścia

konfiguracja Timer1 jako generatora dwóch sygnałów PWM na wyjściach OC1A i OC1B

zmienna licznikowa dla pętli For...Next

początek pętli

pętla wykonywana 256 razy

wpisanie do zmiennej Pwm1a wartości I określającej wypełnienie sygnału na wyjściu OC1A

czekaj 4 ms

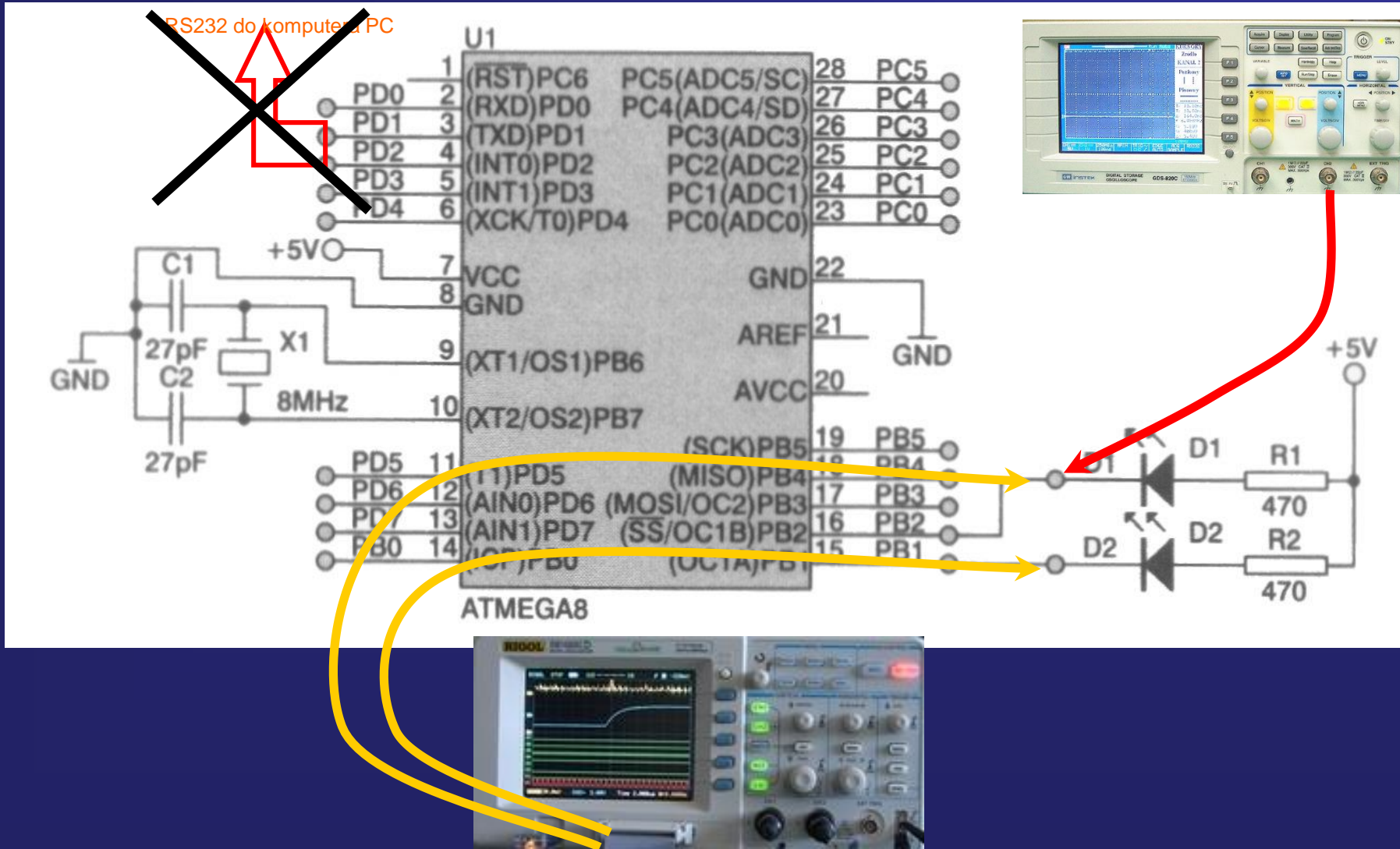
pętla wykonywana 256 razy
zmniejszająca o 1 wartość I

koniec pętli głównej programu

koniec programu

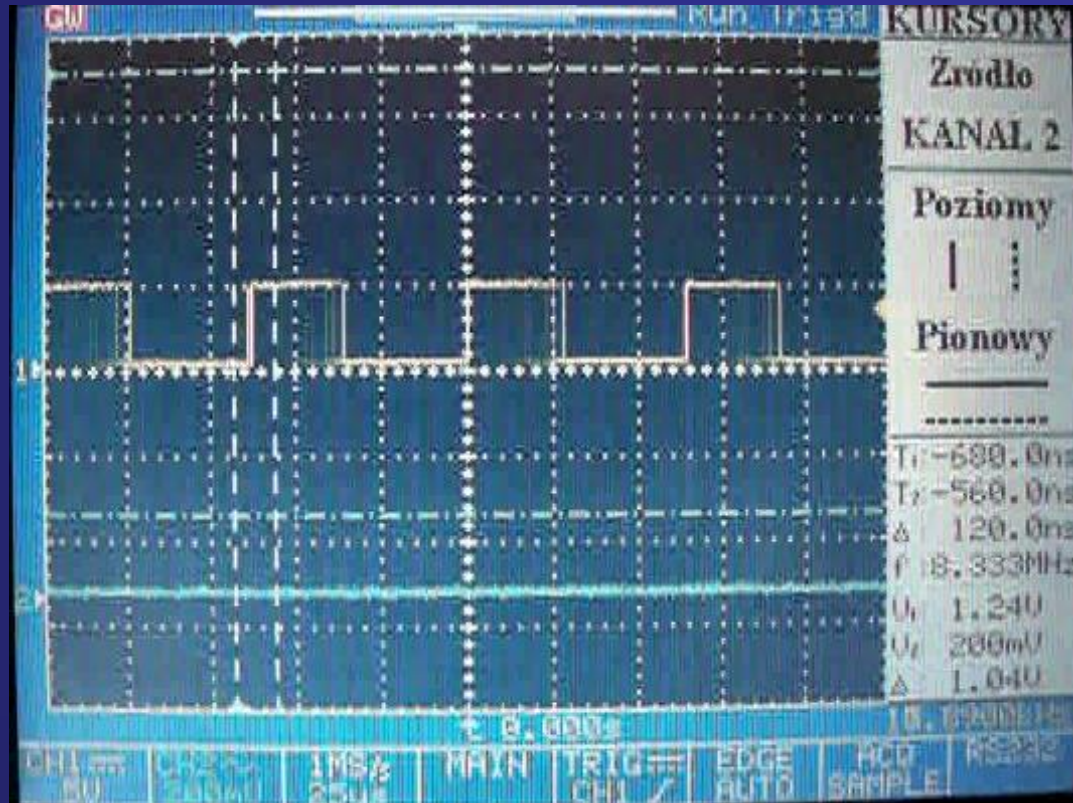
Program 5

Przebadanie zależności czasowych podczas rzeczywistej pracy mikrokontrolera realizującego program 5

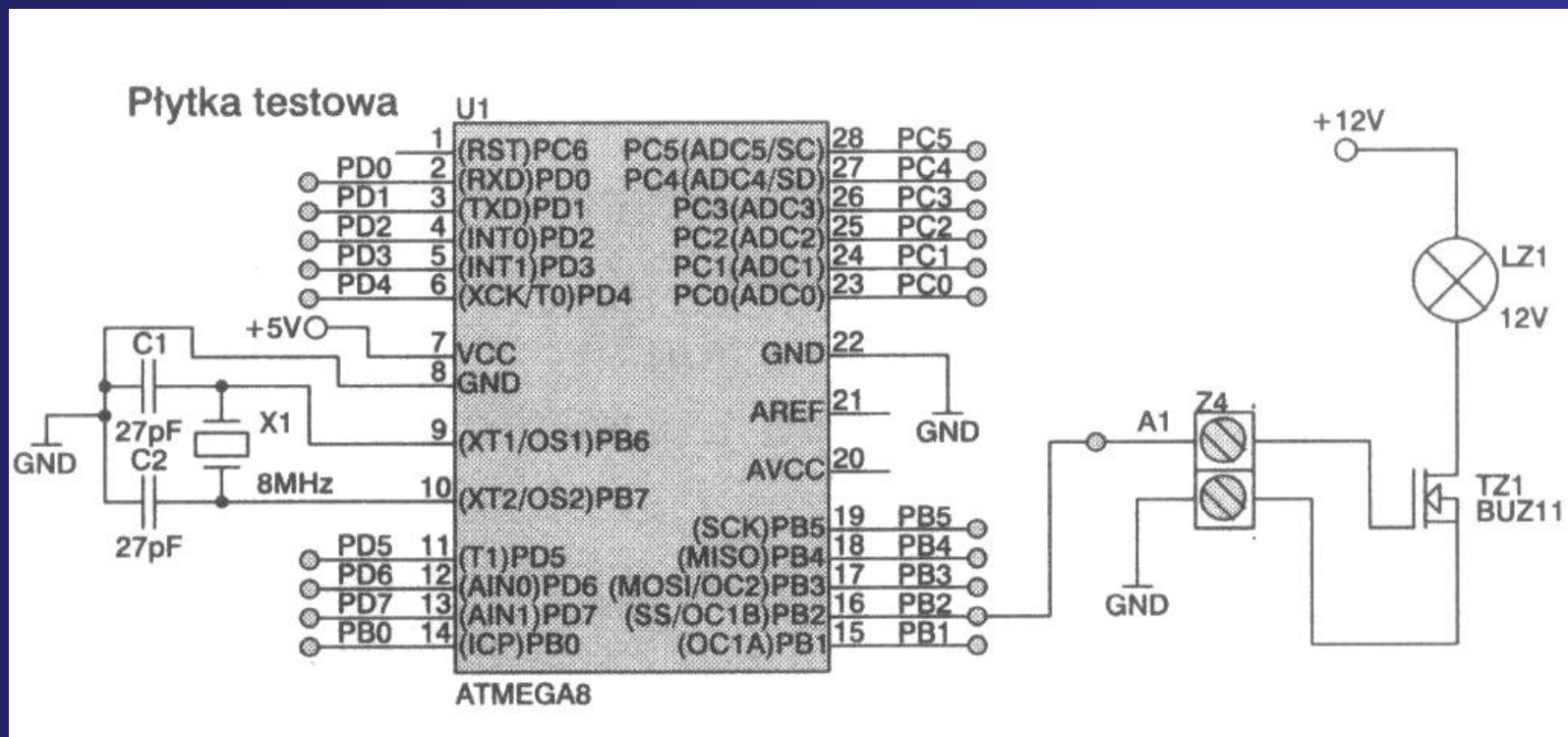


Program 5

Przebadanie zależności czasowych sygnału PWM podczas rzeczywistej pracy mikrokontrolera realizującego program5

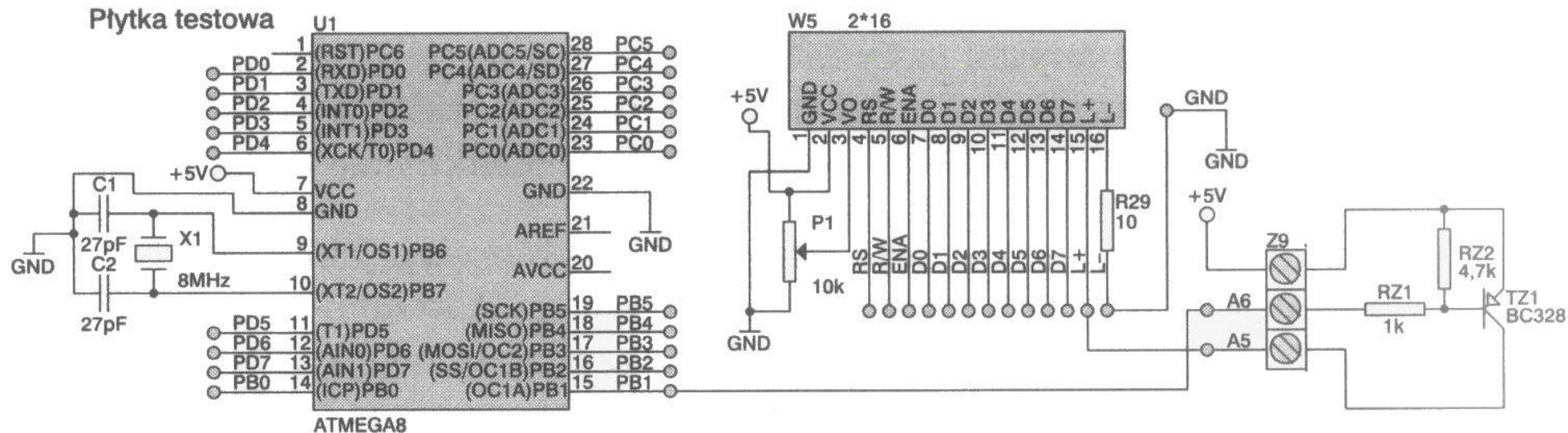


Do czego można wykorzystać sygnał PWM



Schemat układu sterowania jasności świecenia żarówki samochodowej o napięciu 12V za pomocą sygnału PWM

Do czego można wykorzystać sygnał PWM



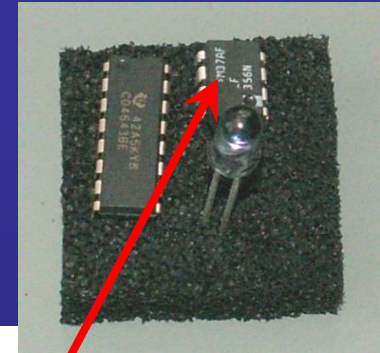
Schemat układu sterowania jasnością podświetlania wyświetlacza LCD za pomocą sygnału PWM

Na koniec
Program 5a
dodatkowy

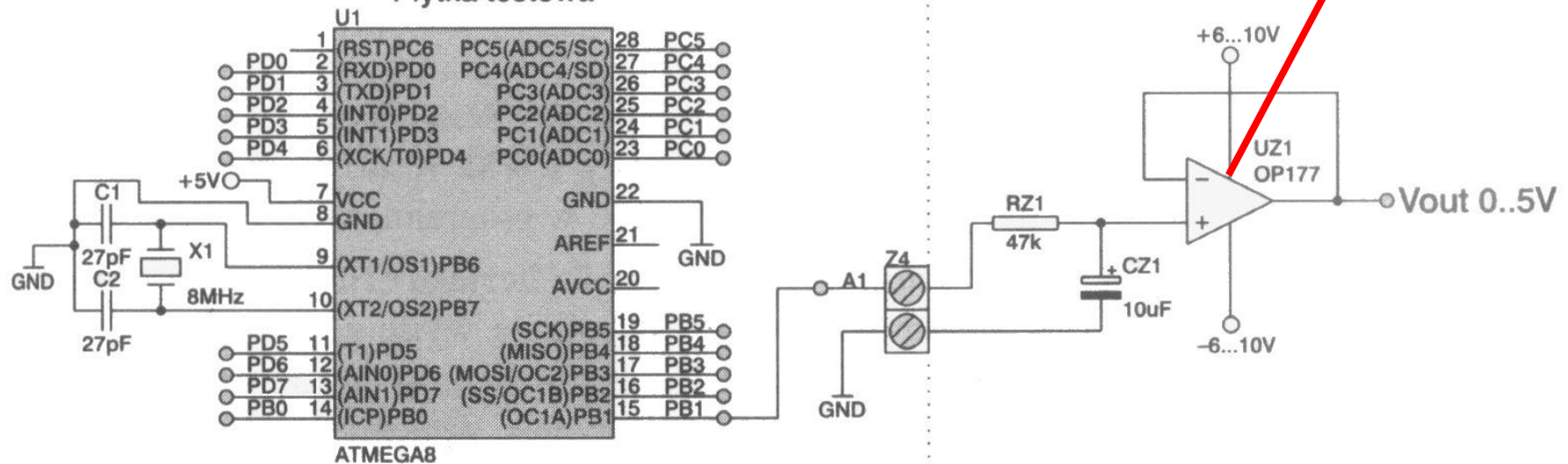
Zastosowanie sygnału PWM w
przetworniku C/A

Program 5a

Zastosowanie sygnału PWM w przetworniku C/A

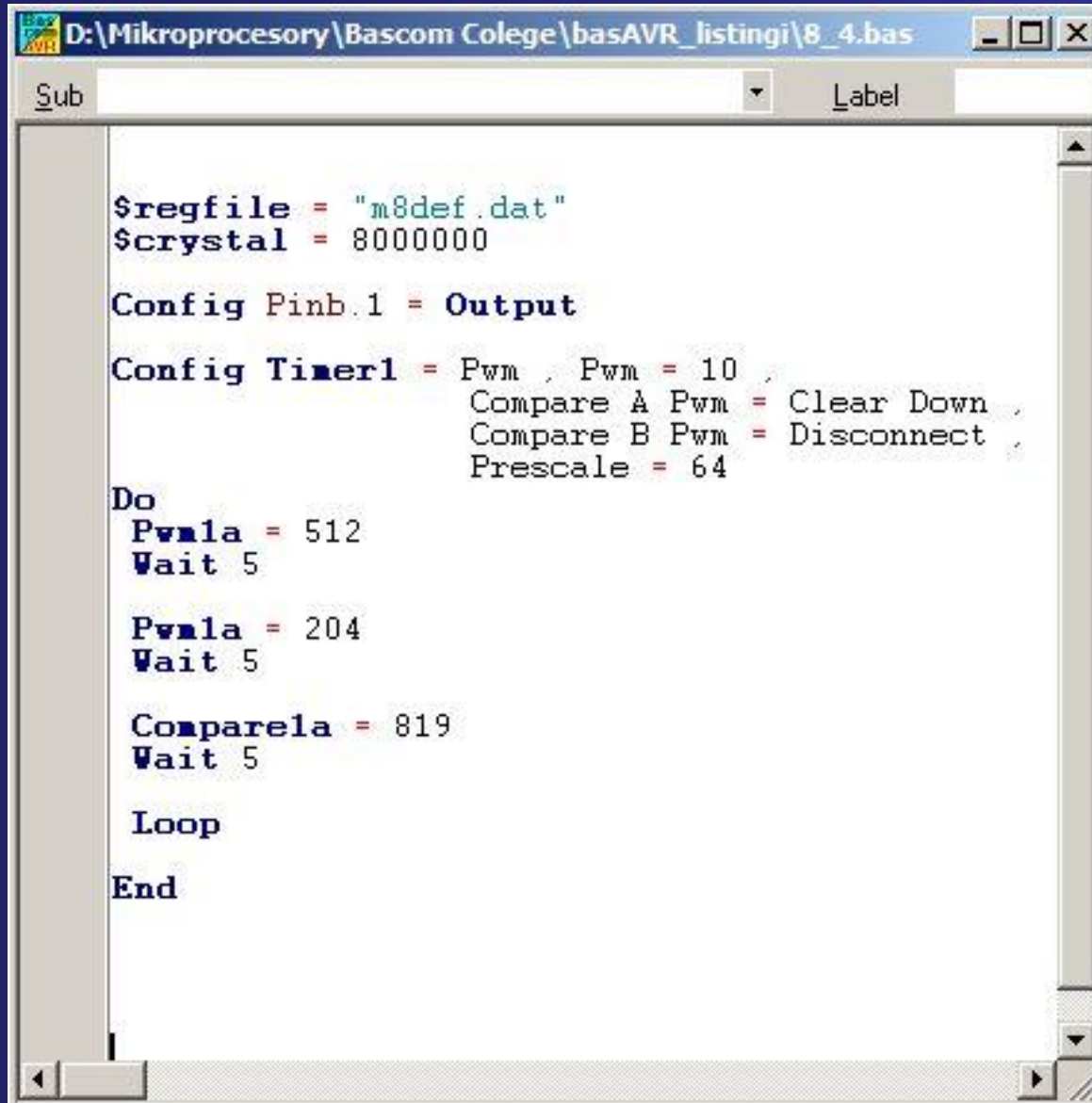


Płytki testowa



Program 5a

Zastosowanie sygnału PWM w przetworniku C/A



```
D:\Mikroprocesory\Bascom Colege\basAVR_listingi\8_4.bas
Sub
Label

$regfile = "m8def.dat"
$crystal = 8000000

Config Pinb.1 = Output

Config Timer1 = Pwm , Pwm = 10 ,
                Compare A Pwm = Clear Down ,
                Compare B Pwm = Disconnect ,
                Prescale = 64

Do
  Pwmla = 512
  Wait 5

  Pwmla = 204
  Wait 5

  Comparela = 819
  Wait 5

Loop

End
```

Symulacja programowa

The screenshot shows the AVR Simulator interface. At the top, there is a toolbar with various simulation controls. Below it, a 'Variables' window is visible, which is currently empty. The main area is a code editor displaying the following C code:

```
1 $regfile = "m8def.dat"
2 $crystal = 8000000
3
4
5
6 Config Pinb.0 = Output
7
8
9
10
11
12 Do
13
14
15 Set Portb.0
16
17 Reset Portb.0
18
19
20 Loop
21 End
22
23
24
25
```

At the bottom of the simulator, a hardware simulation window is shown, featuring a breadboard with several integrated circuits and a numeric keypad. A 'Comparator IN0' window is also visible, showing a value of 0.00.

PC = 0 | Cycles = 0 | Stopped

Jest możliwa ale

nie daje
prawidłowych
rezultatów