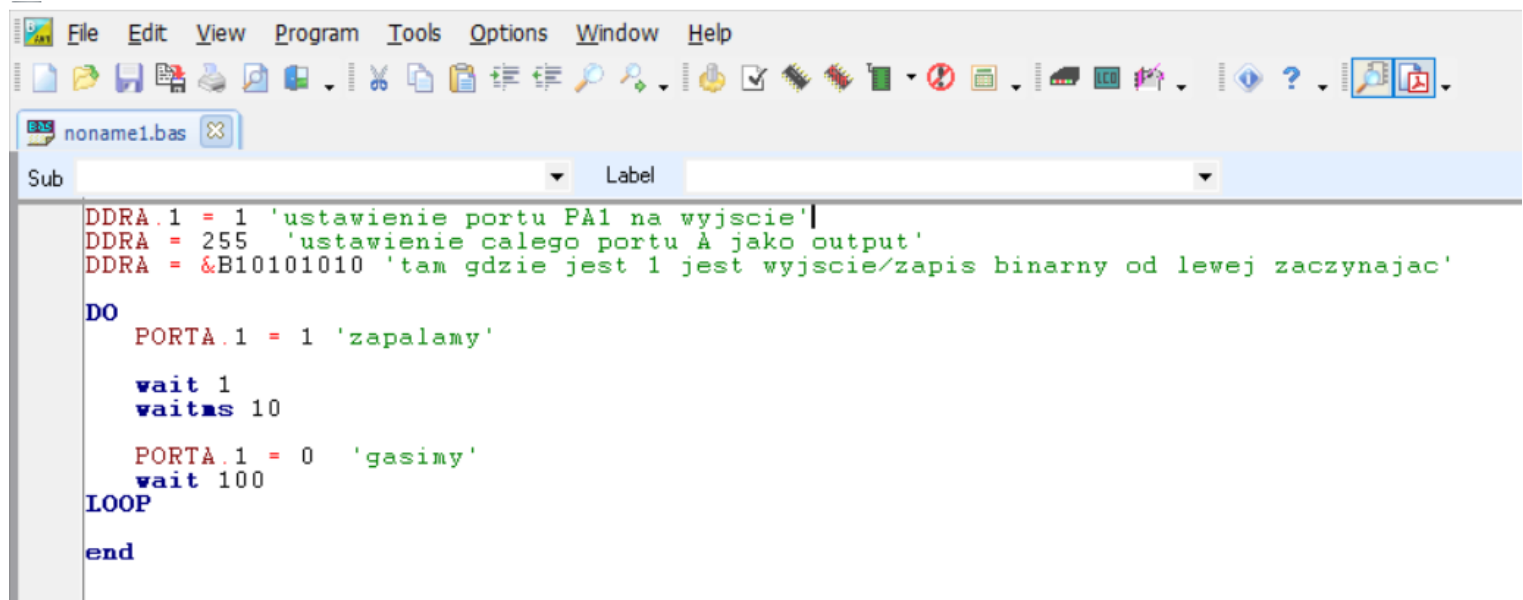


Sprawozdanie
Laboratorium Techniki Komputerowej
Mikrokontrolery

LTK XI
Radosław Terelak
Jakub Nowak
Alan Popiel

1. Ustawianie portu input/output – 15.03.2021



```
File Edit View Program Tools Options Window Help
noname1.bas
Sub Label
DDRA.1 = 1 'ustawienie portu PA1 na wyjście'
DDRA = 255 'ustawienie całego portu A jako output'
DDRA = &B10101010 'tam gdzie jest 1 jest wyjście/zapis binarny od lewej zaczynając'

DO
  PORTA.1 = 1 'zapalamy'
  wait 1
  waitms 10
  PORTA.1 = 0 'gasimy'
  wait 100
LOOP
end
```

Na pierwszych zajęciach ustawialiśmy wyjścia i wejścia na mikro-kontrolerze Attiny24.

Port można ustawić jako wejście lub wyjście na kilka sposobów. **DDRA.1 = 1** – ustawiamy konkretny port, w tym przypadku port PA1 jako wyjście. **DDRA = 255** – ustawiamy cały port PA jako wyjście. **DDRA = &B10101010** – ustawiamy kolejne porty w sposób naprzemienny. Jeden jest wyjściem kolejny wejściem.

Pętlą **DO** stworzyliśmy cykl zapalania i gaszenia diody “podłączonej” do portu 1.

Poleceniem **wait** dodajemy pauzę, abyśmy mogli zaobserwować zmiany w stanach zapalenia. Inaczej operacja wykonywałaby się zbyt szybko dla dostrzeżenia.

DDRA.1 = 1

ustawienie portu PA1 na wyjście

DDRA = 255

ustawienie całego portu A jako output

DDRA = &B10101010

tam gdzie jest 1 jest wyjście/zapis binarny od lewej zaczynając

DO

PORTA.1 = 1

zapalamy

wait 1

waitms 10

PORTA.1 = 0

gasimy

wait 100

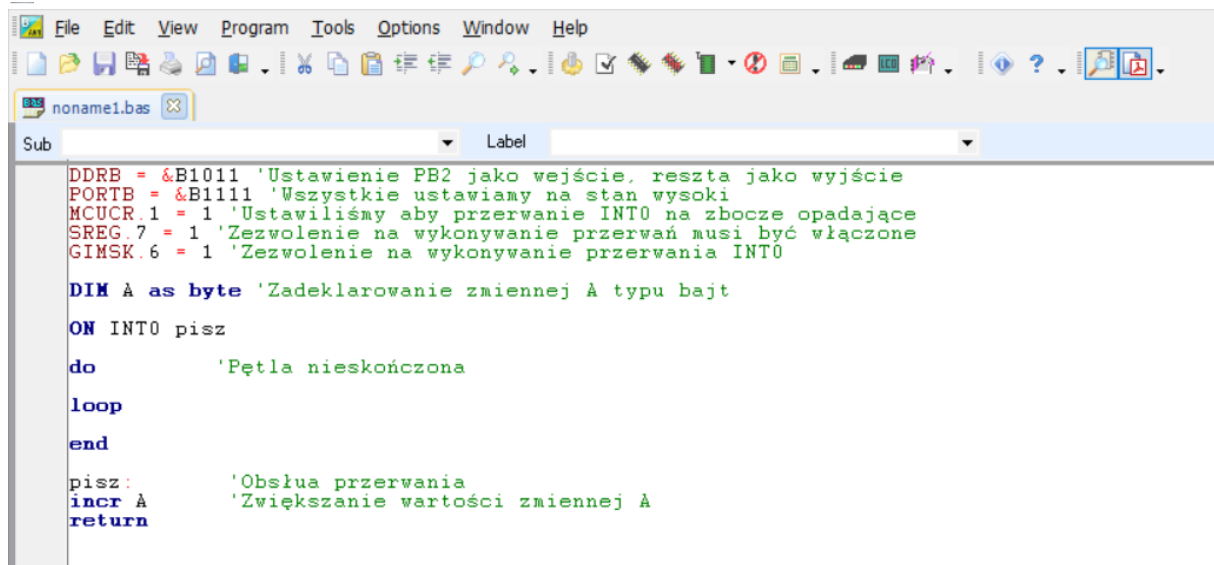
LOOP

End

WNIOSKI:

Na tych zajęciach nauczyliśmy się obsługiwać wejście oraz wyjście mikrokontrolera Attiny24. Dzięki tej wiedzy stworzyliśmy program, który zmieniał naprzemiennie włączenie/wyłączenie diod.

2. Przerwania – 22.03.2021



```
File Edit View Program Tools Options Window Help
noname1.bas
Sub Label
DDRB = &B1011 'Ustawienie PB2 jako wejście, reszta jako wyjście
PORTB = &B1111 'Wszystkie ustawiamy na stan wysoki
MCUCR.1 = 1 'Ustawiliśmy aby przerwanie INT0 na zbocze opadające
SREG.7 = 1 'Zezwolenie na wykonywanie przerw musi być włączone
GIMSK.6 = 1 'Zezwolenie na wykonywanie przerwania INT0

DIN A as byte 'Zadeklarowanie zmiennej A typu bajt
ON INT0 pisz
do 'Pętla nieskończona
loop
end
pisz: 'Obsługa przerwania
incr A 'Zwiększanie wartości zmiennej A
return
```

Na kolejnych zajęciach zajęliśmy się tematyką przerw. Przerwanie to zdarzenie, wstrzymujące działanie programu głównego, w celu uruchomienia specjalnej funkcji obsługi przerwania. Gdy ta funkcja zostanie wykonana następuje powrót do programu głównego. Pin musi być ustawiony na wejście.

Ustawiamy pin **PB2** jako wejście, a pozostałe (**PB0, PB1, PB3**) jako wyjścia. **DDRB = &B1011**

Ustawiamy wszystkie bity portu B na stan wysoki – **PORTB = &1111**

Wybieramy rodzaj przerwania **INT0** jako zbocze opadające, czyli dopiero gdy wartość na wejściu spadnie, dochodzi do przerwania – **MCUCR.1 = 1**

Włączamy zezwolenie na wykonywanie przerw w rejestrze SREG – **SREG.7 = 1** oraz w rejestrze GIMSK – **GIMSK.6 = 1**

Tworzymy zmienną **A** oraz tworzymy przerwanie **INT0** o nazwie **pisz**.

Z pętli nieskończonej możemy wyjść jedynie po wykonaniu przerwania, po czym przechodzimy do obsługi przerwania, gdzie zwiększamy wartość zmiennej **A**. Tą zmienną możemy inkrementować tak długo jak będziemy wykonywać przerwanie.

DDRA = &B1011

ustawienie PB2 jako wejście, reszta jako wyjście

PORTB = &B1111

wszystkie ustawiamy na stan wysoki

MCUCR.1 = 1

ustawienie przerwania na INTO na zboczu opadającym

SREG.7 = 1

zezwoleń na wykonywanie przerwan musi być włączone

GIMSK.6 = 1

zezwoleń na wykonywanie przerwania INTO

DIM A AS byte

zadeklarowanie zmiennej A typu bajt

ON INTO pisz

do

loop

end

pisz:

incr A

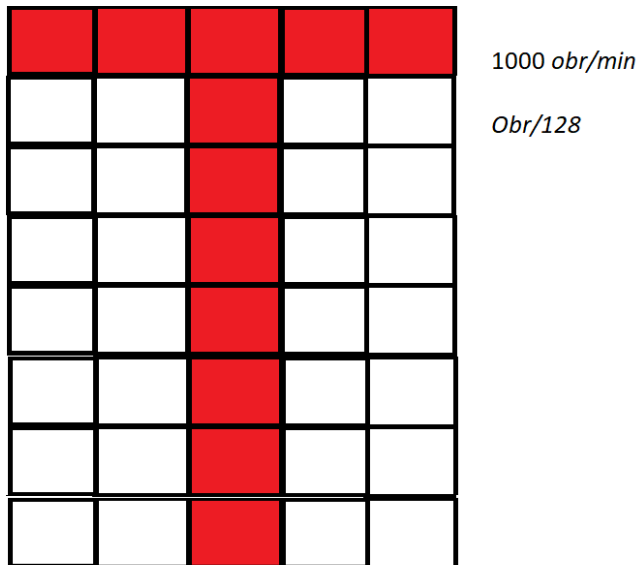
return

WNIOSKI:

Na tych zajęciach nauczyliśmy się obsługi przerwań mikrokontrolera Attiny24. Dowiedzieliśmy się, jak umożliwić mikrokontrolerowi doprowadzenie do przerwania.

3. Wyświetlanie litery – 29.03.2021

Aby wyświetlić literę „T” użyliśmy podobnej techniki z wykorzystaniem przerwań jak na poprzednich zajęciach. Utworzyliśmy 5 zmiennych reprezentujących kolejne rzędy komórek. Nasz obraz składa się z 5 kolumn oraz 8 wierszy. Oto graficzna reprezentacja:



Metodą binarną zapaliliśmy odpowiednie komórki oraz dodaliśmy przerwanie i pętlę nieskończoną. W obsłudze przerwania zmieniamy wartości **portu A**, w zależności od danego rzędu oraz dodaliśmy pauzę **400 mikrosekund**. Po przejściu przez każdą kolumnę gasimy wszystkie komórki i możemy ponowić procedurę.

```
File Edit View Program Tools Options Window Help
zad3.bas
Sub
  DDRA = 255
  DDRB = &B1011 'Ustawienie PB2 jako wejście, reszta jako wyjście
  PORTB = &B1111 'Wszystkie ustawiamy na stan wysoki
  MCUCR.1 = 1 'Ustawiliśmy aby przerwanie INT0 na zbocze opadające
  SREG.7 = 1 'Zezwolenie na wykonywanie przerwań musi być włączone
  GIMSK.6 = 1 'Zezwolenie na wykonywanie przerwania INT0

  DIM k_1 as Byte 'Wyświetlanie litery "T", tworzymy 5 bajtów
  DIM k_2 as Byte
  DIM k_3 as Byte
  DIM k_4 as Byte
  DIM k_5 as Byte

  k_1 = &B10000000 'Zapalamy odpowiednie okienka
  k_2 = &B10000000
  k_3 = 255
  k_4 = &B10000000
  k_5 = &B10000000

  ON INT0 wyswietl 'Dodajemy przerwanie
do 'Pętla oczekująca na input przerwania
loop
end

wyswietl: 'Zmiana stanu PORTA na kolejne wartości wcześniej zainicjowane
PORTA = k_1
waitus 400 'pauza na 400 mikrosekund

PORTA = k_2
waitus 400

PORTA = k_3
waitus 400

PORTA = k_4
waitus 400

PORTA = k_5
waitus 400

PORTA = 0 'Wygaszanie wszystkiego

return
```

DDRA = 255

DDRB = &B1011

Ustawienie PB2 jako wejście, reszta jako wyjście

PORTB = &B1111

Wszystkie ustawiamy na stan wysoki

MCUCR.1 = 1

Ustawiliśmy aby przerwanie INTO było na zbocze opadające

SREG.7 = 1

Zezwolenie na wykonywanie przerw musi być włączone

GIMSK.6 = 1

Zezwolenie na wykonywanie przerwania INTO

DIM k_1 as Byte

Wyświetlenie litery 'T', tworzymy 5 bajtów

DIM k_2 as Byte

DIM k_3 as Byte

DIM k_4 as Byte

DIM k_5 as Byte

k_1 = &B10000000

Zapalamy odpowiednie okienka

k_2 = &B10000000

k_3 = 255

k_4 = &B10000000

k_5 = &B10000000

ON INTO wyswietl

Dodajemy przerwanie

do

Pętla oczekująca na input przerwania

loop

end

wyświetl:

Zmiana stanu PORTA na kolejne wartości wcześniej zainicjonowane

PORTA = k_1

waitus 400

pauza na 400 mikrosekund

PORTA = k_2

waitus 400

PORTA = k_3

waitus 400

PORTA = k_4

waitus 400

PORTA = k_5

waitus 400

PORTA = 0

Wygaszenie wszystkiego

return

WNIOSKI:

Bazując na wiedzy z poprzednich zajęć udało nam się wyświetlić literę "T", w chwili obecnej jest to wykonane za pomocą polecenia *wait*. Nie jest to najlepszy możliwy sposób, ponieważ podczas polecenia *wait* mikrokontroler nie może wykonywać żadnych operacji.

4. Działanie na licznikach mikrokontrolera – 12.04.2021

Na tych zajęciach zajęliśmy się licznikiem 16 bitowym.

W celu zliczenia czasu pełnego obrotu użyliśmy licznika 16 bitowego oraz przypisywaliśmy go do odliczania oraz w celu prawidłowego pokazywania liter przez drugi z liczników. Poniższy kod obrazuje sposób obliczenia czasu 1 obrotu tarczy. Dzięki temu mogliśmy obliczyć 1/128 tarczy, która stopniowo będzie przysyłać kolejne części litery, przy czym cały proces odbywa się w pętli.

```
TCCR1B.1 = 1 'ustawienie pierwszego bitu rejestru TCCR1B na 1

DDRA = &B11111111 'ustawienie całego portu A jako wyjście
DDRB = &B1011 'ustawienie portu PB2 jako wejście, a PB1, 3 i 4 jako wyjście; DDRB - ustawiamy cos w B
PORTB = &B1111 'ustawienie wszystkich bitów na stan wysoki
MCUCR.1 = 1 'ustawiamy ISC01 jako 1, reszta pozostaje jako 0 - ustawiliśmy przerwanie INT0 na zbocze opadające
SREG.7 = 1 'zezwolenie na wykonywanie przerw - ustawienie 7 bitu rejestru SREG jako 1
GIMSK.6 = 1 'ustawiamy 6 bit rejestru GIMSK jako 1, co pozwala nam na wykonanie zewnętrznego przerwania INT0

DIM I as byte 'zadeklarowanie zmiennej I typu bajt
DIM H as byte 'zadeklarowanie zmiennej H typu bajt
DIM D as word 'zadeklarowanie zmiennej D typu word

ON INT0 wyswietl

do 'petla nieskonczona; jedynie przerwanie INT0 powoduje wyjście z niej
loop
end

wyswietl: 'wyswietl zmienia wartosc PORTA na wartosci zmiennych odpowiadajacych kolumnom

L = TCNT1L 'zapisywanie licznika do zmiennej L
H = TCNT1H 'zapisywanie licznika do zmiennej H

TCNT1L = 0 'zerowanie TCNT1L
TCNT1H = 0 'zerowanie TCNT1H

D = H*256 'zapisywanie H do zmiennej D
D = D+L 'zapisywanie L do zmiennej D

PORTA = H 'wyswietlanie najwazniejszych 8 bitow

return
```

TCCR1B.1 = 1

ustawienie pierwszego bitu rejestru TCCR1B na 1

DDRA=&B11111111

ustawienie całego portu A jako wyjście

DDRB = &B1011

ustawienie portu PB2 jako wejście, a PB1.3 oraz PB1.4 jako wyjście

PORTB=&B1111

ustawienie wszystkich bitów na stan wysoki

MCUCR.1=1

ustawiamy ISC01 jako 1. reszta pozostaje jako 0 – ustawiliśmy przerwanie INTO na zbocze opadające

SREG.7=1

zezwoleń na wykonywanie przerw – ustawienie 7. bitu rejestru SREG jako 1

GIMSK.6=1

ustawiamy 6. bit rejestru GIMSK jako 1. co pozwala nam na wykonanie zewnętrznego przerwania INTO

DIM L as Byte

zadeklarowanie zmiennej L typu bajt

DIM H as Byte

zadeklarowanie zmiennej H typu bajt

DIM D as Word

zadeklarowanie zmiennej D typu word

ON INTO wyswietl

do

pętla nieskończona, jedynie przerwanie INTO powoduje wyjście z niej

loop

end

wyswietl:

wyswietl zmienia wartość PORTA na wartości zmiennych odpowiadających kolumnom

L=TCNT1L

zapisywanie licznika do zmiennej L

$H = TCNT1H$

zapisywanie licznika do zmiennej H

$TCNT1L = 0$

zerowanie TCNT1L

$TCNT1H = 0$

zerowanie TCNT1H

$D = H * 256$

zapisywanie H do zmiennej D

$D = D + L$

zapisywanie L do zmiennej D

$PORTA = B$

Wyświetlanie najważniejszych 8 bitów

return

WNIOSKI:

Na tej lekcji nauczyliśmy się korzystać z licznika mikrokontrolera, ustawiać go oraz jak prawidłowo pobierać z niego wartość.

5. Działanie na licznikach mikrokontrolera II – 19.04.2021

Na tych zajęciach zajmowaliśmy się licznikiem 8 bitowym. Naszym celem jest uzyskanie przerwania od licznika 8 bitowego, żeby w momencie wyznaczenia czasu jednego pełnego obrotu, a następnie podzieleniu go na 128 części, w każdej kolejnej części po kolei, wspomniany licznik mógł zgłosić przerwanie. W momencie, kiedy licznik doliczy do wartości, jaka została do niego wprowadzona, zgłosi przerwanie. W naszym przerwaniu zasterujemy T od n-tego wycinka co widać na poniższym kodzie źródłowym.

```
File Edit View Program Tools Options Window Help
zad5.bas
Sub
Label
TCCR1B.1 = 1 'ustawienie pierwszego bitu rejestru TCCR1B na 1
TCCR0B.0 = 1 'ustawienie zerowego bitu rejestru TCCR0B na 1
TCCR0B.1 = 1 'ustawienie pierwszego bitu rejestru TCCR0B na 1

TIMSK0.1 = 1

DDRA = &B11111111 'ustawienie całego portu A jako wyjście
DDRB = &B1011 'ustawienie portu PB2 jako wejście, a PB1, 3 i 4 jako wyjście; DDRB - ustawiamy coś w B
PORTB = &B1111 'ustawienie wszystkich bitów na stan wysoki
MCUCR.1 = 1 'ustawiamy ISC01 jako 1, reszta pozostaje jako 0 - ustawiliśmy przerwanie INT0 na zbocze opadające
SREG.7 = 1 'zezwolenie na wykonywanie przerw - ustawienie 7 bitu rejestru SREG jako 1
GIMSK.6 = 1 'ustawiamy 6 bit rejestru GIMSK jako 1, co pozwala nam na wykonanie zewnętrznego przerwania INT0

DIM I as byte 'zadeklarowanie zmiennej I typu bajt
DIM H as byte 'zadeklarowanie zmiennej H typu bajt
DIM D as word 'zadeklarowanie zmiennej D typu word
DIM I as byte 'zadeklarowanie zmiennej I typu bajt, która będzie służyć jako licznik

DIM k(5) as Byte 'Wyświetlanie litery "T", tworzymy 5 bajtów
k(1) = &B10000000 'Zapalamy odpowiednie okienka
k(2) = &B10000000
k(3) = 255
k(4) = &B10000000
k(5) = &B10000000

ON INT0 wyswietl
ON OC0A pasek

do 'petla nieskonczona; jedynie przerwanie INT0 powoduje wyjście z niej
loop
end

wyswietl: 'wyswietl zmienia wartość PORTA na wartości zmiennych odpowiadających kolumnom
I = TCNT1L 'zapisywanie licznika do zmiennej I (lower)
H = TCNT1H 'zapisywanie licznika do zmiennej H (higher)

TCNT1L = 0 'zerowanie TCNT1L
TCNT1H = 0 'zerowanie TCNT1H

D = 256 * H
D = H + I
D = D / 1024 'podzielenie D przez 1024, bo 8 * 128 = 1024

OCR0A = Low(D) 'zapisanie niższych bitów zmiennej D do rejestru OCR0A,
'który odpowiada za porównywanie z wartością licznika TCNT0

I = 1 'ustawienie wartości zmiennej I jako 1

return

pasek: 'etykieta pasek i obsługa przerwania OC0A
IF I < 6 THEN 'jeśli I jest mniejsze niż 6 to ustawiamy PORTA jako k(I)
PORTA = k(I)
ELSE
PORTA = 0 'jeśli I jest większe lub równe 6 to ustawiamy PORTA jako 0
END IF

INCR I 'zwiększenie wartości zmiennej I o 1

return
```

TCCR1B.1 = 1

ustawienie pierwszego bitu rejestru TCCR1B na 1

TCCR0B.0 = 1

ustawienie zerowego bitu rejestru TCCR0B na 1

TCCR0B.1 = 1

ustawienie pierwszego bitu rejestru TCCR0B na 1

TIMSK0.1 = 1

DDRA = &B11111111

ustawienie całego portu A jako wyjście

DDRB = &B1011

ustawienie portu PB2 jako wejście, a PB1, 3 i 4 jako wyjście; DDRB - ustawiamy coś w B

PORTB = &B1111

ustawienie wszystkich bitów na stan wysoki

MCUCR.1 = 1

ustawiamy ISC01 jako 1, reszta pozostaje jako 0 - ustawiliśmy przerwanie INTO na zbocze opadające

SREG.7 = 1

zezwoleń na wykonywanie przerw - ustawienie 7 bitu rejestru SREG jako 1

GIMSK.6 = 1

ustawiamy 6 bit rejestru GIMSK jako 1, co pozwala nam na wykonanie zewnętrznego przerwania INTO

DIM L as byte

zadeklarowanie zmiennej L typu bajt

DIM H as byte

zadeklarowanie zmiennej H typu bajt

DIM D as word

zadeklarowanie zmiennej D typu word

DIM I as byte

zadeklarowanie zmiennej I typu bajt, która będzie służyć jako licznik

DIM k(5) as Byte

Wyświetlanie litery "T", tworzymy 5 bajtów

k(1) = &B10000000

Zapalamy odpowiednie okienka

k(2) = &B10000000

k(3) = 255

k(4) = &B10000000

k(5) = &B10000000

ON INT0 wyświetl

ON OC0A pasek

do

pętla nieskończona; jedynie przerwanie INT0 powoduje wyjście z niej

loop

end

wyświetl:

wyświetl zmienia wartość PORTA na wartości zmiennych odpowiadających kolumnom

L = TCNT1L

zapisywanie licznika do zmiennej L (lower)

H = TCNT1H

zapisywanie licznika do zmiennej H (higher)

TCNT1L = 0

zerowanie TCNT1L

TCNT1H = 0

zerowanie TCNT1H

$D = 256 * H$

$D = H + L$

$D = D / 1024$

podzielenie D przez 1024, bo $8 * 128 = 1024$

OCR0A = Low(D)

zapisanie niższych bitów zmiennej D do rejestru OCR0A, 'który odpowiada za porównywanie z wartością licznika TCNT0

I = 1

ustawienie wartości zmiennej I jako 1

return

pasek:

etykieta pasek i obsługa przerwania OC0A

IF I < 6 THEN

jeśli I jest mniejsze niż 6 to ustawiamy PORTA jako k(I)

$PORTA = k(I)$

ELSE

PORTA = 0

jeśli I jest większe lub równe 6 to ustawiamy PORTA jako 0

END IF

INCR I

zwiększenie wartości zmiennej I o 1

return

WNIOSKI:

Dołączenie wiedzy o liczniku wraz z kodem z poprzednich zadań pozwoliło nam wyeliminować polecenie *wait*. Teraz mikrokontroler nie musi przerywać swojej pracy, kiedy aktualnie czegoś nie wyświetla i można przekierować jego moc obliczeniową na coś innego.

6. Własny program – 26.04.2021

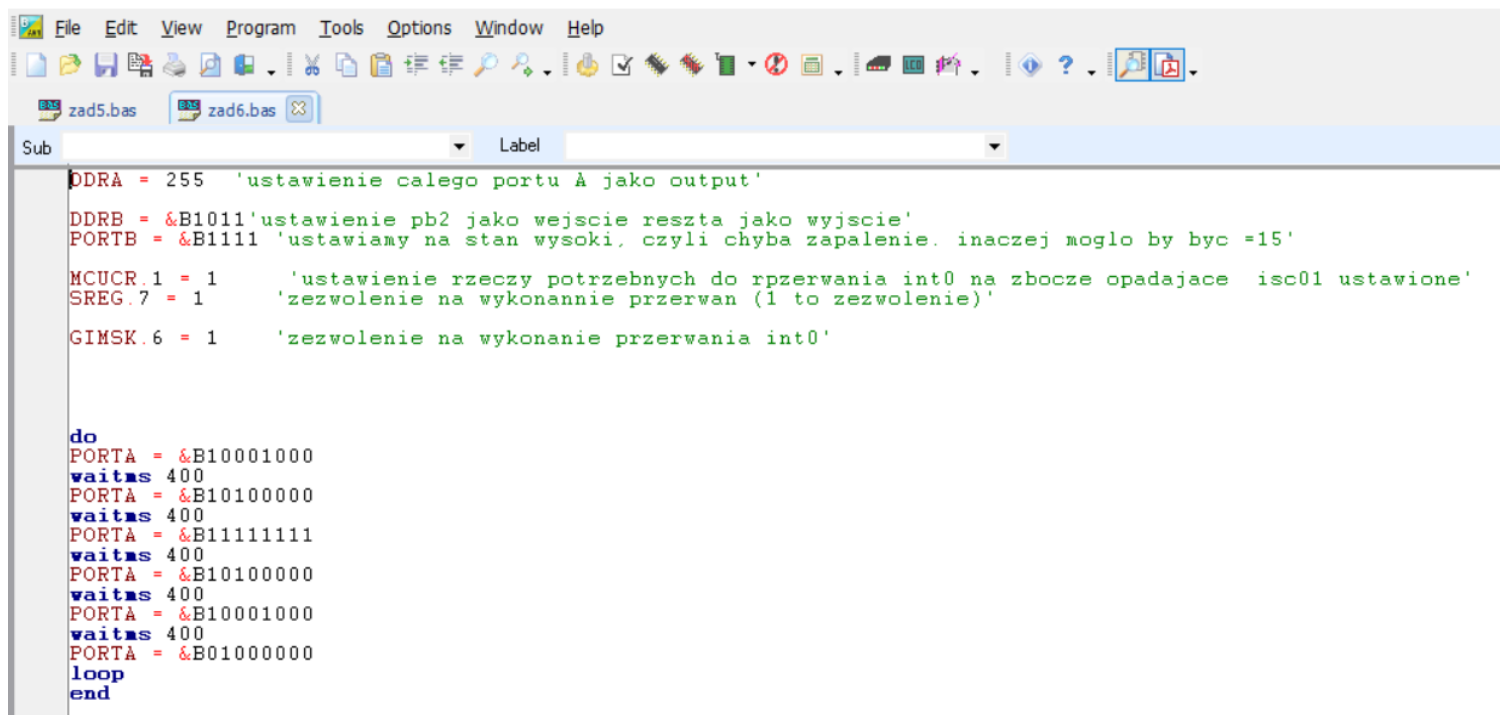
Prosty program do zapalania i gaszenia diod

Najpierw ustawiamy cały port 'A' jako wyjście.

Kolejną rzeczą jest ustawienie 'pb2' jako wejście, a resztę jako wyjście.

Potem ustawiamy na stan wysoki.

Następnie ustawiamy rzeczy potrzebne do przerwań int0 na zbocze opadające



```
File Edit View Program Tools Options Window Help
zad5.bas zad6.bas
Sub Label
DDRA = 255 'ustawienie całego portu A jako output'
DDRB = &B1011 'ustawienie pb2 jako wejście reszta jako wyjście'
PORTB = &B1111 'ustawiamy na stan wysoki, czyli chyba zapalenie. inaczej mogło być =15'
MCUCR.1 = 1 'ustawienie rzeczy potrzebnych do przerwania int0 na zbocze opadające isc01 ustawione'
SREG.7 = 1 'zezwolenie na wykonanie przerwań (1 to zezwolenie)'
GIMSK.6 = 1 'zezwolenie na wykonanie przerwania int0'

do
PORTA = &B10001000
waitas 400
PORTA = &B10100000
waitas 400
PORTA = &B11111111
waitas 400
PORTA = &B10100000
waitas 400
PORTA = &B10001000
waitas 400
PORTA = &B01000000
loop
end
```

ddra = 255

ustawienie całego portu A jako output

ddrB = &B1011

ustawienie pb2 jako wejście reszta jako wyjście

portb = &B1111

ustawiamy na stan wysoki, czyli zapalenie

MCUCR.1 = 1

ustawienie rzeczy potrzebne do przerwania int0 na zbocze opadające

sreg.7 = 1

zezwozenie na wykonywanie przerwań (1 to zezwozenie)

gimsk.6 = 1

zezwozenie na wykonanie przerwania int0

do

porta = &B10001000

waitms 400

porta = &B10100000

waitms 400

porta = &B11111111

waitms 400

porta = &B10100000

waitms 400

porta = &B10001000

waitms 400

porta = &B01000000

loop

end

WNIOSKI:

Napisaliśmy prosty program, żeby zobaczyć, czy uda się go uruchomić na mikrokontrolerze bez błędu, a nie tylko w Bascomie. Udało się to i program zmieniał stan diod zgodnie z oczekiwaniami.