Sprawozdanie z przedmiotu:

**Laboratorium Techniki Komputerowej**

Temat: **MIKROKONTROLERY**

Aleksandra Jenda

Konrad Chęciński

Piotr Dyba

Adrian Matlok

1. W pierwszym punkcie omówimy nasz kod napisany w programie BASCOM. Założeniem tego projektu było zasymulowanie migania diody z pomocą kontrolera Attiny24. (18.03.2021 r.)

DDRA.0=1 - polecenie to ustawia ‘1’ na zerowym bicie,

DDRA=1 - polecenie to nadpisuje poprzednie,

DDRA=255 - polecenie to zamienia podaną liczbę na jej binarną postać i zapisuje ją od najmłodszego bitu,

DDRA=&B10101011 - polecenie to ustawia każdy bit w sposób binarny,

do - polecenie to rozpoczyna pętle,

PORTA.0=1 - zapalenie diody,

wait 3 - wstrzymanie programu na 3 sekundy,

PORTA.0=0 - zgaszenie diody,

waitms 800 - wstrzymanie programu na 800 ms,

loop - zakończenie pętli,

end - zakończenie programu,

Z pomocą wyżej podanego kodu udało nam się zasymulować zapalenie diody na 3 sekundy oraz zgaszenie jej na 800 ms. Czynność ta jest powtarzana w pętli.

1. W tym punkcie przedstawimy nasz kod napisany w programie BASCOM. Program wykonuje akcje po otrzymaniu sygnału z zewnątrz (25.04.2021 r.)

DDRB=&B1011 - ustawienie rejestru DDRB na odpowiednie bity

PORTB=&B1111 - ustawienie rejestru PORTB na odpowiednie bity

MCUCR.1=1 - ustawienie rejestru MCUCR reagowanie przy “wzrastającym” sygnale

GIMSK.6=1 - włączenie przerwań na porcie int0

SREG.7=1 - włączenie globalnie przerwań

DIM a as Byte - deklaracja zmiennej a jako Byte

DIM b as Byte - deklaracja zmiennej b jako Byte

DIM c as Integer - deklaracja zmiennej c jako Int

b=&B10101010 - zmiana wartości zmiennej b

c=1 - zmiana wartości zmiennej c

on int0 przerwanie - na wejście wykona funkcję przerwanie

do - pętla nieskończona

loop

end

przerwanie: - blok poleceń wykonanych podczas wejścia sygnału zewnętrznego

incr a - zwiększanie zmiennej a o 1

toggle b - zmiana bitów z ‘1’ na ‘0’ i odwrotnie

c=c\*2 - pomnożenie wartości c razy 2

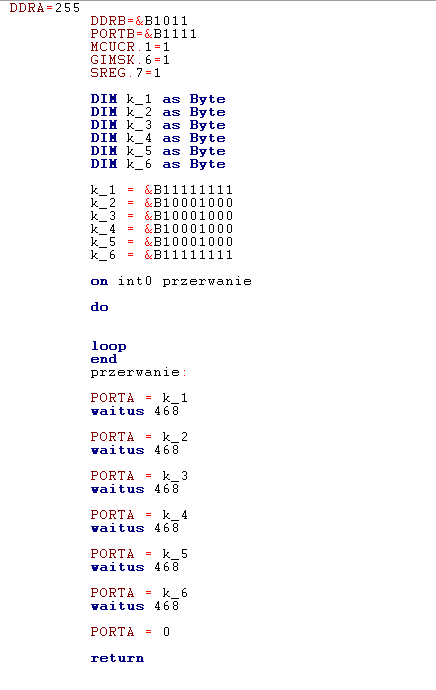
return - koniec bloku

**Powyższa funkcja przerwanie reaguje na sygnał wchodzący do mikroprocesora. Mikroprocesor podczas czekania nie wykonuje żadnej akcji, jest w stanie czuwania (może wykonywać inne funkcje w tym czasie)**

1. **Zadaniem naszego mikrokontrolera Attiny24 było włączanie i wyłącznie diod w celu pojawienia się odpowiedniej litery dzięki zjawisku stroboskopowemu.**

**Do naszego mikrokontrolera będzie docierał sygnał oznaczający zapalenie danej diody, oraz w odpowiednim momencie wyłączenie jej by dzięki zjawisku stroboskopowemu móc zobaczyć określoną literę, którą chcieliśmy otrzymać.**

**Zdjęcie programu:**



Opis instrukcji w programie:

DDRA = 255 – Ustawienie wszystkich portów A jako porty wyjścia

DDRB = &B1011 – Ustawienie portu PB2 jako wejście

PORTB = &B1111 – Ustawienie stanu wysokiego dla wszystkich pinów portu B

MCUCR.1=1 – Wyzwalanie przerwania gdy stan na pinie PBR2 będzie się zmieniał z wysokiego na niski

GIMSK.6=1 – Włączenie przerwań od INT0

SREG.7=1 – Globalnie włączamy przerwanie

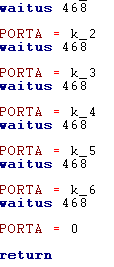
DIM [k\_1 : k\_6] as Byte – Deklarujemy zmienne o nazwach od k\_1 do k\_6

On int0 przerwanie – Tworzymy komendę dla mikrokontrolera, co ma zrobić gdy zgłoszone zostanie przerwanie. Nadajemy mu nazwę przerwanie.

Do

Loop – Tworzymy pustą pętle oraz ustawiamy komendę „end” kończącą program

End



**W tym momencie przypisujemy określony czas jaki dana dioda musi „poczekać” czyli wyłączyć się w celu uzyskania określonej literki. Obliczany jest on poprzez podzielenie 60s przez 1000 obrotów a następnie dzielimy wynik działania 60/1000 przez 128, ponieważ na tyle podzieliliśmy 1 obrót, przez co wyszedł nam czas oczekiwania równy 468 mikro sekund.**

1. W tym punkcie zajmiemy się licznikami (15.04). Użyjemy Timer/Counter 1 16 bitowy . Zegar ustawiamy na taktowanie 8MHz. Tarcza wiruje 1000 obr/min. Wybieramy prescaler ustawiony na 8

Kod:

DDRA=255

DDRB=&B1011 (ustawienie PB2 jako wejście reszta jako wyjście)

PORTB=&B1111 (wszystkie ustawiamy na stan wysoki)

MCUCR.1=1 (ustawienie INT0 na zbocze opadające

SREG.7=1 (zezwolenie na wykonywanie przerwań musi być włączone)

GIMSK.6=1 (zezwolenie na wykonanie przerwania INT0)

DIM A ASY Byte

DIM B ASY Byte

DIM C ASY Word ( Word jest 16 bitowy )

TTCR1B.1=1 ( inaczej SET TTCR1B.1) (ustawienie prescalera licznika 16-bitowego na wartość 8)

on INT0 przerwanie (dodajemy przerwanie)

do

loop

end

przerwanie:

A=TCNT1H

B=TCNT1L (zliczanie licznika)

TCNT1H= 0 ( zerowanie żeby każdy

TCNT1L=0 Kolejny obrót liczyć od zera)

C=B\* 256

C=C+A

PORTA=B ( wyświetlanie 8 najważniejszych bitów)

return

Za pomocą wyżej przedstawionego kodu udało nam się za pomocą licznika zliczyć czas 1 obrotu tarczy. Czynność jest powtarzana w pętli.

1. W tym punkcie zajmiemy się drugim licznikiem. Tym razem licznikiem 8 bitowym. Będziemy zliczać czas pełnego obrotu licznikiem 16 bitowym i przypisywać go do odliczania i odpowiednio pokazywania liter przez drugi licznik

DDRA=255

DDRB=&B1011 'PB2 wejście, reszta wyjście'

PORTB=&B1111 'Wszystkie piny w stan wysoki'

MCUCR.1=1 'Przerwanie int0 na zbocze opadające'

SREG.7=1 'Zezwolenie na wykonywanie przerwań'

GIMSK.6=1 'Zezwolenie na wykonywanie przerwania int0'

TCCR1B.0=0

TCCR1B.1=1 'Preskaler licznika=8'

'SET TCCR1B.1'

TCCR1B.2=0

TCCR0B.0=1

TCCR0B.1=1 'Preskaler licznika=64'

TCCR0B.2=0

TIMSK0.1=1 ‘Ustawienie wywołania przerwania2 przy przepełnieniu TCNT0’

DIM A as Byte

DIM B as Byte

DIM C as Word

DIM I as Byte

I=1

DIM k(6) as Byte

k(1) = &B11111111

k(2) = &B10001000

k(3) = &B10001000

k(4) = &B10001000

k(5) = &B10001000

k(6) = &B11111111

on int0 przerwanie

on OC0A przerwanie2

do

loop

end

przerwanie:

A=TCNT1L

B=TCNT1H 'Zliczanie licznika'

TCNT1H=0

TCNT1L=0 'Zerowanie'

C=B\*256

'C=B'

'SHIFT C,LEFT,8' 'przesunięcie bitowe w lewo o 8'

C=C+A

shift C,right,10

'OCR0A=C/8/128 'Dzielenie czasu obrotu na 128 części i przez 8 (w jednym liczniku przeskaler jest 8 w drugim 64, dlatego dzielimy przez 8)'

OCR0A=LOW(C)

I=0

return

przerwanie2:

TCNT0=0 ‘Wyzerowanie licznika 8 bitowego’

if I<=6 then

PORTA=k(I)

else

PORTA=0

end if

incr I

return

Za pomocą wyżej przedstawionego kodu udało nam się za pomocą licznika zliczyć czas 1 obrotu tarczy, wykorzystać go do wyliczenia 1/128 części tarczy, która wyśle kolejne etapy literki. Czynność jest powtarzana w pętli.

DDRA=255

DDRB=&B1011 'PB2 wejście, reszta wyjście'

PORTB=&B1111 'Wszystkie piny w stan wysoki'

MCUCR.1=1 'Przerwanie int0 na zbocze opadające'

SREG.7=1 'Zezwolenie na wykonywanie przerwań'

GIMSK.6=1 'Zezwolenie na wykonywanie przerwania int0'

TCCR1B.0=0

TCCR1B.1=1 'Preskaler licznika=8'

'SET TCCR1B.1'

TCCR1B.2=0

TCCR0B.0=1

TCCR0B.1=1

TCCR0B.2=0

TIMSK0.1=1

DIM A as Byte

DIM B as Byte

DIM C as Word

DIM I as Byte

I=1

DIM k(19) as Byte

k(1) = &B11111111

k(2) = &B10000000

k(3) = &B10000000

k(4) = &B10000000

k(5) = &B10000000

k(6) = &B00000000

k(7) = &B00000000

k(8) = &B00000001

k(9) = &B00000001

k(10) = &B11111111

k(11) = &B00000001

k(12) = &B00000001

k(13) = &B00000000

k(14) = &B00000000

k(15) = &B11111111

k(16) = &B00010000

k(17) = &B00101000

k(18) = &B01000100

k(19) = &B10000010

on int0 przerwanie

on OC0A przerwanie2

do

loop

end

przerwanie:

A=TCNT1L

B=TCNT1H 'Zliczanie licznika'

TCNT1H=0

TCNT1L=0 'Zerowanie'

C=B\*256

'C=B'

'SHIFT C,LEFT,8' 'przesunięcie bitowe'

C=C+A

shift C,right,10

'OCR0A=C/8/128

OCR0A=LOW(C)

I=0

return

przerwanie2:

TCNT0=0

if I<20 then

PORTA=k(I)

else

PORTA=0

end if

incr I

return