



WYDZIAŁ FIZYKI  
i INFORMATYKI STOSOWANEJ  
Uniwersytet Łódzki



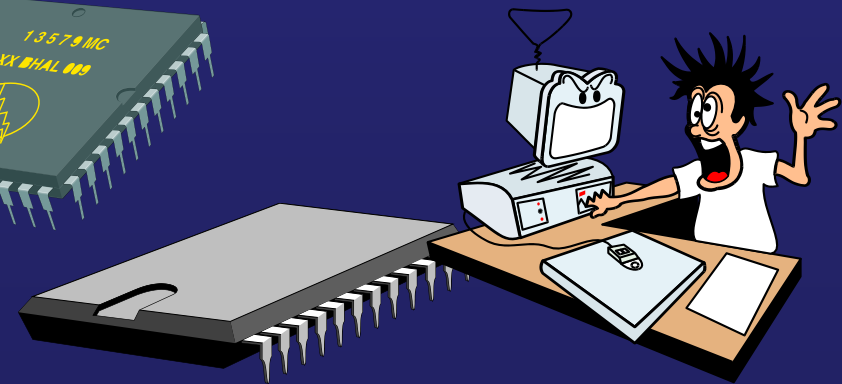
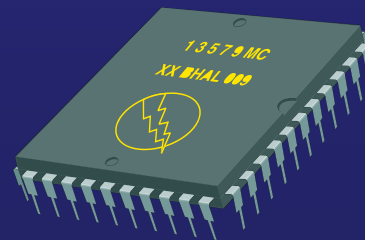
## *Systemy wbudowane*



*Witold Kozłowski*

*Zakład Fizyki i Technologii Struktur Nanometrowych  
90-236 Łódź, Pomorska 149/153*

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



# *Systemy wbudowane*

Kierunek: Informatyka  
PRACOWNIA DYDAKTYCZNA

## **Uwaga !!!**

**Proszę o wyłączenie  
telefonów komórkowych**

**na wykładzie i laboratorium**

*Systemy wbudowane*

Kierunek: Informatyka  
PRACOWNIA DYDAKTYCZNA

# Wykład 2.

**Język Bascom Basic AVR**

**Obsługa portów I/O**

# Zestaw znaków

Zestaw znaków BASCOM obejmuje znaki podstawowego alfabetu, liczby oraz znaki specjalne .

W skład liczb języka BASCOM wchodzić liczby z zakresu 0 do 9 a także zapis w postaci binarnej i szesnastkowej:

Zapis szesnastkowy powinien być poprzedzony przedrostkiem - &H

zapis bitowy powinien być poprzedzony przedrostkiem - &B

Za pomocą znaków podstawowego alfabetu możemy definiować stałe i zmienne oraz przydzielać im wartości:

## STAŁE

**Const nazwa = liczba**

**Const nazwa = „tekst”**

**Const nazwa = wyrażenie**

Przykład:

**Const H = 5** - definicja stałej liczbowej

**Const B = „witaj”** - definicja stałej tekstowej

**Const X = (H+6)+2** - definicja stałej obliczonej

## ZMIENNE

Przykład:

**x = 5**

- zmienna x jest typu Byte

**X = 5.1**

- zmienna x jest typu Single

**x = „witaj”**

- zmienna x jest typu String

**x = X+6**

- zmiennej przypisano wynik operacji matematycznej

## W języku BASCOM BASIC zdefiniowano kilka podstawowych typów danych:

Bit	1/8 bajta	Bit może przyjmować tylko dwie wartości: 0 i 1.
Byte	1 bajt	Bajt może przechowywać dowolną dodatnią liczbę całkowitą z zakresu od 0 do 255.
Word	2 bajty	Typ Word może przechowywać dowolną dodatnią liczbę całkowitą z zakresu od 0 do 65535.
Integer	2 bajty	Typ Integer może przechowywać dowolną liczbę całkowitą z zakresu -32768 do +32767.
Long	4 bajty	Typ Long może przechowywać dowolną liczbę całkowitą z zakresu $-2^{32}$ do $2^{32}-1$ -4294967296 do 4294967295
Single	4 bajty	Typ Single może przechowywać dowolną liczbę stała lub zmiennoprzecinkową.
String	max. 254 bajty	Typ String przechowuje dowolny ciąg znaków o długości nie większej niż 254 znaki. Ciąg ten zakończony jest zawsze znakiem 0. Każdy znak to jeden bajt. Tak więc tekst o długości 10 znaków zajmuje 11 bajtów.

# Zestaw znaków

Poniżej zestawiono znaki które posiadają szczególne znaczenie w programach w języku BASCOM BASIC:

ENTER	Oznacza koniec linii (Znak ten w nomenklaturze ASCII nazywa się CR)
	Pusty (lub spacja) – znak rozdzielający
'	Apostrof – <u>oznacza początek komentarza</u>
*	Gwiazdka – znak operacji mnożenia
+	Plus – znak operacji dodawania
,	Przecinek – <u>znak rozdzielający argumenty instrukcji</u>
-	Minus – znak operacji odejmowania
.	Kropka – oddziela część całkowitą od ułamkowej
/	Kreska ukośna – znak operacji dzielenia
:	Dwukropek – <u>rozdziela instrukcje zapisane w jednej linii</u>
"	Cudzysłów – <u>rozpoczyna i kończy dane tekstowe</u>
;	Średnik – <u>rozdziela argumenty instrukcji wejścia/wyjścia</u>
<	Mniejszy niż – znak operacji porównywania
=	Znak równości – występuje w operacjach przypisania oraz porównywania
>	Większy niż – znak operacji porównywania
\	Odwrotna kreska ukośna – <u>znak dzielenia dla liczb całkowitych</u>
^	Daszek – znak operacji potęgowania

Słowa zastrzeżone:

# Lista dyrektyw kompilatora

---

\$ASM

**\$BAUD**

\$BAUD1

\$BGF

\$BOOT

**\$CRYSTAL**

\$DATA

\$DBG

\$DEFAULT

\$EEPLeave

\$EEPROM

\$EEPROMHEX

\$EXTERNAL

\$INC

\$INCLUDE

\$LCD

\$LCDRS

\$LCDPUTCTRL

\$LCDPUTDATA

\$LCDVFO

\$LIB

\$MAP

\$NOINIT

\$NORAMCLEAR

\$PROG

**\$REGFILE**

\$ROMSTART

\$SERIALINPUT

\$SERIALINPUT1

\$SERIALINPUT2LCD

\$SERIALOUTPUT

\$SERIALOUTPUT1

\$SIM

\$TINY

\$XRAMSIZE

\$XRAMSTART

Słowa zastrzeżone:

# Lista instrukcji CONFIG

Określają  
parametry  
kompilacji

CONFIG 1WIRE

CONFIG ACI

CONFIG ADC

CONFIG ATEMU

CONFIG BCCARD

CONFIG CLOCK

CONFIG COM1

CONFIG COM2

CONFIG DATE

CONFIG DEBOUNCE

CONFIG GRAPHLCD

CONFIG I2CDELAY

CONFIG I2CSLAVE

CONFIG INTx

CONFIG KBD

CONFIG KEYBOARD

CONFIG LCD

CONFIG LCDBUS

CONFIG LCDMODE

CONFIG LCDPIN

CONFIG PIN

CONFIG PORT

CONFIG PS2EMU

CONFIG RC5

CONFIG SCL

CONFIG SDA

CONFIG SERIALIN

CONFIG SERIALIN1

CONFIG SERIALOUT

CONFIG SERIALOUT1

CONFIG SERVOS

CONFIG SPI

CONFIG TCPIP

CONFIG TIMER0

CONFIG TIMER1

CONFIG TIMER2

CONFIG WAITSUART

CONFIG WATCHDOG

CONFIG X10



# Lista instrukcji

---

1WRESET  
1WWRITE  
ALIAS  
BAUD  
BCCALL  
BCDEF  
BCRESET  
BITWAIT  
BYREF, BYVAL  
CALL  
CIRCLE  
CLS  
CLOSE  
CLOSESOCKET  
CONST  
CURSOR  
DATA  
DBG  
DEBOUNCE  
DECR  
DECLARE SUB  
DECLARE FUNCTION  
DEFBIT , DEFBYTE ,  
DEFINT , DEFWORD  
DEFLCDCHAR  
DELAY  
DIM  
DISABLE  
DISPLAY  
DO - LOOP  
DTMFOUT  
ECHO

ELSE  
ENABLE  
END  
EXIT  
FLUSH  
FOR - NEXT  
FOURTHLINE  
FUNCTION  
GET  
GLDCMD  
GLCDDATA  
GOSUB  
GOTO  
HOME  
I2CINIT  
I2CRECEIVE  
I2CSEND  
I2CSTART, I2CSTOP,  
I2CRBYTE, I2CWBYTE  
IDLE  
IF - THEN - ELSE - END IF  
INCR  
INITLCD  
INPUTBIN  
INPUTHEX  
INPUT  
KILL  
LCD  
LCDAT  
LCDCONTRAST  
LINE  
LINE INPUT

LOAD  
LOADADR  
LOADLABEL  
LOCATE  
LOCAL  
LOOKUPSTR()  
LOWERLINE  
ON INTERRUPT  
ON VALUE  
OPEN  
OUT  
POKE  
POPALL  
POWERDOWN  
POWERSAVE  
PRINT  
PRINTBIN  
PS2MOUSEXY  
PSET  
PULSEIN  
PULSEOUT  
PUSHALL  
PUT  
RC5SEND  
RC6SEND  
READ  
RADEEPROM  
READMAGCARD  
REM  
RESET  
RESTORE  
RETURN  
ROTATE  
SEEK

## Lista funkcji

1WIRECOUNT()	DRIVERESET()	IP2STR()	SPC()	SWAP
1WREAD()	DRIVERESET()	ISCHARWAITING()	SPIMOVE()	SPIOUT
1WSEARCHFIRST()	DRIVEWRITESECTOR()	LCASE()	SQR()	START
1WSEARCHNEXT()	EOF()	LEFT()	STR()	STOP
1WVERIFY()	EXP()	LEN()	STRING()	SUB
ABS()	FIX()	LOC()	TAN()	THIRDLINE
ACOS()	FILEATTR()	LOF()	TANH()	TOGGLE
ASC()	FILEDATE()	LOG()	TRIM()	UPPERLINE
ASIN()	FILEDATETIME()	LOG10()	UCASE()	VARPTR()
ATN()	FILELEN()	LOOKUP()	VAL()	WAIT
ATN2()	FILETIME()	LOOKUPSTR()	VARPTR()	WAITMS
BASE6DEC()	FRAC()	LOOKDOWN()	WAITKEY()	WAITUS
BCD()	FREEFILE()	LOW()	X10DETECT()	WHILE - WEND
BIN()	FUSING()	LTRIM()	SELECT CASE – END	WRITE
BINVAL()	GETADC()	MAKEBCD()	SELECT	WRITEEEPROM
BIN2GREY()	GETATKBD()	MAKEDEC()	SENDSCAN	X10SEND
CHECKSUM()	GETDSTIP()	MAKEINT()	SENDSCANKBD	
CHR()	GETDSTPORT()	MAX()	SERIN	
COS()	GETKBD()	MID()	SEROUT	
COSH()	GETRC()	MIN()	SET	
CPEEK()	GETRC5()	PEEK()	SETFONT	
CPEEKH()	GETSOCKET()	POWER()	SETTCP	
CRC8()	GREY2BIN()	RAD2DEG()	SHIFT	
CRC16()	HEX()	RIGHT()	SHIFTCURSOR	
DEG2RAD()	HEXVAL()	RND()	SHIFTIN , SHIFTOUT	
DIR()	HIGH()	ROUND()	SHIFTLCD	
DISKFREE()	HIGHW()	RTRIM()	SHOWPIC	
DISKSIZE()	INITFILESYSTEM()	SEEK()	SHOWPICE	
DRIVECHECK()	INKEY()	SGN()	SOUND	
DRIVEGETIDENTITY()	INP()	SIN()	SONYSEND	
DRIVEINIT()	INSTR()	SINH()	SPIIN	
DRIVEREADSECTOR()	INT()	SPACE()	SPIINIT	

# Podstawy języka - struktura pliku źródłowego

Pisanie programu w Baskom Basicu nie jest trudne ale aby zwiększyć jego przejrzystość programu można podzielić go na bloki:

## Dyrektywy dla kompilatora

```
$regfile = "m8def.dat"
```

```
$crystal = 8000000
```

```
' .....
```

```
' .....
```

'informuje kompilator o pliku dyrektyw wykorzystywanego mikrokontrolera

'informuje kompilator o częstotliwości oscylatora taktującego mikrokontroler

## Instrukcje konfiguracji peryferiów oraz urządzeń zewnętrznych

```
Config Watchdog = 2048
```

```
Config Timer0 = Timer , Prescale = 1024
```

```
Config Portc = Output
```

```
' .....
```

```
' .....
```

'konfiguracja Watchdoga

'konfiguracja Timer0

'konfiguracja portu C

# Podstawy języka - struktura pliku źródłowego

## Deklaracje nagłówek funkcji oraz procedur a także instrukcji konfigurujących przerwania

Declare Sub Inkrementuj(liczb As Byte)

Declare Function Dodaj(c As Integer , D As Integer)

On Timer0 prze\_licz

.....

.....

Procedura inkrementacji

'funkcja dodawania

'przerwanie od przepelnienia Timer0

## Definicje zmiennych, stałych i aliasów

Dim Liczba As Byte

Dim A As Integer , B As Integer

Dim Wynik As Integer

Const X = 5

W Alias A

.....

.....

'definicja zmiennej typu Byte

'definicje zmiennych typu Integer

'definicja zmiennej typu Integer

'definicja stałej X

'definicja aliasu, W wskazuje na zmienną A

# Podstawy języka - struktura pliku źródłowego

## Program główny

A = 100	'przypisanie zmiennej A wartości 100
B = 80	'przypisanie zmiennej B wartości 80
Liczba = 5	'przypisanie zmiennej Liczba wartości 5
Call Inkrementuj(liczba)	'wywołanie procedury Inkrementacji
Print Liczba	'wyświetlenie wartości zmiennej liczba (wartość 6)
Wynik = Dodaj(a , B)	'wywołanie funkcji Dodaj
Print Wynik	'wyświetlenie wartości zmiennej Wynik (wartość 180)
.....	
.....	
End	'Koniec programu głównego

# Podstawy języka - struktura pliku źródłowego

## Definicje wcześniej zadeklarowanych funkcji oraz procedur

```
Sub Inkrementuj(Liczb As Byte)
```

```
    Incr Liczb
```

```
End Sub
```

```
Function Dodaj(c As Integer , D As Integer) As Integer
```

```
    Dodaj = C + D
```

```
End Function
```

```
.....
```

```
.....
```

'Treść procedury inkrementującej wartość parametru Liczb

'inkrementacja zmiennej Liczb

'koniec procedury

'Treść funkcji dodającej wartości parametrów C i D

'instrukcja dodaje wartości

'koniec funkcji

## Pozostałe procedury lub funkcje użytkownika

Dodatkowo w tej części programu mogą znajdować się podprogramy, podprogramy obsługi przerwań oraz tablice stałych.

# Podstawy języka - struktura pliku

```
D:\Mikroprocesory\Bascom Colege\basAVR_listingi\8_3.bas
Sub
Label

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

Config Pind.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 Portd.0
End If
Return
```

Dyrektywy dla kompilatora

Instrukcje konfiguracji peryferiów oraz urządzeń zewnętrznych

Deklaracje nagłówków funkcji oraz procedur a także instrukcji konfigurujących przerwania

Definicje zmiennych, stałych i aliasów

Program główny

Definicje wcześniej zadeklarowanych funkcji oraz procedur

Pozostałe procedury lub funkcje użytkownika

Dodatkowo w tej części programu mogą znajdować się podprogramy, podprogramy obsługi przerwań oraz tablice stałych.

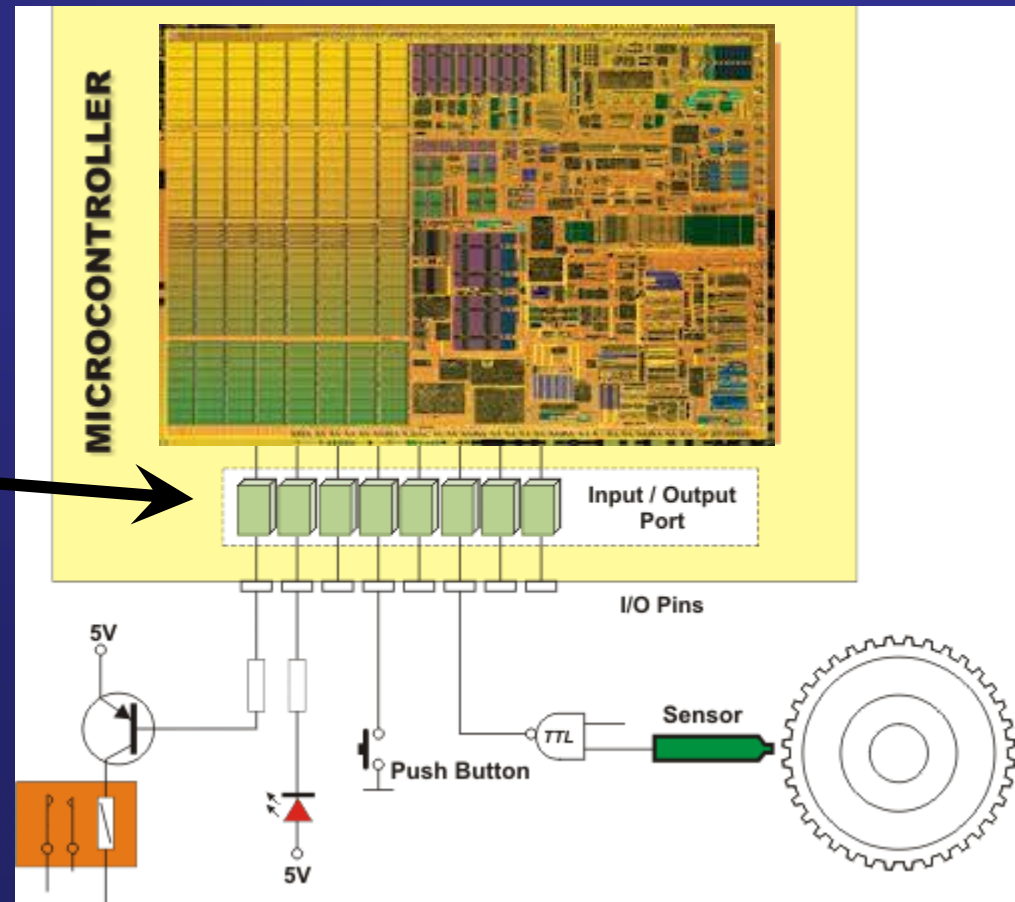
# Konfiguracja portów

Każdy mikrokontroler wyposażony jest w porty służące do komunikowania się mikrokontrolera z otoczeniem.

Liczba dostępnych portów jest różna i zależy od typu mikrokontrolera.

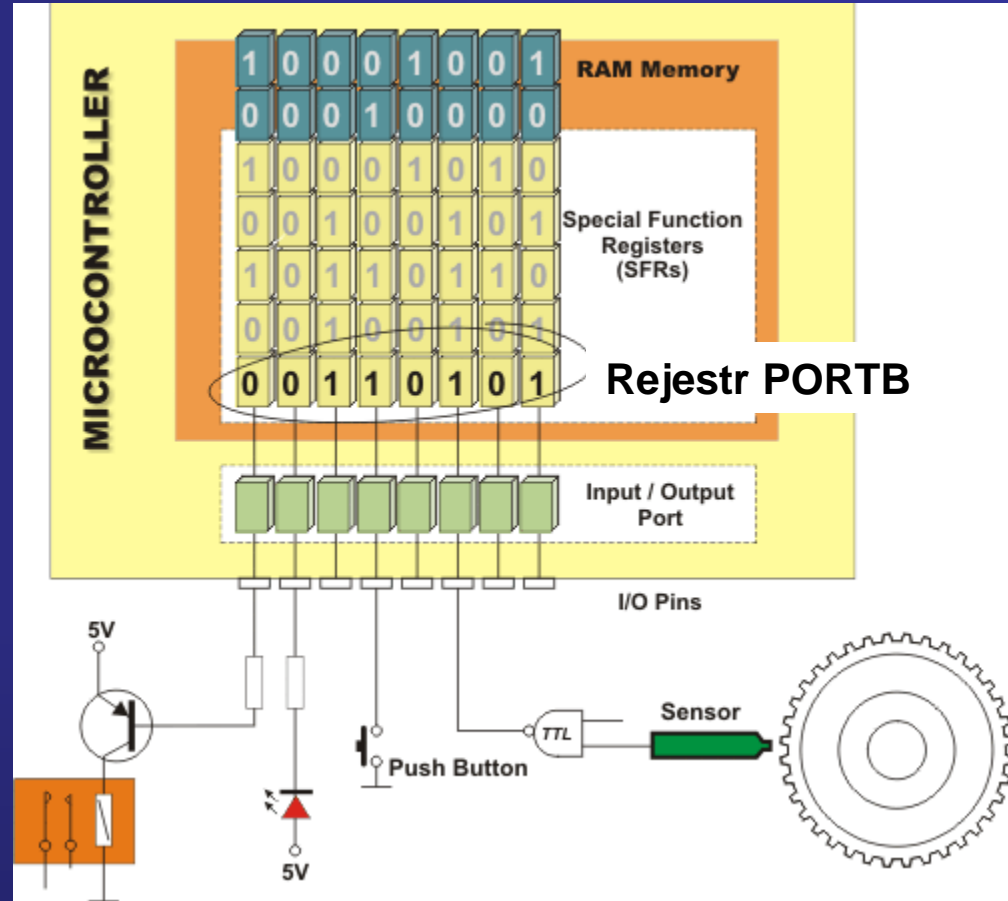
Port składa się z kilku linii,  
dla mikrokontrolerów 8 bitowych  
nie więcej niż 8.

Porty w mikrokontrolerze AVR są  
dwukierunkowe, tzn. mogą być wejściami  
albo wyjściami.





# Konfiguracja portów



W przestrzeni adresowej rejestrów specjalnych są aż trzy rejestry do obsługi portu:

**PORTx** - wartość wpisana do tego rejestru jest dostępna na zewnętrznych liniach portu

**DDRx** - rejestr ten służy do konfigurowania linii portu jako wejścia lub wyjścia

**PINx** - rejestr ten odwzorowuje bezpośrednio stan logiczny wyprowadzeń danego portu

Gdzie x – nazwa portu B,C,D...

# Rejestry specjalne mikrokontrolera Atmega 8

## Jest ich 64

PortB

PortC

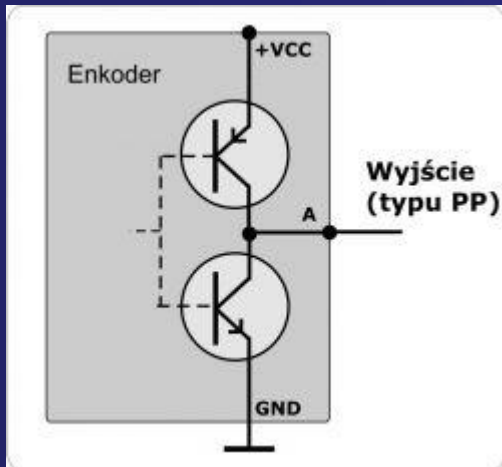
PortD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x3F (0x5F)	SREG	I	T	H	S	V		Z	C
0x3E (0x5E)	SPH	-	-	-	-	-	SP10	SP9	SP8
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
0x3C (0x5C)	Reserved								
0x3B (0x5B)	GICR	INT1	INT0	-	-	-	-	IVSEL	IVCE
0x3A (0x5A)	GIFR	INTF1	INTF0	-	-	-	-	-	-
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TCIE1	OCIE1A	OCIE1B	TOIE1	-	TOIE0
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	-	TOV0
0x37 (0x57)	SPMCR	SPMIE	RWWSB	-	RWWSRE	BLBSSET	POWRT	POERS	SPMEN
0x36 (0x56)	TWCR	TWNT	TWAA	TWSTA	TWSTO	TWAVC	TWEN	-	TWIE
0x35 (0x55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00
0x34 (0x54)	MCUCSR	-	-	-	-	WDRF	SCRF	EXTRF	PORF
0x33 (0x53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00
0x32 (0x52)	TCNT0	Timer/Counter0 (8 Bits)							
0x31 (0x51)	OBSCAL	Oscillator Calibration Register							
0x30 (0x50)	SPICR	-	-	-	-	ACME	PUD	PSR2	PSR10
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
0x2D (0x4D)	TCNT1H	Timer/Counter1 - Counter Register High byte							
0x2C (0x4C)	TCNT1L	Timer/Counter1 - Counter Register Low byte							
0x2B (0x4B)	OCR1AH	Timer/Counter1 - Output Compare Register A High byte							
0x2A (0x4A)	OCR1AL	Timer/Counter1 - Output Compare Register A Low byte							
0x29 (0x49)	OCR1BH	Timer/Counter1 - Output Compare Register B High byte							
0x28 (0x48)	OCR1BL	Timer/Counter1 - Output Compare Register B Low byte							
0x27 (0x47)	ICR1H	Timer/Counter1 - Input Capture Register High byte							
0x26 (0x46)	ICR1L	Timer/Counter1 - Input Capture Register Low byte							
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
0x24 (0x44)	TCNT2	Timer/Counter2 (8 Bits)							
0x23 (0x43)	OCR2	Timer/Counter2 Output Compare Register							
0x22 (0x42)	ASSR	-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB
0x21 (0x41)	WDTCR	-	-	-	WDCE	WDE	WDF2	WDF1	WDF0
0x20 <sup>11</sup> (0x40 <sup>11</sup> )	UBRRFH	URSEL	-	-	-	-	UBRR[11:8]		
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCS21	UCS20	UCPOL
0x1F (0x3F)	EEARH	-	-	-	-	-	-	-	EEAR8
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
0x1D (0x3D)	EEDR	EEPROM Data Register							
0x1C (0x3C)	EEDR	-	-	-	-	-	EERIE	EEMWE	EEWE
0x1B (0x3B)	Reserved								
0x1A (0x3A)	Reserved								
0x19 (0x39)	Reserved								
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x17 (0x37)	DDRB	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0x15 (0x35)	PORTC	-	PORTC8	PORTC6	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x14 (0x34)	DDRC	-	DDC8	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
0x13 (0x33)	PINC	-	PINC8	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x11 (0x31)	DDRD	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x0F (0x2F)	SPDR	SPI Data Register							
0x0E (0x2E)	SPSR	SPIF	WCOL	-	-	-	-	-	SP2X
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
0x0C (0x2C)	UDR	USART I/O Data Register							
0x0B (0x2B)	UCSRA	RXC	TXC	UDRIE	FE	DOR	PE	U2X	MPCM
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCS22	RXBS	TXBS
0x09 (0x29)	UBRRL	USART Baud Rate Register Low byte							
0x08 (0x28)	ACSR	ACD	ACBG	ACC	ACI	ACIE	ACIC	ACIS1	ACIS0
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
0x05 (0x25)	ADCH	ADC Data Register High byte							
0x04 (0x24)	ADCL	ADC Data Register Low byte							
0x03 (0x23)	TWDR	Two-wire Serial Interface Data Register							
0x02 (0x22)	TWAR	TWAR8	TWAR5	TWAR4	TWAR3	TWAR2	TWAR1	TWAR0	TWGCE

# Konfiguracja portów

DDBn	PORTBn	Tryb pracy	Podciąganie	Komentarz
0	0	Wejście	Nie	Trójstanowe (Hi-Z)
0	1	Wejście	Tak	Z linii Pbn może wypływać prąd, gdy końcówka będzie ściągnięta do masy.
1	0	Wyjście	Nie	Stopień wyjściowy typu Push-Pull, stan 0
1	1	Wyjście	Nie	Stopień wyjściowy typu Push-Pull, stan 1

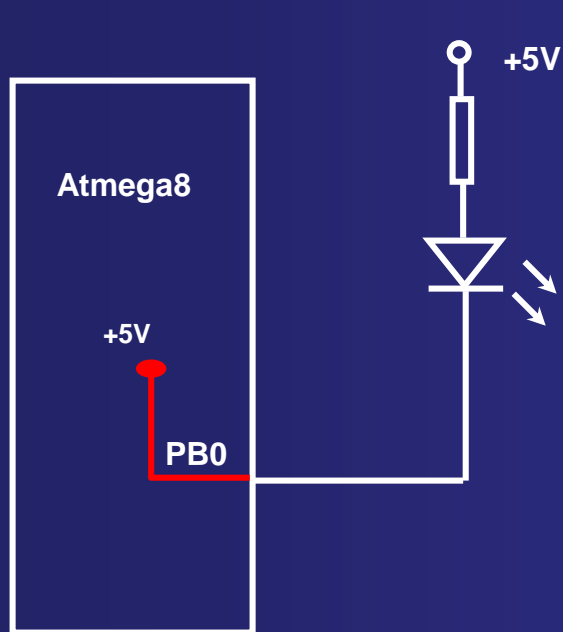
## Wyjście typu "Push-Pull" - PP



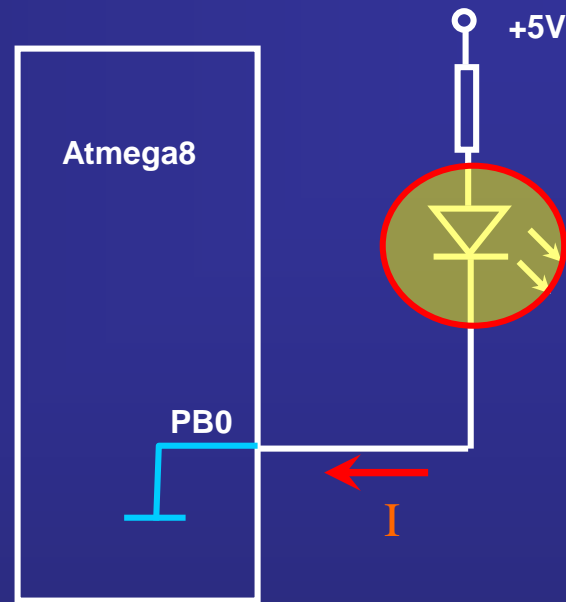
Wyjście typu Push-Pull posiada dwa stany aktywne. W stanie załączenia na wyjście podawane jest napięcie zasilania (+VCC), a w stanie wyłączenia na wyjście podawany jest sygnał masy (GND).

# Wyjście

# Konfiguracja portów



Wyjście ustawione na stan „1”



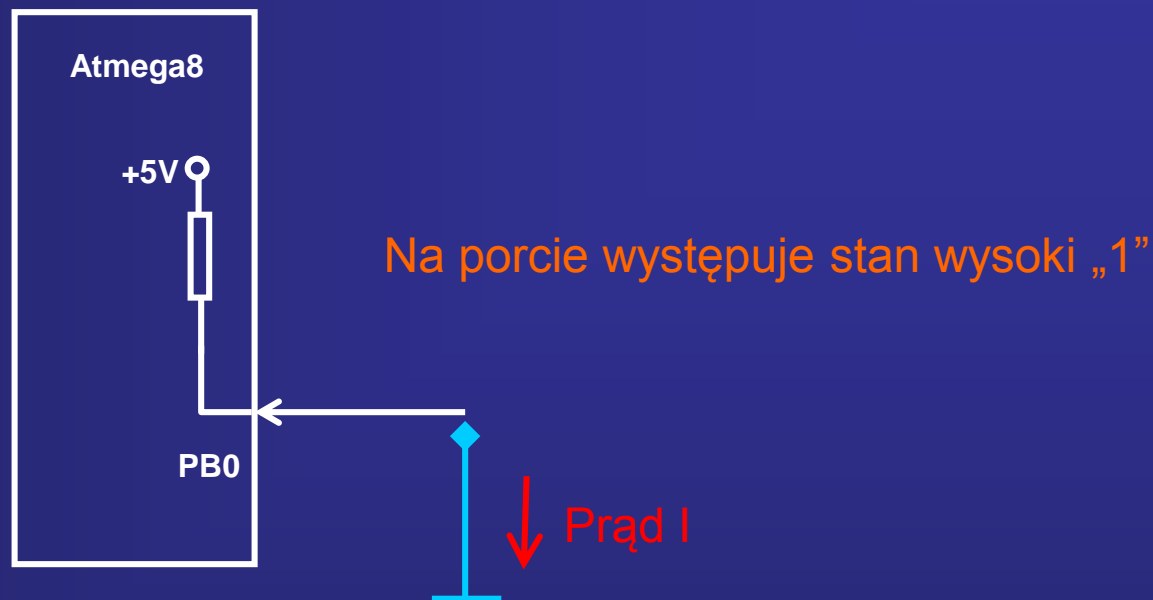
Wyjście ustawione na stan „0”

Port PB (i inne) może także bezpośrednio sterować diodami LED, gdyż prąd wpływający (linia portu na poziomie niskim) może mieć wartość nawet do 20mA.

**Uwaga:** Łączna obciążalność prądowa portów mikrokontrolera AVR nie powinna przekraczać 200mA, gdyż może nastąpić jego uszkodzenie

# Wejście

# Konfiguracja portów



Wszystkie końcówki portów (B,C,D) posiadają rezystory podciągające, które mogą być włączane osobno dla każdego wyprowadzenia.

Gdy końcówki np. portu PB pracując jako **wejścia** i są zewnętrznie ściągnięte do masy, to przy włączonym wewnętrznym podciąganiu będą źródłem prądu wypływającego.

# Konfiguracja portów w Bascom Basic

Ustawia kierunek działania portu.

Składnia:

*Config Portx = tryb*

*Config Pinx.y = tryb*

gdzie:

x – nazwa porty (B,C,D)

y – numer lini portu (0...7)

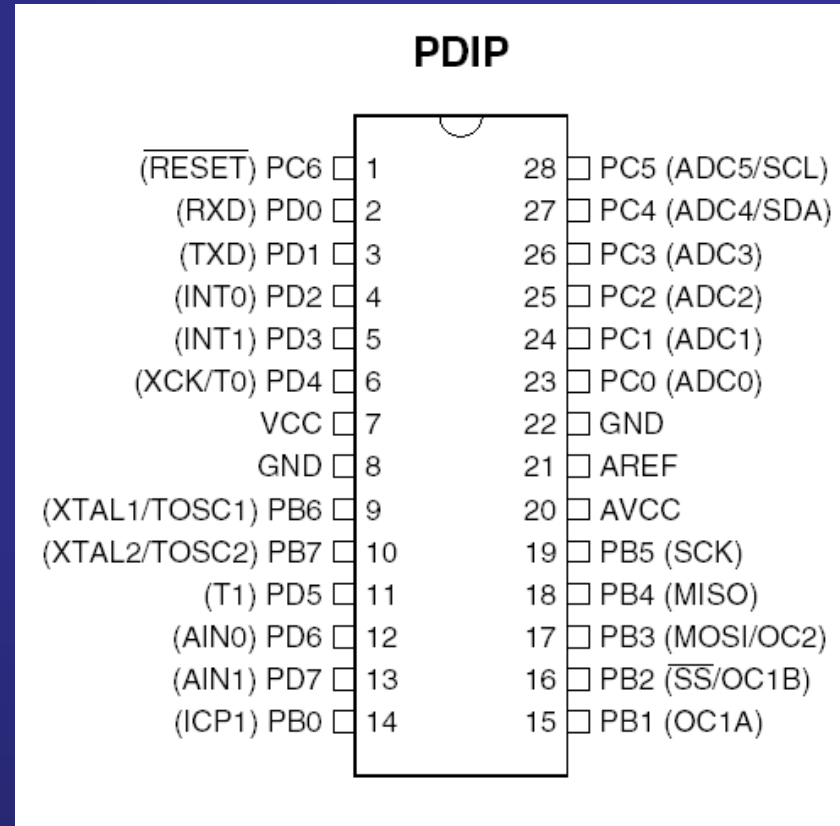
tryb - możliwe jest podanie:

*INPUT*

*OUTPUT*

- gdy port (końcówka) ma być wejściem

- gdy port (końcówka) ma być wyjściem.





Program 0



# Cel ćwiczenia

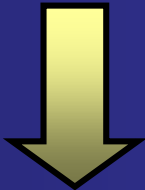
1. Analiza pracy Programu 0 przy użyciu symulatora programowego i sprzętowego
2. Badanie zależności czasowych wykonywanych instrukcji (*Set* i *Reset*) Programu 0 podczas symulacji i rzeczywistej pracy mikrokontrolera
3. Wyznaczenie powyższych wartości wykorzystując przyrządy pomiarowe (oscyloskop i analizator stanów logicznych)
4. Określenie ilości cykli pracy mikrokontrolera przypadających na wykonywane instrukcje
5. Zaprogramowanie mikrokontrolera programem napisanym w assemblerze realizującym to samo zadanie co program 0 napisany w języku wysokiego poziomu w Bascom Basic
6. Porównanie zależności czasowych sterowania portem Pb0 realizowanych programem napisanym w assemblerze i Bascom Basic
7. Określenie ilości cykli pracy mikrokontrolera przypadających na wykonywane instrukcje wysłania do port PB0 stanu wysokiego „1” oraz niskiego „0” realizowanych za pomocą dwóch programów: napisanych w assemblerze i Bascom Basic

# Program 0

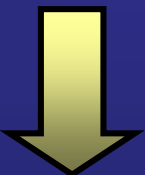
Napisanie programu



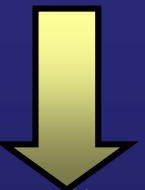
Zapisanie programu w własnym katalogu



Przeprowadzenie kompilacji



Poprawienie błędów składniowych i ponowne przeprowadzenie kompilacji



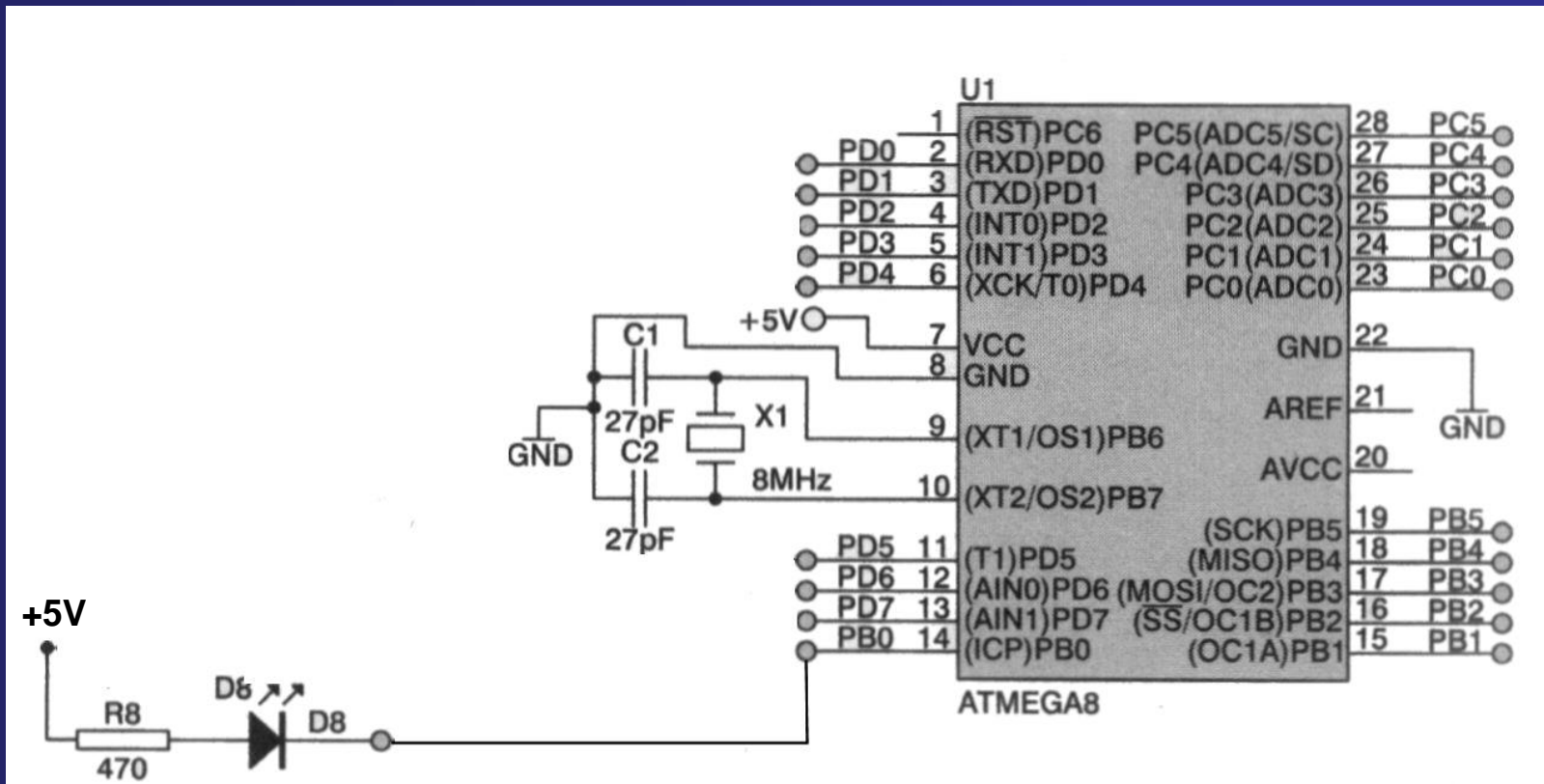
Przeprowadzenie symulacji programowej i sprzętowej



Zaprogramowanie mikrokontrolera

# Program 0

## Schemat połączenia diody LED do linii PBO portu B mikrokontrolera



# Program 0

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

Config Pinb.0 = Output

Do
Set Portb.0
Reset Portb.0
Loop
End
```

informuje kompilator o pliku dyrektyw mikrokontrolera

informuje kompilator o częstotliwości oscylatora taktującego mikrokontroler

linia PB0 jako wyjściowa

początek pętli

ustaw wyjście portu PB0 na „1”

ustaw wyjście portu PB0 na „0”

koniec pętli głównej programu

koniec programu

# Program 0

## Symulacja programowa

AVR Simulator

Variables Locals Watch uP Interrupts

Variable	Value	Hex	Bin

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

\* Bascom AVR \*

	7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0
PB	●	●	●	●	●	●	●	●	IB	●	●	●	●	●	●	●	●
PC	●	●	●	●	●	●	●	●	IC	●	●	●	●	●	●	●	●
PD	●	●	●	●	●	●	●	●	ID	●	●	●	●	●	●	●	●

0.00

0

1 2 3 A  
4 5 6 B  
7 8 9 C  
\* 0 # D

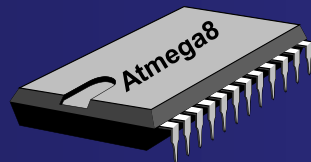
Comparator IN0

PC = 0 Cycles = 0 Stopped

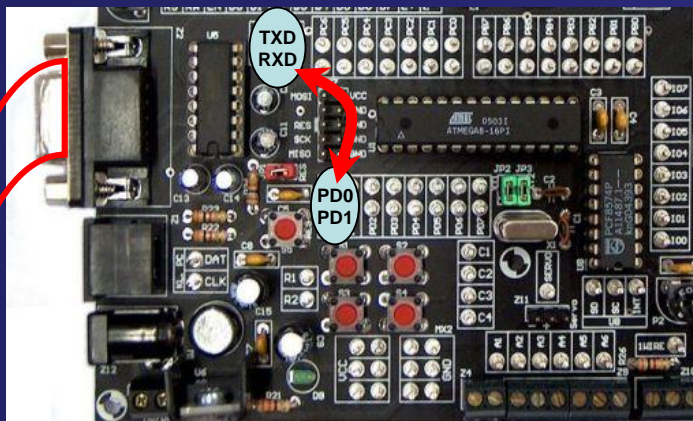
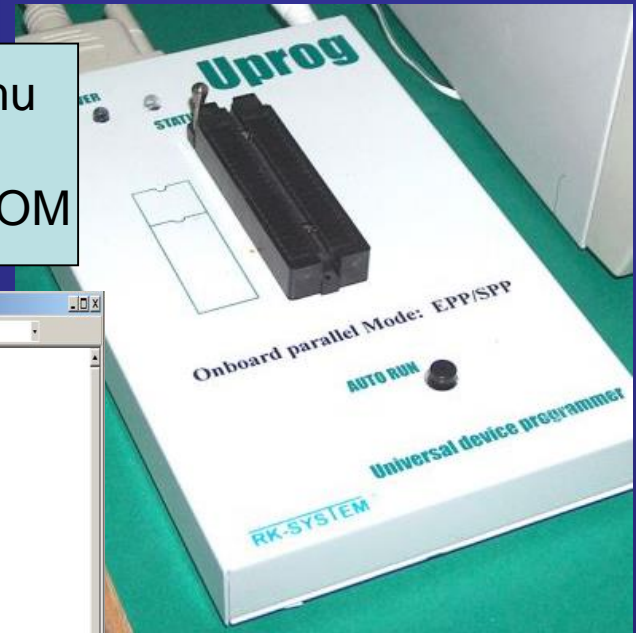
Czerwone diody dotyczą stanów linii portów wyjściowych

# Program 0

## Symulacja sprzętowa

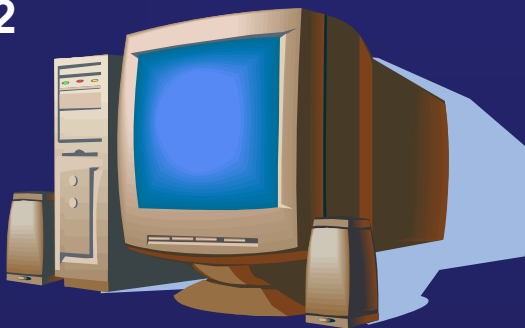


Wpisanie programu  
monitora  
do pamięci Flash ROM



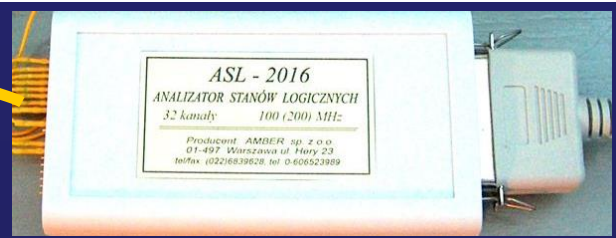
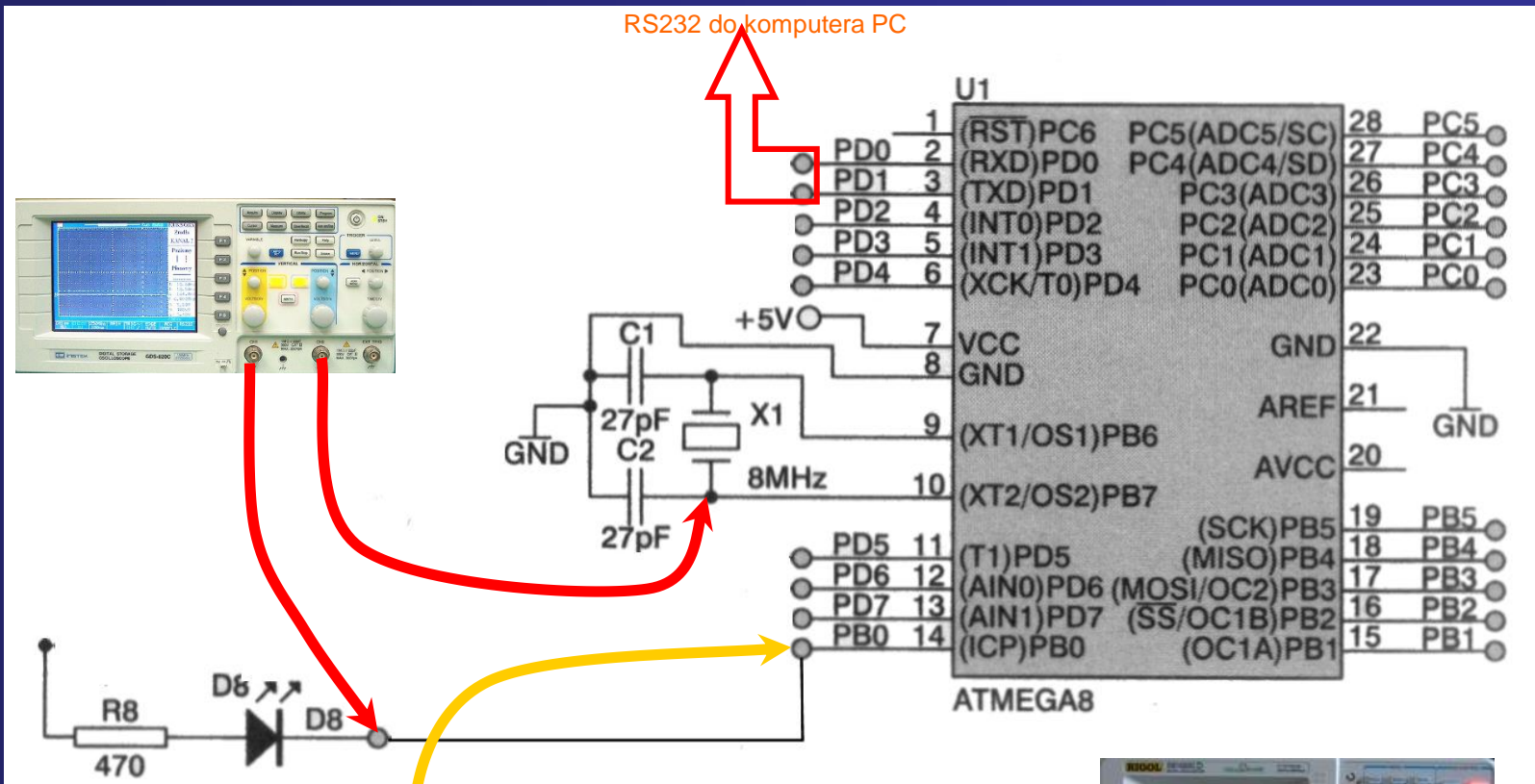
```
Sub  
$regfile = "m8def.dat"  
$crystal = 8000000  
$baud = 19200  
[variables]  
Dim Krx As Byte  
Dim adr As Word  
Dim adr1 As Byte , adr2 As Byte  
Dim V1 As Byte  
[main program]  
Print "BAGNON Version 1.01"  
Do  
Krx = Inkey()  
If Krx = "T" Then  
Print Chr(13)  
Elseif Krx = "U" Then  
adr = Waitkey()  
V1 = Waitkey()  
Out adr , V1  
Print Chr(13)  
Elseif Krx = "E" Then  
adr = Waitkey()  
V1 = Inp(adr)  
Print Chr(13)  
Elseif Krx = "O" Then  
adr1 = Waitkey()  
adr2 = Waitkey()  
V1 = Waitkey()  
adr = adr1 + 256  
adr = adr + adr1  
Out adr , V1  
Print Chr(13)  
Elseif Krx = "P" Then  
Print "?"  
End If  
Loop
```

RS232



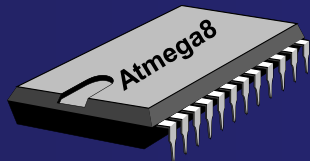
# Program 0

## Schemat połączenia diody LED Oraz przyrządów pomiarowych



# Program 0

## Zaprogramowanie mikrokontrolera docelowym programem 0



Wpisanie  
do pamięci Flash ROM  
Programu0

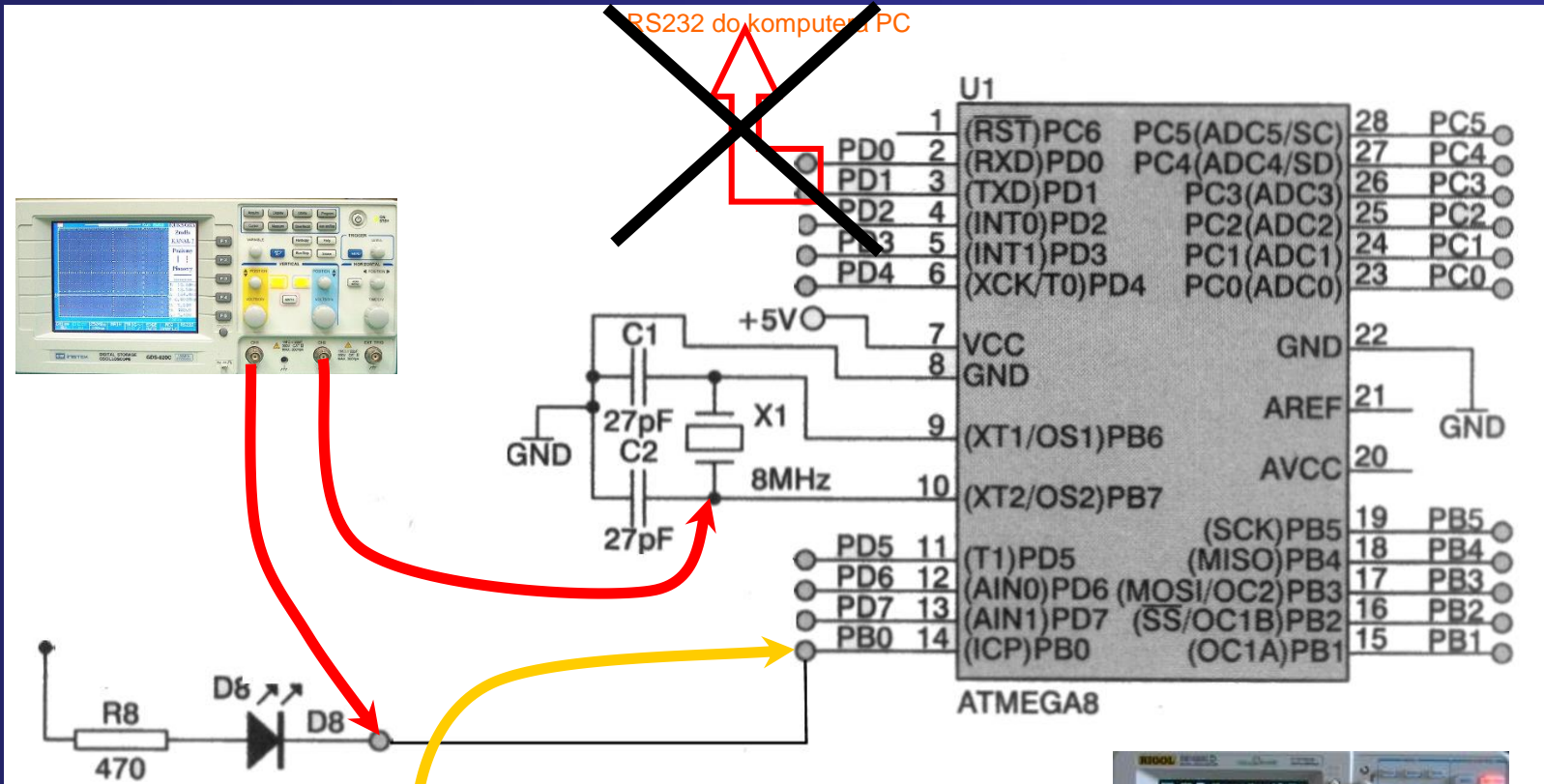
```
Mikroprocesory\Bascom Colege\basAVR_listingi\Dzi...  
Label  
$regfile = "m8def.dat"  
$crystal = 8000000  
  
Config Pinb.0 = Output  
  
Do  
Set Portb.0  
Reset Portb.0  
  
Loop  
End
```





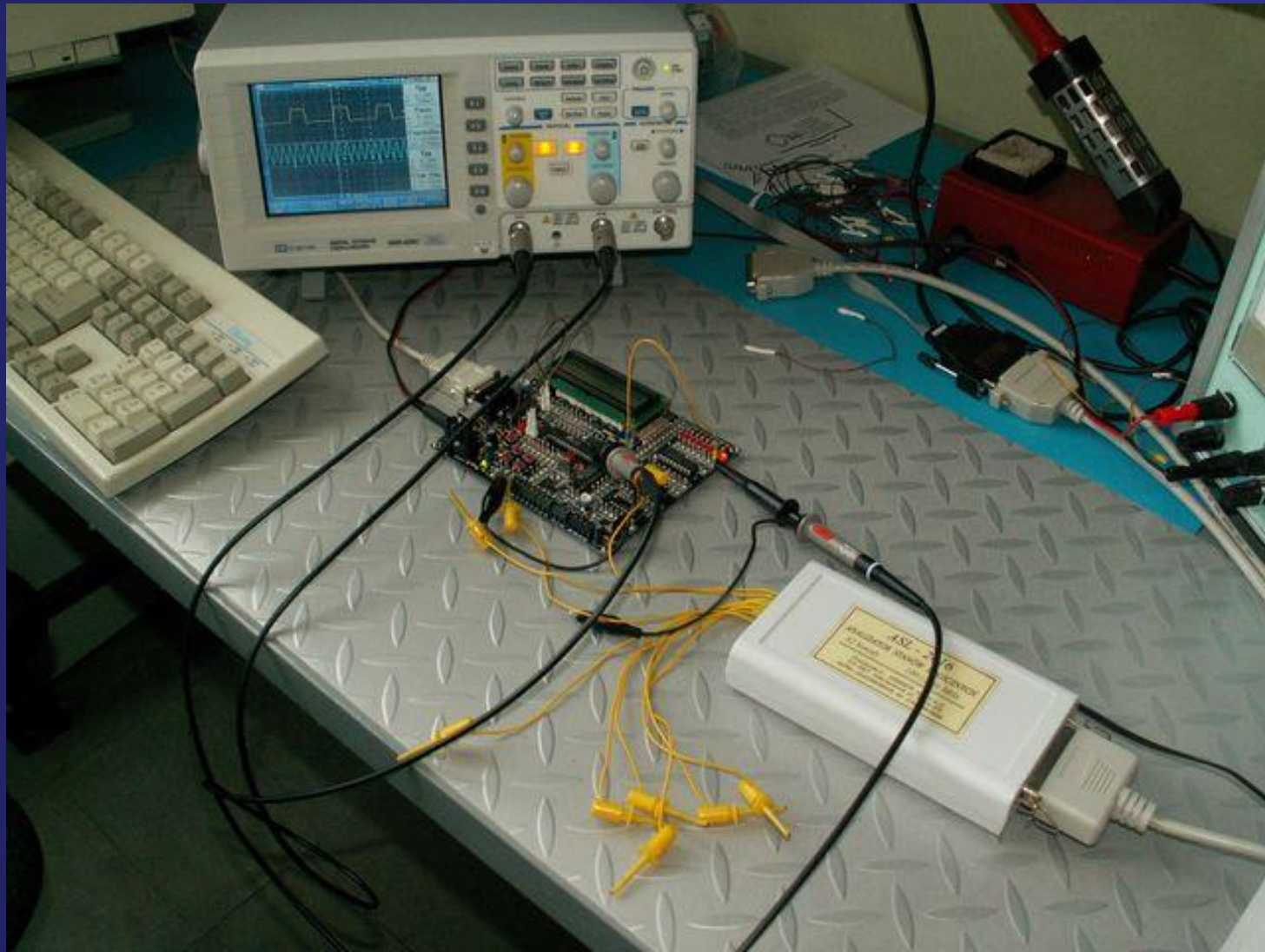
# Program 0

Przebadanie zależności czasowych podczas rzeczywistej pracy mikrokontrolera realizującego program0 i porównanie ich z pracą podczas symulacji sprzętowej



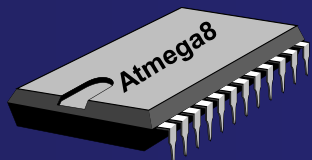
# Program 0

Przebadanie zależności czasowych podczas rzeczywistej pracy mikrokontrolera realizującego program0 i porównanie ich z pracą podczas symulacji sprzętowej



# Program 0 - Assembler

Zaprogramowanie mikrokontrolera programem napisanym w assemblerze realizującym to samo zadanie co Program 0



Wpisanie  
do pamięci Flash ROM  
Programu  
assemblera

```
.include "m8def.inc"
.def temp1 = r16
.def temp2 = r19
.cseg
.org 0x00
rjmp main

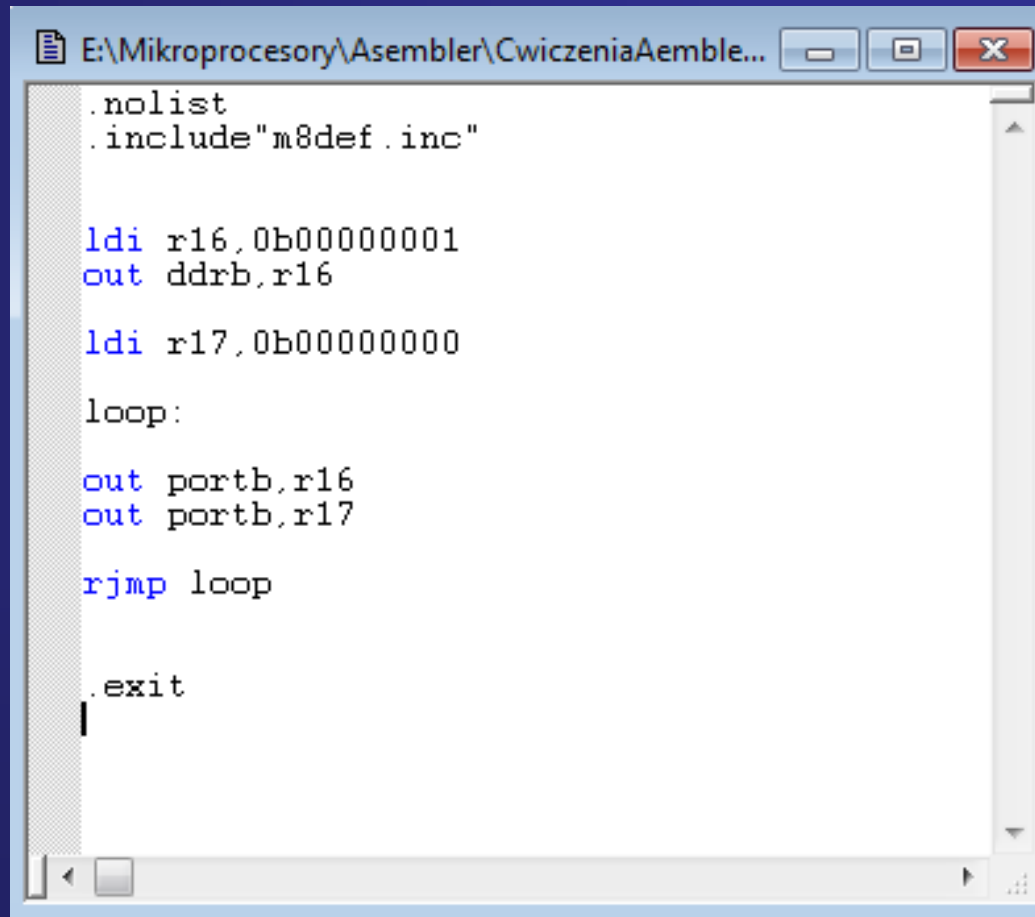
main:
ldi r16, high(RAMEND) ;initialize stackpointer
out SPH, r16
ldi r16, low(RAMEND)
out SPL, r16
cli ;disable global interrupts
ldi temp1, 0x3f ;set PORTB as output (bez 2 najstarszych bitow)
out DDRB, temp1
ldi temp1, 0x3f
ldi temp2, 0x00

loop:
out PORTB, temp1
out PORTB, temp2
rjmp loop ;spowrotem
```



# Program 0 - Assembler

Zaprogramowanie mikrokontrolera programem napisanym w assemblerze realizującym to samo zadanie co Program 0

A screenshot of a text editor window titled "E:\Mikroprocesory\Assembler\CwiczeniaAemble...". The window contains assembly code for an AVR microcontroller. The code includes a preprocessor directive to include "m8def.inc", followed by instructions to load and output values to DDRB, and a loop that outputs values to PORTB. The code ends with an exit instruction.

```
.nolist
#include "m8def.inc"

ldi r16,0b00000001
out ddrb,r16

ldi r17,0b00000000

loop:

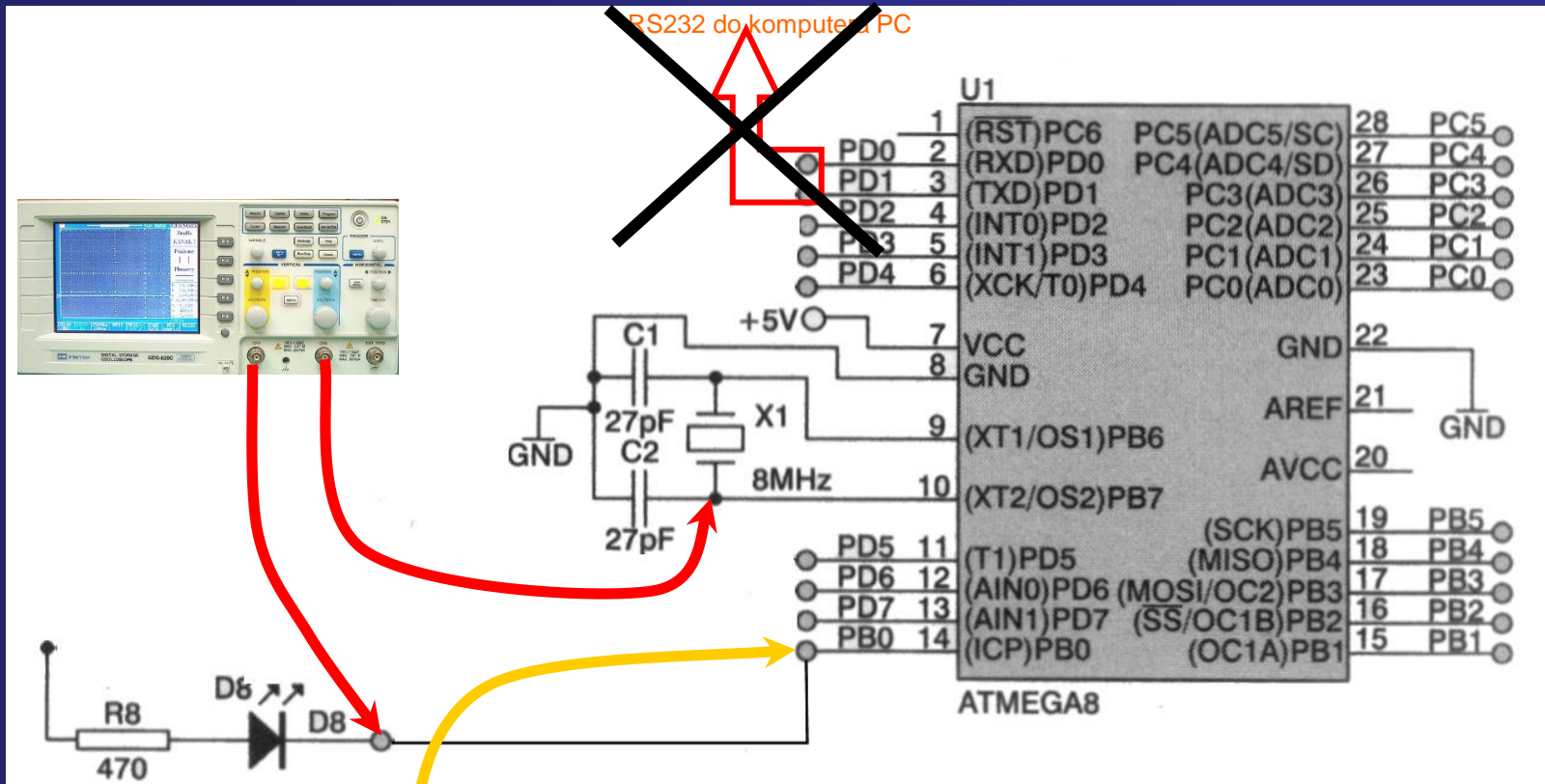
out portb,r16
out portb,r17

rjmp loop

.exit
```

# Program 0- Asembler

Przebadanie zależności czasowych podczas rzeczywistej pracy mikrokontrolera realizującego program napisany w asemblerze i porównanie ich z pracą programu 0 napisanego w Bascom Basic



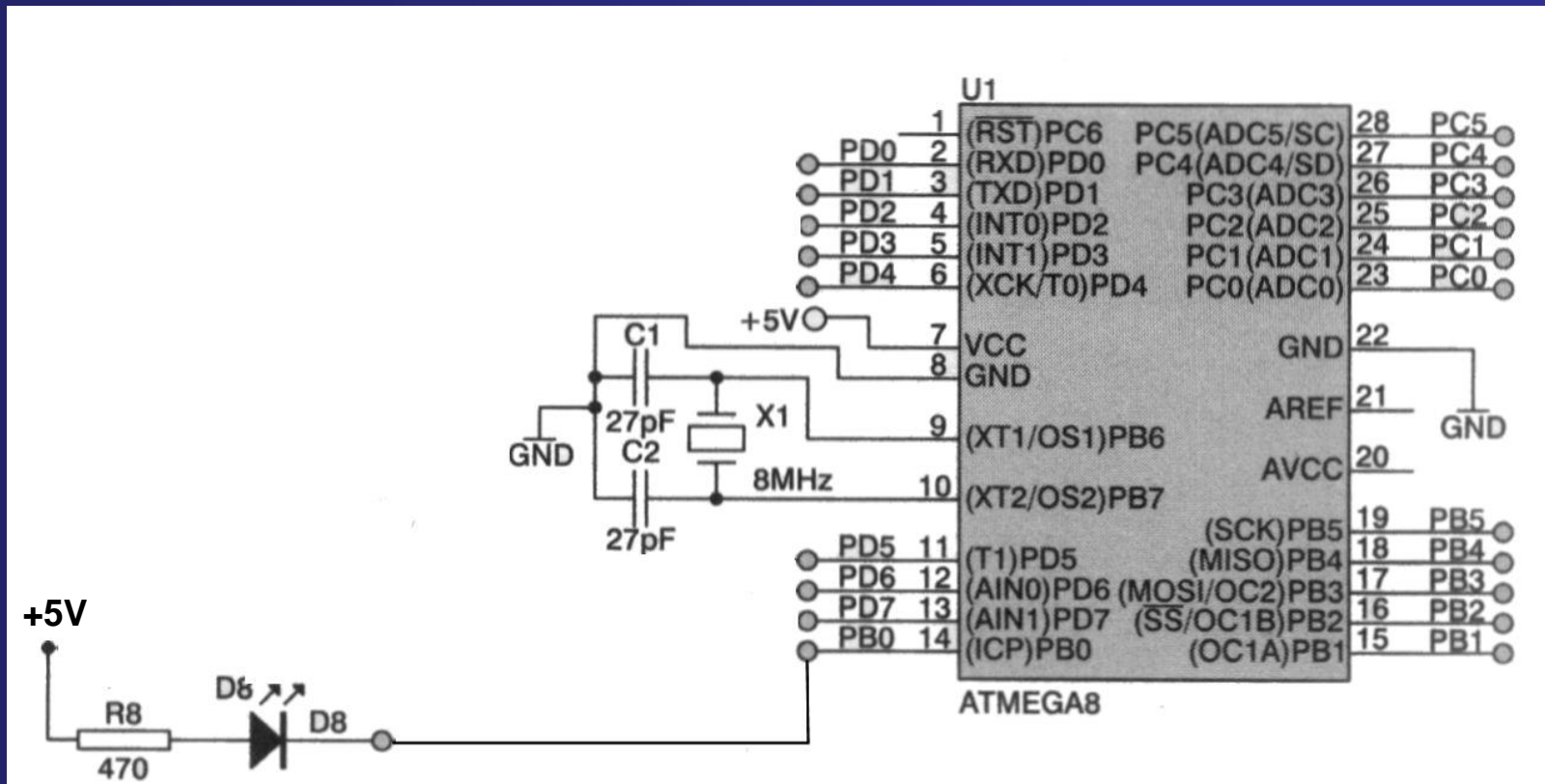
# Program 2

Instrukcja *Toggle* zmieniająca stan portu na przeciwny.

Jest równoznaczna dwóm instrukcją  
*Set* i *Reset*

# Program 2

## Schemat połączenia diody LED do linii PBO portu B mikrokontrolera



Instrukcja *Toggle* zmieniająca stan portu na przeciwny.

# Program 2

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

Config Pinb.0 = Output
Do
Toggle Portb.0

Loop
End
```

informuje kompilator o pliku dyrektyw mikrokontrolera

informuje kompilator o częstotliwości oscylatora  
taktującego mikrokontroler

linia PB0 jako wyjściowa

początek pętli

zmień na przeciwny stan linii PB0 portu B  
mikrokontrolera

koniec pętli głównej programu

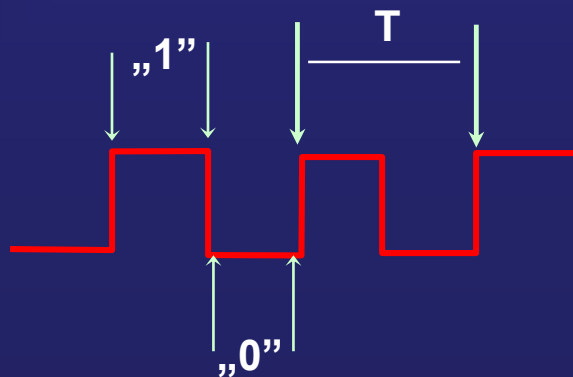
koniec programu



# Cel ćwiczenia 0 - 2

1. Celem przeprowadzonych ćwiczeń jest zapoznanie się z generowaniem impulsów o określonym czasie trwania.

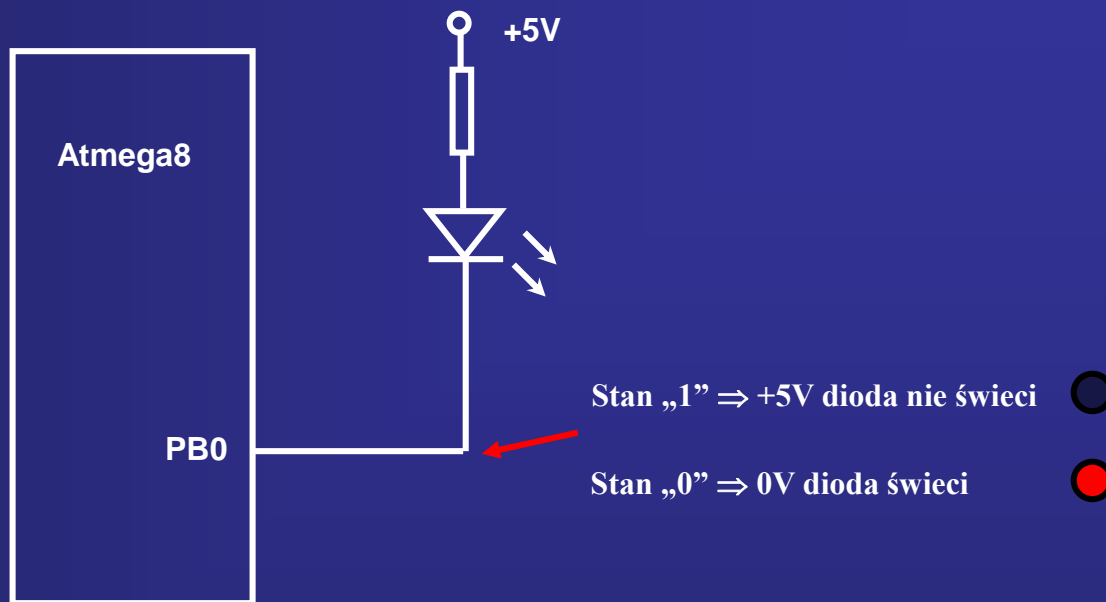
	Program0 - Bascom instrukcje <i>Set</i> , <i>Reset</i>	Program0 - assembler	Program2 – Bascom Instrukcja <i>Toggle</i>
Czas trwania stanu „0” na wyjściu portu PB0			
Czas trwania stanu „1” na wyjściu portu PB0			
T – okres			
f – częstotliwość			



Częstotliwość  
sygnału

$$f = 1/T$$

# Konfiguracja portów



Porty PB, PC, PD mogą bezpośrednio sterować diodami LED, gdyż prąd wpływający (linia portu na poziomie niskim) może mieć wartość

**tylko do 20mA !!!**

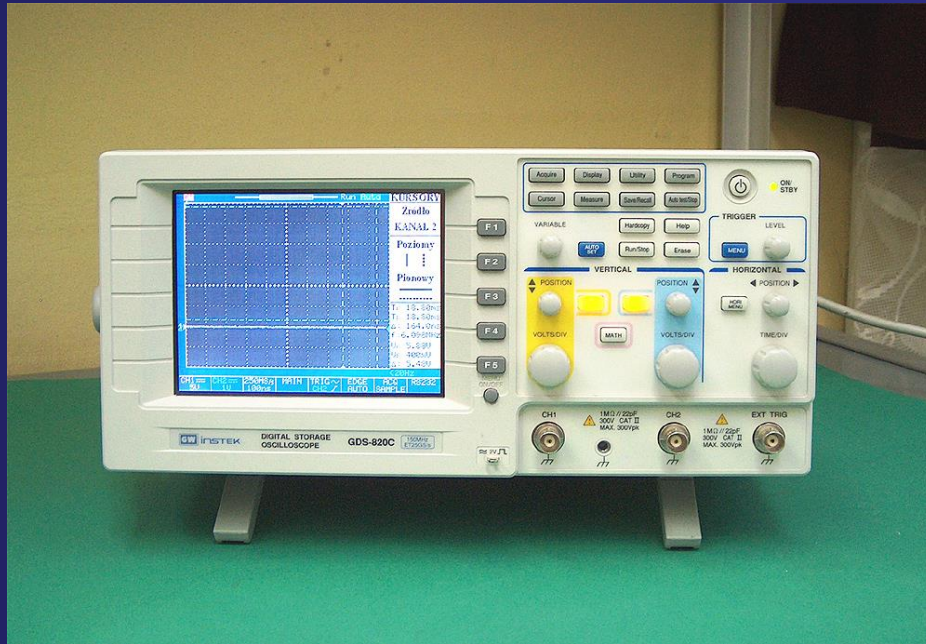
**Uwaga:** Łączna obciążalność prądowa portów mikrokontrolera AVR nie powinna przekraczać 200mA, gdyż może nastąpić jego uszkodzenie



WYDZIAŁ FIZYKI  
i INFORMATYKI STOSOWANEJ  
Uniwersytet Łódzki

# *Przyrządy używane na pracowni*

# Przyrządy używane na pracowni

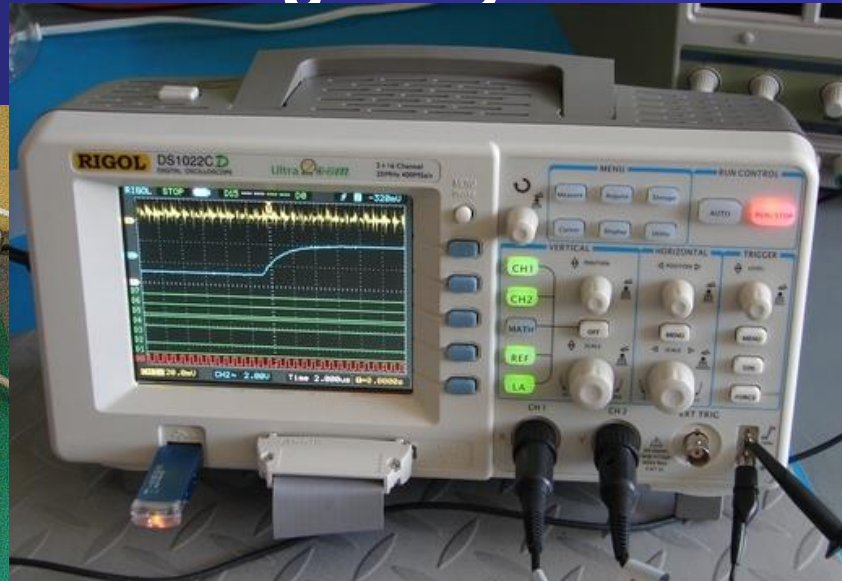


## Oscyloskop cyfrowy

Możemy rejestrować przebiegi napięć zmienne w czasie oraz zapisywać je w pamięci wewnętrznej oscyloskopu

# Przyrządy używane na pracowni

## Analizator Stanów Logicznych



Możemy rejestrowane w czasie stany logiczne („0”, „1”) oraz zapamiętywać je w pamięci wewnętrznej analizatora

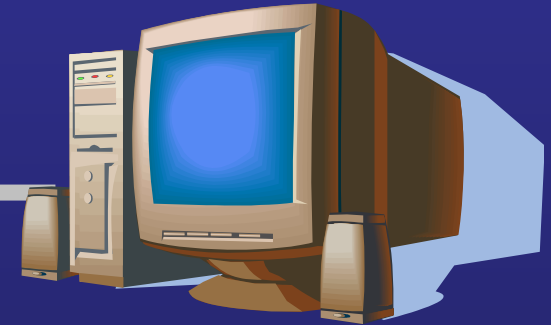


# Przyrządy używane na pracowni

## Analizator Stanów Logicznych ASL - 2016



Połączenie  
analizatora z  
komputerem



Port drukarkowy

LPT1

Adres 378H

Tryb pracy portu  
EPP



# Analizator Stanów Logicznych

## ASL - 2016

LOGIC ANALYZER

data grid trigger setup clear redraw group\_enable print bus up down port

CLK. FREQ. 10 MHz TIME BASE (us/div) 5 HOR. SHIFT < > REF. TIME 0.0000 us TIME 0.0000 us DIFF. TIME 0.000 us TOP BUS

RESET STORED ZOOM 1 1 2 5 SHIFT STEP 50 TRESHOLD (V) 1.80 PRE / POST 0 CLOCK INT BOTTOM BUS

A1	0
A2	0
A3	0
A4	0
A5	0
A6	0
A7	0
A8	0
B1	0
B2	0
B3	0
B4	0
B5	0
B6	0
B7	0
B8	0
C1	0
C2	0
C3	0
C4	0
C5	0
C6	0
C7	0
C8	0
D1	0
D2	0

group\_enable

- A enable
- B enable
- C enable
- D enable
- all enable

LPT1 (378 H)

LPT2 (278 H)

LPT3 (3BC H)

# Analizator Stanów Logicznych ASL - 2016

LOGIC ANALYZER

data grid trigger setup clear redraw group\_enable print bus up down port

CLK FREQ. **10 MHz** TIME BASE (us/div) **5** HOR. SHIFT **< >** REF. TIME **0.0000 us** TIME **0.0000 us** DIFF. TIME **0.000 us** TOP BUS

RESET **STORED** ROOM **1** SHIFT STEP **50** TRESHOLD (V) **1.80** PRE / POST **0** CLOCK **INT** BOTTOM BUS

A1	1	0
A2	2	0
A3	3	0
A4	4	0
A5	5	0
A6	6	0
A7	7	0
A8	8	0
B1		0
B2		0
B3		0
B4		0
B5		0
B6		0
B7		0
B8		0
C1		0
C2		0
C3		0
C4		0
C5		0
C6		0
C7		0
C8		0
D1		0
D2		0

### TRIGGER SETTING

	A1	A2	A3	A4	A5	A6	A7	A8	B1	B2	B3	B4	B5	B6	B7	B8	
1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1
0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	0
X	<input type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	X

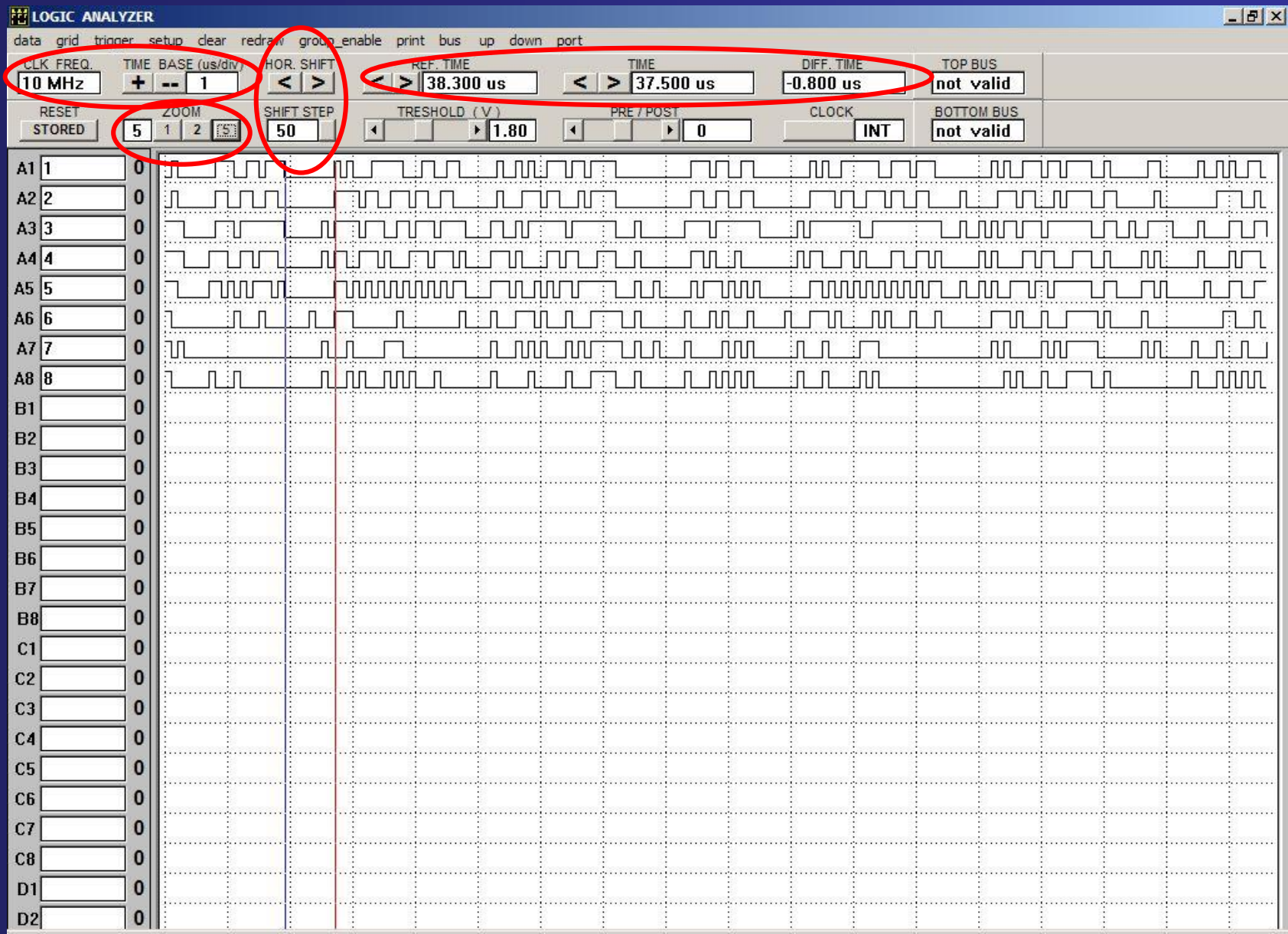
OK  Cancel





# Analizator Stanów Logicznych

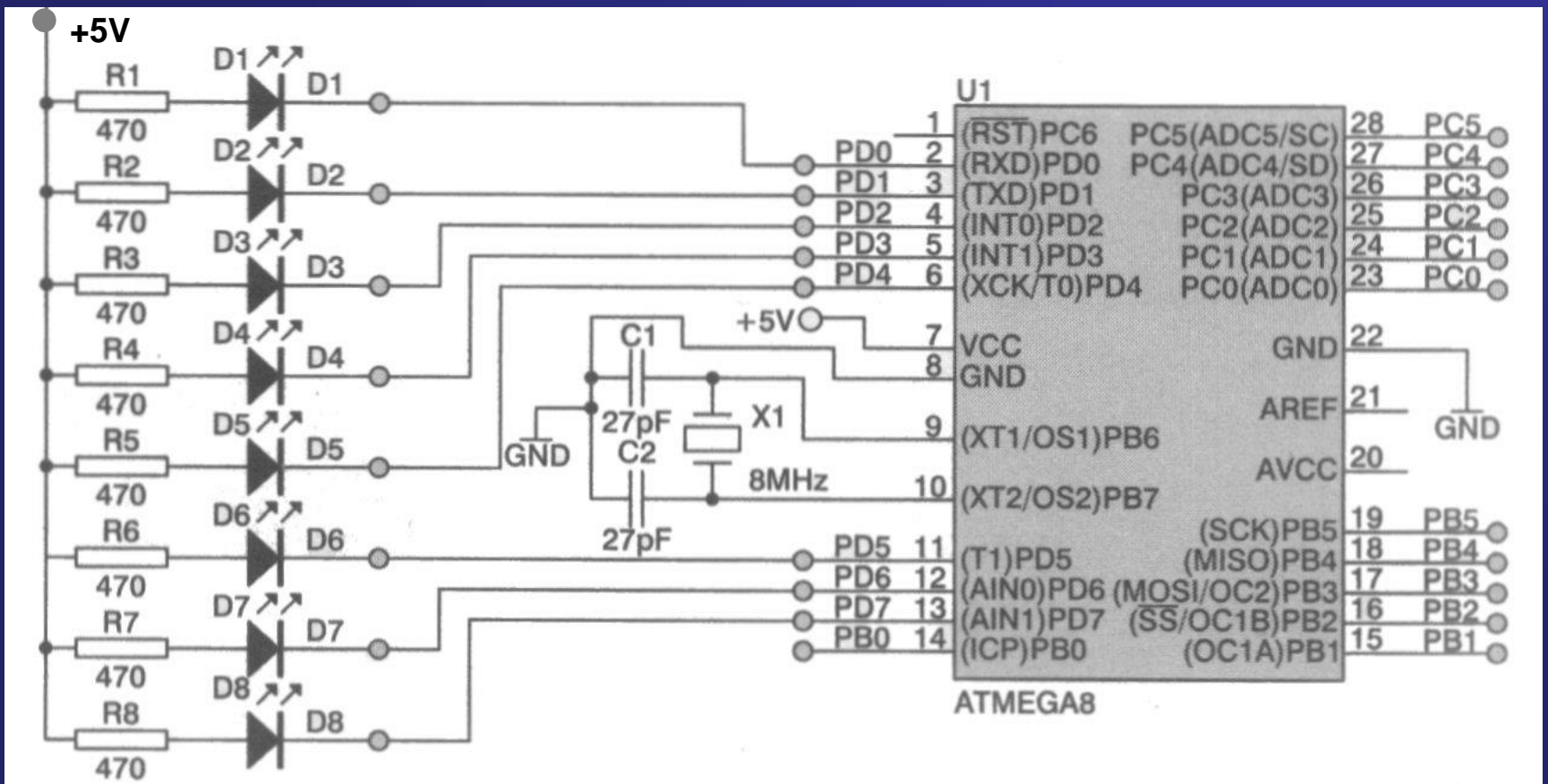
## ASL - 2016



Na koniec  
Program 1  
dodatkowy  
wąż świetlny

# Program 1 dodatkowy

## Schemat dołączenia diod LED do mikrokontrolera



# Program węzła świetlnego sterującego 8-mioma diodami LED dołączonymi do portu D mikrokontrolera

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

Config Portd = Output

Portd = &B01010101

Do
Rotate Portd , Left
Waitms 200
Loop
End
```

informuje kompilator o pliku dyrektyw mikrokontrolera

informuje kompilator o częstotliwości oscylatora taktującego mikrokontroler

linia PB0 jako wyjściowa

wartość początkowa wpisana do portu wyjściowego D, dziesiętnie = 85, hex = 55

początek pętli

przesuwaj wpisane wartości do portu D w lewo

opóźnienie przesunięć o 200 ms

koniec pętli głównej programu

koniec programu

