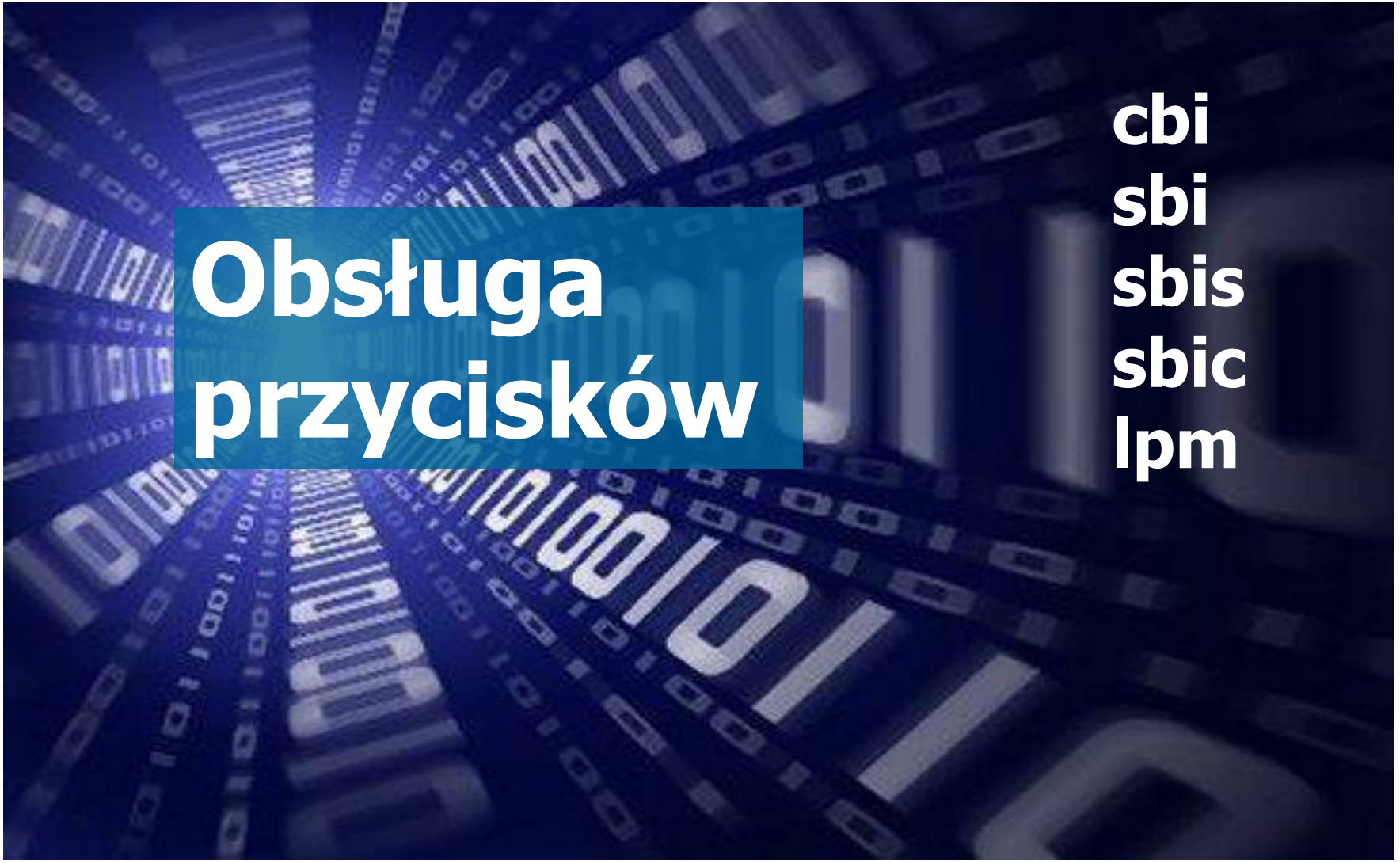

Obsługa przycisków
cbi
sbi
sbis
sbic
lpm

Program – Obsługa klawiatury

CELE

1. **Inicjalizacja portu jako wejście**
2. **Pull-up**
3. **Sterowanie programem w zależności od stanu klawiatury**

Instrukcja cbi

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
cbi	**Clear Bit in I/O register** Zeruj bit w rejestrze funkcyjnym	PC ← PC+1 (PI+0x20)(b) ← 0	- - - - - - - -

składnia
cbi P1,b

Przykłady

```
cbi     ddra,dda0
cbi     porta,porta0
```

Instrukcja sbi

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
sbi	**Set Bit in I/O register** ustaw bit w rejestrze funkcyjnym	PC ← PC+1 (PI+0x20)(b) ← 1	- - - - - - - -

składnia
sbi P1,b

Przykłady

```
sbi     ddra,dda0
sbi     porta,porta0
```

Oznaczenia bitów w portach

```
.nolist

.include"m32def.inc"
.list
.cseg
.org 0x0000


        ldi     R16, high(ramend)
        out     SPH, R16
        ldi     R16, low(ramend)
        out     SPL, R16

        ldi     R16, 0b00010011
        out     ddra, R16


        ldi     R16, 0b00010011
        out     porta, R16
again:
        jmp     again

.exit
```

```
.nolist

.include"m32def.inc"
.list
.cseg
.org 0x0000


        ldi     R16 ,high(ramend)
        out     SPH, R16
        ldi     R16, low(ramend)
        out     SPL, R16

        sbi     ddra, dda0
        sbi     ddra, dda1
        sbi     ddra, dda4

        ldi     R16, 0b00010011
        out     porta, R16
again:
        jmp     again

.exit
```

Instrukcja sbic

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
sbic	**Skip if Bit in I/O register Cleared** Pomiń kolejną instrukcję jeśli bit w rejestrze funkcyjnym jest wyzerowany	(PI+0x20)(b)=1 to PC=PC+1 (PI+0x20)(b)=0 to PC=PC+2 lub 3	- - - - - - - -

składnia
sbic PI,b

Przykłady

sbic pina,pina2

Instrukcja sbis

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
sbis	**Skip if Bit in I/O register Set** Pomiń kolejną instrukcję jeśli bit w rejestrze funkcyjnym jest ustawiony	(PI+0x20)(b)=0 to PC=PC+1 (PI+0x20)(b)=1 to PC=PC+2 lub 3	- - - - - - - -

składnia
sbis PI,b

Przykłady

sbis pina,pina2

Instrukcja sbis

```
        ldi     R16, 0b00000001
        out     porta, R16

        sbis    pinb, pinb0
        rjmp    skok

        ldi     R16, 0b00000000
        out     porta, R16
Skok:
```

Pomiń kolejną instrukcję

Jeśli w rejestrze I/O PINB bit nr0 jest ustawiony czyli jest 1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0	0	1	0	0	0	0	1

Rejestr I/O PINB

Instrukcja sbic

```
        ldi     R16, 0b00000001
        out     porta, R16

        sbic    pinb, pinb0
        rjmp    skok

        ldi     R16, 0b00000000
        out     porta, R16
Skok:
```

Pomiń kolejną instrukcję

Jeśli w rejestrze I/O PINB bit nr0 nie jest ustawiony czyli jest 0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0	0	1	0	0	0	0	0

Rejestr I/O PINB

Instrukcja lpm

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
lpm	**Load Program Memory** Odczytaj pamięć programu	PC ← PC+1 Rd ← (Z)	- - - - - - - -

Mnemonik	Rozwinięcie	Operacja	SREG I T H S V N Z C
lpm	**Load Program Memory** Odczytaj pamięć programu	PC ← PC+1 Rd ← (Z) Z=Z+1	- - - - - - - -

składnia

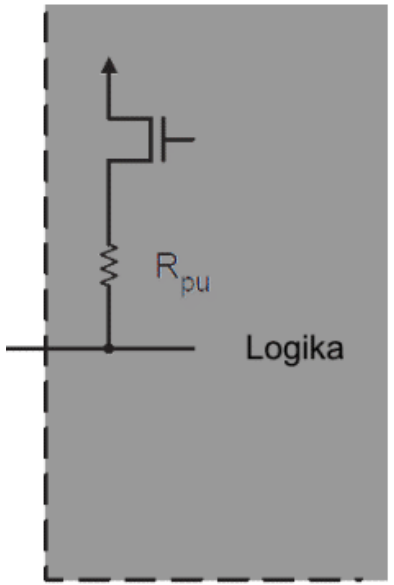
```
lpm     Rd,Z
lpm     Rd,Z+
```

Przykłady

```
lpm     r16,z+
lpm     r17,z
```

Porty B – tryb wejściowy

kilka uwag o portach wejściowych
podciąganie linii pull-up



Na liniach pracujących jako wejścia istnieje możliwość uaktywnienia wewnętrznego rezystora podciągającego (20-50 kOhm) do napięcia zasilania

```
sbi     portb,pb4       ; tylko linia 4 portu B z włączonym pull-up
nop                     ; ustalenie stanu
nop                     ; ustalenie stanu

cbi     portb,pb4       ; tylko linia 4 portu B z wyłączonym pull-up
nop                     ; ustalenie stanu
nop                     ; ustalenie stanu
```

Porty B – tryb wejściowy

Po co nam podciąganie linii

Np. do sprawdzania stanu klawiatury 4 klawiszowej

W tej konfiguracja przyciski dołączone są jedną końcówka do mikrokontrolera a drugą do masy.
W zestawie ZL3AVR JP3 - zwarte.
Niech linie nieaktywnych klawiszy będą w stanie wysokim, jak wciśniemy klawisz to zmienią się na stan niski

```
ldi     R16, 0xff       ; port B z włączonym pull-up
out     portb, R16
nop                     ; ustalenie stanu
nop                     ; ustalenie stanu
```

Program z użyciem przycisków

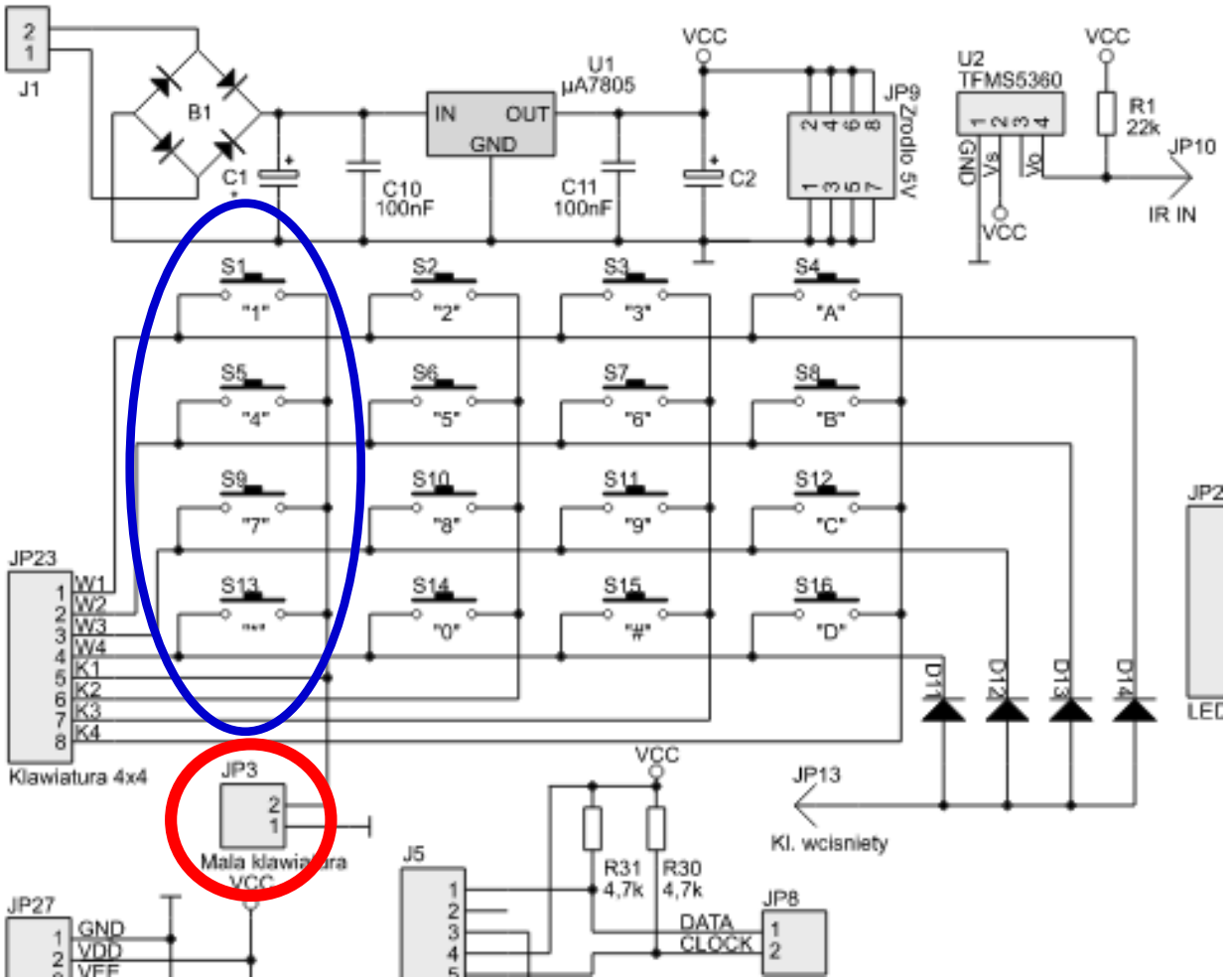
Napisać program, który odczytuje stan klawiatury.
Za pomocą klawiszy sterujemy włączeniem i wyłączeniem diod.
Włącz diodę D9 dla testu, następnie:

Wciśnięty klawisz S1 włącz diodę D2
Puszczenie klawisza S1 wyłączy diodę D2

klawisz S5 włącz diodę D3
Puszczenie klawisza S5 wyłączy diodę D3

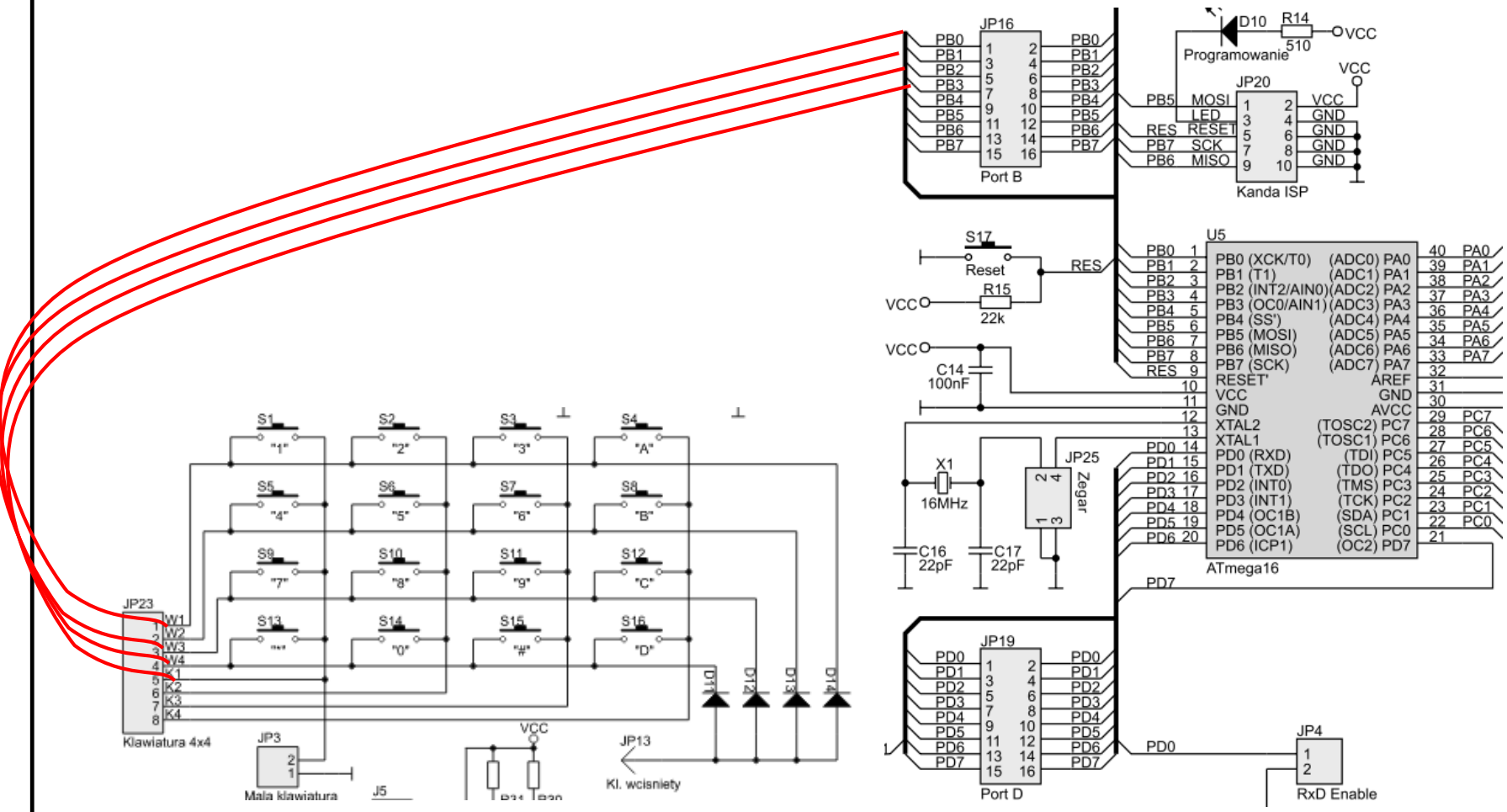
klawisz S9 włącz diodę D4
Puszczenie klawisza S9 wyłączy diodę D4

Klawiatura– schemat połączeń

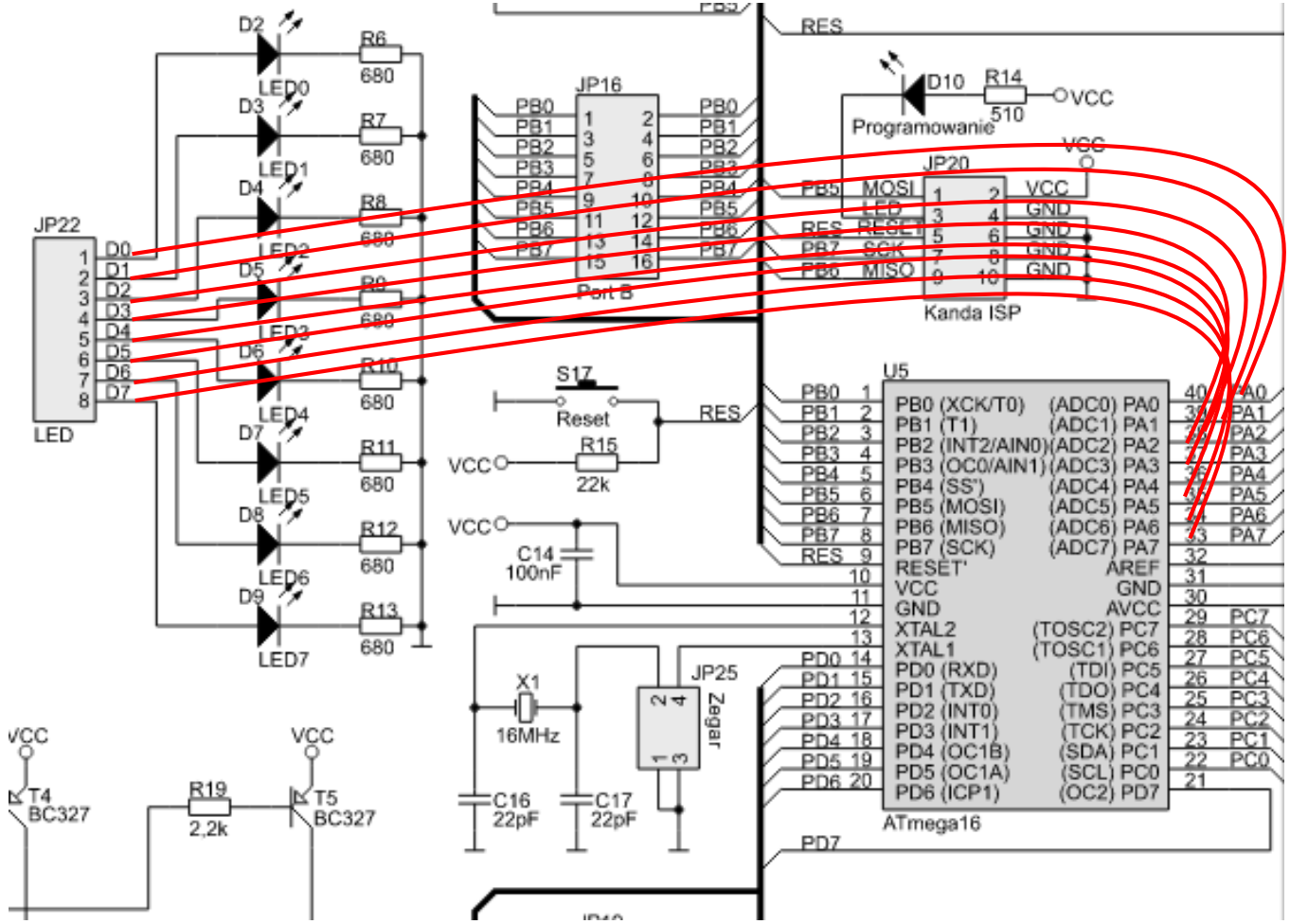


WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Klawiatura– schemat połączeń



Diody – schemat połączeń



Przyciski – schemat połączeń i konfiguracji ZL3AVR

Obwód ćwiczeniowy:

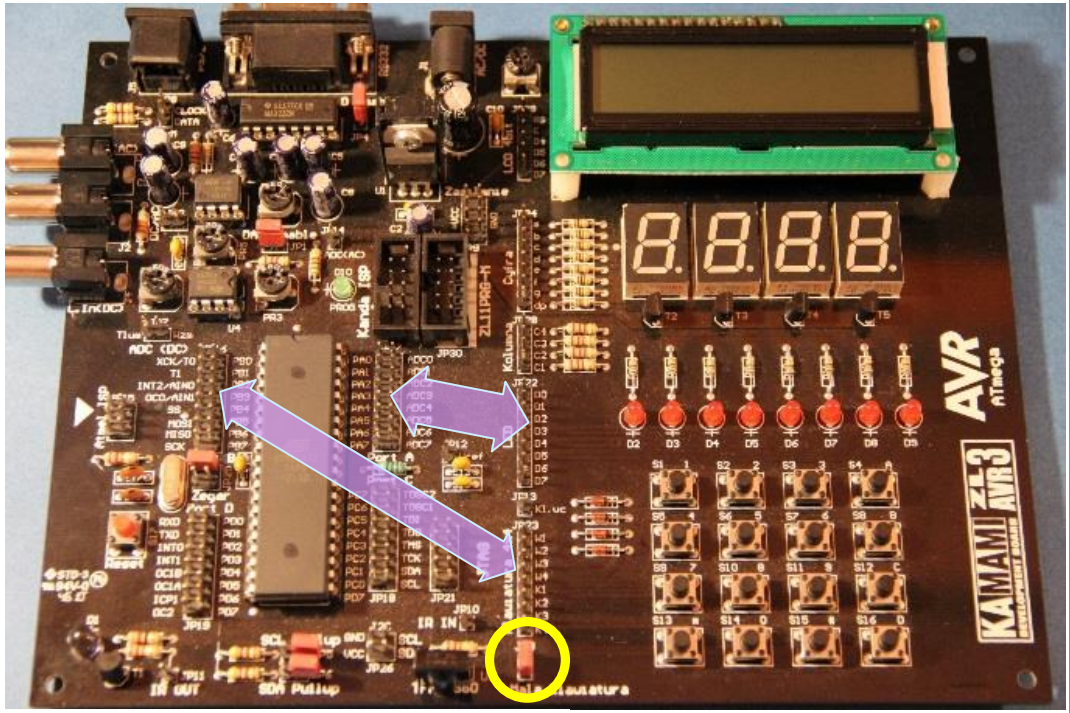
W zestawie ZL3AVR należy:

a) połączyć diody LED z portem PA mikrokontrolera (JP22 – JP17),

b) połączyć wyprowadzenia wierszowe klawiatury matrycowej (W1...W4) z młodszą częścią portu PB mikrokontrolera (JP23:W1...W4 – JP16:PB0...PB3),

c) zmniejszyć klawiaturę matrycową do czteroklawiszowej (JP3 – zwarte),

d) bity konfiguracyjne mikrokontrolera zaprogramować następująco (wykorzystujemy wewnętrzny oscylator nastawiony na częstotliwość 1 MHz):



OCDEN	JTAGEN	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST
*	*	*			*	*	
BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
*	*	*	*	*	*	*	

* – bit zaprogramowany (o zerowej wartości logicznej)

Zadanie

```
.nolist
.include"m32def.inc"
.list
.cseg
.org 0x0000

            ldi         R16, 0b11111111                 ; port A jako wyjście
            out         ddra, R16
            ldi         R16, 0b00000000                 ; port B jako wejście
            out         ddrb, R16
            ldi         R16 ,0b11111111                 ; włącz pull-up w porcie B
            out         portb, R16
            nop                                 ; oczekiwanie na ustalenie stanu
            nop                                 ; oczekiwanie na ustalenie stanu

Petla:

sbis        pinb,pinb0      ;jesli jest 1 na PB0 to pomiń następną linię, jeśli 0 (czyli wciśnięty S1) to skocz do czekaj_na_S1
rcall       czekaj_na_S1
sbis        pinb,pinb1       ;jesli jest 1 na PB1 to pomiń następną linię, jeśli 0 (czyli wciśnięty S2) to skocz do czekaj_na_S2
rcall       czekaj_na_S5
sbis        pinb,pinb2      ;jesli jest 1 na PB2 to pomiń następną linię, jeśli 0 (czyli wciśnięty S3) to skocz do czekaj_na_S3
rcall       czekaj_na_S9

rjmp Petla
```

Zadanie

```
czekaj_na_S1:

        ldi     R16, 0b00000001     ; włącz diodę D2
        out     porta, R16
        sbis    pinb, pinb0         ; czekaj aż zostanie puszczony klawisz S1
        rjmp    czekaj_na_S1
        ldi     R16, 0b00000000     ; wyłącz diodę D2
        out     porta, R16
ret

czekaj_na_S5:

        ldi     R16, 0b00000010     ; włącz diodę D3
        out     porta, R16
        sbis    pinb, pinb1         ; czekaj aż zostanie puszczony klawisz S5
        rjmp    czekaj_na_S5
        ldi     R16, 0b00000000     ; wyłącz diodę D3
        out     porta, R16
ret

czekaj_na_S9:

        ldi     R16, 0b00000100     ; włącz diodę D4
        out     porta, R16
        sbis    pinb, pinb1         ; czekaj aż zostanie puszczony klawisz S9
        rjmp    czekaj_na_S9
        ldi     R16, 0b00000000     ; wyłącz diodę D4
        out     porta, R16
ret
.exit
```

Pamięć programu – odczyt zawartości pamięci *flash*

Instrukcja lpm pozwala na odczyt zawartości pamięci programu w której mogą znajdować się oprócz kodów programu wartości stałe np. tablice danych.

Do adresowania pamięci programu wykorzystujemy tryb adresowania pośredniego z użyciem rejestru indeksowego Z z ewentualną postinkrementacją. Zawartość pamięci programu umieszczana może być w dowolnym rejestrze roboczym Rd.

Zadanie

Napisać program, który odczytuje zawartość pamięci flash od adresu 0x0000 i wyświetla ją na linijce diodowej. Użyjemy 2 klawiszy S1 i S5 klawiatury do sterowania wyświetlania bajt po bajcie pamięci flash.

Po włączeniu układu stan linijki diodowej odzwierciedlać będzie wartość przechowywaną w młodszej części pierwszej komórki pamięci programu. Po naciśnięciu klawisza S1 zobaczymy bajt MSB tej komórki, a późniejsze wciśnięcie S5 spowoduje przejście do LSB komórki drugiej itd. W ten sposób możemy przeglądnąć pamięć programu i zobaczyć, jak ciężką pracę ma translator, zamieniający względnie przyjemne mnemoniki w ciąg zer i jedynek. Jeśli posłużymy się dokumentacją listy rozkazów obsługiwanych przez rdzeń AVR (zawiera ona m.in. wartości kodów przyporządkowywanych instrukcjom – ang. opcodes), program ćwiczeniowy pozwoli sprawdzić poprawność translacji.

Prawdopodobnie lepszym pomysłem okaże się jednak wyjście na spacer.

Zadanie

Pamięć programu – odczyt zawartości pamięci *flash*

```
.INCLUDE "m32def.inc"

            ldi     R16, 0b11111111
            out     DDRA, R16           ; port PA pracuje w trybie wyjściowym

            ldi     R16, 0b00001111
            out     PORTB, R16          ; podciąganie na liniach PB0..3 włączone

            ldi     ZL, low(0)          ; rejestr indeksowy Z wskazuje na młodszy bajt
            ldi     ZH, high(0)         ; pierwszej komórki pamięci programu

Petla:
            lpm     R0, Z+              ; odczyt komórki pamięci flash wskazywanej przez
            out     PORTA, R0           ; rejestr indeksowy Z i jego postinkrementacja
                                        ; wyświetlenie odczytanej wartości na linijce
                                        ; diodowej, dołączonej do portu PA
Czekaj_na_S1:
            sbic    PINB, PINB0         ; oczekiwanie na wciśnięcie klawisza S1,
            rjmp    Czekaj_na_S1        ; dołączonego do linii PB0

            lpm     R0, Z+
            out     PORTA, R0           ; wyświetlenie zawartości następnej komórki

Czekaj_na_S5:
            sbic    PINB, PINB1         ; oczekiwanie na wciśnięcie klawisza S5,
            rjmp    Czekaj_na_S5        ; dołączonego do linii PB1

rjmp        Petla                       ; zapętlenie programu (skok do instrukcji
                                        ; następującej po etykiecie "Petla")
```