|  |  |  |  |
| --- | --- | --- | --- |
| **Wydział Elektrotechniki i Informatyki PL** | **Sprawozdanie** | | |
| **Wykonał:**  Jakub Łabendowicz | **Semestr:**  V | **Grupa:**  IIST 5.4 IO | **Rok akademicki:**  2021/2022 |
| **Temat:** PIO\_driver | | **Data:**  09-12-2021 | |

#include <targets\AT91SAM7.h>

void **time\_delay**(int ms){

volatile int aa,bb;

for(aa=0;aa<=ms;aa++){

for(bb=0;bb<=3000;bb++){

\_\_asm\_\_("NOP");

}}}

**//Sterownik (driver) traktowany jest jako zestaw funkcji przeznaczonych do obsługi danego urządzenia peryferyjnego.**

**//Sterowniki powinny realizować następujące funkcje:**

**//pio\_pcer 1 dla enable, 0 dla diable; a\_b: 0 - PIOA, 1 - PIOB**

void **PIO\_clock\_enable**(int pio\_pcer, int a\_b){

if(pio\_pcer==1)

{

PMC\_PCER|=pio\_pcer<<a\_b+2;

}

if(pio\_pcer==0)

{

PMC\_PCDR|=1<<a\_b+2;

}

}

**//załączanie kontroli nad wybranym PINem dla PIO Controllera (np. PIN 65 kontroluje PB20), dwa parametry: nr linii i stan 0/1 gdzie**

**//1 załącza kontrolę PIO dla danej linii; 0 wyłącza kontrolę PIO dla danej linii**

void **PIO\_enable**(int line\_no, int ena)

{

if(ena==1)

{

PIOB\_PER = 1<<line\_no; //pod kontrolą I/O controllera

}

else if(ena == 0)

{

PIOB\_PDR = 1<<line\_no; //usun kontrole I/O controllera

}

}

**//wybór czy dana linia ma być output czy input - dwa parametry, nr linii i stan: 0-input, 1-output**

void **PIO\_output\_enable**(int line\_no, int ino)

{

if(ino==1)

{

PIOB\_ODR=1<<line\_no; // jako INPUNT

}

else if(ino==0)

{

PIOB\_OER=1<<line\_no; //jako OUTPUT

}

}

**// ustawienie stanu wybranej linii, dwa parametry, nr linii i stan: 0-cleared, 1-set**

void **PIO\_output\_state**(int line\_num, int state)

{

if(state==0)

{

PIOB\_CODR=1<<line\_num; //clear

}

else if(state==1)

{

PIOB\_SODR=1<<line\_num;//set

}

}

**//neguje stan linii output, pamiętać o konfiguracji w OWER - negacja może być przeprowadzona**

void **PIO\_output\_negate**(int numer\_linii)

{

PIOB\_OWER |= (1<<numer\_linii);

PIOB\_ODSR ^= 1<<numer\_linii;

}

**//Funkcje odczytu przycisków**

**// czytaj stan przycisku SW1 lub SW2**

unsigned int **SW\_odczyt**(unsigned int SW\_numer)

{

if((PIOB\_PDSR & (1<<SW\_numer))==0)

{

return 1;

}

else

{

return 0;

}

}

**// funkcja czeka dopóki przycisk jest naciśnięty**

void **SW\_czytaj**(int Sw)

{

while((PIOB\_PDSR & (1<<Sw)) !=0)

{

\_\_asm\_\_("NOP");

}

}

int **main**()

{

PIO\_clock\_enable(1,1);

PIO\_enable(20,1);

PIO\_enable(24,1);

PIO\_enable(25,1);

PIO\_output\_enable(20,1);

**// Przetestować linię OUTPUT - zmiana stanu LCD\_BL co 500ms (użyć time\_delay)**

while(1){

time\_delay(500);

PIO\_output\_negate(20);

}

**//Program ma reagować na wciskanie klawisza SW1 (PB24) zapaleniem LCD\_BL(PB20) i klawisza SW2 gaszeniem LCD\_BL(PB20)**

while(1)

{

SW\_czytaj(24);

if(SW\_odczyt(1)==0)

{

PIO\_output\_state(20,0);

}

else if (SW\_odczyt(2)==0)

{

PIO\_output\_state(20,1);

}

}

**//Program ma reagować zmianą stanu LCD\_BL na każde naciśnięcie klawisza SW1, jedno naciśnięcie - jedna zmiana stanu. Do tego wykorzystać funkcję SW\_czytaj**

bool zmienna=false;

while(1)

{

SW\_czytaj(24);

if(SW\_odczyt(1)==0)

{ if(zmienna==false)

{

PIO\_output\_state(20,0);

zmienna=true;

}else

{

PIO\_output\_state(20,1);

zmienna=false;

}

}

//time\_delay(100);

}

}