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);
    }
}
```