Sprawozdanie

Systemy wbudowane

Ćwiczenie 8:

Czujniki – pomiar temperatury.

Wykonanie:

Busłowski Tomasz

Suchwałko Tomasz

Skrouba Kamil

Zawadzka Magdalena

(Grupa PS3)

Prowadzący zajęcia: **dr inż. Adam Klimowicz**

SW, semestr VI, 22-05-2017, Wydział Informatyki, Politechnika Białostocka

Zakres Materiału

1. Wewnętrzny czujnik temperatury mikrokontrolera.
2. Czujnik temperatury STLM20.
3. Czujnik temperatury TC77.
4. Interfejs SPI.

Zadania do wykonania

1. Korzystając z przykładu 1 napisz program wyświetlający temperaturę procesora na dowolnym wyświetlaczu.
2. Korzystając z przykładu 2 napisz program wyświetlający temperaturę zmierzoną czujnikiem TC77 na dowolnym wyświetlaczu.
3. Napisz program wyświetlający temperaturę na wyświetlaczu LCD korzystając z czujnika wbudowanego w mikrokontroler i czujnika STLM20. Wyprowadzenie czujnika znajdziesz na złączu Con7 (Temp). Do przetwarzania A/C wykorzystaj tryb wielokanałowy przetwornika. Zdefiniuj symbol stopnia „°” na wyświetlaczu LCD.
4. Napisz program porównujący wskazania czujnika STLM20 dla 2 metod obliczania wartości temperatury (liniowej i nieliniowej).

Zadanie 1

Treść:
Korzystając z przykładu 1 napisz program wyświetlający temperaturę procesora na dowolnym wyświetlaczu.

Realizacja:

- Najpierw utowrzyliśmy projekt na podstawie przykładu 1.
- W tym zadaniu (i w nstępnym) wyświetlaliśmy temperaturę na wyświetlaczu alfanumerycznym LCD, więc podłączyliśmy wyświetlacz tak jak w Ćwiczeniu 5.
- Następnie kod programu z przykładu zmodyfikowaliśmy wykorzystaując kod z Ćwiczenia 5. tak, aby wyświetlał odczytaną temperaturę na wyświetlaczu LCD.
- W celu konwersji wartości z **double** na **tablicę znaków** wykorzystaliśmy funkcję **sprintf** z dodatkowo zaimportowanej biblioteki "cstdio".

Kod programu – plik main.c:

```
#include "stm32f10x.h"
#include "stm32f10x_adc.h"
#include "lcd.h"
#include "cstdio"


void GPIO_Config(void);
void RCC_Config(void);
void NVIC_Config(void);
void ADC_Config(void);


int main(void)
{
  volatile unsigned long int i;
  unsigned long int wartoscADC1  = 0;
  double result = 25.0;
  unsigned char temperatura[5];
  RCC_Config();
      GPIO_Config();
      ADC_Config();

      LCD_Initialize();            // Inicjalizuj LCD
      LCD_WriteCommand(LCD_DISPLAY_ONOFF | LCD_DISPLAY_ON);
      LCD_WriteCommand(0x01); // Wyczysc LCD

      while (1) {
            ADC_SoftwareStartConvCmd(ADC1, ENABLE);
            //wyzwolenie pojedynczego pomiaru
            while (!ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC));
            //odczekaj na zakonczenie konwersji
            wartoscADC1  = ADC_GetConversionValue(ADC1);
            //pobiez zmierzona wartosc
        /*Tu wstaw kod konwertuj¹cy wynik wg. Wzoru i prezentuj¹cy go na
wyœwietlaczu*/
            result = (1430.0-wartoscADC1*0.8067)/4.3 + 25.0;
```

```
    sprintf(temperatura, "%f", result);
           for (i=0; i< 5; i++)                          // Wypisz tekst
                 LCD_WriteData(temperatura[i]);

           for (i=0;i<1000000ul;i++);         //oczekanie
           LCD_WriteCommand(0x01);
      };
      while(1);
  return 0;
}

void RCC_Config(void)
//konfigurowanie sygnalow taktujacych
{
  ErrorStatus HSEStartUpStatus;  //zmienna opisujaca rezultat uruchomienia
HSE

  RCC_DeInit();                                        //Reset ustawien
RCC
  RCC_HSEConfig(RCC_HSE_ON);                          //Wlaczenie HSE
  HSEStartUpStatus = RCC_WaitForHSEStartUp();              //Odczekaj az
HSE bedzie gotowy
  if(HSEStartUpStatus == SUCCESS)
  {
    FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);//
    FLASH_SetLatency(FLASH_Latency_2);                 //ustaw zwloke dla
pamieci Flash; zaleznie od taktowania rdzenia
                                                          //0:<24MHz;
1:24~48MHz; 2:>48MHz
    RCC_HCLKConfig(RCC_SYSCLK_Div1);                   //ustaw
HCLK=SYSCLK
    RCC_PCLK2Config(RCC_HCLK_Div1);
//ustaw PCLK2=HCLK
    RCC_PCLK1Config(RCC_HCLK_Div2);
//ustaw PCLK1=HCLK/2
    RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9); //ustaw PLLCLK =
HSE*9 czyli 8MHz * 9 = 72 MHz
    //RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_7); //ustaw PLLCLK =
HSE*7 czyli 8MHz * 7 = 56 MHz - konieczne dla ADC
    RCC_PLLCmd(ENABLE);
//wlacz PLL
    while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);  //odczekaj na
poprawne uruchomienie PLL
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);         //ustaw PLL jako
zrodlo sygnalu zegarowego
    while(RCC_GetSYSCLKSource() != 0x08);              //odczekaj az PLL
bedzie sygnalem zegarowym systemu

      /*Tu nalezy umiescic kod zwiazny z konfiguracja sygnalow zegarowych
potrzebnych w programie peryferiow*/
    RCC_ADCCLKConfig(RCC_PCLK2_Div6);                  // ADCCLK =
PCLK2/6 = 72 MHz /6 = 12 MHz

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//wlacz taktowanie
portu GPIO A
```

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);//wlacz taktowanie
portu GPIO B
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);//wlacz taktowanie
portu AFIO
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE); //wlacz taktowanie
ADC1

  } else {
  }
}


void GPIO_Config(void)
{
  //konfigurowanie portow GPIO
      GPIO_InitTypeDef  GPIO_InitStructure;
      // disable JTAG
      GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
      /*Tu nalezy umiescic kod zwiazany z konfiguracja poszczegolnych
portow GPIO potrzebnych w programie*/
      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 ;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
      GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void ADC_Config(void)
{
  //konfigurowanie przetwornika AC
      ADC_InitTypeDef ADC_InitStructure;

      ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
//Jeden przetwornik, praca niezalezna
      ADC_InitStructure.ADC_ScanConvMode = DISABLE;
//Pomiar jednego kanalu, skanowanie kanalow nie potrzebne
      ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
//Pomiar w trybie jednokrotnym
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
      //Brak wyzwalania zewnetrznego
      ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
      //Wyrownanie danych do prawej - 12 mlodszych bitow znaczacych
      ADC_InitStructure.ADC_NbrOfChannel = 1;
//Liczba uzywanych kanalow =1
      ADC_Init(ADC1, &ADC_InitStructure);
//Incjalizacja przetwornika

      ADC_RegularChannelConfig(ADC1, ADC_Channel_16, 1,
      ADC_SampleTime_239Cycles5);

//Kanal 16 - wewnetrzny czujnik temp. procesora

//Grupa podstawowa, czas probkowania 239,5+12,5=252 cykli = 18us (calkowity
czas przetwarzania)
```

```
//239,5 => 17.1us = czas zalecany w dokumentacji MUC dla miernika
temperatury

      ADC_TempSensorVrefintCmd(ENABLE);
//Wlaczenie czujnika temperatury w procesorze

      ADC_Cmd(ADC1, ENABLE);
//Wlacz ADC1

      ADC_ResetCalibration(ADC1);
//Reset rejestrow kalibracyjnych ADC1
      while(ADC_GetResetCalibrationStatus(ADC1));
//Odczekanie na wykonanie resetu
      ADC_StartCalibration(ADC1);
//Kalibracja ADC1
      while(ADC_GetCalibrationStatus(ADC1));
//Odczekanie na zakonczenie kalibracji ADC1
}
```

Zadanie 2

Treść:
Korzystając z przykładu 2 napisz program wyświetlający temperaturę zmierzoną czujnikiem TC77 na dowolnym wyświetlaczu.

Realizacja:

- Realizaja zadania 2 polegała na modyfikacji zadania 1 – również wyświetlaliśmy temperaturę na wyświetlaczu alfanuerycznym LCD.
- Zgodnie z treścią przykładu 2 podłączymyśmy czujnik temperatury TC77: Con15 łączymy z pinami złącza Con8 (PA4 → CS, PA5 → SCK, PA7 → SIO).
- Dodaliśmy do projektu moduł SPI z biblioteki StdPeriph i zaimportowaliśmy w pliku main.c.

Kod programu – plik main.c (zmienione ciało funkcji main.c):

```
#include "stm32f10x.h"
#include "stm32f10x_adc.h"
#include "stm32f10x_spi.h"
#include "lcd.h"
#include "cstdio"

void GPIO_Config(void);
void RCC_Config(void);
void NVIC_Config(void);
void ADC_Config(void);
void SPI_Config(void);

int main(void)
{
  volatile unsigned long int i;
```

```
  unsigned long int wartoscADC1  = 0;
  double result = 25.0;
  unsigned char temperatura[5];
  int temperatura2=12345;
  RCC_Config();
  GPIO_Config();
  ADC_Config();
  RCC_Config();
  NVIC_Config();
  SPI_Config();

  LCD_Initialize();                                          // Inicjalizuj LCD
  LCD_WriteCommand(LCD_DISPLAY_ONOFF | LCD_DISPLAY_ON);
  LCD_WriteCommand(0x01);    // Wyczysc LCD

#define SPI_Mode_Slave_Mask ((unsigned short int)0xFEFB)  //Maska pozwalajaca wyzerowac
bity trybu pracy wprost w rejestrzez SPIx->CR1

       while (1) {
              SPI1->CR1 |= SPI_Mode_Master;                          //Ustaw tryb master -
wymusi to zmiane stanu NSS na niski
              while (SPI_I2S_GetFlagStatus(SPI1, SPI_I2S_FLAG_RXNE) == RESET); //Czekaj na
dane
              temperatura2 = SPI_I2S_ReceiveData(SPI1);               //Odczytaj dane
              if ((temperatura2&0x04)==0){                            //Sprawdz, czy
zakonczono juz pierwszy pomiar po wlaczeniu ukladu TC77
                     temperatura2=0;                                       //Jesli nie,
ustaw temp=0,  wprzeciwnym razie wynik bedzie bledny (>500stC)
              }
              temperatura2 = temperatura2 >> 3;                        //Usun 3 LSB
              SPI1->CR1 &= SPI_Mode_Slave_Mask;                       //Ustaw tryb slave -
wymusi to zmiane stanu NSS na wysoki
              /*Tu wstaw kod konwertuj¹cy wynik i prezentuj¹cy go np. na wyœwietlaczu*/

              sprintf(temperatura, "%d", temperatura2);
              temperatura[3] = temperatura[2];
              temperatura[2] = '.';
              temperatura[4] = '0';
              for (i=0;i<100000ul;i++);
              for (i=0; i< 5; i++)                                 // Wypisz tekst

              LCD_WriteData(temperatura[i]);
              //for (i=0;i<1000000ul;i++);
              for (i=0;i<4500000ul;i++); // opóŸnienie
              LCD_WriteCommand(0x01);
       };
       while (1);

  return 0;
}
```

Zadanie 3

Treść:

Napisz program wyświetlający temperaturę na wyświetlaczu LCD korzystając z czujnika wbudowanego w mikrokontroler i czujnika STLM20. Wyprowadzenie czujnika znajdziesz na złączu Con7 (Temp). Do przetwarzania A/C wykorzystaj tryb wielokanałowy przetwornika. Zdefiniuj symbol stopnia „°” na wyświetlaczu LCD.

Kod programu:

```
void GPIO_Config(void);
void RCC_Config(void);
void NVIC_Config(void);
void ADC_Config(void);
void write_temp(int temp);
int adc_read(int channel);

int main(void) {
	volatile unsigned long int i;
	volatile char tekst1[10] = {"pomiar z: "};
	volatile char tekst2[8] = {"stlm20: "};
	volatile int temp = 0;
	unsigned long int wartoscADC1 = 0;
	//konfiguracja systemu
	RCC_Config();
	GPIO_Config();
	ADC_Config();
	LCD_Initialize(); // Inicjalizuj
	LCD LCD_WriteCommand(LCD_CLEAR); // Wyczysc LCD

	while (1) {
		//odczyt wewnętrznego czujnika
		wartoscADC1 = adc_read(ADC_Channel_16);
		temp = ((1430-(wartoscADC1*0.8067))/4.3 + 25)*10;
		LCD_WriteCommand( LCD_DDRAM_SET | 0x00 );
		for (i=0; i<10; i++)
		LCD_WriteData(tekst1[i]);

		LCD_WriteCommand( LCD_DDRAM_SET | 0x08 );
		write_temp(temp);
		//odczyt kanalu 0 gpio 0
		wartoscADC1 = adc_read(ADC_Channel_0);
		LCD_WriteCommand( LCD_DDRAM_SET | 0x28 );
		for (i=0; i<8; i++)
		LCD_WriteData(tekst2[i]);
		temp = 85.543*(1.8663 - (wartoscADC1*0.8067/1000))*100;
		write_temp(temp);
		for (i=0;i ;i<3200000ul; i++);
	};
	while (1);
	return 0;
}


void write_temp(int temp) {
	LCD_WriteData(48 + ((temp/1000)%10));
	LCD_WriteData(48 + (temp/100) % 10);
	LCD_WriteData(',');
	LCD_WriteData(48 + (temp/10)%10);
	LCD_WriteData(48 + temp%10);
	LCD_WriteData(0xdf); LCD_WriteData('C');
}

void RCC_Config(void)
//konfigurowanie sygnalow taktujacych
{
	ErrorStatus HSEStartUpStatus; //zmienna opisujaca rezultat uruchomienia
	HSE RCC_DeInit(); //Reset ustawien RCC
	RCC_HSEConfig(RCC_HSE_ON); //Wlaczenie HSE
	HSEStartUpStatus = RCC_WaitForHSEStartUp(); //Odczekaj az HSE bedzie gotowy
	if(HSEStartUpStatus == SUCCESS) {
		FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
		FLASH_SetLatency(FLASH_Latency_2); //ustaw zwloke dla pamieci Flash; zaleznie
		od taktowania rdzenia //0:48MHz
```

```
		RCC_HCLKConfig(RCC_SYSCLK_Div1); //ustaw HCLK=SYSCLK
		RCC_PCLK2Config(RCC_HCLK_Div1); //ustaw PCLK2=HCLK
		RCC_PCLK1Config(RCC_HCLK_Div2); //ustaw PCLK1=HCLK/2
		RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9); //ustaw PLLCLK = HSE*9
		czyli 8MHz * 9 = 72 MHz
		RCC_PLLCmd(ENABLE); //wlacz PLL
		while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET); //odczekaj na poprawne
		uruchomienie PLL
		RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK); //ustaw PLL jako zrodlo sygnalu
		zegarowego
		while(RCC_GetSYSCLKSource() != 0x08); //odczekaj az PLL bedzie sygnalem
		zegarowym systemu
		RCC_ADCCLKConfig(RCC_PCLK2_Div6); // ADCCLK = PCLK2/6 = 72 MHz /6 = 12 MHz
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//wlacz taktowanie portu
		GPIO A
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);//wlacz taktowanie portu
		GPIO B
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE));//wlacz taktowanie portu
		AFIO
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE); //wlacz taktowanie ADC1
	} else { }
}

void GPIO_Config(void) {
	//konfigurowanie portow GPIO
	GPIO_InitTypeDef GPIO_InitStructure;
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
	GPIO_Init(GPIOA, &GPIO_InitStructure); // disable JTAG
	GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
}
int adc_read(int channel)
{
	ADC_RegularChannelConfig(ADC1, channel, 1, ADC_SampleTime_239Cycles5);
	ADC_SoftwareStartConvCmd(ADC1, ENABLE);
	while (!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
	return ADC_GetConversionValue(ADC1);
}

void ADC_Config(void) {
	//konfigurowanie przetwornika AC
	ADC_InitTypeDef ADC_InitStructure;
	ADC_InitStructure.ADC_Mode = ADC_Mode_Independent; //Jeden przetwornik, praca
niezalezna
	ADC_InitStructure.ADC_ScanConvMode = DISABLE; //Pomiar jednego kanalu, skanowanie
kanalow nie potrzebne
	ADC_InitStructure.ADC_ContinuousConvMode = DISABLE; //Pomiar w trybie jednokrotnym
	ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None; //Brak
wyzwalania zewnetrznego
	ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right; //Wyrownanie danych do prawej
- 12 mlodszych bitow znaczacych
	ADC_InitStructure.ADC_NbrOfChannel = 1; //Liczba uzywanych kanalow =1
	ADC_Init(ADC1, &ADC_InitStructure); //Incjalizacja przetwornika
	ADC_RegularChannelConfig(ADC1, ADC_Channel_16, 1, ADC_SampleTime_239Cycles5);
	//Kanal 16 - czujnik temperatury procesor
	ADC_Cmd(ADC1, ENABLE); //Wlacz ADC1
	ADC_ResetCalibration(ADC1); //Reset rejestrow kalibracyjnych ADC1
	while(ADC_GetResetCalibrationStatus(ADC1)); //Odczekanie na wykonanie resetu
	ADC_StartCalibration(ADC1); //Kalibracja ADC1
	while(ADC_GetCalibrationStatus(ADC1)); //Odczekanie na zakonczenie kalibracji ADC1
}
```

Zadanie 4

Treść:
Napisz program porównujący wskazania czujnika STLM20 dla 2 metod obliczania wartości temperatury (liniowej i nieliniowej).

Kod programu:

```
void GPIO_Config(void);
void RCC_Config(void);
void NVIC_Config(void);
void ADC_Config(void);
void write_temp(int temp);
int adc_read(int channel);
int main(void) {
	volatile unsigned long int i;
	volatile char tekst1[8] = {"linia: "};
	volatile char tekst2[10] = {"nielinia: "};
	volatile int temp = 0;
	unsigned long int wartoscADC1 = 0; //konfiguracja systemu
	RCC_Config();
	GPIO_Config();
	ADC_Config();
	LCD_Initialize(); // Inicjalizuj LCD
	LCD_WriteCommand(LCD_CLEAR); // Wyczysc LCD
	while (1) {
		//odcyzt kanalu 0, gpio 0
		wartoscADC1 = adc_read(ADC_Channel_0);
		temp = 85.543*(1.8663 - (wartoscADC1*0.8067/1000))*100;
		LCD_WriteCommand( LCD_DDRAM_SET | 0x00 );
		for (i=0; i<8 ;i++)
		LCD_WriteData(tekst1[i]);
		LCD_WriteCommand( LCD_DDRAM_SET | 0x08 );
		write_temp(temp);
		temp = (-1481.96 + sqrt( 2196200 + ( ((1.8663 - (wartoscADC1*0.8067/1000))
	)/0.00000388 ) ))*100;
		LCD_WriteCommand( LCD_DDRAM_SET | 0x28 );
		for (i=0; i<10 ;i++)
		LCD_WriteData(tekst2[i]);
		write_temp(temp);
		for (i=0; i< 3200000ul;i++)
	};
	while (1);
	return 0;
}

void write_temp(int temp) {
	LCD_WriteData(48 + ((temp/1000)%10));
	LCD_WriteData(48 + (temp/100) % 10);
	LCD_WriteData(',');
	LCD_WriteData(48 + (temp/10)%10);
	LCD_WriteData(48 + temp%10);
	LCD_WriteData(0xdf); LCD_WriteData('C');
}

void RCC_Config(void)
//konfigurowanie sygnalow taktujacych
{
	ErrorStatus HSEStartUpStatus; //zmienna opisujaca rezultat uruchomienia
	HSE RCC_DeInit(); //Reset ustawien RCC
	RCC_HSEConfig(RCC_HSE_ON); //Wlaczenie HSE
	HSEStartUpStatus = RCC_WaitForHSEStartUp(); //Odczekaj az HSE bedzie gotowy
	if(HSEStartUpStatus == SUCCESS) {
```

```
		FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
		FLASH_SetLatency(FLASH_Latency_2); //ustaw zwloke dla pamieci Flash; zaleznie
		od taktowania rdzenia //0:48MHz
		RCC_HCLKConfig(RCC_SYSCLK_Div1); //ustaw HCLK=SYSCLK
		RCC_PCLK2Config(RCC_HCLK_Div1); //ustaw PCLK2=HCLK
		RCC_PCLK1Config(RCC_HCLK_Div2); //ustaw PCLK1=HCLK/2
		RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9); //ustaw PLLCLK = HSE*9
		czyli 8MHz * 9 = 72 MHz
		RCC_PLLCmd(ENABLE); //wlacz PLL
		while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET); //odczekaj na poprawne
		uruchomienie PLL
		RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK); //ustaw PLL jako zrodlo sygnalu
		zegarowego
		while(RCC_GetSYSCLKSource() != 0x08); //odczekaj az PLL bedzie sygnalem
		zegarowym systemu
		RCC_ADCCLKConfig(RCC_PCLK2_Div6); // ADCCLK = PCLK2/6 = 72 MHz /6 = 12 MHz
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//wlacz taktowanie portu
		GPIO A
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);//wlacz taktowanie portu
		GPIO B
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE));//wlacz taktowanie portu
		AFIO
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE); //wlacz taktowanie ADC1
	} else { }
}

void GPIO_Config(void) {
	//konfigurowanie portow GPIO
	GPIO_InitTypeDef GPIO_InitStructure;
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
	GPIO_Init(GPIOA, &GPIO_InitStructure); // disable JTAG
	GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
}

int adc_read(int channel) {
	ADC_RegularChannelConfig(ADC1, channel, 1, ADC_SampleTime_239Cycles5);
	ADC_SoftwareStartConvCmd(ADC1, ENABLE);
	while (!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
	return ADC_GetConversionValue(ADC1);
}

void ADC_Config(void) {
	//konfigurowanie przetwornika AC
	ADC_InitTypeDef ADC_InitStructure;
	ADC_InitStructure.ADC_Mode = ADC_Mode_Independent; //Jeden przetwornik, praca
niezalezna
	ADC_InitStructure.ADC_ScanConvMode = DISABLE; //Pomiar jednego kanalu, skanowanie
kanalow nie potrzebne
	ADC_InitStructure.ADC_ContinuousConvMode = DISABLE; //Pomiar w trybie jednokrotnym
	ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None; //Brak
wyzwalania zewnetrznego
	ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right; //Wyrownanie danych do prawej
- 12 mlodszych bitow znaczacych
	ADC_InitStructure.ADC_NbrOfChannel = 1; //Liczba uzywanych kanalow =1
	ADC_Init(ADC1, &ADC_InitStructure); //Incjalizacja przetwornika
	ADC_RegularChannelConfig(ADC1, ADC_Channel_16, 1, ADC_SampleTime_239Cycles5);
	//Kanal 16 - czujnik temperatury procesor
	ADC_Cmd(ADC1, ENABLE); //Wlacz ADC1
	ADC_ResetCalibration(ADC1); //Reset rejestrow kalibracyjnych ADC1
	while(ADC_GetResetCalibrationStatus(ADC1)); //Odczekanie na wykonanie resetu
	ADC_StartCalibration(ADC1); //Kalibracja ADC1
	while(ADC_GetCalibrationStatus(ADC1)); //Odczekanie na zakonczenie kalibracji ADC1
}
```