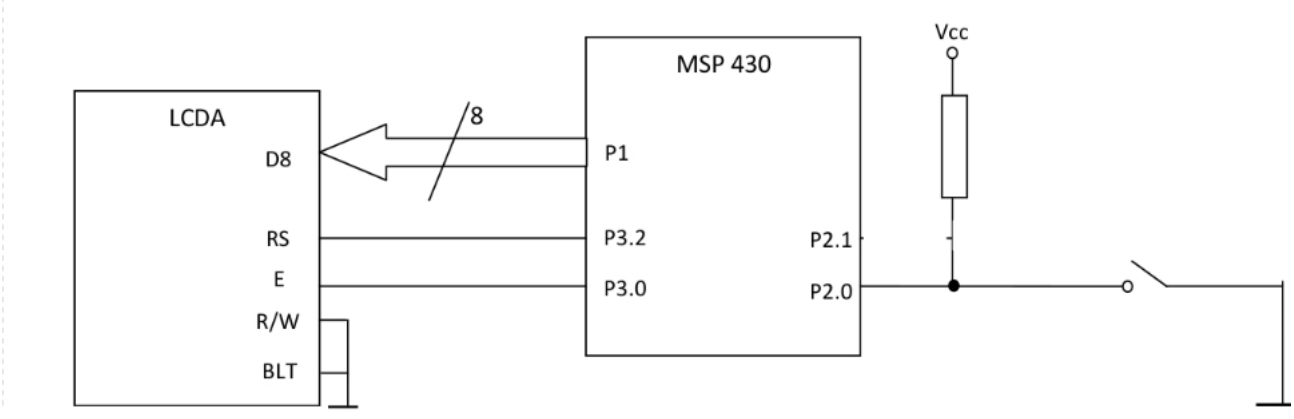
LABORATORIUM TM 6

Sobolewski Konrad , Boczek Bartłomiej, Alek Piotrowski

1. Cel laboratorium

Zadanie polegało na zaprojektowaniu układ do pomiaru temperatury z wyświetlaczem LCD korzystający z wbudowanego czujnika mikrokontrolera MSP430. Wartość maksymalna i minimalna jest zapisywana w pamięci nieulotnej i wyświetlana na żądanie. Zrealizowana została także obsługa DMA

2. Schemat układu



3. Kod

```
#include "msp430x16x.h"
#include "string.h"
#include <cstdio>
#include <stdint.h>
#include <limits.h>

// sygnaly sterujace LCD
#define CTRL_E            0x01  // clear
#define CTRL_RS           0x04  // entry mode

// komendy sterujace LCD
#define LCD_CLEAR         0x01 // clear

inline void strobe_e() {
    P3OUT |= CTRL_E;
    P3OUT &= ~CTRL_E;
}

inline void display_string(char *str) {
    P3OUT = 0x00;
    strobe_e();
    __delay_cycles(1500);

    char c;
    while ((c = *(str++))) {   //wywietlanie znaku po znaku
        P1OUT = c;
        P3OUT = CTRL_RS;
        strobe_e();
        __delay_cycles(100);
    }
}

inline init() {
    // Wyjscie danych na LCD
    P1DIR = 0xFF;
    P1OUT = 0x00;

    // Wyjscie sterujace LCD
    // pin 1 = wyjscie strobujace E
    // pin 2 = wyjscie RS
    P3DIR = 0xFF;
    P3OUT = 0x00;

    // Uruchomienie wyswietlacza HITACHI 44780
    __delay_cycles(2000);     // Duzy delay, zeby LCD wstal poprawnie po resecie
    P1OUT = 0x38;               // Function set ,linia 1 , 8 bitowo
    strobe_e();
    P1OUT = 0x0C;               // Display on/off control
    strobe_e();
    P1OUT = 0x06;               // Entry mode set (increments DDRAM address)
    strobe_e();
    __delay_cycles(1000);
    P1OUT = LCD_CLEAR;
    strobe_e();
    __delay_cycles(1000);
    display_string("Init");
}

volatile int value;  //aktualna temperatura
volatile int show = 0; //pokaz min/max

int counter_odszum = 0;
unsigned short odszum = 0;
int dmacounter=0;

#pragma DATA_SECTION( flashMin, ".infoA" );
int flashMin = 100;

```

```
#pragma DATA_SECTION( flashMax, ".infoA" );
int flashMax = 0;

int samples[8] = { 0 };
int samples2[8] = { 0 };
int* whichTable = samples;

inline void adcInit() {
    ADC12CTL0 = ( MSC | SHT0_4 | ADC12ON | REFON);
    //Multiple sample and conversion | hold | wlacz | generator wlacz
    ADC12CTL1 = ( CONSEQ_0 | ADC12DIV_0 | ADC12SSEL_1 | SHP | SHS_0);
    //Single-channel, single-conversion | bez dzielnika | ACLK | ADC12SC wyzwala |
    ADC12MCTL0 = ( INCH_10 | SREF_1 | EOS);
    // sensor temp | przedzial napiec | end of sequence
    // ADC12IE= 0x0001; //adc12ie0
}

inline void timerAInit() {

    TACTL = MC_1 | ID_0 | TASSEL_1; //mode up ,bez dzielnika,init clock;
    BCSCTL1 &= ~ XTS; // wolny tryb ACLK

    TACCTL0 |= CCIE; //wlacz przerwania zegara
    TACCR0 = 4096; //zliczaj 1 sekunde
}

inline void initButton() {
    P2DIR = 0x00; // kieruenk wejsciowy
    P2IE |= BIT0;  // wlacz przerwania dla zmiany kierunku...
    P2IES |= BIT0;  // .. po malejcym zboczu
    P2IFG &= ~BIT0;  //  wyczysc rejest flag
}

inline void flash_write(int min, int max) {
    _DINT();                            // wylacz przerwania
    WDTCTL = WDTPW | WDTHOLD;
    // erase segment
    while (BUSY & FCTL3)
        ;                   // Check if Flash being used
    FCTL2 = FWKEY + FSSEL_1 + FN3;       // Clk = SMCLK/4
    FCTL1 = FWKEY + ERASE;               // ERASE
    FCTL3 = FWKEY;                       // Clear Lock bit
    flashMin = 0;                          // kasujemy od tego adresu
    while (BUSY & FCTL3) ;//czekamy az skasuje
    FCTL1 = FWKEY;
    FCTL2 = FWKEY + FSSEL_1 + FN0;       // Clk = SMCLK/4
    FCTL3 = FWKEY;                       // Clear Lock bit
    FCTL1 = FWKEY + WRT;                 // Set WRT bit for write operation

    flashMin = min;
    flashMax = max;    // copy value to flash

    FCTL1 = FWKEY;                        // Clear WRT bit
    FCTL3 = FWKEY + LOCK;                 // Set LOCK bit
    _EINT();                 // wlacz przerwania
}

inline void dmaInit() {
    DMACTL0 = DMA0TSEL_6; // ADC12IFG
    DMACTL1 = 0;
    DMA0CTL = DMADT_4 | DMADSTINCR_3 | DMAEN | DMAIE | DMASRCINCR_0;
//    // repeated single transfer | dest adres increment | wlacz | przerwania | source staly
    DMA0SA = &ADC12MEM0; // source
    DMA0DA = &samples; //destination
    DMA0SZ = 8; // word na transfer ( int )
}

void main(void) {
    WDTCTL = WDTPW | WDTHOLD | WDTCNTCL | WDTSSEL; //stop..bow..boww
    char str[17] = { 0 }; //tab na znaki
    int min = flashMin, max = flashMax;
    init();
    adcInit();
    initButton();
    timerAInit();
    dmaInit();

    if (IFG1 & WDTIFG) // jesli reset to obsluz
    {
        P1OUT = LCD_CLEAR;
        strobe_e();
        __delay_cycles(500); //wyświetlacz wymaga delay zeby wszystko poprawnie wyswietlić
        display_string("RESET BY WDT");
        __delay_cycles(60000);
        IFG1 &= ~WDTIFG; //zerujemy flage
    }
    _enable_interrupt();

    while (1) {
        _BIS_SR(LPM3_bits + GIE);
        WDTCTL = WDTPW + WDTCNTCL; //"poglaskanie" watchodga ->odnoowienie jego licznika
        if (show) {
            sprintf(str, "%d%cC,%d%cC,%d%cC", value, (uint8_t) 223, min,
                    (uint8_t) 223, max, (uint8_t) 223); //zapis wartości uint do char* str
        } else {
            sprintf(str, "%d%cC", value, (uint8_t) 223);
        }
        WDTCTL = WDTPW + WDTHOLD; //wstrzymanie odliczania watchdoga przed delay
        P1OUT = LCD_CLEAR;
        strobe_e();
        __delay_cycles(500);
        display_string(str);
        WDTCTL = WDTPW + WDTCNTCL; //"poglaskanie" watchodga ->odnoowienie jego licznika

        //aktuaizacja najwiekszej/najmnijeszej wartosci
        if (value > max) {
            max = value;
            flash_write(min, max);
        }
        if (value < min) {
            min = value;
```

```
180             flash_write(min, max);
181         }
182     }
183 }
184
185 #pragma vector=DACDMA_VECTOR
186 __interrupt void dmadac_ISR() {
187     WDTCTL = WDTPW + WDTCNTCL;
188     if (DMA0CTL & DMAIFG) {
189         //_DINT();
190         DMA0DA = (whichTable == samples) ? samples2 : samples;  //podwojne buforowanie , zmieniamy tablice za kazdym
    razem
191         //_EINT();
192         int i = 0;
193         uint32_t tmp = 0;
194         int tmp2;
195         for (i = 0; i < 8; i++) {
196             tmp += whichTable[i];  // suma probek
197         }
198         whichTable = (whichTable == samples) ? samples2 : samples; //wybieramy bufor z próbkami
199
200         tmp = tmp / 8; //srednia
201         tmp = (2000 * tmp - 5383560) / 19383; //przeskalowanie na temperature
202         tmp2 = (int) tmp;  //do inta dla pewnosci i konwersji typów
203         if (tmp2 != value) {  // jesli zmienił sie pomiar
204                 LPM3_EXIT;// pobudka
205                 value = tmp2; //nadpisz
206         }
207         DMA0CTL &= ~DMAIFG;  //wyczysc flage przerwania
208
209     }
210 }
211
212
213 const int STALA_ODSZUM = 3;
214 #pragma vector=PORT2_VECTOR
215 __interrupt void Port2(void) {
216     WDTCTL = WDTPW + WDTCNTCL;
217     if (!odszum) { //odszum-licznik który dekrementujaemy STALA_ODSZUM razy
218         odszum = STALA_ODSZUM;
219         counter_odszum = 1;
220         P2IFG &= ~BIT0;
221     }
222 }
223
224 // przerwanie zegara - samowylaczajace
225 #pragma vector=TIMERA0_VECTOR
226 __interrupt void TimerA(voia) {
227     ADC12CTL0 |= ( ENC | ADC12SC); //Enable | start conversion
228     WDTCTL = WDTPW + WDTCNTCL;
229     if (odszum) { //strategia większości głosów przy odszumianiu
230         if (!(P2IN & 0x01)) //inkrementuj tylko jezeli przycisk jest wcisniety
231             ++counter_odszum; //inkrementacja licznika do odszumiania
232         else
233             --counter_odszum; //dekrementacja
234
235         if (--odszum == 0) { //koniec odszumiania
236             if (counter_odszum > 0) { //wygrały głosy na to że przysik był więcej razy wciśnięty niż puszczony
237                 LPM3_EXIT;
238                 show = !show; //pokaż/ukryj max min temperaturę
239             }
240         }
241     }
242
243 }
```