Univerzitet u Beogradu – Elektrotehnički fakultet

Projektni izveštaj

iz predmeta Namenski računarski sistemi

Danilo Ristić
0182/2019

Projektni zadatak 10:

Napisati program koji, putem UARTa, omogućava kontrolu osvetljaja određene LE diode. Za svaku diodu je moguće podesiti intenzitet osvetljaja (u opsegu od 0 - 99). Program na kontroleru, putem UARTa, dobija odgovarajuće kontrolne poruke koje sadrže identifikator o diodi kao i njen intenzitet osvetljaja. Nakon primljene kontrolne poruke intenzitet osvetljaja LE diode se ispisuje na LE displeju.

Projekat je izveden na platformi MSP430F5529, koristeći softversko razvojno okruženje Code Composer Studio.

U okviru projekta, izvršava se podešavanje intenziteta osvetljaja LE diode u okviru porta 2 razvojne platforme: dioda 1 - *P2.4* i dioda 2 - *P2.5*. Putem UART-a, korisnik šalje poruke sistemu. Poruke se primaju i izvršavaju se funkcije u slučaju odgovarajućih poruka. U slučaju poruka *a* i *b*, selektuje se koja dioda je aktivna dioda, redom, dioda 1 i dioda 2. Aktivna dioda predstavlja diodu čija vrednost intenziteta (vrednost između 0 i 99) se trenutno prikazuje na sedmo-segmentnom displeju i čija vrednost može da se menja porukama preko UART-a. Poruke u vidu broja u vrednosti između 0 i 99, menjaju intenzitet aktivne diode na vrednost poruke u procentima od unapred zadatog perioda pulse-width modulacije. Za vrednost perioda pulse-width modulacije, izabrana je vrednost od oko 24.4ms, kako bi se uočila razlika pri menjanju procenata intenziteta osvetljaja diode. Za multipleksiranje sedmo-segmentnog displeja korišćen je *TIMER A1* sa periodom od oko 2ms. Pulse-width modulacija dioda se izvršava pomoću tajmera *TIMER A2*. Za konfiguraciju UART-a korišćen je baud rate od 115200bps.

Rešenje projekta se sastoji iz nekoliko funkcija. Pre definisanja funkcija, definisani su makroi, promenljive i konstante potrebne za rad. Glavna (main) funkcija služi za inicijalizaciju svih potrebnih tajmera, displeja i konfiguraciju UART-a. Pored toga u glavnoj funkciji se omogućava rad prekidnih rutina. Nakon toga putem beskonačne petlje *while (1);* se omogućava da se program dalje izvršava koliko je potrebno i daje korisniku na rad. Nakon toga, definisane su prekidne rutine koje omogućavaju glavne funkcionalnosti programa, tj. obradu poruke preko UART-a i menjanje osvetljaja diode (*UARTISR*), i multipleksiranje sedmo-segmentnog displeja (*CCR0ISR*).

Prekidna rutina *UARTISR* se poziva pri aktivaciji `USCI_A1_VECTOR` i ispituje da li je primljena poruka preko UART-a. Ukoliko jeste proverava se koja je poruka i izvršava jedna od potrebnih funkcionalnosti. U slučaju broja od 0 do 99 menja se intenzitet osvetljaja aktivne diode. Prekidna rutina *CCR0ISR* se poziva pri aktivaciji `TIMER_A1_VECTOR` i redom ispisuje potrebnu cifru vrednosti intenziteta osvetljaja na trenutno aktivan sedmo-segmentni displej. Pri završetku ispisa menja se aktivni displej.

Ispisan program koji izvršava datu funkcionalnost iz projekta:

```
/**
 * @file  main.c
 * @brief Projekat iz predmeta NSS, septembar 2023.
 *
 * Napisati program koji putem UARTa omogucava kontrolu osvetljaja odredjene LE
diode.
```

```
 * Za svaku diodu je moguce podesiti intenzitet osvetljaja (u opsegu od 0 do 99).
 * Program na kontroleru, putem UARTa, dobija odgovarajuce kontrolne poruke koje
sadrze identi kator o diodi kao i njen intenzitet osvetljaja.
 * Nakon primljene kontrolne poruke intenzitet osvetljaja LE diode se ispisuje na
LE displeju.
 *
 * @date   septembar 2023.
 * @author Danilo Ristic 0182/2019 (rd190182d@student.etf.bg.ac.rs)
 *
 */

#include <msp430.h>
#include <stdint.h>
#include "ETF_5529_HAL/hal_7seg.h"

#define ASCII2DIGIT(x)      (x - '0') // Makro za konverziju ASCII vrednosti u
broj
#define DIGIT2ASCII(x)      (x + '0') // Makro za konverziju broja u ASCII
vrednost

#define PWM_PERIOD      800 // PWM period ~24.4ms
#define PWM_STEP 8 // Korak za PW -> 1% od 24.4ms

#define DISPLAY_REFRESH_PERIOD        (63)  // ~2ms jer tajmer koristi ACLK
(32768Hz)

volatile active_display_t   activeDisplay; // Koristi se za promenu aktivnog
displeja.
volatile uint8_t data; // Primljena poruka preko UARTa
volatile uint8_t active_diode = 0; // Vrednost posmatrane LE diode (0 - LE dioda
1, 1 - LE dioda 2)
volatile uint8_t intensity_diode_1 = 0; // Trenutna vrednost intenziteta
osvetljaja LE diode 1
volatile uint8_t intensity_diode_2 = 0; // Trenutna vrednost intenziteta
osvetljaja LE diode 2

/**
 * funkcije za odredjivanje vise i nize cifre primljene poruke
 */

uint8_t getLowDigit(uint16_t number){
    return (number - 10*(number/10));
}
uint8_t getHighDigit(uint16_t number){
    return (number/10);
```

```
}

typedef enum{
    DISP1,
    DISP2
}active_display;

active_display activeDisplay; // sedmo-segmentni displej na kome trenutno
ispisujemo cifru

/**
 * @brief Main funkcija
 *
 * Na osnovu poruka sa UARTa odredjuje se osvetljaj LE dioda
 */
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;   // Zaustavljanje watchdog tajmera

    // Inicijalizacija sedmo-segmentnog displeja
    HAL_7Seg_Init();
    activeDisplay = DISP1;

    // Inicijalizacija PW tajmera LE diode 1 i 2
    P2DIR |= BIT4;              // P2.4 out
    P2SEL |= BIT4;              // P2.4 TA
    P2DIR |= BIT5;              // P2.5 out
    P2SEL |= BIT5;              // P2.5 TA
    TA2CCTL1 = OUTMOD_7;        // reset/set outmode
    TA2CCTL2 = OUTMOD_7;        // reset/set outmode
    TA2CCR0 = PWM_PERIOD;       // perioda
    TA2CCR1 = PWM_STEP * intensity_diode_1;   // pocetna PW vrednost za LE diodu
1
    TA2CCR2 = PWM_STEP * intensity_diode_2;   // pocetna PW vrednost za LE diodu
2
    TA2CTL = TASSEL__ACLK | MC__UP; // ACLK izvor, UP mode

    // aktiviranje tajmera A1 za multipleksiranje sedmo-segmentnog displeja
    TA1CCR0     = DISPLAY_REFRESH_PERIOD; // perioda displeja
    TA1CCTL0    = CCIE;
    TA1CTL      = TASSEL__ACLK | MC__UP; // ACLK izvor, UP mode

    // konfiguracija UART-a
    P4SEL |= BIT4 | BIT5;       // postavljanje P4.4 and P4.5 za UART
    UCA1CTL1 |= UCSWRST;        // ulazenje u sw reset
```

```
    UCA1CTL0 = 0;
    UCA1CTL1 |= UCSSEL__ACLK ;
    UCA1BRW = 8;
    UCA1MCTL = UCBRS_6;         // postavljanje na 115200 bps
    UCA1CTL1 &= ~UCSWRST;       // izlazenje iz sw reset
    UCA1IE |= UCRXIE;           // omogucavanje RX interrupt

    // postavljanje data na pocetnu vrednost
    data = '0';
    HAL_7Seg_WriteDigit(ASCII2DIGIT(data));

    __enable_interrupt();      // omogucavanje interrupt-a

    while (1); // Program se konstantno izvrsava, kako se sve funkcionalnosti
izvrsavaju preko interrupt-a
}

/**
 * Prijem i obrada poruke sa UARTA
 */
void __attribute__ ((interrupt(USCI_A1_VECTOR))) UARTISR (void)
{
    switch (UCA1IV)
    {
    case 2:                 // RXIFG
        data = ASCII2DIGIT(UCA1RXBUF);  // prijem poruke
        if (data + '0' == 'a') {
            active_diode = 0;  // ako je poruka slovo 'a', posmatra se LE dioda 1
        } else if (data + '0' == 'b') {
            active_diode = 1;  // ako je poruka slovo 'b', posmatra se LE dioda 2
        } else if (data >= 0 && data <= 99) { // ako je poruka intenzitet
osvetljaja diode, postavlja se ta vrednost
            if (active_diode == 0) {
                intensity_diode_1 = data;
                TA2CCR1 = PWM_STEP * intensity_diode_1; // menja se vrednost
osvetljaja LE diode 1
            } else {
                intensity_diode_2 = data;
                TA2CCR2 = PWM_STEP * intensity_diode_2; // menja se vrednost
osvetljaja LE diode 1
            }
        }
        break;
    default:
        break;
```

```
    }
}


/**
 * Multipleksiranje sedmo-segmentnog displeja
 */
void __attribute__ ((interrupt(TIMER1_A0_VECTOR))) CCR0ISR (void)
{
    switch (activeDisplay)
    {
    case DISP1:
        HAL_7SEG_DISPLAY_1_ON;
        HAL_7SEG_DISPLAY_2_OFF;
        if (active_diode == 0) {
            HAL_7Seg_WriteHexDigit(getHighDigit(intensity_diode_1); // Ako je
aktivirana LE dioda 1, pise se visa cifra njenog osvetljaja
        } else {
            HAL_7Seg_WriteHexDigit(getHighDigit(intensity_diode_2); // Ako je
aktivirana LE dioda 2, pise se visa cifra njenog osvetljaja
        }
        activeDisplay = DISP2;
        break;
    case DISP2:
        HAL_7SEG_DISPLAY_1_OFF;
        HAL_7SEG_DISPLAY_2_ON;
        if (active_diode == 0) {
            HAL_7Seg_WriteHexDigit(getLowDigit(intensity_diode_1);  // Ako je
aktivirana LE dioda 1, pise se niza cifra njenog osvetljaja
        } else {
            HAL_7Seg_WriteHexDigit(getLowDigit(intensity_diode_2);  // Ako je
aktivirana LE dioda 2, pise se niza cifra njenog osvetljaja
        }
        activeDisplay = DISP1;
        break;
    }
}
```