Sprawozdanie z laboratorium

Architektury Komputerów

Ćwiczenie numer: 4

Temat: Timer

Wykonujący ćwiczenie:

- Zaborowska Magda
- Wójtowicz Patryk

Studia dzienne I stopnia

Kierunek: Informatyka

Semestr: III Grupa zajęciowa: Lab 15

Prowadzący ćwiczenie:
Dr inż. Mirosław
Omieljanowicz

.....

OCENA

Data wykonania ćwiczenia

15.11.2021r.

.....

Data i podpis prowadzącego

Cel zadania

Stworzenie stopera obsługiwanego przez cztery przyciski.

- Przycisk 1 włącza odliczanie czasu,
- Przycisk 2 zatrzymuje stoper,
- Przycisk 3 resetuje stoper, można go użyć wyłącznie, gdy stoper nie jest zatrzymany,
- Przycisk 4 zapisuje w górnej linii międzyczas.

Informacje potrzebne do wykonania zadania

W układzie MSP430 można wybrać jeden z trzech zegarów:

- DCO – wewnętrzny zegar taktujący,
- XT2CLK – zasilany standardowym kwarcem lub rezonatorami w zakresie 450Hz – 8MHz,
- LFXT1CLK - może on być użyty z niskoczęstotliwościowym kwarcem 32678Hz, bądź z rezonatorami z zakresu 450kHz- 8MHz (tryb pracy wysokoczęstotliwościowej)

Zegary sterujące dla poszczególnych komponentów mikrokontrolera MSP430:

Zegar sterujący	Zegar źródłowy	Użycie
ACLK	LFXT1CLK	Jako zegar pomocniczy
MCLK	LFXT1CLK XT2CLK DCO	Używany przez system oraz CPU
SMCLK	LFXT1CLK XT2CLK DCO	Współpracuje z urządzeniami peryferyjnymi

Używając płytki EasyWeb należy ustawić taktowanie z rezonatora kwarcowego. Aby to zrobić, źródło zegara podstawowego Basic Clock należy ustawić na taktowanie LFXT1CLK. Jeżeli tego nie wykonamy zegar procesora jest automatycznie ustawiany na DCO, który generuje częstotliwość ok. 800kHz.

Aby ustawić źródło zegara podstawowego należy wykonać następujące czynności:

- Włączyć oscylator.
- Wyczyścić flagę OFIFG.
- Odczekać.
- Sprawdzić stan flagi, dokupi wciąż jest ustawiona, należy powtarzać powyższe kroki.
- Zmienić taktowanie zegarów w rejestrze BCSCTL2

W naszym przypadku napisano poniższy kod:

```
30  // Basic Clock Module ustawiamy na ACLK(zegar 8 MHz ) i dzielimy częstotliwość przez 2 (4 MHz)
31  BCSCTL1 |= XTS;                       // ACLK = LFXT1 = HF XTAL 8MHz
32
33  do
34    {
35    IFG1 &= ~OFIFG;                     // Czyszczenie flgi OSCFault
36    for (i = 0xFF; i > 0; i--);         // odczekanie
37    }
38    while ((IFG1 & OFIFG) == OFIFG);    // dopóki OSCFault jest ciągle ustawiona
39
40  BCSCTL1 |= DIVA_1;                    // ACLK=8 MHz/2=4 MHz
41  BCSCTL2 |= SELM0 | SELM1;             // MCLK= LFTX1 =ACLK
```

Kod 1 Poprawne ustawienie źródła zegara podstawowego

W pracy z timerami istotna jest konfiguracja Timera A. Timer A ma możliwość pracowania w czterech trybach pracy. Wybiera się je w rejestrze TACTL za pomocą znacznika MCx.

Tryb pracy	Znacznik	Funkcja
Stop	MC_0	Zatrzymuje timer
Up	MC_1	Cykliczne zliczanie timera, od zera do wartości zadanej w rejestrze TACCR0.
Continuous	MC_2	Timer cyklicznie liczy od zera do wartości 0xffff
Up/Down	MC_3	Timer cyklicznie inkrementuje od zera do wartości zapisanej w rejestrze TACCR0 , a dekrementuje od wartości w rejestrze TACCR0 do zera

W przypadku, gdy chcemy dodać podzielniki, można je ustawić w tym samym rejestrze za pomocą znacznika IDx:

Znacznik	Podzielnik
ID_0	1
ID_1	2
ID_2	4
ID_3	8

Istotny jest również wybór zegara taktującego. Dostępne są cztery opcje:

Znacznik	Zegra
TASSEL_0	TACLK
TASSEL_1	ACLK
TASSEL_2	SMCLK
TASSEL_3	INCLK

Aby aktywować układ TimerA, należy przyporządkować mu w rejestrze TACTL (Timer_A control) odpowiedni zegar, ustawić podzielniki i włączyć przerwania. W naszym przypadku TimerA ustawiamy na ACLK, tryb pracy Up, 500kHz a przerwania ustawiamy co 100ms.

```
43  // Timer_A  ustawiamy na 500 kHz
44  // a przerwanie generujemy co 100 ms
45  TACTL = TASSEL_1 + MC_1 +ID_3;        // Wybieram ACLK, ACLK/8=500kHz,tryb Up
46  CCTL0 = CCIE;                         // włączenie przerwan od CCR0
47  CCR0=5000;                           // podzielnik 50000: przerwanie generowane co 100 ms
```

Kod 2 Konfiguracja zegara Timera A

Potrzebna jest również procedura obsługi przerwań od Timer A.

```
152  // procedura obsługi przerwania od TimerA
153
154  #pragma vector=TIMERA0_VECTOR
155  __interrupt void Timer_A (void)
156      {
157      licznik+=1; //do zliczania milisekund *100
158      _BIC_SR_IRQ(LPM3_bits);              // wyjście z trybu LPM3
159      }
```

Kod 3 Funkcja obsługi przerwań od Timera A

W zadaniu użyto funkcji z bibliotek z zadania LCD, do inicjalizacji LCD(InitPortsLcd(), InitLCD()) czyszczenia wyświetlacza (cleraDispley()), przesyłania znaków na wyświetlacz(SEND_CHAR()) wyboru między pierwszą i drugą linia wyświetlacza (SEND_CMD()).

Realizacja zadania

Poniższy kod przezentuje poprawnie zrealizowane zadanie stopera:

```c
#include <msp430x14x.h>
#include "lcd.h"
#include "portyLcd.h"

//piny P4.4 - P4.7 jako wejścia
#define KL1 BIT4 &P4IN
#define KL2 BIT5 &P4IN
#define KL3 BIT6 &P4IN
#define KL4 BIT7 &P4IN
//---------------- zmienne globalne -------------
unsigned int i = 0;
unsigned int licznik = 0;
unsigned char znaki[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'}; // tablica znaków możliwych do wyświetlenia
unsigned int ilem_j = 0, ilem_d = 0, ile_s = 0, ile_d = 0,
minuty_j = 0, minuty_d = 0, godz_j = 0, godz_d = 0;                         //licznik jedności i dziesiątek w godzinie, minucie sekundzie i milisekundzie
unsigned int ster = 0;

void Clock(void);   // deklaracja funkcji Clock()
void start_s();     // deklaracja funkcji start_s()

//----------------- main program -------------------
void main(void)
{
    P2DIR |= BIT1; // STATUS LED

    WDTCTL = WDTPW + WDTHOLD; // Wyłączenie WDT

    InitPortsLcd(); // inicjalizacja portów LCD
    InitLCD();      // inicjalizacja LCD
    clearDisplay(); // czyszczenie wyświetlacza

    // Basic Clock Module ustawiamy na ACLK(zegar 8 MHz ) i dzielimy częstotliwość przez 2 (4 MHz)
    BCSCTL1 |= XTS; // ACLK = LFXT1 = HF XTAL 8MHz

    do
    {
```

Kod Programu 1

```c
        IFG1 &= ~OFIFG;                 // Czyszczenie flgi OSCFault
        for (i = 0xFF; i > 0; i--);     // odczekanie
    } while ((IFG1 & OFIFG) == OFIFG);  // dopóki OSCFault jest ciągle ustawiona

    BCSCTL1 |= DIVA_1;        // ACLK=8 MHz/2=4 MHz
    BCSCTL2 |= SELM0 | SELM1; // MCLK= LFTX1 =ACLK

    // Timer_A  ustawiamy na 500 kHz
    // a przerwanie generujemy co 100 ms
    TACTL = TASSEL_1 + MC_1 + ID_3; // Wybieram ACLK, ACLK/8=500kHz,tryb Up
    CCTL0 = CCIE;                   // włączenie przerwan od CCR0
    CCR0 = 5000;                    // podzielnik 50000: przerwanie generowane co 100 ms

    _EINT();    // włączenie przerwań
    start_s();  // wywołanie funkcji wyświetlającej poczatkowy zegar
    for (;;)
    {
        _BIS_SR(LPM3_bits); // przejscie do trybu LPM3
        Clock();            // wywołanie zegara
```

Kod Programu 2

```
    }
}

void Clock(void) // funkcja zegara
{
    if ((KL1) == 0)     // jeżeli klawisz pierwszy jest wciśnięty
    {                   // zmienną sterującą ustawiamy na 1
        ster = 1;       // by można było rozpocząć liczenie stopera
    }
    else if ((KL2) == 0)    // jeżeli klawisz drugi jest wciśnięty
    {                       // zmienną sterująca ustawiamy na 0
        ster = 0;           // by można było zatrzymać liczenie
    }
    else if ((KL3) == 0 && ster == 0)   // jeżeli klawisz trzeci jest wciśnięty
    {                                   // oraz zmienna sterująca jest ustawiona na 0,
        ile_s = 0;                      // czyli stoper jest zatrzymany wtedy następuje
        ile_d = 0;                      // wyzerowanie stopera, czyli przwyrócenie go w stan początkowy
        minuty_j = 0;                       // oraz wyświetlenie ponowne samych zer funkca start_s()
        minuty_d = 0;
        godz_j = 0;
        godz_d = 0;
        ster = 0;
        start_s();
    }
    else if ((KL4) == 0 && ster == 1)       // jeśli wciśnięto klaiwsz czwarty oraz stoper nie jest zatrzymany
    {                                       // to w górnej liny wyświetla zapisuje się międzyczas
        SEND_CMD(DD_RAM_ADDR);  // wyświetlanie w pierwszej lini wyświetlacza
        SEND_CHAR(znaki[godz_d % 10]);      // godziny
        SEND_CHAR(znaki[godz_j % 6]);
        SEND_CHAR(':');
        SEND_CHAR(znaki[minuty_d % 10]);    // minuty
        SEND_CHAR(znaki[minuty_j % 6]);
        SEND_CHAR(':');
        SEND_CHAR(znaki[ile_d % 10]);       // sekundy
        SEND_CHAR(znaki[ile_s % 10]);
        SEND_CHAR(':');
```

Kod Programu 3

```
        SEND_CHAR(znaki[ilem_d % 10]);      // milisekundy
        SEND_CHAR(znaki[ilem_j % 10]);
    }
    if (licznik % 10 == 0 && ster == 1) // gdy mineła sekunda (10 * 100 milisekund) i nie zatrzymano timera
    {
        licznik = 0;
        for (int k = 0; k < 10; k++) // dla każdych 10 ms wyświetla odświerzony czas.
        {
            SEND_CMD(DD_RAM_ADDR2);             // wyświetlanie w drugiej lini wyświetlacza
            SEND_CHAR(znaki[godz_d % 10]);      // godziny
            SEND_CHAR(znaki[godz_j % 6]);
            SEND_CHAR(':');
            SEND_CHAR(znaki[minuty_d % 10]);    // minuty
            SEND_CHAR(znaki[minuty_j % 6]);
            SEND_CHAR(':');
            SEND_CHAR(znaki[ile_d % 10]);       // sekundy
            SEND_CHAR(znaki[ile_s % 10]);
            SEND_CHAR(':');
            SEND_CHAR(znaki[ilem_d % 10]);      // milisekundy
            SEND_CHAR(znaki[ilem_j % 10]);
            P2OUT ^= BIT1;                      //zapal diodę
            ilem_j++;                           // zwiększamy jedności milisekund o 1
        }
        if (godz_j >= 10)   // jeżeli godziny w jednościach są większe równe 10 zwiększamy
        {                   // godziny w dziesiątkach i zerujemy godziny w jednościach
            godz_d++;
            godz_j = 0;
        }
        if (minuty_d >= 6)  // jeżeli minuty w dziesiątkach są większe równe 6 zwiększamy
```

Kod Programu 4

```
120         if (minuty_d >= 6)  // jeżeli minuty w dziesiątkach są większe równe 6 zwiększamy
121         {                   // godziny w jednościach i zerujemy minuty w dziesiątkach
122             godz_j++;
123             minuty_d = 0;
124         }
125         if (minuty_j >= 10) // jeżeli minuty w jednościach są większe równe 10 zwiększamy
126         {                   // minuty w dziesiątek i zerujemy minuty w jednościach
127             minuty_d++;
128             minuty_j = 0;
129         }
130         if (ile_d >= 6)     // jeżeli sekundy w dziesiątkach są większe równe 6 zwiększamy
131         {                   // minuty w jednościach i zerujemy sekundy w dziesiątkach
132             minuty_j++;
133             ile_d = 0;
134         }
135         if (ile_s >= 10)    // jeżeli sekundy w jednościach są większe równe 10 zwiększamy
136         {                   // sekundy w dziesiątkach i zerujemy sekundy w jednościach
137             ile_d++;
138             ile_s = 0;
139         }
140         if (ilem_d >= 10)   // jeżeli milisekundy w dziesiątkach są większe równe 10 zwiększamy
141         {                   // sekundy w jednościach i zerujemy milisekundy w dziesiątkach
142             ile_s++;
143             ilem_d = 0;
144         }
145         if (ilem_j >= 10)   // jeżeli milisekundy w jednościach są większe równe 10 zwiększamy
146         {                   // milisekundy w dziesiątkach i zerujemy milisekundy w jednościach
147             ilem_d++;
148             ilem_j = 0;
149         }
150         SEND_CMD(CUR_SHIFT_LEFT); // powrót kursorem na początek
151     }
152 }
153
154 // procedura obsługi przerwania od TimerA
155 #pragma vector = TIMERA0_VECTOR
```

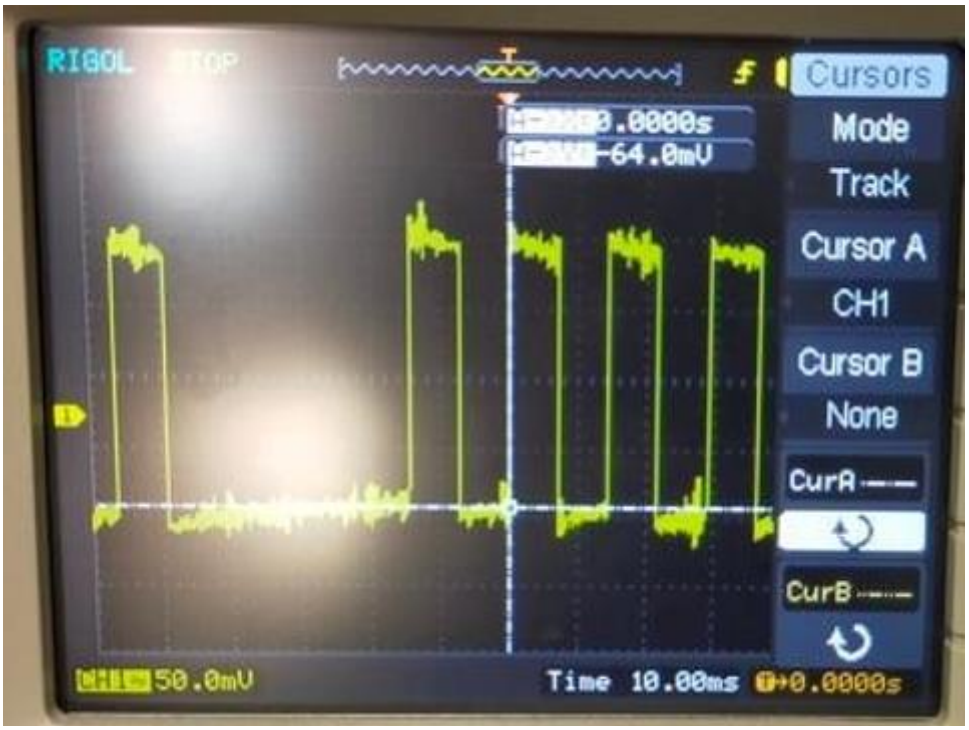
Kod Programu 5

```
156 __interrupt void Timer_A(void)
157 {
158     licznik += 1;           //do zliczania milisekund *100
159     _BIC_SR_IRQ(LPM3_bits); // wyjście z trybu LPM3
160 }
161
162 void start_s() //funkcja wyświetlająca stan początkowy czyli 00:00:00:00
163 {
164     clearDisplay();
165     SEND_CMD(DD_RAM_ADDR2); // zapis w drugiej lini wyświetlacza
166     SEND_CHAR('0');
167     SEND_CHAR('0');
168     SEND_CHAR(':');
169     SEND_CHAR('0');
170     SEND_CHAR('0');
171     SEND_CHAR(':');
172     SEND_CHAR('0');
173     SEND_CHAR('0');
174     SEND_CHAR(':');
175     SEND_CHAR('0');
176     SEND_CHAR('0');
177     SEND_CMD(CUR_SHIFT_LEFT); // powrót kursorem na początek
178 }
```

Kod Programu 6

Badanie poprawności

Po napisaniu kodu należało sprawdzić, czy stoper działa poprawnie, czyli czy wyświetla prawidłowy czas. Użyto do tego oscyloskopu. Jedną nóżkę podpięto do diody Status Led, drugą do masy. Na oscyloskopie ukazał się wykres potwierdzający poprawność. Okres wykresu wynosi 10ms, a nasz stoper miał wyświetlać części dziesiętne i setne sekundy, czyli zmieniał się co 10ms. Zakłócenie w wykresie może być spowodowane drgającą dłonią.



Zdjęcie 1 Oscyloskop przedstawiający poprawność zadania, wykonane na zajęciach Laboratoriów Architektury Komputerów

Bibliografia

[1] Informator Laboratoryjny
[2] http://krzysztof.halawa.staff.iiar.pwr.wroc.pl/wyswietlacz_LCD_HD44780.pdf
[3] http://std2.phys.uni.lodz.pl/mikroprocesory/wyklad/asem_10_lcd.pdf