Politechnika Białostocka

Wydział Elektryczny
Katedra Telekomunikacji i Aparatury Elektronicznej

Instrukcja do zajęć laboratoryjnych z przedmiotu:

Architektura i Programowanie Procesorów Sygnałowych

Kod: TS1C510210

Numer ćwiczenia: 4

Temat ćwiczenia:

Konfiguracja i wykorzystanie systemu przerwań.

Opracował:
dr inż. Dariusz Jańczak

Białystok 2017

Architektura i Programowanie Procesorów Sygnałowych

Temat: Konfiguracja i wykorzystanie systemu przerwań.

1. Cel ćwiczenia

Celem ćwiczenia jest pogłębienie i utrwalenie wiedzy studentów oraz nabycie przez nich umiejętności wykorzystania systemu przerwań w technice mikroprocesorowej. W ramach zajęć studenci zdobywają umiejętności konfiguracji systemu przerwań oraz pisania procedur obsługi przerwań. Zadania realizowane są przy wykorzystaniu procesora sygnałowego TMS320C6713.

2. Zagadnienie do opracowania przed przystąpieniem do zajęć

Przed przystąpieniem do zajęć należy opracować następujące zagadnienia:
- system przerwań: zastosowanie, elementy, cechy, konfiguracja,
- zasady konfiguracji i obsługi systemu przerwań zmiennoprzecinkowego procesora sygnałowego TMS320C6713 (SPRU189 rozdz. 8, SPRU646).

3. Przebieg ćwiczenia

Ćwiczenia laboratoryjne prowadzone są w oparciu o zestaw TMS320C6713 DSP Starter Kit (DSK). DSK składa się ze sprzętu i oprogramowania umożliwiającego realizację procedur przetwarzania sygnałów w oparciu o procesor TMS320C6713.

3.1 Konfiguracji i obsługa systemu przerwań.

W ćwiczeniu należy zrealizować następujące czynności:
- umieścić wektor przerwań od adresu 0x400 (ISTP),
- korzystając, z odpowiednich rejestrów przypisać przerwaniu INT8 obsługę timera1, a INT15 timera0 (MUXL, MUXH)
- napisać procedury obsługi przerwania INT8 oraz INT15 zliczające ilość wywołań przerwania
- odblokować system przerwań (IER, GIE),
- na wstępie przerwania wywoływać na drodze programowej (ISR); następnie uruchomić timer tak by wywoływał przerwanie co 1s,
- zmodyfikować procedurę obsługi przerwania tak by zaświecała i gasła dioda LED.

3.2 Zadania złożone:

Wykonać zestaw zadań przydzielony grupie przez prowadzącego. Zadania obejmują sterowanie portów I/O przy użyciu timerów i systemu przerwań.

4. Sprawozdanie powinno zawierać:

- kody źródłowe wraz z opisem,
- wyniki działania procedur,
- analizę kodów źródłowych stosowanych procedur,
- uwagi i wnioski nasuwające się w trakcie wykonywania ćwiczenia.

5. Wymagania BHP

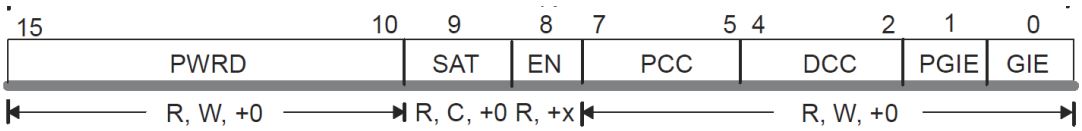
W trakcie realizacji programu ćwiczenia należy przestrzegać zasad omówionych we wstępie do ćwiczeń, zawartych w: „Regulaminie porządkowym w laboratorium" oraz w „Instrukcji obsługi urządzeń elektronicznych znajdujących się w laboratorium z uwzględnieniem przepisów BHP". Regulamin i instrukcja są dostępne w pomieszczeniu laboratoryjnym w widocznym miejscu.

6. Literatura

1. Kowalski H. A., *Procesory DSP dla praktyków*, BTC, Legionowo, 2013
2. Kowalski H. A., *Procesory DSP w przykładach,* BTC, Legionowo, 2012.
3. Dąbrowski A. (red.) *Przetwarzanie Sygnałów Przy użyciu Procesorów Sygnałowych*, Wyd. Politechniki Poznańskiej, Poznań 2000.
4. Texas Instruments, *TMS320C6000 DSP Peripherals Overview*, 2009.
5. Texas Instruments, *TMS320C6000 DSP Interrupt Selector Reference Guide*, 2004.
6. Texas Instruments, TMS320C6000 Chip Support Library API Reference Guide, 2004.
7. Texas Instruments, *TMS320C67x DSP CPU and Instruction Set Reference Guide*, 2006.
8. Texas Instruments, *TMS320C6000 Programmer's Guide*, 2006.
9. Texas Instruments, *TMS320C6000 Optimizing Compiler User's Guide*, 2017.
10. Kehtarnavaz, N., *Real-Time Digital Signal Processing: Based on the TMS320C6000*, Newnes, 2005.
11. Welch T. B., Wright C.H.G., Morrow M.G., *Real-time Digital Signal Processing from Matlab to C with the TMS320C6x DSPs*, Taylor & Francis, 2012.

Dodatek 1.

Maskowanie (blokowanie) globalne przerwań



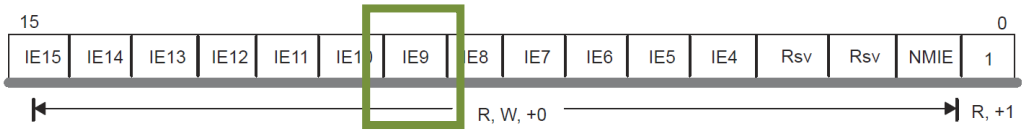
; zamaskowanie globalne przerwań

```
MVC.S2      CSR, B10        ;  odczyt CSR
AND.L2      B10,-2,B10      ;  -2d = 0xFFFE (U2)
MVC.S2      B10,CSR         ;  zapis CSR z GIE=0
```

; odmaskowanie globalne przerwań

```
MVC.S2      CSR, B10        ;  odczyt CSR
OR.L2       1,B10,B10       ;
MVC.S2      B10,CSR         ;  zapis CSR z GIE=1
```

Maskowanie (blokowanie) poszczególnych przerwań



; zamaskowanie przerwania INT9

```
MVK.S2      0xFDFF, B4      ; 0xFDFF =>  bit9 = 0
MVC.S2      IER, B5         ;  odczyt IER
AND.L2      B4,B5,B5        ;
MVC.S2      B5,IER          ;  zapis IER z IE9=0
```

; odmaskowanie przerwania INT9

```
MVK.S2      0x200, B4       ; 0x200 =>  bit9 = 1
MVC.S2      IER, B5         ;  odczyt IER
OR.L2       B4,B5,B5        ;
MVC.S2      B5,IER          ;  zapis IER  z IE9=1
```

Wybór źródeł przerwań

Rejestry wyboru źródeł przerwań i konfiguracji prz. zewn.

Byte Address	Abbreviation	Name	Description
019C0000h	MUXH	Interrupt multiplexer high	Selects which interrupts drive CPU interrupts 10–15 (INT10–15)
019C0004h	MUXL	Interrupt multiplexer low	Selects which interrupts drive CPU interrupts 4–9 (INT4–INT9)
019C0008h	EXTPOL	External interrupt polarity	Sets the polarity of the external interrupts (EXT_INT4–EXT_INT7)

Figure 14–2. Interrupt Multiplexer Low Register (MUXL)

31	30 – 26	25 – 21	20 – 16
Reserved	INTSEL9	INTSEL8	INTSEL7
R, +0	RW, +01001	RW, +01000	RW, +00111

15	14 – 10	9 – 5	4 – 0
Reserved	INTSEL6	INTSEL5	INTSEL4
R, +0	RW, +00110	RW, +00101	RW, +00100

Figure 14–3. Interrupt Multiplexer High Register (MUXH)

31	30 – 26	25 – 21	20 – 16
Reserved	INTSEL15	INTSEL14	INTSEL13
R, +0	RW, +00010	RW, +00001	RW, +00000

15	14 – 10	9 – 5	4 – 0
Reserved	INTSEL12	INTSEL11	INTSEL10
R, +0	RW, +01011	RW, +01010	RW, +00011

LDW, STW – zapis/odczyt

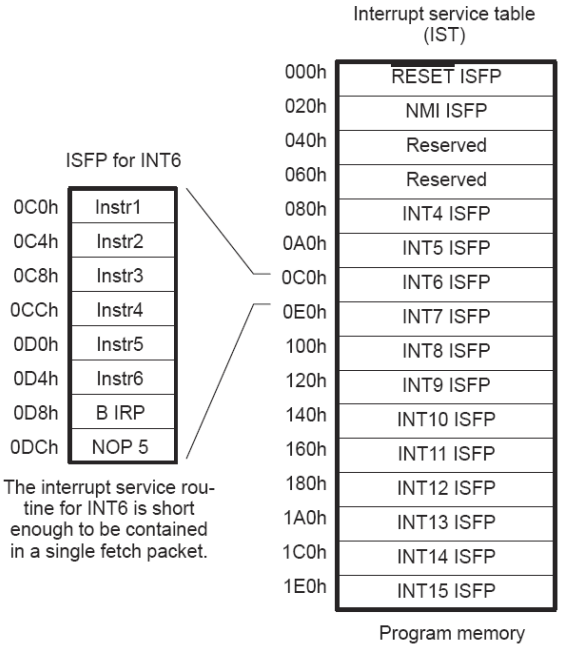
```
            .global start
MUXH        .set    0x019C0000
MUXL        .set    0x019C0004

            .text
; Timer0 na przerwaniu INT5
start:  MVKL.S2     MUXL, B10
            MVKH.S2     MUXL, B10   ;  w B10 adres MUXL

            MVKL.S2     0x0020, B9     ;TINT0 = 00001b   na   INTSEL5

            STW.D2          B9,*B10
```

Dodatek 2.

Wybór źródeł przerwań i odmaskowanie

```
#define MUXH            0x019C0000
#define MUXL            0x019C0004


void main()
{
        . . . . .

        *(volatile unsigned int *)MUXL = 0x20; //INT5=timer0
        *(volatile unsigned int *)MUXH = 0;

        IER = 0x022;
        CSR = CSR | 0x01;

        . . . . . .

        while(1);
        . . . . .
}
```

Tablica wektorów przerwań




```
INT5:       b _timer_int
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
```

Procedura obsługi przerwania

```
interrupt void timer_int()
{
        *(volatile unsigned char *)IO_PORT = ledy;
                ledy = ~ledy;

        return;
}
```