Politechnika Białostocka

Wydział Elektryczny
Katedra Telekomunikacji i Aparatury Elektronicznej

Instrukcja do zajęć laboratoryjnych z przedmiotu:

Architektura i Programowanie Procesorów Sygnałowych

Kod: TS1C510210

Numer ćwiczenia: 3

Temat ćwiczenia:

Obsługa i wykorzystanie timerów i portów wejścia/wyjścia.

Opracowali:
dr inż. Dariusz Jańczak

Białystok 2017

Architektura i Programowanie Procesorów Sygnałowych

Temat: Obsługa i wykorzystanie timerów i portów wejścia/wyjścia.

1. Cel ćwiczenia

Celem ćwiczenia jest ugruntowanie wiedzy studentów i nabycie przez nich umiejętności z zakresu wykorzystania elementów architektury oraz podstawowych funkcji sygnałowych procesorów sygnałowych na przykładzie zmiennoprzecinkowego procesora TMS320C6713. W ramach zajęć studenci ugruntowują wiedzę na temat zasad obsługi portu I/O i timerów oraz zdobywają umiejętności konfiguracji i wykorzystania tych zasobów. Ponadto studenci nabywają umiejętność realizacji i wykorzystania synchronizacji metodą kontroli bitów stanu (pooling).

2. Zagadnienie do opracowania przed przystąpieniem do zajęć

Przed przystąpieniem do zajęć należy opracować następujące zagadnienia:
- ♦ architektura sygnałowego procesora zmiennoprzecinkowego TMS320C6713,
- ♦ obsługa portu I/O układu DSK procesora TMS320C6713,
- ♦ synchronizacja poprzez kontrolę bitów stanu (pooling), (testowanie, zerowanie i ustawianie bitów, skoki warunkowe),
- ♦ zasady obsługi timera (SPRU582).

3. Przebieg ćwiczenia

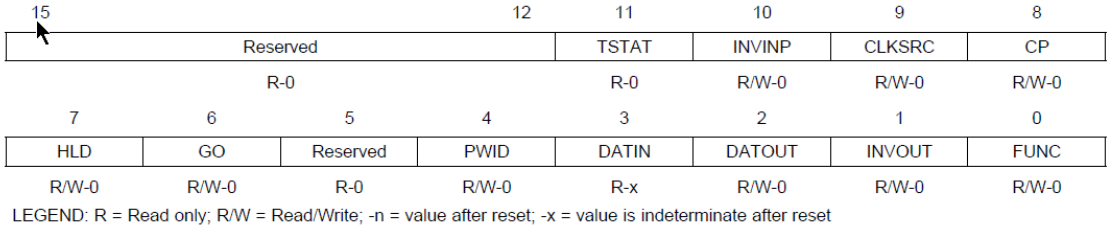
Ćwiczenia laboratoryjne prowadzone są w oparciu o zestaw TMS320C6713 DSP Starter Kit (DSK). DSK składa się ze sprzętu i oprogramowania umożliwiającego realizację procedur przetwarzania sygnałów w oparciu o procesor TMS320C6713.

3.1 Obsługa portu wejścia/wyjścia:
W ćwiczeniu należy zrealizować następujące procedury:
- ♦ napisać procedurę odczytującą i zapisującą wartość z portu I/O (najstarszy bajt pod adresem *0x90080000*),

0x90080000	SW3	SW2	SW1	SW0	LED3	LED2	LED1	LED0
	R 6	R 5	R 4	R 3	W/R 2	W/R 2	W/R 1	W/R 0

- ♦ w oparciu o port I/O napisać następujące procedury:
  - sygnalizującą diodami LED ustawienie przełączników (zmiana położenia przełącznika powoduje zgaszenie lub zapalenie odpowiadającej mu diody),
- ♦ wykorzystując port I/O napisać zrealizować jedną z następujących procedur:
  - sygnalizacja diodami LED w kodzie NKB liczby przełączników w określonej pozycji,
  - zliczającej modulo siedem liczby zmian położenia wybranego przełącznika,

3.1 Konfiguracja i wykorzystanie timerów:
Testując wartość bitu TSTAT (rej. CTL) (pooling) napisać i uruchomić procedurę zapalającą w odstępie sekundowym wybraną diodę.



15 ... 12	11	10	9	8
Reserved	TSTAT	INVINP	CLKSRC	CP
R-0	R-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
HLD	GO	Reserved	PWID	DATIN	DATOUT	INVOUT	FUNC
R/W-0	R/W-0	R-0	R/W-0	R-x	R/W-0	R/W-0	R/W-0

LEGEND: R = Read only; R/W = Read/Write; -n = value after reset; -x = value is indeterminate after reset

Uwagi:
- zastosować wewnętrzne taktowanie (CLKSRC=1). Timer jest wówczas taktowany z częstotliwością 225MHz/4,
- ustawić tryb pracy zegarowej bit C/P=1.

3.2 Zadania złożone:
Wykonać zestaw zadań przydzielony grupie przez prowadzącego.
Przykładowe zadania:

- zapalanie w odstępie półsekundowym kolejnych diod (z wykorzystaniem timera0 i adresowania w trybie cyrkularnym),
- zliczanie czasu – wyświetlanie liczby sekund za pomocą diod LED (z wykorzystaniem timera1).
- zliczanie modulo osiem ilości sekund pomiędzy zmianą położenia określonego przełącznika i wyświetlanie czasu za pomocą diod LED (z wykorzystaniem timera0).

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.

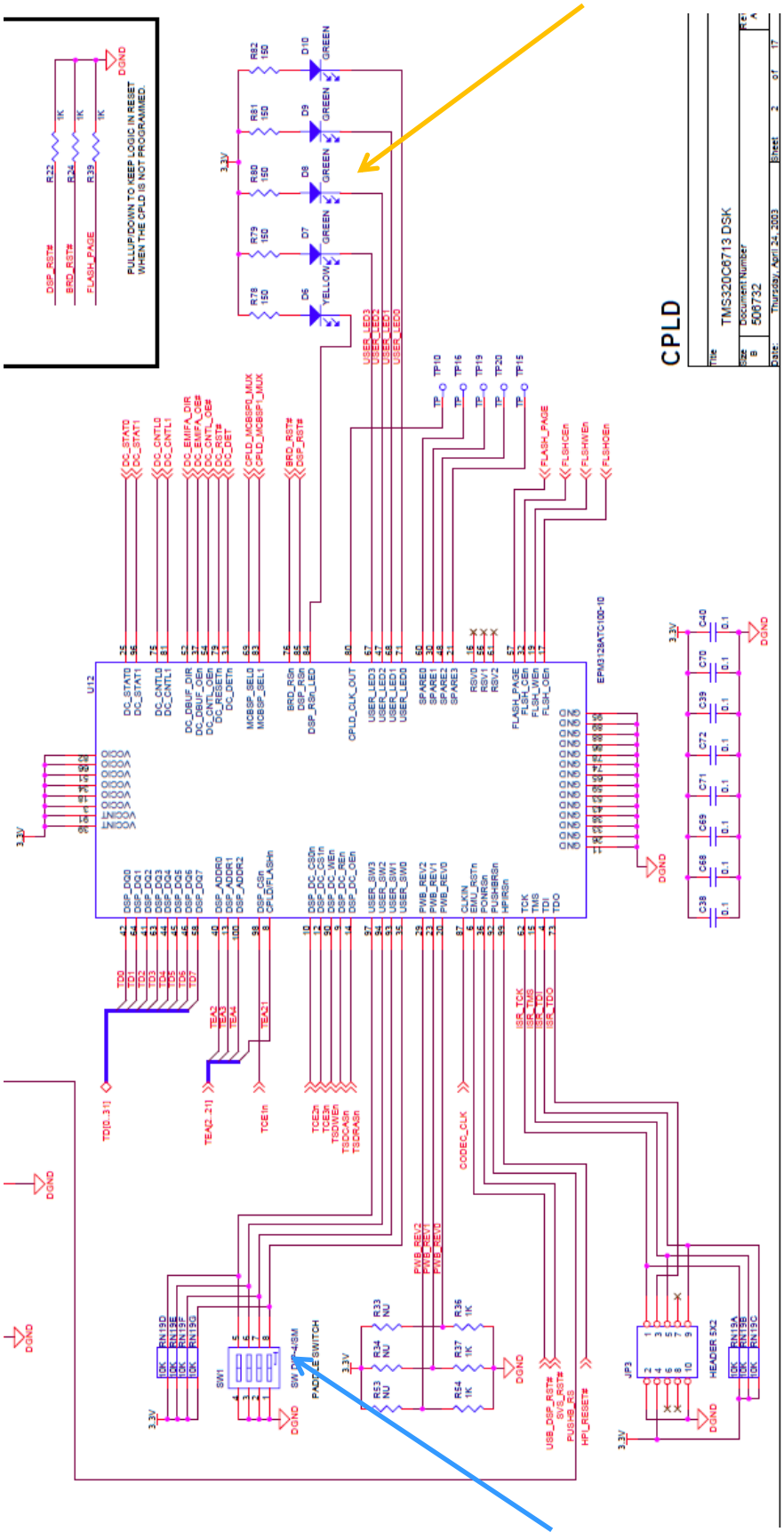
5. Wymagania BHP

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 32-Bit Timer Reference Guide*, 2005.
6. Texas Instruments, *TMS320C67x DSP CPU and Instruction Set Reference Guide*, 2006.
7. Texas Instruments, *TMS320C6000 Programmer's Guide*, 2006.
8. Kehtarnavaz, N., *Real-Time Digital Signal Processing: Based on the TMS320C6000*, Newnes, 2005.
9. 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.


CPLD
TMS320C6713 DSK
PULLUP/DOWN TO KEEP LOGIC IN RESET WHEN THE CPLD IS NOT PROGRAMMED.
DSP_RST#
BRD_RST#
FLASH_PAGE
DGND
3.3V
U12
EPM3128ATC100-10
SW1
SW_DIP-4/SM
PAGE_SWITCH
JP3
HEADER 5X2
USER_LED3
USER_LED2
USER_LED1
USER_LED0
CPLD_CLK_OUT
CODEC_CLK
HPI_RESET#
USB_DSP_RST#
SW6_RST#
PUSH6_RS

Dodatek 2.

```
; konfiguracja timera
CTL .set 01940000h
PRD .set 01940004h
dz_cz .set 56250000 ;dzielnik częstotliwości zeg. CPU do timera


.text
;
; ustawienie timera
; przerwania co 1s
MVKL.S2 CTL, B1 ; w B1 adres CTL
|| MVKL.S1 dz_cz, A10 ;dzielnik częstotliwości zeg. CPU do timera
MVKH.S2 CTL, B1 ; w B1 adres CTL
|| MVKH.S1 dz_cz, A10 ;dzielnik częstotliwości zeg. CPU do timera
MVK.S2 201h, B0 ;słowo sterujące
|| ADD.L1x 4, B1, A0 ; w A0 adres PRD
STW.D2 B0, *B1 ; zapis do CTL
|| STW.D1 A10, *A0 ; zapis do PRD
MVK.S2 201h, B0 ;słowo sterujące
STW.D2 B0, *B1 ; zapis do CTL
|| MVK.S2 02C1h, B0 ;słowo sterujące
STW.D2 B0, *B1 ; zapis do CTL - start timera
```

Dodatek 3.

/* konfiguracja timera przy użyciu csl */

```
#include <csl.h>
#include <csl_timer.h>


TIMER_Handle hTimer;


void main()
{
        hTimer = TIMER_open(TIMER_DEV0,0);
        TIMER_configArgs(hTimer, 0x000002C0, 0x035A4E90, 0x00000000);

//……..


        TIMER_close(hTimer);

}
```

Table 22–1. TIMER Configuration Structure

Syntax	Type	Description
TIMER_Config	S	Structure used to set up a timer device

Table 22–2. TIMER APIs

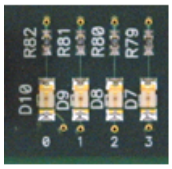
(a) Primary Functions

Syntax	Type	Description
TIMER_close	F	Closes a previously opened timer device
TIMER_config	F	Configure timer using configuration structure
TIMER_configArgs	F	Sets up the timer using the register values passed in
TIMER_open	F	Opens a TIMER device for use
TIMER_pause	F	Pauses the timer
TIMER_reset	F	Resets the timer device associated to the handle
TIMER_resume	F	Resumes the timer after a pause
TIMER_start	F	Starts the timer device running

Dodatek 4.

Wykorzystanie biblioteki bsl

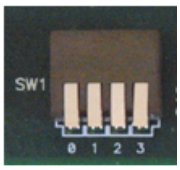
LED API



DSK6713_LED_init() Inicjalizacja "LED-ów"
DSK6713_LED_off() Zgaszenie LEDa
DSK6713_LED_on() Zaświecenie LEDa
DSK6713_LED_toggle() Przełączenie stanu LEDa

Wymagane:
dsk6713bsl.lib
dsk6713.h
dsk6713_led.h

DIP Switch API



DSK6713_DIP_init() Inicjalizacja przełączników DIP
DSK6713_DIP_get() Odczyt przełączników DIP

```
void DSK6713_DIP_init()
        /*Inicjalizacja modułu przełączników DIP; musi być wywołana przed
        użyciem funkcji odczytu przełączników DIP */

Uint32 DSK6713_DIP_get(Uint32 dipNum)  // odczyt  DIP_dipNum
        /*  dipNum - wartości od 0 do 3
         wartość zwracana: 0 - wyłączony;  1 - włączony*/
```

Przykład

```
DSK6713_DIP_init();  // Inicjalizacja

if (DSK6713_DIP_get(2) == 1)
        {   // gdy DIP_2 włączony}
else
        {   // gdy DIP_2 wyłączony }
```