**Wydział Elektroniki i Technik Informacyjnych**
**Politechnika Warszawska**

**Technika Mikroprocesorowa**

# Sprawozdanie z laboratorium nr 3
# Projekt licznika 8-bitowego przy użyciu mikrokontrolera z rodziny MSP430F16x

**Jakub Sikora**
**Konrad Winnicki**
**Marcin Dolicher**

**Warszawa, 7 maja 2018**

# Spis treści

# 1. Zadanie laboratoryjne

## 1.1. Polecenie

Celem trzeciego zadania laboratoryjnego było zaprojektowanie, złożenie, zaprogramowanie i przetestowanie układu z mikrokontrolerem MSP430F16x tak aby działał on jako licznik 8-bitowy zliczający w dół liczbę wciśnięć programowo odszumionego przycisku monostabilnego, który umożliwiał asynchroniczne ładowanie liczby do pamięci z blokowaniem zliczania. Ładowanie liczby powinno odbyć się przy pomocy przerwania maskowalnego. Liczby powinny być przechowywane w Naturalnym Kodzie Binarnym, w skrócie NKB.

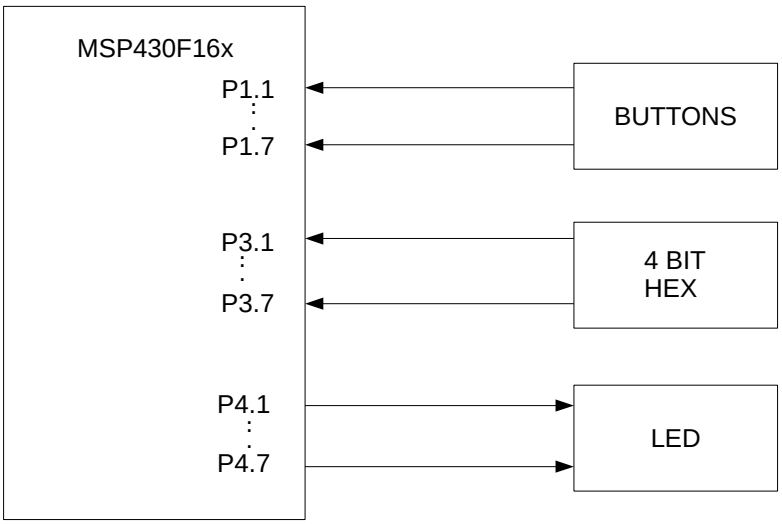
## 1.2. Szczegółowe uwagi

W odróżnieniu do zadania z drugiego laboratorium, wyświetlanie zawartości licznika powinno odbyć się na dwóch wyświetlaczach 7-segmentowych. Liczby powinny być wyświetlane w postaci szesnastkowej. Ładowanie zawartości licznika ma odbywać się za pomocą nastawników hex, a sterowanie licznikiem ma odbywać się za pomocą dwóch przycisków, z czego co najmniej jeden ma być obsługiwany w przerwaniu. Ostatecznie, w momencie gdy procesor nic nie robi, powinien przejść w tryb oszczędzania energii LPM0.

# 2. Projekt licznika

## 2.1. Opis sprzętu

Zmiana platformy z procesora Z80 na mikrokontroler MSP43016x znacząco ułatwiało zadanie i pozwoliło na dokonanie kilku ulepszeń w projekcie. Mikrokontroler ma wbudowane układy podtrzymujące w swoich portach wejścia/wyjścia dlatego też nie są potrzebne dodatkowe układy złożone z buforów trójstanowych. Schemat ideowy układu znajduje się na rysunku poniżej.




Rysunek 2.1. Schemat ideowy

Do portu pierwszego podłączyliśmy zestaw ośmiu przycisków monostabilnych za pomocą których użytkownik może obsługiwać licznik. Do portu trzeciego podłączyliśmy obrotowe nastawniki liczby w kodzie heksadecymalnym. Ostatecznie do portu czwartego podłączyliśmy układ z dwoma wyświetlaczami siedmiosegmentowymi wraz z odpowiednim dekoderem. W porównaniu z rozwiązaniem opartym na procesor Z80, zastosowanie mikrokontrolera znacząco upraszcza ostateczny układ. Mniejsza ilość peryferiali jest możliwa dzięki wewnętrznym rejestrom, które są przypisane do odpowiednich portów.

## 2.2. Opis oprogramowania

W trakcie projektowania oprogramowania skupiliśmy się na tym aby nasze rozwiązanie pobierało jak najmniej energii. W trakcie oczekiwania na wciśnięcie przycisku, procesor miał pozostawać w stanie ograniczonego poboru mocy, podczas którego zegary procesora pozostawały wyłączone. Wyjście z tego stanu odbywało się za pomocą przerwania maskowalnego. Po przyjęciu przerwania i jego obsłużeniu, program miał zadecydować czy ma coś jeszcze do zrobienia czy jednak może wrócić do stanu niskiego poboru mocy. W przypadku asynchronicznego ładowania, po wczytaniu liczby procesor mógł spokojnie przejść w stan oczekiwania. Po wciśnięciu przycisku `CLK`, mikrokontroler powinien rozpocząć procedurę odszumiania przycisku.

### 2.2.1. Odszumianie przycisku

W momencie wciśnięcia przycisku `CLK` program przechodzi do specjalnej procedury podczas której zliczane są kolejne niskie stany przycisku. Po pojawieniu się kolejnych 256 niskich stanów pod rząd, program uznaje że przycisk się już ustabilizował i traktuje to jako jedno wciśnięcie. Program następnie oczekuje na kolejne 256 wysokich stanów, które świadczą o puszczeniu i ustabilizowaniu przycisku. Po odnotowaniu takiej sytuacji, procesor przechodzi w tryb zmniejszonego poboru mocy i oczekuje na kolejne przerwanie pochodzące albo od przycisku `LD` albo `CLK`

### 2.2.2. Przerwania

Procedura przerwania zostaje wywołana przy wciśnięciu przycisku `LD` lub `CLK`. W trakcie tej procedury, sprawdzane jest od którego przycisku przyszło przerwanie. Podczas obsługi przycisku ładowania, pobierana jest wartość z nastawników hex a następnie zostaje ona załadowana do pamięci i wypuszczona na wyświetlacze. Ostatecznie po odświeżeniu, procesor wraca do trybu zmniejszonego poboru mocy. W przypadku przyjęcia przerwania od przycisku zegarowego, program przechodzi do głównej pętli programu a do trybu zmniejszonego poboru mocy przechodzi tylko w po zakończeniu obsługi wciśnięcia przycisku.

### 2.2.3. Kod programu

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
.cdecls C,LIST,"msp430.h" ; Include device header file

;-------------------------------------------------------------------------------
.def RESET ; Export program entry-point to
; make it known to linker.
;-------------------------------------------------------------------------------
.text ; Assemble into program memory.
.retain ; Override ELF conditional linking
; and retain current section.
.retainrefs ; And retain any sections that have
; references to current section.

;-------------------------------------------------------------------------------
```

```
;-------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------
RESET:
MOV.W #__STACK_END,SP ; set up stack
MAIN:
MOV.B #0FFh, &P2DIR

MOV.W #WDTPW—WDTHOLD,&WDTCTL ;Stop watchdog timer
MOV.B #0FFh, &P4DIR
MOV.B #0FFh, &P4OUT
MOV.B #000h, &P1IFG
MOV.B #060h, &P1IE ; aktywacja przerwan od CLK i LD
MOV.B #060h, &P1IES

MOV.B &P3IN, R4 ; inicjacja wartosci licznika
MOV.B R4, &P4OUT ; odswiezenie licznika
MOV #01Fh, R6 ; ilosc cykli debounce
MOV R6, R5 ; ustawienie licznika debounce
BIC #0FFh, R7 ; wybrana obsluga zbocza opadajacego, main sygnalizuje ukonczenie zadan

BIS #GIE+CPUOFF,SR ; aktywacja LPM1 - program zatrzymuje sie w tym miejscu
MAIN_ LOOP: ; poczatek głównej petli
BIC #02h, R7

; poczatek obslugi licznika
BIT #01h, R7
JNZ LOOP_POSITIVE ; wybranie obslugi konkretnego zbocza
LOOP_NEGATIVE: ; obsluga zbocza opadajacego
BIT.B #040h, &P1IN ; stan przycisku CLK
JZ IF_1
MOV R6, R5 ; jesli przycisk niewcisniety, zaladuj licznik debounce
BIS #02h, R7 ; sygnalizacja pracy w toku
BIC.B #040h, &P1IE ; wylaczenie przerwania od CLK(P1.6)
BIS.B #040h, &P1IES ; wybranie zbocza opadajacego
BIC.B #040h, &P1IFG ; wyzerowanie zadania przerwania od CLK
BIS.B #040h, &P1IE ; wlaczenie przerwania od CLK
JMP COUNTER_END
IF_1:
JNZ IF_2
DEC R5 ; jesli przycisk wcisniety, dekrementuj licznik debounce
IF_2:
CMP #0, R5 ; czy licznik debounce równy zero
JZ IF_2_CONTINUE ; jesli licznik debounce zerowy, dalsza obsluga
BIS #02h, R7 ; sygnalizacja pracy w toku
JMP COUNTER_END
IF_2_CONTINUE:
BIS #02h, R7 ; sygnalizacja pracy w toku
DEC R4 ; wykryto poprawne zbocze opadajace, akcja licznika
```

```
MOV.B R4, &P4OUT ; odswiezenie LED
BIS #01h, R7 ; wybranie obslugi zbocza narastajacego
JMP COUNTER_END ; skok na koniec obslugi licznika
LOOP_POSITIVE: ; obsluga zbocza narastajacego
BIS #02h, R7 ; sygnalizacja pracy w toku
BIT.B #040h, &P1IN ; stan przycisku CLK
JNZ IF_3
MOV R6, R5 ; jesli przycisk wcisniety, zaladuj licznik debounce
IF_3:
JZ IF_4
DEC R5 ; jesli przycisk niewcisniety, dekrementuj licznik debounce
IF_4:
CMP #0, R5 ; odswiezenie stanu flag
JZ IF_4_CONTINUE ; jesli licznik debounce zerowy, dalsza obsluga W IF_4_CONTINUE
BIS #02h, R7 ; sygnalizacja pracy w toku
JMP COUNTER_END
IF_4_CONTINUE:
BIC #01h, R7 ; wybranie obslugi zbocza opadajacego
BIC.B #040h, &P1IE ; wylaczenie przerwania od CLK(P1.6)
BIS.B #040h, &P1IES ; wybranie zbocza opadajacego
BIC.B #040h, &P1IFG ; wyzerowanie zadania przerwania od CLK
BIS.B #040h, &P1IE ; wlaczenie przerwania od CLK
MOV R6, R5 ; zaladowanie licznika debounce
COUNTER_END: ; koniec obslugi licznika

; dalsze operacje...

; jesli zadna operacja nie zglosila pracy w toku przejdz w tryb uspienia
BIT #02h, R7
JNZ MAIN_LOOP
BIS #GIE+CPUOFF,SR ; aktywacja LPM3 - program zatrzymuje sie w tym miejscu

JMP MAIN_LOOP ; powrót do poczatku glównej petli

P1_INT_VECTOR: ; procedura obslugi przerwania P1
BIT.B #020h, & P1IFG ; sprawdzenie czy aktywne jest zadanie przerwania lub niski stan na
LD
JNZ P1_INT_VEC_LOAD
BIT.B #020h, & P1IN ;
JNZ P1_INT_VEC_LOAD_DONE
P1_INT_VEC_LOAD: ; ciagle ladowanie gdy niski stan LD
BIC.B #020h, P1IFG ; zgaszenie flagi przerwania od LD(P1.5)
MOV.B &P3IN, R4 ; zaladowanie nowej wartosci licznika
MOV.B R4, &P4OUT ; odswiezenie wyswietlania
JMP P1_INT_VECTOR
P1_INT_VEC_LOAD_DONE:
BIT.B #040h, &P1IFG ; sprawdzenie przerwania od CLK(P1.6)
JZ P1_INT_VEC_CLK_DONE
BIC.B #040h, &P1IE ; interesuje nas tylko pierwsze przerwanie od CLK
BIC.B #040h, &P1IFG ; zgaszenie flagi przerwania od CLK
BIC #CPUOFF+SCG1+SCG0,0(SP) ; powrót z przerwania ze zmienionym SR
```

```
P1_INT_VEC_CLK_DONE:
RETI


;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
.global __STACK_END
.sect .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
.sect ".reset" ; MSP430 RESET Vector
.short RESET

.sect ".int04"
.short P1_INT_VECTOR
```