

	Instytut Informatyki Politechniki Śląskiej Zespół Mikroinformatyki i Teorii Automatów Cyfrowych Laboratorium SMiW			
Rok akademicki	Rodzaj studiów*: SSI/NSI/NSM	Numer ćwiczenia:	Grupa	Sekcja
2018/2019	SSI	7 i 9	5	9
Data i godzina planowana ćwiczenia: dd/mm/rrrr - gg:mm	2018-12-10 – 13:15-14:45	Prowadzący: OA/JP/KT/GD/BSz/GB	JP	
Data i godzina wykonania ćwiczenia: dd/mm/rrrr - gg:mm	2018-12-10 – 13:15-14:45			
<h2>Sprawozdanie</h2>				
Temat ćwiczenia: <h1>Mikrokontrolery z serii AVR cz. 1 i 2</h1>				
Skład sekcji:	<ol style="list-style-type: none"> 1. Drabińska Martyna 2. Krasoń Bartłomiej 3. Miciak Michał 			

1. Opis zadania

Na kontrolerze AVR ATmega2560 zrealizować program realizujący następujące założenie:

TREŚĆ ZADANIA ZE ZDJĘCIA

Zadanie zrealizować w języku asemblera.

2. Kod programu w języku assemblerowym

```
////////////////////////////////////
// Laboratory AVR Microcontrollers Part2
// Program template for lab 9
// Please fill in this information before starting coding
// Authors:
//   Martyna Drabinska
//   Michal Miciak
//   Bartlomiej Krason
// Group: 5
// Section: 9
//
// Task: E
//   sos signal
// Todo:
//
//
// Version: 5.0
////////////////////////////////////
.nolist ;quartz assumption 4Mhz
.include "m2560def.inc"
;////////////////////////////////////
.list
.equ xlength = 100
;////////////////////////////////////
; EEPROM - data non volatile memory segment
.ESEG

;////////////////////////////////////
; StaticRAM - data memory.segment
.DSEG
.ORG 0x200; may be omitted this is default value
; Destination table (xlengthx bytes).
; Replace "xlengthx" with correct value
TAB_RAM: .BYTE xlength

;////////////////////////////////////
; CODE - Program memory segment
; Please Remember that it is "word" address space
;
.CSEG
.org 0x0000 ; may be omitted this is default value
jmp RESET ; Reset Handler

; Interrupts vector table / change to your procedure only when needed
jmp EXT_INT0 ; IRQ0 Handler
jmp EXT_INT1 ; IRQ1 Handler
jmp EXT_INT2 ; IRQ2 Handler
jmp EXT_INT3 ; IRQ3 Handler
jmp EXT_INT4 ; IRQ4 Handler
jmp EXT_INT5 ; IRQ5 Handler
jmp EXT_INT6 ; IRQ6 Handler
jmp EXT_INT7 ; IRQ7 Handler
jmp HPCINT0 ; PCINT0 Handler
jmp HPCINT1 ; PCINT1 Handler
jmp HPCINT2 ; PCINT2 Handler
```

```

jmp WDT ; WDT Handler
jmp TIM2_COMP_A ; Timer2 CompareA Handler
jmp TIM2_COMP_B ; Timer2 CompareB Handler
jmp TIM2_OVF ; Timer2 Overflow Handler
jmp TIM1_CAPT ; Timer1 Capture Handler
jmp TIM1_COMP_A ; Timer1 CompareA Handler
jmp TIM1_COMP_B ; Timer1 CompareB Handler
jmp TIM1_COMP_C ; Timer1 CompareC Handler
jmp TIM1_OVF ; Timer1 Overflow Handler
jmp TIM0_COMP_A ; Timer0 CompareA Handler
jmp TIM0_COMP_B ; Timer0 CompareB Handler,
jmp TIM0_OVF ; Timer0 Overflow Handler
jmp SPI_STC ; SPI Transfer Complete Handler
jmp USART0_RXC ; USART0 RX Complete Handler
jmp USART0_UDRE ; USART0,UDR Empty Handler
jmp USART0_TXC ; USART0 TX Complete Handler
jmp ANA_COMP ; Analog COmparator Handler
jmp ADC ; ADC Conversion Complete Handler
jmp EE_RDY ; EEPROM Ready Handler
jmp TIM3_CAPT ; Timer3 Capture Handler
jmp TIM3_COMP_A ; Timer3 CompareA Handler
jmp TIM3_COMP_B ; Timer3 CompareB Handler
jmp TIM3_COMP_C ; Timer3 CompareC Handler
jmp TIM3_OVF ; Timer3 Overflow Handler
jmp USART1_RXC ; USART1 RX Complete Handler
jmp USART1_UDRE ; USART1,UDR Empty Handler
jmp USART1_TXC ; USART1 TX Complete Handler
jmp TWI ; Two-wire Serial Interface Interrupt Handler
jmp SPM_RDY ; SPM Ready Handler
jmp TIM4_CAPT ; Timer4 Capture Handler
jmp TIM4_COMP_A ; Timer4 CompareA Handler
jmp TIM4_COMP_B ; Timer4 CompareB Handler
jmp TIM4_COMP_C ; Timer4 CompareC Handler
jmp TIM4_OVF ; Timer4 Overlflow Handler
jmp TIM5_CAPT ; Timer5 Capture Handler
jmp TIM5_COMP_A ; Timer5 CompareA Handler
jmp TIM5_COMP_B ; Timer5 CompareB Handler
jmp TIM5_COMP_C ; Timer5 CompareC Handler
jmp TIM5_OVF ; Timer5 Overlflow Handler
jmp USART2_RXC ; USART2 RX Complete Handler
jmp USART2_UDRE ; USART2,UDR Empty Handler
jmp USART2_TXC ; USART2 TX Complete Handler
jmp USART3_RXC ; USART3 RX Complete Handler
jmp USART3_UDRE ; USART3,UDR Empty Handler
jmp USART3_TXC ; USART3 TX Complete Handler

```

////////////////////////////////////

```

EXT_INT0: ; IRQ0 Handler
EXT_INT1: ; IRQ1 Handler
EXT_INT2: ; IRQ2 Handler
EXT_INT3: ; IRQ3 Handler
EXT_INT4: ; IRQ4 Handler
EXT_INT5: ; IRQ5 Handler
EXT_INT6: ; IRQ6 Handler
EXT_INT7: ; IRQ7 Handler
HPCINT0: ; PCINT0 Handler
HPCINT1: ; PCINT1 Handler
HPCINT2: ; PCINT2 Handler
WDT: ; WDT Handler
TIM2_COMP_A: ; Timer2 CompareA Handler
TIM2_COMP_B: ; Timer2 CompareB Handler
TIM2_OVF: ; Timer2 Overflow Handler
TIM1_CAPT: ; Timer1 Capture Handler
TIM1_COMP_A: ; Timer1 CompareA Handler
TIM1_COMP_B: ; Timer1 CompareB Handler
TIM1_COMP_C: ; Timer1 CompareC Handler
TIM1_OVF: ; Timer1 Overflow Handler

```

```

    subi r25, 0x01
    cpi r25, 0x00                ; sprawdzenie końca tablicy

    breq TABLE_S                ; jeżeli koniec tablic, ustaw wskaźnik tabeli na nowo

    elpm R16,Z+                  ; pobranie kolejnej wartości tablicy do r16
    out PORTB, r16                ; przekazanie wartości na port diody
    rjmp COUNTER_SET

TIM0_COMP_A: ; Timer0 CompareA Handler
TIM0_COMP_B: ; Timer0 CompareB Handler
TIM0_OVF: ; Timer0 Overflow Handler
SPI_STC: ; SPI Transfer Complete Handler
USART0_RXC: ; USART0 RX Complete Handler
USART0_UDRE: ; USART0,UDR Empty Handler
USART0_TXC: ; USART0 TX Complete Handler
ANA_COMP: ; Analog COMparator Handler
HADC: ; ADC Conversion Complete Handler
EE_RDY: ; EEPROM Ready Handler
TIM3_CAPT: ; Timer3 Capture Handler
TIM3_COMP_A: ; Timer3 CompareA Handler
TIM3_COMP_B: ; Timer3 CompareB Handler
TIM3_COMP_C: ; Timer3 CompareC Handler
TIM3_OVF: ; Timer3 Overflow Handler
USART1_RXC: ; USART1 RX Complete Handler
USART1_UDRE: ; USART1,UDR Empty Handler
USART1_TXC: ; USART1 TX Complete Handler
TWI: ; Two-wire Serial Interface Interrupt Handler
SPM_RDY: ; SPM Ready Handler
TIM4_CAPT: ; Timer4 Capture Handler
TIM4_COMP_A: ; Timer4 CompareA Handler
TIM4_COMP_B: ; Timer4 CompareB Handler
TIM4_COMP_C: ; Timer4 CompareC Handler
TIM4_OVF: ; Timer4 Overflow Handler
TIM5_CAPT: ; Timer5 Capture Handler
TIM5_COMP_A: ; Timer5 CompareA Handler
TIM5_COMP_B: ; Timer5 CompareB Handler
TIM5_COMP_C: ; Timer5 CompareC Handler
TIM5_OVF: ; Timer5 Overflow Handler
USART2_RXC: ; USART2 RX Complete Handler
USART2_UDRE: ; USART2,UDR Empty Handler
USART2_TXC: ; USART2 TX Complete Handler
USART3_RXC: ; USART3 RX Complete Handler
USART3_UDRE: ; USART3,UDR Empty Handler
USART3_TXC: ; USART3 TX Complete Handler
    reti                ; return from all no used

;////////////////////////////////////
; Program start
RESET:
    ldi r16, 0xFF
    out DDRB, r16
    ;//enable counter1 interrupt

TABLE_S:

    ldi r25, 0x1C                ; 28 - rozmiar tablicy, do licznika
    ldi z1, low(TAB_ROM*2)        ; ładujemy adres tablicy do Z (razy 2 bo są
    ; przechowywane jako word)

    ldi zh, high(TAB_ROM*2)
    ldi r16, byte3(TAB_ROM*2)
    out RAMPZ, r16

COUNTER_SET:
    cli
    LDI R20, (1 << TOIE1)
    STS TIMSK1, R20
    ;//set prescaler 1024

```

```

        ldi r20, 0x05
        STS TCCR1B, r20
        ;//ustawic odpowiednio to tcnt1
        ldi r20, 0xfb
        sts TCNT1H, r20
        ldi r20, 0x61
        sts TCNT1L, r20
        sei
loop:
        rjmp loop

//-----
// Program end - Ending loop
//-----
End:
        rjmp END

//-----
// Table Declaration - place here test values
// Test with different table values and different begin addresses of table (also
// above 0x8000)
//
//.org 0x8000
TAB_ROM: .db      0x80, 0x00, 0x80, 0x00, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80,
0x00, 0x80, 0x80, 0x80, 0x00, 0x80
        .db      0x80, 0x80, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x00,
0x00, 0x00, 0x1A, 0x1B, 0x1C, 0x1D

.EXIT
//-----

```

3. Podsumowanie

Aby rozwiązać zadanie musieliśmy skorzystać z przerwania generowanego przez Timer1 układu:

```
TIM1_OVF: ; Timer1 Overflow Handler
```

Dodatkowo w programie musieliśmy odpowiednio ustawić częstotliwość generowania przerwania, aby spełniała ona założenia zadania.

Generowanie sygnału SOS (miganie diody) zrealizowaliśmy wpisując w odpowiedniej kolejności wartości sygnału do tablicy TAB_ROM, z której kolejne wartości następnie przekazywaliśmy na port B.7, tak aby dioda imitowała nadawanie sygnału SOS.

Realizację zadania rozpoczęliśmy od pisania programu w środowisku Atmel Studio 7 oraz przetestowaniu go w symulatorze środowiska. Po ukończeniu programu, wgraliśmy go na kontroler za pomocą środowiska. Program działał prawidłowo zarówno w symulatorze jak i na kontrolerze.