|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| logowydzialu | Instytut Informatyki Politechniki Śląskiej  Zespół Mikroinformatyki i Teorii Automatów Cyfrowych  **Laboratorium SMiW** | | logoii | |
| **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** |
| ***Sprawozdanie*** | | | | |
| **Temat ćwiczenia:**  Mikrokontrolery z serii AVR cz. 1 i 2 | | | | |
| **Skład sekcji:** | 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.IKod 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\_COMPA ; Timer2 CompareA Handler

jmp TIM2\_COMPB ; Timer2 CompareB Handler

jmp TIM2\_OVF ; Timer2 Overflow Handler

jmp TIM1\_CAPT ; Timer1 Capture Handler

jmp TIM1\_C0MPA ; Timer1 CompareA Handler

jmp TIM1\_C0MPB ; Timer1 CompareB Handler

jmp TIM1\_COMPC ; Timer1 CompareC Handler

jmp TIM1\_0VF ; Timer1 Overflow Handler

jmp TIM0\_COMPA ; Timer0 CompareA Handler

jmp TIM0\_COMPB ; 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 HADC ; ADC Conversion Complete Handler

jmp EE\_RDY ; EEPROM Ready Handler

jmp TIM3\_CAPT ; Timer3 Capture Handler

jmp TIM3\_COMPA ; Timer3 CompareA Handler

jmp TIM3\_COMPB ; Timer3 CompareB Handler

jmp TIM3\_COMPC ; 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\_COMPA ; Timer4 CompareA Handler

jmp TIM4\_COMPB ; Timer4 CompareB Handler

jmp TIM4\_COMPC ; Timer4 CompareC Handler

jmp TIM4\_OVF ; Timer4 Overlflow Handler

jmp TIM5\_CAPT ; Timer5 Capture Handler

jmp TIM5\_COMPA ; Timer5 CompareA Handler

jmp TIM5\_COMPB ; Timer5 CompareB Handler

jmp TIM5\_COMPC ; 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\_COMPA: ; Timer2 CompareA Handler

TIM2\_COMPB: ; Timer2 CompareB Handler

TIM2\_OVF: ; Timer2 Overflow Handler

TIM1\_CAPT: ; Timer1 Capture Handler

TIM1\_C0MPA: ; Timer1 CompareA Handler

TIM1\_C0MPB: ; Timer1 CompareB Handler

TIM1\_COMPC: ; Timer1 CompareC Handler

TIM1\_0VF: ; Timer1 Overflow Handler

subi r25, 0x01

cpi r25, 0x00 ; porównanie rejestru r16 ze stałą 0Xff

breq TABLE\_S ; jeżeli zawartość rejestru r16 faktycznie = 0xFF to

; skok do etykiety Main\_0\_SetFlag

elpm R16,Z+

out PORTB, r16

rjmp COUNTER\_SET

TIM0\_COMPA: ; Timer0 CompareA Handler

TIM0\_COMPB: ; 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\_COMPA: ; Timer3 CompareA Handler

TIM3\_COMPB: ; Timer3 CompareB Handler

TIM3\_COMPC: ; 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\_COMPA: ; Timer4 CompareA Handler

TIM4\_COMPB: ; Timer4 CompareB Handler

TIM4\_COMPC: ; Timer4 CompareC Handler

TIM4\_OVF: ; Timer4 Overlflow Handler

TIM5\_CAPT: ; Timer5 Capture Handler

TIM5\_COMPA: ; Timer5 CompareA Handler

TIM5\_COMPB: ; Timer5 CompareB Handler

TIM5\_COMPC: ; Timer5 CompareC Handler

TIM5\_OVF: ; Timer5 Overlflow Handler

USART2\_RXC: ; USART2 RX Complete Handler

USART2\_UDRE:; USART2,UDR Empty Handler

USART2\_TXC: ; USART2 TX Complete Handler

USART3\_RXC: ; USART3 RX Comlplete 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 interupt

    TABLE\_S:

ldi r25, 0x1C ; 28 do licznika

ldi zl, 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)

;ori r20, (1 << OCIE1A) // porównanie wyjścia interuptu A załączone

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

;ldi r17, 0x00;

sei

loop:

;cpi r17, 0x01

;breq COUNTER\_SET

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.IPodsumowanie

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

TIM1\_0VF: ; Timer1 Overflow Handler

Dodatkowo podczas inicjalizacji programu 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.