Systemy sterowania w elektronice przemysłowej
(mechatronika)

Przydatne ustawienia:
Tools->Options->Text Editor->All Language: Settings: Line numbers

UWAGA! W przypadku maszyny wirtualnej należy ustawić częstotliwość zegara dla JTAGa nie większą niż **2MHz**! Jeśli debugowanie nadal nie działa prawidłowo należy częstotliwość zmniejszyć.

Ćwiczenie 1

Temat: Obsługa rejestrów i portów mikrokontrolera – operacje podstawowe

Zadanie: Obsłużyć diody podpięte do portu C i D (opis połączeń w dokumencie mapa połączeń). Należy skonfigurować port jako wyjście a następnie ustawiać na wyjściu wartość 0 i 1 w nieskończonej pętli. Sprawdzić działanie programu za pomocą debuggera uruchamiając program instrukcja po instrukcji (uruchomienie bez debuggera uniemożliwi zaobserwowanie zmiany stanu na diodzie z powodu braku pętli opóźniającej). Następnie ustawić na diodach następującą wartość 11111010 (LED7-LED0, 1-zapalona, 0-zgaszona). Zmieniać 4 młodsze bity w nieskończonej pętli naprzemiennie na wartości xxxx1010 oraz xxxx0101. (x oznacza poprzednią wartość, która nie może ulec zmianie). Należy wykorzystać maskowanie wartości odczytywanej z portu.

Język: Asembler

Zasoby: port C, D

Weryfikacja: Przy użyciu Atmel Studio/Microchip Studio lub AVR Simulator

Ćwiczenie 2

Temat: Pętla opóźniająca, stos, podprocedury

Zadanie: Zmodyfikować program z ćwiczenia 1 dodając pętlę opóźniającą. Zapalać po jednej diodzie po kolei w pętli opóźniającej. Opóźnienie ustalić na wartość (500+N*50) ms. Częstotliwość taktowania ustalić na 16MHz. W pętli należy wykorzystać dowolną instrukcję. Opóźnienie obliczyć biorąc pod uwagę liczbę cykli potrzebnych na wykonanie jednej iteracji pętli. Wartość opóźnienia w ms zdefiniować jako stałą (dyrektywa EQU) a następnie umieścić ją w rejestrze R i dokonać przeliczenia na potrzebną liczbę cykli. Wartość z tego rejestru wykorzystać jako licznik pętli.

Zapoznać się z dyrektywami CSEG, DSEG oraz ORG. Zdefiniować segment kodu programu oraz ustawić początek programu na adres 0x0 a następnie jako pierwszą instrukcję wykonać skok do adresu znajdującego się poza tablicą wektorów przerwań (mapa pamięci w nocie katalogowej mikrokontrolera, rozdział 11). Skorzystać z przykładu z noty katalogowej.

Zmodyfikować strukturę programu w taki sposób by pod główną etykietą programu znajdowały się jedynie skoki do procedur. Wykorzystać instrukcję skoku z zachowaniem adresu powrotu. W procedurze wykorzystać instrukcję powrotu z podprocedury. Skonfigurować stos wykorzystując kod umieszczony w karcie katalogowej (strona 46 lub poniżej)

```
.cseg
.org    0x0
        jmp     RESET                   ; Reset Handler
.org    0x30
RESET:
        ldi     r16,high(RAMEND)                ; konfiguracja stosu
        out     SPH,r16                         ; konfiguracja stosu
        ldi     r16,low(RAMEND)                 ; konfiguracja stosu
        out     SPL,r16                         ; konfiguracja stosu
        rcall   konfiguracja_portów
LOOP:
        rcall   zapal_led
        rcall   delay
        rcall   zgas_led
        rjmp    LOOP
```

Język: Asembler

Zasoby: port C, D

Weryfikacja: Przy użyciu Atmel Studio/Microchip Studio lub AVR Simulator

N – liczba przypisana do osoby, podana przez prowadzącego

Ćwiczenie 3

Temat: Obsługa przycisku

Zadanie: Zmodyfikować program z ćwiczenia 2. Dodać obsługę jednego z przycisków (port B piny 0-3). Sprawdzić na schemacie elektrycznym płytki jak podłączony jest przycisk czyli jaki stan logiczny zostaje wymuszony w momencie wciśnięcia przycisku. Sprawdzić czy konieczne jest użycie rezystorów podciągających. Wciśnięcie przycisku powinno zapalić jedną z diod na stałe a pozostałe zgasić. Sekwencja zapalania diod ma zostać wstrzymana ale powinna zostać przywrócona po zwolnieniu przycisku.

Sprawdzić czy program reaguje na przycisk niezwłocznie (debugując program instrukcja po instrukcji). W razie potrzeby zmodyfikować program.

Uwaga: Wykorzystanie przerwań od przycisku nie jest konieczne. W przypadku wykorzystania przerwań należy sprawdzić podłączenie microswitchów oraz które wyprowadzenie umożliwia przerwanie zewnętrzne (External interrupt jako dodatkowa funkcja portu I/O)

Język: Asembler
Zasoby: port C, D wyjście, port B wejście
Weryfikacja: Przy użyciu Atmel Studio/Microchip Studio lub AVR Simulator

Ćwiczenie 4

Temat: Obsługa portów i pętla opóźniająca w języku C

Zadanie: Przepisać program z ćwiczenia 3 na język C.

Język: C
Zasoby: port C,D
Weryfikacja: Przy użyciu Atmel Studio/Microchip Studio lub AVR Simulator

Ćwiczenie 5

Temat: Maszyna deszyfrująca

Zadanie: 1.Napisać funkcję zapisującą ciąg znaków do pamięci EEPROM. Funkcja powinna przyjmować 3 argumenty. Wskazówka: można wykorzystać gotowy przykład zapisu do pamięci EEPROM z dokumentacji dla mikrokontrolera Atmega32A i zmodyfikować tak, by funkcja przyjmowała 3 argumenty.
2. Zapoznać się z działaniem szyfru Vigenere'a. Wejściowy (zaszyfrowany) ciąg znaków należy umieścić w pamięci EEPROM pod adresem 0x00. Klucz należy skopiować do EEPROMu pod adres 0x20. Wskazówka: do otrzymania zakodowanego wejściowego ciągu znaków można wykorzystać generatory szyfru Vigenere'a dostępne na stronach internetowych np. https://www.dcode.fr/vigenere-cipher.
3. Zaimplementować algorytm deszyfrujący w mikrokontrolerze Algorytm powinien zapisywać odkodowany ciąg znaków do pamięci EEPROM pod adres 0x40.
Wskazówka: w przypadku gdy klucz jest krótszy od wejściowego ciągu znaków należy go odpowiednio „wydłużyć", powielając znaki klucza do momentu gdy długość klucza będzie równa długości wejściowego ciągu znaków.
4. Sprawdzić działanie algorytmu deszyfrującego na symulatorze.
5. Wprowadzić do EEPROMU przykładowy ciąg znaków do deszyfracji oraz klucz. Sprawdzić poprawność działania algorytmu poprzez porównanie z dostępnymi w internecie algorytmami np.: https://www.dcode.fr/vigenere-cipher
6. ~~Poprawność działania algorytmu będzie sprawdzana na następnych zajęciach za pomocą wiadomości wielokrotnie zaszyfrowanej (odpowiadająca liczbie osób). Każda deszyfracja będzie wymagała innego klucza podanego na zajęciach.~~

Język: C
Zasoby: pamięć EEPROM
Weryfikacja: Przy użyciu Atmel Studio/Microchip Studio lub AVR Simulator

Ćwiczenie 6

Temat: Cykliczne pobieranie próbek z przetwornika A/C

Napisać program, który po uruchomieniu pobiera NB_OF_SAMPLES próbek z przetwornika A/C z częstotliwością SAMPLE_FREQ_HZ (okres odczytu równy 500+N*50 ms). Należy skorzystać ze schematu programu podanego poniżej. Ewentualnie można skorzystać z bibliotek Prycon (dostępne na stronie prowadzącego).
Schemat programu:

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define PRESCALER 1024
#define F_CPU 16000000
#define NB_OF_SAMPLES 10
#define SAMPLE_FREQ_HZ xxx       //wpisać zadaną częstotliwość
                                 //przedrostek jednostki dowolny

uint16_t tab[NB_OF_SAMPLES];

// dodać funkcje obsługi przerwania od timerów

void InitADC()
{
     //… uzupełnij
}

uint16_t StartADC(uint8_t ADCchannel)
{
     //… uzupełnij
}

uint16_t ReadADC(uint8_t ADCchannel)
{
     //… uzupełnij
     return ADC; //tutaj umieść breakpoint
}

void InitTimer1(uint16_t freq)
{
     //… uzupełnij
}

void InitTimer2(uint16_t freq)
```

```
{
	//… uzupełnij
}

int main(void)
{
	InitADC();
	InitTimer1(SAMPLE_FREQ_HZ);
	sei();
	while (1)
	{
		//… uzupełnij
	}
}
```

1. Należy skorzystać z przerwań od 16-bitowego timera, skonfigurowanego w tryb pracy Output Compare (CTC). Cała konfiguracja powinna znaleźć się w funkcji InitTimer(). Uwaga: częstotliwość przerwań od timera powinna zostać podana jako argument funkcji InitTimer(). Wskazówka: do odpowiednich obliczeń wykorzystać wzór z sekcji 16.8.2 z instrukcji do mikrokontrolera ATmega32
2. Odpowiednio skonfigurować przetwornik A/C w funkcji InitADC(). Preskaler należy ustawić na maksymalną wartość, a jako źródło referencyjne wybrać napięcie zasilania. Wykorzystać kanał ADC, do którego podłączony jest potencjometr.
3. Po każdym przerwaniu od timera1 należy uruchomić konwersję w funkcji StartADC(). Jako argument funkcji należy podać numer kanału przetwornika. Do odczytu danej z przetwornika należy użyć przerwania timera2. Timer2 powinien zostać uruchomiony w momencie gdy mamy pewność, że minął czas potrzebny na dokonanie konwersji przez ADC (wskazane jest zachowanie dodatkowego bufora czasowego). Wskazówka: w nocie katalogowej znaleźć informację dot. liczby cykli potrzebnych na dokonanie konwersji przez ADC i wykorzystać „niedogodność" związaną z czasem pierwszej konwersji. ~~Uwaga: na ostatniej linii funkcji ReadADC() należy ustawić breakpoint: posłuży on do ręcznego wpisywania wartości do rejestru ADC.~~
4. Odczytane próbki wpisywać do tablicy tab[]. Po zapisaniu odpowiedniej liczby próbek, kolejne odczyty zignorować. Ustawić pułapkę, która wykona się po zapisaniu zadanej liczby próbek. Sprawdzić poprawność działania programu w debugerze podglądając odczytane wartości w pamięci RAM.

Język: C
Zasoby: przetwornik ADC, timer
Weryfikacja: Przy użyciu Atmel Studio/Microchip Studio lub AVR Simulator

Ćwiczenie 7

Temat: Obsługa wyświetlacza alfanumerycznego

Zadanie: Zmodyfikować program z ćwiczenia 6 dodając obsługę wyświetlacza alfanumerycznego. Na wyświetlaczy wyświetlić wartość odczytaną z przetwornika (w postaci napięcia lub wartości bitowej). Skorzystać z biblioteki obsługi kontrolera wyświetlacza HD44780.

Język: C
Zasoby: przetwornik ADC, timer, wyświetlacz
Weryfikacja: Przy użyciu Atmel Studio/Microchip Studio lub AVR Simulator

N – liczba przypisana do osoby, podana przez prowadzącego

Ćwiczenie 8

Temat: Obsługa timera w trybie PWM

Zadanie: Zmodyfikować program z ćwiczenia 7 dodając obsługę timera w trybie PWM. Wygenerować sygnał PWM o współczynniku wypełnienia zależnym od położenia potencjometru czyli D = wartość_ADC / 1024. Wyprowadzić sygnał na diodę LED podłączoną na porcie PD4 lub PD5 i obserwować zmianę jasności diody zależnie od współczynnika wypełnienia. Dostosować częstotliwość sygnału PWM tak aby niezauważalne było miganie diody. Na wyświetlaczu wyświetlić wartość współczynnika D.

Język: C
Zasoby: port B, port D, timer
Weryfikacja: Przy użyciu Atmel Studio/Microchip Studio lub AVR Simulator

N – liczba przypisana do osoby, podana przez prowadzącego

Ćwiczenie 9

Temat: Komunikacja SPI

Napisać program, który na wyświetlaczu alfanumerycznym wyświetla wartość temperatury z termometru cyfrowego wykorzystującego komunikację SPI. Odczyt temperatury należy realizować cyklicznie z ustaloną częstotliwością z wykorzystaniem przerwań. Sposób realizacji dowolny. Niedozwolone jest użycie funkcji opóźniających. Wykorzystać bibliotekę do obsługi protokołu SPI.

Język: C
Zasoby: termometr cyfrowy, timer, wyświetlacz
Weryfikacja: Przy użyciu symulatora wbudowanego w Atmel Studio lub AVR Simulator

Ćwiczenie 10

Temat: Tryb uśpienia

Zmodyfikować program z ćwiczenia 9 poprzez dodanie trybu uśpienia, w który mikrokontroler wchodzi cyklicznie. Należy wykorzystać odpowiedni tryb uśpienia, który umożliwia wybudzenie poprzez wykorzystywane przerwanie. Wykorzystać bibliotekę *sleep.h*.

Język: C
Zasoby: termometr cyfrowy, timer, wyświetlacz, tryb uśpienia
Weryfikacja: Przy użyciu symulatora wbudowanego w Atmel Studio lub AVR Simulator

Ćwiczenie 11

Temat: Obsługa licznika WatchDog

Napisać program, który po uruchomieniu konfiguruje licznik WDT w tryb Interrupt and System Reset Mode z czasem przepełnienia ok. 200+N*100 mikrosekund. Wykorzystać jeden z rejestrów ogólnego przeznaczenia do zliczania liczby wystąpień przerwań od WDT. Drugi z rejestrów wykorzystać do odczytania wartości z rejestru MCUSR, w którym przechowywana jest informacja o źródle poprzedniego resetu. Odczyt ten należy wykonać tylko na początku programu. Wprowadzić mikrokontroler w jeden stanów uśpienia (bez znaczenia, który gdyż przerwanie od WDT wybudza mikrokontroler z każdego trybu uśpienia) a po każdym wystąpieniu przerwania wprowadzać mikrokontroler w tryb uśpienia ponownie. Zaimplementować warunek, w którym po czwartym przerwaniu od WDT, mikrokontroler nie wchodzi w tryb uśpienia. Należy program zapętlić aby nie zawiesić mikrokontrolera i poczekać na przepełnienie WDT, które powinno zresetować mikrokontroler. Po resecie sprawdzić czy w

rejestrze przechowującym wartość rejestru MCUSR znajduje się informacja o resecie, którego źródłem był WDT.
Wskazówka. Należy dokładnie zapoznać się z rozdziałem 12.4 i 12.5 z noty katalogowej ze szczególnym uwzględnieniem rysunku 12.7. Zwrócić uwagę w jaki sposób jest generowany sygnał przerwania oraz resetu w trybie Interrupt and System Reset Mode. Sprawdzić w jaki sposób uniemożliwia się wystąpienie resetu po wywołaniu przerwania w przypadku prawidłowo działającego programu. Wykorzystać to do celowego umożliwienia przepełnienia WDT po czwartym wystąpieniu przerwania od WDT.

Język: C
Zasoby: przetwornik ADC, timer, watchdog
Weryfikacja: Przy użyciu symulatora wbudowanego w Atmel Studio lub AVR Simulator

Ćwiczenie 12

Temat: Obsługa transmisji szeregowej USART w trybie SPI

Napisać program, który pobiera wartości z pamięci EPROM (jedna ramka 8 bitowa), wysyła na port szeregowy i jednocześnie odbiera z tego samego portu dane oraz nadpisuje pamięć EEPROM. Skonfigurować transmisję szeregową USART w trybie SPI. Ustalić częstotliwość na wartość 10000+N*1000bps. Wykorzystać funkcje z ćwiczenia 5 oraz funkcje konfiguracji i transmisji szeregowej z noty katalogowej. Zweryfikować program poprzez ręczne ustawianie wartości na odpowiadającej linii Rx i równoczesny odczyt na linii Tx.

Język: C
Zasoby: EEPROM, USART
Weryfikacja: Przy użyciu symulatora wbudowanego w Atmel Studio lub AVR Simulator
N – liczba przypisana do osoby, podana przez prowadzącego