Systemy mikroprocesorowe

Użyteczne opcje:

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

Ćwiczenie 1

Temat: Obsługa rejestrów i portów mikrokontrolera – operacje podstawowe

Zadanie: Na porcie D ustawić na wyjściu następującą wartość b11110101. Następnie zmieniać 4 młodsze bity w nieskończonej pętli naprzemiennie na wartości bxxxx1010 oraz bxxxx0101. (x oznacza poprzednią wartość, która nie może ulec zmianie). Należy wykorzystać maskowanie wartości odczytywanej z portu.

Język: Asembler
Zasoby: port D
Weryfikacja: Przy użyciu symulatora wbudowanego w Atmel Studio lub AVR Simulator

Wskazówka: możliwe jest wykonanie zadania za pomocą jednej instrukcji w pętli (funkcja „Toggle pin" >> karta katalogowa)

Ćwiczenie 2

Temat: Pętla opóźniająca

Zadanie: Do portu D podłączone są diody LED w konfiguracji wspólna katoda (*). Zapalać po jednej diodzie po kolei w pętli opóźniającej. Opóźnienie ustalić na wartość (500+N*50) µs. Częstotliwość taktowania ustalić na 16MHz. Opóźnienie obliczyć biorąc pod uwagę liczbę cykli potrzebnych na wykonanie jednej iteracji pętli.

Język: Asembler
Zasoby: port D
Weryfikacja: Przy użyciu symulatora wbudowanego w Atmel Studio lub AVR Simulator
(*) zapalenie diody wymaga ustawienia stanu wysokiego na pinie wyjściowym mikrokontrolera
N – liczba przypisana do osoby, podana przez prowadzącego

Ćwiczenie 3

Temat: Obsługa przycisku

Zadanie: Zmodyfikować program z ćwiczenia 2. Dodać obsługę przycisku, który podłączony jest do pinu 6 portu B. Wciśnięty przycisk zwiera pin do masy. Ponieważ brak wciśnięcia nie powoduje zwarcia do żadnego potencjału (stan nieustalony) należy włączyć rezystory podciągające do zasilania (pull-up resistor). Wciśnięcie przycisku powinno zapalać diodę podłączoną do portu D na pinie 6. Pozostałe diody mają nie zmienić swojego stanu a sekwencja zapalania diod ma zostać wstrzymana. Po zwolnieniu przycisku sekwencja ma zostać uruchomiona ponownie od stanu sprzed wciśnięcia przycisku. Wciśnięcie przycisku należy zrealizować poprzez ręczne ustawienie stanu na odpowiadającym bicie w oknie rejestrów symulatora.
Sprawdzić czy program reaguje na przycisk niezwłocznie (debugując program instrukcja po instrukcji). W razie potrzeby zmodyfikować program.

Język: Asembler
Zasoby: port D wyjście, port B wejście
Weryfikacja: Przy użyciu symulatora wbudowanego w Atmel Studio lub AVR Simulator

Ćwiczenie 4

Temat: Obsługa przerwań oraz pamięci RAM

Zadanie: Zmodyfikować program z ćwiczenia 3. Do odmierzania czasu użyć któregokolwiek z wbudowanych liczników (Timer). Funkcję opóźniającą z ćwiczenia 3 **należy pozostawić** w kodzie programu do ewentualnego sprawdzenia. Wykrywanie przepełnienia licznika ma odbywać się za pomocą przerwania. Sprawdzanie przycisku powinno pozostać bez zmian czyli w trybie poolingu. Wszystkie niezbędne operacje związane z wyświetlaniem sekwencji powinny być wywoływane w funkcji obsługującej przerwanie jako skoki do funkcji. W pętli głównej programu może znajdować się wyłącznie funkcja do obsługi przycisku.

Zarezerwować miejsce w pamięci RAM na zmienną, w której przechowywana będzie wartość opóźnienia w µs. Wprowadzić wartość tego opóźnienia do pamięci RAM na początku programu a następnie wykorzystać ją do obliczenia wartości niezbędnej do konfiguracji licznika.

Język: Asembler
Zasoby: port D wyjście, port B wejście, timer
Weryfikacja: Przy użyciu symulatora wbudowanego w Atmel Studio lub AVR Simulator

Ćwiczenie 5

Temat: Maszyna deszyfrująca

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 Atmega2560.
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.
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 kluczabędzie 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 symulatora wbudowanego w Atmel 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. Należy skorzystać ze schematu programu podanego poniżej.

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 10000

uint16_t tab[NB_OF_SAMPLES];

void InitADC()
{
	//… uzupelnij
}

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

uint16_t ReadADC(uint8_t ADCchannel)
{
	//… uzupelnij
	return ADC; //tutaj umiesc breakpoint
}

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

void InitTimer2(uint16_t freq)
{
	//… uzupelnij
}

int main(void)
{
```

```
    InitADC();
    InitTimer1(SAMPLE_FREQ_HZ);
    sei();
    while (1)
    {
        //… uzupelnij
    }
}
```

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 17.9.2 z instrukcji do mikrokontrolera ATmega2560.
2. Odpowiednio skonfigurować przetwornik A/C w funkcji InitADC(). Preskaler należy ustawić na maksymalną wartość, a jako źródło referencyjne wybrać napięcie zasilania.
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 przed 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[]. Sprawdzić poprawność działania programu w symulatorze podglądając odczytane wartości w pamięci RAM. Po zapisaniu odpowiedniej liczby próbek, kolejne odczyty można zignorować.

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

Ćwiczenie 7

Temat: Tryb uśpienia

Zmodyfikować program z ćwiczenia 2, który w momencie oczekiwania na przerwanie przechodzi w tryb uśpienia. Należy wykorzystać odpowiedni tryb uśpienia, który umożliwia wybudzenie poprzez przerwanie z Timera oraz ADC. Należy dodać obsługę przerwania od ADC (sygnalizujące zakończenie konwersji) eliminując wykorzystanie drugiego timera do odczytu wyniku konwersji. Należy sprawdzić, przerwanie od którego Timera umożliwia wybudzenie ze stanu uśpienia. Do sprawdzenia działania programu wykorzystać wytyczne do ćwiczenia 6.

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

Ćwiczenie 8

Temat: Obsługa licznika WatchDog

Napisać program, który po uruchomieniu konfiguruje licznik WDT w tryb Interrupt and System Reset Mode z czasem przepełnienia ok. 2 sekund. 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 8

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