Systemy wbudowane

Wytyczne i zadania do zaliczenia sprawozdania

1. Wytyczne ogólne

- Każdy przygotowuje sprawozdanie samodzielnie,
- W sprawozdaniu należy ująć:
  - krótki opis środowiska pracy,
  - krótki opis każdego ze zrealizowanych zadań,
  - problemy i trudności, które wynikły w trakcie implementacji
  - źródła, które zostały wykorzystane
  - do każdego zadania kod, z komentarzami (po polsku) linijka po linijce, części kodu, która realizowała dane zadanie
- nie ma potrzeby opisywania kodu, który został przekazany przez prowadzącego
- Sprawozdania należy przesłać w formie pliku PDF a pliki z kodami źródłowymi oraz plikami *.hex proszę dostarczyć spakowane w archiwum ZIP (2 załączniki, proszę nie dołączać plików nagłówkowych) na adres: adam.fron@uwm.edu.pl.
- Sprawozdania będą drobiazgowo sprawdzane pod kątem plagiatów, proszę też przygotować się na ewentualne udzielenie odpowiedzi na kilka pytań do sprawozdania
- Ostateczny termin oddania sprawozdania to piątek 07.06.2024. Wcześniejsze oddanie sprawozdania może jednak poskutkować jakimś bonusem do oceny.

2. Realizacja zadań

- Zadania są realizowane w języku C na platformie Explorer 16/32 z CPU PIC24FJ128GA010 z wykorzystaniem środowiska MPLAB X IDE
- Każde ze sformułowanych zadań jest punktem wyjścia – inwencja i własne modyfikacje są mile widziane, ale nie mogą ograniczać zadanej funkcjonalności
- Zadania w sali mogą być realizowane w dowolnej kolejności (ale już w sprawozdaniu proszę je umieszczać według numeracji)
- W momencie uznania, że dane zadanie zostało wykonane, proszę ten fakt zgłaszać prowadzącemu

3. Zawartość sprawozdania

- Strona tytułowa
- Spis treści
- Opisy środowiska pracy, wykorzystanego sprzętu i oprogramowania
- Treść zadania 1 i jego opracowanie
- …
- Treść zadania 5 i jego opracowanie

4. Jakiej oceny mogę się spodziewać?

- Ocena wyjściowa to 4. Uzyskanie takiej oceny wymaga zrealizowania wszystkich wymaganych zadań (zad. 1 + 3 wybrane) w podstawowej wersji i złożenia sprawozdania poprawnego pod względem językowym i estetycznym, w którym być może są pojedyncze błędy, ale nie utrudniają one lektury i – co najważniejsze – zrozumienia napisanego kodu.
- Ocenę można obniżyć poprzez:
  - niezrealizowanie części zadań,
  - poważne zbrodnie na języku,
  - nieczytelność,

- oddanie sprawozdania po ostatecznym terminie
- Analogicznie, na plus do oceny działa:
  - poprawne zrealizowanie wszystkich zadań i kreatywne podejście do postawionych problemów
  - logiczna i estetyczna struktura pracy
  - wykonanie pięciu zadań zamiast czterech
  - oddanie sprawozdania przed terminem

5. No dobra, ale jak to wpłynie na ocenę na koniec semestru?

Ostateczna ocena będzie zależała w 75% od sprawozdania i 25% od prezentacji. Ewentualne wątpliwości będą rozwiewane w formie krótkiego kolokwium ustnego z indywidualnymi pytaniami.

6. Ok, to jakie są te zadania?

Do wykonania zadanie 1. i wybrane trzy z czterech pozostałych.

Zadanie 1. Program przełączający cyklicznie 6 podprogramów (następny program po 6 to 1, a poprzedni względem 1 to 6). Do przełączania w przód i tył wykorzystujemy przyciski, wynik wyświetlamy na diodach LED.
Podprogramy:
1. 8 bitowy licznik binarny zliczający w górę (0…255)
2. 8 bitowy licznik binarny zliczający w dół (255…0)
3. 8 bitowy licznik w kodzie Graya zliczający w górę (repr. 0…255)
4. 8 bitowy licznik w kodzie Graya zliczający w dół (repr. 255…0)
5. 2x4 bitowy licznik w kodzie BCD zliczający w górę (0…99)
6. 2x4 bitowy licznik w kodzie BCD zliczający w dół (99…0)

Zadanie 2. Alarm! Źródło danych to potencjometr. Nastawę – wartość po przekroczeniu której alarm załącza się – należy ustawić na połowę zakresu potencjometru. Po przekroczeniu wartości nastawy – przez pierwsze 5 sekund mruga jedna dioda, a potem zapalone są wszystkie. Jeśli alarm jest załączony (mruganie, zapalone wszystkie diody) – to po ustawieniu potencjometru poniżej nastawy alarmowej – alarm należy wyłączyć (przerwać mruganie, zgasić diody). Wyłączenie alarmu – wybrany przycisk.

Zadanie 3. Reklama! Temat dowolny, animacje dowolne. Warto tu skorzystać z opcji tworzenia własnych znaków do wyświetlania na ekranie LCD.

Zadanie 4. Kontroler kuchenki mikrofalowej z wyświetlaniem parametrów na ekranie LCD. Do potencjometrów i przycisków należy przypisać: wybór mocy, dodanie czasu, start/stop, reset

Zadanie 5. Zegar szachowy dla dwóch graczy. Naciśnięcie przycisku pierwszego gracza powoduje odmierzanie czasu drugiego gracza od ostatniego zapamiętanego stanu, a naciśnięcie przycisku drugiego gracza – odmierzanie czasu pierwszego. Gra toczy się do końca czasu albo: poddania partii, ustalenia remisu, mata – to są jednak zdarzenia, których nie odnotowujemy na zegarze). Zegar ma wyświetlać czas obu graczy i w razie gdy któryś z graczy nie wykona ruchu w pozostałym mu czasie – wyświetlić komunikat, że gracz przegrał grę przez czas.

7. No dobra, ale jak coś tu w ogóle zrobić?

Założenie projektu

1) Uruchamiamy MPLAB X IDE
2) File ->New Project

3) Wybieramy rodzaj projektu: „Microchip Embedded” -> „Application Project” (w starszych wersjach „Standalone Project”)
4) Select Device: wybieramy CPU. W trzecim oknie „no tool”
5) Select Header: None.
6) Select Tool: XC16 (musi być wyświetlona ścieżka instalacji kompilatora)
7) Select Project Name and Folder: nazywamy projekt i wskazujemy jego lokalizację na komputerze (odznaczamy opcję uruchomienia MCC)

Pierwszy projekt

Skorzystamy z przykładu dostarczonego przez Microchip:

```
// PIC24FJ128GA010 Configuration Bit Settings
// For more on Configuration Bits, see Section 1.1
// consult your device data sheet
// CONFIG2
#pragma config POSCMOD = XT // XT Oscillator mode selected
#pragma config OSCIOFNC = ON // OSC2/CLKO/RC15 as port I/O (RC15)
#pragma config FCKSM = CSDCMD // Clock Switching and Monitor disabled
#pragma config FNOSC = PRI // Primary Oscillator (XT, HS, EC)
#pragma config IESO = ON // Int Ext Switch Over Mode enabled
// CONFIG1
#pragma config WDTPS = PS32768 // Watchdog Timer Postscaler (1:32,768)
#pragma config FWPSA = PR128 // WDT Prescaler (1:128)
#pragma config WINDIS = ON // Watchdog Timer Window Mode disabled
#pragma config FWDTEN = OFF // Watchdog Timer disabled
#pragma config ICS = PGx2 // Emulator/debugger uses EMUC2/EMUD2
#pragma config GWRP = OFF // Writes to program memory allowed
#pragma config GCP = OFF // Code protection is disabled
#pragma config JTAGEN = OFF // JTAG port is disabled
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.
#include <xc.h>
int main(void) {
        unsigned char portValue = 0x55;
        // Port A access
        AD1PCFG = 0xFFFF; // set to digital I/O (not analog)
        TRISA = 0x0000; // set all port bits to be output
        LATA = portValue; // write to port latch
        return 0;
}
```

Żeby to zrobić należy w drzewie projektu kliknąć prawym na „Source Files” i stworzyć nowy plik C Main File (nazwa dowolna).

Wklejamy kod do pliku.

Żeby wgrać kod do urządzenia najpierw wybieramy z paska narzędzi opcję „Build For Debugging Main Project”, a następnie „Make and Program Device Main Project”

Drugi przykład od Microchip:

```
// PIC24FJ128GA010 Configuration Bit Settings
// For more on Configuration Bits, see Section 1.1
// consult your device data sheet
// CONFIG2
#pragma config POSCMOD = XT // XT Oscillator mode selected
#pragma config OSCIOFNC = ON // OSC2/CLKO/RC15 as port I/O (RC15)
#pragma config FCKSM = CSDCMD // Clock Switching and Monitor disabled
#pragma config FNOSC = PRI // Primary Oscillator (XT, HS, EC)
#pragma config IESO = ON // Int Ext Switch Over Mode enabled
// CONFIG1
#pragma config WDTPS = PS32768 // Watchdog Timer Postscaler (1:32,768)
#pragma config FWPSA = PR128 // WDT Prescaler (1:128)
#pragma config WINDIS = ON // Watchdog Timer Window Mode disabled
#pragma config FWDTEN = OFF // Watchdog Timer disabled
#pragma config ICS = PGx2 // Emulator/debugger uses EMUC2/EMUD2
#pragma config GWRP = OFF // Writes to program memory allowed
#pragma config GCP = OFF // Code protection is disabled
```

```
#pragma config JTAGEN = OFF // JTAG port is disabled
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <libpic30.h>
int main(void) {
        unsigned char portValue;
        // Port A access
        AD1PCFG = 0xFFFF; // set to digital I/O (not analog)
        TRISA = 0x0000; // set all port bits to be output
        while(1) {
                portValue = 0x55;
                LATA = portValue; // write to port latch
                // delay value change
                __delay32(1500000); // delay in instruction cycles
                portValue = 0xAA;
                LATA = portValue; // write to port latch
                __delay32(1500000); // delay in instruction cycles
        }
        return -1;
}
```

Do bardziej zaawansowanych projektów potrzebne będą biblioteki, które są dostępne w przesłanym archiwum (dostarczone przez Microchip). Np. skorzystanie z ekranu wymaga wgrania plików lcd.c i lcd.h do folderu, w którym zapisany jest projekt. Następnie w drzewie projektu w MPLAB klikamy prawym na Header Files -> Add Existing Item i wybieramy plik .h. Analogicznie do Source Files ładujemy plik .c.