**Systemy Wbudowane**

Sprawozdanie do zadań 1-5

**Sprawozdanie przygotował:**

Michał Chomczyk 162385

Zadania 1 i 2 zostały wykonane zespołowo przez: Michał Chomczk, Tymoteusz Łuczko, Piotr Dondalski, Paweł Deptuła, Igor Ozga ,Sebastian Celoch.

Zadanie 3 zostało wykonane samodzielnie przez Michała Chomczyka.

Spis treści

[Zadanie 1 2](#_Toc135308488)

[Opis zadania: 2](#_Toc135308489)

[Problemy podczas implementacji: 2](#_Toc135308490)

[Kod: 2](#_Toc135308491)

[Zadanie 2 7](#_Toc135308492)

[Opis zadania: 7](#_Toc135308493)

[Problemy podczas implementacji: 7](#_Toc135308494)

[Kod: 7](#_Toc135308495)

[Zadanie 3 11](#_Toc135308496)

[Opis zadania: 11](#_Toc135308497)

[Problemy podczas implementacji: 11](#_Toc135308498)

[Kod: 11](#_Toc135308499)

[Zadanie 4 16](#_Toc135308500)

[Opis zadania: 16](#_Toc135308501)

[Problemy podczas implementacji: 17](#_Toc135308502)

[Kod: 17](#_Toc135308503)

[Zadanie 5 23](#_Toc135308504)

[Opis zadania: 23](#_Toc135308505)

[Problemy podczas implementacji: 23](#_Toc135308506)

[Kod: 23](#_Toc135308507)

# Zadanie 1

## Opis zadania:

Program przełączający cyklicznie 9 podprogramów (następny program po 9 to 1, a poprzedni względem 1 to 9).

Przycisk S3–następny programPrzyciskS6–poprzedni program

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)

7.3 bitowy wężyk poruszający się lewo-prawo8.Kolejka

9.6 bitowy generator liczb pseudolosowych oparty o konfigurację 1110011

## Problemy podczas implementacji:

Największy problem sprawiło zaimplementowanie switcha pozwalającego na przełączanie się między podprogramami.

## Kod:

#include <stdio.h>

#include <stdlib.h>

#include <stddef.h>

#include <stdbool.h>

#include <xc.h>

#include "app.h"

#define ONE\_VOLT 310

#define ONE\_TENTH\_VOLT 31

#define ONE\_HUNDREDTH\_VOLT 3

void Update\_LCD **(** void **)** **;**

void SYS\_Initialize **(** void **)** **;**

extern void ConvertADCVoltage **(** unsigned int **)** **;**

extern void Hex2Dec **(** unsigned char **)** **;**

APP\_DATA appData **=** **{**

**.**messageLine1 **=** "Explorer 16 Demo" **,**

**.**messageLine2 **=** "Press S3 to cont" **,**

**.**messageTime **=** "Time 00: 00: 00 " **,**

**.**messageADC **=** " Pot = 0.00 Vdc "

**}** **;**

void delay**(**unsigned int ms**)** **{**

unsigned int i**;**

unsigned char j**;**

**for** **(**i **=** 0**;** i **<** ms**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** 200**;** j**++)** **{**

Nop**();**

Nop**();**

Nop**();**

Nop**();**

Nop**();**

**}**

**}**

**}**

int grey**(**int value**)** **{**

**return** value **^** **(**value **>>** 1**);**

**}**

int bcd**(**int value**)** **{**

**return** **((**value**/**10**)** **<<** 4 **|** **(**value**%**10**));**

**}**

void swap**(**int**\*** selected\_subfunction**)** **{**

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **))** **{**

**(\***selected\_subfunction**)** **+=** 1**;**

**if** **((\***selected\_subfunction**)** **==** 10**)** **{**

**(\***selected\_subfunction**)** **=** 1**;**

**}**

**}**

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

**(\***selected\_subfunction**)** **-=** 1**;**

**if** **((\***selected\_subfunction**)** **==** 0**){**

**(\***selected\_subfunction**)** **=** 9**;**

**}**

**}**

**}**

unsigned int xorshift32**(**unsigned long seed**)** **{**

int config **=** 115**;**

unsigned long x **=** seed**;**

x **\*=** config**;**

x **^=** x **<<** 13**;**

x **^=** x **<<** 5**;**

**return** x**;**

**}**

unsigned long generator**(**unsigned long seed**)** **{**

int config **=** 115**;**

unsigned long x **=** seed**;**

x **\*=** config**;**

x **^=** x **<<** 13**;**

x **^=** x **<<** 5**;**

**return** x**;**

**}**

int main **(** void **)**

**{**

SYS\_Initialize **(** **)** **;**

TIMER\_SetConfiguration **(** TIMER\_CONFIGURATION\_RTCC **)** **;**

ADC\_SetConfiguration **(** ADC\_CONFIGURATION\_AUTO\_SAMPLE\_CONVERT **)** **;**

unsigned char later **=** 0**;**

int running\_prog **=** 1**;**

unsigned long seed **=** 21337**;**

**while** **(** 1 **)**

**{**

// Zadanie 1

later **=** 0**;**

**while(**running\_prog **==** 1**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d "**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

LATA **=** later**;**

later**++;**

delay**(**500**);**

**if** **(**later **==** 255**)** **{**

later **=** 0**;** // reset

**}**

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

// Zad 2

later **=** 255**;**

**while(**running\_prog **==** 2**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d "**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

LATA **=** later**;**

later**--;**

delay**(**500**);**

**if** **(**later **==** 0**)** **{**

later **=** 255**;** // reset

**}**

delay**(**500**);** // czas na zmiane przyc

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

// Zad 3

later **=** 0**;**

**while(**running\_prog **==** 3**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d"**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

LATA **=** grey**(**later**);**

later**++;**

delay**(**500**);**

**if** **(**later **==** 255**)** **{**

later **=** 0**;** // reset

**}**

delay**(**500**);** // czas na zmiane przyc

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

// Zad 4

later **=** 255**;**

**while(**running\_prog **==** 4**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d"**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

LATA **=** grey**(**later**);**

later**--;**

delay**(**500**);**

**if** **(**later **==** 0**)** **{**

later **=** 255**;** // reset

**}**

delay**(**500**);** // czas na zmiane przyc

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

// z5

later **=** 0**;**

**while(**running\_prog **==** 5**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d"**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

LATA **=** bcd**(**later**);**

later**++;**

delay**(**500**);**

**if** **(**later **==** 99**)** **{**

later **=** 0**;**

**}**

delay**(**500**);** // czas na zmiane przyc

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

// z6

later **=** 99**;**

**while(**running\_prog **==** 6**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d"**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

LATA **=** bcd**(**later**);**

later**--;**

delay**(**500**);**

**if** **(**later **==** 0**)** **{**

later **=** 99**;**

**}**

delay**(**500**);** // czas na zmiane przyc

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

// z7

later **=** 7**;**

**while(**running\_prog **==** 7**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d"**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

int i **=** 0**;**

**for** **(**i **=** 0**;** i **<** 5**;** i**++){**

LATA **=** later**;**

later **<<=** 1**;**

delay**(**500**);**

**}**

**for** **(**i **=** 0**;** i **<** 5**;** i**++){**

LATA **=** later**;**

later **>>=** 1**;**

delay**(**500**);**

**}**

delay**(**500**);** // czas na zmiane przyc

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

// z8

int ledy\_turned\_on **=** 7**;**

int leds\_on **=** 1**;**

int leds\_queued **=** 0**;**

**while(**running\_prog **==** 8**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d"**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

**for(**int i **=** 0**;** i **<** ledy\_turned\_on**;** i**++){**

LATA **=** leds\_on **+** leds\_queued**;**

leds\_on **<<=** 1**;**

delay**(**300**);**

**}**

leds\_queued **+=** leds\_on**;**

LATA **=** leds\_queued**;**

delay**(**500**);**

leds\_on **=** 1**;**

ledy\_turned\_on**--;**

**if(**ledy\_turned\_on **==** **-**1**){**

ledy\_turned\_on **=** 7**;**

leds\_queued **=** 0**;**

**}**

delay**(**500**);** // czas na zmiane przyc

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

// z9

**while(**running\_prog **==** 9**){**

LCD\_ClearScreen**();**

char str**[**20**];**

sprintf**(**str**,** "program %d"**,** running\_prog**);**

LCD\_PutString**(**str**,** **sizeof(**str**));**

delay**(**500**);**

seed **=** generator**(**seed**);**

int result **=** **(**int**)** **(**seed **%** 64**);**

LATA **=** result**;**

delay**(**500**);** // czas na zmiane przyc

**if** **(**BUTTON\_IsPressed**(** BUTTON\_S3 **)** **||** BUTTON\_IsPressed**(** BUTTON\_S6 **))** **{**

swap**(&**running\_prog**);**

**}**

**}**

**}**

**}**

# Zadanie 2

## Opis zadania:

W oparciu o program przykładowy oraz dokumentację czujnika temperatury TC1047A:

Alarm!

Jako źródło danych należy wziąć potencjometr czujnik temperatury TC 1047A, nastawę – wartość po

przekroczeniu której alarm załącza się – ustawić na temperaturę 25 stopni.

Po przekroczeniu wartości nastawy – przez pierwsze 3 sekundy mruga jedna dioda, a potem zapalone

są wszystkie.

Jeśli alarm jest załączony (mruganie, zapalone wszystkie diody) – to po spadku temperatury poniżej

nastawy alarmowej – alarm należy wyłączyć (przerwać mruganie, zgasić diody)

Wyłączenie alarmu – przycisk S6

## Problemy podczas implementacji:

Problem pojawił się w zmianie wartości potencjometru na stopnie Celsjusza oraz sam czujnik temperatury działał niespójnie.

## Kod:

#include <stdio.h>

#include <stdlib.h>

#include <stddef.h>

#include <stdbool.h>

#include "app.h"

#include <p24fj128ga010.h> //z zad 1

#define ONE\_VOLT 310

#define ONE\_TENTH\_VOLT 31

#define ONE\_HUNDREDTH\_VOLT 3

#define SCALE 308L // z zad 1

void Update\_LCD**(**void**);**

void SYS\_Initialize**(**void**);**

extern void ConvertADCVoltage**(**unsigned int**);**

extern void Hex2Dec**(**unsigned char**);**

APP\_DATA appData **=** **{**

**.**messageLine1 **=** "Explorer 16 Demo"**,**

**.**messageLine2 **=** "Press S3 to cont"**,**

**.**messageTime **=** "Time 00: 00: 00 "**,**

**.**messageADC **=** " Pot = 0.00 Vdc "

**};**

void delay**(**unsigned int ms**)** **{**

unsigned int i**;**

unsigned char j**;**

**for** **(**i **=** 0**;** i **<** ms**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** 200**;** j**++)** **{**

Nop**();**

Nop**();**

Nop**();**

Nop**();**

Nop**();**

**}**

**}**

**}**

int alarmActive **=** 0**;**

void triggerAlarm**(**int threshold**)** **{**

int loopCounter**;**

**if** **(**appData**.**adhunds **-** '0' **>** threshold**)** **{**

**if** **(**alarmActive **==** 0**)** **{**

**for** **(**loopCounter **=** 0**;** loopCounter **<=** 10**;** loopCounter**++)** **{**

appData**.**temp1 **=** ADC\_Read10bit**(**ADC\_CHANNEL\_TEMPERATURE\_SENSOR**);**

ConvertADCVoltage**(**appData**.**temp1**);**

**if** **(**appData**.**adhunds **-** '0' **<=** threshold**)** **{**

**continue;**

**}**

LED\_On**(**LED\_D9**);**

delay**(**150**);**

LED\_Off**(**LED\_D9**);**

delay**(**150**);**

**}**

**if** **(**appData**.**adhunds **-** '0' **>** threshold**)** **{**

LATA **=** 255**;**

**}**

alarmActive **=** 1**;**

**}**

**}** **else** **{**

LATA **=** 0**;**

alarmActive **=** 0**;**

**}**

**}**

void task2**()** **{**

float baseTemp **=** 25**;**

**if** **(**BUTTON\_IsPressed**(**BUTTON\_S4**))** **{**

triggerAlarm**(-**1**);**

**}**

appData**.**temp1 **=** ADC\_Read10bit**(**ADC\_CHANNEL\_TEMPERATURE\_SENSOR**);**

ConvertADCVoltage**(**appData**.**temp1**);**

float voltage **=** **(**appData**.**adones **-** '0'**)+(**appData**.**adtens **-** '0'**)\***0.1 **+** **(**appData**.**adhunds **-** '0'**)\***0.01**;**

float temperature **=** **(**voltage **-** 0.1**)\***100 **-** 46**;**

char str**[**20**];** //size of the number

sprintf**(**str**,** "temperatura %g stopni"**,** temperature**);**

LCD\_PutString**(**str**,** **sizeof** **(**str**));**

**if** **(!(**BUTTON\_IsPressed**(**BUTTON\_S6**)))** **{**

**if** **(**baseTemp **<=** temperature**)** **{**

**for** **(**int i **=** 0**;** i **<** 10**;** i**++)** **{**

**if** **(**baseTemp **>** temperature**)** **{**

**break;**

**}**

LED\_On**(**LED\_D3**);**

delay**(**150**);**

LED\_Off**(**LED\_D3**);**

delay**(**150**);**

**}**

LATA **=** 255**;**

**}** **else** **{**

LATA **=** 0**;**

**}**

**}** **else** **{**

LATA **=** 0**;**

delay**(**3000**);**

**}**

delay**(**300**);**

LCD\_ClearScreen**();**

appData**.**adc\_lcd\_update **=** 0**;**

**}**

int main**(**void**)** **{**

SYS\_Initialize**();**

TIMER\_SetConfiguration**(**TIMER\_CONFIGURATION\_RTCC**);**

ADC\_SetConfiguration**(**ADC\_CONFIGURATION\_AUTO\_SAMPLE\_CONVERT**);**

**while** **(**1**)** **{**

task2**();**

**};**

**}**

void Update\_LCD**(**void**)** **{**

LCD\_PutChar**(**'T'**);**

LCD\_PutChar**(**'i'**);**

LCD\_PutChar**(**'m'**);**

LCD\_PutChar**(**'e'**);**

LCD\_PutChar**(**' '**);**

LCD\_PutChar**(**appData**.**tens **+** 0x30**);**

LCD\_PutChar**(**appData**.**ones **+** 0x30**);**

Hex2Dec**(**appData**.**minutes**);**

LCD\_PutChar**(**':'**);**

LCD\_PutChar**(**' '**);**

LCD\_PutChar**(**appData**.**tens **+** 0x30**);**

LCD\_PutChar**(**appData**.**ones **+** 0x30**);**

Hex2Dec**(**appData**.**seconds**);**

LCD\_PutChar**(**':'**);**

LCD\_PutChar**(**' '**);**

LCD\_PutChar**(**appData**.**tens **+** 0x30**);**

LCD\_PutChar**(**appData**.**ones **+** 0x30**);**

**}**

void ConvertADCVoltage**(**unsigned int adc\_conv\_data**)** **{**

/\* reset values \*/

appData**.**adones **=** 0**;**

appData**.**adtens **=** 0**;**

appData**.**adhunds **=** 0**;**

appData**.**adthous **=** 0**;**

**while** **(**adc\_conv\_data **>** 0**)** **{**

/\* test for 1 volt or greater \*/

**if** **(**adc\_conv\_data **>** **(**ONE\_VOLT **-** 1**))** **{**

/\* increment 1 volt counter \*/

appData**.**adones**++;**

/\* subtract 1 volt \*/

adc\_conv\_data **-=** ONE\_VOLT**;**

**}**

/\* test for 0.1 volt \*/

**else** **if** **(**adc\_conv\_data **>** **(**ONE\_TENTH\_VOLT **-** 1**))** **{**

/\* increment tenths \*/

**if** **(**appData**.**adtens **<** 9**)** **{**

appData**.**adtens**++;**

**}** **else** **{**

/\* tenths has rolled over \*/

appData**.**adones**++;**

/\* so increment ones and reset tenths \*/

appData**.**adtens **=** 0**;**

**}**

adc\_conv\_data **-=** ONE\_TENTH\_VOLT**;**

**}** /\* test for 0.01 volt \*/

**else** **if** **(**adc\_conv\_data **>** **(**ONE\_HUNDREDTH\_VOLT **-** 1**))** **{**

/\* increment hundredths \*/

**if** **(**appData**.**adhunds **<** 9**)** **{**

appData**.**adhunds**++;**

**}** **else** **{**

/\* hundredths has rolled over \*/

appData**.**adtens**++;**

/\* so increment tenths and reset hundredths \*/

appData**.**adhunds **=** 0**;**

**}**

adc\_conv\_data **-=** ONE\_HUNDREDTH\_VOLT**;**

**}** **else** **if** **(**adc\_conv\_data **<=** **(**ONE\_HUNDREDTH\_VOLT **-** 1**))** **{**

appData**.**adthous**++;**

adc\_conv\_data**--;**

**}**

**}**

appData**.**adones **+=** 0x30**;**

appData**.**adtens **+=** 0x30**;**

appData**.**adhunds **+=** 0x30**;**

appData**.**adthous **+=** 0x30**;**

**}**

void Hex2Dec**(**unsigned char count**)** **{**

/\* reset values \*/

appData**.**hunds **=** 0**;**

appData**.**tens **=** 0**;**

appData**.**ones **=** 0**;**

**while** **(**count **>=** 10**)** **{**

**if** **(**count **>=** 200**)** **{**

count **-=** 200**;**

appData**.**hunds **=** 0x02**;**

**}**

**if** **(**count **>=** 100**)** **{**

count **-=** 100**;**

appData**.**hunds**++;**

**}**

**if** **(**count **>=** 10**)** **{**

count **-=** 10**;**

appData**.**tens**++;**

**}**

**}**

appData**.**ones **=** count**;**

**}**

# Zadanie 3

## Opis zadania:

W oparciu o program przykładowy dla Explorer 16/32 oraz PIC24FJ128GA010.

Wszelkie komunikaty z zadao powinny byd wyświetlone na wyświetlaczu LCD 16x2.

Reklama!

Atrakcyjna, - temat dowolny

## Problemy podczas implementacji:

Osoba pisząca to sprawozdanie jest mało kreatywna więc największą trudność sprawiło wymyślenie tematu reklamy.

## Kod:

#include <stdio.h>

#include <stdlib.h>

#include <stddef.h>

#include <stdbool.h>

#include "app.h"

#define ONE\_VOLT 310

#define ONE\_TENTH\_VOLT 31

#define ONE\_HUNDREDTH\_VOLT 3

void Update\_LCD **(** void **)** **;**

void SYS\_Initialize **(** void **)** **;**

extern void ConvertADCVoltage **(** unsigned int **)** **;**

extern void Hex2Dec **(** unsigned char **)** **;**

APP\_DATA appData **=** **{**

**.**messageLine1 **=** "Explorer 16 Demo" **,**

**.**messageLine2 **=** "Press S3 to cont" **,**

**.**messageTime **=** "Time 00: 00: 00 " **,**

**.**messageADC **=** " Pot = 0.00 Vdc "

**}** **;**

void delay**(**unsigned int ms**)**

**{**

unsigned int i**;**

unsigned char j**;**

**for** **(**i **=** 0**;** i **<** ms**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** 200**;** j**++)** **{**

Nop**();**

Nop**();**

Nop**();**

Nop**();**

Nop**();**

**}**

**}**

**}**

int main **(** void **)**

**{**

/\* Call the System Intialize routine\*/

SYS\_Initialize **(** **)** **;**

/\* Infinite Loop \*/

**while** **(** 1 **)**

**{**

LCD\_PutChar **(**'W'**)** **;**

LCD\_PutChar **(**'e'**)** **;**

LCD\_PutChar **(**'l'**)** **;**

LCD\_PutChar **(**'c'**)** **;**

LCD\_PutChar **(**'o'**)** **;**

LCD\_PutChar **(**'m'**)** **;**

LCD\_PutChar **(**'e'**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**'t'**)** **;**

LCD\_PutChar **(**'r'**)** **;**

LCD\_PutChar **(**'a'**)** **;**

LCD\_PutChar **(**'v'**)** **;**

LCD\_PutChar **(**'e'**)** **;**

LCD\_PutChar **(**'l'**)** **;**

LCD\_PutChar **(**'e'**)** **;**

LCD\_PutChar **(**'r'**)** **;**

delay**(**2000**);**

LCD\_ClearScreen**();**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

delay**(**2000**);**

LCD\_ClearScreen**();**

LCD\_PutChar **(**'C'**)** **;**

LCD\_PutChar **(**'o'**)** **;**

LCD\_PutChar **(**'m'**)** **;**

LCD\_PutChar **(**'e'**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**'t'**)** **;**

LCD\_PutChar **(**'o'**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**'A'**)** **;**

LCD\_PutChar **(**'T'**)** **;**

LCD\_PutChar **(**'O'**)** **;**

LCD\_PutChar **(**'M'**)** **;**

LCD\_PutChar **(**'I'**)** **;**

LCD\_PutChar **(**'C'**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**'W'**)** **;**

LCD\_PutChar **(**'R'**)** **;**

LCD\_PutChar **(**'A'**)** **;**

LCD\_PutChar **(**'N'**)** **;**

LCD\_PutChar **(**'G'**)** **;**

LCD\_PutChar **(**'L'**)** **;**

LCD\_PutChar **(**'E'**)** **;**

LCD\_PutChar **(**'R'**)** **;**

delay**(**2000**);**

LCD\_ClearScreen**();**

LCD\_PutChar **(**'W'**)** **;**

LCD\_PutChar **(**'e'**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**'h'**)** **;**

LCD\_PutChar **(**'a'**)** **;**

LCD\_PutChar **(**'v'**)** **;**

LCD\_PutChar **(**'e'**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**'B'**)** **;**

LCD\_PutChar **(**'O'**)** **;**

LCD\_PutChar **(**'O'**)** **;**

LCD\_PutChar **(**'Z'**)** **;**

LCD\_PutChar **(**'E'**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**'a'**)** **;**

LCD\_PutChar **(**'n'**)** **;**

LCD\_PutChar **(**'d'**)** **;**

LCD\_PutChar **(**' '**)** **;**

LCD\_PutChar **(**'G'**)** **;**

LCD\_PutChar **(**'I'**)** **;**

LCD\_PutChar **(**'R'**)** **;**

LCD\_PutChar **(**'L'**)** **;**

LCD\_PutChar **(**'S'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

LCD\_PutChar **(**'$'**)** **;**

delay**(**2000**);**

LCD\_ClearScreen**();**

**}** **;**

**}**

void Update\_LCD **(** void **)**

**{**

LCD\_PutChar **(** 'T' **)** **;**

LCD\_PutChar **(** 'i' **)** **;**

LCD\_PutChar **(** 'm' **)** **;**

LCD\_PutChar **(** 'e' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

Hex2Dec **(** appData**.**minutes **)** **;**

LCD\_PutChar **(** ':' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

Hex2Dec **(** appData**.**seconds **)** **;**

LCD\_PutChar **(** ':' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

**}**

void ConvertADCVoltage **(** unsigned int adc\_conv\_data **)**

**{**

/\* reset values \*/

appData**.**adones **=** 0 **;**

appData**.**adtens **=** 0 **;**

appData**.**adhunds **=** 0 **;**

appData**.**adthous **=** 0 **;**

**while** **(** adc\_conv\_data **>** 0 **)**

**{**

/\* test for 1 volt or greater \*/

**if** **(** adc\_conv\_data **>** **(** ONE\_VOLT **-** 1 **)** **)**

**{**

/\* increment 1 volt counter \*/

appData**.**adones**++** **;**

/\* subtract 1 volt \*/

adc\_conv\_data **-=** ONE\_VOLT **;**

**}**

/\* test for 0.1 volt \*/

**else** **if** **(** adc\_conv\_data **>** **(** ONE\_TENTH\_VOLT **-** 1 **)** **)**

**{**

/\* increment tenths \*/

**if** **(** appData**.**adtens **<** 9 **)**

**{**

appData**.**adtens**++** **;**

**}**

**else**

**{**

/\* tenths has rolled over \*/

appData**.**adones**++** **;**

/\* so increment ones and reset tenths \*/

appData**.**adtens **=** 0 **;**

**}**

adc\_conv\_data **-=** ONE\_TENTH\_VOLT **;**

**}**

/\* test for 0.01 volt \*/

**else** **if** **(** adc\_conv\_data **>** **(** ONE\_HUNDREDTH\_VOLT **-** 1 **)** **)**

**{**

/\* increment hundredths \*/

**if** **(** appData**.**adhunds **<** 9 **)**

**{**

appData**.**adhunds**++** **;**

**}**

**else**

**{**

/\* hundredths has rolled over \*/

appData**.**adtens**++** **;**

/\* so increment tenths and reset hundredths \*/

appData**.**adhunds **=** 0 **;**

**}**

adc\_conv\_data **-=** ONE\_HUNDREDTH\_VOLT **;**

**}**

**else** **if** **(** adc\_conv\_data **<=** **(** ONE\_HUNDREDTH\_VOLT **-** 1 **)** **)**

**{**

appData**.**adthous**++** **;**

adc\_conv\_data **--** **;**

**}**

**}**

appData**.**adones **+=** 0x30 **;**

appData**.**adtens **+=** 0x30 **;**

appData**.**adhunds **+=** 0x30 **;**

appData**.**adthous **+=** 0x30 **;**

**}**

void Hex2Dec **(** unsigned char count **)**

**{**

/\* reset values \*/

appData**.**hunds **=** 0 **;**

appData**.**tens **=** 0 **;**

appData**.**ones **=** 0 **;**

**while** **(** count **>=** 10 **)**

**{**

**if** **(** count **>=** 200 **)**

**{**

count **-=** 200 **;**

appData**.**hunds **=** 0x02 **;**

**}**

**if** **(**count **>=** 100**)**

**{**

count **-=** 100 **;**

appData**.**hunds**++** **;**

**}**

**if** **(**count **>=** 10 **)**

**{**

count **-=** 10 **;**

appData**.**tens**++** **;**

**}**

**}**

appData**.**ones **=** count **;**

**}**

# Zadanie 4

## Opis zadania:

W oparciu o program przykładowy dla Explorer 16/32 oraz PIC24FJ128GA010.

Wszelkie komunikaty z zadania powinny być wyświetlone na wyświetlaczu LCD 16x2.

Kontroler kuchenki mikrofalowej.

S3 – wybór mocy 800W, 600W, 350W, 200W

S6 – dodanie czasu 30 s

S5 – Start/Stop

S4 – Reset

## Problemy podczas implementacji:

Przyciski na płytce sprawiały trudności pod tym względem że czasami trzeba było kilkukrotnie nacisnąć dany przycisk aby dana funkcja programu zaczęła działać.

## Kod:

Main.c

#include <stdio.h>

#include <stdlib.h>

#include <stddef.h>

#include <stdbool.h>

#include "app.h"

#define ONE\_VOLT 310

#define ONE\_TENTH\_VOLT 31

#define ONE\_HUNDREDTH\_VOLT 3

void Update\_LCD **(** void **)** **;**

void SYS\_Initialize **(** void **)** **;**

extern void ConvertADCVoltage **(** unsigned int **)** **;**

extern void Hex2Dec **(** unsigned char **)** **;**

APP\_DATA appData **=** **{**

**.**messageLine1 **=** "Explorer 16 Demo" **,**

**.**messageLine2 **=** "Press S3 to cont" **,**

**.**messageTime **=** "Time 00: 00: 00 " **,**

**.**messageADC **=** " Pot = 0.00 Vdc "

**}** **;**

void delay**(**unsigned int ms**)**

**{**

unsigned int i**;**

unsigned char j**;**

**for** **(**i **=** 0**;** i **<** ms**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** 200**;** j**++)** **{**

Nop**();**

Nop**();**

Nop**();**

Nop**();**

Nop**();**

**}**

**}**

**}**

int main **(** void **)**

**{**

/\* Call the System Intialize routine\*/

SYS\_Initialize **(** **)** **;**

/\* Display welcome message \*/

LCD\_PutString **(** **(**char**\*)** **&**appData**.**messageLine1**[**0**]** **,** **sizeof** **(**appData**.**messageLine1 **)** **-** 1 **)** **;**

LCD\_PutString **(** **(**char**\*)** **&**appData**.**messageLine2**[**0**]** **,** **sizeof** **(**appData**.**messageLine2 **)** **-** 1 **)** **;**

/\*Initialize Timer\*/

TIMER\_SetConfiguration **(** TIMER\_CONFIGURATION\_RTCC **)** **;**

/\*Initialize ADC configure it as 10 bit interrupt driven mode\*/

ADC\_SetConfiguration **(** ADC\_CONFIGURATION\_AUTO\_SAMPLE\_CONVERT **)** **;**

/\* Display initial ADC and Timer messages \*/

LCD\_PutString **(** **(**char**\*)** **&**appData**.**messageTime**[**0**]** **,** **sizeof** **(**appData**.**messageTime **)** **-** 1 **)** **;**

LCD\_PutString **(** **(**char**\*)** **&**appData**.**messageADC**[**0**]** **,** **sizeof** **(**appData**.**messageADC **)** **-** 1 **)** **;**

char moc**[**12**]=**"Moc: "**;**

char czas**[**11**]=**"Czas: "**;**

int czasWrt**=**0**;**

int mocWrt**=**200**;**

char mocWrtStr**[**3**]=**"000"**;**

char czasSek**[**2**]=**"00"**;**

char czasMin**[**2**]=**"00"**;**

void tostring**(**char str**[],** int num**)**

**{**

**if** **(**num**==**0**)**

**{**

str**[**0**]=**'0'**;**

str**[**1**]=**'0'**;**

**}**

**else{**

int numCp**,** i**,** rem**,** len **=** 0**,** n**;**

numCp**=**num**;**

n **=** num**;**

**while** **(**n **!=** 0**)**

**{**

len**++;**

n **/=** 10**;**

**}**

**if** **(**num**==**1**)**

**{**

len**++;**

**}**

**if** **(**num**==**2**)**

**{**

len**++;**

**}**

**if** **(**num**==**3**)**

**{**

len**++;**

**}**

**if** **(**num**==**4**)**

**{**

len**++;**

**}**

**if** **(**num**==**5**)**

**{**

len**++;**

**}**

**if** **(**num**==**6**)**

**{**

len**++;**

**}**

**if** **(**num**==**7**)**

**{**

len**++;**

**}**

**if** **(**num**==**8**)**

**{**

len**++;**

**}**

**if** **(**num**==**9**)**

**{**

len**++;**

**}**

**for** **(**i **=** 0**;** i **<** len**;** i**++)**

**{**

rem **=** numCp **%** 10**;**

numCp **=** numCp **/** 10**;**

str**[**len **-** **(**i **+** 1**)]** **=** rem **+** '0'**;**

**}**

**}**

**}**

/\* Infinite Loop \*/

**while** **(** 1 **)**

**{**

tostring**(**czasSek **,**czasWrt**%**60**);**

tostring**(**czasMin **,**czasWrt**/**60**);**

tostring**(**mocWrtStr **,**mocWrt**);**

LCD\_PutString **(** moc **,** **sizeof(**moc**)** **)** **;**

LCD\_PutString **(** mocWrtStr **,** **sizeof(**mocWrtStr**)** **)** **;**

LCD\_PutString **(** "W" **,** 1 **)** **;**

LCD\_PutString **(** czas **,** **sizeof(**czas**)** **)** **;**

LCD\_PutString **(** czasMin **,** **sizeof(**czasMin**)** **)** **;**

LCD\_PutString **(** ":" **,** 1 **)** **;**

LCD\_PutString **(** czasSek **,** **sizeof(**czasSek**)** **)** **;**

delay**(**500**);**

LCD\_ClearScreen**();**

**while((**BUTTON\_IsPressed**(**BUTTON\_S3**)))**

**{**

**if(**mocWrt**==**200**)**

**{**

mocWrt**=**350**;**

**break;**

**}**

**if(**mocWrt**==**350**)**

**{**

mocWrt**=**600**;**

**break;**

**}**

**if(**mocWrt**==**600**)**

**{**

mocWrt**=**800**;**

**break;**

**}**

**if(**mocWrt**==**800**)**

**{**

mocWrt**=**200**;**

**break;**

**}**

**}**

**while((**BUTTON\_IsPressed**(**BUTTON\_S6**)))**

**{**

czasWrt**+=**30**;**

delay**(**500**);**

**}**

**while((**BUTTON\_IsPressed**(**BUTTON\_S5**))&&(**czasWrt**>**0**))**

**{**

delay**(**1000**);**

**while(**czasWrt**!=**0**)**

**{**

**if((**BUTTON\_IsPressed**(**BUTTON\_S5**)))**

**{**

tostring**(**czasSek **,**czasWrt**%**60**);**

tostring**(**czasMin **,**czasWrt**/**60**);**

tostring**(**mocWrtStr **,**mocWrt**);**

LCD\_PutString **(** moc **,** **sizeof(**moc**)** **)** **;**

LCD\_PutString **(** mocWrtStr **,** **sizeof(**mocWrtStr**)** **)** **;**

LCD\_PutString **(** "W" **,** 1 **)** **;**

LCD\_PutString **(** czas **,** **sizeof(**czas**)** **)** **;**

LCD\_PutString **(** czasMin **,** **sizeof(**czasMin**)** **)** **;**

LCD\_PutString **(** ":" **,** 1 **)** **;**

LCD\_PutString **(** czasSek **,** **sizeof(**czasSek**)** **)** **;**

delay**(**1000**);**

**break;**

**}**

czasWrt**--;**

tostring**(**czasSek **,**czasWrt**%**60**);**

tostring**(**czasMin **,**czasWrt**/**60**);**

tostring**(**mocWrtStr **,**mocWrt**);**

LCD\_PutString **(** moc **,** **sizeof(**moc**)** **)** **;**

LCD\_PutString **(** mocWrtStr **,** **sizeof(**mocWrtStr**)** **)** **;**

LCD\_PutString **(** "W" **,** 1 **)** **;**

LCD\_PutString **(** czas **,** **sizeof(**czas**)** **)** **;**

LCD\_PutString **(** czasMin **,** **sizeof(**czasMin**)** **)** **;**

LCD\_PutString **(** ":" **,** 1 **)** **;**

LCD\_PutString **(** czasSek **,** **sizeof(**czasSek**)** **)** **;**

delay**(**1000**);**

LCD\_ClearScreen**();**

**}**

**}**

**while((**BUTTON\_IsPressed**(**BUTTON\_S4**)))**

**{**

czasWrt**=**0**;**

mocWrt**=**200**;**

delay**(**1000**);**

**}**

**}** **;**

**}**

void Update\_LCD **(** void **)**

**{**

LCD\_PutChar **(** 'T' **)** **;**

LCD\_PutChar **(** 'i' **)** **;**

LCD\_PutChar **(** 'm' **)** **;**

LCD\_PutChar **(** 'e' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

Hex2Dec **(** appData**.**minutes **)** **;**

LCD\_PutChar **(** ':' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

Hex2Dec **(** appData**.**seconds **)** **;**

LCD\_PutChar **(** ':' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

**}**

void ConvertADCVoltage **(** unsigned int adc\_conv\_data **)**

**{**

/\* reset values \*/

appData**.**adones **=** 0 **;**

appData**.**adtens **=** 0 **;**

appData**.**adhunds **=** 0 **;**

appData**.**adthous **=** 0 **;**

**while** **(** adc\_conv\_data **>** 0 **)**

**{**

/\* test for 1 volt or greater \*/

**if** **(** adc\_conv\_data **>** **(** ONE\_VOLT **-** 1 **)** **)**

**{**

/\* increment 1 volt counter \*/

appData**.**adones**++** **;**

/\* subtract 1 volt \*/

adc\_conv\_data **-=** ONE\_VOLT **;**

**}**

/\* test for 0.1 volt \*/

**else** **if** **(** adc\_conv\_data **>** **(** ONE\_TENTH\_VOLT **-** 1 **)** **)**

**{**

/\* increment tenths \*/

**if** **(** appData**.**adtens **<** 9 **)**

**{**

appData**.**adtens**++** **;**

**}**

**else**

**{**

/\* tenths has rolled over \*/

appData**.**adones**++** **;**

/\* so increment ones and reset tenths \*/

appData**.**adtens **=** 0 **;**

**}**

adc\_conv\_data **-=** ONE\_TENTH\_VOLT **;**

**}**

/\* test for 0.01 volt \*/

**else** **if** **(** adc\_conv\_data **>** **(** ONE\_HUNDREDTH\_VOLT **-** 1 **)** **)**

**{**

/\* increment hundredths \*/

**if** **(** appData**.**adhunds **<** 9 **)**

**{**

appData**.**adhunds**++** **;**

**}**

**else**

**{**

/\* hundredths has rolled over \*/

appData**.**adtens**++** **;**

/\* so increment tenths and reset hundredths \*/

appData**.**adhunds **=** 0 **;**

**}**

adc\_conv\_data **-=** ONE\_HUNDREDTH\_VOLT **;**

**}**

**else** **if** **(** adc\_conv\_data **<=** **(** ONE\_HUNDREDTH\_VOLT **-** 1 **)** **)**

**{**

appData**.**adthous**++** **;**

adc\_conv\_data **--** **;**

**}**

**}**

appData**.**adones **+=** 0x30 **;**

appData**.**adtens **+=** 0x30 **;**

appData**.**adhunds **+=** 0x30 **;**

appData**.**adthous **+=** 0x30 **;**

**}**

void Hex2Dec **(** unsigned char count **)**

**{**

/\* reset values \*/

appData**.**hunds **=** 0 **;**

appData**.**tens **=** 0 **;**

appData**.**ones **=** 0 **;**

**while** **(** count **>=** 10 **)**

**{**

**if** **(** count **>=** 200 **)**

**{**

count **-=** 200 **;**

appData**.**hunds **=** 0x02 **;**

**}**

**if** **(**count **>=** 100**)**

**{**

count **-=** 100 **;**

appData**.**hunds**++** **;**

**}**

**if** **(**count **>=** 10 **)**

**{**

count **-=** 10 **;**

appData**.**tens**++** **;**

**}**

**}**

appData**.**ones **=** count **;**

**}**

**System.c**

#include <xc.h>

#include "app.h"

// CONFIG2

#pragma config POSCMOD = XT // Primary Oscillator Select (XT Oscillator mode selected)

#pragma config OSCIOFNC = OFF // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = PRI // Oscillator Select (Primary Oscillator (XT, HS, EC))

#pragma config IESO = ON // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)

// CONFIG1

#pragma config WDTPS = PS32768 // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128 // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))

#pragma config FWDTEN = OFF // Watchdog Timer Enable (Watchdog Timer is disabled)

#pragma config ICS = PGx2 // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

#pragma config GWRP = OFF // General Code Segment Write Protect (Writes to program memory are allowed)

#pragma config GCP = OFF // General Code Segment Code Protect (Code protection is disabled)

#pragma config JTAGEN = OFF // JTAG Port Enable (JTAG port is disabled)

void SOSC\_Configuration **(** void **)** **;**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_OscillatorFail **(** void **)** **;**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AddressError **(** void **)** **;**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_StackError **(** void **)** **;**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_MathError **(** void **)** **;**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AltOscillatorFail **(** void **)** **;**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AltAddressError **(** void **)** **;**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AltStackError **(** void **)** **;**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AltMathError **(** void **)** **;**

void SYS\_Initialize **(** void **)**

**{**

/\* Enable LEDs\*/

// LED\_Enable ( LED\_D9 ) ;

// LED\_Enable ( LED\_D10 ) ;

/\* Turn On LEDs\*/

// LED\_On ( LED\_D9 ) ;

// LED\_On ( LED\_D10 ) ;

/\* Enable Switch S3\*/

BUTTON\_Enable **(** BUTTON\_S3 **)** **;**

/\* Enable ADC to the Potentiometer channel \*/

ADC\_ChannelEnable **(** ADC\_CHANNEL\_POTENTIOMETER **)** **;**

/\* Initialize LCD\*/

PRINT\_SetConfiguration **(** PRINT\_CONFIGURATION\_LCD **)** **;**

/\* Configure Secondary Ocillator for Timer 1 to work as RTC counter\*/

SOSC\_Configuration **(** **)** **;**

**}**

void SOSC\_Configuration **(** void **)**

**{**

char a **,** b **,** c **,** **\***p **;**

a **=** 2 **;**

b **=** 0x46 **;**

c **=** 0x57 **;**

p **=** **(**char **\*)** **&**OSCCON **;**

asm volatile **(**"mov.b %1,[%0] \n"

"mov.b %2,[%0] \n"

"mov.b %3,[%0] \n" **:** /\* no outputs \*/ **:** "r"**(** p **)** **,** "r"**(** b **)** **,** "r"**(** c **)** **,**

"r"**(** a **)** **)** **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_ADC1Interrupt **(** void **)**

**{**

static int count **=** 0 **;**

/\* Simple "I am here" indicator \*/

**if** **(** count**++** **==** 2000 **)**

**{**

LED\_Toggle **(** LED\_D9 **)** **;**

count **=** 0 **;**

**}**

/\* Save the Potentiometer data \*/

appData**.**temp1 **=** ADC1BUF0 **;**

/\* set flag to update LCD \*/

appData**.**adc\_lcd\_update **=** 1 **;**

/\* reset ADC interrupt flag \*/

IFS0bits**.**AD1IF **=** 0 **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_T1Interrupt **(** void **)**

**{**

/\* update seconds \*/

**if** **(** appData**.**seconds **<** 59 **)**

**{**

appData**.**seconds**++** **;**

**}**

/\* update minutes \*/

**else**

**{**

appData**.**seconds **=** 0x00 **;**

**if** **(** appData**.**minutes **<** 59 **)**

**{**

appData**.**minutes**++** **;**

**}**

/\* update hours \*/

**else**

**{**

appData**.**minutes **=** 0x00 **;**

**if** **(** appData**.**hours **<** 23 **)**

**{**

appData**.**hours **++** **;**

**}**

**else**

**{**

appData**.**hours **=** 0x00 **;**

**}**

**}**

**}**

/\* set flag to update LCD \*/

appData**.**rtc\_lcd\_update **=** 1 **;**

/\* Toggle LED at 1 Hz rate \*/

LED\_Toggle **(** LED\_D10 **)** **;**

/\* reset Timer 1 interrupt flag \*/

IFS0bits**.**T1IF **=** 0 **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_OscillatorFail **(** void **)**

**{**

INTCON1bits**.**OSCFAIL **=** 0 **;** //Clear the trap flag

**while** **(**1**)** **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AddressError **(** void **)**

**{**

INTCON1bits**.**ADDRERR **=** 0 **;** //Clear the trap flag

**while** **(**1**)** **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_StackError **(** void **)**

**{**

INTCON1bits**.**STKERR **=** 0 **;** //Clear the trap flag

**while** **(**1**)** **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_MathError **(** void **)**

**{**

INTCON1bits**.**MATHERR **=** 0 **;** //Clear the trap flag

**while** **(**1**)** **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AltOscillatorFail **(** void **)**

**{**

INTCON1bits**.**OSCFAIL **=** 0 **;**

**while** **(**1**)** **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AltAddressError **(** void **)**

**{**

INTCON1bits**.**ADDRERR **=** 0 **;**

**while** **(**1**)** **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AltStackError **(** void **)**

**{**

INTCON1bits**.**STKERR **=** 0 **;**

**while** **(**1**)** **;**

**}**

void \_\_attribute\_\_ **(** **(** \_\_interrupt\_\_ **,** auto\_psv **)** **)** \_AltMathError **(** void **)**

**{**

INTCON1bits**.**MATHERR **=** 0 **;**

**while** **(**1**)** **;**

**}**

# Zadanie 5

## Opis zadania:

W oparciu o program przykładowy dla Explorer 16/32 oraz PIC24FJ128GA010.

Wszelkie komunikaty z zadania powinny byd wyświetlone na wyświetlaczu LCD 16x2.

Zegar szachowy dla dwóch graczy

odmierzający czas SD (Sudden Death – całkowity czas na wykonanie ruchów w partii). Naciśnięcie

przycisku pierwszego gracza powoduje odmierzanie czasu drugiego gracza, a naciśnięcie przycisku

drugiego gracza – odmierzanie czasu pierwszego. Gra toczy się do poddania partii, ustalenia remisu,

Mata – to są jednak zdarzenia, których nie odnotowujemy na zegarze. Zegar ma wyświetlid czas

każdego z graczy i w razie gdy któryś z graczy nie wykona ruchu w pozostałym mu czasie wyświetlid

komunikat, że gracz przegrał grę przez czas.

Nastawa czasu – na podstawie potencjometru P2, 3 ustawienia czasu: 5min, 3min, 1 min

S3 – pierwszy gracz

S6 – drugi gracz

## Problemy podczas implementacji:

Przyciski na płytce sprawiały trudności pod tym względem że czasami trzeba było kilkukrotnie nacisnąć dany przycisk aby dana funkcja programu zaczęła działać.

## Kod:

#include <stdio.h>

#include <stdlib.h>

#include <stddef.h>

#include <stdbool.h>

#define FCY 4000000ULL

#include <libpic30.h>

#include "app.h"

#define ONE\_VOLT 310

#define ONE\_TENTH\_VOLT 31

#define ONE\_HUNDREDTH\_VOLT 3

void Update\_LCD **(** void **)** **;**

void SYS\_Initialize **(** void **)** **;**

extern void ConvertADCVoltage **(** unsigned int **)** **;**

extern void Hex2Dec **(** unsigned char **)** **;**

APP\_DATA appData **=** **{**

**.**messageLine1 **=** "Explorer 16 Demo" **,**

**.**messageLine2 **=** "Press S4 to cont" **,**

**.**messageTime **=** "Chess timer " **,**

**.**messageADC **=** " "

**}** **;**

int main **(** void **)**

**{**

/\* Call the System Intialize routine\*/

SYS\_Initialize **(** **)** **;**

/\* Display welcome message \*/

LCD\_PutString **(** **(**char**\*)** **&**appData**.**messageLine1**[**0**]** **,** **sizeof** **(**appData**.**messageLine1 **)** **-** 1 **)** **;**

LCD\_PutString **(** **(**char**\*)** **&**appData**.**messageLine2**[**0**]** **,** **sizeof** **(**appData**.**messageLine2 **)** **-** 1 **)** **;**

/\* wait here until switch S3 is pressed \*/

**while** **(!**BUTTON\_IsPressed **(** BUTTON\_S4 **))** **;**

/\*Initialize Timer\*/

TIMER\_SetConfiguration **(** TIMER\_CONFIGURATION\_RTCC **)** **;**

/\*Initialize ADC configure it as 10 bit interrupt driven mode\*/

ADC\_SetConfiguration **(** ADC\_CONFIGURATION\_AUTO\_SAMPLE\_CONVERT **)** **;**

int time1 **=** 300**;**

bool count1 **=** **false;**

int time1M **=** 0**;**

int time1S **=** 0**;**

int time2 **=** 300**;**

bool count2 **=** **false;**

int time2M **=** 0**;**

int time2S **=** 0**;**

bool g1Lose **=** **false;**

bool g2Lose **=** **false;**

bool game **=** **false;**

unsigned int pot **=** 0**;**

/\* Infinite Loop \*/

**while** **(** 1 **)**

**{**

**if(!**g1Lose **&&** **!**g2Lose **){**

**if** **(** appData**.**adc\_lcd\_update **)**

**{**

**if(!**game**){**

ConvertADCVoltage **(** appData**.**temp1 **)** **;**

float w **=** **(**appData**.**adones**-**'0'**)** **+** **(**appData**.**adtens**-**'0'**)\***0.1f **+** **(**appData**.**adhunds**-**'0'**)** **\*** 0.01f**;**

float x **=** w**/**3.3**;**

int y **=** x **\*** 100**;**

**if(**y **<** 33**){**

time1 **=** 300**;**

time2 **=** 300**;**

**}**

**if(**y **<** 66 **&&** y **>** 33**){**

time1 **=** 180**;**

time2 **=** 180**;**

**}**

**if(**y **>** 66**){**

time1 **=** 60**;**

time2 **=** 60**;**

**}**

**}**

time1M **=** time1**/**60**;**

time1S **=** time1 **-** **(**time1M **\*** 60**);**

time2M **=** time2**/**60**;**

time2S **=** time2 **-** **(**time2M **\*** 60**);**

LCD\_PutChar **(** 'G' **)** **;**

LCD\_PutChar **(** 'r' **)** **;**

LCD\_PutChar **(** 'a' **)** **;**

LCD\_PutChar **(** 'c' **)** **;**

LCD\_PutChar **(** 'z' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** '1' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** 'G' **)** **;**

LCD\_PutChar **(** 'r' **)** **;**

LCD\_PutChar **(** 'a' **)** **;**

LCD\_PutChar **(** 'c' **)** **;**

LCD\_PutChar **(** 'z'**)** **;**

LCD\_PutChar **(** ' '**)** **;**

LCD\_PutChar **(** '2' **)** **;**

LCD\_PutChar **(** time1M **/** 10 **+** '0' **)** **;**

LCD\_PutChar **(** time1M **%** 10 **+** '0' **)** **;**

LCD\_PutChar **(** ':' **)** **;**

LCD\_PutChar **(** time1S**/**10 **+** '0' **)** **;**

LCD\_PutChar **(** time1S **%** 10 **+** '0' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** time2M**/**10 **+** '0' **)** **;**

LCD\_PutChar **(** time2M **%** 10 **+** '0' **)** **;**

LCD\_PutChar **(** ':' **)** **;**

LCD\_PutChar **(** time2S**/**10 **+** '0' **)** **;**

LCD\_PutChar **(** time2S **%** 10 **+** '0' **)** **;**

**if(**BUTTON\_IsPressed **(** BUTTON\_S3 **)){**

game **=** **true;**

count1 **=** **true;**

count2 **=** **false;**

**}**

**if(**count1 **&&** time2 **>** 0**){**

time2 **=** time2 **-** 1**;**

\_\_delay\_ms**(**1000**);**

**}**

**if(**BUTTON\_IsPressed **(** BUTTON\_S6 **)){**

game **=** **true;**

count1 **=** **false;**

count2 **=** **true;**

**}**

**if(**count2 **&&** time1 **>** 0**){**

time1 **=** time1 **-** 1**;**

\_\_delay\_ms**(**1000**);**

**}**

**}**

**if(**time1 **==** 0**){**

**if(!**g1Lose**){**

LCD\_ClearScreen**();**

LCD\_PutString**(**"Gracz1 przegral!"**,** 16**);**

LCD\_PutString**(**"\* Koniec czasu \*"**,** 16**);**

g1Lose**=true;**

**}**

**}**

**if(**time2 **==** 0**){**

**if(!**g2Lose**){**

LCD\_ClearScreen**();**

LCD\_PutString**(**"Gracz2 przegral!"**,** 16**);**

LCD\_PutString**(**"\* Koniec czasu \*"**,** 16**);**

g2Lose**=true;**

**}**

**}**

**}**

**}**

void Update\_LCD **(** void **)**

**{**

LCD\_PutChar **(** 'T' **)** **;**

LCD\_PutChar **(** 'i' **)** **;**

LCD\_PutChar **(** 'm' **)** **;**

LCD\_PutChar **(** 'e' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

Hex2Dec **(** appData**.**minutes **)** **;**

LCD\_PutChar **(** ':' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

Hex2Dec **(** appData**.**seconds **)** **;**

LCD\_PutChar **(** ':' **)** **;**

LCD\_PutChar **(** ' ' **)** **;**

LCD\_PutChar **(** appData**.**tens **+** 0x30 **)** **;**

LCD\_PutChar **(** appData**.**ones **+** 0x30 **)** **;**

**}**

void ConvertADCVoltage **(** unsigned int adc\_conv\_data **)**

**{**

/\* reset values \*/

appData**.**adones **=** 0 **;**

appData**.**adtens **=** 0 **;**

appData**.**adhunds **=** 0 **;**

appData**.**adthous **=** 0 **;**

**while** **(** adc\_conv\_data **>** 0 **)**

**{**

/\* test for 1 volt or greater \*/

**if** **(** adc\_conv\_data **>** **(** ONE\_VOLT **-** 1 **)** **)**

**{**

/\* increment 1 volt counter \*/

appData**.**adones**++** **;**

/\* subtract 1 volt \*/

adc\_conv\_data **-=** ONE\_VOLT **;**

**}**

/\* test for 0.1 volt \*/

**else** **if** **(** adc\_conv\_data **>** **(** ONE\_TENTH\_VOLT **-** 1 **)** **)**

**{**

/\* increment tenths \*/

**if** **(** appData**.**adtens **<** 9 **)**

**{**

appData**.**adtens**++** **;**

**}**

**else**

**{**

/\* tenths has rolled over \*/

appData**.**adones**++** **;**

/\* so increment ones and reset tenths \*/

appData**.**adtens **=** 0 **;**

**}**

adc\_conv\_data **-=** ONE\_TENTH\_VOLT **;**

**}**

/\* test for 0.01 volt \*/

**else** **if** **(** adc\_conv\_data **>** **(** ONE\_HUNDREDTH\_VOLT **-** 1 **)** **)**

**{**

/\* increment hundredths \*/

**if** **(** appData**.**adhunds **<** 9 **)**

**{**

appData**.**adhunds**++** **;**

**}**

**else**

**{**

/\* hundredths has rolled over \*/

appData**.**adtens**++** **;**

/\* so increment tenths and reset hundredths \*/

appData**.**adhunds **=** 0 **;**

**}**

adc\_conv\_data **-=** ONE\_HUNDREDTH\_VOLT **;**

**}**

**else** **if** **(** adc\_conv\_data **<=** **(** ONE\_HUNDREDTH\_VOLT **-** 1 **)** **)**

**{**

appData**.**adthous**++** **;**

adc\_conv\_data **--** **;**

**}**

**}**

appData**.**adones **+=** 0x30 **;**

appData**.**adtens **+=** 0x30 **;**

appData**.**adhunds **+=** 0x30 **;**

appData**.**adthous **+=** 0x30 **;**

**}**

void Hex2Dec **(** unsigned char count **)**

**{**

/\* reset values \*/

appData**.**hunds **=** 0 **;**

appData**.**tens **=** 0 **;**

appData**.**ones **=** 0 **;**

**while** **(** count **>=** 10 **)**

**{**

**if** **(** count **>=** 200 **)**

**{**

count **-=** 200 **;**

appData**.**hunds **=** 0x02 **;**

**}**

**if** **(**count **>=** 100**)**

**{**

count **-=** 100 **;**

appData**.**hunds**++** **;**

**}**

**if** **(**count **>=** 10 **)**

**{**

count **-=** 10 **;**

appData**.**tens**++** **;**

**}**

**}**

appData**.**ones **=** count **;**

**}**