Pomoc do mplab

Spis treści

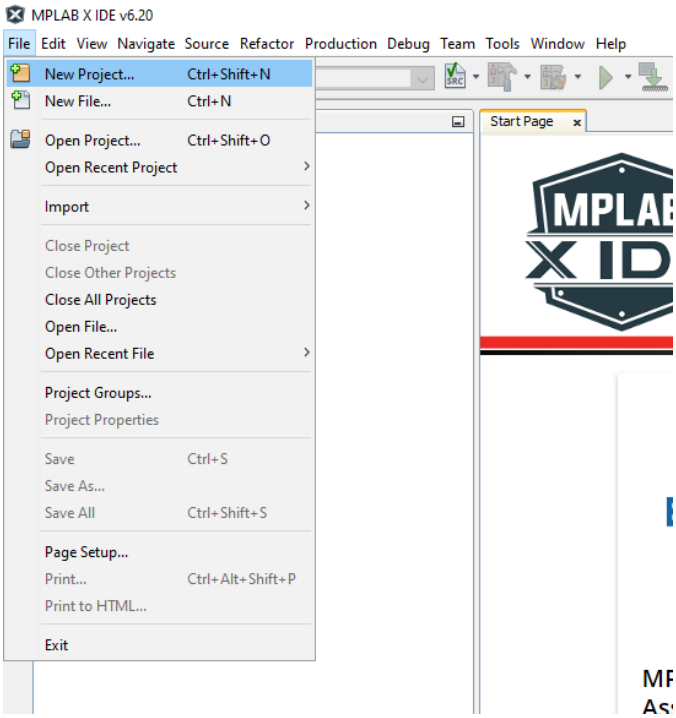
Wstęp 1
Stworzenie nowego projektu. 1
Konfiguracja bitów 5
Dodanie plików adc, lcd oraz buttons 6
Przydatne biblioteki 8
Opóźnienie i świecenie diod 9
Potencjometr 9
Przyciski 10
Ekran LCD 11

Wstęp

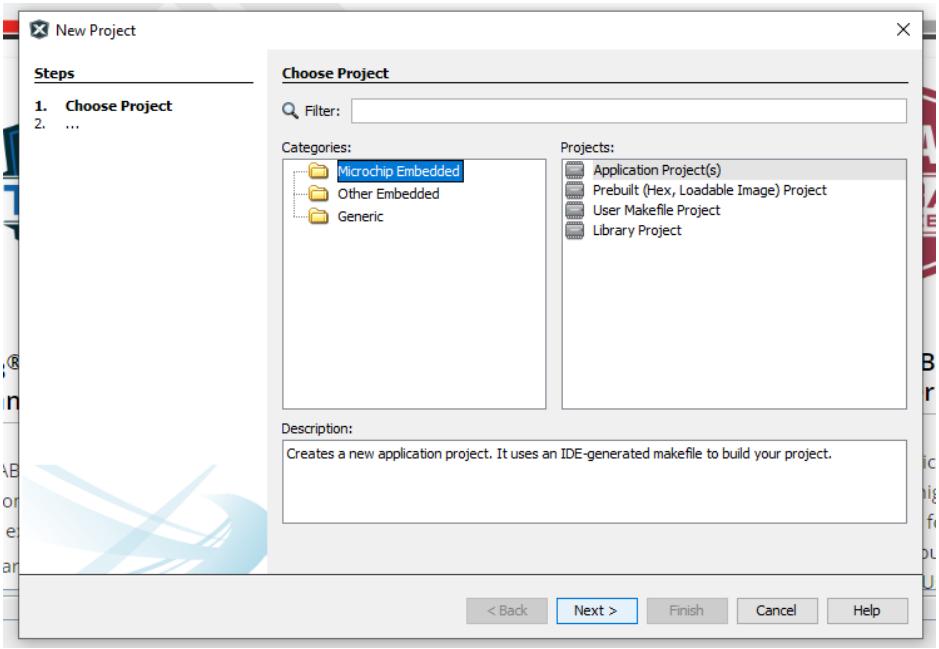
Cel dokumentu: Ułatwienie rozpoczęcia pracy z mikrokontrolerem PIC24FJ128GA010 w środowisku MPLAB oraz szybkie wdrożenie obsługi podstawowych peryferiów (LCD, ADC, przyciski, LED).

Stworzenie nowego projektu.

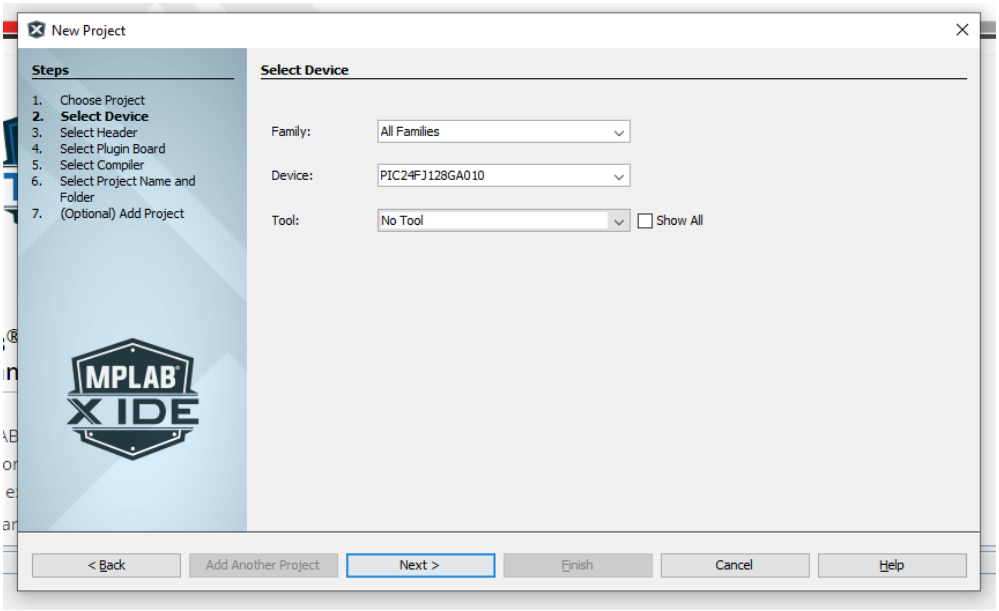
1. Należy wybrać z paska u góry zakładkę „File” a następnie wybrać „New Project”.

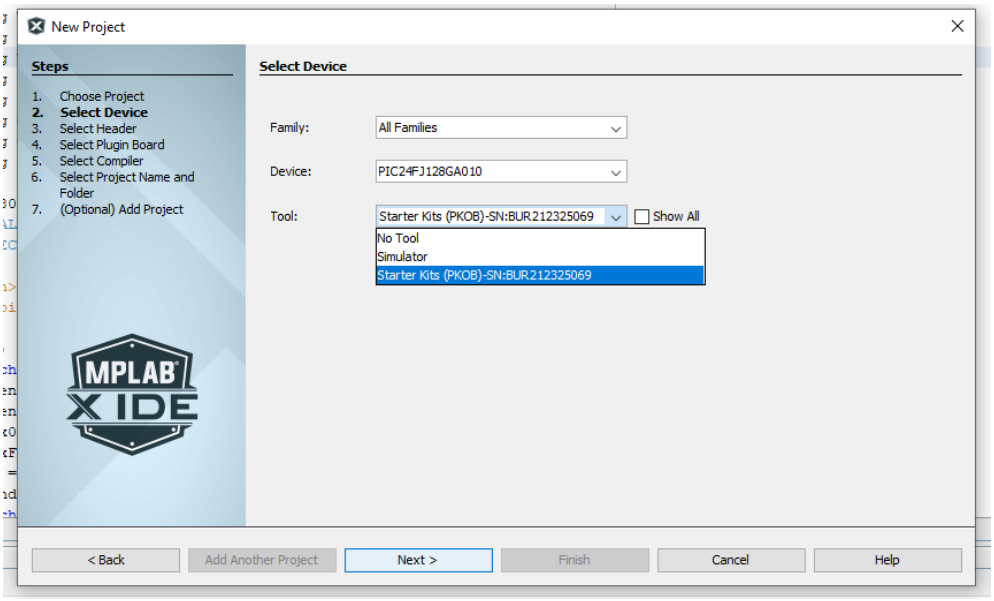


2. Wybieramy w Categories „Microchip Embedded” oraz w Projects „Application Project(s)” .

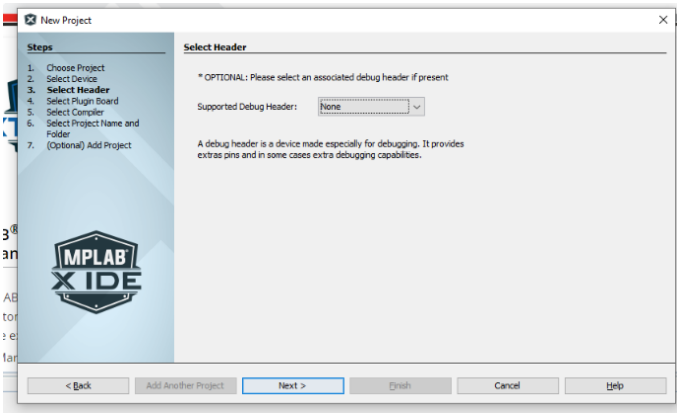


3. W okienku Device wybieramy model naszego mikrokontrolera „PIC24FJ128GA010” oraz w okienku Tool „Starter Kits...”.

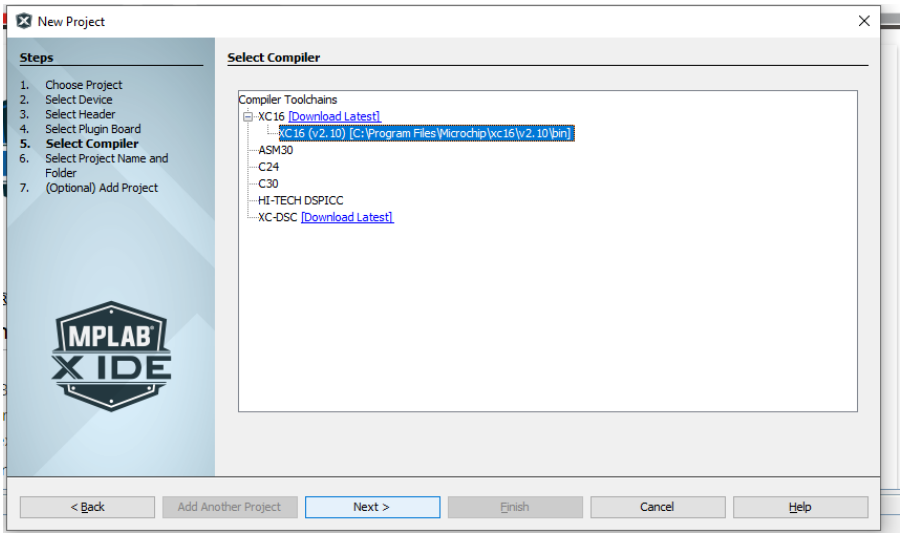




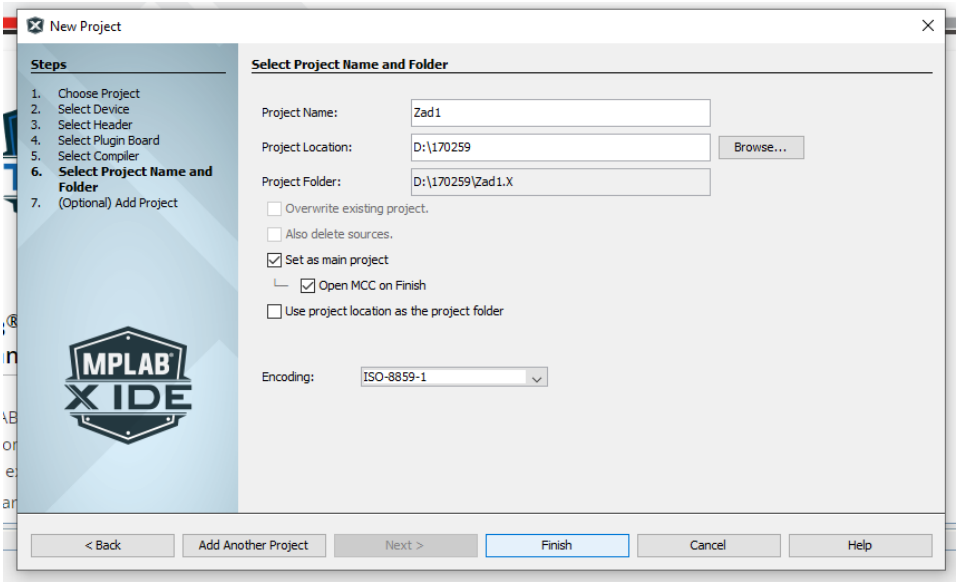
4. W supported debug header wybieramy ,,None”.



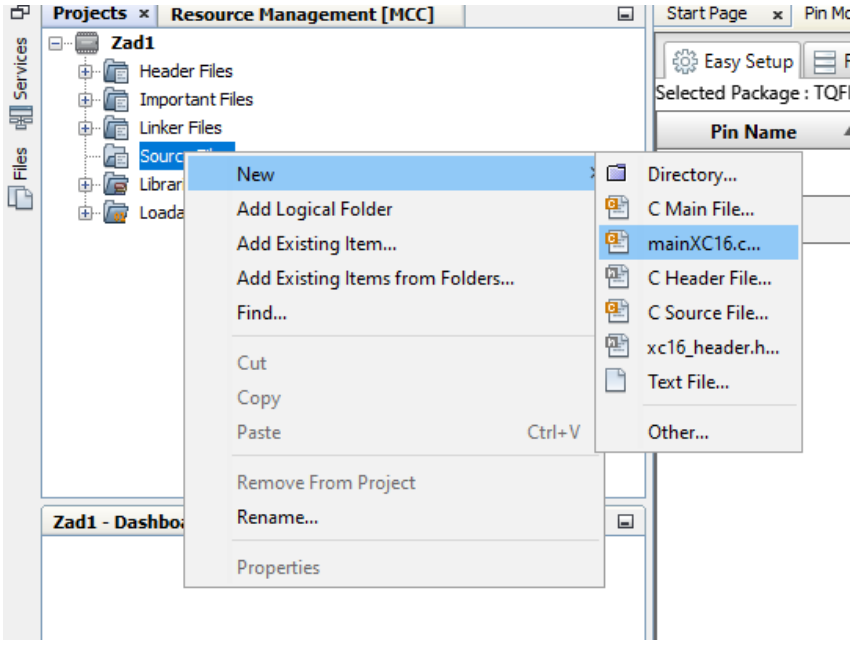
5. Wybieramy XC16 i wybieramy wersję którą potrzebujemy.



6. Nazywamy projekt oraz wybieramy lokalizację projektu. Warto również zaznaczyć opcję „Set as main project”.

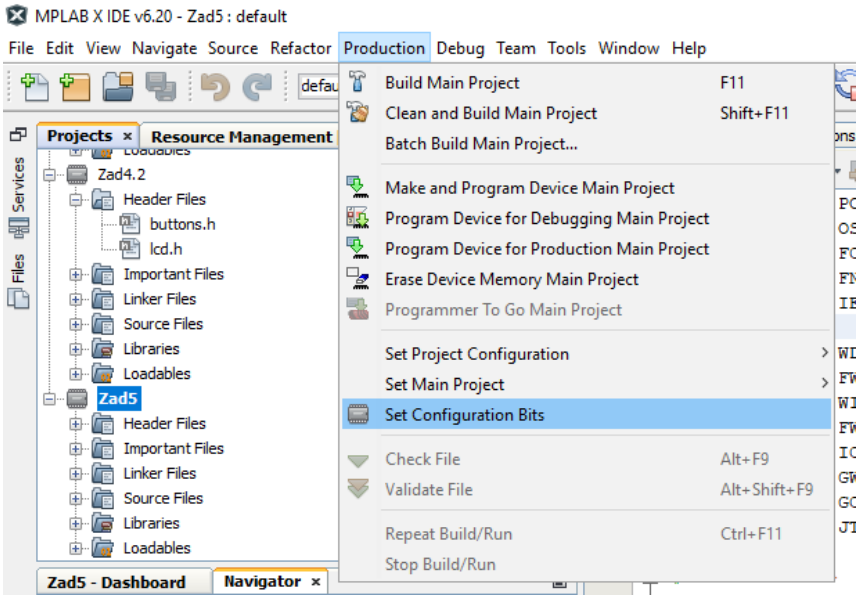


7. Tworzymy plik „main” w którym będzie kodować programy.



Konfiguracja bitów

1. Wybieramy zakładkę „Production” a następnie „Set Configuration Bits”.



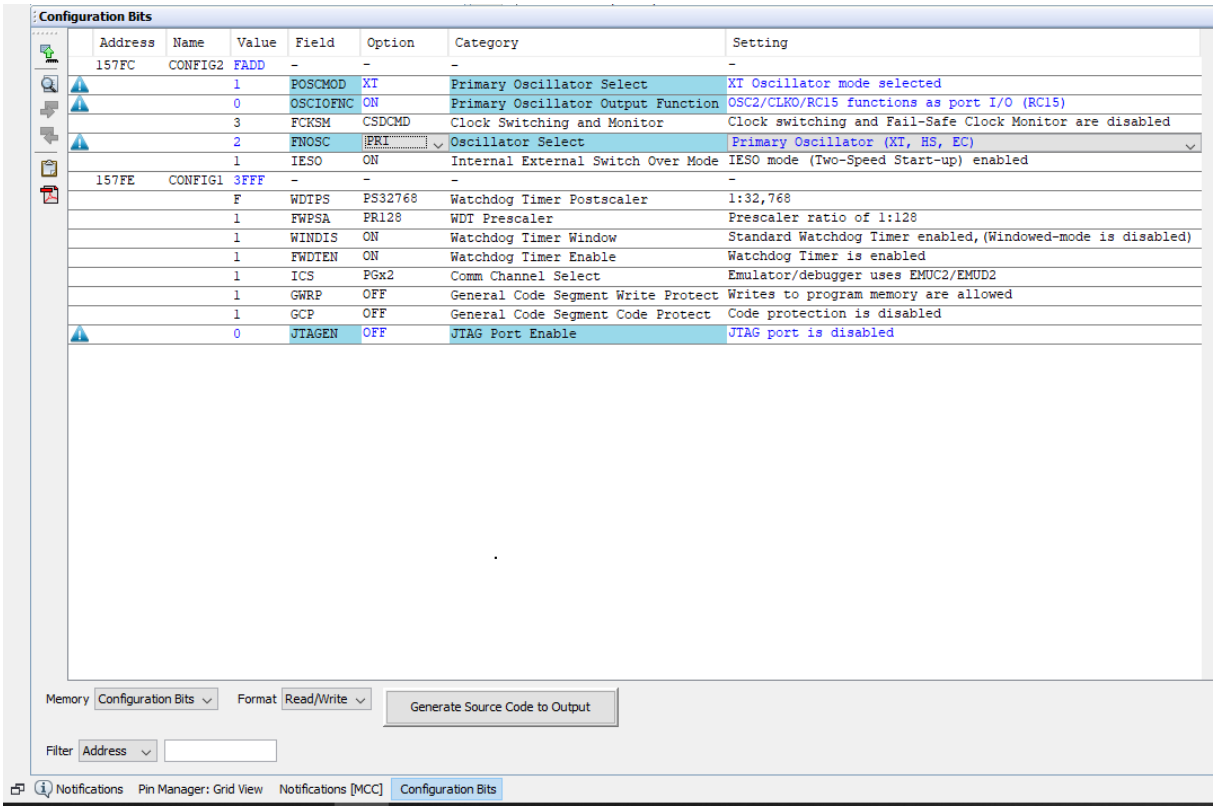
2. Ustawiamy:

   POSCMOD na „XT”

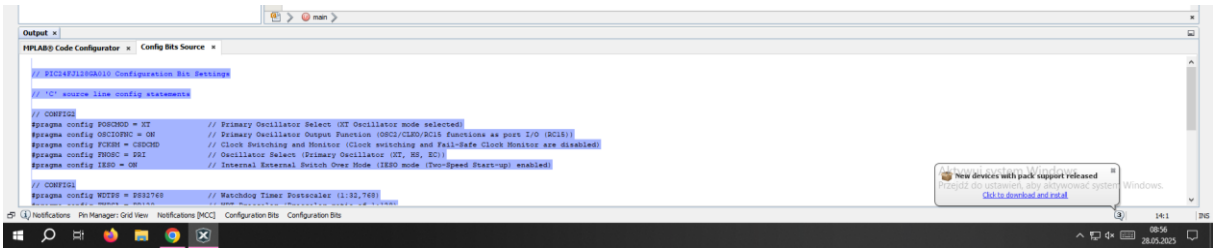
   OSCIOFN na „ON”

   FNOSC na „PRI”

   JTAGEN na „OFF” i wybieramy „Generate Source Code to Output”



3. Kopiujemy wygenerowany kod i wklejamy do naszego pliku „main” na samą górę projektu. Wygenerowany kod znajduje się na dole ekranu w zakładce Output dokładnie „Config Bits Source”.



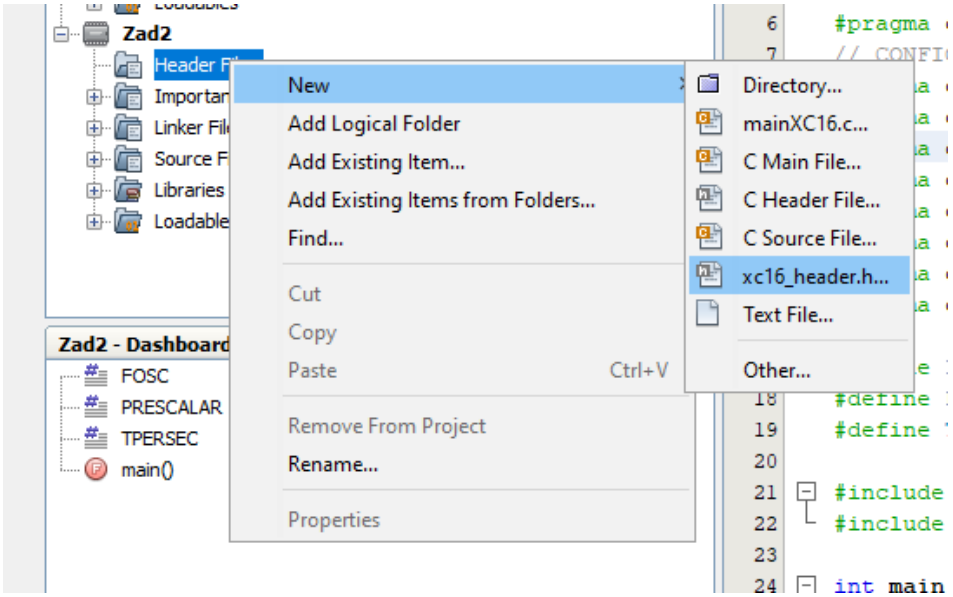
Dodanie plików adc, lcd oraz buttons

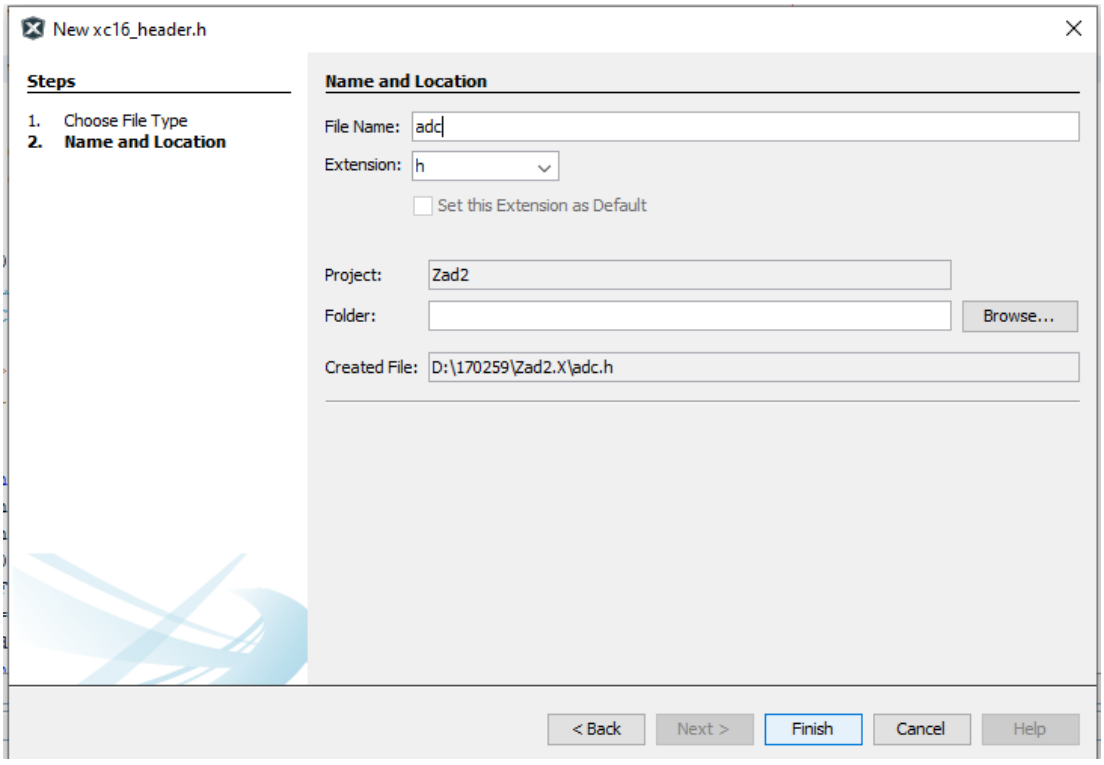
1. Pliki z rozszerzeniem .h dodajemy do katalogu „Header Files”.
2. Pliki z rozszerzeniem .c dodajemy do katalogu „Source Files”.
3. Pamiętaj o zaimportowaniu danego pliku.

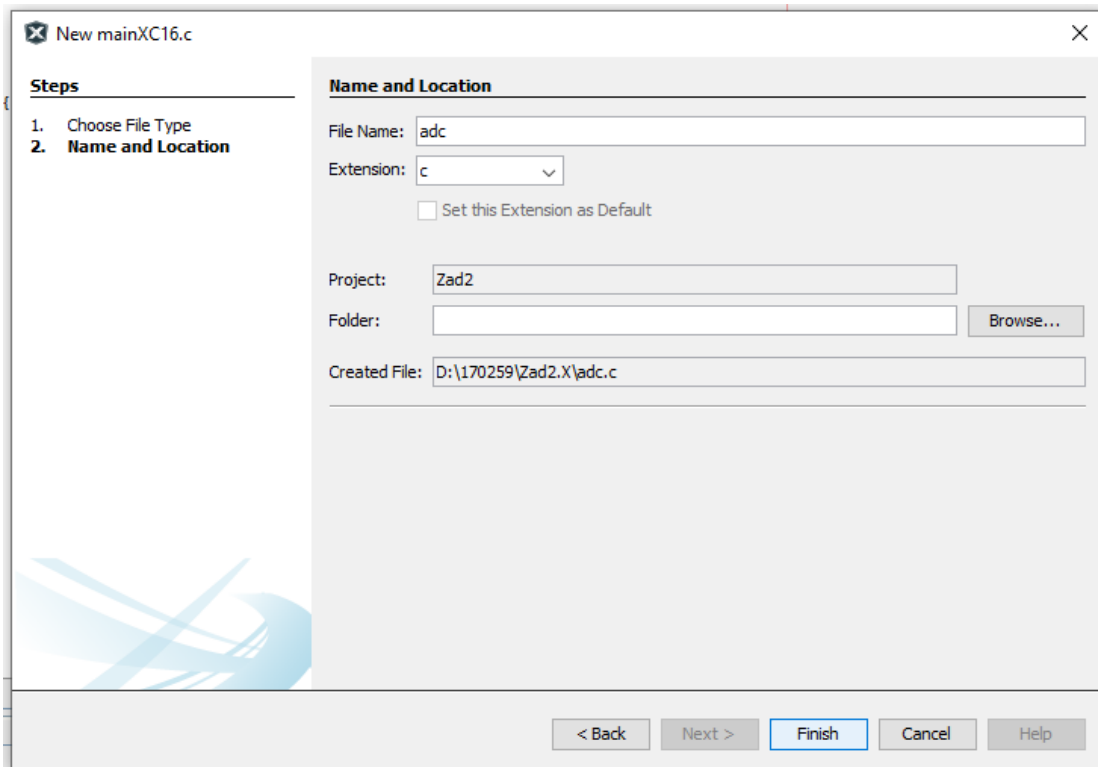
Importowanie plików z rozszerzeniem .h

```
#include "lcd.h"

#include "buttons.h"

#include "string.h"
```

#include "adc.h"







Przydatne biblioteki

Biblioteka	Opis
#include <xc.h>	Główna biblioteka: rejestry, konfiguracja bitów, dostęp do sprzętu.
#include <libpic30.h>	Biblioteka pomocnicza: __delay_ms(), __delay_us(), unlock/lock.
#include <stdint.h>	Typy liczbowe: uint8_t, int16_t, uint32_t itd.
#include <stdbool.h>	Typ logiczny bool, true, false.
#include <math.h>	Funkcje matematyczne: sqrt(), sin(), cos() itd.
#include <string.h>	Funkcje tekstowe: strcpy(), strlen(), memcmp() itd.
#include <stdio.h>	(Opcjonalnie) dla printf() – jeśli biblioteka I/O jest dostępna.

Opóźnienie i świecenie diod

Rozdział poświęcony wyjaśnieniu funkcji rejestrów (odnośnie diod) i instrukcji opóźnienia.

TRISA = 0x0000; → wszystkie piny portu A są wyjściami, dzięki czemu możemy zapalać diody

LATA = wartość → ustawia stan 0 lub 1 na pinac portu A, 0 – dioda nie świeci, 1 – dioda świeci

LATA = 0xFFFF // wszystkie diody świecą ,

LATA = 0x0000 // wszystkie diody są zgaszone,

LATA = 0x0001 // pierwsza dioda świeci

AD1PCFG = 0xFFFF // Wszystkie piny analogowe konwertujemy na cyfrowe

Opóźnienie jest ważne, gdyż wszystkie nasze polecenia w kodzie robią się w milisekundach przez co w przypadku świecenia diod, gdy świecimy a potem je gasimy i znowu zapalamy i nie użyjemy opóźnienia możemy nawet nie zaobserwować tej zmiany.

__delay32(1000000); // komenda odpowiedzialna za opóźnienie

Poprawne świecenie diod:

__delay32(1000000);

LATA = 0x0000;

__delay32(1000000);

LATA = 0xFFFF;

Gdy fizycznie naciskasz przycisk, styki nie łączą się idealnie, tylko drgają przez kilka milisekund – mogą wysyłać wiele szybkich impulsów zamiast jednego czystego sygnału.

Dlatego usuwa się drgania styków („debounce”) poprzez dodania opóźnienia np.:

prev6 = PORTDbits.RD6;

__delay32(1000000); // opóźnienie ~100 ms przy 8 MHz i preskalerze

current6 = PORTDbits.RD6;

Potencjometr

Rozdział poświęcony obsłudze potencjometru.

#include "adc.h" // dodanie biblioteki do potencjometru

ADC_SetConfiguration(ADC_CONFIGURATION_DEFAULT);

ADC_ChannelEnable(ADC_CHANNEL_POTENTIOMETER); // inicjalizacja potencjometru

adc_value = ADC_Read10bit(ADC_CHANNEL_POTENTIOMETER // odczytuje wartość potencjometru

if (adc_value == 0xFFFF){continue;} // Wartość 0xFFFF oznacza błąd odczytu (sprawdzenie poprawności działalności potencjometru)

Przyciski

1. Podejście bez plików buttons.

Pierwsze co musimy zrobić to ustawić port D jako wejście (inicjalizacja przycisków).

TRISD = 0xFFFF // Ustawia cały port D jako wejście

Następnie deklarujemy wartości żeby móc odczytywać stany przycisków.

Tutaj zaprezentuje 2 przyciski: RD6 oraz RD13

char current6 = 0, prev6 = 0;   // dla przycisku RD6

char current13 = 0, prev13 = 0;   // dla przycisku RD13

Odczyt wartości dla przycisku RD6:

prev6 = PORTDbits.RD6;

__delay32(1000000);   // małe opóźnienie antydrganiowe (warto dodać)

current6 = PORTDbits.RD6;

Musimy stworzyć pętle if żeby wiedzieć czy przycisk został wciśnięty:

if (prev6 == 1 && current6 == 0) // 1 → nie wciśnięty przycisk , 0 → wciśnięty

Fragment kodu żeby zwiększyć wartość zmiennej np. żeby zmienić program:

```
if (prev6 == 1 && current6 == 0) {
  value++;              // przełącz na następny tryb
  if (value >= 7) value = 1;
  portValue = 1;        // resetuj wartość startową
}
```

2. Obsługiwanie przycisków z pomocą plików buttons.

Dzięki plikom buttons inicjalizowanie jak i odczytywanie czy przycisk został naciśnięty staje się prostsze.

Pierwsze co robimy to inicjalizujemy przyciski.

```
BUTTON_Enable(BUTTON_S6);
BUTTON_Enable(BUTTON_S3);
BUTTON_Enable(BUTTON_S4);
BUTTON_Enable(BUTTON_S5);
```

Następnie żeby odczytać wartość czy przycisk został naciśnięty wystarczy

Prosta funckja BUTTON_IsPressed(BUTTON_S6).

Ekran LCD

Na początku musimy zainicjalizować nasz ekran LCD.

LCD_Initialize();

Na ekranie LCD możemy wyświetlać liczby, słowa oraz znaki specjalne.

Żeby wyświetlić na ekranie pojedyncze znaki lub przechodzić do nowych linii służy ta funkcja:

LCD_PutChar('\n'); // przejście do nowej linii

LCD_PutChar(0x55); // wstawienie U

Żeby wyświetlić słowo lub zdanie używamy tej funkcji:

LCD_PutString(”napis’’ , 5); // wstawienie słowa „napis”, po przecinku podajesz liczbę znaków argumentu

Żeby wyczyścić ekran:
LCD_ClearScreen();