Akademia Górniczo-Hutnicza
w Krakowie
Katedra Elektroniki
WIET

Laboratorium
Techniki Mikroprocesorowej 2

Ćwiczenie 1

Wprowadzenie do środowiska Keil µVision i platformy FRDM-KL05Z

Autor: Sebastian Koryciak

ver. 12.10.2020

1. Cel ćwiczenia

Niniejsze ćwiczenia mają na celu przeprowadzenie Uczestników laboratorium przez utworzenie pierwszego projektu w środowisku µVision. Na początku przedstawiono krótko charakterystykę zestawu FRDM-KL05Z, a następnie etapy tworzenia, programowania, debugowania (w zakresie podstawowym) i symulowania programu napisanego w języku C.

Punkty 1.1 i 1.2 należy wykonać jednorazowo tylko jeżeli na danym komputerze nie ma wymaganego oprogramowania lub sterowników.

1.1. Instalacja oprogramowania

W celu rozpoczęcia wykonywania zadań należy zainstalować następujące programy i sterowniki:

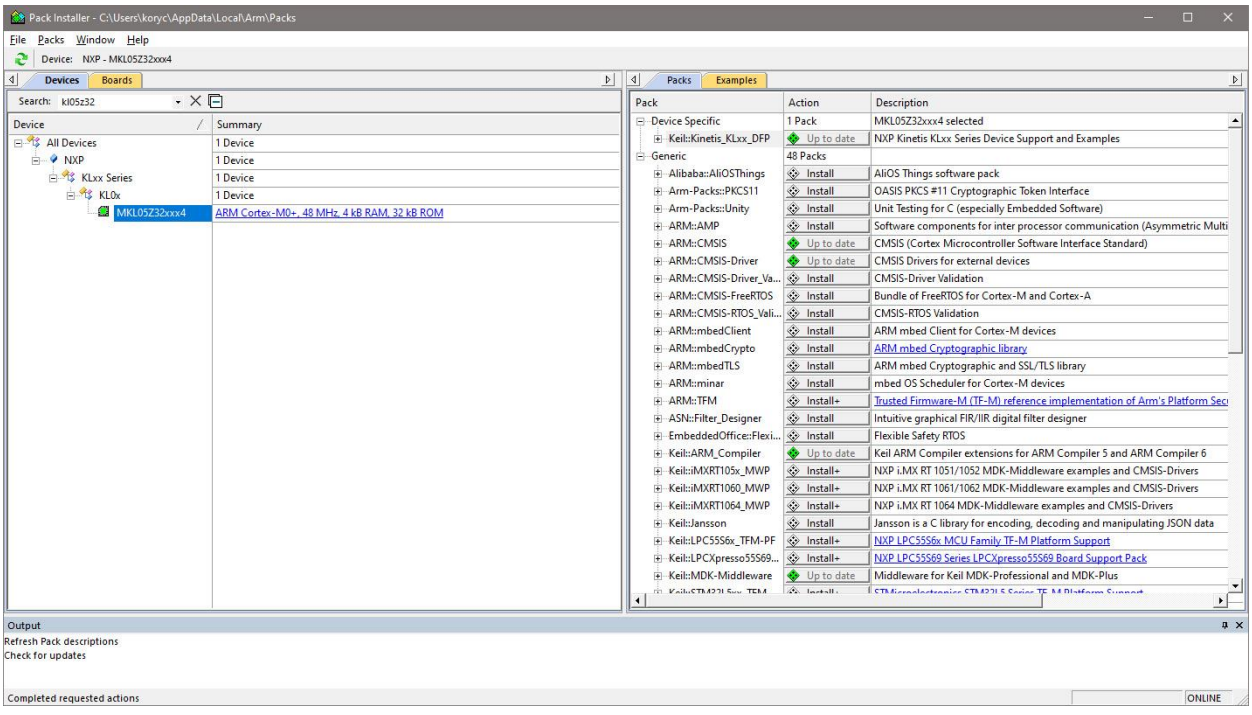
Keil µVision: *https://www.keil.com/demo/eval/arm.htm*
(wersja demo ogranicza wielkość programu do 32kB, co nie przeszkadza w wykonaniu wszystkich ćwiczeń i większości projektów)

Sterownik Segger OpenSDA: *https://www.segger.com/downloads/jlink/JLink_Windows.exe*
(pozwala na komunikację z platformą przy pomocy programu µVision)

1.2. Instalacja dodatkowych pakietów µVision

Po prawidłowej instalacji środowiska Keil automatycznie zostanie otwarte okno Pack Installer. W przeciwnym wypadku należy uruchomić program uVision , a następnie odnaleźć i nacisnąć ikonę o symbolu umieszczonym obok.

Następnie należy wpisać w okienku ‘Search’ znajdującym się po lewej stronie nazwę wykorzystywanego procesora, w naszym przypadku ‘kl05z32’. W oknie po prawej stronie pojawią się możliwe do zainstalowania pakiety. Należy upewnić się, że zainstalowane („Up to date”) są następujące pakiety: **Keil::Kinetis_KLxx_DFP** oraz **ARM::CMSIS**. Po prawidłowej instalacji okno ‘Pack Installer’ powinno wyglądać jak poniżej.



Rysunek 1. Wygląd okna ‘Pack Installer’ po zainstalowaniu wymaganych pakietów.

1.3. Styl kodowania i komentarzy

W trakcie wykonywania ćwiczeń oraz tworzenia projektu zalecane jest stosowanie się do poniżej umieszczonych reguł.

Formatowanie składni w języku ANSI C:
https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html

Komentowanie kodu:
https://www.doxygen.nl/manual/docblocks.html

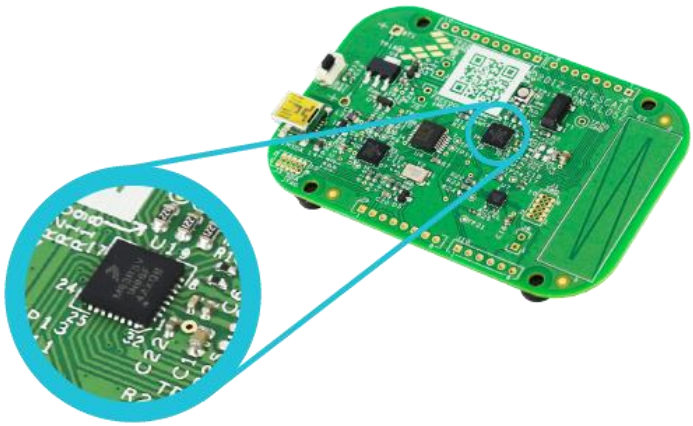
2. Platforma sprzętowa

Firma NXP zaprojektowała i wyprodukowała serię zestawów wyposażonych w nowoczesne mikrokontrolery z rdzeniem ARM Cortex-M0+ o zróżnicowanych możliwościach. Odpowiednio do możliwości zastosowanego mikrokontrolera dobrano moduły peryferyjne zamontowane na płytce drukowanej zestawu.

2.1. FRDM-KL05Z – co na pokładzie?

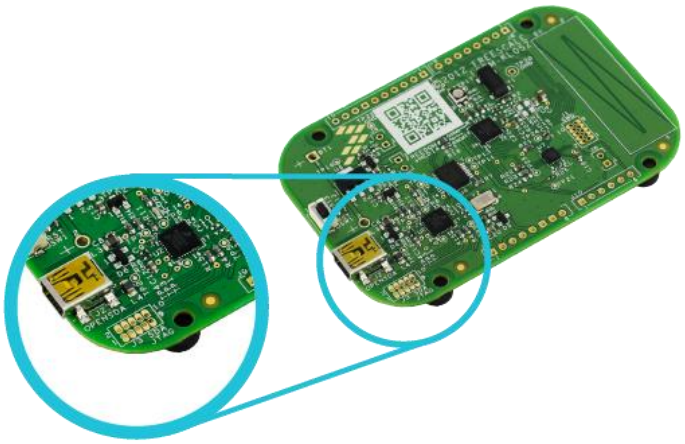
Zestaw FRDM-KL05Z jest wyposażony w mikrokontroler MKL05Z32VFM4. Należy on do rodziny Kinetis L z 32-bitowym rdzeniem Cortex-M0+. Na płytce znajdziemy procesory:

- 32 bitowy mikrokontroler MKL05Z32VFM4z rodziny Kinetis L zawierający:
  - 32 KB pamięci Flash,
  - 4 KB pamięci SRAM,
  - maksymalna częstotliwość taktowania rdzenia: 48 MHz,
  - zegar RTC,
  - układ Watchdog,
  - DMA,
  - 12-bitowy DAC,
  - 12-bitowy SAR ADC,
  - niskonapięciowy moduł UART,
  - moduł I2C,
  - moduł SPI,



Rysunek 2. Położenie procesora KL05Z.

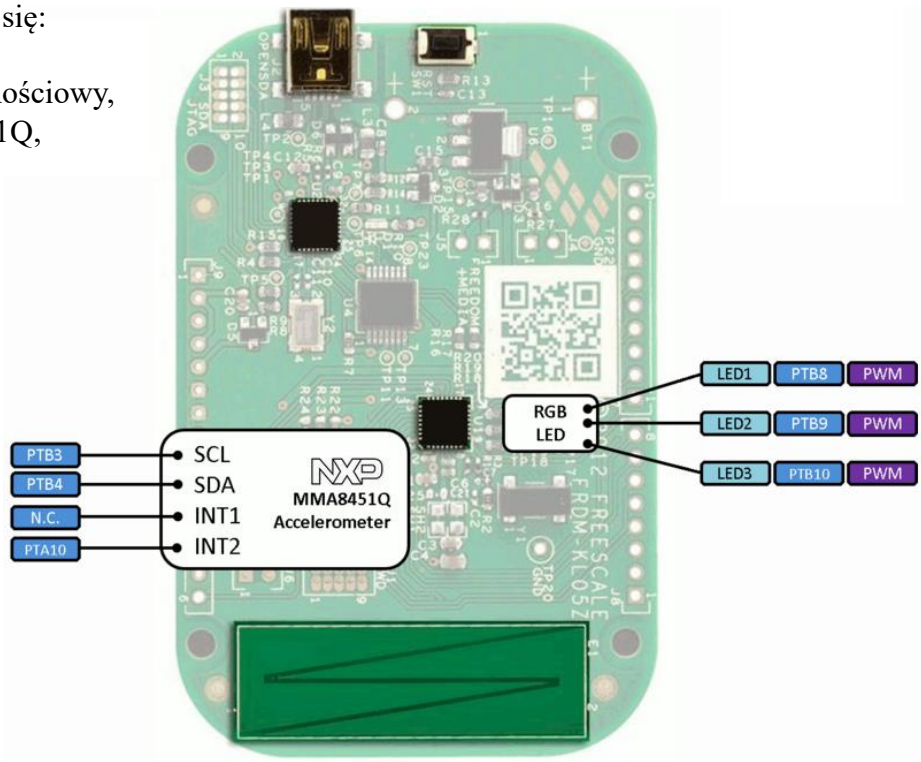
- 32 bitowy mikrokontroler PK20DX128VFM5 z rdzeniem Cortex-M4 pełniący rolę interfejsu OpenSDA (programowanie i debugowanie).



Rysunek 3. Położenie interfejsu OpenSDA.

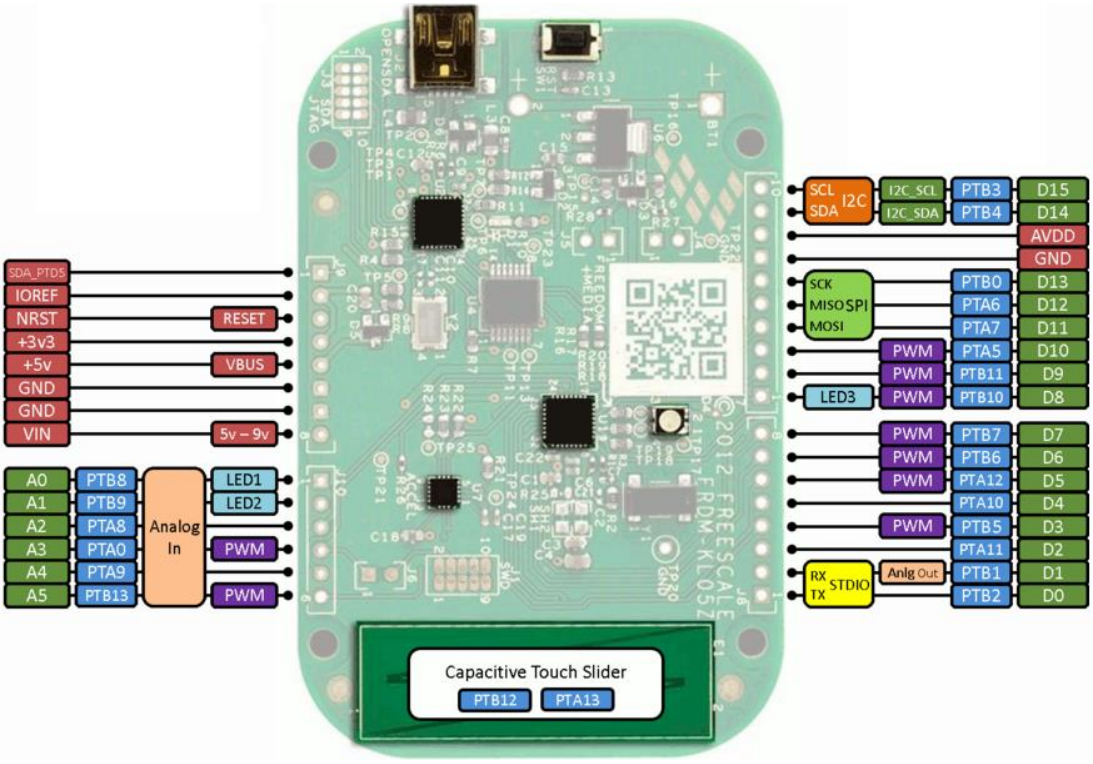
Na płytce ponadto znajdują się:

- dotykowy slider pojemnościowy,
- akcelerometr MMA8451Q,
- dioda LED RGB,
- przycisk Reset.



Rysunek 4. Położenie diody LED RGB i akcelerometru.

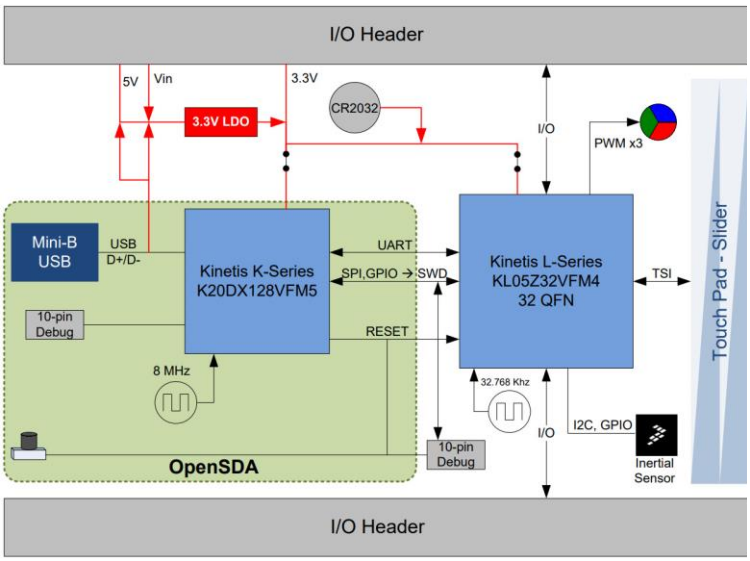
Platforma ewaluacyjna wyposażona jest w złącza szpilkowe zgodne ze standardem ARDUINO.



Rysunek 5. Położenie i oznaczenie złącz ARDUINO oraz płytki dotykowej.

2.2. Driver i bootloader

OpenSDA (Open Serial Debug Adapter) to otwarty interfejs szeregowy i debuger.



Rysunek 6. Połączenie głównych komponentów na platformie.

Pozwala on na komunikację szeregową pomiędzy hostem, a docelowym mikrokontrolerem. Interfejs OpenSDA w zestawie Freedom zrealizowano z wykorzystaniem mikrokontrolera z rodziny Kinetis K20 charakteryzującego się m.in. pamięcią Flash o dużej pojemności oraz zintegrowanym kontrolerem USB. W przestrzeni adresowej pamięci Flash jest zawarte wydzielone Urządzenie Pamięci Masowej (MSD). Pełni ono rolę bootloadera, który pozwala na łatwe i szybkie ładowanie aplikacji do pamięci mikrokontrolera docelowego.

Na platformę jest wgrany bootloader OpenSDA w wersji firmy Segger, kompatybilny z J-Link. OpenSDA w tej wersji pozwala na:

- programowanie procesora docelowego,
- debugowanie z 2 sprzętowymi i większą ilością programowych pułapek,
- komunikowanie się z procesorem docelowym za pomocą emulowanego portu COM.

Więcej informacji na temat drivera można odnaleźć na stronie:

https://www.segger.com/products/debug-probes/j-link/models/other-j-links/opensda-sda-v2/

Pracę interfejsu OpenSDA sygnalizuje **zielona dioda LED** umieszczona zaraz obok procesora z rodziny Kinetis K20. Zachowania diody:

- miganie diody z częstotliwością około 10Hz – do platformy podłączono przez port USB samodzielne źródło zasilania (np. powerbank),
- świecenie diody ciągłe – platforma podłączona przez port USB do komputera.

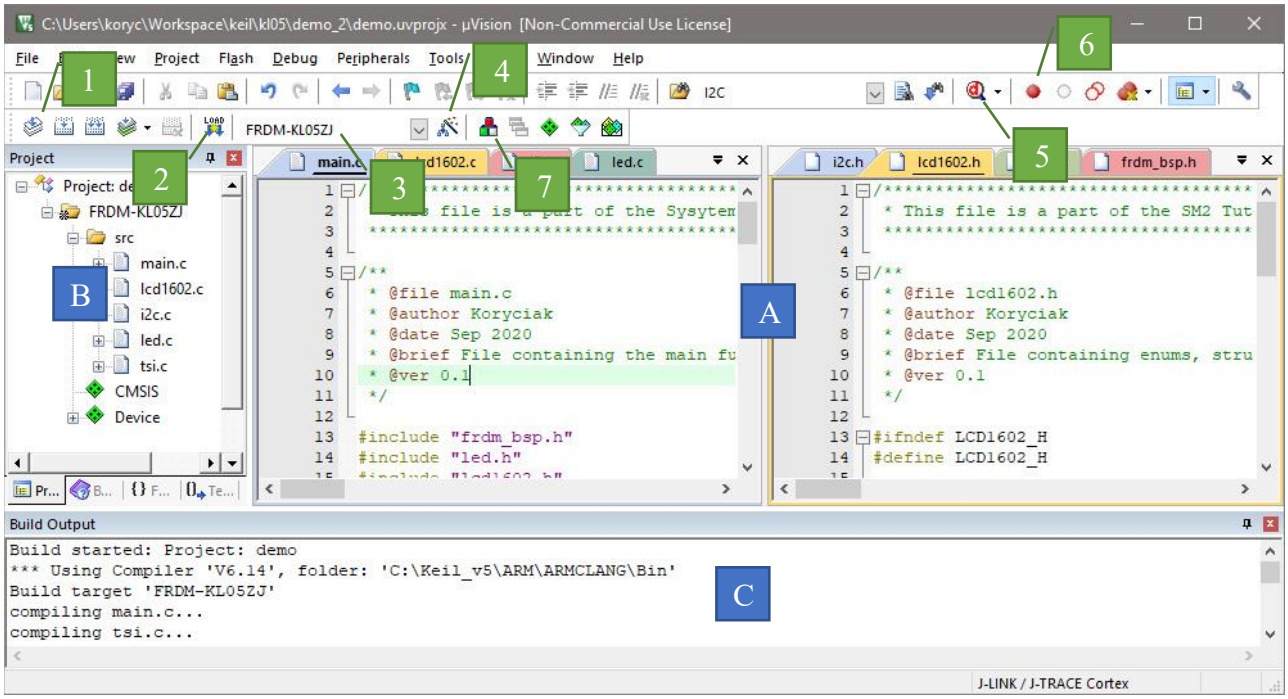
Ponadto podłączona platforma do komputera przy pomocy portu USB zgłasza się jako dwa urządzenia:

- urządzenie pamięci masowej pod nazwą FRDM-KL05ZJ,
- wirtualny port COM, którego numer sprawdzamy w następujący sposób (WIN10): *Start > Panel sterowania > Menadżer urządzeń > Porty (COM i LPT) > JLink CDC*.

UWAGA! Jeżeli platforma zachowuje się w inny niż opisany powyżej sposób proszę samodzielnie wgrać firmware OpenSDA od Segger lub skontaktować się z prowadzącym zajęcia.

3. Środowisko Keil µVision

Program µVision jest środowiskiem stworzonym przez firmę Keil (ARM) przeznaczonym do programowania i debugowania mikrokontrolerów opartych o architekturę ARM.



Rysunek 7. Główne okna i elementy programu.

Program składa się z okien:

A. Edytor (istnieje możliwość podglądu większej ilości plików, w tym przypadku 2),
B. Okno ze strukturą projektu,
C. Konsola z informacjami o postępie kompilacji, programowania, debugowania.

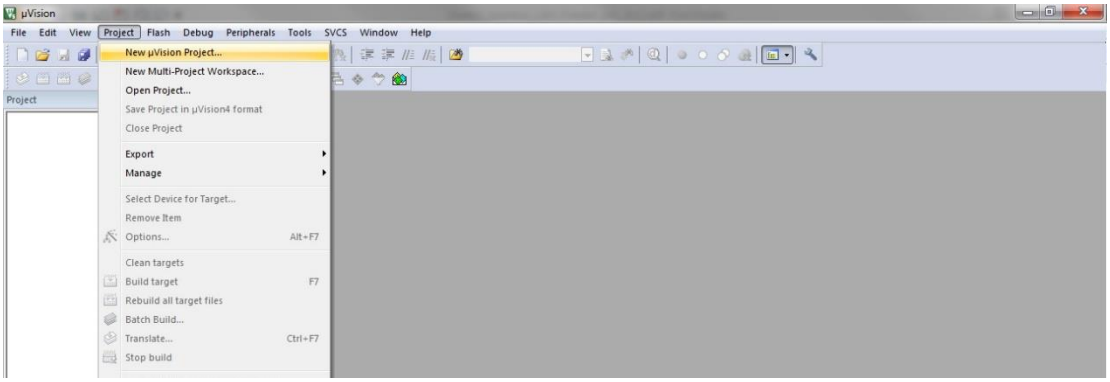
Ponadto najczęście wykorzystywane będą ikony:

1. Zbuduj projekt (F7),
2. Wgraj zbudowany program do pamięci flash (F8),
3. Wybierz obiekt do którego tworzony jest projekt,
4. Opcje dla danego obiektu (kompilator, linek, debuger, ...),
5. Włącz debugowanie (Ctrl+F5),
6. Wrzuć w miejscu gdzie się znajduje kursor pułapkę.
7. Zarządzanie obiektami i katalogami projektu.

3.1. Pierwszy projekt

Proszę uruchomić aplikację µVision i stworzyć projekt w następujący sposób:

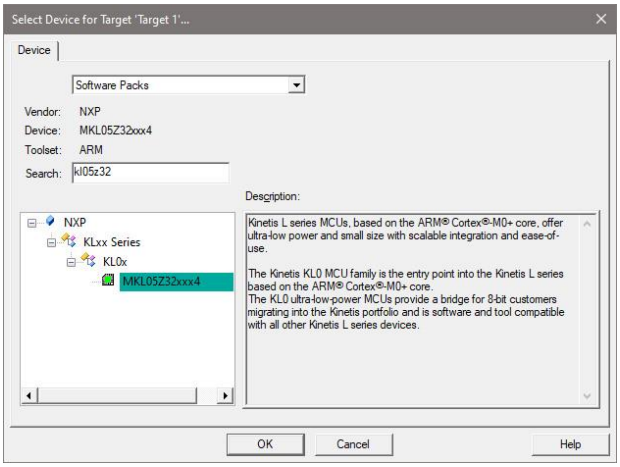
1. Z menu aplikacji proszę wybrać Project > New uVision Project …



Rysunek 8. Opcja tworzenia nowego projektu.

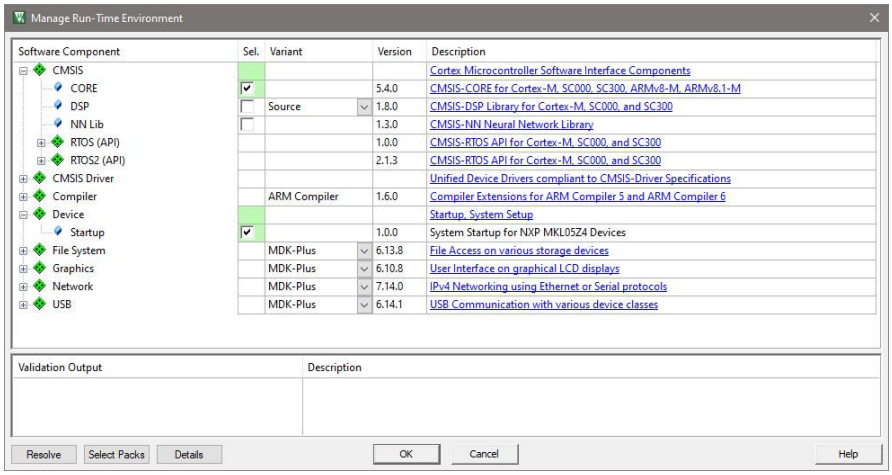
2. Proszę wskazać ścieżkę dostępu do nowo tworzonego projektu (nowy katalog, np. "*C:/MT2_Tutorials/Tut1/led_blink/*"). Projekt nazwać *led_blink.*

3. Kolejnym krokiem jest wybór mikrokontrolera na którym będziemy pracować. Wyszukujemy 'MKL05Z32xxx4' korzystając z opcji 'Search'.



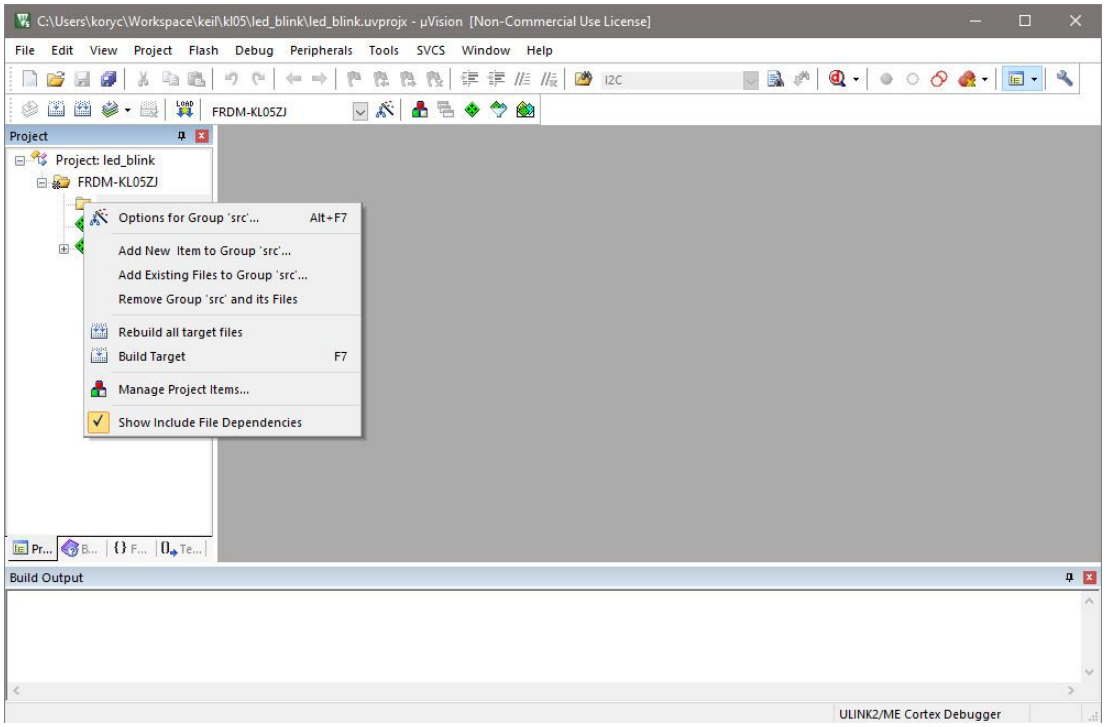
Rysunek 9. Okno wyboru procesora.

4. Nowo otwarte okno służy do wyboru bibliotek. Do tworzonego projektu będziemy potrzebowali: CMSIS > CORE, oraz Device > Startup. Zaznacz jak na rysunku poniżej.



Rysunek 10. Okno wyboru bibliotek.

5. Nazwę 'Target1' zmieniamy na 'FRDM-KL05ZJ', natomiast 'Source Group 1' na 'src'.
6. Następnie klikając prawym przyciskiem myszy na folder 'src' wybieramy opcję „Add New Item to Group 'src'...".



Rysunek 11. Widok świeżo stworzonego projektu led_blink.

7. W nowo otwartym oknie wybieramy język C, oraz wpisujemy nazwę pliku „main.c".

4. Pierwszy program – miganie czerwoną diodą

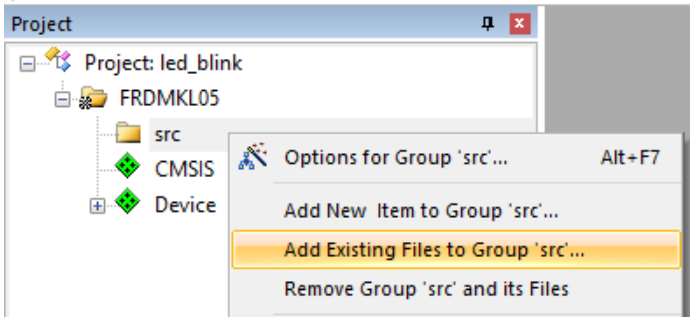
W celu stworzenia i uruchomienia pierwszego programu potrzebne będą dodatkowe pliki. Proszę odnaleźć archiwum *„tut1_files.zip"* umieszczone w tym samym miesjcu co instrukcja.

1. Z rozpakowanego archiwum *„tut1_files.zip"* proszę skopiować bezpośrednio do katalogu projektu (np. "*C:/MT2_Tutorials/Tut1/led_blink/*") następujące pliki: "*frdm_bsp.h*", "*led.c*", "*led.h*",
2. Proszę stworzyć swój pierwszy program dodając następującą treść do pliku main.c:

```
1  /**
2   * @file main.c
3   * @author Koryciak
4   * @date Sep 2020
5   * @brief File containing the main function.
6   * @ver 0.1
7   */
8
9  #include "led.h"
10
11 /**
12  * @brief Simple delay
13  * @param c - delay control
14  * @return NULL
15  */
```

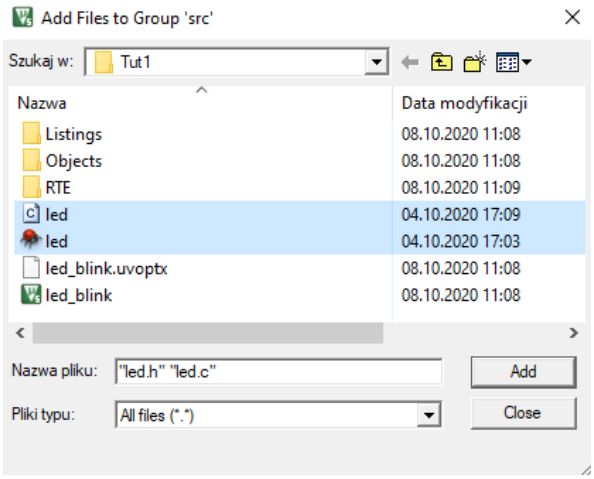
```
16  void delay(unsigned char d){
17
18        for(volatile int i=0; i<(5000*d); i++);
19
20  }
21
22  /**
23   * @brief The main loop.
24   *
25   * @return NULL
26   */
27  int main (void) {
28  volatile unsigned char c = 0;
29
30    LED_Init ();        /* initialize all LEDs */
31
32    while(1){
33              c += 51;
34              LED_Ctrl(LED_RED, LED_ON);
35              delay(c);
36              LED_Ctrl(LED_RED, LED_OFF);
37              delay(c);
38        }
39  }
40
```

3. Następnie klikając prawym przyciskiem myszy na folder 'src' (w programie Keil) wybieramy opcję „Add Existing Files to Group 'src'...” i dodajemy wszystkie pliki z rozszerzeniem *.c.



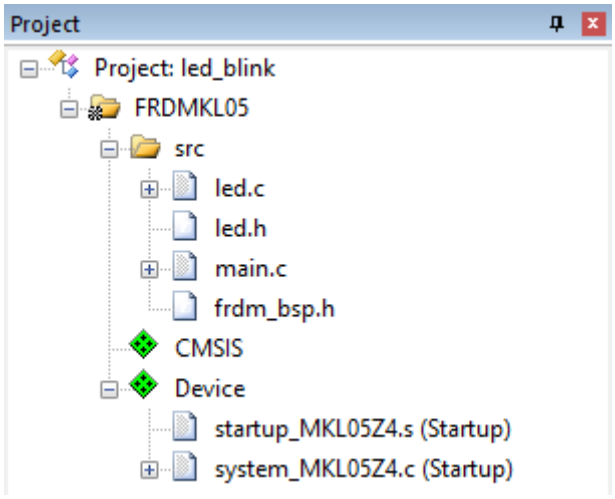
Rysunek 12. Sposób dodawania istniejących plików do projektu.

4. Proszę dodać do projektu pliki “*frdm_bsp.h*”, “*led.c*” i “*led.h*”.



Rysunek 13. Okno wyboru źródeł do projektu.

Po prawidłowym wykonaniu powyższych poleceń struktura projektu powinna wyglądać jak na rysunku poniżej:



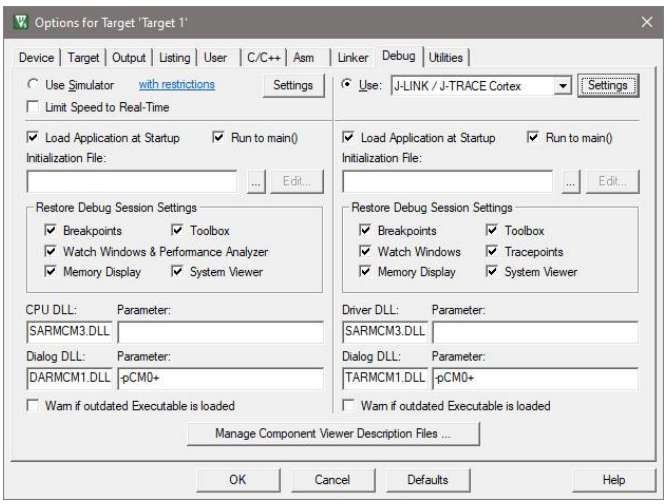
Rysunek 14. Wygląd okna projektu led_blink.

Proszę przeanalizować zawartość pliku „main.c".

Dioda RGB umieszczona na płytce wymaga inicjalizacji, która jest przeprowadzana przez funkcję LED_Init(). Funkcja LED_Ctrl() pozwala na włączenie lub wyłączenie wybranego koloru diody, dlatego przyjmuje dwa argumenty. Przykładowo, aby zaświecić zieloną diodę należy wpisać parametry do funkcji: LED_Ctrl(LED_GREEN, LED_ON).
…

4.1. Uruchomienie aplikacji led_blink

1. Po uzupełnieniu zawartości pliku main.c proszę zbudować projekt. Robimy to wybierając z menu Project > Build Target, bądź też przyciskając klawisz F7 (przycisk 1 z punktu 3). Następnie jednorazowo należy skonfigurować opcje debugera. W tym celu klikamy na ikonę Options for Target … (przycisk 4 z punktu 3).
2. W nowo otwartym oknie wybieramy zakładkę Debug, a w niej w opcji „Use" odnajdujemy „J-Link / J-TRACE Cortex".

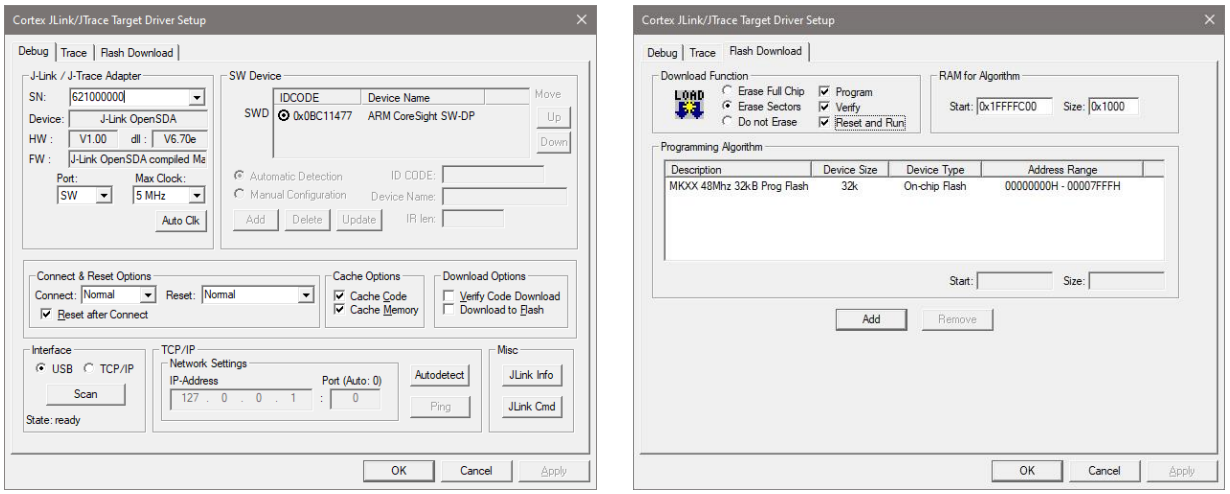


Rysunek 15. Okno opcji dla obiektu.

3. Następnie klikamy na Settings. Jeżeli robimy to pierwszy raz danego dnia w nowo otwartym oknie zaznaczamy opcję „Do not show …” i klikamy „Accept”.

Rysunek 16. Okno licencji J-Link.

4. W zakładce Debug zmieniamy „Port” na „SW”, a w zakładce Flash Download okna „Cortex Jlink…” zaznaczamy opcję „Reset and Run”, po czym akceptujemy przez „OK”.



Rysunek 17. Okna konfiguracji opcji debugera i wgrywania pamięci flash.

5. Po aktualizacji wszystkich ustawień odnajdujemy i klikamy na ikonę oznaczoną „LOAD” (przycisk 2 z punktu 3).

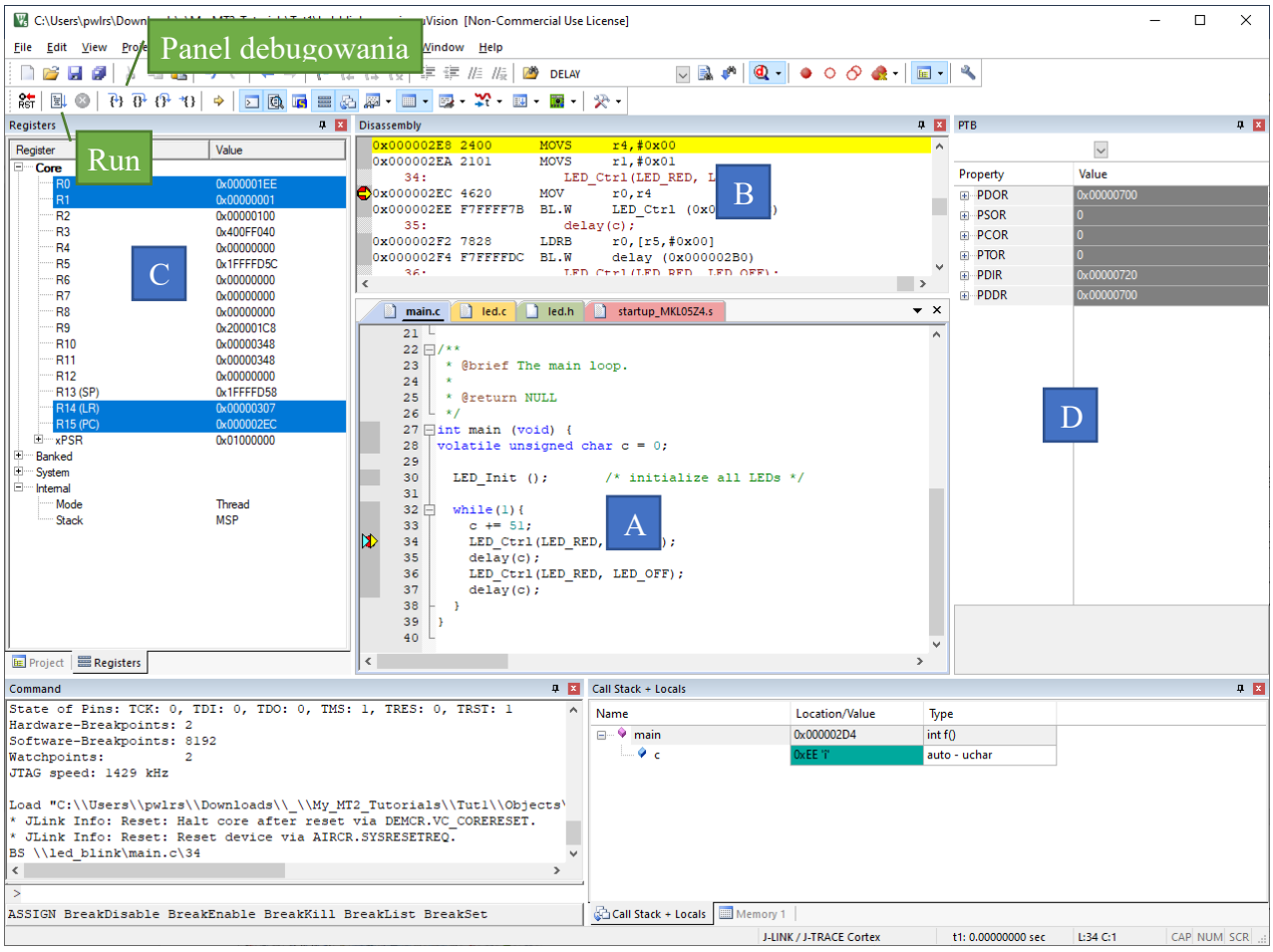
4.2. Debugowanie kodu

Przy programowaniu niezbędną funkcją jest możliwość debugowania kodu. Dlatego też teraz sprawdzimy, jak działa nasz pierwszy program.

1. W pliku main.c proszę najechać kursorem na 34 linię kodu, a następnie nacisnąć F9 (przycisk 6 z punktu 3). W ten sposób do kodu wstawiliśmy tzw. Breakpoint.
2. Następnie w celu rozpoczęcia sesji debugowania proszę nacisnąć kombinację klawiszy Ctrl+F5 (przycisk 5 z punktu 3).
3. Proszę otworzyć zawartość portów GPIO: Peripherals > System Viewer > GPIO > PTB.

Perspektywa programu do debugowania składa się z okien:

A. Kod naszego programu w C,
B. Kod naszego programu po kompilacji w języku ASM,
C. Okno z aktualną zawartością rejestrów procesora,
D. Podgląd wybranych przez nas rejestrów peryferyjnych.



Rysunek 18. Widok perspektywy debugowania.

1. Proszę parokrotnie nacisnąć przycisk ‘Run’ (F5) i sprawdzić jakie zmiany następują w poszczególnych rejestrach modułu PTB.
2. Krokować program możemy przy pomocy przycisku ‘Step’ (F11) znajdującym się na panelu debugowania.
3. Proszę wypróbować funkcje pozostałych przycisków z tego panelu (Step Over, Step Out, Run to...)
4. Wychodzimy z perspektywy debugowania drugi raz klikając na ikonę (przycisk 5 z punktu 3), lub za pomocą kombinacji Ctrl+F5.

4.3. Użycie panelu dotykowego i dołączanego wyświetlacza LCD

Poniższe zadanie polega na zbudowaniu projektu i aplikacji, dzięki której można wykorzystać następujące elementy:

RGB LED – diod w trzech różnych kolorach (led.c)

TSI – płytka dotykowa znajdująca się na platformie FRDM (tsi.c)

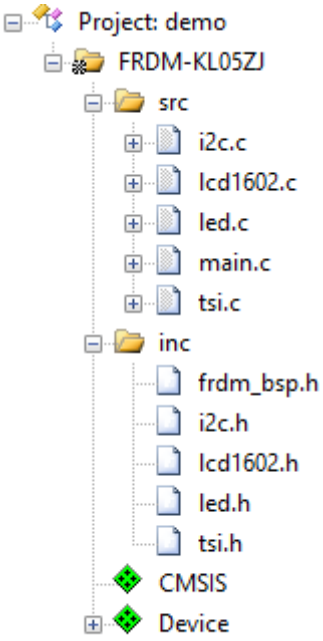
LCD 2x16 – wyświetlacz z interfejsem I2C (lcd1602.c, i2c.c)

Biblioteki zawierają podstawowe funkcje pozwalające rozpocząć pracę z poszczególnymi komponentami.

1. Proszę stworzyć nowy projekt o nazwie „lcd_demo” zgodnie z punktem 3.1 tej instrukcji.
2. Stwórz plik „main.c” zgodnie z opisem w sekcji 3.1.
3. Wklej poniższy kod do pliku „main.c”

```
/*****************************************************************
 * This file is a part of the LCD display Demo (C).              *
 *****************************************************************/

/**
 * @file main.c
 * @author Koryciak
 * @date Sep 2020
 * @brief File containing the main function.
 * @ver 0.1
 */

#include "frdm_bsp.h"
#include "led.h"
#include "lcd1602.h"
#include "tsi.h"

/**
 * @brief The main loop.
 *
 * @return NULL
 */
int main (void) {

      TSI_Init ();        /* initialize slider   */
      LED_Init ();        /* initialize all LEDs */
      LCD1602_Init();          /* initialize LCD */
      LCD1602_Backlight(TRUE);

      LED_Welcome();
      LCD1602_SetCursor(3,0);
      LCD1602_Print("- A G H -");
      LCD1602_SetCursor(3,1);
      LCD1602_Print("KE @ WIET");

      DELAY(1000)

      LED_Welcome();
      LCD1602_ClearAll();
      LCD1602_SetCursor(0,0);
      LCD1602_Print("Microprocessor");
      LCD1602_SetCursor(0,1);
      LCD1602_Print("Technology");

  while(1) if (TSI_ReadSlider() > 0) LED_Blink(LED_RED, 100);
}

```

4. Z rozpakowanego archiwum *„tut1_files.zip”* proszę skopiować bezpośrednio do katalogu projektu (np. “*C:/MT2_Tutorials/Tut1/lcd_demo/*”) wszystkie pliki.
5. Proszę dodać do projektu wszystkie pliki z podziałem na sekcje „src” i „inc”.



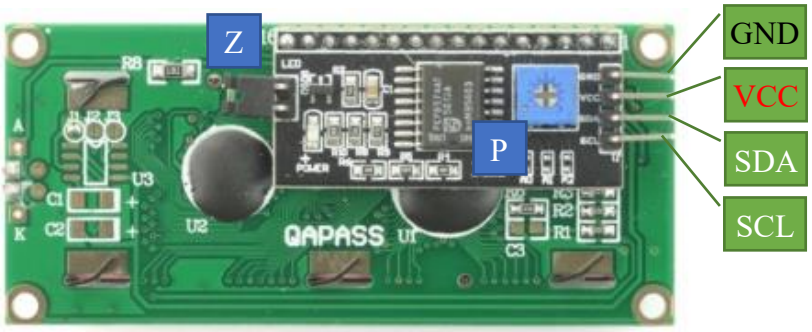
Rysunek 19. Wygląd okna projektu lcd_demo.

6. Proszę zbudować projekt (Project->Build Target lub F7).

Podłączenie zewnętrznego wyświetlacza LCD 2x16 do płytki FRDM-KL05.

1. Proszę podłączyć wyświetlacz LCD wykorzystując dołączone kabelki według poniższej mapy pinów (patrz Rysunki 5 i 20).

FRDM-KL05	**LCD**
PTB3	SCL
PTB4	SDA
+5V	VCC
GND	GND



Rysunek 20. Widok na wyświetlacz LCD 2x16 od tyłu.

2. Wyświetlacz LCD na odwrocie zawiera zworkę „LED” (oznaczoną ‘Z’ na Rysunku 20) włączającą podświetlenie ekranu, oraz potencjometr (niebieski element oznaczony ‘P’) służący do ręcznej regulacji kontrastu wyświetlanych znaków.

Uruchomienie aplikacji:

1. Proszę postępować zgodnie z punktem 4.1 w celu uruchomienia aplikacji na płytce.
2. Zmodyfikuj wyświetlany tekst i ponownie uruchom aplikację.
3. Proszę przetestować płytkę dotykową, a następnie zmodyfikować jej działanie w aplikacji. Proszę sprawdzić jakie wartości może zwracać funkcja TSI_ReadSlider. Następnie należy zmodyfikować główną pętlę programu (*while(1)*) tak, aby płytka dotykowa była podzielona na 3 przedziały, a każdy z nich w odpowiedzi na dotyk migał inną diodą.

UWAGA! W przypadku braku wyświetlania jakichkolwiek znaków na wyświetlaczu należy problem przeanalizować w następujący sposób:

a. sprawdź wszystkie połączenia realizowane kabelkami,

b. spróbuj wyregulować kontrast przy pomocy potencjometru,

c. zmień adres wyświetlacza LCD w kodzie (plik *lcd1602.c, linijka 23*)

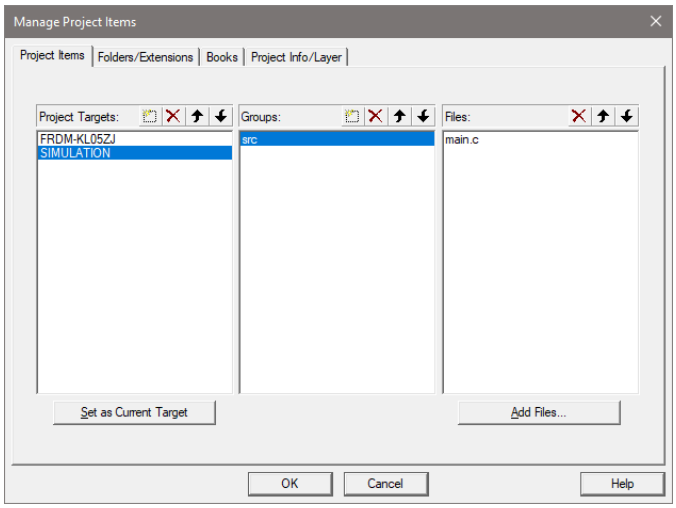
i. `#define PCF8574_ADDRESS 0x27 /* I2C slave device */`

ii. `#define PCF8574_ADDRESS 0x3F /* I2C slave device */`

5. Symulowanie kodu aplikacji *led_blink*

Często nie mamy dostępu do sprzętu w momencie pisania oprogramowania. W takich momentach przydaje się symulator.

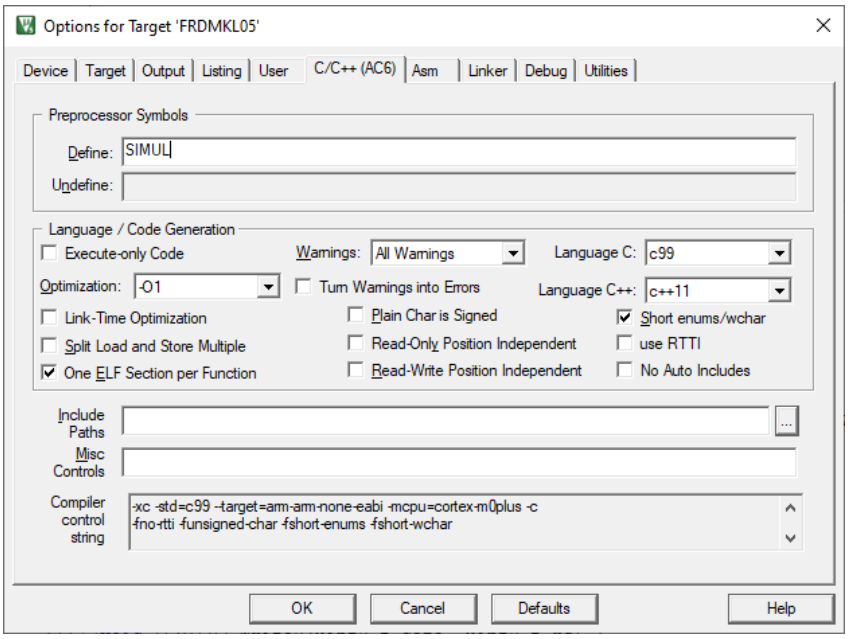
1. Proszę otworzyć projekt *led_blink*, a następnie w celu symulowania kodu stworzymy nowy obiekt. W tym celu wchodzimy do zarządzania elementami projektu (przycisk 7 z punktu 3), a tam dodajemy ‘Project Targets’ o nazwie ‘SIMULATION’.



Rysunek 21. Okno zarządzania elementami projektu.

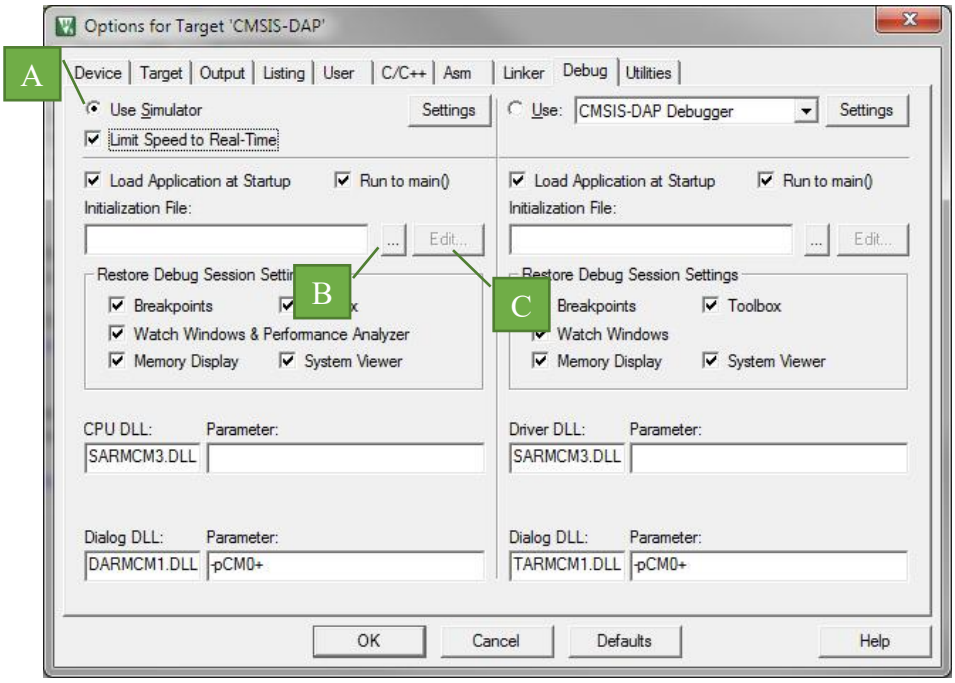
2. Po zmianie obiektu aktywnego na ‘SIMULATION’ (przycisk 3 z punktu 3), wchodzimy w konfigurację (przycisk 4 z punktu 3).

3. W zakładce „C/C++(AC6)” w sekcji ‘Preprocessor Symbols’ definiujemy ‘SIMUL’.



Rysunek 22. Okno opcji kompilatora.

4. W zakładce „Debug” wybieramy Use Simulator (A).



Rysunek 23. Okno opcji symulatora.

5. Obok okienka zatytułowanego „Initialization File” klikamy na ikonę trzech kropek (B), wpisujemy nazwę „init” i ją zatwierdzamy przyciskiem Open. Program zapyta, czy utworzyć plik o tej nazwie, co akceptujemy.
6. Nowo stworzony plik należy zmodyfikować. Dlatego też otwieramy go przyciskiem Edit (C). Okno „Options of Target …” możemy już zamknąć przyciskiem OK.
7. Do pliku init.ini wpisujemy:

```
MAP 0x40000000, 0x400FFFFF read write
```

8. Zawartość pliku należy zapisać, a następnie zlokalizować i otworzyć plik „system_MKL05Z4.c” (znajdziemy go w oknie Project, w pakiecie Device)

9. W otwartym pliku systemowym lokalizujemy funkcję ‘SystemInit’ i dodajemy kod na jej samym początku i końcu zgodnie z poniższym schematem (pogrubiony, linie 107, 193).

```
106  void SystemInit (void) {
107  #ifndef SIMUL  /* skip if simulation */
108  #if (DISABLE_WDOG)
...
...
...
192  #endif /* (CLOCK_SETUP == 2) */
193  #endif /* SIMUL */
194  }
```

10. Całość zapisujemy i budujemy projekt (F7).
11. Następnie proszę o wykonanie tych samych kroków co w punkcie 4.2. Jakie są różnice? Czy można śledzić działanie peryferiów?