

Ćwiczenie 1: Środowisko pracy KEIL uVision MDK5. Budowa zestawu uruchomieniowego ZL30ARM

Zakres materiału

1. Środowisko programistyczne Keil uVision i język C.
2. Budowa uniwersalnego zestawu uruchomieniowego ZL30ARM.
3. Zasady bezpiecznej pracy z urządzeniami.
4. Programator/debugger ZL30PRGv2_1 (STLink).

Wprowadzenie do ćwiczeń

Keil uVision MDK5 jest programem działającym w systemie operacyjnym Windows, służącym do tworzenia aplikacji dla mikrokontrolerów ARM różnych firm. Umożliwia tworzenie programów z wykorzystaniem różnych narzędzi językowych, debugowanie, symulację i programowanie mikrokontrolerów.

Uniwersalny zestaw uruchomieniowy ZL30ARM przeznaczony jest do testowania aplikacji realizowanych na bazie mikrokontrolerów ARM firmy STMicroelectronics. Na płycie umieszczono mikrokontroler STM32F103CB.

Programowanie mikrokontrolera odbywa się za pomocą programatora ZL30PRG zgodnego z STLink. Programator przed programowaniem należy podłączyć do zestawu uruchomieniowego, a po zaprogramowaniu można go odłączyć.

Pytania kontrolne

1. Co to jest zintegrowane środowisko programistyczne?
2. Do czego służy środowisko uVision MDK?
3. Jakie są zasady bezpiecznej pracy z urządzeniami elektronicznymi?
4. Co to jest i do czego służy zestaw uruchomieniowy ZL30ARM?
5. Co to jest programator?

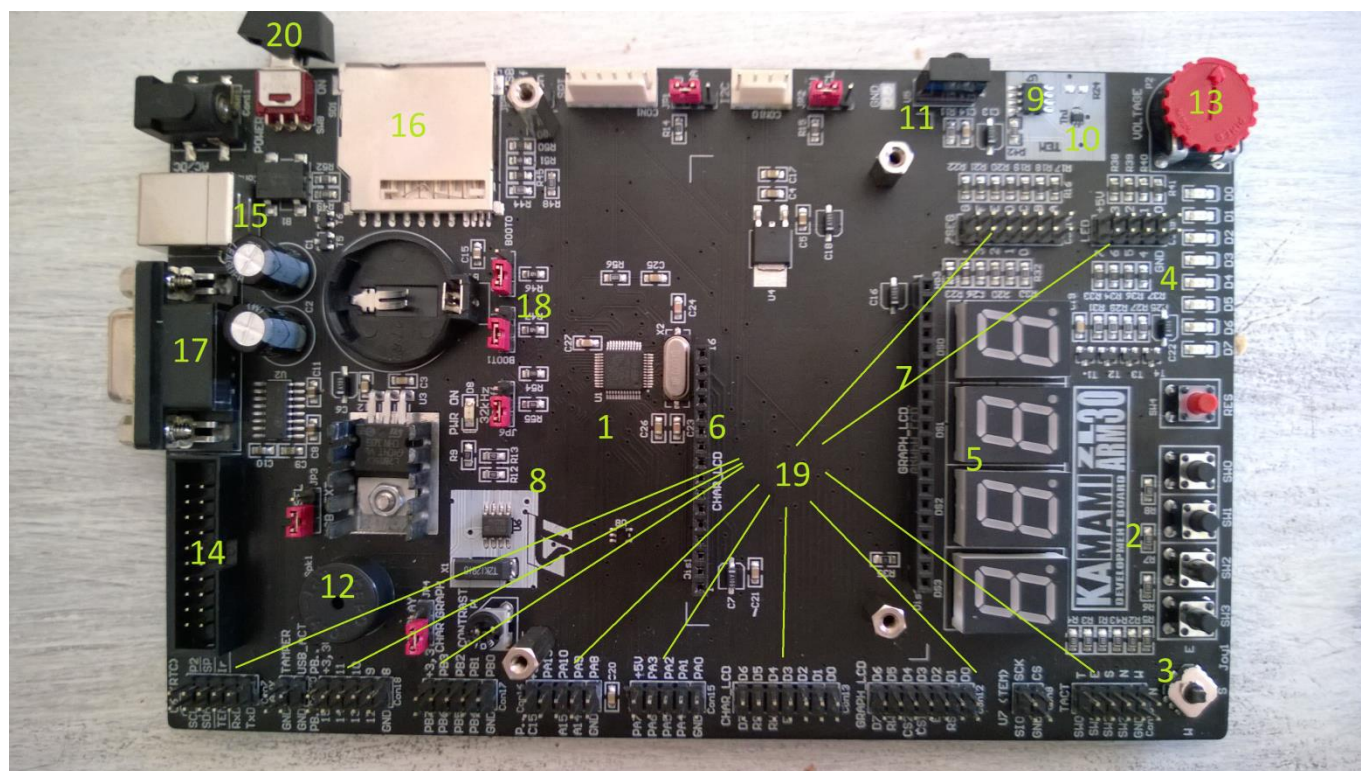
Przykłady i ćwiczenia

Ćwiczenie 1: Zapoznaj się z budową zestawu uruchomieniowego ZL30ARM. Skorzystaj z dokumentu „ZL30ARM – zestaw uruchomieniowy dla mikrokontrolerów STM32F103” na stronie producenta (patrz „Zakładki”).

Budowę zestawu uruchomieniowego ZL30ARM pokazano na rys 1. Zestaw ZL30ARM wyposażony jest w:

1. mikrokontroler STM32F103CBT6 w obudowie LQFP48 (m.in. 128 kB pamięci Flash, 20 kB pamięci SRAM, 2xSPI, 2xI2C, 3xUART, USB, CAN, ADC)
2. 4-przyciskowa klawiatura
3. 5-pozycyjny joystick
4. 8 diod LED
5. czterocyfrowy wyświetlacz siedmiosegmentowy LED
6. złącze dla alfanumerycznego wyświetlacza LCD 2x16 znaków (LCD1602)
7. złącze dla graficznego wyświetlacza LCD 128x64 pikseli ze sterownikiem KS0108 (LCD12864)
8. układ zegara M41T00 (STMicroelectronics) pracujący na magistrali I2C wraz z podstawką na baterię CR2032
9. układ termometru cyfrowego TC77 (Microchip) pracujący na magistrali SPI
10. układ termometru analogowego STLM20 (STMicroelectronics) z wyjściem napięciowym

11. odbiornik podczerwieni TSOP31236 (36 kHz)
12. przetwornik piezoelektryczny
13. potencjometr umożliwiający podanie napięcia na wejście przetwornika analogowo-cyfrowego wbudowanego w mikrokontroler
14. 20-wyprowadzeniowe złącze JTAG umożliwiające programowanie pamięci oraz debugowanie programu
15. złącze USB umożliwiające transmisję danych pomiędzy komputerem PC a mikrokontrolerem STM32
16. złącze kart pamięci SD
17. złącze DB9 umożliwiające transmisję danych za pomocą łącza RS232
18. zworki służące do wyboru typu pamięci, z której zostanie uruchomiony mikrokontroler
19. Wyprowadzenia portów mikrokontrolera oraz układów zewnętrznych w formie złącz szpilkowych do łączenia oddzielnymi przewodami
20. Włacznik napięcia zasilania



Rys 1. Zestaw uruchomieniowy ZL30ARM

Programator ZL30PRG v2_1 (zgodny z STLink) korzysta ze złącza USB (po stronie komputera PC) oraz interfejsu JTAG lub SW (po stronie mikrokontrolera). Spełnia on 2 podstawowe zadania:

- pozwala zaprogramować pamięć mikrokontrolera („wgrać” program do pamięci),
- pozwala śledzić wykonanie programu i stan mikrokontrolera w przypadku jego debugowania (poszukiwania błędów).

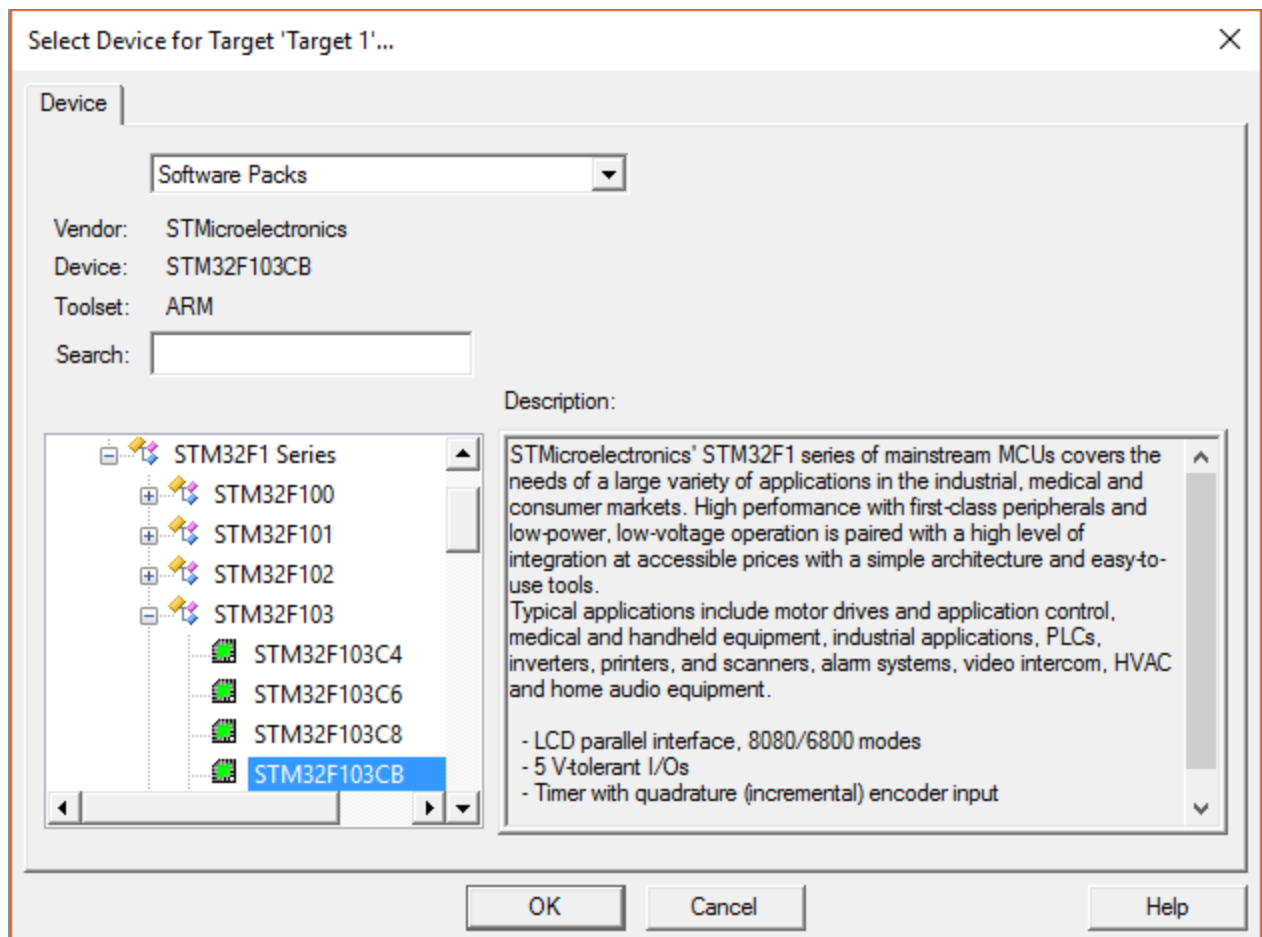
Programator należy włożyć w złącze JTAG. Przy operowaniu płytką należy uważać, by nie wyłamać programatora ze złącza.



Rys. 2. Programator/debugger ZL30PRGv2_1

Ćwiczenie 2: Zapoznaj się ze środowiskiem Keil uVision MDK5 korzystając z poniższych wskazówek.

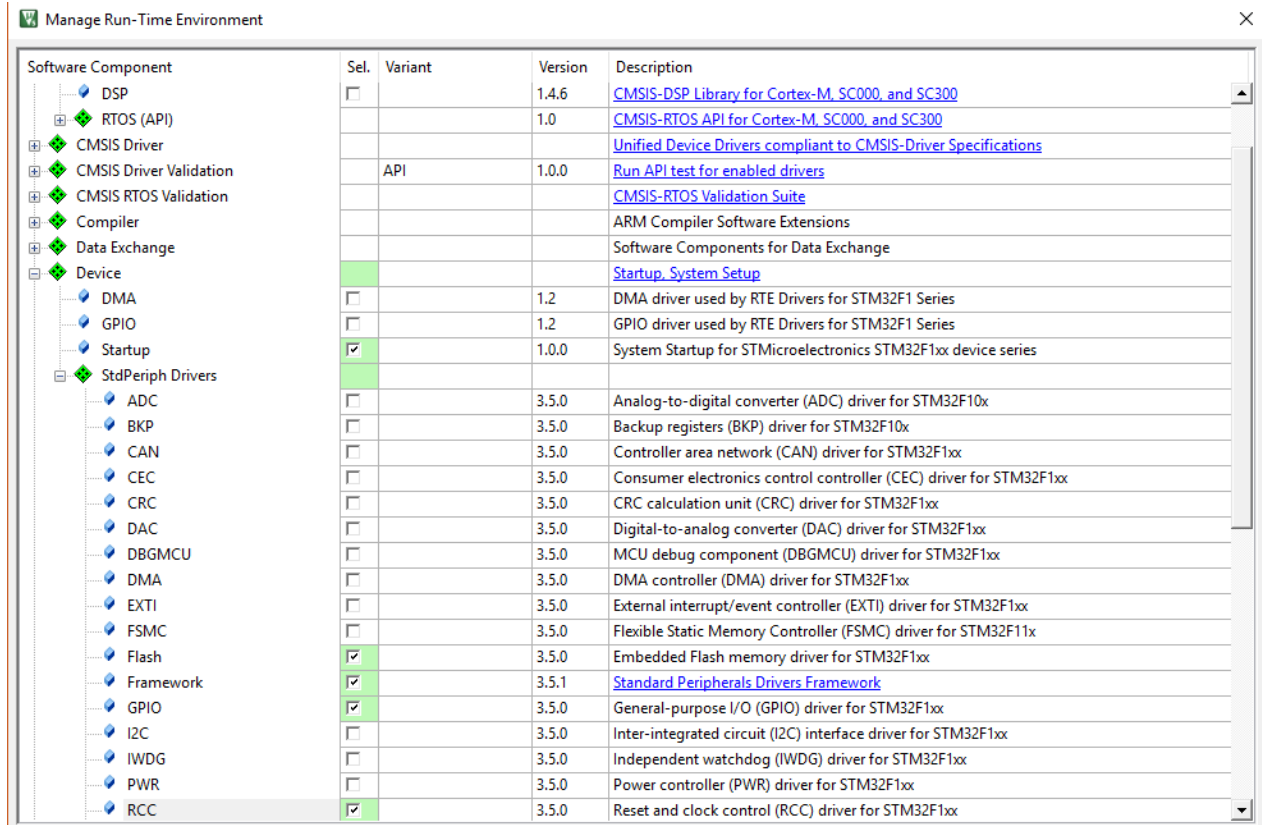
1. Przed przystąpieniem do pracy stwórz folder, w którym będzie umieszczony projekt.
2. Następnie uruchom Keil μ Vision 5. Jeżeli po uruchomieniu został załadowany jakiś projekt – zamknij go (menu *Project > Close Project*).
3. Z menu wybierz *Project > New uVision Project....* W oknie, które się otworzy wybierz folder, który przed chwilą utworzyłeś dla swojego projektu, a następnie podaj nazwę swojego projektu. Po wybraniu przycisku *Zapisz* otworzy się okno *Select Device for Target*, w którym należy wybrać procesor, dla którego tworzony będzie projekt. Należy wybrać *STMicroelectronics > STM32F1 Series > STM32F103 > STM32F103CB* (Rys.3.)



Rys. 3. Wybór mikrokontrolera

4. Po kliknięciu *OK*, pojawi się okno, w którym można wybrać, które z dołączonych do środowiska μ Vision pakietów obsługi sprzętu chcemy dodać do projektu. Wykorzystanie w projekcie pakietów znacznie upraszcza procedurę jego tworzenia. Ponadto w pakietach znaleźć można wiele gotowych, przydatnych bibliotek funkcji. Najważniejsze grupy pakietów to *CMSIS* i *Device*. Pakiet *CMSIS* zapewnia obsługę najważniejszych podukładów rdzenia Cortex-M i jest niezależny od wersji procesora. Pakiety z grupy *Device* oparte są przede wszystkim na bibliotekach dostarczonych przez STMicroelectronics i dotyczą konkretnej rodziny procesora, w naszym przypadku STM32F1. Pakiety dostarczone z Keil zostały nieznacznie uzupełnione i zmienione w stosunku do oryginalnych bibliotek od STM. Spośród dostępnych pakietów należy wybrać (Rys. 4):
 - *CMSIS > Core* – obsługa rdzenia
 - *Device > startup* – plik startowy, uruchamiający i konfiguruje mikrokontroler
 - *Device > StdPeriph Drivers >*
 - *Flash* – funkcje obsługi pamięci Flash
 - *Framework* – podstawowe funkcje obsługi rdzenia i kontrolera przerwań
 - *GPIO* – funkcje obsługi portów GPIO
 - *RCC* – funkcje obsługi konfiguracji sygnałów taktujących

Po wybraniu wszystkich potrzebnych pakietów kliknij OK. Jeśli w czasie pracy nad programem okaże się, że potrzebne są dalsze pakiety, których nie wybrano na etapie tworzenia projektu, można będzie je dołączyć później.



Rys. 4. Wybór pakietów

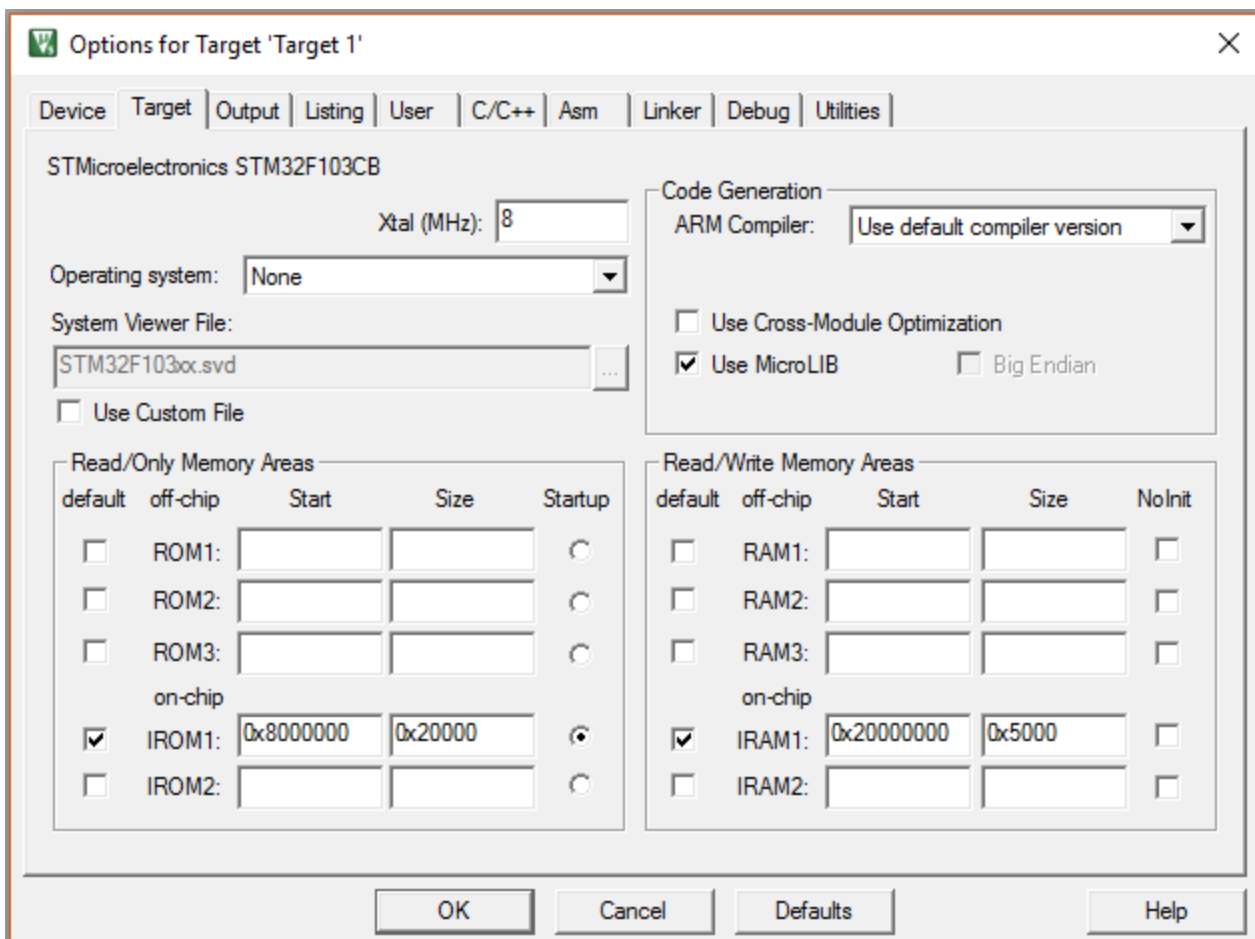
5. Z menu wybierz File>New... Utworzymy główny plik programu. W otwartym, nowym pliku wpisz:

```
#include "stm32f10x.h"

int main(void)
{
}
```

Następnie zapisz utworzony plik w folderze projektu pod nazwą main.c.

6. Kliknij prawym klawiszem myszy w oknie Project na grupie Source Group 1 i wybierz Add Existing Files to Group 'Source Group 1'... W otwartym oknie wybierz plik main.c i kliknij Add, a następnie Close. W ten sposób do projektu został dodany główny plik kodu.
7. Po dodaniu plików kliknij na najwyższym elemencie struktury plików (*Target 1*) prawym klawiszem myszy i wybierz *Options for Target*. W kolejnych zakładkach okna znajdują się opcje konfiguracji projektu. Większość z nich jest już ustawiona automatycznie przez środowisko Keil.
- Zakładka Device
 - Można tu zmienić typ procesora, dla którego przeznaczony jest program. Pozostawiamy go bez zmian ponieważ wybrany został w pierwszym etapie tworzenia projektu.
 - Zakładka Target (Rys. 5)
 - Zawiera ustawienia dotyczące rezonatora kwarcowego taktującego procesor oraz ustawienia obszarów pamięci. W pole Xtal (MHz) należy wpisać wartość 8.
 - Ustawienia obszarów pamięci pozostawić bez zmian, ponieważ standardowo ustawione one są w ten sposób, że kod programu znajdzie się w pamięci Flash. W zakładce tej włącz opcję Use MicroLIB. Powoduje ona użycie niestandardowej, niezgodnej z ANSI C biblioteki wykonawczej języka C. Biblioteka ta pozwala jednak uzyskać mniejszy kod wykonywalny programu.



Rys. 5. Zakładka target

c. Zakładka Output

- Zawiera opcje związane z generacją plików binarnych tworzonych podczas kompilacji. Podczas kompilacji pliki binarne będą przechowywane w podfolderze \Objects utworzonym automatycznie przez środowisko Keil w folderze projektu. Ponadto należy wybrać pole Create Executable i zaznaczyć pola: Debug Information, Create HEX File oraz Browse Information.

d. Zakładka Listing

- Zawiera opcje związane z zapisem plików asemblera tworzonych podczas kompilacji. Podczas kompilacji pliki te będą przechowywane w podfolderze \Listings utworzonym automatycznie przez środowisko Keil w folderze projektu.

e. Zakładka User

- Zawiera pola, w których można wpisać dodatkowe polecenia, jakie mają być wykonane przed i po kompilacji oraz po linkowaniu. Wszystkie opcje pozostawiamy bez zmian.

f. Zakładka C/C++

- Zawiera opcje kompilatora języka C. W polu Optimization można wybrać poziom optymalizacji kodu.
- Poniżej listy wyboru poziomu optymalizacji wybierz opcję One ELF Section per Function.
- Można także włączyć opcję Optimize for time, która powoduje, że optymalizacja kodu ma na celu przyspieszenie działania gotowego programu, ale za cenę powiększenia rozmiaru zajętej pamięci. Gdy opcja jest wyłączona optymalizacja ma na celu zmniejszenie zajętości pamięci, co jednak może oznaczać wolniejsze wykonywanie programu.

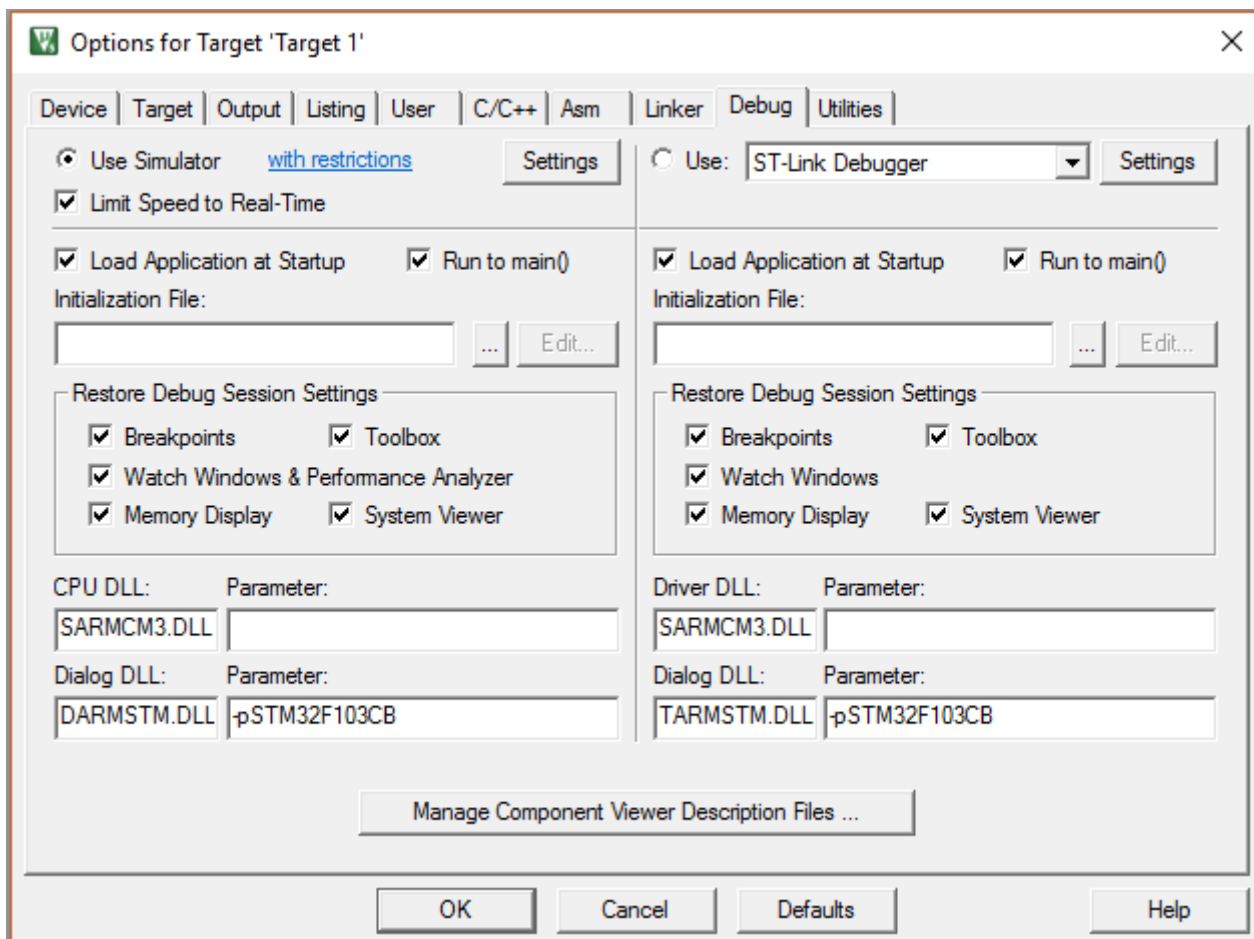
g. Zakładka Asm

- Zawiera opcje kompilatora Assemblera. Wszystkie opcje pozostawiamy bez zmian.

h. Zakładka Linker

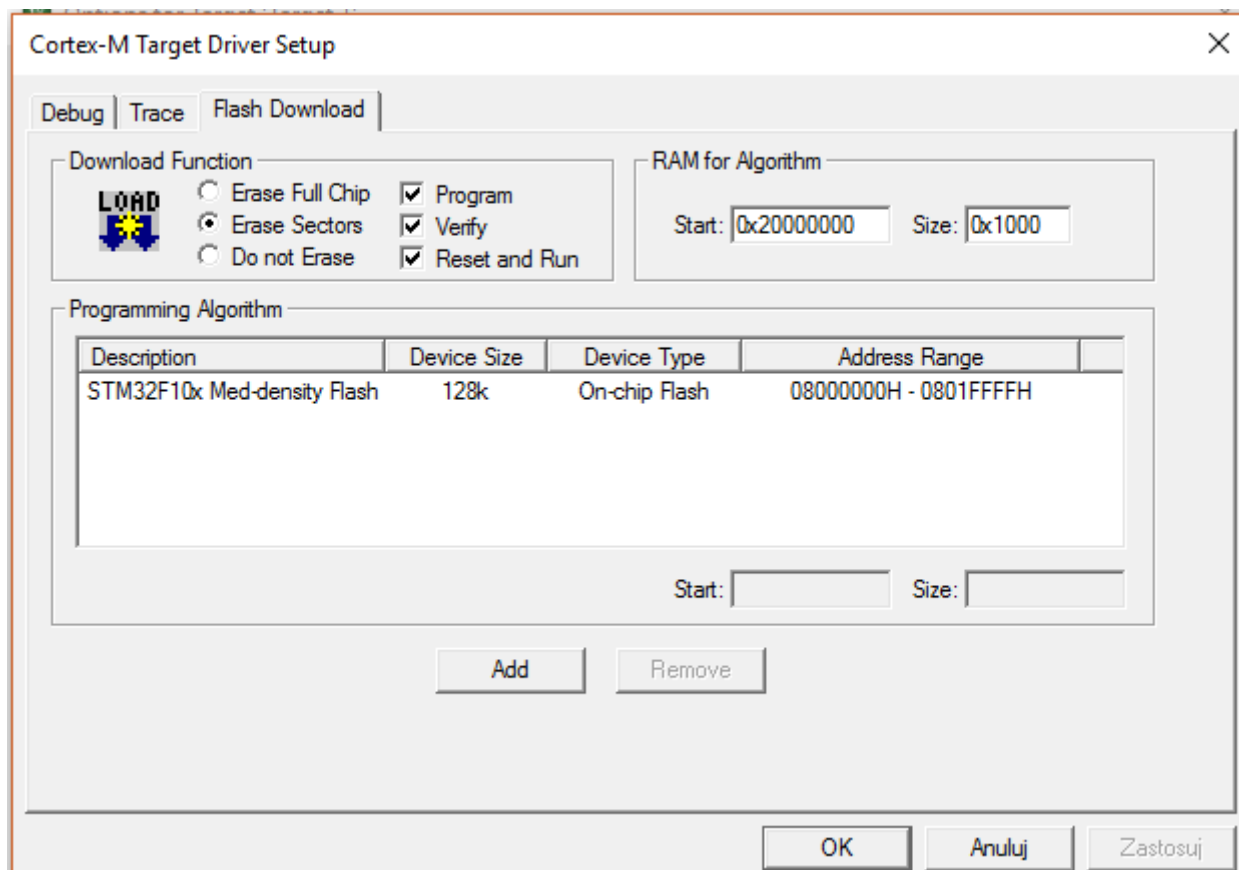
- Zawiera opcje linkera.
- Należy włączyć Use Memory Layout from Target Dialog.

i. Zakładka Debug (Rys. 6)



Rys. 6. Zakładka Debug

- Zawiera opcje debugera.
- Najważniejszą opcją jest wybór, czy w momencie włączenia sesji debugowania ma zostać użyty symulator procesora, czy też środowisko ma połączyć się z rzeczywistym procesorem. Opcja ta będzie wielokrotnie zmieniana podczas pracy z projektem.
- Ustaw Use Simulator oraz zaznacz pole Limit Speed to Real-Time. Opcja ta pozwala zachować rzeczywiste zależności czasowe podczas symulacji. W praktyce jednak, symulator zwykle działa znacznie wolniej niż rzeczywisty procesor.
- Sprawdź wybór bibliotek DLL używanych podczas sesji debugowania programu i odpowiedzialnych za dostępne okna dialogowe (funkcje) debugera. Powinny być używane biblioteki dedykowane dla układów STM. W lewym polu Dialog DLL powinno być DARMSTM.DLL, zaś w prawym - TARMSTM.DLL. W obu polach Parameter powinno być - pSTM32F103CB (razem ze znakiem "-") (patrz Rys. 6).
- W prawej części okna wybierz z listy ST-Link Następnie wybierz przycisk Settings i w oknie Cortex-M Target Driver Setup wybierz zakładkę Flash Download. W oknie Programming Algorithm powinien być ustawiony STM32F10x Med-density Flash. Jeśli okno jest puste wybierz przycisk Add i z listy wybierz STM32F10x Med-density Flash (Rys. 7).
- W zakładce Settings warto zaznaczyć opcję Reset and Run, która spowoduje restart mikrokontrolera po załadowaniu programu do pamięci Flash i uruchomienie załadowanego programu (Rys. 7)



Rys. 7. Ustawienia debuggera sprzętowego –zakładka Flash Download

j. Zakładka Utilities

- Zawiera opcje dotyczące programatora. Ponieważ korzystamy z układu zgodnego z ST-Link, który pełni rolę zarówno programatora jak i debugera, zaznaczamy pole Use Debug Driver.

8. Uzupełnij funkcję main o kod programu:

```
GPIO_InitTypeDef GPIO_InitStructure; // deklarujemy strukturę do inicjalizacji portu



RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); //włącz taktowanie portu GPIO A

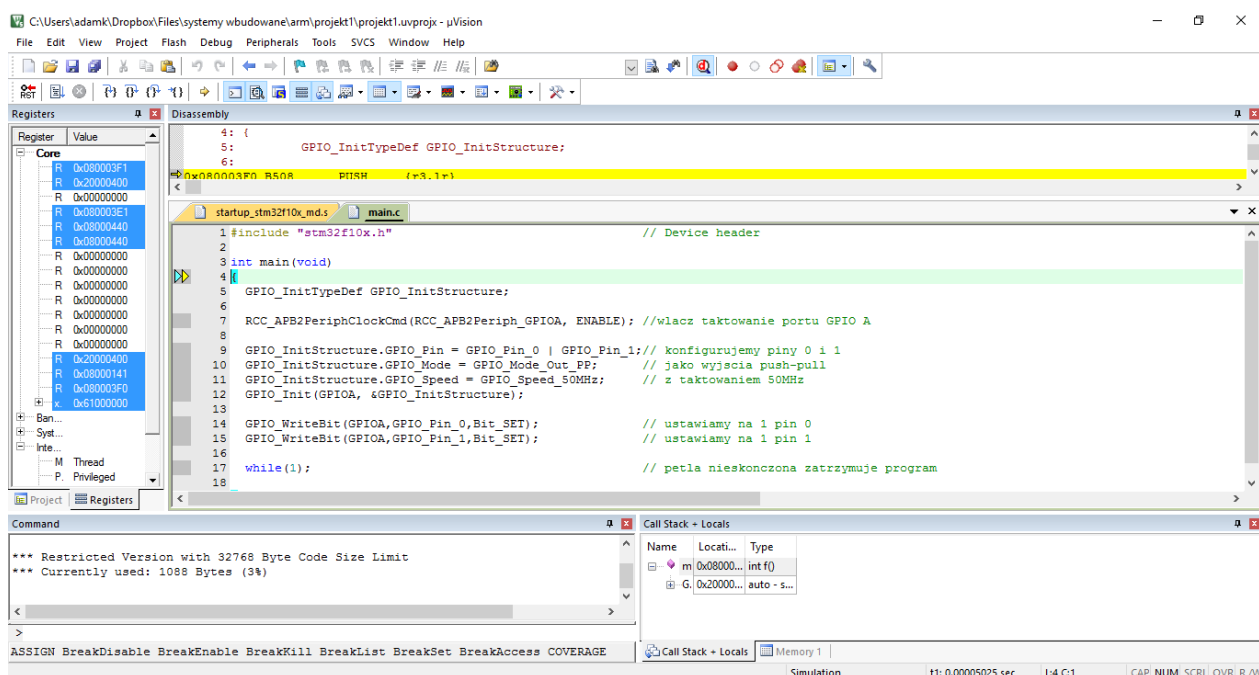
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1; // konfigurujemy piny 0 i 1
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; // jako wyjścia push-pull
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; // z taktowaniem 50MHz
GPIO_Init(GPIOA, &GPIO_InitStructure); // inicjalizujemy port GPIO A

GPIO_WriteBit(GPIOA, GPIO_Pin_0, Bit_SET); // ustawiamy na 1 pin 0
GPIO_WriteBit(GPIOA, GPIO_Pin_1, Bit_SET); // ustawiamy na 1 pin 1







while(1); // petla nieskończona zatrzymuje program
```

- Gdy kod programu jest gotowy, należy go skompilować i zlinkować. Z menu należy wybrać polecenie Project > Rebuild All Target Files, co spowoduje skompilowanie wszystkich plików lub Project > Build (klawisz F7), co spowoduje skompilowanie tylko plików, które były modyfikowane. Jeżeli kompilacja się powiedzie rozpoczyna się proces linkowania. Jeżeli on także przebiegnie bez błędów pojawi się komunikat podający rozmiar wygenerowanego kodu oraz informację o wygenerowaniu pliku HEX
- Na płytce prototypowej należy połączyć przewodami piny złącza Con15 (wyprowadzenia portu A) z pinami złącza Con9 (diody LED): PA0 → 0, PA1 → 1.

11. Aby procesor został zaprogramowany, należy skompilowany program umieścić w jego pamięci. W tym celu wystarczy kliknąć ikonkę Download () na pasku narzędzi w głównym konie środowiska Keil μ Vision. Po zakończeniu programowania i weryfikacji jego poprawności w oknie komunikatów powinien pojawić się komunikat Programming Done. Verify OK. Aby uruchomić program należy zresetować mikrokontroler przyciskiem Reset na zestawie uruchomieniowym.
12. Aby przejść do trybu debugera kliknij ikonę (). Układ okien zmieni się na pokazany na Rys. 8. W tym trybie możliwe jest śledzenie wykonania programu krok po kroku oraz podgląd: wszystkich rejestrów procesora, wszystkich rejestrów związanych z układami peryferyjnymi oraz dowolnego obszaru pamięci. Widoczne okna to:
- Okno Register: można tam na bieżąco oglądać zawartość rejestrów procesora R0-R15 i xPSR
 - okno Disassembly zawierające kod asemblera wygenerowany na podstawie kodu w języku C. Są to kolejne instrukcje wykonywane przez procesor. Warto zauważyć, że kompilator może nie tylko rozłożyć jedną instrukcję w języku C na kilka instrukcji asemblerowych, ale także połączyć kilka instrukcji w jedną, w celu optymalizacji programu.
 - Wybierając z menu polecenie Peripherals, a następnie System Viewer możemy otworzyć po prawej stronie okno z podglądem zawartości rejestrów odpowiedzialnych za obsługę danego układu peryferyjnego lub bloku procesora. Dzięki temu można także szybko znaleźć adres interesującego nas rejestru.
 - Wybierając z menu polecenie View > Symbols window można otworzyć okno z listą symboli użytych w programie. Po rozwinięciu grupy main, pojawi się lista adresów wszystkich funkcji i zmiennych użytych w programie.
 - Memory – pozwala obejrzeć zawartość wskazanego obszaru pamięci. Dodanie zmiennej do jednego z tych okienek można uzyskać klikając na nazwie zmiennej prawym klawiszem myszy i wybierając polecenie Add 'zmienna' to....
 - okno Symulatora (dostępne w menu Peripherals – uwaga, nie w każdej sytuacji jest dostępne), które pozwala w bardziej przystępny sposób zobaczyć najważniejsze rejestry dotyczące danego podukładu procesora oraz zmieniać ich stan.



Rys. 8. Okno debugera

13. W celu zapoznania się z debuggerem umieść w kodzie punkt zatrzymania (breakpoint) w linii ustawiającej bit 0 portu A. W tym celu kliknij w polu na lewo od numeru linii, dla której chcesz ustawić punkt zatrzymania. Ustawiony punkt zaznaczony jest czerwonym symbolem.
14. Następnie kliknij przycisk Reset , co spowoduje ustawienie programu w punkcie startu. Program może być wykonywany krokowo (ikony    ) lub wykonany do najbliższego punktu stopu (ikona ). Jeśli punkt stopu nie został zdefiniowany – ikona ta po prostu uruchamia program.
15. Dodatkowo w zakładce Memory można wpisać adres 0x40010800 – jest to adres portu GPIOA. Dodatkowo, można też otworzyć okno symulatora portu GPIOA (menu *Peripherals > General Purpose I/O > GPIOA*).
16. Obserwuj zmiany, jakie zachodzą w oknie w poszczególnych oknach środowiska. Zauważ także, że w dolnej części okna środowiska wskazywany jest czas, jaki upłynął od uruchomienia programu
17. Ponownie zresetuj program i dodatkowo otwórz okno symulatora PRCC (Peripherals > Power, Reset & Clock Control). Pozwoli ono obejrzeć, czy procesor rzeczywiście został skonfigurowany do pracy z pełną prędkością. Następnie wykonuj program linia po linii i obserwuj jak zmieniają się ustawienia procesora.
18. Spróbuj powtórzyć proces debugowania używając debugera sprzętowego (Options for Target 1 → Debug → Use: STLink Debugger).

Ćwiczenie 3: Zapoznaj się z szablonem projektu *template.zip* (do pobrania ze strony). Rozpakuj archiwum. Obejrzyj strukturę projektu. Zapoznaj się z zawartością pliku *main.c*. Spróbuj skompilować projekt.