Systemy Wbudowane

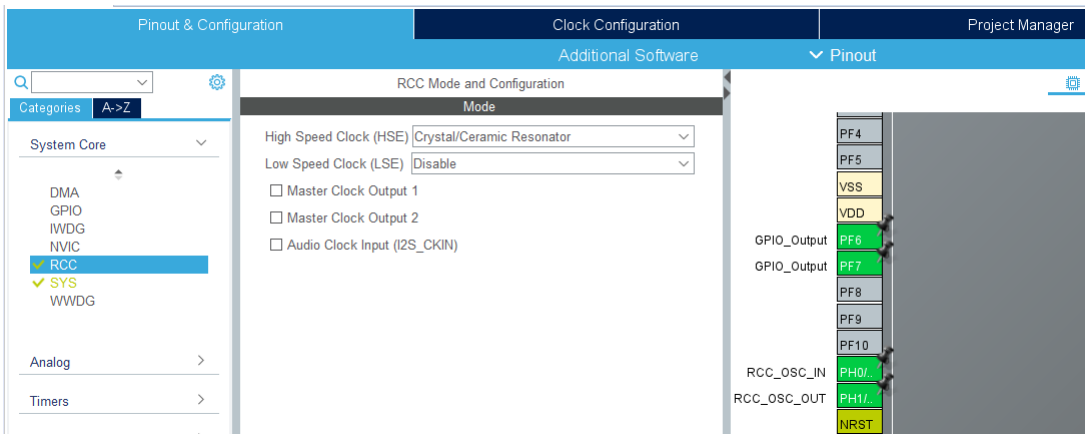
Laboratorium 3:

Pomiar wydajności wybranych operacji arytmetycznych

Celem ćwiczenia jest poznanie ograniczeń systemów wbudowanych zbudowanych w oparciu o mikrokontrolery w zakresie zdolności do realizacji obliczeń w czasie rzeczywistym.

3.1 Ćwiczenie 1

Utworzyć nowy projekt w środowisku STMCubeIDE dla mikrokontrolera STM32F429ZI. Następnie, w zakładce "System Core → RCC" zezwolić na pracę zewnętrznego rezonatora kwarcowego (*ang. Crystal/Ceramic Resonator*).

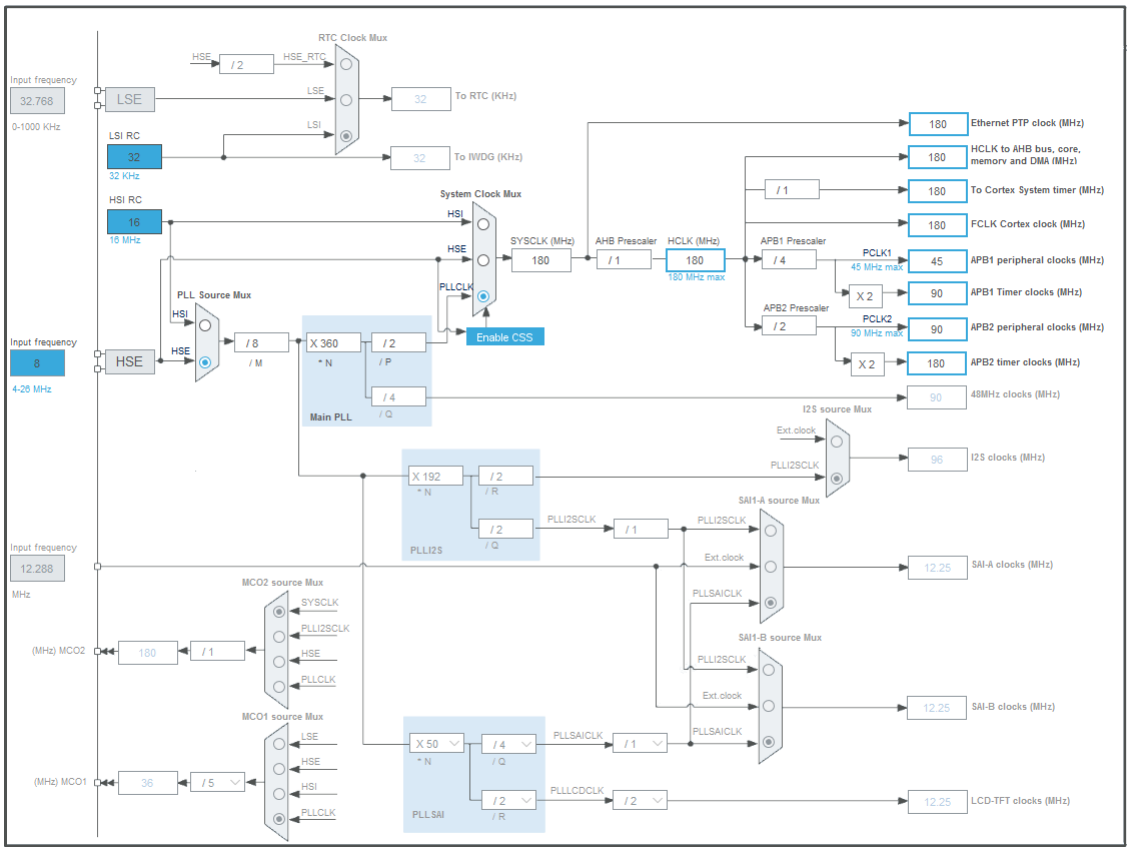


Rysunek 3.1: Konfiguracja rezonatora kwarcowego

Zauważ, że linie PH0/OSC_IN oraz PH1/OSC_OUT zostały automatycznie skonfigurowane przez środowisko programistyczne do pracy w trybach odpowiednio RCC_OSC_IN oraz RCC_OSC_OUT. Taka konfiguracja była potrzebna, gdyż do tych uniwersalnych linii kontrolera na płytce ewaluacyjnej STM32F4-Discovery jest podłączony zewnętrzny rezonator kwarcowy o częstotliwości 8MHz. Rezonator ten zostanie wykorzystany w ćwiczeniu jako precyzyjne źródło sygnału taktującego mikrokontroler.

Dodatkowo należy skonfigurować linie mikrokontrolera oznaczone PF6, PF7 do pracy w trybie portów wyjściowych GPIO (GPIO_Output). Oprogramowanie realizowane w ramach niniejszego ćwiczenia będzie sterowało stanem tych linii w tempie realizowanych kolejno obliczeń arytmetycznych. Za pomocą oscyloskopu cyfrowego możliwe będzie dokonanie precyzyjnego pomiaru czasu trwania impulsów cyfrowych na liniach PF6 i PF7. W ten sposób zostanie oszacowany czas realizacji poszczególnych operacji arytmetycznych w mikrokontrolerze.

Ustawić parametry zegarów systemowych podobnie jak w Laboratorium 2, tj. zgodnie z Rysunkiem 3.2. Taka konfiguracja pozwala na uzyskanie najwyższej dostępnej wydajności systemu z procesorem STM32F429ZIT6, tj. pracę jednostki centralnej z zegarem o częstotliwości 180MHz.



Rysunek 3.2: Konfiguracja zegarów umożliwiające uzyskanie częstotliwości 180MHz

W automatycznie wygenerowanym projekcie we wskazanych miejscach dodać kod jak poniżej.

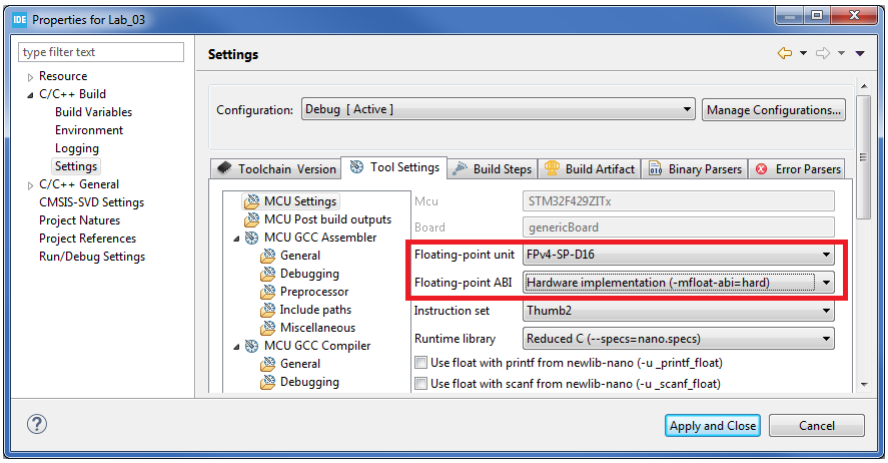
```
/* USER CODE BEGIN PV */
volatile int32_t ai, bi, ci;
volatile double ad, bd, cd;
volatile float as, bs, cs;
/* USER CODE END PV */
```

oraz

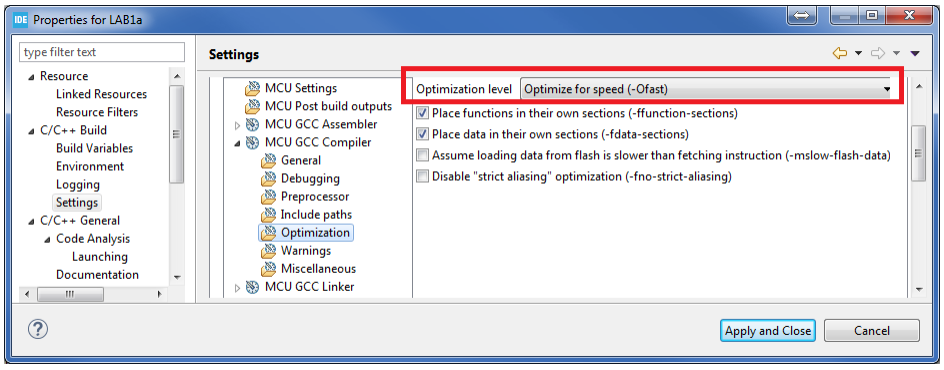
```
/* USER CODE BEGIN WHILE */
while (1) {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    // Zainicjowanie wartości zmiennych.
    ai = 1; bi = 10;
    as = 1.0; bs = 10.0;
    ad = 1.0; bd = 10.0;

    // Znacznik dla oscyloskopu: początek sekwencji pomiaru.
    HAL_GPIO_WritePin(GPIOF, GPIO_PIN_7, GPIO_PIN_SET);

    // Test_1 (TS1): wyznacznie czasu sterowania - czasu zmiany sygnału ze stanu niskiego na wysoki i odwrotnie.
    HAL_GPIO_WritePin(GPIOF, GPIO_PIN_6, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOF, GPIO_PIN_6, GPIO_PIN_RESET);
```

```
    // Test 2 (TS2): wyznaczenie czasu będącego sumą operacji dodawania liczb całkowitych i czasu sterowania.
    HAL_GPIO_WritePin(GPIOF, GPIO_PIN_6, GPIO_PIN_SET);
    ci = ai + bi;
    HAL_GPIO_WritePin(GPIOF, GPIO_PIN_6, GPIO_PIN_RESET);

    // Test 3 (TS3): wyznaczenie czasu operacji dodawania liczb rzeczywistych ... .
    HAL_GPIO_WritePin(GPIOF, GPIO_PIN_6, GPIO_PIN_SET);
    cd = ad + bd;
    HAL_GPIO_WritePin(GPIOF, GPIO_PIN_6, GPIO_PIN_RESET);

    // Test 4 : itd...

    // koniec sekwencji pomiaru.
    HAL_GPIO_WritePin(GPIOF, GPIO_PIN_7, GPIO_PIN_RESET);
}
/* USER CODE END 3 */
```

Ustawić parametry projektu zgodnie z rysunkiem poniżej. Aby wywołać pokazane niżej okno właściwości projektu należy kliknąć PPM na nazwie projektu w "Project Explorer" i wybrać z menu kontekstowego opcję "Properties".



Rysunek 3.3: Konfiguracja jednostki zmiennoprzecinkowej



Rysunek 3.4: Konfiguracja optymalizacji kodu

Zatwierdzić zmiany i zamknąć okno wciskając klawisz "Apply and Close", a następnie wybrać z menu Project kolejno opcję Clean i Build All.

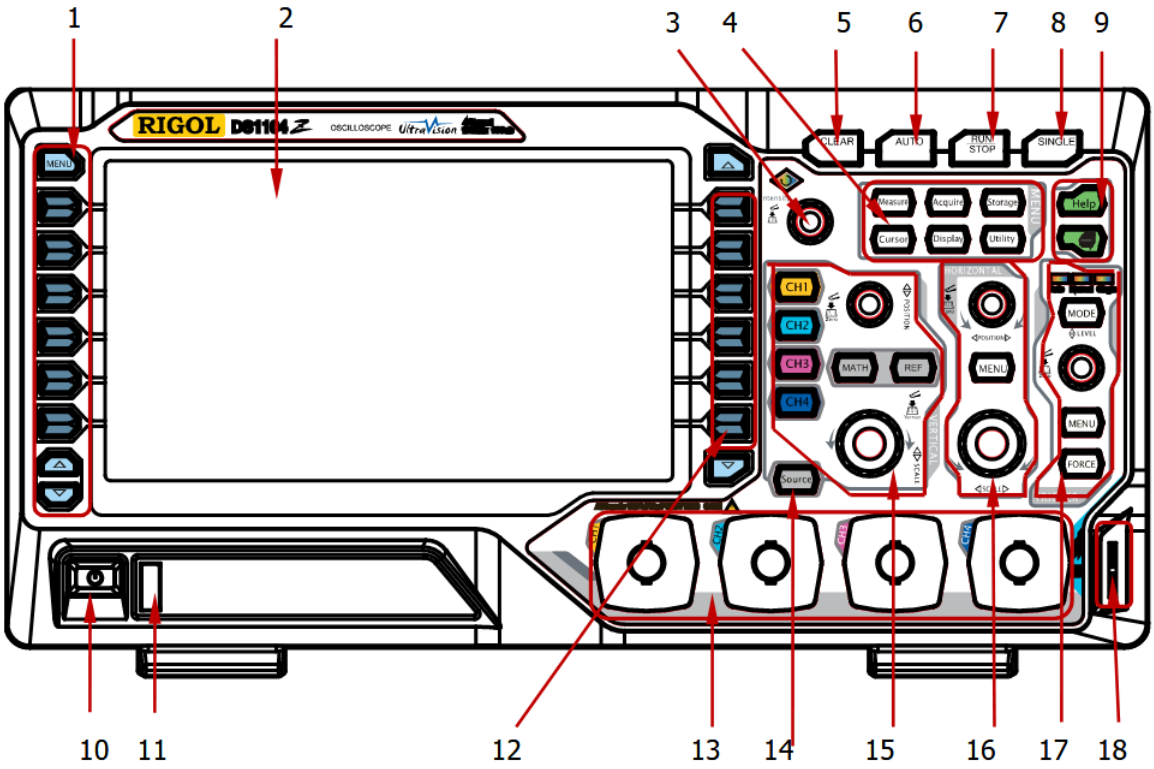
Do linii GPIO PF6 i PF7 podłączyć elektryczne przewody stykowe, których drugie końce powinny zostać podłączone sond, co przedstawia poniższe zdjęcie. Na zdjęciu poniżej pokazano również połączenie żabek sond do uziemienia (GND).

Rysunek 3.5: Podłączenie badanych sygnałów do sond i oscyloskopu

Uwaga, należy zachować ostrożność aby nie doprowadzić do zwarcia linii sygnałowych z innymi liniami płyty ewaluacyjnej.

Uruchomić oscyloskop oraz wgrać program do płytki sterownika i uruchomić go.

Działanie oscyloskopu



Rysunek 3.6: Podstawowe funkcje oscyloskopu - źródło:
https://www.batronix.com/pdf/Rigol/UserGuide/DS1000Z_UserGuide_EN.pdf

Do oscyloskopu można podłączyć cztery sondy oznaczone kolejno kolorami (żółty, jasnoniebieski, różowy i niebieski - **(13)**). Ich włączenie umożliwiają przyciski CH1, CH2, CH3 oraz CH4 umieszczone w sekcji **(15)** (świecący przycisk oznacza aktywny kanał). Jednokrotne kliknięcie na dany kanał powoduje że można dany sygnał przesuwać i skalować - w pionie - panel **(15)** oraz poziomie - panel **(16)**. Oscyloskop może pracować w różnych trybach np. AUTO **(6)**, dla którego oscyloskop stara się sam wykryć częstotliwość i amplitudę sygnałów, RUN/STOP **(7)** gdzie domyślnie wyświetlane jest nakładanie się kolejnych przebiegów lub SINGLE **(8)** gdzie wyświetlany jest pojedyńczy przebieg sygnału. Aby ustawić sygnał wzorcowy (według którego ustalana będzie częstotliwość sygnału) należy wybrać **(14)** i za pomocą menu **(12)** ustawić wybrany kanał (Source na np. CH1 lub CH2), typ sygnału źródłowego (Type na np. Edge) i za pomocą pokrętła z panelu **(17)** ustawić linię wykrywania sygnału w połowie widocznego sygnału. Syngały można mierzyć za pomocą np. opcji dostępnych w panelu **(4)** - przycisk Measure uaktywnia lewe menu **(1)** z opcjami pomiar, natomiast przycisk Cursor uaktywnia możliwość pomiaru przez ręczne ustawianie kursorów (znaczników czasu) na ekranie. Warto zwrócić także uwagę na to, że część pokręteł można też kliknąć wywołując dodatkowe opcje (np. wybór opcji z menu można zatwierdzić klikając pokrętłem **(3)**). Wyniki pomiarów należy samodzielnie odczytać analizując skale wyświetlacza (np. H 100ns na rysunku poniżej) lub informacje wyświetlane przy użyciu opcji Cursor.

Konfigurując oscyloskop za pomocą pokręteł znajdujących się na jego panelu doprowadzić do sytuacji, aby możliwy był pomiar szerokości (tj. czasu trwania) impulsów widocznych na pierwszym kanale oscyloskopu. W tym celu najlepiej ustawić wyzwalanie stanem narastającym zboczem kanału drugiego oscyloskopu. Uzyskany wynik powinien być zbliżony do tego przedstawionego na zdjęciu poniżej.

Następnie zmierzyć (w przybliżeniu) czas trwania stanu wysokiego trzech kolejnych impulsów na kanale pierwszym. Zanotować te wartości jako TS1, TS2 i TS3. Wartości te powinny być zbliżone do: TS1 = 58ns, TS2 = 90ns oraz TS3 = 390ns.

Rysunek 3.7: Konfiguracja oscyloskopu umożliwiająca pomiar szerokości generowanych impulsów

(Uwaga, wyniki uzyskane w konkretnym doświadczeniu mogą się dość znacznie różnić od tych przedstawionych na powyższym rysunku. Różnice mogą sięgać nawet +/-50%, szczególnie w przypadku pomiaru stosunkowo krótkich odcinków czasu - na przykład rzędu 100 nanosekund. Różnice mogą wynikać ze sposobu, w jaki kod programu użytkownika zostanie zoptymalizowany przez kompilator. Wynika to z faktu, że jakość optymalizacji kodu silnie zależy od wielu różnych czynników. W szczególności od sposobu, w jaki projektant oprogramowania użyje poszczególnych instrukcji języka C do realizacji konkretnego zadania.)

Na rysunku powyżej można zauważyć trzy zmiany stanu sygnału (linia żółta): najkrótsza zmiana - reprezentuje sam czas zmiany stanu GPIO, średnia zmiana (GPIO + dodawanie liczba całkowitych), najdłuższa zmiana (GPIO + dodawanie liczb rzeczywistych).

W wyżej wykonanym doświadczeniu, wartość TS1 odpowiada za czas sterowania linią GPIO i będzie używana do korekty podczas wyznaczania czasu trwania innych operacji, zgodnie z metodą poniżej:
T2 = TS2 – TS1,
gdzie, tak wyznaczona wartość T2 = 32ns i oznacza czas realizacji operacji dodawania liczb całkowitych w systemie procesorowym z mikrokontrolerem STM32F429ZI. Analogicznie:
T3 = TS3 – TS1,
gdzie, tak wyznaczona wartość T3 = 332ns i oznacza czas realizacji operacji dodawania liczb rzeczywistych podwójnej precyzji (ang. double-precision floating point numbers) w systemie procesorowym z mikrokontrolerem STM32F429ZI wspomaganym przez sprzętowy moduł FPU.

Za pomocą pokazanej wyżej metody dokonać pomiaru czasu wykonywania przez mikrokontroler poniższych operacji arytmetycznych:

a) dodawanie liczb DOUBLE

b) mnożenia liczb DOUBLE

c) dzielenia liczb DOUBLE

d) instrukcji trygonometrycznej sinus dla liczb DOUBLE, tj. cd = sin(ad); // biblioteka math lub cmath

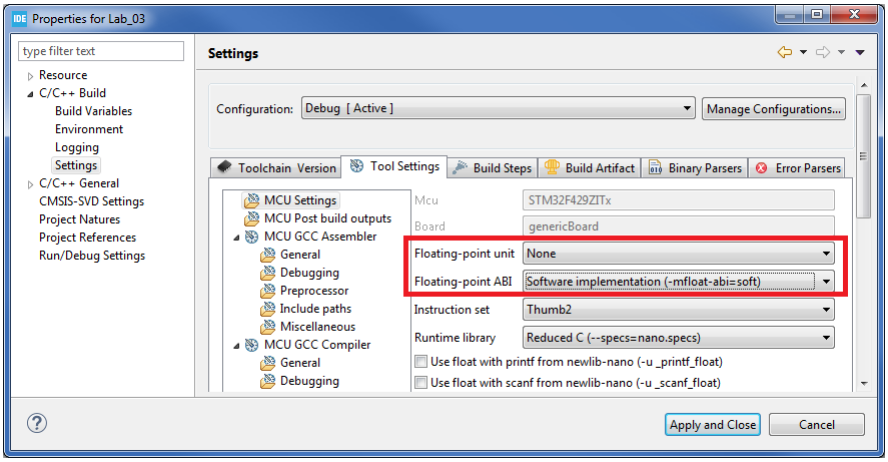
Uzyskane wyniki zestawić w tabeli i przedstawić w sprawozdaniu razem ze zdjęciem ekranu oscyloskopu.

Uwaga, aby móc skorzystać z funkcji sin, sinf, exp i expf należy dołączyć do projektu bibliotekę odpowiednią matematyczną za pomocą następującej dyrektywy preprocesora:

```
/* USER CODE BEGIN Includes */
#include "math.h"
/* USER CODE END Includes */
```

3.2 Ćwiczenie 2

Zmienić konfigurację projektu wyłączając wspomaganie obliczeń przez sprzętowa jednostkę zmiennoprzecinkową (ang. floating-point unit, FPU) zgodnie z rysunkiem poniżej a następnie wyczyścić (Clean) i przebudować projekt (Build All) i powtórzyć pomiary z ćwiczenia 1.



Rysunek 3.8: Konfiguracja jednostyki zmiennoprzecinkowej - wyłączenie akceleracji

Uzyskane wyniki, zestawić z wynikami uzyskanymi z zadania pierwszego we wspólnej tabeli, aby możliwe było porównanie wydajności realizacji poszczególnych operacji arytmetycznych przy wspomaganiu przez sprzętowa jednostkę FPU oraz przy braku takiego wspomagania.

Przeanalizować uzyskane wyniki i wnioski z tej analizy zawrzeć w sprawozdaniu.

Dla tego laboratorium należy przygotować sprawozdanie zawierające odpowiednie kody źródłowe oraz zdjęcia przedstawiające działanie programów (dla wszystkich ćwiczeń)