Instrukcja do laboratorium z przedmiotu „Programowalne układy System on Chip”

dr hab. inż. Bogdan Pankiewicz
Katedra Systemów Mikroelektronicznych
Wydział Elektroniki Telekomunikacji i Informatyki
Politechnika Gdańska
Gdańsk, grudzień 2014, marzec 2016, luty 2017, luty 2018

Spis treści

1. Organizacja zajęć laboratoryjnych.

W ramach zajęć laboratoryjnych należy wykonać cztery następujące ćwiczenia:

a) ćwiczenie wstępne (ocena zal/nzal),
b) ćwiczenie 1 – wykorzystanie mikrokontrolera (ocena 0-10pkt)
c) ćwiczenie 2 – wykorzystanie mikrokontrolera i analogowych układów programowalnych (ocena 0-10pkt),
d) ćwiczenie 3 – wykorzystanie mikrokontrolera, układów analogowych oraz bloków UDB/Verilog/USB (ocena 0-10pkt).

Ćwiczenia powyższe należy wykonać samodzielnie lub w grupach dwuosobowych.

2. Tematy ćwiczeń laboratoryjnych.

Ćwiczenie 1: Należy wykorzystać dołączony do zestawu wyświetlacz LCD. W ramach zadania należy wykonać układ stopera. Przycisk SW3 ma służyć do startu/zatrzymania stopera. Przycisk SW2 ma służyć do zerowania pomiaru. Dodatkowo dioda LED4 w czasie trwania pomiaru ma błyskać z częstotliwością 10Hz, w momencie zatrzymania pomiaru świecić światłem ciągłym a po wyzerowaniu licznika powinna być wygaszona. Należy koniecznie użyć blok TIMERa, który co 10ms ma zgłaszać przerwanie. To przerwanie ma być wzorcem czasu (zliczane) stopera. Zaleca się do sterowania wyświetlaczem wykorzystanie gotowego bloku sprzętowego o nazwie ***Character LCD***.

Ćwiczenie 2-część a): Należy wykorzystać dołączony do zestawu wyświetlacz LCD. W ramach zadania należy wykonać układ mierzący napięcie wytwarzane na potencjometrze umieszczonym na płytce prototypowej oraz układ pomiaru tego samego napięcia wzmocnionego 4 razy przez wzmacniacz analogowy. Na wyświetlaczu powinien być zaprezentowany wynik pomiaru bezpośredniego, pomiaru napięcia wzmocnionego oraz błąd wzmocnienia w postaci odchyłki od wzmocnienia równego 4 wyrażonego w procentach. Zaleca się do sterowania wyświetlaczem wykorzystanie gotowego bloku sprzętowego o nazwie ***Character LCD***.

Ćwiczenie 2-część b): W zadaniu tym należy projekt z ćwiczenia 2a wzbogacić o nadajnik RS232 i wynik pomiaru wysyłać również poprzez nadajnik RS232 do terminala w komputerze PC. Nadajnik RS232 należy wykonać na jeden dowolny z trzech następujących sposobów:

1) Nadajnik RS232 może być opisany w języku HDL Verilog. Do połączenia z komputerem należy użyć kabla RS232 podłączonego do złącza P7.
2) Jako nadajnik RS232 może być wykorzystany gotowy moduł sprzętowy z kolekcji elementów bibliotecznych. Do połączenia z komputerem należy użyć kabla RS232 podłączonego do złącza P7.
3) Nadajnik RS232 powinien być osadzony jako moduł komunikacyjny USB wbudowany w układ PSoC5LP. Do połączenia z komputerem należy wykorzystać kabel Mini-USB podłączony do złącza J2.

Ćwiczenie 3: Bazując na równaniu prądu krzemowej diody półprzewodnikowej należy skonstruować układ pomiaru temperatury. Dla diody półprzewodnikowej zależność opisująca prąd $i_D = f(v_D)$ dana jest następującym równaniem:

$$i_D = I_S\left(e^{\frac{v_D}{mV_T}} - 1\right) \qquad (1)$$

gdzie: I_S - prąd nasycenia, m - współczynnik emisji, $V_T = \frac{kT}{q}$ - potencjał termiczny, k - stała Boltzmana ($k = 1.381 * 10^{-23}\left[\frac{VC}{K}\right]$), T - temperatura bezwzględna w Kelwinach, q - ładunek elektronu

($q = 1.602 * 10^{-19}[C]$), $V_T \approx 25mV$ dla T=300K. Współczynnik emisji dla krzemu jest w przybliżeniu równy 1 i taki należy wstępnie przyjąć w dalszej części, $I_S = J_S A$, gdzie J_S jest parametrem technologicznym nazywanym gęstością prądu nasycenia natomiast A jest powierzchnią przekroju poprzecznego złącza p-n. Jako technikę pomiarową należy użyć pomiaru napięcia na diodzie przy pobudzeniu jej dwoma znanymi i różnymi prądami. Pomijając jedynkę we wzorze (1) napięcie na diodzie można wyrazić równaniem:

$$v_D = mV_T \ln\left(\frac{i_D}{I_S}\right) \tag{2}$$

Zakładając, że pobudzamy diodę dwoma stałymi wartościami prądów I_{D1} oraz I_{D2}, różnicę stałych napięć na diodzie V_{D1} oraz V_{D2} możemy określić jako:

$$V_{D1} - V_{D2} = mV_T \ln\left(\frac{I_{D1}}{I_S}\right) - mV_T \ln\left(\frac{I_{D2}}{I_S}\right) = mV_T \ln\left(\frac{I_{D1}}{I_{D2}}\right) \tag{3}$$

Stąd, po przekształceniach, wyrażenie na temperaturę bezwzględną (w Kelwinach) można uzyskać jako:

$$T = \frac{(V_{D1} - V_{D2})q}{mk \ln(I_{D1}/I_{D2})} \tag{4}$$

W ramach ćwiczenia należy cyklicznie generować 2 różne prądy płynące przez diodę i równocześnie mierzyć napięcie na diodzie a następnie używając wzoru (4) wyznaczyć temperaturę. Wartość tej temperatury wyrażoną w stopniach Celsjusza z dokładnością obliczeniową równą 0.1C należy wyświetlić na wyświetlaczu LCD. Po wstępnym zweryfikowaniu działania układu pomiarowego należy dodać port RS323 (w dowolnej formie jako osadzony lub poprzez USB) i umożliwić zmianę współczynnika emisji *m* w zakresie od 1 do 2 poprzez polecenia z terminala RS232. Kryteria wystawienia oceny: sprawny tylko pomiar z ustalonym m - do 3,5; dodana możliwość zmiany współczynnika m - do 5,0.

Uwaga 1: Zamiast diody pomiarowej praktyczniej jest użyć tranzystor bipolarny w połączeniu diodowym (kolektor zwarty z bazą) a zakres prądów wymuszanych w obwodzie ustawić w zakresie do około 30μA. Współczynnik emisji dla tranzystorów krzemowych jest w równy w przybliżeniu *m* = 1.005.

Uwaga 2: W celu uniknięcia spadków napięć na wewnętrznych kluczach układu połączeniowego należy użyć 4 wyprowadzeń zewnętrznych (zamiast dwóch), 2 z nich do generacji prądu i 2 do pomiaru napięcia na diodzie/tranzystorze. Wyprowadzenia te powinny być zwarte ze sobą poza układem PSoC5LP.

3. Szczegółowy opis przebiegu wykonania ćwiczeń laboratoryjnych.

Poniżej przedstawiono szczegółowy opis postępowania w czasie wykonywania ćwiczeń laboratoryjnych. Opis ten bazuje na przykładzie prostego projektu wykorzystującego jeden przycisk i diodę LED.

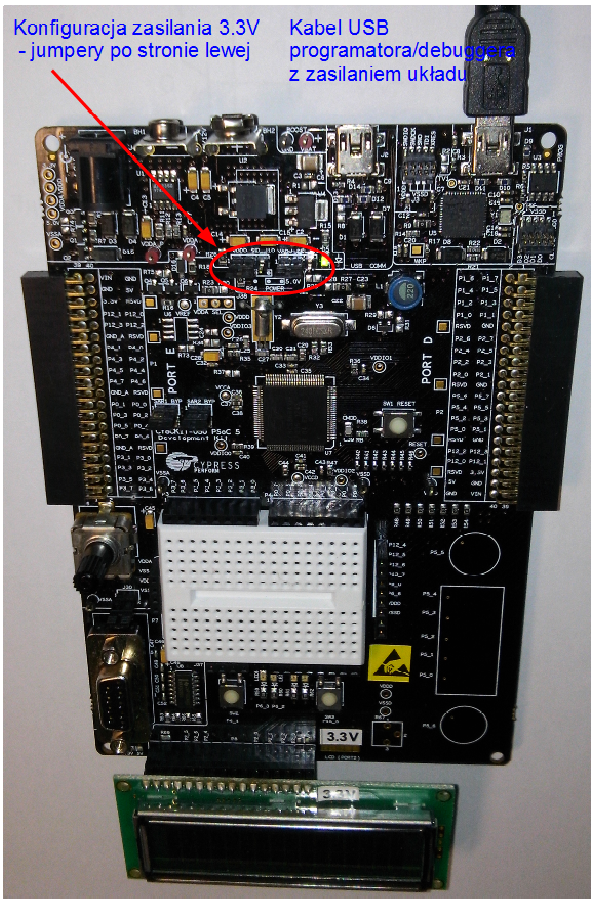
3.1. Wykorzystana płytka prototypowa CY8CKIT-050 5LP.

Dokumentację do płytki prototypowej wykorzystywanej w ramach zajęć laboratoryjnych można znaleźć na stronie internetowej producenta www.cypress.com oraz można również wykorzystać ściągniętą dokumentację na stronie laboratorium: http://www.ue.eti.pg.gda.pl/~bpa/pusoc/kit_files/. Na płytce CY8CKIT-050 5LP umieszczony jest układ PSOC z serii 5LP o symbolu CY8C5868AXI-LP035. Oprócz układu PSOC dostępne są m.in.: diody LED (4 szt.), przyciski (2 szt.), złącze do wyświetlacza i sam wyświetlacz LCD 2 x 16 znaków, pole prototypowe 17 linii po 10 złącz, wyprowadzenia wszystkich portów układu, programator/debuger, złącze USB podłączone do dedykowanych wyprowadzeń układu PSOC, konwerter napięć RS232 wraz ze złączem oraz pozostałe elementy peryferyjne.

UWAGA: dołączony do zestawu wyświetlacz LCD jest przystosowany do zasilania wyłącznie napięciem równym 3,3 V. Złącze do którego podłączany jest wyświetlacz można skonfigurować na 3,3V jak i na 5 V. Skonfigurowanie zasilania na 5 V spowoduje uszkodzenie dołączonego do zestawu wyświetlacza LCD. Domyślna konfiguracja jest na 3,3 V i w czasie zajęć laboratoryjnych nie będzie ona zmieniana.

Zmiana napięcia zasilającego na 5 V wykonywana jest poprzez wlutowanie rezystora R72 i usunięcie rezystora R71.

Konfigurację zworek typu JUMPER do zasilania części analogowej jak i cyfrowej układu PSOC5LP napięciem 3,3 V oraz sposób instalacji wyświetlacza LCD (jeśli jest potrzebny w projekcie) przedstawiono na zdjęciu poniżej:



Rys. 1. Zdjęcie płytki prototypowej CY8CKIT-050 PSOC 5LP wraz z wyświetlaczem LCD i konfiguracją napięcia zasilania VDDA jak i VDDD równą 3,3 V.

3.2. Ćwiczenie wstępne – mruganie diodą LED.

W ramach ćwiczenia wstępnego zostanie wykorzystany przycisk o nazwie SW2 podłączony do portu P6[2] oraz dioda LED o nazwie LED4 podłączona do portu P6[3]. Zarówno przycisk jak i dioda są połączone drugim swoim wyprowadzeniem do linii masy, co można znaleźć na schemacie płytki umieszczonym w dokumentacji płytki na stronie 42. W ramach ćwiczenia wstępnego należy wykonać prostą funkcję polegającą na mruganiu diodą LED4 z częstotliwością 1Hz, natomiast po przyciśnięciu przycisku SW2 częstotliwość ma wzrosnąć do 5Hz. Powyższe zadanie zostanie wykonane na 3 sposoby:

a) całkowicie programowo poprzez wbudowany w układ PSOC mikrokontroler ARM Cortex M3,
b) z wykorzystaniem wbudowanych cyfrowych bloków sprzętowych,
c) z wykorzystaniem bloków UDB i języka HDL Verilog.

Niezależnie od zakresu wykorzystanych zasobów układu PSOC proces projektowania składa się z następujących etapów:

a) narysowanie schematu z bloków dostępnych w PSOC, tą część wykonuje się poprzez przeciąganie dostępnych bloków/elementów z katalogu komponentów (***Component Catalog***) do obszaru schematu a następnie połączenie ich z użyciem menu znajdującego się po lewej stronie arkusza schematu, schemat zapisywany jest w pliku **`*.cysch`**

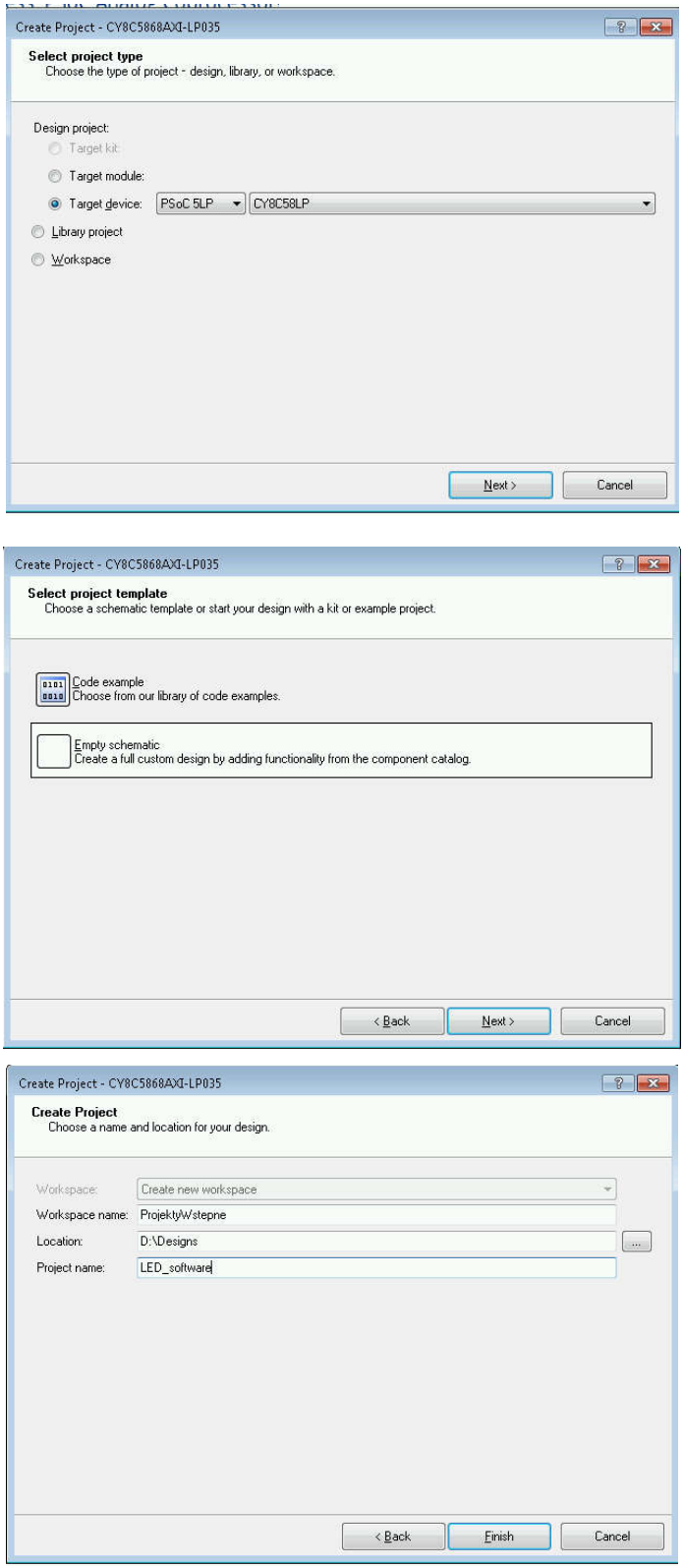
b) skonfigurowanie bloków wykorzystanych w schemacie projektu, konfiguracja polega na podwójnym kliknięciu w dany blok i wprowadzeniu na kolejnych zakładkach wymaganych parametrów i właściwości komponentu,
c) przypisanie portów układu PSOC do wykorzystanych w schemacie wyprowadzeń układu, aby to wykonać należy kliknąć na plik **`*.cydwr`** i w zakładce ***Pins*** dokonać przypisania wyprowadzeń układu PSOC, w pozostałych zakładkach można także dokonać kolejnych szczegółowych konfiguracji,
d) zbudowanie projektu poprzez wybranie menu ***Build/Build Design_Name*** lub naciśnięcie klawiszy ***Shift + F6*** lub przycisku menu  , zbudowanie projektu wygeneruje pliki biblioteczne dla osadzonych w pkt. a) bloków układu, poprzez rozwinięcie drzewa wygenerowanych plików i wybranie pliku nagłówkowego **`*.h`** można zapoznać się z wygenerowanymi automatycznie API dla danego bloku,
e) napisanie aplikacji właściwej w pliku **`main.c`** z odniesieniem do wygenerowanych wcześniej bibliotek,
f) zaprogramowanie układu PSOC poprzez wybranie menu ***Debug/Program*** lub klawiszy ***Ctrl + F5*** lub przycisku menu  i przetestowanie działania projektu,
g) ewentualne debugowanie projektu poprzez wybranie menu ***Debug/Debug***.

Poniżej, szczegółowo, wraz ze zrzutami z ekranu monitora został opisany sposób wykonania projektu wstępnego w trzech wyżej wymienionych wersjach wykorzystania komponentów sprzętowych.

3.2.1. Uruchomienie oprogramowania IDE i utworzenie nowego projektu.

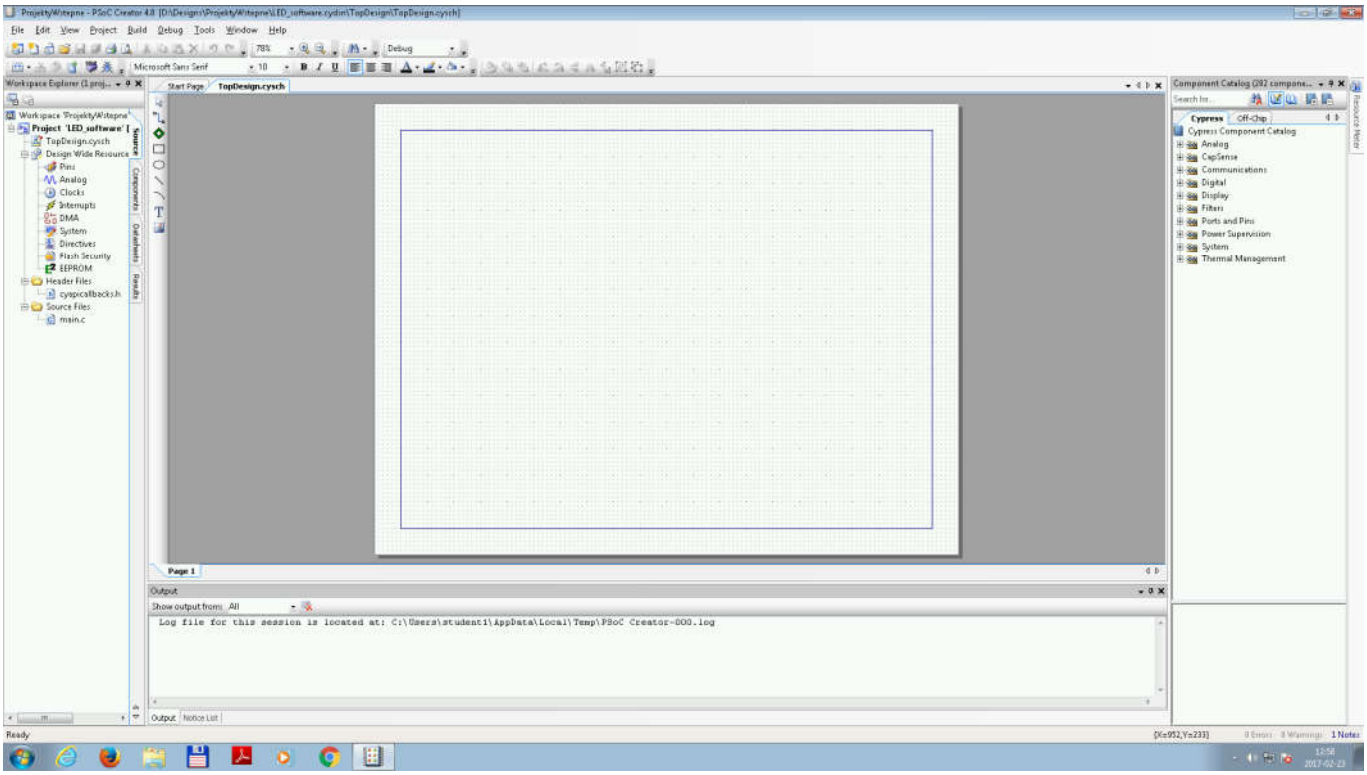
Najpierw należy uruchomić oprogramowanie IDE poprzez wybranie menu ***Start/Wszystkie programy/Cypress/PSoC Creator 4.1/PSoC Creator 4.1***. lub poprzez kliknięcie na ikonę ***PSoC Creator 4.1*** umieszczoną na pulpicie. Następnie tworzymy nową przestrzeń roboczą i nowy projekt w ramach tej przestrzeni. W tym celu wybieramy menu ***File/New/Project*** i wypełniamy zgodnie z poniższym rysunkiem.

Uwaga: zajęcia laboratoryjne oraz projekt mogą się odbywać w sali EA308 lub w sali EA337. W sali EA308 projekty należy zapisywać w folderze **`C:\Designs`** natomiast w sali EA337 w folderze **`D:\Designs`**. Zapis w innym miejscu może spowodować automatyczne skasowanie projektu po ponownym zalogowaniu do komputera.



Rys. 2. Utworzenie nowej przestrzeni roboczej nazwanej ***ProjektyWstepne*** oraz projektu w ramach tej przestrzeni o nazwie ***LED_software***.

Po kliknięciu ***Finish*** okno programu ***PSoC Creator 4.1*** powinno wyglądać jak na poniższym rysunku.



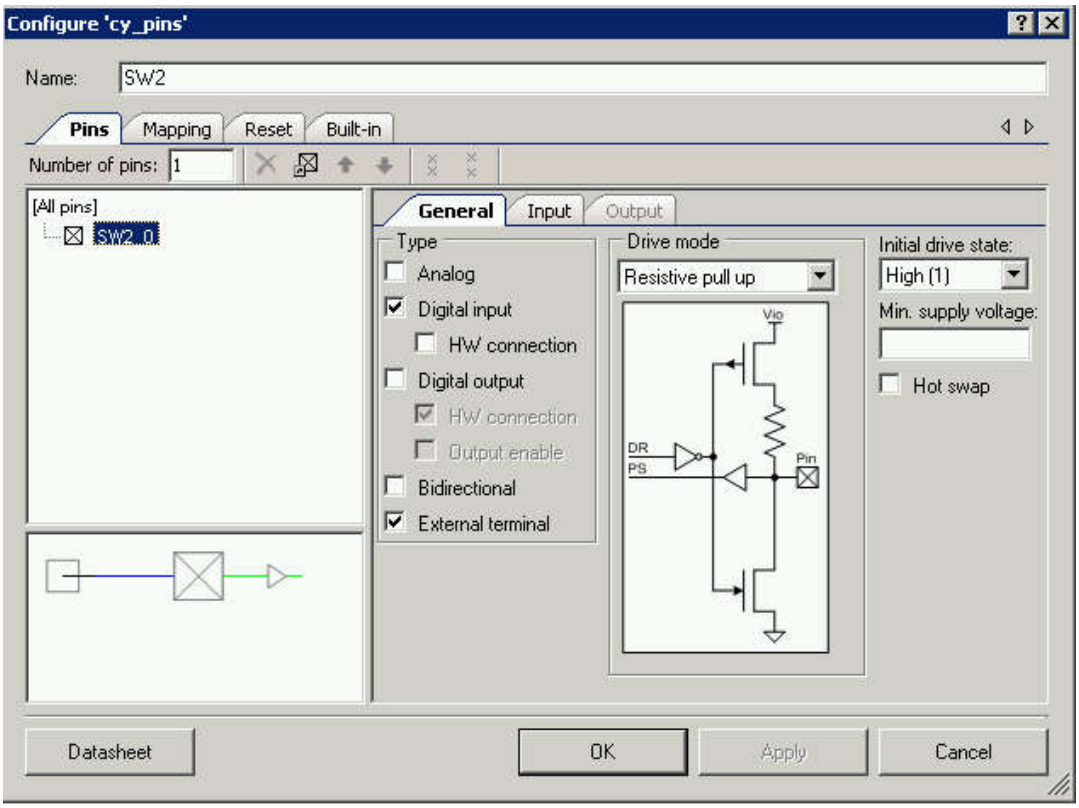
Rys. 3. Okno programu po utworzeniu projektu o nazwie ***LED_software***.

3.2.2. Utworzenie schematu projektu i skonfigurowanie wykorzystanych bloków.

Pomimo, że w pierwszej wersji projektu nie wykorzystujemy bloków sprzętowych a jedynie mikrokontroler, narysowanie schematu umożliwi wykorzystanie wygenerowanych automatycznie API dla wyprowadzeń układu PSoC i w ten sposób umożliwi bardzo szybką realizację projektu bez konieczności szczegółowego wgłębiania się w niezbędne do skonfigurowania rejestry układu. W naszym projekcie schemat składa się wyłącznie z wyprowadzenia wejściowego do którego połączony jest przycisk SW2 oraz wyjściowego do którego podłączona jest dioda LED4. Aby narysować schemat należy w polu ***Component Catalog*** rozwinąć linię ***Ports and Pins*** a następnie przeciągnąć ***Digital Input Pin*** oraz ***Digital Output Pin*** do obszaru **`TopDesign.cysch`**. Obszar ten po przeciągnięciu wyprowadzeń powinien wyglądać jak poniżej:

Rys. 4. Wstawienie wyprowadzeń zewnętrznych do projektu.

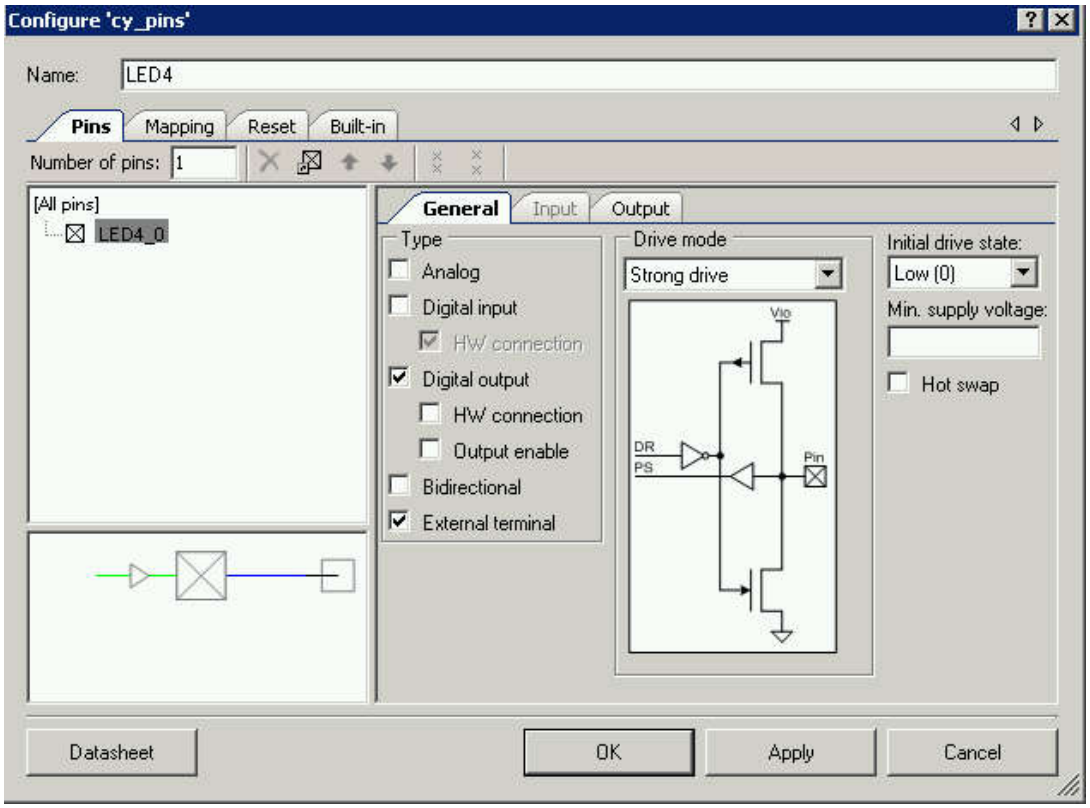
Następnie podwójnie klikamy w wyprowadzenie wejściowe ***Pin_1*** i konfigurujemy jak to przedstawiono poniżej na rysunku.



Rys. 5. Konfiguracja wyprowadzenia wejściowego jako podciągającego do zasilania.

Zaznaczenie opcji ***External terminal*** umożliwia dorysowanie schematu układu realizowanego poza układem PSoC. Ta część schematu kolorowana jest na niebiesko. Schemat układu wewnątrz PSoC kolorowany jest na zielono dla bloków cyfrowych i na czerwono dla bloków analogowych.

Następnie podwójnie klikamy w wyprowadzenie wyjściowe ***Pin_2*** i konfigurujemy jak poniżej na rysunku:



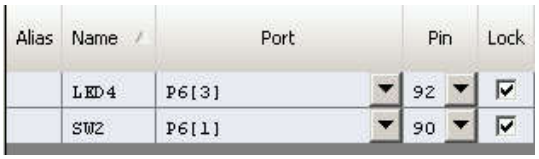
Rys. 6. Konfiguracja wyprowadzenia wyjściowego.

Dla ułatwienia zrozumienia działania układu oraz dla celów dokumentacyjnych warto jest dorysować część projektu realizowaną poza strukturą układu PSoC. W tym celu w polu ***Component Catalog*** klikamy na zakładkę ***Off-Chip*** i dorysowujemy schemat do postaci jak na poniższym rysunku. Aby narysować linię połączeniową wciskamy klawisz ***W***. Aby obrócić element klikamy prawym klawiszem myszki na dany element i wybieramy podmenu ***Shape***.

Rys. 7. Schemat wprowadzony w kolorze niebieskim oznacza elementy zewnętrzne w stosunku do układu PSoC 5LP.

3.2.3. Przypisanie wyprowadzeń do bloków I/O.

W celu przypisania wyjść należy kliknąć w obszarze ***Workspace Explorer*** na ***LED_software.cydwr*** a następnie w zakładce ***Pins*** (u dołu arkusza) w tabeli z prawej strony wybrać dla wyprowadzenia **`SW2`** port **`P6[1]`** a dla **`LED4`** port **`P6[3]`**. Po tej operacji tabela powinna wyglądać jak na poniższym rysunku:



Rys. 8. Przypisanie nazwom logicznym LED4 i SW2 wyprowadzeń układu PSoC 5LP.

Konfigurację należy zapisać poprzez wciśnięcie kombinacji klawiszy ***Ctrl + S***.

3.2.4. Generacja API.

Generację API (ang. Application Programming Interface) wykonuje się poprzez zbudowanie projektu. W tym celu wybieramy menu ***Buid/Build LED_software*** lub naciskamy kombinację klawiszy ***Shift + F6***. Po tej operacji w panelu ***Workspace Explorer*** pojawi się szereg dołączonych plików bibliotecznych do projektu. Pliki biblioteczne zaczynają się nazwą wstawionego do schematu elementu. W naszym przypadku będą to m.in. pliki **`LED4.h`** oraz **`SW2.h`**. Po podwójnym kliknięciu na plik otwiera się jego zawartość i można przeglądać API utworzone dla danego komponentu. Poniżej przykład fragmentu pliku **`SW2.h`**:

```
/***************************************
*        Function Prototypes
***************************************/

void    SW2_Write(uint8 value) ;
void    SW2_SetDriveMode(uint8 mode) ;
uint8   SW2_ReadDataReg(void) ;
uint8   SW2_Read(void) ;
uint8   SW2_ClearInterrupt(void) ;
```

Często po samej nazwie wygenerowanej funkcji można domyślić się jej działania. Szczegóły funkcji dostępne są w dokumentacji danego komponentu, którą można otworzyć poprzez kliknięcie (na schemacie lub katalogu komponentów) prawym klawiszem myszki i wybranie podmenu ***Open Datasheet ...*** .

3.2.5. Utworzenie programu głównego projektu i wgranie do docelowego układu.

Pusty program główny projektu jest automatycznie tworzony i umieszczany w sekcji ***Workspace Explorer*** w pliku ***main.c***. Podwójne kliknięcie powoduje jego otworzenie i umożliwia napisanie programu realizowanego w mikrokontrolerze ARM Cortex M3. Program realizowany w mikrokontrolerach jest zazwyczaj nieskończoną pętlą. W naszym przypadku program modyfikujemy jak przedstawiono poniżej:

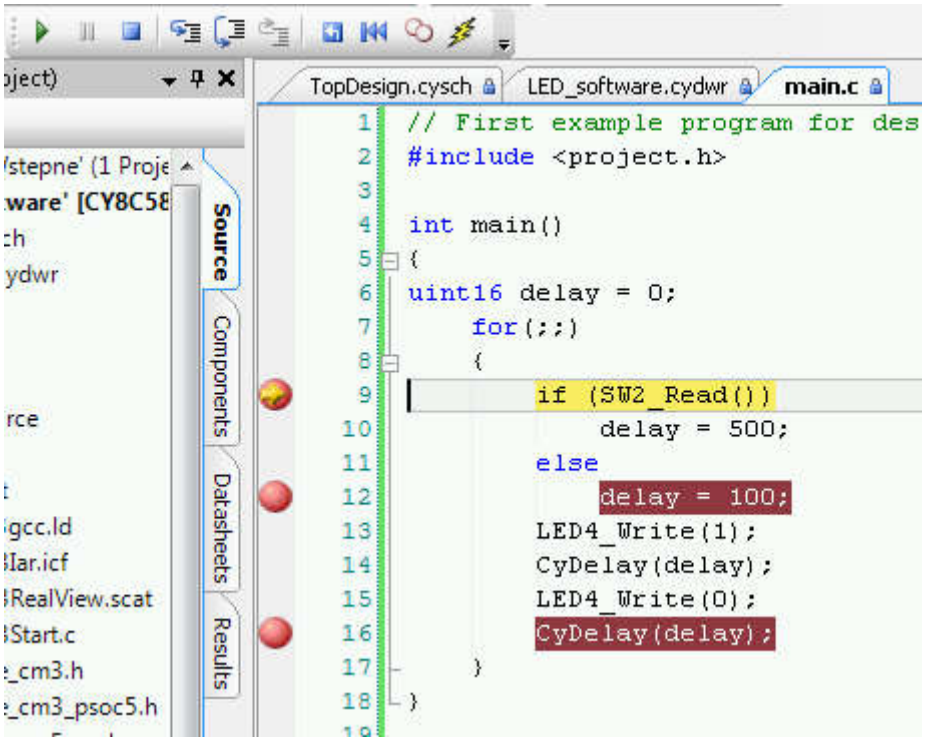
```
// First example program for design LED_software
#include <project.h>

int main()
{
uint16 delay = 0;
    for(;;)
    {
        if (SW2_Read())
            delay = 500;
        else
            delay = 100;
        LED4_Write(1);
        CyDelay(delay);
        LED4_Write(0);
        CyDelay(delay);
    }
```

}

Po utworzeniu programu możemy go skompilować i zaprogramować układ docelowy PSoC 5LP. W tym celu wybieramy menu ***Debug/Program*** lub kombinację klawiszy ***Ctrl + F5***.

3.2.6. Debugowanie projektu.

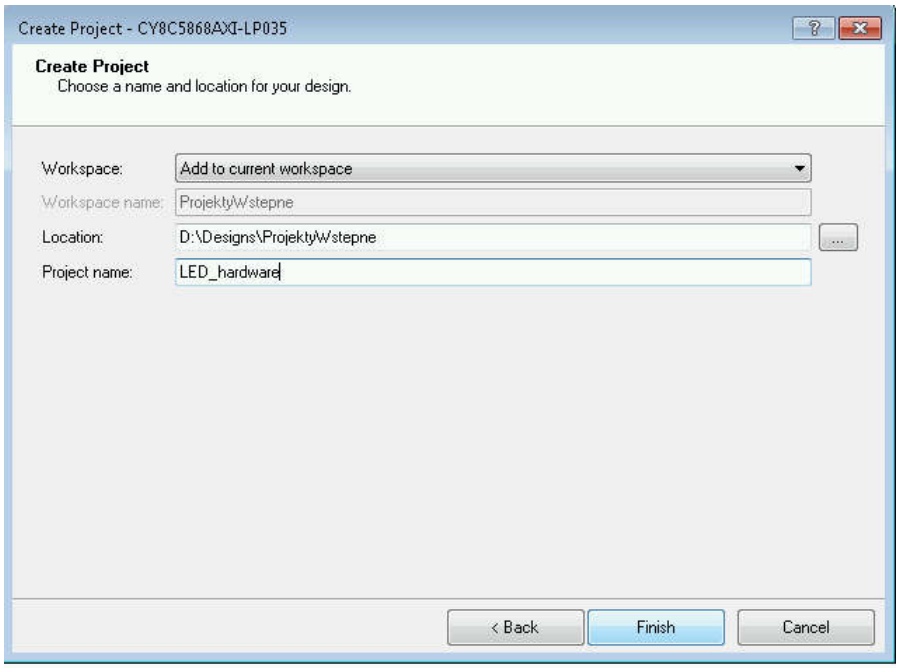
W przypadku gdy wgrany do układu PSoC 5LP projekt działa nieprawidłowo można wykonać jego debugowanie. Najpierw należy połączyć debuger z układem docelowym (menu ***Debug/Select Debug Target***) a następnie należy wskazać połączony układ PSoC i nacisnąć przycisk ***Connect***. Kolejnym krokiem jest uruchomienie debugera poprzez menu ***Debug/Debug*** lub klawisz ***F5***. Po uruchomieniu debugera pojawi się typowe menu dla tego rodzaju programów z możliwością wykonywania pojedynczych instrukcji, startu i zatrzymania programu, wchodzenia i wychodzenia z danych funkcji, wstawiania i usuwania breakpointów. W czasie debugowania poszczególne rozkazy programu wykonywane są w rzeczywistym układzie PSoC.



Rys. 9. Uruchomiony debuger projektu. Żółta strzałka wskazuje bieżący punkt zatrzymania programu, czerwone kropki wskazują miejsca wstawienia breakpointów.

3.2.7. Dodanie nowego projektu do bieżącej przestrzeni roboczej.

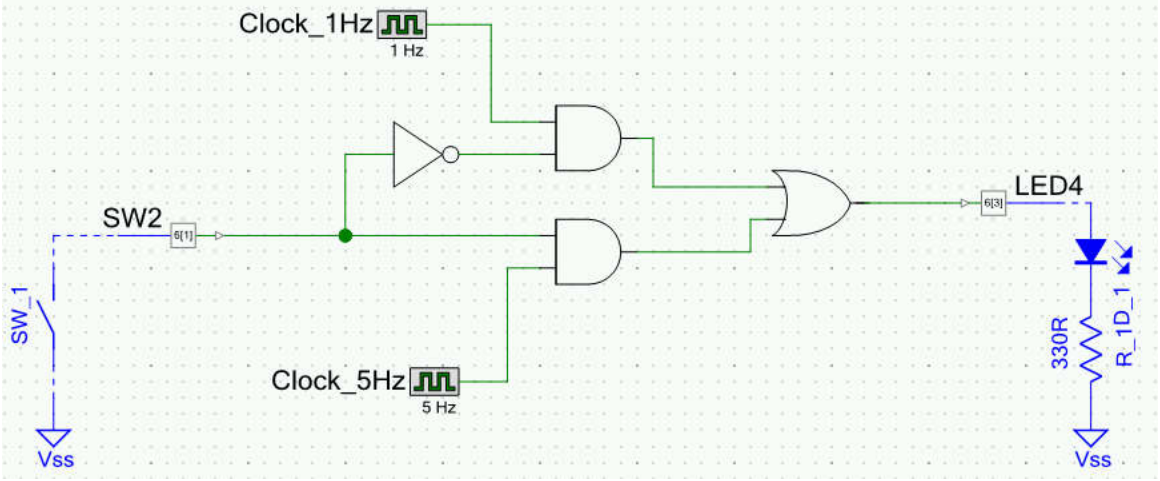
W celu wykonania zadania w sposób całkowicie sprzętowy do bieżącej przestrzeni roboczej zostanie dodany nowy projekt o nazwie **`LED_hardware`**. W tym celu należy wybrać menu ***File/New/Project*** a następnie należy wypełnić formularze jak poprzednio z wyjątkiem ostatniego, trzeciego, który należy wypełnić jak na poniższym rysunku i nacisnąć przycisk ***Finish***. W polu ***Workspace Explorer*** pojawi się nowy projekt, którego nazwa zostanie automatycznie pogrubiona, co oznacza że jest to aktywny projekt.



Rys. 10. Dodanie projektu „LED_hardware" do bieżącej przestrzeni roboczej.

3.2.8. Wykonanie projektu w wersji sprzętowej.

Podobnie jak poprzednio, pierwszym krokiem jest utworzenie schematu projektu. Tym razem korzystamy z elementów sprzętowych dostępnych w układzie PSoC 5LP w taki sposób aby nie było potrzeby korzystania z mikrokontrolera. Przykład możliwego rozwiązania zadania w sposób sprzętowy przedstawiony jest na poniższym rysunku:



Rys. 11. Schemat układu realizującego miganie diody LED4 z dwoma różnymi częstotliwościami w zależności od stanu przycisku SW2.

Kolejnym etapem jest przyporządkowanie wyprowadzeń, co należy wykonać identycznie jak w punkcie 3.2.3. Następnie należy postępować zgodnie z punktami 3.2.4-6 przy czym program główny należy zostawić pusty a debugowanie wg punktu 6 jest bezcelowe bo mikrokontroler nie wykonuje żadnych istotnych operacji (pusta pętla).

3.2.9a. Projekt w wersji z wykorzystaniem bloków UDB i języka Verilog.

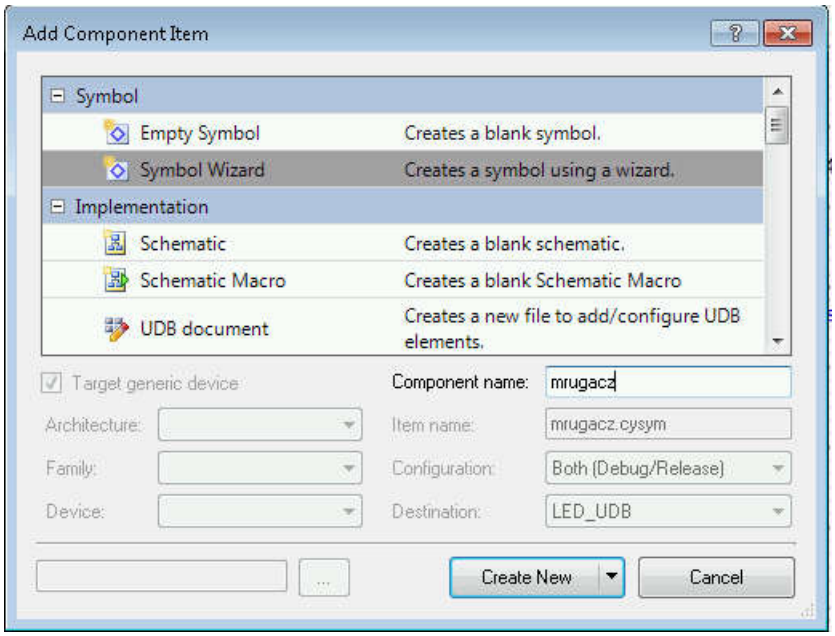
Najpierw dodajemy nowy projekt o nazwie LED_UDB do bieżącej przestrzeni roboczej. Następnie rysujemy schemat z elementów jak na rysunku poniżej, funkcje logiczne na sygnałach zegara i wejściowym zrealizowane zostaną w dodatkowym komponencie, który zostanie opisany w języku Verilog.

Rys. 12. Inicjalny schemat do ćwiczenia z wykorzystaniem bloków UDB i języka Verilog.

W kolejnym kroku należy przypisać odpowiednie wyprowadzenia zewnętrzne układu PSoC do wyprowadzeń SW2 oraz LED4 (jak w punkcie 3.2.3). W tym stanie projekt jest gotowy do uzupełnienia o nowy komponent opisany w działaniu poprzez język Verilog.

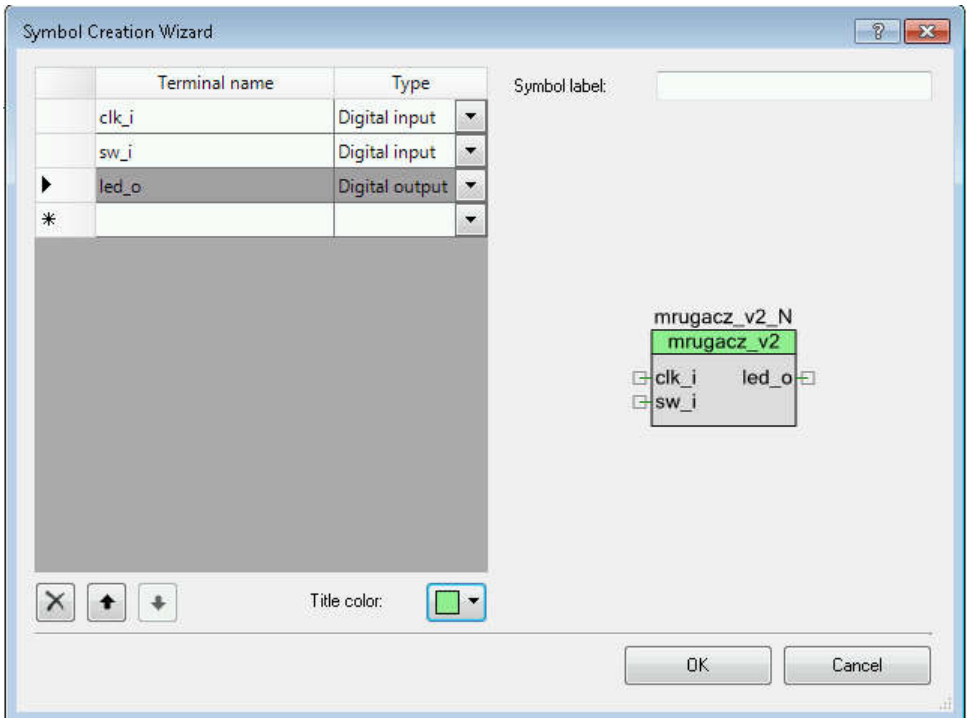
3.2.9b. Utworzenie nowego komponentu i opisanie jego działania w języku Verilog.

W celu utworzenia symbolu nowego komponentu należy w polu ***Workspace Explorer*** wybrać boczną zakładkę ***Components*** a następnie kliknąć prawym klawiszem myszki na nazwie projektu i wybrać podmenu ***Add Component Item***. Następnie w formularzu należy wybrać **`Symbol Wizard`** i nadać nazwę nowo tworzonemu komponentowi np. **`mrugacz`**.



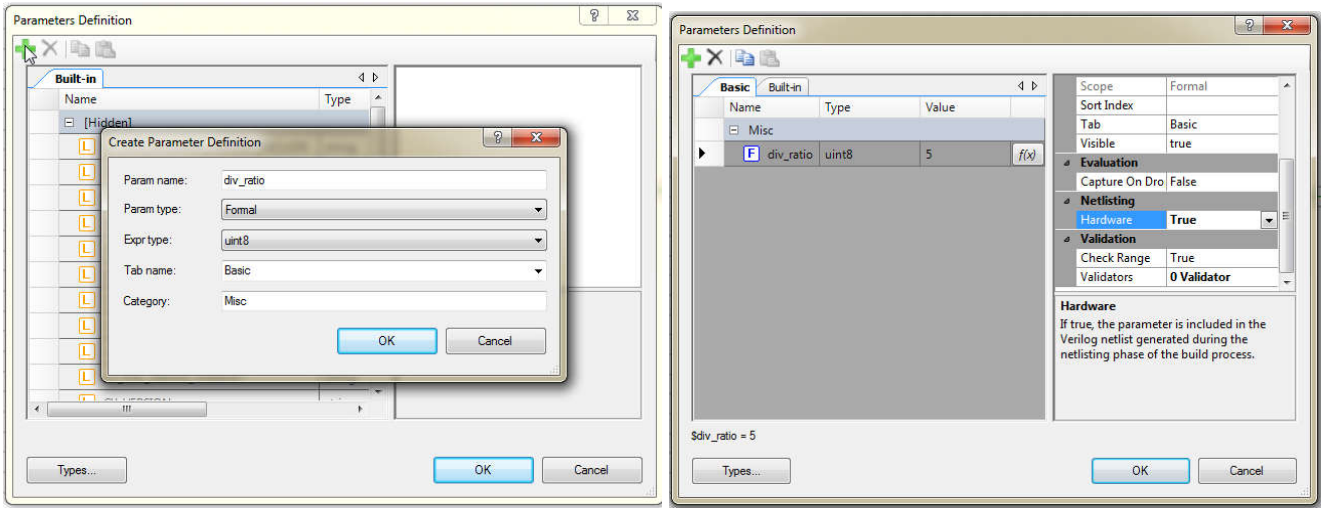
Rys. 13. Dodawanie własnego komponentu do biblioteki.

Po naciśnięciu przycisku ***Create New*** pojawi się nowe okno, gdzie należy podać nazwy i kierunki sygnałów tworzonego komponentu. Jako kolor nagłówka zaleca się wybrać zielony, co jest zgodne z zasadami kolorowania układów cyfrowych w IDE.



Rys. 14. Konfigurowanie symbolu komponentu.

Po kliknięciu ***OK*** pojawi się graficzny edytor symbolu, który umożliwia dalsze zmodyfikowanie symbolu. W naszym przypadku dodamy do symbolu parametr, który później przy osadzaniu komponentu będzie można modyfikować. W tym celu klikamy w pustym miejscu schematu komponentu prawym klawiszem myszki i wybieramy podmenu ***Symbol Parameters...*** a następnie klikamy w duży zielony symbol + i wypełniamy kolejne dane jak na rys. 15 z lewej strony. Po kliknięciu ***OK*** pojawi się formularz jak na rys. 15 po stronie prawej. Tutaj wstawiamy wartość parametru na równą 5 oraz zmieniamy wartość pola ***Hardware*** na ***True*** co umożliwi później wstawianie tego komponentu z dostosowaną indywidualnie wartością parametru **`div_ratio`**.



Rys. 15. Konfigurowanie własnego parametru o nazwie **`div_ratio`**.

Klikamy ***OK*** a następnie ponownie klikamy w pustym miejscu schematu komponentu prawym klawiszem myszki i wybieramy podmenu ***Generate Verilog...*** W nowym oknie klikamy ***Generate*** co spowoduje automatyczną generację kodu Verilog zawierającego tylko deklarację modułu, portów i parametrów. Kod ten należy uzupełnić opisem działania układu. Poniżej przedstawiona jest przykładowa zawartość kodu realizująca podział częstotliwości sygnału **`clk_i`** przez współczynnik równy parametrowi ***div_ratio*** i

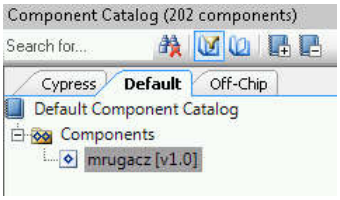
przekazujący do wyjścia sygnał **clk_i** lub podzielony sygnał **clk_i** w zależności od wartości sygnału **sw_i**.

```
//`#start header` -- edit after this line, do not edit this line
// ========================================
// (c) KSMI PG
// ========================================
`include "cypress.v"
//`#end` -- edit above this line, do not edit this line
module mrugacz (
      output led_o,
      input   clk_i,
      input   sw_i
);
      parameter div_ratio = 5;

//`#start body` -- edit after this line, do not edit this line

reg [4:0] counter;
reg led;
assign led_o = (sw_i==1) ? clk_i : led;
always @ (posedge clk_i)
begin
    if (counter == div_ratio-1)
        counter = 0;
    else
        counter = counter+1;
    if (counter == (div_ratio-1)/2)
        led = 1;
    if (counter == 0)
        led = 0;
end
//`#end` -- edit above this line, do not edit this line
endmodule
//`#start footer` -- edit after this line, do not edit this line
//`#end` -- edit above this line, do not edit this line
```

W czasie budowania projektu (klawisze ***Shift + F6***) ewentualne błędy zostaną wyszczególnione w oknie wyjściowym, należy je wtedy poprawić i ponowić budowanie.

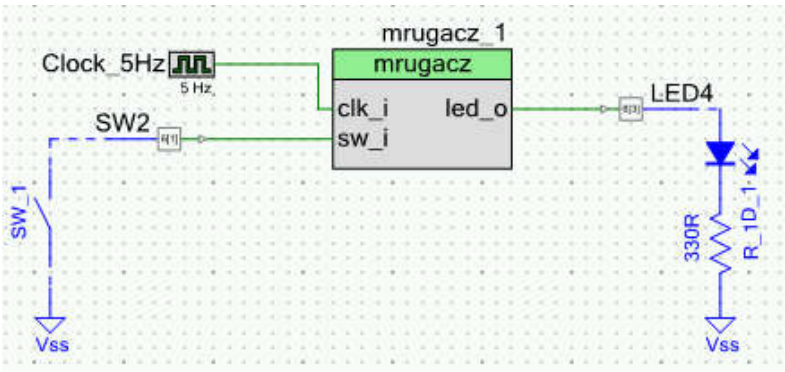
3.2.9c. Wstawienie nowo utworzonego komponentu w projekcie i końcowa implementacja do układu PSoC 5LP.

Kolejnym etapem jest wstawienie nowo utworzonego komponentu. W tym celu należy przełączyć się na schemat projektu i z pola ***Component Catalog*** z zakładki ***Default*** przeciągnąć element do schematu.



Rys. 16. Domyślne umiejscowienie biblioteki własnych komponentów.

Następnie należy wykonać połączenia aby uzyskać schemat jak na poniższym rysunku.



Rys. 17. Schemat projektu z wykorzystaniem własnego komponentu opisanego przy użyciu języka HDL Verilog.

Kolejne kroki są zgodne z punktami 3.2.4 - 6 przy czym program główny należy zostawić pusty a debugowanie wg punktu 6 jest bezcelowe gdyż mikrokontroler nie wykonuje żadnych istotnych operacji (pusta pętla).

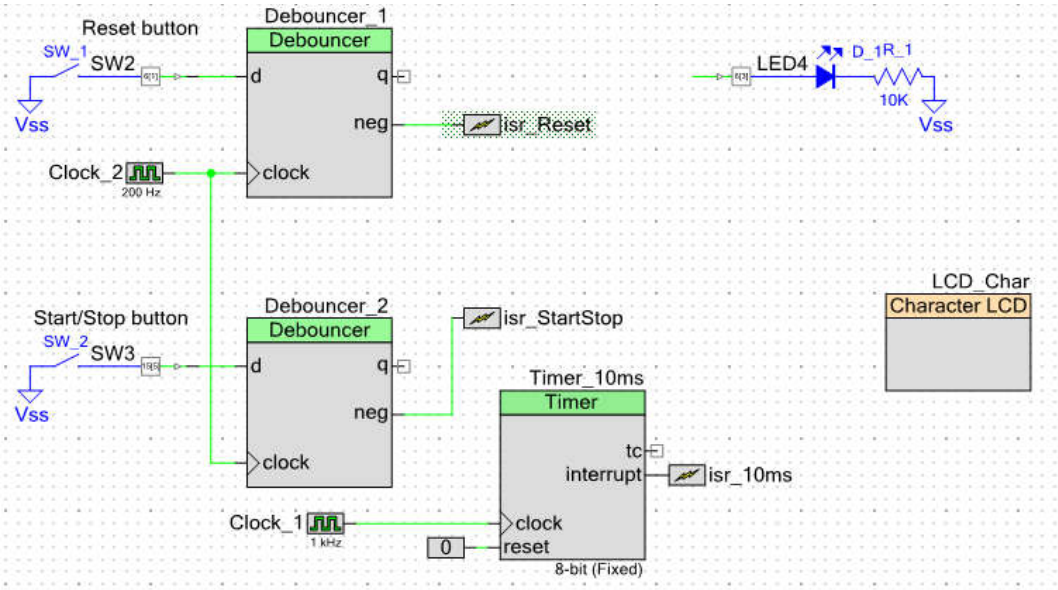
4. Dodatki przydatne w realizacji zadań laboratoryjnych.

4.1. Wykorzystanie przerwań.

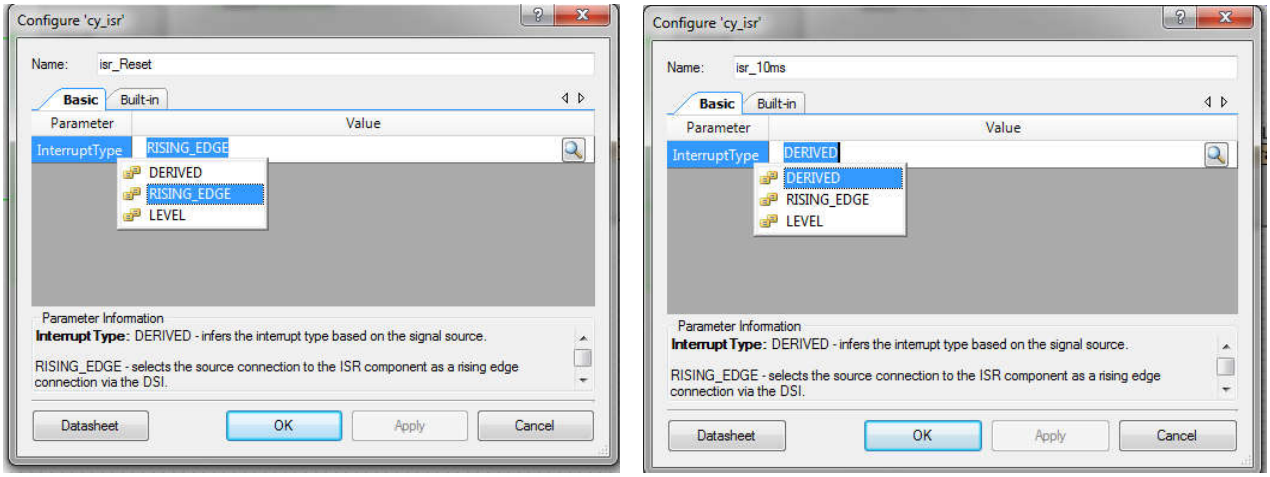
Wykorzystanie przerwań w układach PSoC opisane jest szczegółowo w dokumencie AN54460. Poniżej wyszczególniono najważniejsze kroki niezbędne do prawidłowego skonfigurowania przerwania:

4.1.1. Narysowanie schematu i skonfigurowanie przerwania.

W tym celu należy narysować schemat zawierający komponent **`System\Interrupt`** a następnie go skonfigurować. Przykład schematu zawierającego 3 przerwania przedstawiony jest poniżej. Dwa z nich (**`isr_Reset`** oraz **`isr_StartStop`**) są ustawione na wyzwalanie zboczem narastającym a przerwanie **`isr_10ms`** jest ustawione na wyzwalanie zdarzeniem.



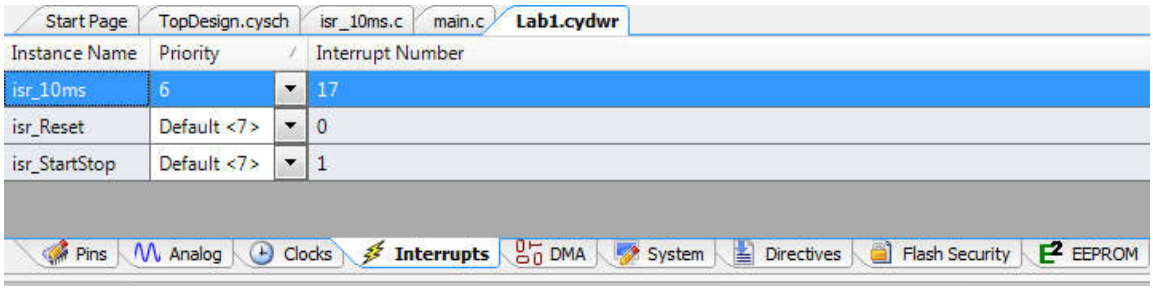
Rys. 18. Schemat układu wykorzystującego przerwania.



Rys. 19. Konfiguracja przerwań.

4.1.2. Konfiguracja priorytetów przerwań.

W celu skonfigurowania przerwań należy w oknie **`Workspace Explorer`** kliknąć na plik *.cydwr a następnie w dolnej zakładce **`Interrupts`** skonfigurować priorytety przerwań. Domyślnym priorytetem jest poziom 7. Najwyższy priorytet to poziom 0. Poniżej przedstawiono przykładową konfigurację priorytetów przerwań.



Rys. 20. Konfiguracja przerwań.

4.1.3. Generacja API.

W celu generacji API wciskamy ***Shift + F6*** lub menu ***Build/Build***. W oknie ***Window Explorer*** w folderze ***Generated_Source*** pojawią się wygenerowane automatycznie pliki nagłówków i funkcji obsługi.

4.1.4. Uzupełnienie kodu obsługi przerwania.

Zadaniem projektanta jest uzupełnienie automatycznie utworzonych kodów obsługi przerwania. W tym celu w oknie ***Window Explorer*** odszukujemy plik **`*.c`** zawierający nazwę przerwania (w naszym przypadku będzie to plik o nazwie np.: **`isr_10ms.c`**) i uzupełniamy 2 następujące miejsca w pliku:

a) miejsce deklaracji własnych elementów języka, przykład uzupełnienia poniżej,

```
/*******************************************************************************
*  Place your includes, defines and code here
********************************************************************************/
/* `#START isr_10ms_intc` */
#include "Timer_10ms.h"
extern uint8 flag;
struct time_st
      {
            unsigned char min2;unsigned char min1;unsigned char secs2;unsigned char
secs1;unsigned char ssec2;unsigned char ssec1;
      };
extern struct time_st time;
```

```
/* `#END` */
```

b) funkcja obsługi przerwania, w naszym przykładzie będzie to funkcja **CY_ISR(isr_10ms_Interrupt)**, funkcje obsługi przerwań mają nazwę CY_ISR a jako argument przekazywana jest nazwa przerwania, przykład uzupełnienia poniżej,

```
CY_ISR(isr_10ms_Interrupt)
{
    /*  Place your Interrupt code here. */
    /* `#START isr_10ms_Interrupt` */
      Timer_10ms_ReadStatusRegister(); <-- ten fragment kodu dezaktywuje przerwanie
                                       <--Timer_10ms i umożliwia wyjście z niego

if (flag == 1)
{
.
.
.
}
    /* `#END` */
}
```

Należy zawsze wprowadzać własny kod pomiędzy znaczniki:
`/* `#START` **nazwa uzupełnianego fragmentu**` */`
`/* `#END` */`
bo w przeciwnym przypadku ponowna generacja API (klawisze ***Shift+F6***) usunie poprzednio wprowadzony kod.

4.1.5. Uzupełnienie kodu programu głównego.

W programie głównym projektu **main.c** należy włączyć obsługę przerwań (indywidualnych i globalnie zezwolić na przerwania) oraz zdeklarować zmienne globalne, które mają być widziane przez funkcje obsługi przerwań. Poniżej przedstawiono fragmenty kodu ze stosownym komentarzem.

```
uint8 flag = 0;  //flag=0 counter stops, flag=1 counter runs  <--deklaracja zmiennych
uint8 state = 0; //state=0 - RESET, state=1 RUN, state=2 STOP <--globalnych
struct time_st                                                <--widocznych w funkcjach
      {                                                       <--obsługi przerwań
            unsigned char min2;unsigned char min1;unsigned char secs2;unsigned char
secs1;unsigned char ssec2;unsigned char ssec1;
      };
struct time_st time;

int main()
{
    // variables declaration

      // components and interrupt init
      Timer_10ms_Start();                  <-- uruchomienie Timera (i jego przerwania)
      isr_10ms_Start();                    <-- uruchomienie przerwania isr_10ms
      isr_Reset_Start();                   <-- uruchomienie przerwania isr_Reset
      isr_StartStop_Start();               <-- uruchomienie przerwania isr_StartStop
      CyGlobalIntEnable;                   <-- globalne zezwolenie na przerwania
.
.
.
```

4.2. Wykorzystanie złącza USB jako portu RS232

Aby szybko wykorzystać port USB jako RS232 najlepiej jest skorzystać z gotowego przykładu.

4.2.1. Wykorzystanie projektu przykładowego.

Aby wczytać projekt przykładowy i dołączyć do istniejącej przestrzeni roboczej należy wybrać menu ***File/New/Project*** a następnie kliknąć w polu ***Next*** a na następnym oknie zaznaczyć wybrać **`Code Example`** i wybrać interesujący projekt z listy dostępnych. Można filtrować listę poprzez wybór słowa kluczowego, w naszym wypadku wybieramy **`UART`** jako słowo kluczowe a następnie projekt **`USB_UART`** i klikamy na przycisk ***Next*** i ***Finish***. Następnie projekt należy przeglądnąć, skompilować i sprawdzić działanie a w dalszej kolejności skopiować poprzez schowek zarówno wybrane fragmenty schematu jaki i kodu do projektu docelowego.

4.3. Pytania i odpowiedzi (Q&A)

Q1: Funkcje **`sprinf`** oraz **`printf`** nie drukują wartości typu **`float`**. Co należy zmienić aby można było powyższych funkcji używać do drukowania typów zmiennoprzecinkowych?

A1: Domyślnie wyłączona jest biblioteka odpowiadająca za formatowanie zmiennych typu **`float`**. Aby ją włączyć należy:

- w menu **`Project/Build Settings/ARM GCC.../Linker`** ustawić opcję **`Use newlib-nano Float Formatting`** na **`True`** oraz
- w menu **`Design Wide Resurces`** w zakładce **`System`** należy ustawić wielkość **`Heap Size`** na **`0x0200`**.

Alternatywnie można samemu napisać funkcję formatującą zmienne typu **`float`**. Przykład takiej funkcji wraz z wywołaniem przedstawiony jest poniżej. Funkcja **`float_to_string`** zwraca sformatowany string **`text_pointer`** zawierający wartość zmiennej **`number`** z liczbą miejsc po przecinku równą **`dec_points`**.

```
#include "project.h"
#include "stdio.h"
void float_to_string(char text_pointer[17],float number,uint8 dec_points)
{
    uint8 i,j;
    int32 full_number;

    for(i=0;i<dec_points;i++)
        {
           number = number*10.0;
        }
    full_number = number;
    sprintf(text_pointer,"%ld",full_number);
    j=strlen(text_pointer);
    if (j<=dec_points)
    {
        for (i=dec_points;i>=1;i--)
            {
                if (i-dec_points+j>0)
                    text_pointer[i+1]=text_pointer[i-dec_points+j-1];
                else
                    text_pointer[i+1]='0';
            }
        text_pointer[dec_points+2]='\0';
        text_pointer[1]='.';
        text_pointer[0]='0';
    }
    else
    {
        j=strlen(text_pointer);
```

```
        for (i=j;i>=j-dec_points;i--)
            {
                text_pointer[i+1]=text_pointer[i];
            }
        text_pointer[j+2]='\0';
        text_pointer[j-dec_points]='.';
    }
}
```

Przykładowe wywołanie funkcji:

```
char text[17]="";
float32 tK10,tC10=25.0;
float_to_string(text,tC10,2);
```

Q2: Dlaczego przetworniki ADC generują wartości odczytywanych napięć w zakresie do 5V podczas gdy spodziewany zakres napięć jest do 3.3V?

A2: Prawdopodobnie jest błędnie ustawiona wartość napięcia zasilającego używana jako napięcie referencyjne. Aby to sprawdzić lub zmienić tą wartość należy w menu **`Design Wide Resurces`** w zakładce **`System`** sprawdzić ustawione wartości napięć zasilających. Domyślne ustawienie na płytkach wykorzystywanych w laboratorium wynosi 3.3V dla wszystkich rodzajów napięć.