C2000 Piccolo LanuchPad (6)

Łatwa inicjalizacja systemowa procesora serii Piccolo F2802x

Inicjalizacja procesora serii Piccolo F2802x jest dosyć skomplikowana. Zastosowanie biblioteki driverlib z pakietu programowego controlSUITE znacznie ułatwia wykonanie tego zadania. Zaprezentowane w artykule postępowanie pozwala na dokładne zapoznanie się z inicjowaniem systemowym procesorów serii Piccolo F2802x z użyciem biblioteki driverlib w środowisku programowym CCSv5.

Do tworzenia w środowisku CCSv5 programów przeznaczonych dla procesorów rodziny TMS320Piccolo F2802x firmy Texas Instruments potrzebny jest pakiet programowy controlSUITE tej firmy. Zawiera on oprogramowanie „firmware", biblioteki, opisy zestawów sprzętowych oraz projekty przykładowe dla wszystkich serii procesorów rodziny C2000. Projekty przykładowe pakietu controlSUITE zwierają na początku kodu programu sekwencję inicjalizacji systemowej układu procesorowego serii Piccolo F2802x.

Konfiguracja sprzętowa i programowa

Do wykonania ćwiczenia potrzebny jest komputer z zainstalowanym (darmowym) oprogramowaniem:

- Środowisko *Code Composer Studio* v5.4.0 (Maj 2013) firmy Texas Instruments [1, 13, 15]. Umożliwia tworzenie w środowisku CCSv5 programów przeznaczonych dla procesorów serii Piccolo TMS320F2802x.
- Pakiet programowy controlSUITEv3.2.1 (Czerwiec 2013) firmy Texas Instruments [2, 13, 15]. Zawiera oprogramowanie „firmware", biblioteki, opisy zestawów sprzętowych oraz projekty przykładowe dla wszystkich serii procesorów rodziny C2000.

Platforma sprzętowa wymaga tylko jednego elementu:

- Zestaw ewaluacyjny *C2000 Piccolo LaunchPad* firmy Texas Instruments z układem procesorowym TMS320F28027 Piccolo firmy Texas Instruments (zawiera kabel USB-A USB-mini) [10, 12]

W folderze *C:\home_dir* komputera zostanie utworzony nowy folder *work_SYS*. Wymagane są prawa dostępu (zapisu i modyfikacji) dla tej ścieżki dyskowej. Możliwe jest umieszczenie foldera *home_dir* na innym wolumenie dyskowym z prawami dostępu.

Cel ćwiczenia

Celem ćwiczenia jest praktyczne poznanie inicjalizowania systemowego układu procesorowego serii Piccolo F2802x przy użyciu biblioteki *driverlib* pakietu programowego controlSUITEv3 oraz środowiska *Code Composer Studio* v5. Zastosowano przykładowy projekt ***Example_F2802xGpioSetup*** z tego pakietu pracujący na zestawie ewaluacyjnym *C2000 Piccolo LaunchPad*. Ćwiczenie jest zorganizowane tak, że działania są wykonywane w kolejnych punktach i krokach uzupełnionych o opisy.

Ćwiczenie umożliwia: poznanie organizacji plików nagłówkowych projektu oraz poznanie budowy i inicjalizowania: układu generacji sygnału zegarowego, modułu CPU Watchdog, układu obsługi przerwań i modułu PIE oraz kalibracji oscylatorów wewnętrznych i modułu ADC.

Dodatkowe informacje

Dotychczas w EP na temat zestawu ewaluacyjnego C2000 Piccolo LaunchPad:

- „Zestaw ewaluacyjny C2000 Piccolo LaunchPad", EP 01/2013
- „C2000 Piccolo LanuchPad (1) – Pierwszy program w środowisku programowym CCS v5", EP 02/2013
- „C2000 Piccolo LanuchPad (2) – Łatwe programowanie z pakietem controlSUITE", EP 03/2013
- „C2000 Piccolo LanuchPad (3) – Łatwe programowanie do pamięci Flash", EP 04/2013
- „C2000 Piccolo LanuchPad (4) - Łatwa obsługa szyny SPI", EP 05/2013
- „C2000 Piccolo LanuchPad (5) - Łatwa obsługa szyny I²C", EP 07/2013

Opisy

Dane techniczne i parametry elektryczne układu procesorowego serii Piccolo F2802x są zamieszczone w dokumencie Texas Instruments [3] a istotne informacje na temat błędnego działania układu procesorowego serii Piccolo F2802x zawiera errata [4]. Opis układu generacji sygnału zegarowego, modułu CPU Watchdog, obsługi przerwań, modułu PIE oraz modułu GPIO układu procesorowego serii Piccolo F2802x jest zamieszczony w dokumencie *TMS320x2802x Piccolo System Control and Interrupts* [5]. Procedura *Device_cal* jest umieszczona w pamięci Boot ROM układów procesorowych serii F2802x i jest opisana w dokumencie *TMS320x2802x Piccolo Boot ROM Reference Guide* [7]. Opis opis zestawu ewaluacyjnego *C2000 Piccolo LaunchPad* jest zamieszczony w dokumencie *LAUNCHXL-F28027 C2000 Piccolo LaunchPad Experimenter Kit, User's Guide* [10]. Opis oprogramowania „firmware" pakietu programowego controlSUITEv3 jest zamieszczony w dokumencie *F2802x Firmware Development Package USER'S GUIDE v. 210* [8]. Opis biblioteki *driverlib* pakietu programowego controlSUITEv3 jest zamieszczony w dokumencie *F2802x Peripheral Driver Library USER'S GUIDE v. 210* [9].

Dokładne omówienie budowy układu procesorowego serii Piccolo F2802x jest zamieszczone w książce: Henryk A. Kowalski, „Procesory DSP dla praktyków", BTC, Warszawa, 2011 [13]

Dokładne omówienie inicjowania układu procesorowego serii Piccolo F2802x jest zamieszczone w książce: Henryk A. Kowalski, „Procesory DSP w przykładach", BTC, Warszawa, 2012 [15].

Dokładne omówienie zestawu ewaluacyjnego *C2000 Piccolo LaunchPad* jest zamieszczone w artykule: Henryk A. Kowalski, „Zestaw ewaluacyjny C2000 Piccolo LaunchPad", EP 01/2013 [12].

Dokładne omówienie środowiska CCSv5 oraz pakietu controlSUITEv3 jest zamieszczone w artykule: Henryk A. Kowalski, „C2000 Piccolo LanuchPad (2) – Łatwe programowanie z pakietem controlSUITE", EP 03/2013 [13].

Model programowy procesora serii Piccolo F2802x

Pakiet programowy „firmware" (F2802x Firmware Development Package) dostarcza wsparcia dla dwóch modeli programowania układów procesorowych serii F2802x Piccolo. Jest to: model bezpośredniego dostępu do rejestrów (header files) oraz model drajwerów programowych (library). Każdy z tych modeli może być zastosowany osobno lub łącznie. Opis jest zamieszczony w dokumentach [8, 9] dostępnych w ścieżce *doc* pakietu programowego „firmware".

Biblioteka *driverlib* modelu drajwerów programowych dostarcza API do sterowania modułami peryferyjnymi. Drajwery programowe zapewniają kontrolę nad modułami i nie wymagają bezpośredniego dostępu do ich rejestrów. Model może nie udostępniać wszystkich możliwości funkcjonalnych modułu peryferyjnego. Opis modelu jest zamieszczony w dokumentach [8, 9] dostępnych w ścieżce *doc* pakietu programowego „firmware".

Zastosowanie w programie modelu drajwerów programowych wymaga dołączenia do projektu biblioteki *driverlib.lib*. Na początku programu należy włączyć pliki nagłówkowe drajwerów modułów, które będą w programie używane. Następnie należy zainicjować wskazanie (handle) na strukturę dla tego modułu. Potem można już używać funkcji obsługi tego modułu.

Metody obsługi modułów peryferyjnych układów procesorowych serii Piccolo F2802x udostępniane przez bibliotekę *driverlib* są bardzo podobne dla wszystkich modułów.

Podłączenie i skonfigurowanie zestawu C2000 Piccolo LaunchPad

Po zainstalowaniu środowiska CCSv5 [1, 13] można pierwszy raz dołączyć zestaw ewaluacyjny C2000 Piccolo LaunchPad [10, 12] kablem USB do wolnego portu USB komputera. System Windows automatycznie rozpoznaje układ. Zostaną zainstalowane sterowniki systemu Windows dla emulatora XDS100v2 [15]. Należy poczekać aż system potwierdzi, że sprzęt jest gotowy do pracy.

Do poprawnej pracy programu przykładowego wymagana jest podstawowa (standardowa) konfiguracja przełączników płytki drukowanej zestawu [12]:

- Założone zwory JP1 („3V3"), JP3 („5V") i JP2 („GND"). Oznacza to zasilanie układu procesorowego Piccolo F28027 z gniazdka USB.
- Przełącznik S1 („Boot") skonfigurowany następująco: S1.1 – do góry (ON), S1.2 – do góry, S1.3 – do góry. W praktyce oznacza to bootowanie układu procesorowego Piccolo F28027 z pamięci Flash.
- Przełącznik S4 („Serial") skonfigurowany w pozycji do góry (ON). Oznacza to dołączenie portu UART układu procesorowego Piccolo F28027 do układu emulatora, a tym samym do wirtualnego portu COM na komputerze PC.

Zestaw ewaluacyjny jest dostarczany z wpisanym do pamięci Flash układu procesorowego Piccolo F28027 programem przykładowym *Example_F2802xLaunchPadDemo*. Program automatycznie zaczyna pracować po dołączeniu zestawu do portu USB [12].

Uruchamianie środowiska CCSv5

Po uruchomienia środowiska CCSv5 pokazywane jest okno edycyjne *Workspace Launcher* ustawiania lokalizacji foldera roboczego.

1. W oknie *Workspace* należy wpisać ścieżkę dla lokalizacji folderu (*workspace*) roboczego projektu. Można ją też wskazać przy użyciu standardowego przycisku *Browse* systemu Windows. Odznaczenie (wyłączenie) opcji *Use this as the default and do not ask again* oznacza pracę z osobnym folderem roboczym. Folder z projektem można umieścić w folderze roboczym. Ale nie odwrotnie. Przy ponownym uruchomieniu środowiska CCSv5 pokazywana jest w oknie *Workspace Launcher* ścieżka lokalizacji folderu roboczego używana przy ostatnim zamknięciu CCSv5.

W oknie *Workspace* wpisz ścieżkę i nazwę foldera roboczego. Powinna być ona krótka i musi być zlokalizowana na dysku w miejscu, dla którego są uprawnienia dostępu (zapisu). Dla indywidualnej pracy proponowana jest ścieżka **<C:/*home_dir*>**. Dla tego ćwiczenia proponowana jest nazwa foldera ***/work_SYS***. Można umieścić folder *home_dir* na innym wolumenie dyskowym z prawami dostępu.

Po kliknięciu na przycisk *OK* okna *Workspace Launcher* otwierane jest okno startowe środowiska CCSv5 (i ładowane są poszczególne elementy środowiska). Można to obserwować na pasku postępu w prawym dolnym rogu okna.

Przy uruchamianiu środowiska sprawdzana jest w sieci dostępność aktualizacji. Środowisko CCSv5 przy pierwszym uruchamianiu może pobierać sporo aktualizacji. Może to trwać dosyć długo i należy koniecznie poczekać przed rozpoczęciem dalszej pracy na zakończenie inicjalizacji środowiska i pokazanie okna *Welcome* lub *Home*. Jeśli zostały wykryte i pobrane z sieci nowe lub aktualniejsze komponenty to wyświetlane jest okno wyboru komponentów do aktualizacji. Po kliknięciu przycisku *Finish* wyświetlane jest okno informacyjne. Zainstalowanie nowych komponentów wymaga zamknięcia i ponownego uruchomienia środowiska CCSv5.

Projekty przykładowe pakietu controlSUITE

W oknie *TI Resource Explorer* perspektywy *CCS Edit* pokazywana jest strona *Welcome* (w html). Zawiera ona graficznie menu główne.

Istotne informacje są zgrupowane na stronie *Home*. Można ją otworzyć po kliknięciu w oknie *TI Resource Explorer* na ikonkę *Home*.

Po kliknięciu na odnośnik *Examples* pokazywane jest po lewej stronie okna drzewo dokumentacji i dostępnych projektów przykładowych.

Jeśli pokazywana jest tylko jedna linia controlSUITE z gałęzią *English* to udostępnia ona tylko dokumen-

tację pakietu. Aby dodać dostęp do przykładowych projektów należy na dole strony *Home* kliknąć na odnośnik *Configure Resource Explorer*. W oknie dialogowym *Package Configuration* trzeba kliknąć na *Add*. Następnie trzeba wskazać folder *C:\ti\controlSUITE* i kliknąć *OK*. Nazwa controlSUITE pojawi się w oknie wyboru. Należy kliknąć *OK*. Po dłuższej chwili pojawi się w drzewie okna *TI Resource Explorer* druga linia controlSUITE zawierająca pozycje: development kits, device_support oraz libs.

Zastosowanie projektu Example_F2802xGpioSetup

2. Dla pracy z rodziną układów procesorowych Piccolo F2802x rozwiń w oknie *TI Resource Explorer* drugą pozycję *controlSUITE*. Następnie rozwiń w tym oknie drzewo *controlSUITE* → *device_support* → *f2802x* → *v210* → *f2802x_examples*. Potem kliknij na nazwę wybranego projektu ***Example_F2802xGpioSetup***.

W prawym oknie zostanie wyświetlona instrukcja jak krok po kroku zbudować i uruchomić projekt.

Krok1: Importowanie projektu Example_F2802xGpioSetup do CCSv5

Krok1 umożliwia zaimportowanie wybranego projektu do CCSv5.

3. W oknie *TI Resource Explorer* kliknij na odnośnik kroku 1.

Po poprawnym wykonaniu importowania w oknie *Project Explorer* pojawia się drzewo projektu i w oknie *TI Resource Explorer* pokazywany jest zielony znaczek ✔ na prawo od linii nazwy kroku.

Projekt *Example_F2802xGpioSetup* został zaimportowany z kopiowaniem projektu i pliku ***Example_2802xGpioSetup.c*** do foldera roboczego projektu.

Krok2: Budowanie projektu Example_F2802xGpioSetup

Krok2 umożliwia wykonanie budowania wybranego projektu.

4. W oknie *TI Resource Explorer* kliknij na odnośnik kroku 2.

W oknie *Console* pokazywane są bieżące informacje o postępie budowania. W oknie *Problems* pokazywane są opisy błędów, ostrzeżeń i informacji. Po poprawnym wykonaniu budowania pokazywany jest w oknie *TI Resource Explorer* zielony znaczek ✔ na prawo od linii nazwy kroku.

Kliknięcie na odnośnik kroku 2 powoduje automatyczne budowanie projektu – podobnie jak po przyciśnięciu przycisku *Build*. Powinno to spowodować zapisanie wszystkich plików ze zmianami przed rozpoczęciem budowania projektu.

5. W oknie *Project Explorer* rozwiń drzewo projektu i kliknij na jego nazwę. Został zbudowany projekt w konfiguracji budowania o nazwie RAM.

Budowanie projektu ***Example_F2802xGpioSetup*** zostało zakończone poprawnie. Został utworzony wynikowy plik binarny Example_2802xGpioSetup.out (zobacz okno *Console*). Zostały jednak zgłoszone cztery ostrzeżenia (zobacz okno *Problems*). Na razie są one nieistotne.

Krok3: Definiowanie konfiguracji sprzętowego systemu docelowego

Krok3 umożliwia zdefiniowanie konfiguracji sprzętowej systemu docelowego dla projektu. Na początku pole *Connection* pokazuje typ „none”.

6. W oknie *TI Resource Explorer* kliknij na odnośnik kroku 3.

W oknie dialogowym *Debugger Configuration* rozwiń listę wyboru.

7. Wybierz pozycję ***Texas Instruments XDS100v2 USB Emulator***. Kliknij *OK*.

W oknie *TI Resource Explorer* pole *Connection* pokazuje teraz typ *Texas Instruments XDS100v2 USB Emulator*. Zielony znaczek ✔ pokazywany jest na prawo od linii nazwy kroku.

Utworzony plik konfiguracji sprzętowej TMS320F28027.ccxml jest teraz pokazany w gałęzi *targetConfigs* drzewa projektu w oknie *Project Explorer*. Jest on ustawiony jako Active/Default (aktywny i domyślny).

Krok4: Uruchamianie sesji debugowej dla projektu Example_F2802xGpioSetup

Krok4 umożliwia uruchomienie sesji debugowej dla projektu. Dotychczas praca środowiska CCSv5 nie wymagała fizycznej obecności sprzętu docelowego. Wykonanie kroku 4 wymaga wcześniejszego dołączenia zestawu ewaluacyjnego *C2000 Piccolo LaunchPad* do komputera z zainstalowanym środowiskiem CCSv5 [12].

8. W oknie *TI Resource Explorer* kliknij na odnośnik kroku 4.

Kliknięcie na odnośnik kroku 4 powoduje automatyczne rozpoczęcie sesji debugowej – podobnie jak po przyciśnięciu przycisku *Debug*.

Postęp działania środowiska CCSv5 można obserwować na pasku stanu w prawym dolnym rogu okna. Może to trwać dosyć długo i należy koniecznie poczekać przed rozpoczęciem dalszej pracy na zakończenie ładowania kodu i pokazania się okna perspektywy *CCS Debug*.

Wgląd w projekt Example_F2802xGpioSetup

9. Zauważ, że praca programu została zatrzymana na pierwszej linii kodu funkcji *main()*.
10. Otwórz okno *Disassembly* z menu *View* → *Disassembly*. W tym oknie można dokładnie zobaczyć jak naprawdę pracuje układ procesorowy Piccolo F28027.

Najpierw trzeba poprawić program projektu *Example_F2802xGpioSetup*. Aplikacja czasu rzeczywistego nie może kończyć swojego wykonania. W praktyce oznacza to iż funkcja main() musi na końcu zawierać pętlę nieskończoną. Może ona być dodana po ostatnim „użytkowym” kodzie jak jest to zrobione np. w innym projekcie przykładowym *Example_F2802xLEDBlink* pakietu controlSUITE.

11. Przed końcem kodu funkcji main() dodaj pętlę

```
for(;;){
 asm(„ NOP”);
 }
```

12. Wykonaj samo budowanie projektu (bez ponownego startowania sesji debugowej). Przełącz się na perspektywę *CCS Edit*. Kliknij na przycisk *Build*. Nie używaj przycisku *Debug*.

Tab. 1. Rejestry układu oscylatora, układu PLL, układu włączania sygnałów zegarowych modułów peryferyjnych, układu sterowania trybu niskiego poboru prądu (LPM) oraz modułu CPU Watchdog układów procesorowych serii Piccolo TMS320F2802x. [5]

Nazwa	Adres	Opis(1)	Zmienna wskazująca strukturę opisu
XCLK	0x0000 7010	Rejestr sterujący XCLKOUT/XCLKIN	myCLK
PLLSTS	0x0000 7011	Rejestr stanu PLL	myPLL
CLKCTL	0x0000 7012	Rejestr sterujący zegara	
PLLOCKPRD	0x0000 7013	Rejestr okresu stabilizacji sygnału PLL	
INTOSC1TRIM	0x0000 7014	Rejestr trymowania wewnętrznego oscylatora 1	
INTOSC2TRIM	0x0000 7016	Rejestr trymowania wewnętrznego oscylatora 2	
LOSPCP	0x0000 701B	Rejestr preskalera zegara LSPCLK	
PCLKCR0	0x0000 701C	Rejestr sterujący 0 włączania zegarów modułów peryferyjnych.	
PCLKCR1	0x0000 701D	Rejestr sterujący 1 włączania zegarów modułów peryferyjnych.	
LPMCR0	0x0000 701E	Rejestr sterujący modułu LPM	
PCLKCR3	0x0000 7020	Rejestr sterujący 3 włączania zegarów modułów peryferyjnych.	
PLLCR	0x0000 7021	Rejestr sterujący PLL	
SCSR	0x0000 7022	Rejestr sterujący i stanu systemu	myWdog
WDCNTR	0x0000 7023	Rejestr zegara modułu CPU Watchdog	
WDKEY	0x0000 7025	Rejestr klucza modułu CPU Watchdog	
WDCR	0x0000 7029	Rejestr sterujący modułu CPU Watchdog	
BORCFG	0x985	Rejestr konfiguracyjny modułu BOR	

(1) Wszystkie rejestry mają ochronę EALLOW

13. Na pytanie czy załadować plik wynikowy kodu przyciśnij przycisk *Yes*. Poczekaj na załadowanie programu. Przełącz się na perspektywę *CCS Debug*.

Pliki nagłówkowe projektu

Na początku pliku *Example_2802xGpioSetup.c* włączany jest plik nagłówkowy *DSP28x_Project.h*.

#include “DSP28x_Project.h” // Device Headerfile and Examples Include Files

Plik *DSP28x_Project.h* znajduje się w ścieżce *C:\TI\controlSUITE\device_support\f2802x\v210*.

W pliku *DSP28x_Project.h* znajduje się wywołanie następnych dwóch plików nagłówkowych

#include “F2802x_Device.h”
// DSP2802x Headerfile Include File
#include “f2802x_common/include/F2802x_Examples.h” // DSP2802x Examples Include File

Główny plik definicyjny projektu *F2802x_Device.h* znajduje się w ścieżce

C:\TI\controlSUITE\device_support\f2802x\v210

W pliku jest wykonywany wybór typu układu procesorowego (DSP28027PT) oraz definiowane są symbole i typy stosowane w całym projekcie.

Drugi plik nagłówkowy *DSP2802x_Examples.h* znajduje się w ścieżce

C:\TI\controlSUITE\device_support\f2802x\v210\f2802x_common\include

Plik zawiera na początku definicje symboli używanych do inicjacji rejestrów PLLCR i DIVSEL modułu PLL układu procesorowego serii Piccolo F2802x. Te ustawienia określają częstotliwość zegara systemowego układu procesorowego.

Plik określa też parametr CPU_RATE = 16.667L będący wartością okresu zegara systemowego wyrażoną w ns dla częstotliwości SYSCLKOUT = 60 MHz.

W pliku definiowany jest wskaźnik *Device_cal* na funkcję kalibracji oscylatorów wewnętrznych i modułu ADC. Na koniec definiowana jest funkcja opóźnienia *DELAY_US(A)* wyskalowana w mikrosekundach.

List.1
```
#include “f2802x_common/include/clk.h”
#include “f2802x_common/include/flash.h”
#include “f2802x_common/include/gpio.h”
#include “f2802x_common/include/pie.h”
#include “f2802x_common/include/pll.h”
#include “f2802x_common/include/wdog.h”
```

Dalej w pliku *Example_2802xGpioSetup.c* włączane są pliki nagłówkowe obsługi modułów peryferyjnych z zastosowaniem modelu drajwerów programowych (bilioteka *driverlib*) użyte w projekcie (list.1).

Pliki znajdują się w ścieżce *C:\TI\controlSUITE\device_support\f2802x\v210\f2802x_common\include*. Każdemu plikowi nagłówkowemu odpowiada plik kodu *.c o tej samej nazwie.

Moduł CPU Watchdog

Układy procesorowe serii Piccolo F2802x posiadają moduł CPU Watchdog [5, 14]. Moduł CPU Watchdog posiada 8-bitowy licznik zliczający do góry z użyciem sygnału zegarowego OSCCLK. Gdy licznik osiągnie wartość maksymalną (wystąpi przepełnienie) moduł generuje wyjściowy impuls - poziom niski sygnałów /WDSRT oraz /XRS o długości 512 cykli zegara OSCCLK. Jeśli jest włączone zgłaszanie przerwania od modułu CPU Watchdog to zamiast tego generowany jest tylko sygnał /WDINT. Moduł CPU Watchdog może także wyprowadzać procesor z trybu obniżonego poboru mocy.

Dodatkowo procesory serii Piccolo F2802x posiadają moduł NMI Watchdog. Należy zwrócić uwagę, że moduł NMI Watchdog jest innym modułem niż CPU Watchdog.

Moduł CPU Watchdog jest bardzo istotny w aplikacjach czasu rzeczywistego. Ma on dosyć prostą budowę. Toteż metody jego obsługi udostępniane przez bibliotekę driverlib są dosyć łatwe do opisania.

Deklaracje dla obsługi modułu CPU Watchdog znajdują się w pliku nagłówkowym *wdog.h* w ścieżce

C:\TI\controlSUITE\device_support\f2802x\v210\f2802x_common\include

Funkcje biblioteczne obsługi modułu CPU Watchdog znajdują się z pliku źródłowym *wdog.c* w ścieżce

C:\TI\controlSUITE\device_support\f2802x\v210\f2802x_common\source

14. W pliku *Example_F2802xGpioSetup.c* odszukaj linię (62) kodu *WDOG_Handle myWDog;* Przyciśnij klawisz *Ctrl*, najedź kursorem na nazwę typu *WDOG_Handle* i kliknij na nią lewym klawiszem myszy.

Wskazanie na strukturę *WDOG_Obj* (jako nowy typ *WDOG_Handle*) jest zdefiniowane w pliku *wdog.h*

typedef struct WDOG_Obj * WDOG_Handle;

List.2
```
//! \brief Defines the watchdog (WDOG) object
typedef struct _WDOG_Obj_
{
    volatile uint16_t   SCSR;      //!< System Control & Status Register
    volatile uint16_t   WDCNTR;    //!< Watchdog Counter Register
    volatile uint16_t   rsvd_1;    //!< Reserved
    volatile uint16_t   WDKEY;     //!< Watchdog Reset Key Register
    volatile uint16_t   rsvd_2[3]; //!< Reserved
    volatile uint16_t   WDCR;      //!< Watchdog Control Register
} WDOG_Obj;
```

List.3
```
WDOG_Handle WDOG_init(void *pMemory, const size_t numBytes)
{
    WDOG_Handle wdogHandle;
    if(numBytes < sizeof(WDOG_Obj))
    return((WDOG_Handle)NULL);
    // assign the handle
    wdogHandle = (WDOG_Handle)pMemory;
    return(wdogHandle);
} // end of WDOG_init() function
```

Struktura *WDOG_Obj* opisu modułu CPU Watchdog jest zdefiniowana w pliku wdog.*h* . Jest to zdefiniowanie nowego typu (list.2). Struktura obejmuje wszystkie rejestry sterujące modułu CPU Watchdog układów procesorowych serii Piccolo F2802x (patrz tab.1). Pomiędzy rejestrami modułu znajdują się obszary nieistotne deklarowane jako *rsvd_x*.

15. W oknie edycyjnym wróć do pliku głównego *Example_F2802xGpioSetup.c*.

Dalej w na początku funkcji main() jest definiowana zmienna *myWdog* typu *WDOG _Handle* określająca główny wskazanie (handle) na strukturę opisu dla modułu CPU Watchdog

WDOG_Handle myWDog;

16. Przyciśnij klawisz *Ctrl* i najedź kursorem na nazwę funkcji *WDOG_init*. Gdy nazwa jest podkreślona kliknij na nią lewym klawiszem myszy. Zostanie otworzony plik z deklaracją tej funkcji z krótkim opisem praktycznie identycznym z opisem w dokumentacji.

Funkcja *WDOG_init* realizuje inicjowanie zmiennej *myWDog*. Deklaracja funkcji *WDOG_init* jest zamieszczona w pliku *wdog.h*

Wywołanie funkcji *WDOG_init* jest zamieszczone na początku programu w linii

myWDog = WDOG_init((void *)WDOG_BASE_ADDR, sizeof(WDOG_Obj));

Wywołanie funkcji *WDOG_init* jest zrealizowane z argumentami będącymi zdefiniowanymi dwoma wartościami. Adres w przestrzeni danych określający początek obszaru adresowego rejestrów sterujących i danych modułu CPU Watchdog jest zamieszczony w pliku *wdog.h*

#define WDOG_BASE_ADDR (0x00007022)

Druga wartość podaje liczbę bajtów struktury opisu modułu CPU Watchdog. Uwaga, dla procesorów serii Piccolo F2802x bajt ma rozmiar słowa, czyli 16 bitów.

17. W oknie edycyjnym wróć do pliku głównego *Example_F2802xGpioSetup.c*.
18. Zauważ przypadkową zawartość zmiennych pokazywanych w oknie *Variables*.
19. Przejdź do linii 71 kodu w pliku głównym z wywołaniem funkcji *WDOG_init*. Kliknij kilka razy na przycisk pracy krokowej *Step Over* na pasku narzędziowym okna *Debug*.
20. Kliknij na przycisk pracy krokowej *Step Into* na pasku narzędziowym okna *Debug*.

Wyświetlany jest komunikat. Problem jest spowodowany wygenerowaniem biblioteki *driverlib* w lokalizacji innej niż standardowa ścieżka pakietu controlSUITE. Można doraźnie zaradzić problemom dostępu.

21. Kliknij na przycisk *Locate File*. Wskaż ścieżkę

C:\ti\controlSUITE\device_support\f2802x\v210\f2802x_common\source

Definicja funkcji *WDOG_init* jest zamieszczona w pliku *wdog.c* (list.3).

Po wywołaniu funkcji *WDOG_init* można w oknie *Variables* zobaczyć aktualne wartości argumentów tego wywołania (**rysunek 1**).

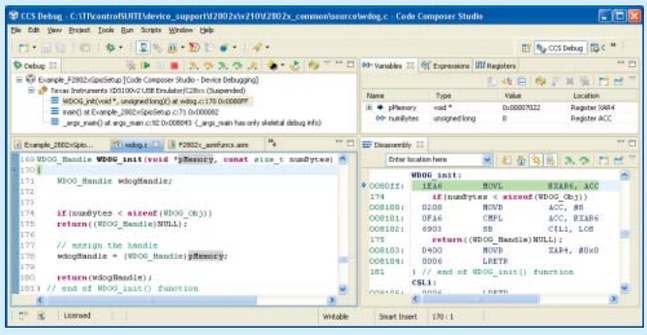
22. Kliknij kilka razy na przycisk pracy krokowej *Step Over* aż wykonanie powróci do pliku głównego.

Zmienna *myWDog* w rezultacie wywołania funkcji *WDOG_init* przyjmuje wartość adresową 0x00007022 (**rysunek 2**, porównaj z tab. 1).

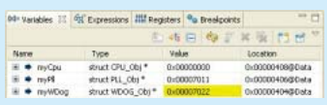
Teraz można używać funkcji obsługi modułu CPU Watchdog z biblioteki *driverlib*. Każda funkcja obsługi modułu CPU Watchdog wymaga podania wskazania na strukturę opisu modułu CPU Watchdog jako pierwszy argument wywołania. Następne argumenty zależą od realizowanej funkcjonalności.

Linia (74) wywołania funkcji *WDOG_disable* w programie głównym projektu wygląda następująco

WDOG_disable(myWDog);



Rysunek 1. Stan perspektywy *CCS Debug* po wywołaniu funkcji WDOG_init



Rysunek 2. Ustawienie wartości adresowej zmiennej myWDog po wywołaniu funkcji WDOG_init

23. Zobacz deklarację funkcji WDOG_disable. Przyciśnij klawisz *Ctrl* i najedź kursorem na nazwę funkcji *WDOG_disable*. Gdy nazwa jest podkreślona kliknij na nią lewym klawiszem myszy.

Deklaracja funkcji *WDOG_disable* jest zamieszczona w pliku *wdog.h*

24. Przejdź do linii (74) kodu w pliku głównym z wywołaniem funkcji *WDOG_disable*. Kliknij kilka razy na przycisk pracy krokowej *Step Over* na pasku narzędziowym okna *Debug*.
25. Kliknij na przycisk pracy krokowej *Step Into* na pasku narzędziowym okna *Debug*.

Definicja funkcji *WDOG_disable* jest zamieszczona w pliku *wdog.c* (list.4).

Wykonanie funkcji *WDOG_disable* powoduje wpisanie jedynki do bitu WDDIS(WDCR[6]) czyli zablokowanie działania modułu CPU Watchdog. Można to zobaczyć w oknie *Registers* po rozwinięciu zestawu rejestrów SYSCTRL. Opis rejestrów jest zamieszczony w tab.1.

26. Kliknij kilka razy na przycisk pracy krokowej *Step Over* aż wykonanie powróci do pliku głównego.

Inicjowanie układu generacji zegara systemowego

Układy procesorowe serii Piccolo F2802x posiadają dwa wewnętrzne oscylatory (INTOSC1 i INTOSC2), wewnętrzny oscylator dla zewnętrznego kwarcu oraz moduł PLL [5, 14]. Blok dostarcza sygnały zegarowe dla rdzenia układu procesorowego oraz sterowanie dla modułu LPM (obniżonej mocy), modułu CPU Watchdog, NMI Watchdog i CPU Timer2 (**rysunek 3**). Oscylatory wewnętrzne INTOSC1 i INTOSC2 nie wymagają elementów zewnętrznych.

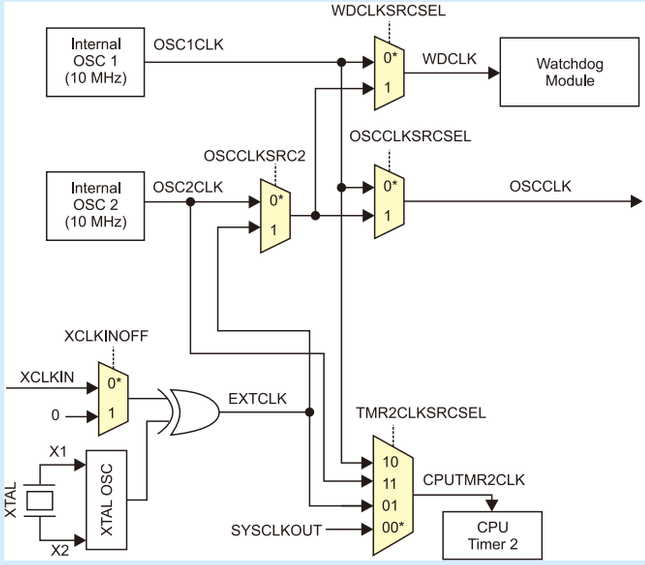
Są cztery sposoby generowania sygnału zegarowego OSCCLK dla układów procesorowych serii Piccolo F2802x [14]:

1. Oscylator wewnętrzny INTOSC1 (10 MHz): Pozwala na dostarczenie zegara dla modułu CPU Watchdog, rdzenia i licznika CPU TIMER2.
2. Oscylator wewnętrzny INTOSC2 (10 MHz): Pozwala na dostarczenie zegara dla modułu CPU Watchdog, rdzenia i licznika CPU TIMER2.
3. Oscylator dla zewnętrznego rezonatora kwarcowego: Układ wewnętrznego oscylatora układów procesorowych serii Piccolo F2802x umożliwia generację sygnału zegarowego OSCCLK z zastosowaniem zewnętrznego rezonatora kwarcowego. Jest on dołączany do wyprowadzenia X1 oraz X2 układu procesorowego. Wyprowadzenia X1 i X2 są dostępne tylko w większej obudowie.
4. Zewnętrzny sygnał zegarowy dołączony do wejścia XCLKIN. Wejście XCLKIN jest multipleksowane pomiędzy wyprowadzeniami GPIO19 lub GPIO38 układu procesorowego.

Deklaracje dla obsługi układu generacji zegara systemowego znajdują się w pliku nagłówkowym *clk.h* w ścieżce *C:\TI\controlSUITE\device_support\f2802x\v210\f2802x_common\include*

Funkcje biblioteczne obsługi układu generacji zegara systemowego znajdują się z pliku źródłowym *clk.c* w ścieżce *C:\TI\controlSUITE\device_support\f2802x\v210\f2802x_common\source*

List.4
```
void WDOG_disable(WDOG_Handle wdogHandle)
{
    WDOG_Obj *wdog = (WDOG_Obj *)wdogHandle;
    uint16_t regValue = wdog->WDCR;
    // set the bits
    regValue |= WDOG_WDCR_WDDIS_BITS;
    ENABLE_PROTECTED_REGISTER_WRITE_MODE;
    // store the result
    wdog->WDCR = regValue | WDOG_WDCR_WRITE_ENABLE;
    DISABLE_PROTECTED_REGISTER_WRITE_MODE;
    return;
} // end of WDOG_disable() function
```



Rysunek 3. Układ generowania sygnału zegarowego dla układów procesorowych serii Piccolo F2802x [11]

Struktura *CLK_Obj* opisu układu generacji zegara systemowego obejmuje wszystkie rejestry sterujące tego modułu układów procesorowych serii Piccolo F2802x (patrz tab.1).

Wskazanie na strukturę *CLK_Obj* (jako nowy typ) jest zdefiniowane w pliku *clk.h*

typedef struct CLK_Obj *CLK_Handle;

Następnie jest definiowana zmienna *myCLK* typu *CLK_Handle* określająca główny wskazanie (handle) na strukturę dla układu generacji zegara systemowego

CPU_Handle myCpu;

Funkcja *CLK_init* realizuje inicjowanie zmiennej *myCLK*. Deklaracja funkcji *CLK_init* jest zamieszczona w pliku *clk.h*. Każda funkcja obsługi układu generacji zegara systemowego wymaga podania wskazania na strukturę opisu *CLK_Obj* jako pierwszy argument wywołania. Następne argumenty zależą od realizowanej funkcjonalności.

Linia wywołania funkcji *CLK_setOscSrc* w programie głównym projektu wygląda następująco

//Select the internal oscillator 1 as the clock source

CLK_setOscSrc(myClk, CLK_OscSrc_Internal);

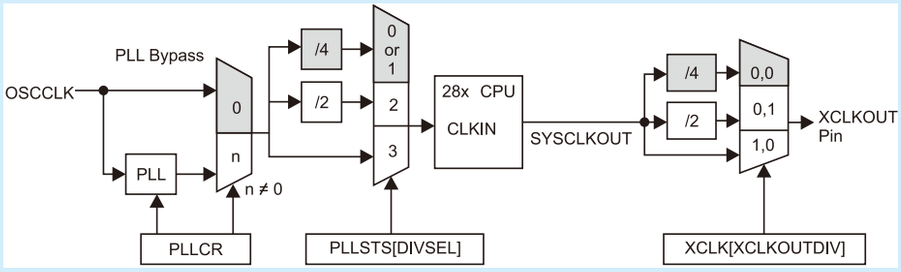
Definicja wartości wyliczanych dla drugiego argumentu jest zdefiniowana w pliku *clk.h* (list. 5).

Wykonanie funkcji *CLK_setOscSrc* z agrumentem *CLK_OscSrc_Internal* powoduje wybranie oscylatora

List.5
```
//! \brief Enumeration to define the clock oscillator source
typedef enum
{
 CLK_OscSrc_Internal=(0 << 0), //!< Denotes an internal oscillator source
 CLK_OscSrc_External=(1 << 0)  //!< Denotes an external oscillator source
} CLK_OscSrc_e;
```

wewnętrznego INTOSC1 (10MHz) jako źródło sygnału zegarowego OSCCLK.

Układ generacji sygnałów zegarowych, po włączeniu zasilania układu procesorowego serii Piccolo F2802x lub wykonaniu sprzętowej operacji RESET, jest domyślnie konfigurowany następująco (**rysunek 4**):



Rysunek 4. Generowanie systemowego sygnału zegarowego SYSCLKOUT oraz sygnału XCLKOUT układów procesorowych serii Piccolo F2802x [5]. Zacienione pola wskazują ustawienia domyślne

- Sygnał na wejście PLL jest podawany z generatora wewnętrznego INTOSC1 (10 MHz)
- Układ PLL jest wyłączony
- Dzielnik układu PLL jest ustawiony na dział przez 4 co oznacza, że systemowy sygnał zegarowy SYSCLKOUT jest ustawiony na 2.5 MHz
- Dzielnik podziału sygnału wyjściowego XCLKOUT jest ustawiony na dział przez 4.
- Sygnał wyjściowy XCLKOUT nie jest podawany na wyprowadzenie układu procesorowego

Linia wywołania funkcji *CLK_setOscSrc* w programie nic nie zmienia, bo i tak domyślnie jest wybrany oscylator wewnętrzny INTOSC1.

Linia wywołania funkcji *PLL_setup* w programie głównym projektu wygląda następująco

```
// Setup the PLL for x12 /2 which will yield 60Mhz = 10Mhz * 12 / 2
PLL_setup(myPll, PLL_Multiplier_12, PLL_DivideSelect_ClkIn_by_2);
```

Funkcja *PLL_setup* realizuje inicjowanie modułu PLL. Deklaracja funkcji *PLL_setup* jest zamieszczona w pliku pll.*h* (list.6). Wykonanie funkcji *PLL_setup* powoduje ustawienie mnożnika modułu PLL na 12 i podzielnika na 2. W rezultacie sygnał zegarowy OSCCLK=10 MHz daje systemowy sygnał zegarowy SYSCLKOUT=60 MHz.

27. Skonfiguruj wyprowadzenie GPIO18 jako sygnał wyjściowy XCLKOUT.

    Za linią kodu CLK_disableAdcClock(myClk);

    wstaw nową linię kodu

    GPIO_setMode(myGpio, GPIO_Number_18, GPIO_18_Mode_XCLKOUT);

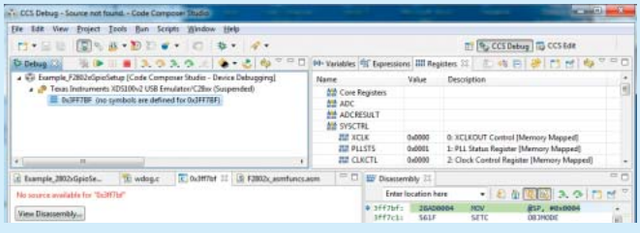
28. Zmień wartość podzielnika dla sygnału wyjściowego XCLKOUT na podział przez jeden.

    Wstaw kolejną nową linię kodu

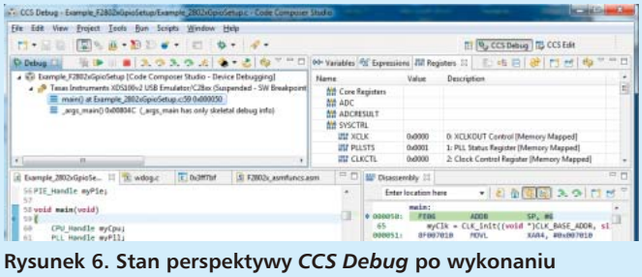
    CLK_setClkOutPreScaler(myClk, CLK_ClkOutPreScaler_SysClkOut_by_1);

29. Wykonaj samo budowanie projektu (bez ponownego startowanie sesji debugowej). Przełącz się na perspektywę CCS Edit. Kliknij na przycisk *Build*. Nie używaj przycisku *Debug*.

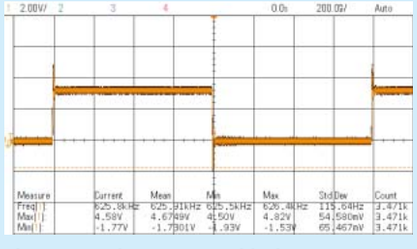
30. Na pytanie czy załadować plik wynikowy kodu przyciśnij przycisk *Yes*. Poczekaj na załadowanie programu. Przełącz się na perspektywę CCS Debug.
31. Dołącz sondę oscyloskopu do sygnału XCLKOUT (złącze J1.7 zestawu ewaluacyjny *C2000 Piccolo LaunchPad*).
32. Kliknij na przycisk *Reset CPU* na pasku narzędziowym okna *Debug*.
33. Przyciśnij przycisk RESET (S2) na płytce zestawu *C2000 Piccolo LaunchPad*.



Rysunek 5. Stan perspektywy *CCS Debug* po wykonaniu sprzętowej operacji RESET



Rysunek 6. Stan perspektywy *CCS Debug* po wykonaniu operacji Restart poprzedzonej sprzętową operacją RESET



Rysunek 7. Sygnał XCLKOUT układu procesorowego F28027 Piccolo po uruchomieniu programu poprzedzonego sprzętową operacją RESET – praca z ustawieniami domyślnymi rejestrów serujących modułu PLL oraz podzielnika sygnału XCLKOUT.

List.6

```
//! \brief     Sets the phase lock loop (PLL) divider and multiplier
//! \param[in] pllHandle  The phase lock loop (PLL) object handle
//! \param[in] clkMult    The clock multiplier value
//! \param[in] divSelect  The divide select value
void PLL_setup(PLL_Handle pllHandle, const PLL_Multiplier_e clkMult, const PLL_DivideSelect_e divSelect);
```

34. Kliknij na przycisk *Reset CPU* na pasku narzędziowym okna *Debug*.

W ten sposób została wykonana sprzętowa operacja RESET układu procesorowego F28027 Piccolo bez utraty programu wpisanego do pamięci RAM i bez wychodzenia ze środowiska CCSv5 (**rysunek 5**).

35. Wykonaj ponowne uruchomienie programu. Kliknij na przycisk *Restart*. Praca programu zostanie zatrzymana na pierwszej linii kodu funkcji *main()* (**rysunek 6**).
36. Wykonaj program do linii z wywołaniem funkcji *CLK_setClkOutPreScaler*. Zaznacz tą linię, kliknij na nią lewym klawiszem myszy.
37. Kliknij prawym klawiszem na zaznaczoną linię (poza tekstem kodu) i wybierz pozycję *Run to Line*.

Program zostanie uruchomiony i zatrzymany na zaznaczonej linii kodu.

38. Zobacz na oscyloskopie przebieg sygnału XCLKOUT.

Układ generacji zegara systemowego i moduł PLL pracuje z ustawieniami domyślnymi (x12, /4). Podzielnik sygnału wyjściowego XCLKOUT ma domyślne ustawienie podziału przez cztery (/4) Dlatego sygnał wyjściowy ma częstotliwość XCLKOUT=10MHz/4/4= 625kHz (**rysunek 7**).

39. Wykonaj zaznaczoną linię kodu. Kliknij na przycisk pracy krokowej *Step Over* na pasku narzędziowym okna *Debug*. Zobacz na oscyloskopie przebieg sygnału XCLKOUT (**rysunek 8**).

Zmiana podzielnika sygnału wyjściowego na jeden daje sygnał XCLKOUT=10MHz/4= 2.5MHz

40. Wykonaj dwie następne linie kodu (razem z wywołaniem funkcji *PLL_setup*). Kliknij dwa razy na przycisk pracy krokowej *Step Over* na pasku narzędziowym okna *Debug*.
41. Zobacz w oknie *Registers* zmianę ustawienia rejestru sterującego PLLSTS modułu PLL (**rysunek 9**).
42. Zobacz na oscyloskopie przebieg sygnału XCLKOUT (**rysunek 10**).

Wykonanie ustawienia modułu PLL na generowanie systemowego sygnału zegarowego SYSCLKOUT= 10MHz * 12/2 = 60MHz daje sygnał wyjściowy XCLKOUT=60MHz.

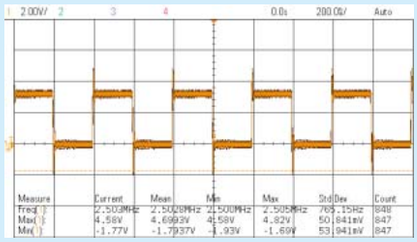
43. Wykonaj ponowne uruchomienie programu. Kliknij na przycisk pracy krokowej *Restart* na pasku narzędziowym okna *Debug*. Praca programu zostanie zatrzymana na pierwszej linii kodu funkcji *main()*.
44. Zobacz w oknie *Registers* brak zmiany zawartości rejestrów PLLSTS i CLKCTL modułu PLL. Nie ma też zmiany zawartości rejestru XCLK ustawienia podzielnika sygnału XCLKOUT.
45. Zobacz na oscyloskopie przebieg sygnału XCLKOUT. Sygnał wyjściowy XCLKOUT dalej ma 60MHz.

Po wykonaniu ponownego uruchomienia programu przez debuger układ procesorowy F28027 Piccolo rozpoczął pracę z poprzednio ustawioną częstotliwością systemowego sygnału zegarowego. W tym przypadku jest SYSCLKOUT = 10MHz * 12/2 = 60MHz oraz sygnał wyjściowy XCLKOUT=60MHz.

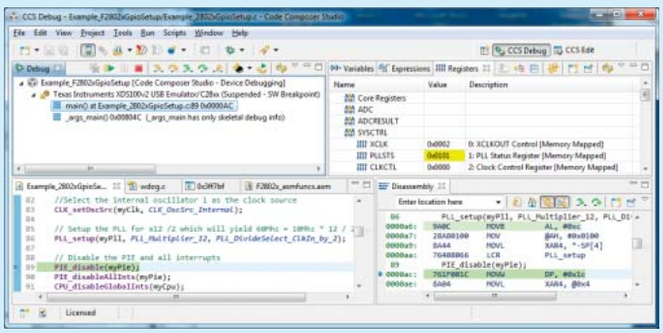
46. Kliknij na przycisk *Reset CPU* na pasku narzędziowym okna *Debug*.
47. W ten sposób została wykonana debugowa operacja RESET układu procesorowego F28027 Piccolo bez utraty programu wpisanego do pamięci RAM.

Zobacz w oknie *Registers* zmianę zawartości rejestru XCLK ustawienia podzielnika sygnału XCLKOUT (rys. 5). Została ustawiona wartość domyślna. Zobacz w oknie *Registers* brak zmiany zawartości rejestrów PLLSTS i CLKCTL modułu PLL.

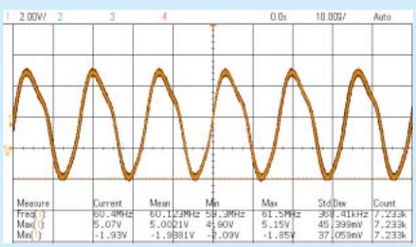
48. Wykonaj ponowne uruchomienie programu. Kliknij na przycisk pracy krokowej *Restart* na pasku narzędziowym okna *Debug*. Praca programu zostanie zatrzymana na pierwszej linii kodu funkcji *main()*.



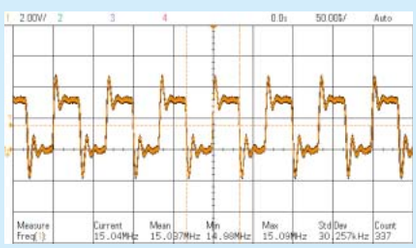
Rysunek 8. Sygnał XCLKOUT układu procesorowego F28027 Piccolo po uruchomieniu z ustawieniami domyślnymi rejestrów serujących modułu PLL oraz ze zmienionym podzielnikiem sygnału wyjściowego XCLKOUT na jeden



Rysunek 9. Stan perspektywy *CCS Debug* po skonfigurowaniu modułu PLL na pracę z maksymalną częstotliwością systemowego sygnału zegarowego. XCLKOUT = SYSCLKOUT



Rysunek 10. Sygnał XCLKOUT układu procesorowego F28027 Piccolo po skonfigurowaniu modułu PLL na pracę z maksymalną częstotliwością systemowego sygnału zegarowego oraz z podzielnikiem sygnału wyjściowego XCLKOUT ustawionym na jeden.



Rysunek 11. Sygnał XCLKOUT układu procesorowego F28027 Piccolo po wykonaniu debugowej operacji RESET. Podzielnik sygnału wyjściowego XCLKOUT jest ustawiony na domyślny podział przez cztery.

49. Ponownie wykonaj program do linii z wywołaniem funkcji *CLK_setClkOutPreScaler*. Zaznacz tą linię, kliknij na nią lewym klawiszem myszy.
50. Kliknij prawym klawiszem na zaznaczoną linię (poza tekstem kodu) i wybierz pozycję *Run to Line*.

Program zostanie uruchomiony i zatrzymany na zaznaczonej linii kodu.

51. Zobacz na oscyloskopie przebieg sygnału XCLKOUT (**rysunek 11**). Sprawdź częstotliwość sygnału.

Po wykonaniu debugowej operacji RESET układ procesorowy F28027 Piccolo rozpoczął pracę z poprzednio ustawioną częstotliwością systemowego sygnału zegarowego. W tym przypadku jest SYSCLKOUT = 10MHz * 12/2 = 60MHz oraz sygnał wyjściowy XCLKOUT=60MHz. Jednak zmienił się podzielnik sygnału XCLKOUT w rejestrze XCLK na podział przez cztery. Dlatego XCLKOUT = 60MHz/4=15MHz.

Wartość domyślna jest wpisywana do rejestrów sterujących modułu PLL (PLLCR i PLLSTS) tylko podczas operacji RESET układu procesorowego spowodowanej przez zewnętrzny sygnał /XRS lub moduł CPU Watchdog.

Operacja RESET układu procesorowego wymuszona przez emulator sprzętowy (przez port JTAG) lub moduł wykrywania zaniku sygnału zegarowego (NMI Watchdog) nie powoduje zmiany zawartości rejestrów sterujących modułu PLL. Ale ustawia wartość domyślną innych rejestrów sterujących, np. modułu GPIO oraz podzielnika dla sygnału XCLKOUT.

Zawartość rejestru XCLK z polem bitowym podzielnika dla sygnału XCLKOUT oraz ustawienia portów GPIO nie jest zmieniania podczas ponownego uruchamiania programu (polecenie *Restart*).

Kalibracja oscylatorów wewnętrznych INTOSC1 i INTOSC2 oraz modułu ADC

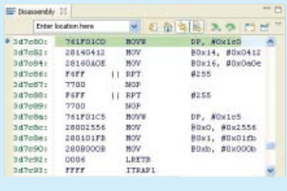
Procedura *Device_cal* jest umieszczona w pamięci Boot ROM układów procesorowych serii F2802x [7, 14].

Procedura *Device_cal()* wykonuje kalibrację oscylatorów wewnętrznych INTOSC1 i INTOSC2 oraz modułu ADC. Wartości do kalibracji są pobierane z pamięci OTP.

Kalibracja jest wykonywana automatycznie podczas normalnego wykonania procedury bootowania i nie jest potrzebna żadna akcja wykonywana przez użytkownika. Jeśli jednak zostanie pominięta inicjalizacja wykonywana z pamięci Boot ROM (np. w środowisku CCS) to kalibracja musi być wykonana przez program użytkownika. Błąd inicjalizacji układu procesorowego powoduje, że oscylatory i moduł ADC będą pracować z parametrami poza specyfikacją.

Programowe wywołanie procedury *Device_cal()* w kodzie użytkownika jest wykonywane w kilku krokach:

- Utwórz wskaźnik do funkcji *Device_cal()*. Jest to realizowane poprzez deklarację procedury.



Rys.12 Kod procedury *Device_cal* kalibracji oscylatorów wewnętrznych INTOSC1 i INTOSC2 oraz modułu ADC.

- Włącz zegar modułu ADC (w rejestrze PCLKR0)
- Wywołaj funkcję wskazywaną przez *Device_cal*.
- Wyłącz zegar modułu ADC (w rejestrze PCLKR0)

Funkcje *CLK_enableAdcClock(myClk)* oraz *CLK_disableAdcClock(myClk)* z biblioteki *driverlib* stosowane są do włączania i wyłączania sygnału zegarowego modułu peryferyjnego ADC.

Wywołanie procedury *Device_cal* w programie głównym projektu wygląda następująco

```
CLK_enableAdcClock(myClk);
(*Device_cal)();
CLK_disableAdcClock(myClk);
```

Deklaracja procedury *Device_cal* jest zamieszczona w pliku *F2802x_Examples.h* w ścieżce

C:\TI\controlSUITE\device_support\f2802x\v210\f2802x_common\include

```
// The following pointer to a function call calibrates the ADC and internal oscillators
#define Device_cal (void   (*)(void))0x3D7C80
```

Procedura *Device_cal* jest zrealizowana w języku asemblerowym rodziny procesorów C2000 (rys.12).

Obsługa przerwań i modułu PIE

CPU procesorów rodziny TMS320C2000 obsługuje tylko 16 przerwań maskowalnych z przypisanymi priorytetami sprzętowymi INT1-INT14, DLOGINT oraz RTOSINT. Procesor posiada wiele modułów peryferyjnych dlatego CPU nie może bezpośrednio obsłużyć wszystkich tych przerwań. Do tego służy specjalny moduł - sterownik przerwań układów peryferyjnych PIE (Peripheral Interrupt Expansion).

Procesory rodziny TMS320C2000 posiadają moduł PIE, który multipleksuje przerwania generowane przez wiele modułów peryferyjnych i dołącza je do niewielu przerwań rdzenia CPU procesora. Dokładny opis działania modułu PIE jest zamieszczony w [5, 14].

W dalszej części programu wykonywana jest podstawowa inicjalizacja układu przerwań i modułu PIE:

- Wywołanie funkcji *PIE_disable(myPie)* powoduje wyłączenie modułu PIE – sterownika przerwań układów peryferyjnych.
- Wywołanie funkcji *PIE_disableAllInts(myPie);* powoduje zablokowanie obsługi przerwań należących do wszystkich 12-tu grup przerwań peryferalnych.
- Wywołanie funkcji *CPU_disableGlobalInts(myCpu);* pwoduje globalne wyłączenie obsługi przerwań procesora.
- Wywołanie funkcji *CPU_clearIntFlags(myCpu);* powoduje wyzerowanie wszystkich pozycji znacznika przerwań CPU.
- Wywołanie funkcji *PIE_setDebugIntVectorTable(myPie);* powoduje wypełnienie wektorów przerwań w tablicy wektorów przerwań PIE adresami domyślnych procedur obsługi.
- Wywołanie funkcji *PIE_enable(myPie);* powoduje włączenie modułu PIE.

Aby użyć przerwań trzeba jeszcze włączyć globalną obsługę przerwań. Dalej znajduje się również fragment kodu dla konfiguracji budowania projektu z użyciem pamięci Flash. Ustawiana jest wartość zmiennej _FLASH i kod sekcji pamięci przepisywany jest z pamięci Flash do pamięci RAM.

Uruchamianie przykładowych programów pakietu programowego controlSUITEv3 umożliwia pozna-

nie sposobów programowania układów procesorowych Piccolo F2802x. Przedstawione postępowanie pokazuje typowy sposób działania projektu dla większości instalacji środowiska programowego. Jednak mogą występować różnice zachowania się środowiska dla instalacji na różnych komputerach.

Henryk A. Kowalski
kowalski@ii.pw.edu.pl

Bibliografia

[1] Code Composer Studio, strona produktu http://www.ti.com/ccs
[2] controlSUITE Getting Started Guide (Rev. B), SPRUGU2B , 09 June 2011
[3] TMS320F28027, TMS320F28026, TMS320F28023, TMS320F28022, TMS320-F28021, TMS320F280200, Piccolo Microcontrollers, Data Sheet, SPRS523I, 31 Jul 2012
[4] TMS320F28027, TMS320F28026, TMS320F28023, TMS320F28022, TMS320-F28021, TMS320F280200, Piccolo MCU, Silicon Errata, SPRZ292J, 31 Jan 2012
[5] TMS320x2802x Piccolo System Control and Interrupts, SPRUFN3D, 13 Feb 20013
[6] TMS320x2802x, 2803x Piccolo Analog-to-Digital Converter (ADC) and Comparator, SPRUGE5F, 31 Dec 2011
[7] TMS320x2802x Piccolo Boot ROM Reference Guide (Rev. A),), SPRUFN6A, 28 Oct 2009

kod źródłowy w ramach pakietu controlSUITE w ścieżce C:\TI\controlSUITE\libs\utilities\boot_rom\2802x

[8] F2802x Firmware Development Package USER'S GUIDE v. 210 [f2802x-FRM-EX-UG.pdf], pakiet controlSUITE
[9] F2802x Peripheral Driver Library USER'S GUIDE v. 210 [f2802x-DRL-UG.pdf], pakiet controlSUITE
[10] LAUNCHXL-F28027 C2000 Piccolo LaunchPad Experimenter Kit, User's Guide, SPRUHH2, 25 Jul 2012
[11] C2000 Piccolo Workshop, Workshop Guide and Lab Manual, December 2010, Texas Instruments
[12] Henryk A. Kowalski,"Zestaw ewaluacyjny C2000 Piccolo LaunchPad", Elektronika Praktyczna 01/2013
[13] Henryk A. Kowalski„C2000 Piccolo LanuchPad (1) – Pierwszy program w środowisku programowym CCSv5", Elektronika Praktyczna 02/2013
[14] Henryk A. Kowalski, Procesory DSP dla praktyków, BTC, Warszawa, 2011 http://ii.pw.edu.pl/kowalski/dsp/book/
[15] Henryk A. Kowalski, Procesory DSP w przykładach, BTC, Warszawa, 2012 http://ii.pw.edu.pl/kowalski/dsp/book/