C2000 Piccolo LaunchPad (2)

Łatwe programowanie z pakietem controlSUITE

Do tworzenia w środowisku CCSv5 programów przeznaczonych dla procesorów z rodziny C2000 firmy Texas Instruments jest potrzebny pakiet programowy controlSUITE tej firmy. Zawiera on firmware, biblioteki, opisy zestawów sprzętowych oraz projekty przykładowe dla wszystkich serii procesorów rodziny C2000. Dla układów procesorowych serii Piccolo F2802x przeznaczona jest biblioteka driverlib.

Na początku artykułu jest opisany pakiet programowy controlSUITEv3.1.2, a następnie ćwiczenie praktyczne z zastosowaniem biblioteki *driverlib* z tego pakietu oraz środowiska *Code Composer Studio* v5 i zestawu ewaluacyjnego *C2000 Piccolo LaunchPad*. Zastosowano przykładowy projekt *Example_F2802xAdc_TempSensorConv* z pakietu programowego controlSUITE. Ćwiczenie jest zorganizowane w taki sposób, że działania są wykonywane w kolejnych punktach i krokach uzupełnionych o szczegółowe opisy.

Instalowanie i użytkowanie środowiska CCSv5.3.0 [1] oraz pakietu programowego controlSUITEv3.1.2 [2] zostało opisane w artykule [12]. Dokładny opis zestawu ewaluacyjnego *C2000 Piccolo LaunchPad* został zamieszczony w artykule [11]. Dokładny opis układów procesorowych serii Piccolo F2802x został zamieszczony w książce [13]. Ich dane techniczne i parametry elektryczne są zamieszczone w dokumentach [3, 4, 5, 6].

Dodatkowe informacje:
ftp://ep.com.pl, user: 63048, pass: 632vmey5
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

Model programowy procesora serii Piccolo F2802x

Pakiet programowy „firmware" (F2802x Firmware Development Package) dostarcza wsparcia dla dwóch modeli programowania układów procesorowych serii Piccolo F2802x:

- **Model bezpośredniego dostępu do rejestrów (header files)**. Rejestry każdego modułu peryferyjnego są zdefiniowane w odpowiednim pliku nagłówkowym (*.h). Te definicje określają jednoznacznie położenie każdego rejestru oraz każdego bitu i pola bitowego w każdym rejestrze. Pliki nagłówkowe tylko definiują te struktury. Deklarowanie tych struktur jest udostępniane w pliku *F2802x_GlobalVariableDefs.c*. Aby zastosować model plik ten musi być dołączony do projektu. Trzeba też dołączyć plik definicyjny konfiguracji pamięci dla rejestrów modułów peryferyjnych *F2802x_Headers_nonBIOS.cmd*. Używanie modelu bezpośredniego dostępu do rejestrów jest bardzo proste. Uzyskany kod jest optymalny czasowo i krótki. Lecz poprawne zastosowanie tego modelu wymaga dokładnej wiedzy o działaniu modułu, pól bitowych każdego rejestru, interakcji pomiędzy nimi i sekwencji ich użycia.
- **Model drajwerów programowych (library)**. Biblioteka dostarcza API do sterowania modułami peryferyjnymi. Drajwery programowe zapewniają kontrolę nad modułami i nie wymagają bezpośredniego dostępu do ich rejestrów. Kod programu uzyskany z zastosowaniem modelu drajwerów programowych może nie być optymalny czasowo i może nie być najkrótszy. Jednak kod może być znacząco bardziej czytelny i może umożliwić szybkie tworzenie oprogramowania bez szczegółowej znajomości detali konstrukcyjnych modułów peryferyjnych. Model może nie udostępniać wszystkich możliwości funkcjonalnych modułu peryferyjnego.

Każdy z tych modeli może być zastosowany osobno lub łącznie. Opis jest zamieszczony w dokumentach [8, 9] dostępnych w ścieżce *\doc* pakietu programowego „firmware".

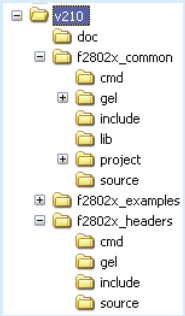
Budowa pakietu controlSUITEv3.x

Pakiet programowy controlSUITE jest zbiorem oprogramowania i narzędzi programowych dla układów procesorowych rodziny TMS320C2000 [2]. Zawiera również szczegółową dokumentację techniczną modułów sprzętowych. Pakiet controlSUITE jest instalowany w domyślnej ścieżce *C:\ti\controlSUITE* i zawiera główne elementy:

- Pakiety wsparcia tworzenia oprogramowania w ścieżce *\device_support*.
- Materiały dla zestawów sprzętowych w ścieżce *\development_kits*.
- Biblioteki w ścieżce *\libs*.

Każda seria układów procesorowych rodziny TMS320C2000 ma własny pakiet programowy wsparcia tworzenia oprogramowania. Są to pakiety o nazwie *Header Files and Examples* lub *Firmware Development Package*. Struktura pakietów wszystkich serii jest taka sama. Dla serii Piccolo F2802x, najnowsza wersja pakietu znajduje się w ścieżce *C:\ti\controlSUITE\device_support\f2802x\v210* i zawiera (**rysunek 1**):

- Opis modeli programowania procesora zamieszczony w dokumentach [8, 9] dostępnych w ścieżce *\doc*.



Rysunek 1. Struktura folderów pakietu „firmware" F2802x pakietu controlSUITEv3

- Pliki modelu bezpośredniego dostępu do rejestrów w ścieżce \ *f2802x_headers*. W folderze *cmd* znajdują się pliki definicyjne konfiguracji pamięci dla modułów peryferyjnych układu procesorowego. W folderze *source* znajduje się plik *F2802x_GlobalVariableDefs.c* z definicjami zmiennych tego modelu.
- Pliki modelu drajwerów programowych oraz pliki wspólne dla przykładowych projektów w ścieżce *f2802x_common*. W folderze *cmd* znajdują się pliki definicyjne konfiguracji pamięci dla różnych układów procesorowych oraz dla pracy z pamięci RAM i Flash. W folderze *source* znajdują się pliki kodu źródłowego. W folderze *include* wszystkie pliki nagłówkowe. W folderze *lib* znajdują się bilioteki: matematyczna – IQmath, obsługi modułu HRPWM oraz biblioteka drajwerów programowych *drverlib*. W folderze *project* znajduje się projekt CCS dla biblioteki *drverlib*. Plik wynikowy budowania tego projektu jest wpisywany do foldera *lib*.
- Pliki *DSP28x_Project.h* oraz *F2802x_Device.h* na poziomie głównego foldera wersji.

Pełny schemat płytki zestawu ewaluacyjnego C2000 Piccolo LaunchPad, rysunki ścieżek oraz rysunki rozłożenia elementów jest dostępny w standardowej ścieżce *C:\ti\controlSUITE\development_kits\C2000_LaunchPad\LAUNCHXL-F28027\HwDevPkg*.

Konfiguracja sprzętowa i programowa

Do wykonania ćwiczenia potrzebny jest komputer z zainstalowanym oprogramowaniem:

- Środowisko *Code Composer Studio* v5.3.0 firmy Texas Instruments [1, 12, 14]. Umożliwia tworzenie w środowisku CCSv5 programów przeznaczonych dla procesorów serii Piccolo TMS320F2802x.
- Pakiet programowy controlSUITE v3.1.2 firmy Texas Instruments [2, 12, 14]. Zawiera oprogramowanie „firmware", biblioteki, opisy zestawów sprzętowych oraz projekty przykładowe dla wszystkich serii procesorów rodziny C2000.

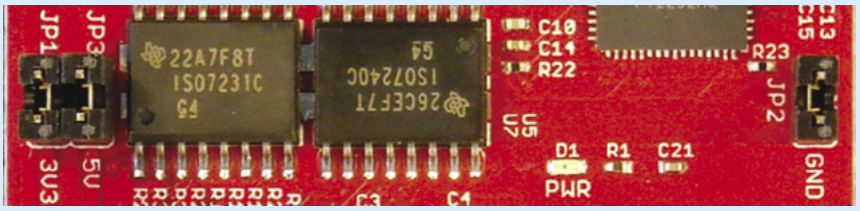
Platforma sprzętowa wymaga tylko jednego elementu:

- Zestaw ewaluacyjny *C2000 Piccolo LaunchPad* firmy Texas Instruments z układem procesorowym Piccolo TMS320F28027 firmy Texas Instruments (zawiera kabel USB-A USB-mini) [10, 11]

W folderze *C:\home_dir* komputera zostanie utworzony nowy folder *workEx2*. Wymagane są prawa dostępu (zapisu i modyfikacji) dla tej ścieżki dyskowej. Możliwe jest umieszczenie foldera *home_dir* na innym wolumenie dyskowym z prawami dostępu.

Podłączenie i skonfigurowanie zestawu C2000 Piccolo LaunchPad

Po zainstalowaniu środowiska CCSv5 [1, 12] można pierwszy raz dołączyć zestaw ewaluacyjny *C2000 Piccolo LaunchPad* [10, 11, 12] kablem USB do wolnego portu USB komputera. System Windows automatycznie rozpoznaje układ. Zostaną zainstalowane sterowniki systemu Windows dla emulatora XDS100v2 [14]. Należy poczekać aż system potwierdzi, że sprzęt jest gotowy do pracy.



Fotografia 2. Zwory JP1, JP2 i JP3 konfigurowania zasilania układu procesorowego Piccolo F28027

Do poprawnej pracy programu przykładowego wymagana jest podstawowa (standardowa) konfiguracja przełączników płytki drukowanej zestawu [11]:

- Zwory JP1, JP2 i JP3 konfigurowania zasilania układu procesorowego Piccolo F28027. Założone zwory JP1 („3V3"), JP2 („5V") i JP3 („GND") pokazano na **fotografii 2**. Oznacza to zasilanie układu procesorowego Piccolo F28027 z gniazdka USB.
- Przełącznik S1 konfigurowania trybu bootowania układu procesorowego Piccolo F28027. Przełącznik S1 („Boot") skonfigurowany następująco: S1.1 – do góry (ON), S1.2 – do góry, S1.3 – do góry (**fotografia 3**). W praktyce oznacza to bootowanie układu procesorowego Piccolo F28027 z pamięci Flash.
- Przełącznik S4 służy do konfigurowania dołączenie portu UART układu procesorowego Piccolo F28027 do układu emulatora XDS100v2. 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 (**fotografia 4**).

Fotografia 3. Przełącznik S1 konfigurowania trybu bootowania

Fotografia 4. Przełącznik S4 konfigurowania dołączenia portu UART

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 [11].

Uruchomienie środowiska CCSv5

1. W oknie *Workspace* wpisz ścieżkę i nazwę folderu roboczego.

Powinna być ona krótka i musi być zlokalizowana w miejscu, dla którego są uprawnienia dostępu (zapisu). Dla indywidualnej pracy proponowana jest ścieżka **<C:/*home_dir*>**. Dla tego ćwiczenia proponowana jest nazwa folderu ***workEx2***. 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. Może to trwać dosyć długo i należy koniecznie poczekać na zakończenie inicjalizacji środowiska przed rozpoczęciem dalszej pracy.

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

Inicjalizowanie i konfigurowanie modułu ADC procesora serii Piccolo F2802x

Opis modułu ADC układów procesorowych serii Piccolo F2802x jest zamieszczony w dokumencie [6]. Dokładny opis budowy modułu ADC jest zamieszczony w książce Henryk A. Kowalski „Procesory DSP dla praktyków" [13]. Dokładny opis inicjalizacji i zastosowania modułu ADC jest zamieszczony w książce Henryk A. Kowalski „Procesory DSP w przykładach" [14]. Opis modelu programowania procesora jest zamieszczony w dokumentach [8, 9] dostępnych w pakiecie controlSUITE w ścieżce *\f2802x\v210\doc*.

Inicjalizacja modułu ADC układów procesorowych serii Piccolo F2802x rozpoczyna się od włączenia sygnału zegarowego dla modułu [5].

```
CLK_enableAdcClock(myClk);
```

Następnie wykonywana jest kalibracja modułu ADC oraz oscylatorów wewnętrznych z użyciem danych wpisanych do układu scalonego w procesie produkcyjnym.

```
(*Device_cal)();
```

Na koniec inicjalizacji modułu wykonywana jest sekwencja programowego włączania zasilania poszczególnych układów wewnętrznych modułu ADC:

- włączane jest zasilanie diody „bandgap" - ADC_enableBandGap(myAdc);
- włączane jest zasilanie układu napięcia odniesienia - ADC_enableRefBuffers(myAdc);
- włączane jest zasilanie układu analogowego modułu ADC - ADC_powerUp(myAdc);
- włączany jest cały układ ADC - ADC_enable(myAdc);
- wybierane jest wewnętrzne źródło napięcia odniesienia do pracy ADC - ADC_setVoltRefSrc(myAdc, ADC_VoltageRefSrc_Int);

W rejestrze ADCCTL1 modułu ADC zostały ustawione trzy bity i jeden bit został wyzerowany.

Dopiero teraz można przystąpić do konfigurowania rejestrów modułu ADC.

Wewnętrzny czujnik temperatury jest wybierany jako źródło sygnału dla kanału analogowego A5. Sygnał z wyprowadzenia ADCINA5 jest odłączany od kanału multipleksera.

```
ADC_enableTempSensor(myAdc);
```

Dla układu SOC0 jest wybierany do przetwarzania sygnał z kanału analogowego A5.

```
ADC_setSocChanNumber (myAdc, ADC_SocNumber_0, ADC_SocChanNumber_A5);
//Set SOC0 channel select to ADCINA5
```

Dla układu SOC1 jest wybierany do przetwarzania sygnał z kanału analogowego A5.

```
ADC_setSocChanNumber (myAdc, ADC_SocNumber_1, ADC_SocChanNumber_A5);
//Set SOC1 channel select to ADCINA5
```

Dla układu SOC0 ustawiany jest czas próbkowania (okno czasowe) na 7 cykli zegara systemowego.

```
ADC_setSocSampleWindow(myAdc, ADC_SocNumber_0, ADC_SocSampleWindow_7_cycles);
//Set SOC0 acquisition period to 7 ADCCLK
```

Dla układu SOC1 ustawiany jest czas próbkowania(okno czasowe) na 7 cykli zegara systemowego.

```
ADC_setSocSampleWindow(myAdc, ADC_SocNumber_1, ADC_SocSampleWindow_7_cycles);
//Set SOC1 acquisition period to 7 ADCCLK
```

Ustawiany jest wybór sygnału EOC1 (koniec przetwarzania dla SOC1) jako źródła generowania przerwania ADCINT1.

```
ADC_setIntSrc(myAdc, ADC_IntNumber_1, ADC_IntSrc_EOC1);
//Connect ADCINT1 to EOC1
```

Włączane jest zezwolenie na zgłaszanie przerwań przez moduł ADC

```
ADC_enableInt(myAdc, ADC_IntNumber_1);
//Enable ADCINT1
```

Zmiany w rejestrach sterujących wykonywane są przez funkcje biblioteki *driverlib* tylko dla wybranych bitów. Pozostałe bity pozostają bez zmian. W powyższej inicjalizacji modułu zostało w projekcie przykładowym przyjęte założenie, że rejestry modułu ADC mają zawartość domyślną, taką jak po wykonaniu operacji RESET dla modułu. Dzięki temu domyślnie wybieranych jest kilka istotnych opcji działania modułu A/D:

- Wybór źródła wyzwalania dla SOC0, bity ADCSOC0CTL(TRIGSEL[15:11])=0x00 – Wyzwalanie tylko przez program
- Wybór źródła wyzwalania dla SOC1, bity ADCSOC1CTL(TRIGSEL[15:11])=0x00 – Wyzwalanie tylko przez program
- Sterowanie generowania przerwania INT, bit IMPULSEPOS(ADCCTL1[2])=0x0 - Sygnał EOC (koniec przetwarzania) jest zgłaszany zgodnie z impulsem rozpoczęcia próbkowania na początku przetwarzania

Ostatnia opcja jest błędnie ustawiona. Lepsza jest opcja zgłaszania sygnału EOC tuż przed zakończeniem operacji przetwarzania. Przy obecnych ustawieniach, zgłoszenia sygnału EOC może być (błędnie) wykonane przed wpisaniem rezultatu przetwarzania do rejestru wyniku. Przypadkowo (dla tego projektu), rezultat działania programu jest poprawny.

Niestety śledzenie wykonania kodu obsługi ADC w tym projekcie musi być wykonywane na poziomie asemblerowym, co przekracza ramy tego artykułu.

Lepsze jest ustawienie:

- IMPULSEPOS(ADCCTL1[2])=0x1 - Sygnał EOC jest zgłaszany 1 cykl zegara tuż przed wpisaniem rezultatu przetwarzania do rejestru wyniku,

Niestety, biblioteka *driverlib* nie udostępnia funkcji umożliwiającej ustawienie bitu IMPULSEPOS. Można to zrobić z wykorzystaniem struktur dostarczanych przez model bezpośredniego dostępu do rejestrów.

Założenie, że rejestry modułu ADC mają zawartość domyślną jest prawdziwe gdy projekt z obsługą modułu ADC jest wpisywany do pamięci układu procesorowego i uruchamiany jako pierwszy po włączeniu zasilania układu procesorowego. Jeśli wcześniej był już na tym układzie procesorowym uruchomiony inny program używający modułu ADC to stan rejestrów tego modułu może być dowolny.

Dlatego lepiej przed rozpoczęciem programowania modułu wymusić programowo wykonanie operacji RESET tego modułu. Funkcja *ADC_reset(myADC)* służy w bibliotece *driverlib* do wymuszania operacji RESET. Ustawia ona bit RESET w rejestrze sterującym ADCCTL1 co powoduje, że wszystkie bity rejestrów oraz stany maszyny stanów są inicjowane do stanu takiego samego jak po wykonaniu sprzętowej operacji RESET układu procesorowego. Bit ten daje jednorazowy efekt, czyli po wykonaniu inicjalizacji jest on automatycznie zerowany. Wymagane jest odczekanie minimum dwóch cykli zegara systemowego przed wpisaniem nowej zawartości do rejestru ADCCTL1.

W trakcie uruchamiania programu można sobie z opisanym problemem poradzić bez modyfikowania kodu. Należy po załadowaniu kodu do pamięci układu procesorowego kliknąć na ikonę *Reset CPU*. Spowoduje to wykonanie emulacyjnej operacji Reset - trochę różnej od operacji sprzętowej (dokładniejszy opis w książce [13]). Następnie należy kliknąć na ikonę *Restart*. Program ponownie zostanie uruchomiony do pierwszej instrukcji funkcji *main()*. Jednak tym razem rejestry modułów peryferyjnych zawierają już domyślną zawartość.

na stronie *Home*. Można ją otworzyć po kliknięciu 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 dokumentację pakietu. Aby dodać przykłady należy na stronie *Home* kliknąć na odnośnik *Configure Resource Explorer*. W oknie dialogowym *Package Configuration* trzeba kliknąć na *Add*, wskazać folder C:\ti\controlSUITE i kliknąć *OK*. Nazwa controlSUITE pojawi się w oknie wyboru i 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_ F2802xAdc_TempSensorConv

2. Dla pracy z rodziną Piccolo F2802x rozwiń w oknie *TI Resource Explorer* drugą pozycję *controlSUITE*. Następnie należy rozwinąć drzewo *controlSUITE* → *device_support* → *f2802x* → *v210* → *f2802x_examples*. Potem trzeba kliknąć na nazwę wybranego projektu ***Example_F2802xAdc_TempSensorConv***. W prawym oknie zostanie wyświetlona instrukcja jak krok po kroku zbudować i uruchomić projekt.

Krok1: Importowanie projektu Example_F2802xAdc_ TempSensorConv do CCS

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 a w oknie *TI Resource Explorer* pokazywany jest zielony znaczek ✓ na prawo od linii nazwy kroku.

Projekt *Example_F2802xAdc_TempSensorConv* został zaimportowany z kopiowaniem projektu i pliku ***Example_2802xAdc_TempSensorConv.c*** do foldera roboczego projektu.

Krok2: Budowanie projektu Example_F2802xAdc_ TempSensorConv

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.

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_F2802xAdc_TempSensorConv* zostało zakończone poprawnie. Został utworzony wynikowy plik binarny *Example_F2802xAdc_TempSensorConv.out*. Jednak Zostały zgłoszone cztery ostrzeżenia (zobacz okno *Problems*). Na razie są one nieistotne.

Krok3: Definiowanie konfiguracji sprzętowego systemu docelowego

Krok3 umożliwia zdefiniowanie konfiguracji sprzętowego 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 ✓ jest pokazywany 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_ F2802xAdc_TempSensorConv

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 [11] oraz wcześniejszego utworzeniu sprzętowej konfiguracji docelowej.

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

Powoduje to automatyczne rozpoczęcie sesji debuggowej – 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*.

9. Zbliż rękę do czterech diod LED na dole płytki zestawu ewaluacyjnego *C2000 Piccolo LaunchPad*.

Można także dotknąć palcem do płytki w tych okolicach. Typowo występuje przypadkowe gaszenie i zapalanie diod. To zjawisko nie jest groźnie dla działania zestawu.

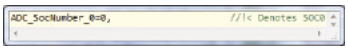
Dołączanie pliku do projektu

Skorzystanie z podglądu stanu pól bitowych rejestrów sterowania modułów peryferyjnych wymaga dołączenia do projektu pliku definicyjnego struktur modelu bezpośredniego dostępu do rejestrów. Zestawienie nazw zmiennych (struktur) udostępnianych przez plik jest podane w tab. 2.12 (str. 32) dokumentacji pakietu firmware [8]. Dla modułu ADC jest udostępniana struktura rejestrów sterowania *AdcRegs* i struktura rejestrów wyniku *AdcResult*.

10. Przełącz się do perspektywy *CCS Edit*. W oknie *Project Explorer* kliknij prawym klawiszem myszy na linię nazwy projektu *Example_F2802xAdc_TempSensorConv*. Z podręcznego menu wybierz *Add Files* oraz ścieżkę C:\ti\controlSUITE\device_support\f2802x\v210\f2802x_headers\source. Zaznacz plik *F2802x_GlobalVariableDefs.c* i kliknij na *Otwórz*. W oknie *File Operation* zaznacz opcję ***Link to files***. Kliknij *OK*. Plik zostanie dołączony (nie dodany) do projektu.
11. Wykonaj samo budowanie projektu (bez ponownego startowanie sesji debugowej). Kliknij na przycisk *Bu-*

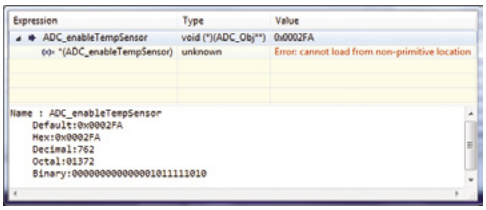
Podgląd deklaracji i definicji funkcji i zmiennych w środowisku programowym CCSv5

Po najechaniu kursorem (bez klikania) na nazwę funkcji lub zmiennej pokazywane jest okno z szczegółowymi informacjami (rysunek 5). Dla zmiennych typów wyliczanych jest pokazywane zdefiniowanie bieżącej wartości.



Rys.5. Okno informacji o zmiennej

Najechanie na typ wyliczany powoduje pokazanie jego pełnego zdefiniowania (rysunek 6). Niestety, zdefiniowanie wielu elementów biblioteki driverlib nie jest trywialne i mechanizm podpowiedzi nie w pełni sobie radzi.

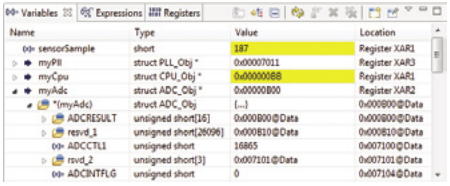


Rys.6. Okno informacji o strukturze

Jest drugi sposób sprawdzania deklaracji funkcji. Przyciśnij klawisz *Ctrl* i najedź kursorem na nazwę funkcji (lub zmiennej). Gdy nazwa jest podkreślona kliknij na nią lewym klawiszem myszy. Zostanie otworzony plik z deklaracją tej funkcji (lub zmiennej) i z krótkim opisem praktycznie identycznym z opisem w dokumentacji.

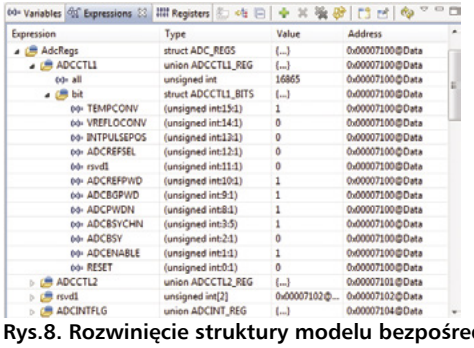
Zawartość zmiennych strukturalnych najwygodniej oglądać w oknach *Variables* i *Expressions*.

W przypadku struktur udostępnianych przez bibliotekę driverlib nie pracuje to dobrze (rysunek 7). Wskazania na struktury są typowo poprawne. Jednak rozwinięcia struktur są typowo błędnie pokazywane jako takie same. Rozwinięcia bitowe rejestrów nie uwzględniają podziału na pola bitowe i nie podają nazw z tym związanych.



Rys.7. Rozwinięcie struktury z biblioteki driverlib

Pokazywanie zawartości struktur bardzo dobrze działa w przypadku zmiennych modelu bezpośredniego dostępu do rejestrów udostępnianych po dołączeniu do projektu pliku *F2802x_GlobalVariableDefs.c*. Dodatkowo wartości pogrupowane są według nazw bitów i pól bitowych zgodnych z dokumentacją firmową (rysunek 8).



Rys.8. Rozwinięcie struktury modelu bezpośredniego

ild. Nie używaj przycisku *Debug*. Na pytanie czy załadować plik wynikowy kodu przyciśnij przycisk *Yes*.

12. Przełącz się do perspektywy *CCS Debug*. Zauważ w oknie edytora, że praca programu została zatrzymana na pierwszej linii kodu funkcji *main()*.
13. Otwórz okno *Disassembly* z menu *View* → *Disassembly*.

Wgląd w projekt Example_F2802xAdc_TempSensorConv

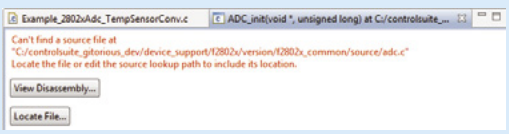
14. Zapoznaj się z komentarzem na początku pliku ***Example_2802xAdc_TempSensorConv.c***.

Krótki opis projektu przykładowego oraz założenia i wymagania sprzętowe są zamieszczone na początku głównego pliku każdego projektu przykładowego z pakietu programowego controlSUITE.

15. Dodaj zmienne *temp*, *degC* i *degK* oraz struktury *AdcRegs* i *AdcResult* do okna *Expressions*. Dwukliknij w oknie edytora na nazwę zmiennej (zaznacz). Kliknij prawym klawiszem myszy na zaznaczoną zmienną. Z podręcznego menu wybierz *Add Watch Expression* i kliknij OK. Zobacz ramkę *„Podgląd deklaracji i definicji funkcji i zmiennych”*.
16. W pliku ***Example_2802xAdc_TempSensorConv.c*** kliknij (zaznacz) na linię 57 kodu *myAdc = ADC_init()*. Kliknij prawym klawiszem na zaznaczoną linię (poza tekstem kodu) i wybierz pozycję *Run to Line*. Program zostanie uruchomiony i zatrzymany na zaznaczonej linii kodu
17. Kliknij na przycisk pracy krokowej *Step Into* na pasku narzędziowym okna *Debug*.

Wyświetlany komunikat pokazano na **rysunku 9**. Problem jest spowodowany wygenerowaniem biblioteki *driverlib* w lokalizacji innej niż standardowa ścieżka pakietu controlSUITE. Można doraźnie zaradzić problemom dostępu.

18. Kliknij na przycisk *Locate File*. Wskaż ścieżkę *C:\ti\controlSUITE\device_support\f2802x\v210\f2802x_common\source*.
19. Kliknij kilka razy na przycisk pracy krokowej *Step Over* na pasku narzędziowym okna *Debug*. Aż przejdziesz do linii 58 (następnej) kodu w pliku głównym.
20. W oknie *Variables* rozwiń strukturę *myAdc*. Zauważ, że zawiera ona pole 16-tu rejestrów wyniku modułu ADC a następnie rejestry sterujące tego modułu.
21. W oknie *Variables* rozwiń strukturę *myCpu*. Zauważ, że **błędnie** jest pokazywane rozwinięcie struktury opisu modułu ADC.
22. Stosując przycisk *Step Over* przejdź w pracy krokowej do linii 68 kodu.
23. Zobacz deklarację funkcji *CLK_enableAdcClock* (patrz ramka *„Podgląd deklaracji i definicji…”*). Zostanie otworzony plik *clk.h* z deklaracją tej funkcji i krótkim opisem (praktycznie identycznym z opisem w dokumentacji). Funkcja *CLK_enableAdcClock* włącza sygnał zegara systemowego dla modułu ADC [5, 9].
24. Przejdź do następnej linii kodu *(*Device_cal)();* i zobacz opis przy jej deklaracji. Jest to wywołanie funkcji asemblerowej wpisanej do pamięci BOOT ROM [7]. Wykonuje ona kalibrację modułu ADC oraz oscylatorów wewnętrznych z użyciem danych wpisanych do układu scalonego w procesie produkcyjnym.
25. Wykonaj program do linii 99 kodu. Zaznacz tą linię, kliknij prawym klawiszem (poza tekstem kodu) i wybierz pozycję *Run to Line*. W tej linii rozpoczyna się inicjalizacja modułu ADC.
26. Zapoznaj się kolejno ze wszystkimi liniami inicjalizacji modułu ADC. Zobacz ramkę *„Inicjalizowanie i konfigurowanie modułu ADC”*. Oglądaj opisy zamieszczone przy deklaracjach kolejnych używanych funkcji z bi-



Rysunek 9. Informacja o błędnych ścieżkach dostępu kodu źródłowego biblioteki *driverlib*

blioteki *driverlib*. W oknie *Expressions* obserwuj zmianę zawartości rejestrów sterujących modułu ADC.

27. Zapoznaj się z resztą kodu w pętli *for(;;)*.
28. Na początku pętli nieskończonej wymuszane jest programowe wyzwolenie pracy układu SOC0 oraz SOC1.

```
//Force start of conversion on SOC0 and
SOC1
ADC_forceConversion(myAdc, ADC_
SocNumber_0);
ADC_forceConversion(myAdc, ADC_
SocNumber_1);
```

Następnie jest oczekiwanie na ustawienie zgłoszenia przerwania (znacznika) w rejestrze modułu ADC. Zgłoszone przerwanie nie jest obsługiwane przez moduł obsługi przerwań peryferyjnych (PIE).

```
//Wait for end of conversion.
while(ADC_getIntStatus(myAdc, ADC_
IntNumber_1) == 0) {}
```

Zgłoszenie przerwania (znacznik) jest kasowane.

```
// Clear ADCINT1
ADC_clearIntFlag(myAdc, ADC_IntNumber_1);
```

Odczytywany jest rezultat drugiego przetwarzania, zrealizowanego przez układ SOC1. Wynik pierwszego przetwarzania jest nieprawidłowy z powodu błędnego działania modułu ADC układów procesowych serii Piccolo F2802x [4].

```
// Get temp sensor sample result from SOC1
temp = ADC_readResult(myAdc, ADC_
ResultNumber_1);
```

Odczytana wartość binarna jest konwertowana do postaci stopni Celcjusza i Kelvina.

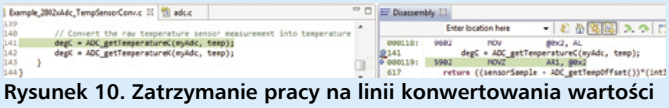
```
// Convert the raw temperature sensor
measurement into temperature
degC = ADC_getTemperatureC(myAdc, temp);
degK = ADC_getTemperatureK(myAdc, temp);
```

Uruchamianie projektu Example_F2802xAdc_TempSensorConv

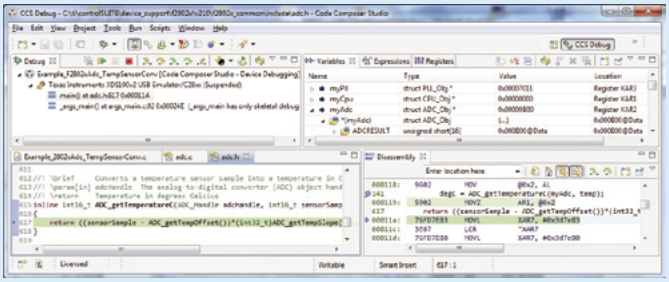
29. Ustaw pułapkę w linii 141 z kodem *degC = ADC_getTemperatureC(myAdc, temp);* Dwukliknij na linię po lewej stronie od numeru linii kodu.
30. Kliknij na przycisk *Resume* na pasku narzędziowym okna *Debug*. Program zostanie uruchomiony i zatrzymany na pułapce (**rysunek 10**).

Zauważ, że kod asemblerowy linii 141 kodu źródłowego jest umieszczony pod adresem 0x000119.

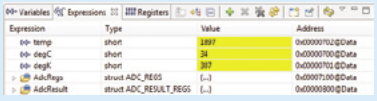
31. Kliknij na przycisk pracy krokowej *Step Into* na pasku narzędziowym okna *Debug*. Zobacz w oknie *Debug*, że pomimo pokazywania przejścia do kodu źródłowego funkcji wywołanej z funkcji *main()* dalej stos pokazuje pozostawanie wykonania w wątku *main()*. Zobacz w oknie *Disassembly*, że obecnie wykonywana instrukcja asemblerowa jest umieszczona bezpośrednio pod kolejnym adresem przestrzeni programu. Zobacz w pliku *adc.h*, że definicja funkcji *ADC_getTemperatureC* jest poprzedzona słowem kluczowym *inline*. W takiej sytuacji kompilator w miejsce wywołania funkcji wstawia rozwinięcie jej kodu (**rysunek 11**). Dokładne omówienie zagadnienia jest zamieszczone w książce [13].
32. Kliknij na przycisk *Resume*. Zobacz wartość zmiennych *temp*, *degC* i *degK* oknie *Expressions*.
33. Kliknij kilka razy na przycisk *Resume* i obserwuj wartości tych zmiennych (**rysunek 12**).



Rysunek 10. Zatrzymanie pracy na linii konwertowania wartości pomiaru



Rysunek 11. Rozwinięcie asemblerowe kodu funkcji *ADC_getTemperatureC*



Rysunek 12. Wartość zmiennych pomiaru temperatury

Zaprezentowane w artykule postępowanie pozwala na poznanie programowania z użyciem biblioteki *driverlib* w środowisku programowym CCSv5. Uruchamianie przykładowych projektów pakietu programowego controlSUITEv3 umożliwia poznanie sposobów programowania układów procesorowych Piccolo F2802x.

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, SPRUFN3C, 29 Oct 2009
[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] Henryk A. Kowalski, "Zestaw ewaluacyjny C2000 Piccolo LaunchPad", Elektronika Praktyczna 01/2013
[12] Henryk A. Kowalski, „C2000 Piccolo LanuchPad (1) – Pierwszy program w środowisku programowym CCSv5", Elektronika Praktyczna 02/2013
[13] Henryk A. Kowalski, Procesory DSP dla praktyków, BTC, Warszawa, 2011 http://ii.pw.edu.pl/kowalski/dsp/book/
[14] Henryk A. Kowalski, Procesory DSP w przykładach, BTC, Warszawa, 2012 http://ii.pw.edu.pl/kowalski/dsp/book/