C2000 Piccolo LaunchPad (3)

Łatwe programowanie do pamięci Flash

Zestaw ewaluacyjny C2000 Piccolo LaunchPad (LANUCHXL-28027) jest dostarczany z wpisanym do pamięci Flash układu procesorowego programem przykładowego projektu Example_F2802xLaunchPadDemo. Przykład ten pozwala na łatwe zapoznanie się ze sposobem organizowania projektu do pracy z pamięci Flash. Inne projekty przykładowe pakietu programowego controlSUITE dla układów procesorowych serii Piccolo F2802x mają również tak samo zdefiniowane konfiguracje budowania pozwalające na pracę z pamięci Flash.

W artykule jest opisane ćwiczenie praktyczne z zastosowaniem biblioteki *driverlib* pakietu programowego controlSUITE oraz środowiska *Code Composer Studio* v5 i zestawu ewaluacyjnego *C2000 Piccolo LaunchPad*. Zastosowano przykładowy projekt *Example_F2802xLaunchPadDemo* i opisano jego działanie. Ćwiczenie jest zorganizowane tak, 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 [10]. Opis pakietu programowego controlSUITEv3.1.2 oraz zastosowanie biblioteki *driverlib* pakietu controlSUITE zostało opisane w artykule [11]. Dokładny opis zestawu ewaluacyjnego *C2000 Piccolo LaunchPad* został zamieszczony w artykule [9]. Dokładny opis układów procesorowych serii Piccolo F2802x został zamieszczony w książce [12]. Ich dane techniczne i parametry elektryczne są zamieszczone w dokumentach [3, 4, 5]. Budowa pamięci Flash, procedura inicjalizacji pamięci i rejestry konfiguracyjne jest omówiona w książkach [12, 13].

Konfiguracja sprzętowa i programowa

Do wykonania ćwiczenia potrzebny jest komputer z zainstalowanym oprogramowaniem:

- Środowisko *Code Composer Studio* v5.3.0 firmy Texas Instruments [1, 11, 13]. Umożliwia tworzenie w środowisku CCSv5 programów przeznaczonych dla procesorów serii Piccolo TMS320F2802x.
- Pakiet programowy controlSUITE v3.1.2 firmy Texas Instruments [2, 6, 7, 11, 13]. 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) [8, 9]

W folderze *C:\home_dir* komputera zostanie utworzony nowy folder *workEx3*. Wymagane są prawa dostępu (zapisu i modyfikacji) dla tej ścieżki dyskowej. Możliwe jest umieszczenie foldera *home_dir* na innym wolumenie dyskowym z prawami dostępu.

Dodatkowe informacje:
ftp://ep.com.pl, user: 75282, pass: 852sjb64
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

Podłączenie i skonfigurowanie zestawu C2000 Piccolo LaunchPad

Po zainstalowaniu środowiska CCSv5 [1, 10] można pierwszy raz dołączyć zestaw ewaluacyjny *C2000 Piccolo LaunchPad* [8, 9] kablem USB do wolnego portu USB komputera. System Windows automatycznie rozpoznaje układ. Zostaną zainstalowane sterowniki systemu Windows dla emulatora XDS100v2 [13]. 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 [9]:

- Zwory JP1, JP2 i JP3 konfigurowania zasilania układu procesorowego Piccolo F28027. Założone zwory JP1 („3V3"), JP2 („5V") i JP3 („GND"). 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. W praktyce oznacza to bootowanie układu procesorowego Piccolo F28027 z pamięci Flash.
- Przełącznik S4 konfigurowania dołączenie portu UART układu procesorowego Piccolo F28027do 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.

Program przykładowy *Example_F2802xLaunchPadDemo* automatycznie zaczyna pracować po dołączeniu zestawu do portu USB [9] (patrz ramka).

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

083-089_piccolo(3).indd 83 2013-04-01 23:18:04

```
"C:/TI/controlSUITE/development_kits/C2000_LaunchPad/f2802x_examples/C2kLaunchPadDemo/../../f2802x_common/cmd/F28027.cmd"
, line 117: warning:
  memory range not found: FLASHD, on page 0
  .text              : >> FLASHA | FLASHC | FLASHD.      PAGE = 0
```

Rysunek 1. Opis błędu wykrytego w pliku konfiguracji pamięci

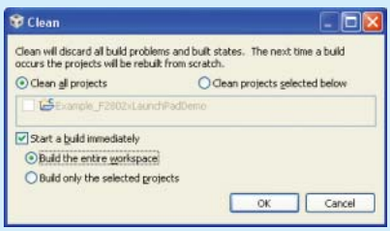
dostępu (zapisu). Dla indywidualnej pracy proponowana jest ścieżka <**C:/*home_dir***>. Dla tego ćwiczenia proponowana jest nazwa folderu ***workEx3***. Można umieścić folder *home_dir* na innym wolumenie dyskowym z prawami dostępu.

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.

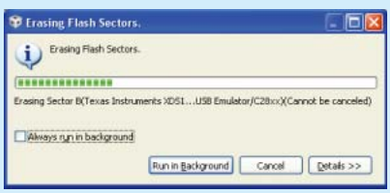
Zastosowanie projektu Example_ F2802xLaunchPadDemo

2. Dla pracy z przykładami dla zestawu ewaluacyjnego *C2000 Piccolo LaunchPad* rozwiń w oknie *TI Resource Explorer* drugą pozycję *controlSUITE*. Następnie należy rozwinąć drzewo *controlSUITE* → *development_kits* → *C2000_LaunchPad* → *f2802x_examples*. Potem trzeba kliknąć na nazwę wybranego projektu ***Example_F2802xLaunchPadDemo***. W prawym oknie zostanie wyświetlona instrukcja jak krok po kroku zbudować i uruchomić projekt.

Krok1: Importowanie projektu Example_F2802xLaunchPadDemo do CCS



Rysunek 2. Okno ustawień polecenia *Clean*



Rysunek 3. Okno informacyjne kasowania pamięci Flash

Krok1 umożliwia zaimportowanie wybranego projektu do CCSv5.

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

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

Projekt *Example_ F2802xLaunchPadDemo* został zaimportowany z kopiowaniem pliku ***Example_2802xLaunchPadDemo.c*** do foldera roboczego projektu.

Krok2: Budowanie projektu Example_F2802xLaunchPadDemo

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 ***Flash***.

Budowanie projektu *Example_F2802xLaunchPadDemo* zostało zakończone (prawie) poprawnie. Został utworzony wynikowy plik binarny *Example_F2802xLaunchPadDemo*.out.

Błędy zgłaszane przez kompilator podczas pierwszego budowania projektu.

Po zakończaniu pierwszego budowania projektu *Example_F2802xLaunchPadDemo* zgłaszane są jednak cztery ostrzeżenia . Są one pokazane w oknie *Problems*. Jeśli okno jest puste, to należy kliknąć lewym klawiszem na linię nazwy projektu w oknie *Project Explorer*. Linia zostanie pogrubiona, co oznacza wybór projektu. Dokładny opis pierwszego błędu z listy w oknie *Problems* jest pokazany w oknie *Console* (**rysunek 1**).

Przyczyną błędu jest wstawienie kropki zamiast przecinka na końcu listy bloków pamięci Flash w pliku konfiguracji pamięci *F28027.cmd* (patrz ramka).

Plik znajduje się poza folderem roboczym projektu w standardowej ścieżce *C:\TI\controlSUITE\development_kits\C2000_LaunchPad\f2802x_common\cmd*.

6. Otwórz plik *F28027.cmd* z menu *File* → *Open File*.
7. W linii 117 pliku zamień kropkę na przecinek. Zapisz zmiany do pliku. Kliknij na przycisk *Save*.
8. Ponownie wykonaj budowanie projektu. Kliknij na przycisk *Build* na pasku narzędzi perspektywy *CCS Edit*. Nie używaj przycisku *Debug*.
9. Zobacz rezultat budowania w oknie *Console*. Typowo jest tam pokazana informacja: *gmake: Nothing to be done for `all'*. Ponieważ zmiana dotyczyła zawartości pliku zewnętrznego to projekt nie ma o tym informacji. Próba wyboru innych opcji z menu *Project* daje ten sam rezultat. Jest jednak inny sposób.
10. Wybierz z menu *Project* polecenie *Clean* (**rysunek 2**). Kliknij *OK*.

Wykonanie polecenia *Clean* spowoduje usunięcie wszystkich plików pośrednich z poprzedniego budowania projektu i wykonanie nowego budowania od początku. Tym razem pozostały tylko trzy nieistotne (obecnie) ostrzeżenia.

Krok3: Definiowanie konfiguracji sprzętowego systemu docelowego

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

11. W oknie *Project Explorer* kliknij na odnośnik kroku 3.

W oknie dialogowym *Debugger Configuration* rozwiń listę i wybierz pozycję ***Texas Instruments XDS100v2 USB Emulator***. Kliknij OK. 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).

083-089_piccolo(3).indd 84 2013-04-01 23:18:05

Rysunek 4. Okno informacyjne programowania pamięci Flash

Programowanie pamięci Flash

Programowanie pamięci Flash jest wykonywane w dwóch krokach, W pierwszym kroku pamięć Flash jest kasowana (**rysunek 3**).

W drugim kroku wpisywany jest do niej kod programu z pliku *.out (**rysunek 4**).

UWAGA! Zatrzymanie programowania na kroku kasowania powoduje trwałe zablokowanie dostępu do układu procesorowego!

Krok4: Uruchamianie sesji debugowej dla projektu Example_F2802xLaunchPadDemo

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 oraz wcześniejszego utworzenia sprzętowej konfiguracji docelowej.

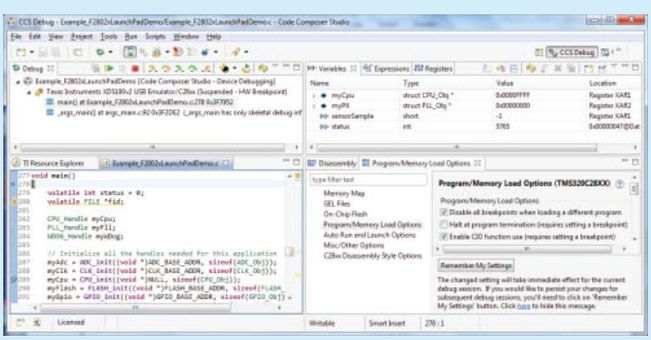
12. W oknie *TI Resource Explorer* kliknij na odnośnik kroku 4.

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

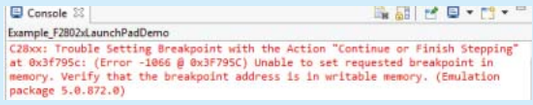
Wykonanie w perspektywie *CCS Edit* polecenia *Debug* powoduje wykonanie budowania inkrementacyjnego aktywnego projektu, uruchomienie debuggera, automatyczne dołączenie debuggera (connection) do docelowego układu procesorowego oraz załadowanie/zaprogramowanie kodu wynikowego (programu) do pamięci wewnętrznej RAM/Flash układu procesorowego. Otwierana jest również perspektywa *CCS 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*.

Załadowany programu jest uruchamiany i jego wykonywanie jest zatrzymywane na pierwszej instrukcji funkcji main().



Rysunek 5. Projekt z ustawioną pułapką



Rysunek 6. Informacja o błędzie spowodowanym brakiem wolnych pułapek sprzętowych układu procesorowego z rdzeniemC28x

Rysunek 7. Ustawienia projektu wymagające użycia pułapek sprzętowych układu procesorowego z rdzeniem C28x

Debugowanie programu wpisanego do pamięci Flash

13. Otwórz okno *Disassembly* z menu *View* → *Disassembly*.
14. Ustaw pułapkę w linii 289 z kodem *myCpu = CPU_init ();* Dwukliknij na linię po lewej stronie od numeru linii kodu (**rysunek 5**).
15. Kliknij trzykrotnie na przycisk pracy krokowej *Step Over* na pasku narzędziowym okna *Debug*. Zobacz informację w oknie *Console* (**rysunek 6**).

Wszystkie procesory rodziny TMS320C2000 zawierają dwa układy (unit) analizy AU1 i AU2. Układ analizy AU1 zlicza zdarzenia lub monitoruje szyny adresowe. Układ analizy AU2 monitoruje szyny adresowe i szyny danych. Można skonfigurować te dwa układy jako pułapki sprzętowe lub punkty podglądu (watchpoint). Dodatkowo, układ AU1 może być skonfigurowany jako 40-bitowy licznik wydajności (benchmark). Zawartość licznika jest zwiększana o jeden dla każdego cyklu zegarowego CPU. Dokładny opis zagadnienia jest zamieszczony w książce [12].

Ustawienie pułapki w linii kodu źródłowego C w pliku *Example_2802xLaunchPadDemo.c* spowodowało automatyczne zastosowanie pułapki sprzętowej, ponieważ kod jest umieszczony w pamięci Flash. Wskazuje na to ikonka w oknie edycyjnym, z podłożonym pod koło krzyżykiem.

Krokowe wykonanie kodu wymaga zastosowania pułapki sprzętowej. Jedna została wykorzystana przez aktywną pułapkę w linii kodu. Próba krokowego wykonania kodu przy aktywnej pułapce spowodowała zgłoszenie błędu w oknie Console. Zgłaszany przez CCS błąd jest spowodowany wykorzystaniem obu dostępnych pułapek sprzętowych. W jaki spsób została użyta druga pułapka sprzętowa?

16. Wybierz z menu pozycję *Tools->Debugger Options->Program/Memory Load Options* (**rysunek 7**).

Wybrana opcja obsługi CIO powoduje wykorzystanie pułapki sprzętowej (dla programu z kodem w pamięci Flash).

17. Odznacz opcję *Enable CIO function use*. Kliknij przycisk *Remember My Settings*. Zmiany ustawień dla tego projektu zostaną zapamiętane.
18. Kliknij trzy razy na przycisk pracy krokowej *Step Over*. Tym razem praca krokowa już działa.

083-089_piccolo(3).indd 85 2013-04-01 23:18:05

Rysunek 8. Informacja o błędnych ścieżkach dostępu do kodu źródłowego biblioteki *driverlib*.

19. Zdejmij pułapkę w linii 289. Kliknij dwukrotnie na linię po lewej stronie od numeru linii kodu.
20. Kliknij na przycisk pracy krokowej *Step Into* na pasku narzędziowym okna *Debug*.

Wyświetlany jest komunikat pokazany na **rysunku 8**. Problem jest spowodowany wygenerowaniem biblioteki *driverlib* w lokalizacji innej niż standardowa ścieżka pakietu controlSUITE. Można doraźnie zaradzić problemom dostępu. Lepszym rozwiązaniem może być ponowne zbudowanie biblioteki z poprawnymi ścieżkami, ale wkracza to trochę poza zakres tego artykułu.

21. Kliknij na przycisk *Locate File*. Wskaż ścieżkę C:\TI\controlSUITE\development_kits\C2000_LaunchPad\f2802x_common\source.

Dołączanie pliku do projektu

Skorzystanie z podglądu stanu pól bitowych rejestrów sterowania modułu ADC wymaga dołączenia do projektu pliku definicyjnego struktur modelu bezpośredniego dostępu do rejestrów. Dla modułu ADC udostępniana jest struktura rejestrów sterowania *AdcRegs* i struktura rejestrów wyniku *AdcResult*.

22. Sprawdź rozmiar pliku wynikowego *Example_F2802xLaunchPad_demo.out* dostępnego w ścieżce *C:\home_dir\work_Ex3\Example_F2802xLaunchPadDemo\Flash*
23. Przełącz się do perspektywy *CCS Edit*.
24. Sprawdź w pliku *Example_F2802xLaunchPad_demo.map* (ta sama ścieżka) wykorzystanie pamięci w sektorach A i C pamięci Flash. W oknie *Project Explorer* rozwiń gałąź *Flash* i dwukliknij na nazwę pliku. Jest to bardzo pożyteczny plik zawierający dokładne informacje o rozmieszczeniu sekcji wynikowych kodu i zmiennych w pamięci układu procesorowego. Zamknij podgląd pliku.
25. W oknie *Project Explorer* kliknij prawym klawiszem myszy na linię nazwy projektu *Example_F2802xLaunchPadDemo*. Z podręcznego menu wybierz *Add Files* oraz ścieżkę *C:\TI\controlSUITE\development_kits\C2000_LaunchPad\f2802x_headers\source*.
26. 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.
27. Wykonaj samo budowanie projektu (bez ponownego startowanie sesji debugowej). Kliknij na przycisk *Build*. Nie używaj przycisku *Debug*. Na pytanie czy załadować plik wynikowy kodu przyciśnij przycisk *Yes*. Poczekaj na zakończenie programowania kodu do pamięci Flash.
28. Ponownie sprawdź rozmiar pliku wynikowego *Example_F2802xLaunchPad_demo.out*.
29. Ponownie sprawdź w pliku *Example_F2802xLaunchPad_demo.map* wykorzystanie pamięci w sektorach A i C pamięci Flash.

Rozmiar kodu wynikowego po dołączeniu pliku *F2802x_GlobalVariableDefs.c* nie zwiększył się.

```
FLASHC 003f2000 00002000 0000120c
FLASHA 003f6000 00001f80 00001f80
```

Wykorzystanie pamięci w sektorach A i C przez sekcję kodu jest takie samo przed i po dołączeniu pliku.

Za to rozmiar pliku wynikowego *Example_F2802xLaunchPad_demo.out* zwiększył się ze 196KB do 241kB. Jest to spowodowane większą liczbą danych debugowych wstawionych do tego pliku.

Wgląd w projekt Example_F2802xLaunchPadDemo

30. Przełącz się do perspektywy *CCS Debug*.
31. Zapoznaj się z komentarzem na początku pliku ***Example_2802xLaunchPadDemo.c***.

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

32. Dodaj zmienne *referenceTemp* i *currentTemp* 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.

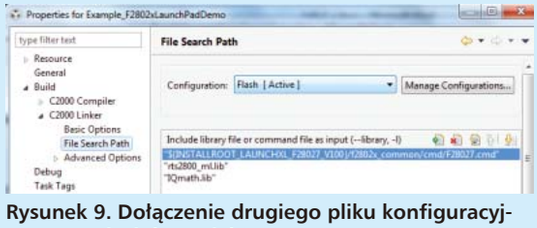
Konfigurowanie do pracy z pamięcią Flash

W projekcie *Example_F2802xLaunchPadDemo* są dołączone dwa pliki definicyjne pamięci. Jeden to plik *DSP2802x_Headers_nonBIOS.cmd*. W perspektywie *CCS Edit* jest on widoczny w drzewie projektu w oknie *Project Explorer*. W pliku są zdefiniowane struktury (rejestrów) modułów peryferyjnych układu procesorowego oraz modułu PIE. Jest on stosowany dla projektów bez użycia systemu czasu rzeczywistego (SYS/BIOS).

Podstawowe zdefiniowanie użytych w projekcie obszarów pamięci układu procesorowego jest zamieszczone w drugim pliku. Lecz jego umiejscowienie w projekcie nie jest takie proste. Drugi plik *F28027.cmd* jest wskazany w opcjach wywołania linkera.

33. W oknie *Project Explorer* w perspektywie *CCS Edit* kliknij prawym klawiszem myszy na linię nazwy projektu i z podręcznego menu wybierz pozycję *Properties*.
34. W oknie *Properties for Example_F2802xLaunchPadDemo* rozwiń gałąź *Build->C200 Linker->File Search Path* (**rysunek 9**). Pozwoli to na sprawdzenie ustawień linkera.

W polu opcji linkera "-l" jest wskazanie na plik *„${INSTALLROOT_LAUNCHXL_F28027_V100}/f2802x_common/cmd/F28027.cmd"*. Podana jest ścieżka dostępu do pliku zdefiniowana pośrednio poprzez symbol ścieżki instalacji pakietu firmware dla zestawu ewaluacyjnego *C2000 Piccolo LaunchPad*. Jest to ścieżka inna niż dla standardowego pakietu firmware. W pakiecie programowym controlSUITEv3.1.2 został dla zestawu ewalu-



Rysunek 9. Dołączenie drugiego pliku konfiguracyjnego pamięci do projektu

083-089_piccolo(3).indd 86 2013-04-01 23:18:05

acyjnego zastosowany skopiowany standardowy pakietu firmware F2802x w wersji V100 (starej).

35. W oknie *Properties for Example_F2802xLaunchPadDemo* kliknij na gałąź *Build* oraz otwórz zakładkę *Variables*. Zdefiniowane są tam symbole ścieżek (**rysunek 10**).

    To nie są jedyne predefiniowane symbole w projekcie.
36. Rozwiń gałąź *Build->C2000 Compiler->Advanced Options->Predefined Symbols*. Są tam zdefiniowane inne ważne symbole np. _Flash" oraz „_DEBUG".

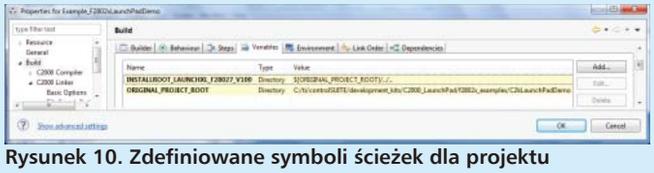
    Następne ważne symbole są zdefiniowane w pliku konfiguracji pamięci.
37. W perspektywie *CCS Edit* zobacz zawartość pliku *F28027.cmd* (patrz ramka).

Na początku pliku *F28027.cmd* zdefiniowane są dostępne obszary pamięci z przypisaniem do przestrzeni adresowej programu – Page0 lub przestrzeni adresowej danych – Page1.

Następnie w pliku *F28027.cmd* wykonywane jest przypisanie sekcji do zdefiniowanych na początku obszarów pamięci. Każda sekcja inicjalizowana musi być przypisana do pamięci nieulotnej Flash. Sekcja kodu o nazwie .text jest przypisana do pamięci Flash w sektorach A, C i D.

Sekcja *ramfuncs* jest zdefiniowana w specjalny sposób. Dla niej określony jest adres ładowania (LOAD) do pamięci Flash (adres FLASHA). Pod tym adresem będzie zaprogramowana zawartość sekcji w trakcie ładowania kodu. Zdefiniowany jest również adres wykonania (RUN) w pamięci RAM (adres PRAML0). Jest to adres, pod który zostanie przepisana zawartość sekcji z pamięci Flash do pamięci RAM. Przepisywanie jest wykonywane w programie użytkownika. Następnie z sekcją *ramfuncs* związane są zmienne (symbole) języka C z dostępem na poziomie asemblerowym. Dlatego ich nazwy są na początku uzupełnione znakiem podkreślenia. Dokładne omówienie zagadnienia jest zamieszczone w książce [13].

W pliku *Example_F2802xLaunchPad_demo.map* jest pokazane rozmieszczenie sekcji w pamięci po wykonaniu budowania projektu.



Rysunek 10. Zdefiniowane symboli ścieżek dla projektu

Przepisywanie kodu programu z pamięci Flash do pamięci RAM

Inicjalizacja ustawień rejestrów konfiguracyjnych pamięci Flash układu procesorowego serii Piccolo F2802x musi być wykonywana przez kod umieszczony w pamięci RAM.

Kod przeznaczony do wykonywania z pamięci RAM jest przypisywany przez linker do sekcji *ramfuncs* z użyciem dyrektywy *#pragma CODE_SECTION*. W pliku *flash.c* (ścieżka *f2802x_common/source/*) zastosowana jest linia przypisania

```
#pragma CODE_SECTION(FLASH_setup,
"ramfuncs");
```

W pliku *flash.c* wszystkie funkcje obsługi rejestrów konfiguracyjnych pamięci Flash są przypisane do sekcji *ramfuncs*. Do tej sekcji jest również przypisany kod z pliku *F2802x_asmfuncs.asm* projektu przykładowego.

Dla sekcji *ramfuncs* linker definiuje trzy symbole:

- **RamfuncsLoadStart** – adres ładowania (LOAD) do pamięci Flash (adres FLASHA). Pod tym adresem będzie zawartość sekcji zaprogramowana w trakcie ładowania kodu.
- **RamfuncsRunStart** – adres wykonania (RUN) w pamięci RAM (adres PRAML0). Jest to adres, pod który zostanie przepisana zawartość sekcji z pamięci Flash do pamięci RAM.
- **RamfuncsLoadSize** – rozmiar obszaru do przepisania.

Linker ustawia wartość tych symboli na podstawie pliku konfiguracji pamięci *F28027.cmd*. Symbole te są udostępniane jako zmienne w pliku *F2802x_GlobalVariableDefs.c* dołączonym do biblioteki *driverlib*.

```
extern Uint16 RamfuncsLoadStart;
extern Uint16 RamfuncsLoadSize;
extern Uint16 RamfuncsRunStart;
```

Działanie programu przykładowego Example_F2802xLaunchPadDemo

Zestaw ewaluacyjny C2000 Piccolo LaunchPad (LANUCHXL-28027) jest dostarczany z wpisanym do pamięci Flash układu procesorowego programem przykładowym *Example_F2802xLaunchPadDemo*.

Program automatycznie zaczyna pracować po dołączeniu zestawu do portu USB. Świecą kolejno cztery diody LED wskazując na prawo – na przycisk S3. Po przyciśnięciu (i przytrzymaniu) na przycisk S3 (prawy) program przechodzi do pomiaru wewnętrznej temperatury układu procesorowego.

Dodatkowo program przykładowy wysyła informacje poprzez port szeregowy układu procesorowego typu UART. Aby je wykorzystać należy najpierw zidentyfikować numer wirtualnego portu COM. W tym celu należy kliknąć prawym klawiszem myszy na *Mój komputer* (np. w menu Start). Wybrać *Właściwości* a następnie *Sprzęt* oraz *Menedżer urządzeń*. Na liście *Porty (COM i LPT)* należy znaleźć port o nazwie USB Serial Port (COMX), gdzie X jest numerem.

Program przykładowy był uruchamiany z obsługą komunikacji na PC poprzez program PuTTY (do pobrania darmowo ze strony *http://www.putty.org/*). Próby zastosowania innych programów komunikacyjnych nie dały dobrych rezultatów. Poprawna praca wymaga ustawienia parametrów komunikacji 115200 8N1. Po uruchomieniu programu PuTTY wybierz typ połączenia *Serial* a następnie wpisz poprawny numer portu COM oraz szybkość transmisji i kliknij *Open*.

Na razie w oknie programu PuTTY jest pusto. Należy teraz przycisnąć przycisk reset RST (S2, lewy). Zostanie wyświetlona spora plansza.

Zgodnie z wyświetlanym napisem przyciśnij przycisk S3 (prawy). Po naciśnięciu przycisku mierzona jest aktualna temperatura wewnętrzna układu procesorowego i zapisywana jako temperatura odniesienia Tref (w stopniach Celsjusza). Program przechodzi do trybu cyklicznego pomiaru temperatury bieżącej Tcur. Program pokazuje różnicę temperatury bieżącej i odniesienia: Tcur-Tref. Świecąca lewa dioda LED (D2), o wadze w zapisie binarnym 1000b równej wartości dziesiętnej 8, oznacza zgodność temperatury bieżącej z temp. odniesienia (Tref+0°C). Dodatnia różnica temperatur zwiększa wyświetlaną binarnie wartość a ujemna różnica ją zmniejsza. Czyli pokazywany zakres jest niesymetryczny od Tref-7 (0000b) do Tref+8 (1111b). W prawym dolnym rogu planszy wyświetlanej na PC przez program PuTTY jest pokazywany rezultat aktualnego pomiaru. Oprócz aktualnej różnicy pokazywana jest tam też aktualna temperatura w stopniach Celsjusza.

Ponowne przyciśnięcie przycisku S3 powoduje ustawienie temperatury odniesienia na wartość temperatury bieżącej.

083-089_piccolo(3).indd 87 2013-04-01 23:18:05

Rysunek 11 Ustawienia parametrów wypełniania obszaru pamięci RAM

Zawartość pamięci Flash od adresu początkowego wskazywanego przez *RamfuncsLoadStart* jest przepisywana do pamięci RAM, rozpoczynając od adresu wskazywanego przez *RamfuncsRunStart*. Używana jest funkcja *memcpy* z biblioteki czasu wykonania rts2800_ml.lib.

38. W pliku *Example_F2802xLaunchPadDemo.c* kliknij (zaznacz) na linię 316 z kodem

```
memcpy(&RamfuncsRunStart,
&RamfuncsLoadStart,
(size_t)&RamfuncsLoadSize);
```

39. Kliknij prawym klawiszem (poza tekstem kodu) i wybierz pozycję Run to Line. Program zostanie uruchomiony i zatrzymany na zaznaczonej linii kodu
40. Dodaj zmienne *RamfuncsRunStart*, *RamfuncsLoadStart* oraz struktury *RamfuncsLoadSize* do okna *Expressions*.
41. Otwórz okno *Memory Browser* z menu *View* → *Memory Browser*. Wybierz przestrzeń adresową *Program*, wpisz adres 0x8000 (*RamfuncsRunStart*) i kliknij *Enter*. Zawartość pamięci nie jest przypadkowa, ponieważ kod został już przepisany przez program uruchomiony po włączeniu zasilania zestawu uruchomieniowego.
42. Kliknij prawym klawiszem myszy w oknie *Memory Browser* i wybierz *Fill Memory*. Wpisz adres startowy 0x8000, przestrzeń adresową *Program*, długość 0x20 oraz wartość wstawianą zero (**rysunek 11**).
43. Na pasku narzędziowym okna *Memory Browser* kliknij na przycisk *Open New View*. Przeciągnij nowe okno pamięci tak, aby było ułożone poniżej poprzedniego (**rysunek 12**).

```
Plik definicji pamięci F28027.cmd
MEMORY
{
PAGE 0:    /* Program Memory */
           /* Memory (RAM/FLASH/OTP) blocks can be moved to PAGE1 for data allocation */
 PRAML0     : origin = 0x008000, length = 0x000800 /* on-chip RAM block L0 */
 OTP        : origin = 0x3D7800, length = 0x000400 /* on-chip OTP */
 FLASHD     : origin = 0x3F0000, length = 0x002000 /* on-chip FLASH */
 FLASHC     : origin = 0x3F2000, length = 0x002000 /* on-chip FLASH */
 FLASHA     : origin = 0x3F6000, length = 0x001F80 /* on-chip FLASH */
 CSM_RSVD   : origin = 0x3F7F80, length = 0x000076 /* Part of FLASHA. Program with */
/* all 0x0000 when CSM is in use. */
 BEGIN      : origin = 0x3F7FF6, length = 0x000002 /* Part of FLASHA. Used for "boot */
/* to Flash" bootloader mode. */
 CSM_PWL_P0 : origin = 0x3F7FF8, length = 0x000008 /* Part of FLASHA. CSM password */
/* locations in FLASHA */

 IQTABLES   : origin = 0x3FE000, length = 0x000B50 /* IQ Math Tables in Boot ROM */
 IQTABLES2  : origin = 0x3FEB50, length = 0x00008C /* IQ Math Tables in Boot ROM */
 IQTABLES3  : origin = 0x3FEBDC, length = 0x0000AA /* IQ Math Tables in Boot ROM */

 ROM        : origin = 0x3FF27C, length = 0x000D44 /* Boot ROM */
 RESET      : origin = 0x3FFFC0, length = 0x000002 /* part of boot ROM  */
 VECTORS    : origin = 0x3FFFC2, length = 0x00003E /* part of boot ROM  */

PAGE 1 : /* Data Memory */
       /* Memory (RAM/FLASH/OTP) blocks can be moved to PAGE0 for program allocation */
       /* Registers remain on PAGE1                                                  */

 BOOT_RSVD : origin = 0x000000, length = 0x000050 /* Part of M0, BOOT rom will use*/
/* this for stack */
 RAMM0     : origin = 0x000050, length = 0x0003B0 /* on-chip RAM block M0 */
 RAMM1     : origin = 0x000400, length = 0x000400 /* on-chip RAM block M1 */
 DRAML0    : origin = 0x008800, length = 0x000800 /* on-chip RAM block L0 */
 FLASHB    : origin = 0x3F4000, length = 0x002000 /* on-chip FLASH */
}

/* Allocate sections to memory blocks.
 Note:
 codestart user defined section in DSP28_CodeStartBranch.asm
used to redirect code execution when booting to flash
 ramfuncs  user defined section to store functions that
will be copied from Flash into RAM
*/

SECTIONS
{
 /* Allocate program areas: */
 .cinit     : > FLASHA | FLASHC | FLASHD,       PAGE = 0
 .pinit     : > FLASHA | FLASHC | FLASHD,      PAGE = 0
 .text      : >> FLASHA | FLASHC | FLASHD,       PAGE = 0
 Codestart : > BEGIN        PAGE = 0
 ramfuncs   : LOAD = FLASHA,
              RUN = PRAML0,
              LOAD_START(_RamfuncsLoadStart),
              LOAD_SIZE(_RamfuncsLoadSize),
              RUN_START(_RamfuncsRunStart),
              PAGE = 0

 csmpasswds  : > CSM_PWL_P0   PAGE = 0
 csm_rsvd    : > CSM_RSVD     PAGE = 0

 /* Allocate uninitalized data sections: */
 .stack    : > RAMM0        PAGE = 1
 .ebss     : > DRAML0       PAGE = 1
 .esysmem  : > DRAML0       PAGE = 1
 .sysmem   : > DRAML0       PAGE = 1
 .cio      : >> RAMM0 | RAMM1 | DRAML0       PAGE = 1

 /* Initalized sections go in Flash */
 /* For SDFlash to program these, they must be allocated to page 0 */
 .econst   : > FLASHA       PAGE = 0
 .switch   : > FLASHA       PAGE = 0

}
Spory kawałek pliku to zakomentowane definicje potrzebne w przypadku używania biblioteki IQMath.lib.
Zostały one pominięte.
```


083-089_piccolo(3).indd 88 2013-04-01 23:18:06

44. W nowym oknie wybierz przestrzeń adresową *Program*, wpisz adres 0x3F70C2 (*RamfuncsLoadStart*) i kliknij *Enter*.
45. Kliknij na przycisk pracy krokowej *Step Over* na pasku narzędziowym okna *Debug*. Zauważ zgodność zawartości obu obszarów adresowych pamięci.

Inicjalizowanie ustawień pamięci Flash

Dostęp do różnych obszarów pamięci wewnętrznej układu procesorowego wymaga różnej liczby cykli opóźnienia. Dla pamięci RAM opóźnienia mają stałą wartość, a dla pamięci Flash i OTP opóźnienia są programowane przez użytkownika w rejestrach konfiguracyjnych. Przed rozpoczęciem konfigurowania pamięci Flash wszystkie dostępy do pamięci Flash lub OTP muszą być zakończone. Dotyczy to instrukcji znajdujących się w potoku CPU, odczytów danych, oraz operacji pobierania kodu z wyprzedzeniem. Budowa pamięci Flash, procedura inicjalizacji pamięci i rejestry konfiguracyjne jest dokładnie omówiona w książkach [12, 13].

Inicjalizowanie ustawień pamięci Flash jest wykonywane w funkcji *FLASH_setup(myFlash)* z biblioteki *driverlib*.

46. W pliku *Example_F2802xLaunchPadDemo.c* kliknij (zaznacz) na linię 348 kodu *FLASH_setup(myFlash);* Zaznacz tą linię, kliknij prawym klawiszem (poza tekstem kodu) i wybierz pozycję *Run to Line*.
47. Kliknij na przycisk pracy krokowej *Step Into* na pasku narzędziowym okna *Debug*.
48. Dodaj zmienną strukturalną *FlashRegs* do okna *Expressions*. Wpisz jej nazwę w pole *Add new expression*. Rozwiń drzewo struktury *FlashRegs* oraz struktury bitowe rejestru *FBANKWAIT* (**rysunek 13**).

Pole bitowe RANDWAIT określa czas oczekiwania dla swobodnego dostępu do pamięci Flash. Pole bitowe PAGEWAIT określa czas oczekiwania dla stronicowanego dostępu do pamięci Flash. Dokładne omówienie zagadnienia jest zamieszczone w książce [12].

49. Stosując przycisk *Step Over* przejdź w pracy krokowej do końca kodu funkcji *FLASH_setup* (linia 336 kodu). Zobacz jak zmieniła się wartość liczby cykli opóźnienia dostępu do pamięci Flash.

Okazuje się, że nadal opóźnienie jest ustawione na domyślną (maksymalną) liczbę 15 cykli zegara systemowego.

Brak zmiany jest spowodowany błędnym brakiem aktywnego kodu w funkcji FLASH_setup biblioteki *driverlib* dla symbolu definicji zegara CPU_FRQ_60MHZ, ustawionego w tym projekcie.

Typowo w każdym projekcie włączony jest plik nagłówkowy *DSP28x_Project.h* znajdujący się na głównym poziomie drzewa pakietu firmware. Z tego pliku inkludowane są następne dwa pliki:

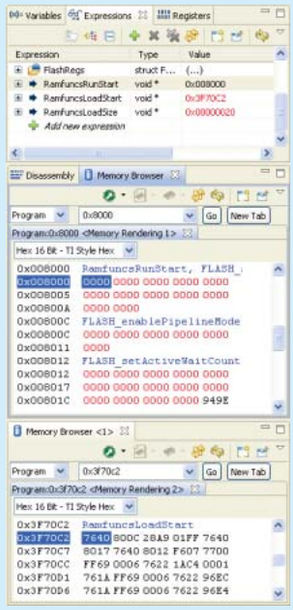
- F2802x_Device.h – umieszczony w tym samym miejscu. Zdefiniowane są w nim symbole opisujące używany układ scalony. Domyślnie definiowane są symbole w tym:

```
#define TARGET 1
#define DSP28_28027PT TAR-
GET
```

- F2802x_Examples.h (ścieżka / f2802x_common/include/)

Zdefiniowane jest w nim makro DELAY_US wyskalowane w mikrosekundach oraz symbole z wartościami domyślnymi ustawień dla F28027 Piccolo w tym:

```
CPU_RATE 16.667L // for
a 60MHz CPU clock speed
(SYSCLKOUT)
#define CPU_FRQ_60MHZ 1 //
60 Mhz CPU Freq (10 MHz
input clock)
```

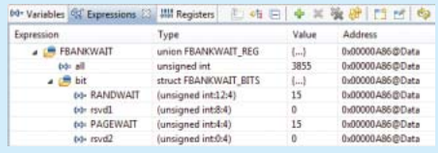


Rysunek 12. Zmienne sterujące i obszary adresowe dla przepisywania kodu z pamięci Flash do RAM

Inne symbole definicji zegara to CPU_FRQ_50MHZ oraz CPU_FRQ_40MHZ.

Uruchamianie projektu Example_F2802xLaunchPadDemo

50. Kliknij na przycisk *Resume* na pasku narzędziowym okna *Debug*. Opis działania programu i jego obsługi jest zamieszczony w ramce *„Działanie programu przykładowego Example_F2802xLaunchPadDemo”*.



Rysunek 13. Ustawienia zdefiniowania opóźnienia dostępu do pamięci Flash

Zaprezentowane w artykule postępowanie pozwala na poznanie programowania pamięci Flash układów procesorowych Piccolo F2802x w środowisku programowym CCSv5. Uruchamianie przykładowego projektu z pakietu programowego controlSUITEv3 umożliwia poznanie sposobów pracy programu z pamięci Flash układów procesorowych Piccolo F2802x.

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

083-089_piccolo(3).indd 89 2013-04-01 23:18:06