C2000 Piccolo LanuchPad (9)

Łatwa obsługa modułu PWM procesora serii Piccolo F2802x

Moduł generujący sygnał PWM jest jednym z najważniejszych elementów układu procesorowego serii Piccolo F2802x. Moduł umożliwia generowanie złożonych sygnałów cyfrowych przy minimalnym obciążeniu procesora. Moduł realizuje funkcję przetwarzania analogowo-cyfrowego, gdzie współczynnik wypełnienia pełni funkcję odpowiednika ustawiania wartości analogowej przetwornika analogowo-cyfrowego. Zastosowanie biblioteki driverlib z pakietu programowego controlSUITE znacznie ułatwia obsługę modułu ePWM procesorów serii Piccolo F2802x 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.

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 I^2C", EP 07/2013.
- C2000 Piccolo LanuchPad (6) – Łatwa inicjalizacja systemowa procesora serii Piccolo F2802x", EP 09/2013.
- „C2000 Piccolo LanuchPad (7) – Łatwa obsługa wyświetlacza LCD", EP 11/2013.
- „C2000 Piccolo LanuchPad (8) – Budowanie biblioteki drivelib dla procesorów serii Piccolo F2802x", EP 12/2013.

Konfiguracja sprzętowa i programowa

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

- Środowisko *Code Composer Studio* v5.5.0.00077 (Sep 9, 2013) firmy Texas Instruments [1, 13, 15]. Umożliwia tworzenie w środowisku CCSv5 programów przeznaczonych dla procesorów serii Piccolo TMS320F2802x.
- Pakiet programowy controlSUITE v3.2.3 (07-Nov-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. Jest nim 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_PWM*. Wymagane są prawa dostępu (zapisu i modyfikacji) dla tej ścieżki dyskowej. Możliwe jest umieszczenie foldera *home_dir* na innym wolumenie dyskowym z prawami dostępu.

Do wykonania ćwiczenia jest potrzebny oscyloskop dwukanałowy z sondami.

Cel ćwiczenia

Celem ćwiczenia jest praktyczne poznanie programowania modułu ePWM układu procesorowego serii Piccolo F2802x przy użyciu biblioteki *driverlib* pakietu programowego controlSUITEv3 oraz środowiska *Code Composer Studio* v5. Zastosowano przykładowy projekt ***Example_F2802xEPwmUpAQ*** 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 budowy i inicjalizowania modułu ePWM, poznanie sposobu dołączania pliku do projektu, poznanie sposobu debugowania programu w trybie Real-Time oraz poznanie pracy swobodnej modułu ePWM i obsługę krytycznych przerwań..

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 modułu ePWM układu procesorowego serii Piccolo F2802x jest zamieszczony w dokumencie *TMS320x2802x, 2803x Piccolo Enhanced Pulse Width Modulator (ePWM) Module* [6]. Opis konfigurowania wyprowadzeń GPIO oraz obsługi przerwań jest zamieszczony w dokumencie *TMS320x2802x Piccolo System Control and Interrupts* [5]. 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 control-

SUITEv3 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" [14].

Dokładne omówienie pracy z modułem ePWM układu procesorowego serii Piccolo F2802x jest zamieszczone w książce Henryk A. Kowalski, „Procesory DSP w przykładach" [15].

Dokładne omówienie zestawu ewaluacyjnego *C2000 Piccolo LaunchPad* jest zamieszczone w artykule Henryk A. Kowalski „Zestaw ewaluacyjny C2000 Piccolo LaunchPad" [12].

Dokładne omówienie środowiska CCSv5 oraz pakietu controlSUITEv3 jest zamieszczone w artykule Henryk A. Kowalski „C2000 Piccolo LanuchPad (2) – Łatwe programowanie z pakietem controlSUITE" [11].

Opis instalowania najnowszej wersji środowiska CCS i pakietu controlSUITE jest zamieszczony w artykule Henryk A. Kowalski „C2000 Piccolo LanuchPad (8) – Budowanie biblioteki drivelib dla procesorów serii Piccolo F2802x" [7].

Dołączenie 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 jest wymagana 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].

Szybkość zegara podstawy czasu modułu ePWM
TBCLK = SYSCLKOUT / (2x HSPCLKDIV × 2CLKDIV) dla HSPCLKDIV>0
TBCLK = SYSCLKOUT / (2CLKDIV) dla HSPCLKDIV=0,
gdzie
HSPCLKDIV[2:0](TBCTL[9:7]) oraz CLKDIV[2:0](TBCTL[12:10]) są polami bitowymi rejestru TBCTL.
Okres TPWM generowanego sygnału PWM jest określany poprzez zawartość rejestru TBPRD
TPWM = (TBPRD + 1) x TTBCLK
Częstotliwość sygnału PWM jest definiowana jako
FPWM = 1/ (TPWM).

Uruchamianie środowiska CCSv5

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

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.

1. 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_PWM***. 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 dokumentację pakietu.

Aby dodać dostęp do przykładowych projektów należy na dole strony *Home* kliknąć na odnośnik *Configure Resource Explorer*.

Jeśli w białym polu wyboru okna dialogowego *Package Configuration* jest pokazywana nazwa *controlSUITE* to należy na nią kliknąć a następnie należy kliknąć przycisk *Remove* oraz przycisk *OK*. Okno jest zamykane i środowisko CCS usuwa niepoprawnie zbudowaną bazę informacji o projektach przykładowych. Następnie na

dole strony *Home* należy ponownie kliknąć na odnośnik *Configure Resource Explorer*.

Jeśli w białym polu wyboru okna dialogowego *Package Configuration* jest pusto to 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_F2802xEPwmUpAQ

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_F2802xEPwmUpAQ***.

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

Krok1: Importowanie projektu *Example_F2802xEPwmUpAQ* 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_F2802xEPwmUpAQ* został zaimportowany z kopiowaniem projektu i pliku ***Example_2802xEPwmUpAQ.c*** do foldera roboczego projektu.

Krok2: Budowanie projektu *Example_F2802xEPwmUpAQ*

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_F2802xEPwmUpAQ* zostało zakończone poprawnie. Został utworzony wynikowy plik binarny ***Example_2802xEPwmUpAQ***.out (zobacz okno *Console*). Zostały jednak zgłoszone 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_F2802xEPwmUpAQ*

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*.

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 ePWM1 udostępniana jest struktura rejestrów sterowania EPwm1Regs.

9. Przełącz się do perspektywy *CCS Edit*. W oknie *Project Explorer* kliknij prawym klawiszem myszy na linię nazwy projektu *Example_F2802xEPwmUpAQ*. 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.

10. 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*.

11. Przełącz się do perspektywy *CCS Debug*. Zauważ w oknie edytora, że praca programu została zatrzymana na pierwszej linii kodu funkcji *main()*.

12. Otwórz okno *Disassembly* z menu *View* → *Disassembly*. W tym oknie można dokładnie zobaczyć jak naprawdę pracuje układ procesorowy Piccolo F28027.

Wgląd w projekt Example_F2802xEPwmUpAQ

13. Zapoznaj się z komentarzem na początku pliku ***Example_2802xEPwmUpAQ.c.***

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

14. Dodaj struktury *EPwm1Regs*, *EPwm2Regs* oraz *EPwm3Regs* do okna *Expressions*.

Konfigurowanie wyprowadzeń cyfrowych I/O (GPIO) dla modułu ePWM1/2/3

W funkcji *main()* wykonywane jest konfigurowanie wyprowadzeń przypisanych do modułów ePWM (od linii 149). Dla każdego wyjścia EPWM1A oraz EPWM1B można przypisać tylko jedno wyprowadzenie GPIO. Jest to przypisanie sztywne i nie można go zmieniać. I tak np. wyjście EPWM1A może być przypisane tylko do wyprowadzenia GPIO0, a wyjście EPWM1B może być przypisane tylko do wyprowadzenia GPIO1. Dodatkowo dla tych wyprowadzeń wyłączany jest (niepotrzebny) układ podciągania. Tak samo jest przypisywane są wyjścia modułów ePWM2 oraz ePWM3.

Konfigurowanie modułu ePWM1/2/3

Dalej w funkcji main() najpierw wyłączany jest zegar taktowania wszystkimi modułami. Następnie wywoływane są funkcje konfigurowania poszczególnych modułów ePWM. Wszystkie trzy moduły ePWM są konfigurowane w taki sam sposób. Ale z różnymi wartościami parametrów. Na koniec ponownie włączany jest zegar taktowania wszystkimi modułami ePWM.

15. Kliknij lewym klawiszem myszki na linię 175 wywołania funkcji *InitEPwm1Example();*. Kliknij ponownie prawym klawiszem myszki na tą linię i wybierz polecenie *Run to line*. Kliknij na przycisk Step Into . Zauważ, że w oknie *Debug* został pokazany następny poziom stosu wywołań funkcji.

Podmoduł podstawy czasu (TB)

W funkcji *InitEpwm1Example()* najpierw włączany jest zegar systemowy dla modułu ePWM1.

Następnie konfigurowany jest podmoduł podstawy czasu (TB). Ustawiany jest tryb zliczania w górę dla pracy licznika podstawy czasu bez synchronizacji fazy. Licznik podstawy czasu jest zerowany i ustawiana jest szybkość zegara podstawy czasu oraz okres przebiegu PWM. Okres wszystkich modułów jest taki sam.

Zegar podstawy czasu TBCLK jest przeskalowaną wersją sygnału zegara systemowego SYSCLKOUT (zobacz ramka). Szybkość pracy modułów ePWM1 i ePWM2 jest taka sama. Moduł ePWM3 pracuje z czterokrotnie szybszym zegarem.

Podmoduł porównania licznika (CC)

Ustawiany jest tryb wpisywania do rejestrów porównania z użyciem rejestrów pośrednich (CC_SHADOW = 0x0) oraz ładowanie (aktualizowanie) rejestrów roboczych dla zdarzenia CTR=Zero (CC_CTR_ZERO = 0x0) czyli dla TBCTR=0x0000.

Dalej ustawiane są początkowe wartości rejestru porównania CMPA=50 i CMPB=50 dla modułu ePWM1. Wartości minimalne/maksymalne wynoszą po 50/1950 dla obu komparatorów tego modułu.

Dla modułu ePWM2 są takie same ustawienia.

Dla modułu ePWM3 ustawiane początkowe wartości rejestru porównania CMPA=50 i CMPB=1050. Wartości minimalne/maksymalne dla CMPA wynoszą po 50/950 a dla CMPB 1050/1950.

Ustawienie rejestrów CMPA i CMPB modułów ePWM są zmieniane w kodzie obsługi ich przerwań.

Podmoduł kwalifikacji i akcji (AQ)

Dla modułu ePWM1 dla zdarzenia CTR=Zero (czyli dla TBCTR=0x0000) wykonywana jest akcja ustawienia poziomu wysokiego na wyjściu EPWM1A oraz EPWM1B. Gdy zawartość licznika podstawy czasu TBCTR jest równa zawartości rejestru CMPA wykonywana jest akcja zerowania – ustawienia poziomu niskiego na wyjściu EPWM1A. Gdy zawartość licznika podstawy czasu TBCTR jest równa zawartości rejestru CMPB wykonywana jest akcja zerowania – ustawienia poziomu niskiego na wyjściu EPWM1B.

Dla modułu ePWM2 dla zdarzenia CTR=Zero (czyli dla TBCTR=0x0000) wykonywana jest akcja zerowania –ustawienia poziomu niskiego na wyjściu EPWM2A oraz EPWM2B.

Gdy zawartość licznika podstawy czasu TBCTR jest równa zawartości rejestru CMPA wykonywana jest ustawienia poziomu wysokiego na wyjściu EPWM2A. Gdy zawartość licznika podstawy czasu TBCTR jest równa zawartości rejestru CMPB wykonywana jest akcja ustawienia poziomu wysokiego na wyjściu EPWM2B.

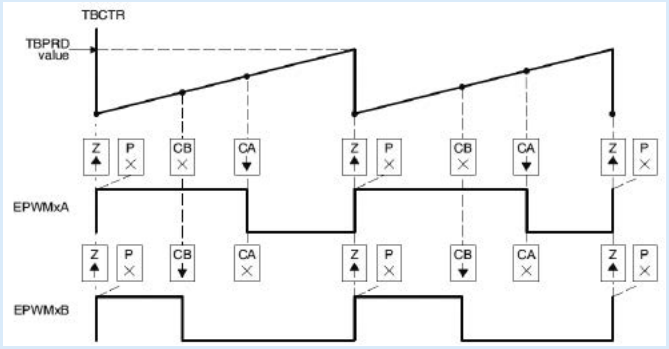
Dla modułu ePWM3 zdefiniowanie akcji wygląda inaczej. Dla zdarzenia CTR=Zero (czyli dla TBCTR=0x0000) wykonywana jest akcja zmiany poziomu na wyjściu EPWM3B. Gdy zawartość licznika podstawy czasu TBCTR jest równa zawartości rejestru CMPA wykonywana jest akcja ustawienia poziomu wysokiego na wyjściu na wyjściu EPWM3A. Gdy zawartość licznika podstawy czasu TBCTR jest równa zawartości rejestru CMPB wykonywana jest akcja zerowania – ustawienia poziomu niskiego na wyjściu EPWM3A.

Symboliczne (graficzne) oznaczenia akcji są pokazane na rys. 1. Ich dokładne znaczenie jest omówione w książce Henryk A.Kowalski Procesory DSP dla praktyków" [14].

Na rys. 2 jest pokazany przykładowy sygnał obu wyjść modułu ePWM z zaznaczonymi akcjami. Jest on

S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
SW X	Z X	CA X	CB X	P X	Do Nothing
SW ↓	Z ↓	CA ↓	CB ↓	P ↓	Clear Low
SW ↑	Z ↑	CA ↑	CB ↑	P ↑	Set High
SW T	Z T	CA T	CB T	P T	Toggle

Rysunek 1. Akcje podmodułu kwalifikacji i akcji AQ modułu ePWM [SPRUGE9E]



Rysunek 2. Asymetryczny przebieg PWM podczas pracy modułu ePWM [Typ1] w trybie zliczania w górę [SPRUGE9E]

zgodny ze sposobem pracy modułu ePWM1. Jednak wartości CMPA i CMPB modułu ePWM zastosowane w programie są sobie równe.

Podmoduł ET (Event-Trigger)

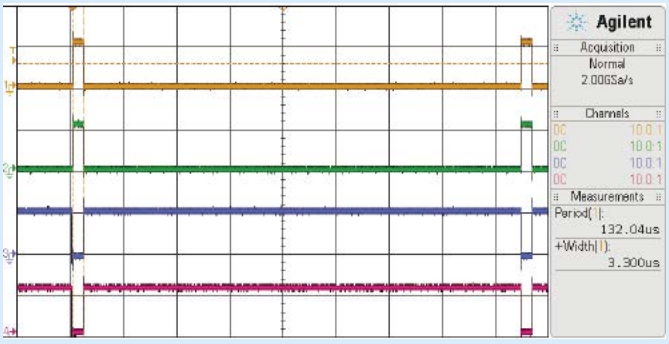
Podmoduł ET umożliwia określenie sposobu generowania żądania obsługi przerwania. Przerwanie EPWM1_INT może być zgłaszane dla zdarzenia CTR=0 (ET_CTR_ZERO = 0x1), czyli dla TBSCTR = 0x0000. Włączane jest zezwolenie na zgłaszanie przerwania do PIE. Jednak tylko co trzecie zdarzenie CTR=0 spowoduje zgłoszenie przerwania do modułu PIE (ET_3RD = 0x3). Tak samo skonfigurowane są moduły ePWM2 i ePWM3 dla zgłaszania przerwań EWPM2_INT oraz EPWM3_INT.

Na końcu funkcji konfigurowania modułu ePWM1 jest inicjowana struktura *epwm1_info* zawierająca informacje o stanie działania modułu. To samo jest wykonywane dla modułów ePWM2 i ePWM3.

Procedury obsługi przerwań modułu ePWM

W procedurze obsługi przerwań zgłaszanych przez moduł ePWM1 wykonywana jest aktualizacja stanu struktury *epwm1_info* informacji o stanie działania modułu. Następnie zerowany jest znacznik INT[ETFLG(0)] zgłoszenia żądania obsługi przerwania modułu ePWM1 poprzez wpis jedynki do bitu INT rejestru ETCLR (kasowania). Na końcu procedury włączane (przywracane) jest dla grupy 3 w PIE zezwolenie na zgłaszanie przerwania do CPU.

Aktualizacja stanu struktury *epwm1_info* informacji o stanie działania modułu jest wykonywana w funkcji *update_compare*. Jest ona wywoływana procedury obsługi przerwania modułów ePWM. Obsługa co 10-tego przerwania powoduje zmianę – zwiększenie/zmniejszenie ustawienia poziomu komparatora CMPA oraz CMPB. Gdy ustawienia osiągną wartość minimalną/maksymalną wykonywane jest odwracanie kierunku zmian. Najłatwiej zobaczyć to w działaniu, po uruchomieniu programu w dalszej części tego ćwiczenia.



Rysunek 3. Sygnały EPWM1A (od góry), EPWM1B, EPWM2A oraz EPWM2B

Praca programu Example_2803xEPwmUpAQ

Po wykonaniu konfigurowania modułów ePWM włączany jest w funkcji *main()* jednocześnie (synchronicznie) zegar taktowania wszystkimi modułami ePWM (bit TBCLKSYNC[PCLKCR0(2)]. Oznacza to, że zegary poszczególnych modułów ePWM mają stałą fazę.

Adresy procedur obsługi przerwań modułów EPWM są przypisywane do odpowiednich wektorów przerwań modułu PIE.

Ustawiane jest zezwolenie na obsługę przerwania INT3 zgłaszanego przez moduł PIE do CPU. Ustawiane jest zezwolenie na obsługę przerwań zgłaszanych do modułu PIE w grupie 3 (przerwanie INT3.1. INT3.2 oraz INT3.3). Teraz włączane jest globalne zezwolenie na zgłaszanie przerwań w trybie normalnym (ustawienie na jedynkę bitu INTM) oraz w trybie debugowym (ustawienie na jedynkę bitu DBGM) pracy układu procesorowego.

Praca modułu ePWM w trybie asymetrycznym (up)

Koniec funkcji *main()* to typowa pętla nieskończona. Taka pętla realizuje proces tła (wątek podstawowy) o najniższym priorytecie. Każde przerwanie sprzętowe definiuje własny wątek o priorytecie wyższym niż proces tła.

16. Dołącz sondę jednego kanału oscyloskopu do sygnału EPWM1A (GPIO0) oraz sondę drugiego kanału do sygnału EPWM1B (GPIO1).

17. Wykonaj polecenie *Resume*. Zaobserwuj przebiegi w obu kanałach. Zaobserwuj na ekranie przesuwanie zbocza zmiany poziomu sygnału. Zmierz (na ekranie) okres obu sygnałów.

Przykładowy przebieg sygnałów EPWM1A, EPWM1B, EPWM2A oraz EPWM2B jest pokazany na rys. 3. Okres wszystkich sygnałów jest taki sam. Zmiana wypełnienia jest również taka sama. Jedynie polaryzacja sygnałów EPWM2A oraz EPWM2B jest odwrócona.

Debugowanie programu w trybie Real-Time

18. W funkcji funkcji *update_compare*() ustaw pułapkę w drugiej linii kodu

```
epwm_info->EPwmTimerIntCount = 0;.
```

Jeśli ikonka pułapki pokazywana na lewo od linii jest szara to oznacza, że debugger nie pracuje w trybie Real-time (pułapka jest nieaktywna).

19. Aby włączyć tryb Real-time należy z menu wybrać pozycję *Tools → Debugger Options → Auto Run and Launch Options* (rys. 5).

Typowo po wystartowaniu debugera automatycznie zaznaczana jest opcja *Enable polite mode (respect HPI, DBGM and FRAMEID)*. Należy zaznaczyć opcję *Enable realtime mode (critical interrupts serviced when halted, rude/polite mode …)*. Jeśli pojawi się okno podpowiedzi o zapisywaniu ustawień to należy kliknąć *OK*.

20. Zdejmij nieaktywną pułapkę (dwukliknij na szarą ikonkę pułapki).

Ponownie dwukliknij na tą samą linię. Jeśli pojawi się okno to należy kliknąć na przycisk *Rude Retry*. Spowoduje to przejście do trybu debugowego z wymuszeniem

obsługi działań debugowych. Procesor został zatrzymany w tej linii kodu obsługi przerwania. Jednocześnie jest zablokowana obsługa wszystkich pozostałych przerwań przez procesor (bit INTM(ST1(0)] =1). Na wyjściu wszystkich modułów EPWM1/2/3 nie jest generowany sygnał PWM. Tryby debugowe pracy procesora zostały dokładnie omówione w książce Henryk A.Kowalski „ Procesory DSP dla praktyków" [14].

21. Sprawdź wartości struktury *epwm_info* w oknie *Variables*. Sprawdź wartości struktur w oknie *Expressions*. Sprawdź, czy program działa zgodnie z opisem w funkcji *update_compare*.

22. Użyj poleceń pracy krokowej *Step Over* (lub *Step Into*) do wykonywania kolejnych linii kodu. Zobacz w oknie *Expressions*, że pracuje licznik podstawy czasu TBCTR modułów ePWM.

Po kilkukrotnym wykonaniu poleceń pracy krokowej pokazywane jest wykonanie kodu podstawowej pętli nieskończonej w linii (195)

```
asm(„ NOP");
```

Dlaczego program nie wchodzi w obsługę przerwań?

Ponieważ, jeśli zostało wydane polecenie STEP 1 (pracy krokowej) to CPU nie może obsługiwać żadnego przerwania, włącznie z /NMI oraz /RS.

Aby wykonać fragment kodu z włączoną obsługą przerwań należy ustawić pułapkę na końcu tego fragmentu i wykonać polecenie *Run*.

23. Zmodyfikuj program

W pętli nieskończonej, za instrukcją asm wstaw nową linię kodu (opóźnienie o 1s)

```
DELAY_US(1000000L);
```

24. 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*. Przełącz się do perspektywy *CCS Debug*.

25. Kilka razy powtórz polecenie *Resume* . Obserwuj na ekranie sygnały wyjściowe PWM. Sygnały wyjściowe powinny być generowane z przesuwaniem zbocza przez czas ok. 1 sek.

26. Usuń pułapkę. Wykonaj polecenie *Resume* . Dołącz sondę jednego kanału oscyloskopu do sygnału EPWM2A (GPIO2) oraz sondę drugiego kanału do sygnału EPWM2B (GPIO3). Zaobserwuj przebiegi w obu kanałach.

27. Dołącz sondę jednego kanału oscyloskopu do sygnału EPWM3A (GPIO4) oraz sondę drugiego kanału do sygnału EPWM3B (GPIO5). Zaobserwuj przebiegi w obu kanałach.

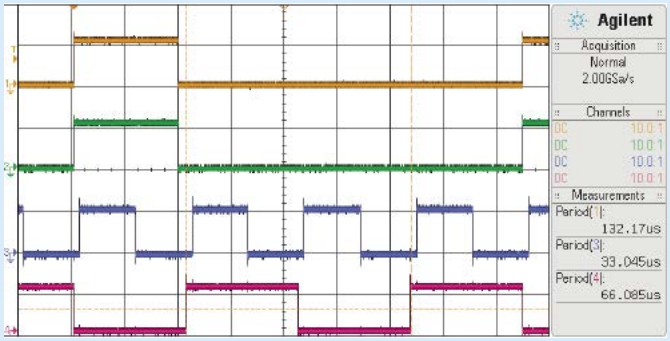
Przykładowy przebieg sygnałów EPWM1A, EPWM1B, EPWM3A oraz EPWM3B jest pokazany na rys.4. Okres sygnału EPWM3B jest dwukrotnie mniejszy niż sygnału EPWM1A. Okres sygnału EPWM3A jest czterokrotnie mniejszy niż sygnału EPWM1A. Dla sygnału EPWM3A przesuwane są zbocza sygnału. Dla sygnału EPWM3B wypełnienie jest stałe.

Praca swobodna modułu ePWM po zatrzymaniu debagowym

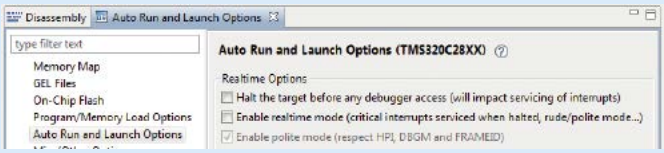
28. Zmodyfikuj program. Na końcu każdej procedury inicjalizacji modułu ePWM dodaj linię kodu ustawiania pola FREE_SOFT[TBCTL(15:14)]. Zmieniaj numer 1/2/3 modułu ePWM w linii

```
Epwm1Regs.TBCTL.bit.FREE_SOFT = 0x2;
```

Pole to określa zachowanie licznika podstawy czasu (a tym samym całego modułu) podczas emulacji. Do-



Rysunek 4. Sygnały EPWM1A (od góry), EPWM1B, EPWM3A oraz EPWM3B



Rysunek 5. Ustawianie opcji pracy debugera w trybie czasu rzeczywistego.

myślnie ustawione jest na zero – zatrzymaj po następnym zwiększenie/zmniejszeniu zawartości licznika. Ustawienie na 0x2 oznacza pracę swobodną, czyli bez zatrzymania działania modułu ePWM po zatrzymaniu działania CPU na pułapce.

29. Wykonaj budowanie projektu i załadowanie kodu wynikowego do układu procesorowego. Wykonaj polecenie *Resume*. Zaobserwuj przebiegi sygnałów EPWM1A (GPIO0) oraz EPWM3A (GPIO4) na oscyloskopie.

30. Ustaw pułapkę wewnątrz pętli nieskończonej. Wykonaj kilka razy polecenie Run. Zaobserwuj przebiegi sygnałów EPWM1A (GPIO0) oraz EPWM3A (GPIO4) na oscyloskopie.

Po zatrzymaniu działania programu sygnały są nadal generowane przez moduły ePWM. Nie jest tylko wykonywana obsługa przerwań, a tym samym modyfikacja wypełnienia sygnałów.

Obsługa krytycznych przerwań po zatrzymaniu debagowym

31. Zmodyfikuj program.

Przed linią kodu

```
EINT;
//Enable Global interrupt INTM
```

dodaj linię kodu

```
SetDBGIER(IER);
//Configure the DBGIER for realtime debug
```

Funkcja asemblerowa SetDBGIER wykonuje przepisanie zawartości rejestru IER do rejestru DBGIER. Jest to wykonywane w bezpieczny sposób poprzez użycie stosu. Funkcja asemblerowa SetDBGIER jest zdefiniowana w pakiecie programowym firmware. Funkcja jest umieszczona w pliku DSP2802x_DBGIER.asm i znajduje się w ścieżce C:\TI\controlSUITE\device_support\f2802x\v210\DSP2802x_common\source.

32. Dołącz plik DSP2803x_DBGIER.asm do projektu.

Przełącz się do perspektywy *CCS Edit*. W oknie *Project Explorer* kliknij prawym klawiszem myszy na linię nazwy projektu *Example_F2802xEPwmUpAQ*. Z podręcznego menu wybierz *Add Files* oraz wskaż powyższą ścieżkę. Zaznacz plik *DSP2802x_DBGIER.asm* i kliknij na *Otwórz*. W oknie *File Operation* zaznacz opcję ***Link to files***. Kliknij *OK*. Plik zostanie dołączony (nie dodany) do projektu.

33. 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*. Przełącz się do perspektywy *CCS Debug*.

34. Wykonaj polecenie *Resume*. Po zatrzymaniu wykonania programu na pułapce zaobserwuj przebiegi sygnałów EPWM1A (GPIO0) oraz EPWM3A (GPIO4) na oscyloskopie.

Po zatrzymaniu działania programu sygnały są nadal generowane przez moduły ePWM oraz jest wykonywana obsługa przerwań, a tym samym modyfikacja wypełnienia sygnałów.

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 Enhanced Pulse Width Modulator (ePWM) Module [SPRUGE9E.pdf], 23 Mar 2011

[7] C2000 Piccolo LanuchPad (8) – Budowanie biblioteki drivelib dla procesorów serii Piccolo F2802x", EP 12/2013

[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 „C2000 Piccolo LanuchPad (2) – Łatwe programowanie z pakietem controlSUITE", Elektronika Praktyczna 03/2013

[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/