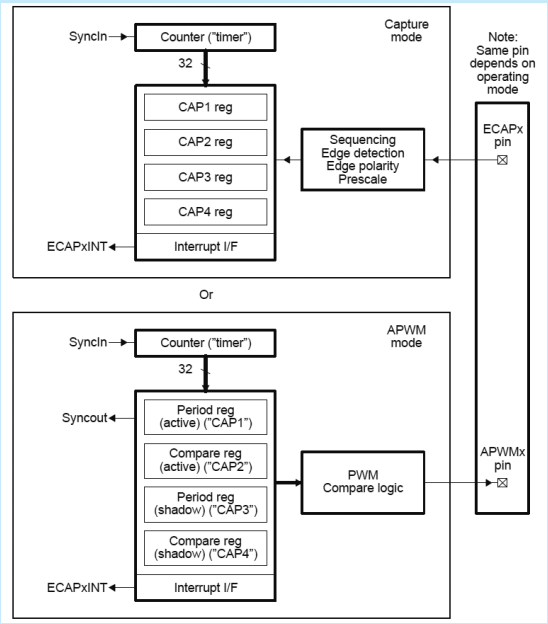
C2000 Piccolo LanuchPad (9)

Łatwa obsługa modułu eCAP procesora serii Piccolo F2802x

Moduł eCAP reprezentuje kompletny kanał zbierania danych o zależnościach czasowych sygnałów cyfrowych dla układów procesorowych serii Piccolo F2802x/3x. Moduł eCAP ma stosunkowo prostą budowę. Jednak sposób jego pracy jest dosyć skomplikowany i zależny od ustawienia wielu bitów sterujących. Zastosowanie biblioteki driverlib z pakietu programowego controlSUITE znacznie ułatwia obsługę modułu eCAP w środowisku programowym CCSv5.



Rysunek 1. Skonfigurowanie modułu eCAP przy pracy w trybie CAPTURE (góra) oraz w trybie APWM (dół) [6]

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
- „C2000 Piccolo LanuchPad (9) – Łatwa obsługa modułu PWM procesora serii Piccolo F2802x", EP 1/2014

Moduł eCAP może pracować w jednym z dwóch trybów (**rysunek 1**).

1. **Tryb CAPTURE**. Moduł eCAP obsługuje cztery niezależne zdarzenia. Zdarzenie polega na wykryciu zbocza sygnału cyfrowego na podzielonym sygnale wejściowym.
2. **Tryb APWM**. Moduł eCAP pracuje jako dodatkowy (auxiliary) moduł PWM. Do modułu eCAPx zostaje przypisane wyprowadzenie GPIO układu procesorowego jako wyprowadzenie ECAPx. Dla układu procesorowego Przypisane do modułu eCAP wyprowadzenie układu procesorowego pracuje jako wejście w trybie CAPTURE oraz jako wyjście w trybie APWM. Piccolo TMS320F28027 wejście/wyjście ECAP1 może być przypisane do wyprowadzenia GPIO5 lub GPIO19.

Do tworzenia w środowisku CCSv5 programów przeznaczonych dla procesorów rodziny Piccolo TMS320F2802x firmy Texas Instruments potrzebny jest pakiet programowy controlSUITE tej firmy. Zawiera on oprogramowanie „firmware", biblioteki, opisy zestawów sprzętowych oraz projekty przykładowe dla wszystkich serii procesorów rodziny C2000. Projekty przykładowe pakietu controlSUITE zwierają na początku kodu programu sekwencję inicjalizacji systemowej układu procesorowego serii Piccolo F2802x.

Konfiguracja sprzętowa i programowa

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

- Środowisko *Code Composer Studio* v5.5.0.00077 (Sep 9, 2013) firmy Texas Instruments [1, 13, 15]. Umożliwia tworzenie w środowisku CCSv5 programów przeznaczonych dla procesorów serii Piccolo TMS320F2802x.
- Pakiet programowy controlSUITE v3.2.4 (10-Dec-2013) firmy Texas Instruments [2, 13, 15]. Zawiera oprogramowanie „firmware", biblioteki, opisy zestawów sprzętowych oraz projekty przykładowe dla wszystkich serii procesorów rodziny C2000.

Platforma sprzętowa wymaga tylko jednego elementu:

- Zestaw ewaluacyjny *C2000 Piccolo LaunchPad* firmy Texas Instruments z układem procesorowym Piccolo TMS320F28027 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 z sondą.

Cel ćwiczenia

Celem ćwiczenia jest praktyczne poznanie programowania modułu eCAP układu procesorowego serii Piccolo F2802x przy użyciu biblioteki *driverlib* pakietu programowego controlSUITEv3 oraz środowiska *Code Composer Studio* v5. Zastosowano dwa przykładowe projekty ***Example_F2802xECap_Capture_Pwm*** oraz ***Example_F2802xECap_apwm*** z tego pakietu pracujące 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 eCAP przy pracy w trybie CAPTURE oraz APWM, poznanie sposobu rozbudowy biblioteki *driverlib* o własne funkcje oraz poznanie sposobu debugowania programu podczas pracy dwóch peryferyjnych modułów sprzętowych.

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 eCAP układu procesorowego serii Piccolo F2802x jest zamieszczony w dokumencie *TMS320x2802x, 2803x Piccolo Enhanced Capture Module (eCAP)* [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 controlSUITEv3 jest zamieszczony w dokumencie *F2802x Peripheral Driver Library USER'S GUIDE v. 210* [9].

Dokładne omówienie budowy układu procesorowego serii Piccolo F2802x jest zamieszczone w książce Henryk A. Kowalski „Procesory DSP dla praktyków" [14].

Dokładne omówienie pracy z modułem eCAP 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].

Podłączenie i skonfigurowanie zestawu C2000 Piccolo LaunchPad

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

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

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

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

Uruchamianie środowiska CCSv5

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

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_eCAP***. 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.

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.

Praca modułu eCAP w trybie CAPTURE

Rejestr licznika TSCTR jest 32-bitowy i określa podstawę czasową dla zdarzeń modułu eCAP. Jest on dołączony do zegara systemowego SYSCLKOUT układu procesorowego. Dla każdego z czterech sygnałów ładowania LD1 do LD4 możliwa jest opcja zerowania licznika TSCTR. Wartość licznika TSCTR jest zapisywana najpierw do rejestru CAPn a potem wykonywane jest zerowanie. Taki sposób pracy jest użyteczny do różnicowego pomiaru czasu.

Cyfrowy sygnał wejściowy może być podawany z wyprowadzenia układu procesorowego bezpośrednio do modułu eCAP lub może być podzielony przez wartość w zakresie 1 do 62.

Moduł eCAP w trybie CAPTURE może pracować w jednym z dwóch trybów:

- tryb ciągły
- tryb jednokrotny.

Podczas pracy modułu eCAP w trybie ciągłym licznik Mod4 zlicza w górę od zera do 3 i ponownie od zera. Wartości z licznika TSCTR są wpisywane do rejestrów CAP1 do CAP4 w sposób cykliczny (modulo 4).

Podczas pracy w trybie jednokrotnym praca licznika Mod4 jest kontrolowana przez logikę sterowania. Logika kontroluje operacje Reset, start oraz stop licznika Mod4. Dodatkowo dwubitowy rejestr StopValue jest używany do porównania stanu licznika Mod4. Moduł eCAP po zaprogramowaniu (armed) czeka od 1 do 4 zdarzeń na sygnał porównania i (po wcześniejszym wpisaniu aktualnego stanu licznika TSCTR do odpowiedniego rejestru CAPx) zamraża stan licznika Mod4 i zawartość rejestrów CAP1 do CAP4. Ponowne zaprogramowanie modułu (rearm) przygotowuje moduł do kolejnej sekwencji działania. Powoduje to wyzerowanie licznika Mod4 oraz zezwolenie na wpis do rejestrów CAP1 do CAP4.

Zastosowanie pierwszego projektu: Example_F2802xECap_Capture_Pwm

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_F2802xECap_Capture_Pwm***.

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

Krok1A: Importowanie projektu Example_F2802xECap_Capture_Pwm 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_F2802xECap_Capture_Pwm* został zaimportowany z kopiowaniem projektu i pliku ***Example_2802xECap_Capture_Pwm.c*** do foldera roboczego projektu.

Krok2A: Budowanie projektu Example_F2802xECap_Capture_Pwm

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*.

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_F2802xECap_Capture_Pwm* zostało zakończone poprawnie. Został utworzony wynikowy plik binarny ***Example_2802xECap_Capture_Pwm.out*** (zobacz okno *Console*). Zostały jednak zgłoszone ostrzeżenia (zobacz okno *Problems*). Na razie są one nieistotne.

Krok3A: 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).

Krok4A: Uruchamianie sesji debugowej dla projektu Example_F2802xECap_Capture_Pwm

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

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

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

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

Wgląd w projekt Example_F2802xECap_Capture_Pwm

9. W perspektywie *CCS Debug* zauważ w oknie edytora, że praca programu została zatrzymana na pierwszej linii kodu funkcji *main()*.
10. Otwórz okno *Disassembly* z menu *View* → *Disassembly*. W tym oknie można dokładnie zobaczyć jak naprawdę pracuje układ procesorowy Piccolo F28027.
11. Zapoznaj się z komentarzem na początku pliku ***Example_2802xECap_Capture_Pwm.c.***

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

Konfigurowanie wyprowadzeń cyfrowych I/O (GPIO) dla modułu eCAP1 i ePWM3

W funkcji *main()* wykonywane jest konfigurowanie wyprowadzenia GPIO5 przypisanego jako wejście ECAP1 do modułu eCAP1. Dodatkowo wyprowadzenie GPIO4 jest przypisane jako wyjście EPWM3A modułu ePWM.

Dalej w funkcji *main()* adres procedury obsługi przerwania ECAP1_INT modułu eCAP1 jest przypisywany do odpowiedniego wektora przerwań grupy 4 modułu PIE. Ustawiane jest zezwolenie na obsługę przerwania INT4 zgłaszanego przez moduł PIE do CPU. Ustawiane jest zezwolenie na obsługę przerwań zgłaszanych do modułu PIE w grupie 4. 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.

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.

Konfigurowanie modułu ePWM3

W funkcji *main()* wywoływana jest funkcja *InitEPwmTimer()* konfigurowania modułu ePWM3.

Moduł ma ustawiony zegar wewnętrzny na częstotliwość SYSCLKOUT/2, czyli 60 MHz/2=30 MHz (okres T_{TBCLK}=33.33 ns).

Okres T_{PWM} generowanego sygnału PWM jest określany poprzez zawartość rejestru TBPRD

$T_{PWM}=(TBPRD+1)\times T_{TBCLK}$

Przełączanie stanu wyjścia wykonywane jest co okres, co daje dodatkowy dział go przez 2.

Czyli generowany przebieg ma wypełnienie 50% i okres

$T_{PRD}=(TBPRD+1)\times T_{TBCLK}\times 2=(10+1)\times 33.33\ ns\times 2$

Na początku działania programu okres sygnału EPWM3A wynosi PWM3_TIMER_MIN=10 co daje okres sygnału T_{PRD}= =733.33 ns (1.363 MHz), gdzie czas trwania poziomu niskiego i wysokiego wynosi 366.66 ns.

Podczas pracy programu okres wzrasta do wartości PWM3_TIMER_MAX = 8000 co daje okres sygnału T_{PRD}= =533.4 µs (1.875 kHz), gdzie czas trwania poziomu niskiego i wysokiego wynosi 266.7 µs

Konfigurowanie modułu eCAP1 do pracy w trybie CAPTURE

W funkcji main() wywoływana jest funkcja *InitECapture()* konfigurowania modułu eCAP1. W pierwszej linii kodu funkcji włączany jest zegar systemowy dla modułu eCAP1.

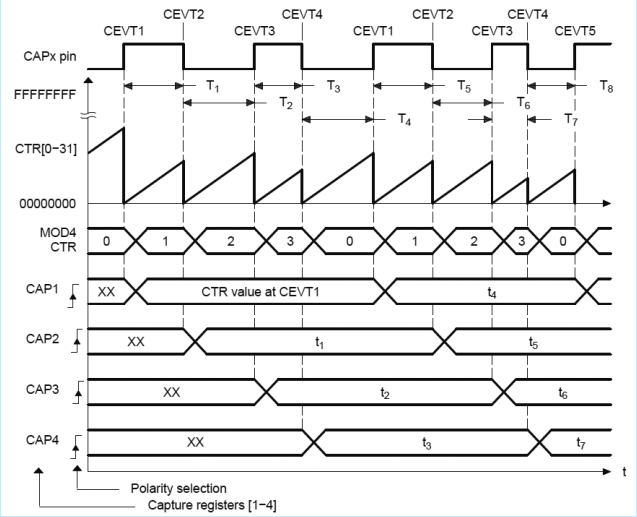
Następnie blokowane jest zgłaszanie przerwań ze wszystkich zdarzeń modułu eCAP1 i kasowane są wszystkie znaczniki przerwań. Dalej blokowany jest wpis do rejestrów CAP1-CAP4 zawartości licznika TSCTR przy zdarzeniu oraz zatrzymany jest (zamrożany) licznik TSCTR (znacznik czasowy – Time Stamp).

Dopiero teraz można przystąpić do konfigurowania rejestrów modułu eCAP1.

Najpierw dokonywany jest wybór sposobu pracy modułu eCAP1 w trybie CAPTURE – jako praca jednokrotna (One-Shot).

Następnie określany jest sposób zatrzymania sekwensera (licznika Mod4) – wybierane jest zdarzenie CEVTx które zostanie obsłużone przed zamrożeniem rejestrów CAPx, czyli przed zatrzymaniem sekwencji. Tutaj wybierane jest zatrzymywanie sekwencji po wpisaniu do rejestru CAP4.

Następne cztery linie kodu określają polaryzację zbocza dla zdarzenia: zdarzenia CEVT1 i CEVT3 są zgła-



Rysunek 2. Przykład pracy modułu eCAP w trybie CAPTURE [6]

szane na zboczu opadającym sygnału, zdarzenia CEVT2 i CEVT4 są zgłaszane na zboczu narastającym sygnału.

Kolejne cztery linie kodu określają pracę w różnicowym trybie czasowym – zawartości licznika TSCTR jest zerowana po wpisie do rejestru CAP1-4.

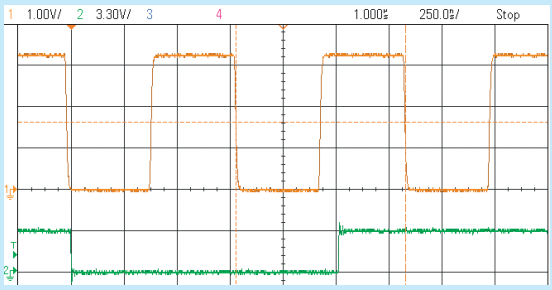
Dalej włączane jest zezwolenie na synchronizację licznika TSCTR w reakcji na sygnał SYNCI lub programowe ustawienie bitu SWSYNC oraz jako wyjściowy sygnał synchronizacji wybierany jest sygnał wejściowy synchronizacji (pass through).

Potem wykonywana jest sekwencja uruchamiania modułu eCAP1. Włączane jest zezwolenie na wpisanie do rejestrów CAP1-CAP4 zawartości licznika TSCTR przy odpowiednim zdarzeniu. Uruchamiany jest licznik TSCTR do pracy swobodnej. Następnie wykonywane jest sterowanie ponownym zaprogramowaniu modułu eCAP1 w następującej sekwencji: zerowanie licznika Mod4, odblokowanie (odmrażanie) licznika Mod4, zezwolenie na ładowanie rejestrów CAPx.

Na koniec włączane jest zezwolenie na zgłaszanie przerwania gdy wystąpi zdarzenie CEVT4, czyli po sekwencji czterech zdarzeń.

Moduł pracuje z wstępnym podziałem (preskaler) cyfrowego sygnału wejściowego podawanego bezpośrednio z wyprowadzenia układu procesorowego ustawionym domyślnie na podział przez jeden.

Na **rysunku 2** jest pokazany przykład skonfigurowania modułu eCAP1 w trybie CAPTURE i z pracą ciągłą. Odwrotnie, niż w przypadku programu *Example_F2802xECap_Capture_Pwm* zdarzenia CEVT1 i CEVT3 wyzwalane są zboczem narastającym a zdarzenia CEVT2 i CEVT4 wyzwalane są zboczem opadającym sygnału wejściowego (bez preskalera). Pozostałe ustawienia są



Rysunek 3. Przebieg sygnału EPWM3A (kanał 1) dla typowego sposobu pracy modułu eCAP

takie same. Do rejestrów CAP1-4 zapisywany jest czas różnicowy (delta) pomiędzy zdarzeniami, np. czas T1 pomiędzy zdarzeniami CEVT1 i CEVT2, wpisywany do rejestru CAP2 w momencie czasowym zdarzenia CEVT2.

Na podstawie zawartości rejestrów CAPn można dla sygnału wejściowego policzyć okres i współczynnik wypełnienia. I tak okres wynosi np. Tp1=t1+t2, a współczynnik wypełnienia DC1=(t1/Tp1)×100% itd.

Procedura obsługi przerwania modułu eCAP1

Przerwanie ECAP1_INT zgłaszane jest przez moduł eCAP1 po wpisaniu nowych wartości do wszystkich rejestrów CAP1-4.

W procedurze *ecap1_isr* obsługi przerwania ECAP1_INT wykonywane jest porównanie zawartości rejestrów CAP2-4 z bieżącą wartością rejestru okresu modułu ePWM3. Należy zauważyć, że zegar taktowania wewnętrznego modułu eCAP1 to SYSCLKOUT a modułu ePWM3 to SYSCLKOUT/2.

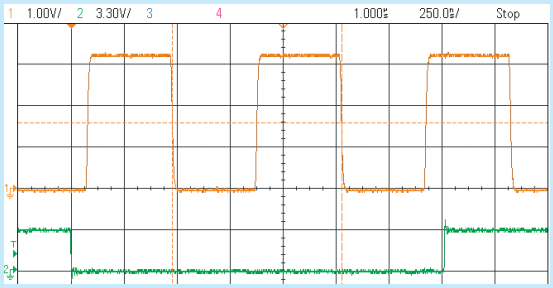
Sygnał na wejściu ECAP1 jest asynchroniczny w stosunku do taktowania procesora. Dlatego najpierw jest synchronizowany z zegarem systemowym SYSCLKOUT. Może to powodować skrócenie lub wydłużenie czasu trwania podsynchronizowanego sygnału wewnętrznego o jeden okres zegara SYSCLKOUT.

Dalej w procedurze jest modyfikowana zawartość rejestru okresu modułu ePWM3. Początkowo jest on ustawiony na wartość minimalną PWM3_TIMER_MIN=10 oraz wskaźnik zmian jest ustawiony na zwiększanie zawartości (*EPwm_TIMER_UP*). Jeśli wartość jest mniejsza od maksimum (PWM3_TIMER_MAX) to wykonywane jest jej zwiększenie. W przeciwnym wypadku rozpoczyna się zmniejszanie okresu modułu ePWM3. Zmiana okresu sygnału wejściowego jest wykonywana tylko po zebraniu kompletu danych w rejestrach CAP1-4 i zatrzymaniu działania modułu eCAP1. Najłatwiej zobaczyć to w działaniu, na wideo „Projekt_ECAP” (w materiałach dodatkowych na załączonym CD/DVD – kanały są zorganizowane tak samo jak na rys. 5) lub po uruchomieniu programu w dalszej części tego ćwiczenia.

Następnie zerowany jest znacznik zgłoszenia żądania obsługi przerwania CEVT4 modułu eCAP1. Włączane jest zezwolenie zgłaszania przerwań przez moduł eCAP1. Ostatnią operacją wykonywaną z modułem eCAP1 jest ponowne „uzbrojenie” (rearm) modułu: zerowany jest wskaźnik Mod4 rejestrów CAP1-4, aktywowana jest praca wskaźnika Mod4 włączana jest praca modułu. Na koniec procedury włączane (przywracane) jest dla grupy 4 w PIE zezwolenie na zgłaszanie przerwania do CPU.

Przykładowy przebieg sygnału EPWM3A dla typowego sposobu pracy modułu eCAP1 jest pokazany na **rysunku 3**. W kanale 2 poziom wysoki sygnalizuje pracę procedury *ecap1_isr* obsługi przerwania ECAP1_INT. Należy zauważyć, że poziom niski w kanale 2 pojawia się później niż wykonywane jest ponowne „uzbrojenie” (rearm) modułu. Pierwsze zbocze opadające określa zawartość rejestru CAP1. Pierwsze zbocze narastające określa zawartość rejestru CAP2. Drugie zbocze opadające określa zawartość rejestru CAP3. Drugie zbocze narastające określa zawartość rejestru CAP4. Następnie zgłaszane jest przerwanie ECAP1_INT.

Jednak praca modułu ePWM3 i modułu eCAP jest asynchroniczna. Dlatego po ponownym „uzbrojeniu”



Rysunek 4. Przebieg sygnału EPWM3A dla opóźnionego wstąpienia zbocza opadającego

(rearm) modułu eCAP1 wystąpienie zbocza opadającego może być opóźnione, jak to jest pokazane na **rysunku 4**.

W przypadku stwierdzenia w procedurze *ecap1_isr* błędu porównania wywoływana jest procedura *Fail*. Procedura ta przerywa działanie programu i wywołuje zatrzymanie debugowe (instrukcja asemblerowa *ESTOP0*). Działa to poprawnie tylko wtedy gdy jest dołączony emulator sprzętowy – wtedy wejście procesora przyjmuje poziom TRST=1. Gdy emulator nie jest dołączony to procesor interpretuje instrukcję asemblerową *ESTOP0* jako instrukcję *NOP*. Powoduje to zakończenie procedury *Fail* i powrót do wykonywania programu *Example_2802xECap_Capture_Pwm*.

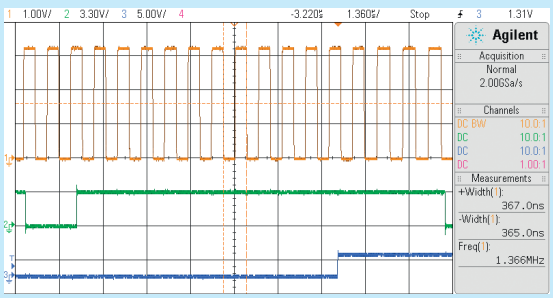
Taka sytuacja jest możliwa, gdyż w projekcie *Example_F2802xECap_Capture_Pwm* jest również zdefiniowana konfiguracja konfiguracji budowania o nazwie FLASH. Procesor Piccolo F28027 zestawu C2000 Piccolo LaunchPad z zaprogramowaną pamięcią Flash może zostać uruchomiony bez dołączonego emulatora.

Dlatego bezpieczniej jest dodać w procedurze *Fail* dodatkową linię kodu z pętlą nieskończoną.

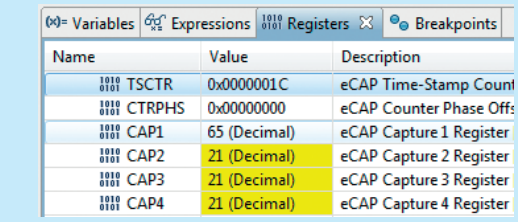
```
void Fail()
{
 asm(" ESTOP0");
 for(;;);
}
```

Działanie programu Example_2802xECap_Capture_Pwm

12. Dołącz sondę jednego kanału oscyloskopu do sygnału EPWM3A (GPIO4).
13. Wykonaj polecenie *Resume*. Przebieg powinien mieć kształt zbliżony do prostokąta. Zaobserwuj na ekranie zmianę okresu sygnału. Zmierz (na ekranie oscyloskopu) okres sygnału dla największego okresu sygnału.
14. Wykonaj polecenie *Suspend*.
15. W procedurze *ecap1_isr* ustaw pułapkę w linii (241) kodu



Rysunek 5. Przebieg sygnału EPWM3A (kanał 1) generowany dla najkrótszego okresu



Rysunek 6. Przykład stanu rejestrów modułu eCAP

EPwm3TimerDirection = EPwm_TIMER_UP;

16. Wykonaj polecenie *Resume*.
17. Działanie programu zostanie zatrzymane na linii 241.

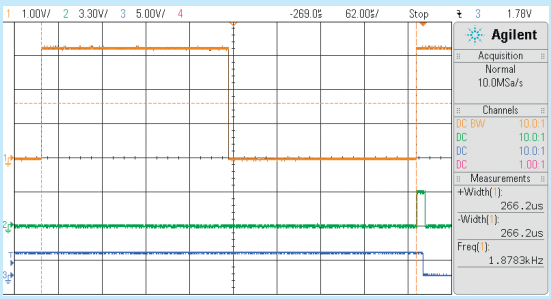
Przykładowy przebieg sygnału EPWM3A dla minimalnego okresu jest pokazany na **rysunku 5**. W kanale 2 poziom wysoki sygnalizuje pracę procedury *ecap1_isr* obsługi przerwania ECAP1_INT. W kanale 3 poziom wysoki sygnalizuje pracę procedury *ecap1_isr* ze zwiększaniem okresu sygnału EPWM3A. Wartości czasu poziomu niskiego i wysokiego sygnału EPWM3A są bardzo zbliżone do ustawionych w trakcie konfigurowania modułu ePWM3. Przy odczytach pomiaru czasu trzeba pamiętać o wpływie ustawienia podstawy czasu oscyloskopu na rozdzielczość (i dokładność) czasową pomiaru. Pewien wpływ na wartości odczytywane przez oscyloskop ma również włączony w kanale tego sygnału filtr 20MHz, który wydłuża czasy zboczy sygnału do 15ns. Wyłączenie filtru powoduje jednak występowanie sporych przerzutów w obrazie przebiegu.

18. Sprawdź zawartość rejestrów CAP2-4 modułu eCAP1. W oknie *Registers* rozwiń strukturę eCAP1.

Zawartość rejestrów CAP2-4 powinna być zbliżona do podwójnej minimalnej zawartości rejestru okresu modułu ePWM3 (**rysunek 6**). Dla zwiększenia czytelności odczytu zawartości zaznacz w oknie *Registers* rejestry CAP1-CAP4 lewym przyciskiem myszy. Kliknij na nie prawym przyciskiem myszy i z menu wybierz *Number Format* → *Decimal*.

19. W procedurze *ecap1_isr* zdejmij pułapkę w linii (241) kodu.
20. Ustaw pułapkę w linii (232) kodu
    EPwm3TimerDirection = EPwm_TIMER_DOWN;
21. Wykonaj polecenie *Resume*. Działanie programu zostanie zatrzymane na linii 232. Przykładowy przebieg sygnału EPWM3A dla maksymalnego okresu jest pokazany na **rysunku 7**.
22. Po zatrzymaniu działania programu na linii 232 sprawdź zawartość rejestrów CAP2-4 modułu eCAP1.

Tym razem zawartość tych rejestrów jest zbliżona do podwójnej maksymalnej zawartości rejestru okresu modułu ePWM3.



Rysunek 7. Przebieg sygnału EPWM3A (kanał 1) generowany dla najdłuższego okresu

Zwróć uwagę, że w obu przypadkach zawartość rejestru CAP1 jest inna (np. większa) niż zawartość rejestrów CAP2-4.

Preskaler sygnału wejściowego

Cyfrowy sygnał wejściowy może być podzielony przez wartość w zakresie od 1 do 62. Jednak biblioteka *diverlib* nie udostępnia tej funkcjonalności. Udostępnia ona jednak typ wyliczany *CAP_Prescale_e* oraz stałą *CAP_ECCTL1_PRESCALE_BITS* w pliku *cap.h* dołączonym już do projektu *Example_F2802xECap_Capture_Pwm*. Dlatego w łatwy sposób można opracować własną funkcję ustawiania preskalera.

Stała *CAP_ECCTL1_PRESCALE_BITS* definiuje pozycję pola *PRESCALE* w rejestrze sterującym ECTL1 modułu eCAP1. Typ wyliczany *CAP_Prescale_e* obejmuje wszystkie ustawienia podziału.

Funkcja *CAP_setCapEvtFltPrescale* do ustawiania preskalera sygnału wejściowego jest pokazana w ramce. Ma ona organizację taką jak inne funkcje obsługi modułu eCAP1 biblioteki *driverlib* znajdujące się w pliku *cap.c* dołączonym do projektu *Example_F2802xECap_Capture_Pwm*.

23. Przełącz się do perspektywy *CCS Edit*. W pliku *Example_2802xECap_Capture_Pwm.c* wstaw kod funkcji *CAP_setCapEvtFltPrescale* (ramka) bezpośrednio za kodem funkcji *main()*.

W funkcji *InitECapture()* wstaw nową linię kodu
CAP_setCapEvtFltPrescale(myCap, CAP_Prescale_By_2);
bezpośrednio przed linią wystartowania licznika modułu
CAP_enableTimestampCounter(myCap); // Start Counter

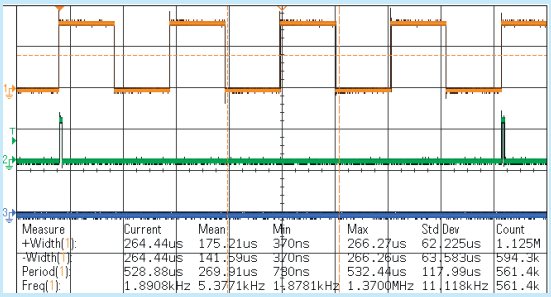
24. W procedurze *ecap1_isr()* zmień w liniach porównania wymnożenie przez 2 (*2±1) na wymnożenie przez 4 (*4±1).
25. Wykonaj samo budowanie projektu (bez ponownego startowanie sesji debugowej). W perspektywie *CCS Edit* kliknij na przycisk *Build*. Nie używaj przycisku *Debug*. Na pytanie czy załadować plik wynikowy kodu przyciśnij przycisk *Yes*. Przełącz się do perspektywy *CCS Debug*.
26. Wykonaj polecenie Resume. Działanie programu zostało zatrzymane w procedurze *Fail*.
27. Sprawdź zawartość rejestrów CAP2-4 modułu eCAP1 w oknie *Registers*.

Obecnie zmierzony czas połowy okresu (CAP2-4) wynosi 43. A w liniach porównania dla TBPRD=10 porównuje się go do wartości 40±1. Wystąpienie problemu jest spowodowane błędnym wzorem zastsowanym w liniach porównania procedury. Czas trwania poziomu niskiego (i wysokiego) sygnału (równy wartości okresu ustawionej dla modułu ePWM3) wynosi TBPRD+1 (zamiast błędnego liczenia TBPRD).

W procedurze *ecap1_isr()* zmień w liniach porównania sposób liczenia czasu na poprawny
(PWM_getPeriod(myPwm1) +1)*4

28. Wykonaj samo budowanie projektu (bez ponownego startowanie sesji debugowej).
29. Przełącz się do perspektywy *CCS Debug*. Wykonaj polecenie Resume.

Obecnie zdarzenia będą generowane w module eCAP z odstępem czasowym dwa razy większym niż w poprzedniej wersji programu. Jednak obraz przebiegu PWM na ekranie oscyloskopu będzie taki sam jak przed modyfikacją kodu. Na **rysunku 8** są pokazane parametry



Rysunek 8. Minimalne i maksymalne wartości okresu przebiegu sygnału EPWM3A

dla minimalnej i maksymalnej wartości okresu przebiegu PWM. Są one bardzo zbliżone do wartości ustawionych w trakcie konfigurowania modułu ePWM3.

Praca modułu eCAP w trybie APWM

Moduł eCAP pracuje w trybie APWM jako jednokanałowy generator sygnału PWM z licznikiem 32-bitowym pracującym w trybie zliczania w górę.

Przy pracy w czasie rzeczywistym stosowane jest podwójne buforowanie. Wtedy rejestr CAP1 pracuje jako aktywny rejestr okresu (APRD), CAP2 jako aktywny rejestr porównania (ACMP), CAP3 jako pośredni (shadow) rejestr okresu, CAP4 jako pośredni rejestr porównania.

Zawartość rejestru porównania ACMP (CAP2) definiuje czas trwania poziomu aktywnego (włączenia) sygnału wyjściowego. Poziom aktywny zaczyna się na początku cyklu (przy początkowej zawartości licznika APRD (CAP1) i trwa aż do uzyskania wartości równej zawartości rejestru porównania ACMP (CAP2). Można ustawiać polaryzację sygnału wyjściowego - wyjście może mieć aktywny poziom wysoki lub niski.

Zastosowanie drugiego projektu: Example_F2802xECap_apwm

30. W perspektywie *CCS Edit* w oknie *TI Resource Explorer* rozwiń drugą pozycję *controlSUITE*. Następnie rozwiń w tym oknie drzewo *controlSUITE* → *device_support* → *f2802x* → *v210* → *f2802x_examples*. Potem kliknij na nazwę wybranego projektu ***Example_F2802xECap_apwm***.

Krok1B: Importowanie projektu Example_F2802xECap_apwm do CCSv5

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

Projekt *Example_F2802xECap_apwm* został zaimportowany z kopiowaniem projektu i pliku ***Example_2802xECap_apwm.c*** do foldera roboczego projektu. W oknie *Project Explorer* został pokazany drugi projekt ustawiony jako aktywny.

Krok2B: Budowanie projektu Example_F2802xECap_apwm

32. W oknie *TI Resource Explorer* kliknij na odnośnik kroku 2.
33. Został zbudowany projekt w konfiguracji budowania o nazwie RAM. Budowanie projektu ***Example_F2802xECap_apwm*** zostało zakończone poprawnie. Został utworzony wynikowy plik binarny

***Example_2802xECap_apwm*.out** (zobacz okno *Console*). Zostały jednak zgłoszone ostrzeżenia (zobacz okno *Problems*). Na razie są one nieistotne.

Krok3B: Definiowanie konfiguracji sprzętowego systemu docelowego

34. W oknie *TI Resource Explorer* kliknij na odnośnik kroku 3. W oknie dialogowym *Debugger Configuration* rozwiń listę wyboru. Wybierz pozycję ***Texas Instruments XDS100v2 USB Emulator***. Kliknij *OK*.

Krok4B: Uruchamianie sesji debugowej dla projektu Example_F2802xECap_apwm

35. W oknie *TI Resource Explorer* kliknij na odnośnik kroku 4.
36. W oknie *Launching Debug Session* kliknij *Yes*.
37. Poczekaj na zakończenie ładowania kodu i pokazanie się okna perspektywy *CCS Debug*.
38. Zapoznaj się z komentarzem na początku pliku *Example_2802xECap_apwm.c*

Konfigurowanie modułu eCAP1 do pracy w trybie APWM

W funkcji *main()* wykonywane jest konfigurowanie wyprowadzenia GPIO5 przypisanego jako wyjście ECAP1 dla modułu eCAP1.

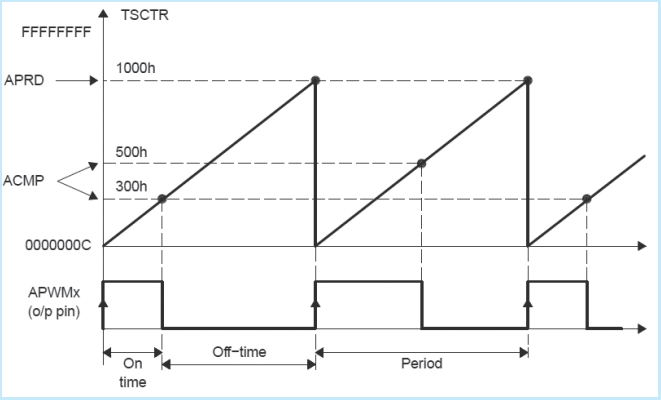
Najpierw włączany jest zegar systemowy dla modułu eCAP1. Następnie włączany jest tryb pracy APWM, ustawiany jest rejestr okresu (CAP1) i rejestr porównania (CAP2). Ostatnia znacząca linia kodu to włączenie pracy licznika TSCTR jako licznik podstawy czasu.

Na **rysunku 9** pokazano przykład skonfigurowania modułu eCAP1 w trybie APWM. Jest on zgodny ze sposobem skonfigurowania w projekcie *Example_F2802xECap_apwm* z wyjątkiem wartości porównania (0x00989680) oraz okresu (0x01312D00).

39. Dołącz sondę jednego kanału oscyloskopu do sygnału ECAP1 (GPIO5).
40. Wykonaj polecenie Resume  .

Przebieg powinien mieć kształt zbliżony do prostokąta o okresie ok. 332.8 ms (ok. 3 Hz).

Henryk A. Kowalski
kowalski@ii.pw.edu.pl
Wideo: Piotr T. Kowalski



Rysunek 9. Generowanie sygnału PWM na wyjściu APWM podczas pracy modułu eCAP w trybie CAPTURE [6]

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 Capture Module (eCAP) Reference Guide [SPRUFZ8.pdf], 03 May 2009
[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/

Funkcja ustawiania preskalera

```
//! \brief      Sets the PRESCALE
value
//! \param[in] capHandle  The
capture (CAP) object handle
//! \param[in] prescale  The
PRESCALE value
void CAP_setCapEvtFltPrescale(CAP_
Handle capHandle, const CAP_
Prescale_e prescale)
{
  CAP_Obj *cap = (CAP_Obj *)
capHandle;
  // clear the bits
  cap->ECCTL1 &= (~CAP_ECCTL1_
PRESCALE_BITS);
  // Set the new value
  cap->ECCTL1 |= prescale;
  return;
} // end of CAP_
setCapEvtFltPrescale() function
```