MSP430 w przykładach (5)

Konfigurowanie Watchdoga

W urządzeniach, które działają bez obsługi człowieka, a zatrzymanie ich pracy może mieć tragiczne konsekwencje (np.: czujniki dymu, detektory nieszczelności instalacji gazowej, systemy alarmowe) zawsze powinien być stosowany układ Watchdog. Wszystkie aktualnie produkowane MSP430 mają wbudowany moduł Watchdog-Timer (WDT), który może pracować w trybie układu Watchdog. Dodatkowo do nóżki #RST/NMI mikrokontrolera można dołączyć zewnętrzny, dodatkowy timer Watchdog. W artykule omówimy pracę układu Watchdog w MSP430.

Układ Watchdog ma za zadanie stać na straży ciągłości pracy mikrokontrolera. W wypadku, gdy mikrokontroler przestanie pracować, zawiesi się, to układ Watchdog generuje sygnał zerujący i wymusza restart mikrokontrolera.

Watchdog jest wbudowany w strukturę mikrokontrolera. Jest to rodzaj timera, który pracuje niezależnie od CPU, zliczając impulsy sygnału zegarowego doprowadzonego do wejścia, od zera do wartości maksymalnej, zdefiniowanej przez programistę. Jeśli wartość licznika osiągnie wartość maksymalną, to z kolejnym taktem sygnału zegarowego licznik wyzeruje się, a na jego wyjściu zostanie wygenerowany sygnał zerujący mikrokontroler.

Czas, pracy Watchdoga oblicza się jako (5.1):

(5.1) tPracy = (mLicznik + 1) / fZegara

gdzie:

tPracy – czas pracy układu Watchdog [s],

fZegara – częstotliwość sygnału zegarowego taktującego układ [Hz].

mLicznik – maksymalna wartość licznika zdefiniowana przez programistę.

Zadaniem programisty jest okresowe zerowanie licznika Watchdoga, co w żargonie programistów nazywa się „poganianiem Watchodoga”. Aby to zrobić, w kodzie programu umieszcza się instrukcje zerowania licznika Watchodga, które wywoływane są cyklicznie, a czas pomiędzy instrukcjami musi być krótszy, niż czas pracy układu Watchdog. Zerowanie licznika ma nie dopuścić do sytuacji, w której licznik Watchodga osiągnie wartość maksymalną i wygeneruje sygnał restartujący mikrokontroler.

Moduł Watchdog-Timer

Schemat blokowy modułu Watchdog-Timer (WDT) wbudowanego w MSP430f1232 ilustruje **rysunek 1**. Moduł jest konfigurowany za pomocą 16-bitowego rejestru WDTCTL. Bardziej znaczący bajt rejestru jest przeznaczono na hasło, mniej znaczący na bity konfiguracyjne. Żeby zmienić bity konfiguracyjne rejestru zawsze trzeba podać hasło. W wypadku dostępu do rejestru bez hasła, bądź z błędnym hasłem, jest wykonywany restart PUC mikrokontrolera. Hasło ma wartość 0x5A (w środowisku IAR definicja WDTPW). Podczas odczytu rejestru bardziej znaczący bajt ma zawsze wartość 0x69, mniej znaczący ilustruje ustawienie bitów konfiguracyjnych. Opis bitów konfiguracyjnych rejestru WDTCTL zamieszczono w materiałach dodatkowych na płycie CD EP1/2013 oraz na serwerze FTP.

Dodatkowe informacje:
W materiałach dodatkowych zamieszczonych na CD-EP1/2013 oraz na serwerze FTP prezentujemy filmy ilustrujące działanie przykładów opisywanych w artykule.
ftp://ep.com.pl, user: 86304, pass: 418ogqs3

Moduł Watchdog Timer, poza podstawowym trybem pracy Watchdog (restart), może dodatkowo pracować w trybie Timer (zegar). O wyborze trybu pracy decyduje ustawienie bitu WDTTMSEL. W trybie pracy jako Watchdog, licznik układu zerujemy (poganiamy Watchdog) ustawiając bit konfiguracyjny WDTCNTCL. W sytuacji, gdy licznik nie zostanie wyzerowany i minie ustawiony czas, to w rejestrze IFG1 zostanie ustawiony bit WDTIFG (flaga przerwania). Dodatkowo, zostanie ustawiony sygnał PUC i wykona się restart mikrokontrolera. W trybie pracy zegara, gdy minie zaprogramowany czas, to w rejestrze IFG1 także zostanie ustawiony bit WDTIFG (flaga przerwania). Nie zostanie jednak ustawiony sygnał PUC i nie wykona się restart mikrokontrolera. Licznik układu Watchdog „przekręci się” i rozpocznie pracę od wartości zero. Moment ustawienia flagi WDTIFG możemy przechwycić w przerwaniu, a fakt, że flaga ustawia się cyklicznie w równych odcinkach czasu odpowiadających cyklowi pracy układu WDT, wykorzystać do odmierzania czasu.

Moduł Watchdog Timer może być taktowany jednym z dwóch wewnętrznych sygnałów zegarowych: ACLK, SMCLK. O wyborze sygnału taktującego decyduje bit TASSELx. Maksymalna wartość licznika definiowana jest przy pomocy bitu WDTISx. Wybrać można jedną spośród 4 wartości: 32767, 8191, 511, oraz 63. W praktyce bardzo często moduł Watchdog Timer taktowany jest sygnałem zegarowym ACLK, o częstotliwości 32768 Hz (kwarc zegarkowy dołączony do źródła LFXT1), a maksymalną wartości licznika ustawiana jest na 32767. Wówczas – zgodnie ze wzorem 5.1 – czas pracy układu wynosi 1 sekundę. Po podzieleniu częstotliwości sygnału ACLK przez 2, 4 lub 8 (konfiguracja w module Basic Clock), czas pracy układu można wydłużyć do 2, 4 lub 8 sekund.

Po starcie mikrokontrolera moduł Watchdog Timer pracuje w trybie restartu. Źródłem sygnału taktującego WDT jest sygnał SMCLK. Częstotliwość sygnału SMCLK jest równa częstotliwości sygnału DCOCLK. Dla mikrokontrolera MSP430f1232 wynosi ona około 740 kHz. Maksymalna wartość licznika wynosi 32767. Zgodnie ze wzorem 5.1, czas pracy układu WDT to około 45 ms. Po takim czasie od startu mikrokontrolera układ Watchdog

Przerwania
W materiałach dodatkowych zamieszczony na płycie CD prezentujemy dokument opisujący działanie systemu przerwań w MSP430 z serii 1xx. W mikrokontrolerze MSP430f1232 z układem Watchdog są związane dwa wektory przerwań. Jeden obsługuje przerwania od modułu Watchdog Timer pracującego w trybie Timer (zegar), natomiast drugi przerwania od nóżki #RST/NMI pracującej w trybie przerwań niemaskowalnych NMI.

wygeneruje sygnał PUC. Mikrokontroler ponownie uruchomi się, a po 45 ms ponownie i ponownie, i tak będzie restartował się do wyłączenia zasilania. Dlatego też na samym początku programu pierwszą rzeczą, którą powinniśmy zrobić, to skonfigurowanie modułu Watchdog Timer.

W zależności od założeń projektu możemy:
- wyłączyć układ WDT,
- rozpocząć sterowanie pracą WDT,
- przełączyć układ WDT w tryb pracy zegara

Zewnętrzny układ Watchdog

Przykład dołączenia zewnętrznego układu Watchdog do mikrokontrolera MSP430 ilustruje **rysunek 2**.

Zewnętrzny układ Watchdog z komunikuje się z MSP430 za pomocą dwóch linii I/O. Jedna z nich jest używana przez mikrokontroler do zerowania zewnętrznego Watchdoga. Za pomocą drugiej układ Watchdog zeruje mikrokontroler.

Sygnał zerujący mikrokontroler jest doprowadzony do linii #RST/NMI. Jej wyzerowanie powoduje wygenerowanie sygnału POR i restart mikrokontrolera. Tryb pracy linii #RST/NMI możemy konfigurować. Po starcie mikrokontrolera, linia pracuje w trybie restartu, ale ustawiając w rejestrze WDTCTL bit WDTNMI linię możemy ustawić w tryb obsługi przerwań NMI. Wówczas, wyzerowanie linii nie spowoduje restartu mikrokontrolera, a w rejestrze IFG1 zostanie ustawiona flaga przerwania niemaskowalnego NMIIFG.

W praktyce, linia #RST/NMI zwykle pracuje w trybie restartu. Jest do niej dołączony zewnętrzny układ Watchdog, albo – jak w wypadku modułu „Komputerek" – przycisk zerowania.

Przykłady

Dalej zaprezentujemy dwa przykłady obsługi modułu Watchdog-Timer. W pierwszym, nazwanym „Symulator latarni morskiej", omówimy działanie modułu WDT pracującego w trybie Watchdog (restart). W drugim – „Sekundnik" – działanie modułu WDT pracującego w trybie Timer (zegar). Oba przykłady należy uruchomić korzystając z modułu „Komputerek". Kody źródłowe zamieszczamy na płycie CD-EP1/2013 i serwerze FTP.

Przykład: „Latarnia morska"

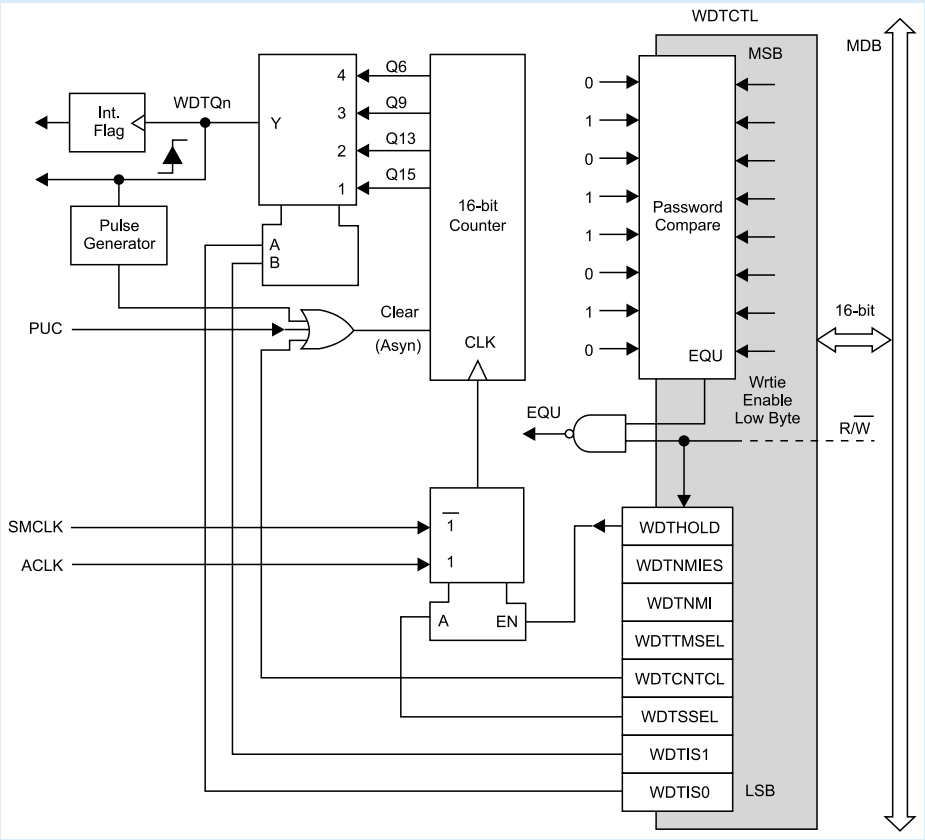
Program „Symulator latarni morskiej" ma na celu przejrzyste i obrazowe zademonstrowanie działania układu Watchdog. Nazwa może być nieco myląca, ponieważ praca programu ma niewiele wspólnego z działaniem prawdziwej latarni morskiej. Program napisano w 3 wersjach. W każdej z nich układ Watchdog jest obsługiwany w inny sposób. W materiałach dodatkowych jest prezentowany program w wersji numer jeden. Aby przejść do drugiej wersji programu, należy linie kodu oznaczone cyfrą 1 opatrzyć komentarzem oraz usunąć komentarz sprzed linii oznaczonych cyfrą 2. Zmiana wersji programu z drugiej na trzecią przebiega analogicznie z tym, że oczywiście cyfry rosną o 1.

Naszą „latarnię morską" uruchamiamy korzystając z modułu „Komputerek". W module należy zworki JP2 i JP3 (linie sterujące diodami LED1, LED2) przełączyć w pozycję LED. Pozostałe należy ustawić w pozycji IO/Off. W pierwszej wersji programu, należy zdjąć zworki JP7 i JP8 dołączające rezonator kwarcowy do źródła zegarowego LFXT1. W wersji drugiej oraz trzeciej, zworki należy przełączyć w pozycję LF.

Działający moduł „Komputerek", pracujący pod kontrolą aplikacji „Latania morska", wysyła sygnały świetlne przy użyciu diody świecącej RGB. Zależnie od konfiguracji modułu, mikrokontroler steruje diodą koloru czerwonego (LED1, P2.3), albo zielonego(LED2, P2.4). Obsługiwana dioda jest włączana raz na sekundę i świeci przez około pół sekundy (jeden błysk na sekundę). Barwa światła informuje o warunkach panujących w porcie. W programie przyjęto, że światło koloru czerwonego powiadamia o złych warunkach w hipotetycznym porcie, a światło koloru zielonego o dobrych. Po włączeniu zasilania modułu (restart POR) mikrokontroler steruje diodą koloru czerwonego, jednak używając przycisku SW1 (wejście P1.1) użytkownik może przełączyć urządzenie na obsługę diody koloru zielonego. Dodatkowo, w programie zasymulowano wystąpienie błędu (zawieszenie się programu). Żeby wywołać błąd, należy przycisnąć przycisk SW2. Przycisk (wejście P1.0) obsługiwany jest w trybie przerwań i jego przyciśnięcie powoduje wywołanie procedury obsługi przerwania. W procedurze umieściliśmy instrukcję *while(1)* (pętla nieskończona). Instrukcja ta zatrzymuje wykonanie programu, a mikrokontroler przestaje sterować diodą LED (latarnia morska przestaje nadawać sygnały świetlne).

W pierwszej wersji programu zatrzymujemy pracę układu Watchdog za pomocą instrukcji WDTCTL=WDTPW+WDTHOLD;. Po naciśnięciu przycisku SW2 i zasymulowaniu wystąpienia błędu program „zawiesza się" i mikrokontroler przestaje sterować diodą LED. Odnosząc się do przykładu latarni morskiej, to latarnia przestaje nadawać sygnały świetlne, a statki zbliżające się do brzegu tracą orientację, mogą wpaść na skały i rozbić się. Prowadzi do analogii, w której brak układu Watchdog może prowadzić do tragedii!.

W drugiej wersji programu, nie zatrzymujemy pracy układu Watchdog. Strażnik pozostaje włączony zmieniamy jednak parametry pracy układu za pomocą instrukcji WDTCTL=WDTPW+WDTSSEL;. Ustawiając w rejestrze konfiguracyjnym WDTCTL bit WDTSSEL zmieniamy taktowanie licznika z sygnału SMCLK na sygnał ACLK (źródło LFXT1, kwarc zegarkowy, częstotliwość 32768 Hz). Maksymalna wartość licznika pozostaje bez zmian i nadal ma wartość 32767. Czas pracy układu Watchdog, wyznaczony zgodnie ze wzorem 5.1, wynosi 1 sekundę. W programie dioda również migocze z częstotliwością 1 Hz, dlatego należy wydłużyć czas pracy Watchdoga. W tym celu w rejestrze konfigurującym generator taktujący Basic Clock ustawiamy bit odpowiedzialny za podzielenie częstotliwości sygnału ACLK przez 2, np. za pomocą instrukcji BCSCTL1|=DIVA_1;. Częstotliwość sygnału zegarowego ACLK po podzieleniu przez 2 wynosi 16384 Hz. Zmiana częstotliwości sygnału spowodowała, że czas pracy układu Watchdog wydłużył się do 2 sekund.



Rysunek 1. Schemat blokowy modułu Watchdog-Timer w MSP430f1232

Po zakończeniu konfigurowania i zdefiniowaniu czasu pracy układu Watchdog, należy w kodzie programu umieścić instrukcję, która będzie zerować licznik układu W omawianym programie należy to zrobić raz po zakończeniu konfiguracji układu oraz cyklicznie w pętli głównej programu: WDTCTL=WDTPW+WDTSSEL+WDTCNTCL;.

Pętla główna programu wykonuje się w czasie około 1 sekundy, czas pracy układu Watchdog to 2 sekundy. Układ Watchdog jest „poganiany" z każdym wykonaniem pętli (raz na sekundę) i nie ma możliwości, aby w prawidłowo działającym programie licznik układu przepełnił się, a Watchdog wygenerował sygnał PUC zerujący mikrokontroler. W wypadku, gdy zostanie wciśnięty przycisk SW2 i zasymulowany błąd, to rejestr Watchdog przestanie być aktualizowany, ponieważ nie wykona się pętla główna programu. Minie czas pracy układu Watchdog (2 sekundy), licznik przepełni się i układ Watchdog wygeneruje sygnał PUC powodując restart CPU. Mikrokontroler ponownie uruchomi się i będzie sterował diodą LED.

Obsługa układu Watchdog zapewnia ciągłość pracy urządzenia. Odnosząc się do przykładu z „latarnią morską", gdy program sterujący latarnią zawiesi się, to układ Watchdog ponownie uruchomi mikrokontroler i latarnia będzie nadawała sygnały świetlne. Sytuacja z pierwszej wersji programu, w której latarnia przestała nadawać sygnały świetlne nie powtórzy się. Teraz ciągłość pracy latarni będzie zapewniona. Sternicy statków bez trudu odnajdą ląd i drogę do portu, nie stracą orientacji, nie wpadną na skały i nie rozbiją się.

W przypadku, gdy latarnik stwierdzi, że w porcie panują dobre warunki i używając przycisku SW1 zmieni nadawane sygnały na zielone, a podczas tej operacji program zawiesi się, to moduł Watchdog wymusi restart mikrokontrolera, a po uruchomieniu latarnia będzie nadawała sygnały świetlne koloru czerwonego. Dzieje się tak, ponieważ po starcie programu zerujemy zmienną globalną fWarunkiWPorcie, w której jest przechowywana informacja o kolorze świecenia (0 – kolor czerwony, 1 – kolor zielony). Nawet gdyby zmienna nie była przez nas zerowana, to bez specjalnego sposobu zadeklarowania, zostałaby ona wyzerowana automatycznie przez moduł inicjujący kompilatora języka C (po starcie mikrokontrolera zmienne globalne inicjowane są na wartość zero).

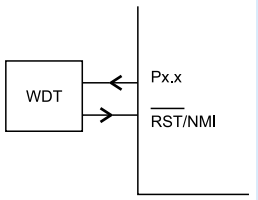
W trzeciej wersji programu, po starcie nie zerujemy zmiennej globalnej fWarunkiWPorcie, a przed deklaracją zmiennej umieszczono dyrektywę __no_init. Jej dodanie przed deklaracją zmiennej sprawia, że po stracie mikrokontrolera zmienna nie jest automatycznie zerowana przez moduł inicjujący języka C. W programie głównym sprawdzamy, czy mikrokontroler uruchomił się w wyniku restartu spowodowanego przez układ Watchdog (ustawiona flaga WDTIFG), czy też z innej przyczyny (np.: włączenie zasilania „Komputerka"):

```
if(IFG1 & WDTIFG == 1)
  IFG1 &=~ WDTIFG; else fWarunkiWPorcie = 0;
```

Jeśli start mikrokontrolera wymusił restart od układu Watchdog, to zerujemy flagę WDTIFG (flaga nie jest zerowana automatycznie), w przeciwnym wypadku inicjujemy zmienną fWarunkiWPorcie (w naszym wypadku przypisujemy jej wartość 0). Teraz, jeśli zostanie wciśnięty przycisk SW2, program zawiesi się, a układ Watchdog wymusi restart mikrokontrolera, to po starcie CPU pamięć SRAM nie zostanie zmodyfikowana (restart PUC nie czyści pamięci SRAM mikrokontrolera, aby umożliwić wznowienie pracy przerwanej aplikacji). Zmienna globalna fWarunkiWPorcie będzie miała wartość sprzed restartu (dyrektywa __no_init – brak automatycznej inicjalizacji). Odnosząc się do przykładu latarni morskiej, jeśli latarnia nadawała sygnały świetlne koloru zielonego (latarnik uznał, że w porcie są dobre warunki i ustawił nadawanie zielonego światła) i program sterujący zawiesił się, a Watchdog wymusił restart urządzenia, to po ponownym uruchomieniu latarnia nadal będzie nadawała sygnały świetlne koloru zielonego. Identycznie jak w drugiej wersji oprogramowania, układ Watchdog zapewni latarni ciągłość pracy, ale dodatkowo w wypadku restartu na skutek zadziałania układu Watchdog, latarnia nie utraci konfiguracji.

Przykład: „Sekundnik"

W przykładzie „Sekundnik" zademonstrujemy, w jaki sposób modułu Watchdog Timer ustawić w tryb pracy jako Timer (zegar) i użyć do odmierzania czasu.



Rysunek 2. Podłączenie układu Watchdog do mikrokontrolera MSP430

Korzystając z modułu będziemy odmierzali 1-sekundowe odcinki czasu, a upływające sekundy będą wyświetlane na ekranie LCD. Pomiędzy pomiarami MSP430f1232 będzie przebywał w trybie uśpienia LPM3.

Przed rozpoczęciem pracy z modułem „Komputerek", zworki JP7 i JP8 dołączające rezonator kwarcowy do źródła zegarowego LFXT1, należy ustawić w pozycji LF. Pozostałe zworki układu należy ustawić w pozycji IO/Off, a w złączu szpilkowym Dis1 zamontować wyświetlacz LCD.

Kod programu „Sekundnik" zamieszczono na płycie CD-EP1/2013 i serwerze FTP. W pierwszych liniach programu dołączane są pliki nagłówkowe. Następnie jest deklarowana zmienna globalna zawierająca licznik sekund lSekund. Zmienna jest typu unsigned long (32 bity) i może przyjmować wartości od zera do ponad 4 miliardów. Ponieważ w programie zmienna jest inkrementowana jest raz na sekundę, to wartość maksymalną osiągnie po około 140 latach.

W programie głównym, za pomocą polecenia WDTCTL=WDTPW+WDTSSEL+WDTTMSEL+WDTCNTCL;, definiujemy parametry pracy modułu Watchdog Timer. Moduł WDT jest przełączany z trybu pracy Watchdog (restart) w tryb Timer (zegar) (bit WDTMSEL). Wydawana jest instrukcja zerująca licznik (bit WDTCTL) oraz definiowane źródło sygnału taktującego licznik (bit WDTSSEL). Następnie bity konfiguracyjne wraz z hasłem zapisu (bit WDTPW) wysyłane są do rejestru konfiguracyjnego WDTCTL. Po ustawieniu bitów w rejestrze konfiguracyjnym, moduł Watchdog Timer rozpoczyna pracę w trybie Timer (zegar). Licznik układu jest taktowany sygnałem zegarowym ACLK o częstotliwości 32768 Hz (źródło LFXT1, kwarc zegarkowy), maksymalna wartość licznika wynosi 32767 (wyzerowany bit WDTISx), a czas pracy układu to 1 sekunda (zgodnie z wzorem 5.1). Potem konfigurowane są linie I/O mikrokontrolera, definiowane parametry pracy wyświetlacza LCD, a w pierwszym wierszu ekranu jest wyświetlany komunikat *„Sekundy:"*.

Teraz jest włączane źródło przerwania od przepełnienia licznika modułu Watchdog Timer oraz obsługa przerwań maskowalnych mikrokontrolera:

```
IE1 |= WDTIE;
__bis_SR_register(GIE);
```

W pętli głównej programu, wartość zmiennej licznik sekund lSekund jest przekształcana na ciąg znaków oraz wyświetlana w drugim wierszu ekranu. Następnie, mikrokontroler jest wprowadzany w tryb uśpienia LPM3. Będąc w uśpieniu, sam mikrokontroler pobiera prąd o natężeniu 0,8 μA, a cały „Komputerek" – plus prąd pobierany przez wyświetlacz LCD. Sygnały zegarowe SMCLK i MCLK są wyłączone. Wyłączona jest także jednostka centralna CPU oraz wewnętrzny generator taktujący DCO. Jedynym aktywnym sygnałem zegarowym jest ACLK. Ponieważ licznik modułu WDT jest taktowany sygnałem ACLK, to są zliczane impulsy licznika. W momencie, gdy licznik osiągnie wartość maksymalną, to z kolejnym taktem zegara licznik przepełni się i zacznie zliczać od wartości zero. W chwili przepełnienia licznika, w rejestrze IFG1 zostanie ustawiona flaga WDTIFG. Ponieważ włączyliśmy źródło przerwania od przepełnienia licznika, to wraz z ustawieniem flagi WDTIFG zostanie wywołana procedura obsługi przerwania dla układu WDT.

```
#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
  ++lSekund;
   __bic_SR_register_on_exit(CPUOFF + SCG0
+ SCG1);
}
```

W procedurze inkrementujemy wartość zmiennej licznik sekund lSekund oraz wywołujemy instrukcję __bic_SR_register_on_exit(…), która powoduje, że po wyjściu z procedury obsługi przerwania mikrokontroler nie wraca do trybu uśpienia LPM3, ale rozpoczyna wykonanie kolejnych instrukcji. W omawianym programie są to instrukcje pętli głównej. Wówczas „nowa", zwiększona o jeden wartość zmiennej licznika sekund jest przekształcana na ciąg znaków alfanumerycznych i wyświetlana na ekranie LCD. Następnie, mikrokontroler ponownie jest wprowadzany jest w tryb uśpienia LPM3. Gdy minie czas pracy układu Watchdog (1 sekunda), to licznik układu WDT ponownie przepełni się, zostanie ustawiona flaga przerwania WDTIFG i rozpocznie się wykonanie procedury obsługi przerwania. Cykl pracy programu powtórzy się i zostaną odmierzone kolejne sekundy.

Łukasz Krysiewicz, EP

Analiza błędów oprogramowania

W wypadku, gdy w urządzeniu z włączonym układem Watchdog (wewnętrzny/zewnętrzny) wystąpi błąd, program zawiesi się, a układ Watchdog wykona restart mikrokontrolera, to korzystając z przerwań (Timer / NMI) można wykryć miejsce, w którym jest błąd w oprogramowaniu. Podczas pracy z zewnętrznym układem Watchdog dołączonym do nóżki #RST/NMI, należy nóżkę ustawić w tryb pracy przerwania NMI oraz zaimplementować procedurę obsługi przerwania NMI. W wypadku pracy z wewnętrznym modułem Watchdog Timer pracującym w trybie Watchdog (restart), układ należy przestawić w tryb pracy Timer (zegar) oraz zaimplementować procedurę obsługi przerwania. W obu wypadkach trzeba zaprogramować urządzenie, ustawić w procedurze obsługi przerwania pułapkę sprzętową (Breakpoint) oraz uruchomić emulator. W momencie, gdy wystąpi błąd i zadziała układ Watchdog (poziom niski na nóżce #RST/NMI lub przepełnienie licznika Watchdog Timer), to rozpocznie się wykonywanie procedury obsługi przerwania, a program zatrzyma się na pułapce sprzętowej. Wówczas w opcjach oprogramowania IAR (*View -> Call Stack*) można podejrzeć dane odłożone na stosie programowym. Ostania odłożona wartość wskaże miejsce w programie, z którego program skoczył do procedury obsługi przerwania. Jest to miejsce, w którym program zawiesił się i wystąpił potencjalny błąd w programie.

Pracując z układem Watchdog należy również pamiętać, że jego użycie uniemożliwia emulowanie projektu, ponieważ po zatrzymaniu programu na pułapce sprzętowej układ Watchdog zeruje mikrokontroler. Z tego powodu na etapie tworzenia i testowania oprogramowania układ Watchdog (wewnętrzny/zewnętrzny) powinien być wyłączony. Dopiero w ukończonym projekcie należy aktywować pracę układu Watchdoga i zaimplementować jego obsługę.

PUC i POR

Sygnały PUC, oraz POR zerują mikrokontroler. Różnica pomiędzy nimi jest taka, że restart wymuszony za pomocą PUC nie czyści pamięci SRAM mikrokontrolera. W materiałach dodatkowych na CD/FTP zamieszczono dokument opisujący działanie układu zerowania w MSP430 z serii 1xx.