Projektowanie wbudowanych systemów sterowania odpornych na zakłócenia

Magdalena Szymczyk

Projektowanie niezawodnego, nieczułego na zakłócenia, wbudowanego systemu sterowania jest jednym z podstawowych aspektów tworzenia takiego systemu. Artykuł przybliża ideę systemu monitorującego pracę systemu wbudowanego. Określa podstawowe założenia i wymagania dotyczące takiego mechanizmu, opisuje możliwe rozwiązania pewnych problemów.

Pojawienie się mikroprocesorów zrewolucjonizowało układy sterowania. Zbieranie danych z różnego rodzaju czujników, skomplikowane przetwarzanie informacji, wypracowanie odpowiedniego sygnału sterowania, optymalizacja wydajności zamkniętego układu sterowania – nie byłyby możliwe bez obecności układu mikroprocesorowego. Mikroprocesory są obecne niemal we wszystkich obszarach naszego życia: od urządzeń domowego użytku, samochodów po samoloty i próbniki kosmiczne. Od ich poprawnego działania niejednokrotnie zależy nasze życie (rozruszniki serca, układy podtrzymujące podstawowe funkcje życiowe, czy w większej skali – bezpieczne działanie elektrowni atomowych). Są to zazwyczaj dedykowane układy mikroprocesorowe, wykonujące specyficzne działania.

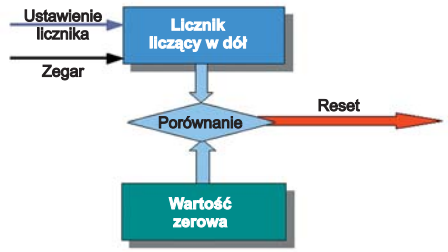
W odróżnieniu od tradycyjnych systemów komputerowych (bazujących na procesorach ogólnego zastosowania jak RISC czy CISC), systemy wbudowane mają różnorodne ograniczenia począwszy od wielkości, poprzez podstawowe parametry charakteryzujące układy elektroniczne (dostępna pamięć, zużycie energii, ograniczenia związane z zasobami systemowymi), na kosztach skończywszy. Nie należy zapominać o tym, że układy te pracują często w ekstremalnych warunkach środowiska zewnętrznego, co sprzyja powstawaniu błędów w ich działaniu. Skutki błędów softwarowych w systemach wbudowanych są znacznie poważniejsze niż dla systemów typu desktop. Doskonale pamiętamy histerię roku 2000, kiedy to ogromne koszty zostały poniesione na poprawianie błędu formatu daty w systemach wbudowanych. Znane są też tragiczne konsekwencje błędnie działającego urządzenia do naświetlań promieniowaniem X [3]. Systemy wbudowane znacznie lepiej tolerują błędnie działający program niż układy typu desktop, nie wynika to jednak z faktu, że oprogramowanie nigdy nie zawodzi, lecz z obecności mechanizmu umożliwiającego powrót do normalnej pracy, gdy z jakichś przyczyn oprogramowanie źle funkcjonuje.

Spektakularnym dowodem, że taki mechanizm zwany „watchdog" (skrót WDT) jest konieczny przy budowaniu odpornych na błędy systemów wbudowanych, jest misja Pathfinder na Marsa. Oprogramowanie systemu zawiesiło się, lecz dzięki obecności odpowiednio zaprojektowanego mechanizmu, który zajmuje się detekcją nieprawidłowego zachowania się oprogramowania, możliwe było zresetowanie całego systemu oraz wgranie nowego, poprawnie działającego oprogramowania. Można powiedzieć, że tego typu rozwiązane jest ostatnią linią obrony przed błędami, gdy wszystkie inne metody zawiodą. Mechanizm WDT znany jest od lat i stosowany na przykład w programowalnych sterownikach logicznych (PLC). Tylko niektóre procesory znanych producentów są zabezpieczone przed przekłamaniami na poziomie bitów (spowodowanymi choćby promieniowaniem kosmicznym), przykładem jest McKinley [5] czy Itanium 2 Intela [5]. Jednakże większość systemów wbudowanych nie ma możliwości resetowania systemu z zewnątrz, a czasami zdarzają się przypadki nietypowego zachowania, co wprawia w stan zdziwienia lub nawet przerażenia ich użytkowników. Można zatem powiedzieć, że każdy system wbudowany powinien zostać wzbogacony o taki mechanizm.

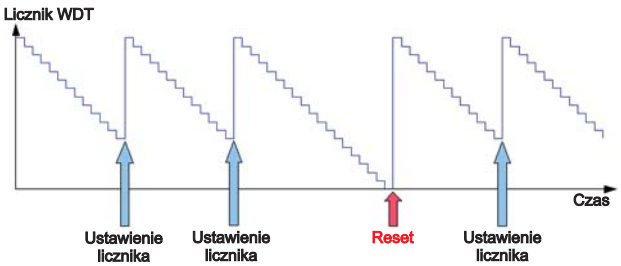
Działający system wbudowany może w pewnym momencie zawiesić się, co często zdarza się, gdy taki system umieszczony na przykład w próbniku kosmicznym nie jest dostępny bezpośrednio dla operatora kontrolującego jego pracę. W innych przypadkach, szybkość działania operatora może być zbyt mała w stosunku do ograniczeń czasowych nałożonych na system. Rozwiązaniem jest zatem zastosowanie układu elektronicznego, który by automatycznie wykrywał anomalie oprogramowania i resetował procesor. Mechanizmem takim jest WDT. Można wyróżnić następujące rodzaje mechanizmu WDT: wewnętrzny, zewnętrzny, śledzący zmienne stanu programu, resetujący lub nieresetujący zewnętrzne urządzenia. WDT może charakteryzować się różnym stopniem złożoności, a w swej najprostszej wersji jest to zwykły licznik zliczający w dół, od pewnej wartości początkowej do zera. Aplikacja systemu wbudowanego ma odpowiednio dobraną wartość początkową licznika i periodycznie ustawia na nowo taką wartość. Jeżeli licznik osiągnie wartość zero zanim oprogramowanie go wyzeruje, to taka sytuacja oznacza niepoprawnie działający program i konieczność wysłania sygnału resetującego procesor. Procesor jak

dr inż. Magdalena Szymczyk
– Katedra Automatyki Wydział Elektrotechniki, Automatyki, Informatyki i Elektrotechniki AGH

i oprogramowanie zachowają się po tym sygnale tak, jakby operator wyłączył i włączył zasilanie. Na rys. 1 przedstawiono typowe rozwiązanie, a na rys. 2 pokazano jego przebiegi czasowe. Mechanizm WDT może być wbudowanym elementem procesora lub układem działającym niezależnie.



Rys. 1. Koncepcja mechanizmu WDT [4]



Rys. 2. Diagram czasowy dla mechanizmu WDT z rys. 1

Idea mechanizmu WDT

Załóżmy, że w pewnym systemie wbudowanym oprogramowanie działa w zamkniętej pętli i jeden przebieg trwa przeciętnie około 20 ms, a w najgorszym przypadku 40 ms. Watchdog przyłączony jest do linii sygnalizującej przerwanie o wysokim priorytecie w systemie, np. RESET lub do niemaskowalnego przerwania (ang. *non-maskable interrupt* – NMI). Po włączeniu, WDT odczekuje 60 ms, a następnie wysyła sygnał RESET do procesora, powodując zresetowanie całego systemu. Jedynym sposobem, aby zapobiec tej sytuacji, jest poinformowanie WDT o poprawnej pracy przez wysłanie przez system główny przed upływem 60 ms impulsu (tzw. sygnału życia), który ponownie spowoduje start w odliczaniu 60-milisekundowego interwału. Jest to ostatnia operacja wykonywana przed przejściem do nowej iteracji pętli. Według Niella Murphy [6] przed zresetowaniem licznika należy wykonać pewne dodatkowe operacje zabezpieczające, do których można zaliczyć np. sprawdzenie głębokości stosu, ustalenie liczby zaalokowanych buforów czy ustalenie stanu aktualnie wykorzystywanych mechanicznych elementów systemu. Dodatkowo powinno się ustawiać flagi sygnalizujące prawidłowe wykonanie pewnych części kodu. Przed wyzerowaniem licznika dokonuje się analizy ustawienia flag – jeżeli wszystko przebiegło poprawnie, to licznik zaczyna odliczać następny interwał, w przeciwnym przypadku system przechodzi w stan błędnej pracy.

Jeżeli sytuacja rozwija się niepomyślnie, ze względu na problem sprzętowy lub programowy, licznik czasowy przekracza swój limit i generowany jest sygnał resetu systemu. Niestety, tak prosty mechanizm nie zawsze wystarcza do prawidłowego zainicjowania poprawnej pracy systemu.

Wewnętrzny WDT

Wewnętrzny WDT to taki, który jest wbudowany w układ scalony procesora. Wysoko zintegrowane procesory zawierają wiele dodatkowych układów, w których z kolei często wbudowane są proste mechanizmy WDT (np. Toshiba TPM96141); elementy te generują NMI, gdy ich licznik przekroczy limit czasu, co nie zawsze jest rozwiązaniem najlepszym [5]. Korzystne jest, gdy po wygenerowaniu przerwania NMI, system odczeka chwilę i dopiero potem wykonywany jest jego reset (czego procesor Toshiby nie potrafi zrobić). Zatem, aby WDT pracował poprawnie konieczny jest sprzętowy reset umożliwiający procesorowi wznowienie poprawnej pracy. Ustawienie samego licznika instrukcji (rejestr PC) może nie zainicjować poprawnej pracy CPU. Dobrą cechą WDT jest to, że dopiero dwie, bezpośrednio następujące po sobie operacje zapisu danych o wartościach np. 0x55 i 0xaa, mogą dezaktywować timer, zatem przypadkowe operacje pisania programu głównego z małym prawdopodobieństwem mogą wyłączyć ten mechanizm. Ze szczególną ostrożnością należy podchodzić do takich WDT, których rejestry sterujące mogą zostać zmodyfikowane w trakcie wykonywania programu, a to grozi wyłączeniem watchdoga. Jeżeli przekroczenie limitu czasu WDT nie powoduje ustawienia odpowiedniego stanu na odpowiednim wyprowadzeniu procesora, to konieczne jest dołożenie sprzętu resetującego każde urządzenie zewnętrzne. Bez tego taki reset przywróci do życia tylko CPU, lecz niepoprawnie działające peryferia spowodują, że system będzie dalej pracował niewłaściwie.

Zewnętrzne WDT

Część urządzeń nadzorujących pracę procesora i zarządzających sygnałem resetu ma wbudowane mechanizmy WDT. Nie wszystkie jednak zawsze zadziałają poprawnie (np. chip TI UCC3946, SMT czy też Maxim MAX 823), dlatego należy być świadomym ich zalet i wad [5]. Dobrze jest, gdy WDT działają w pewnym oknie czasowym. Oznacza to, że sygnał świadczący o poprawnej pracy procesora jest przyjmowany w ściśle określonym wycinku czasu i jest mało prawdopodobne, że błędnie działający program prześle wtedy prawidłowo dwie ściśle określone wartości, co z kolei spowoduje wyzerowanie licznika watchdoga.

Cechy dobrego WDT

Podstawowym wymaganiem dla systemu sprawdzającego poprawność pracy systemu wbudowanego jest konieczność jego dalszej pracy, niezależnie od tego, co stało się z kodem głównego systemu, a w konsekwencji i z współpracującymi urządzeniami zewnętrznymi.

Gdyby błędnie działający kod modyfikował kod czy rejestry watchdoga, to taki system nie miałby szans na powrót do normalnej pracy. Stworzenie nawet prostego mechanizmu WDT nie jest zatem zadaniem trywialnym. Mechanizm taki nie powinien robić żadnych założeń co do aktualnego stanu oprogramowania czy sprzętu. Jeżeli coś poszło źle, to jego zadaniem jest przywrócić system do normalnej pracy lub ewentualnie wprowadzić w stan bezpiecznej pracy. Liczba bitów licznika jest zależna od rozrzutu czasu wykonania poszczególnych operacji w pętli głównej. Watchdog powinien także dawać możliwość zapisania pewnych informacji, które umożliwiłyby później odtworzenie przyczyny błędu. Dobry mechanizm WDT jest niezależny od systemu głównego. Jest jeden poprawny sposób interwencji WDT, a mianowicie wysłanie sygnału resetu do procesora oraz do wszystkich podłączonych urządzeń zewnętrznych, a nie wysłanie tylko przerwania NMI. W niektórych rozwiązaniach stosuje się sygnał NMI, aby zapisać informacje pomocne do odtworzenia przyczyny problemu, jak i przesłać wiadomość do urządzeń zewnętrznych o tym, że CPU wstrzymuje pracę na krótką chwilę. Natychmiast po tych operacjach wysyłany jest sygnał resetu. Czas zwłoki pomiędzy stwierdzeniem błędnego działania systemu wbudowanego a odpowiedzią watchdoga powinien być jak najkrótszy. Należy pamiętać, że system wbudowany powinien pozostać po uszkodzeniu w takim modzie pracy, aby był nieszkodliwy dla otoczenia, zatem sprzęt powinien zostać przywrócony do życia lub wprowadzony w stan bezpiecznej pracy. Zegar licznika powinien być niezależny od zegara systemu wbudowanego, a zatem każdy WDT wbudowany w CPU systemu głównego nie jest bezpieczny. Tworząc watchdog trzeba być pewnym, że żadne okoliczności związane z pracą oprogramowania systemu wbudowanego nie spowodują przeprogramowania WDT (jego rejestrów, pinów itp.). Zatem sposób dostępu do jego elementów wewnętrznych powinien być ustawiony na status „write protected". Inna możliwość, to modyfikacja rejestrów WDT poprzez poprawną identyfikację hasła, a tym hasłem może być sekwencja dwóch określonych wartości. Prawidłowo zbudowany mechanizm WDT monitoruje pracę całego systemu, a nie tylko przyjmuje sygnały („jest żywy" – w domyśle procesor) pochodzące z pętli głównej; może okazać się, że jest więcej procesów wysyłających te sygnały, a któryś aktualnie został zawieszony. Użytkownik powinien być poinformowany o fakcie resetu systemu (świecąca dioda, komunikat).

Opracowywany system wbudowany często w fazie projektowej współpracuje z debuggerem. Mechanizm WDT działa wtedy niezależnie od pozostałej części systemu i w momencie zatrzymania pracy systemu, czas odliczany jest dalej i po przekroczeniu limitu system jest oczywiście resetowany. Dobrze zatem jest mieć możliwość włączania i wyłączania WDT. Gdy WDT jest na zewnątrz CPU, to prostym rozwiązaniem jest umieszczenie zworki na płycie, aby można było rozłączyć połączenie między CPU a licznikiem w trakcie debuggowania. Na poziomie oprogramowania dobrze jest wykorzystać kompilację warunkową, aby usunąć kod związany z WDT.

W przypadku tworzenia wewnętrznego WDT (dla bardzo tanich systemów wbudowanych) istnieje kilka praktycznych rad dających możliwość tworzenia bardziej niezawodnego rozwiązania opartego w głównej mierze na rozwiązaniach programistycznych. Dobrym zwyczajem jest tworzenie tzw. punktów kontrolnych w programie watchdoga, które modyfikują pewne zmienne w programie pętli głównej. Gdy wartość sprawdzana w watchdogu różni się od tej w programie, następuje wzbudzenie WDT, w przeciwnym razie wysyłany jest sygnał życia.

Do budowy zewnętrznych WDT można użyć prostych mikroprocesorów, aby system monitorujący pracę systemu wbudowanego miał niezależny zegar, własną pamięć i wbudowane liczniki. Nie wydaje się sensowne wykorzystanie całego komputera do budowy WDT innego systemu, choć można użyć procesorów jako monitorów poprawnej pracy pozostałych, np. w systemie wieloprocesorowym.

Wnioski

Coraz większa liczba systemów wbudowanych stosowanych w urządzeniach o znaczeniu krytycznym ze względu na bezpieczeństwo powoduje, że wymagania dotyczące ich odporności na uszkodzenia stale rosną. Dość znaczącym elementem tego typu aplikacji staje się monitor, który obserwuje sposób wykonywania programu oraz stan pracy systemu po błędzie. Prosty licznik stosowany jest już od lat, lecz współczesne rozwiązania mechanizmu WDT stają się coraz bardziej odporne na błędy.

Żaden watchdog nie jest idealny, ale nawet prosty może wyłapać 99 % błędnych działań kodu programu, a mnożąc pozostałą wartość przez prawdopodobieństwo błędnie działającego kodu (bardzo małe) – otrzymujemy wartość bliską zeru.

Bibliografia

1. Baranowski R.: Mikrokontrolery AVR ATmega w praktyce, 2005, BTC.
2. Barr M.: Introduction to Watchdog Timers, Embedded Systems Design, 2001, http://www.embedded.com/columns/beginerscorner/9900324?_requestid=301900.
3. Berger A.: Embedded System Design, 2002,CMP Books.
4. Chakravarty S., Tomar R., Arora M.: Need a watchdog for improved system fault tolerance? Freescale Semiconductor, CommsDesign , 2008 http://www.commsdesign.com/showArticle.jhtml;jsessionid=G0N4FX04LDO5QQSNDLRSKHSCJUNN2JVN?articleID=211600055.
5. Labrosse J. (et al.): Embedded Software, 2008, Elsevier.
6. Murphy N.: Watchdog Timers, Embedded Systems Design, 2003, http://www.embedded.com/columns/beginerscorner/9900324?_requestid=301900.