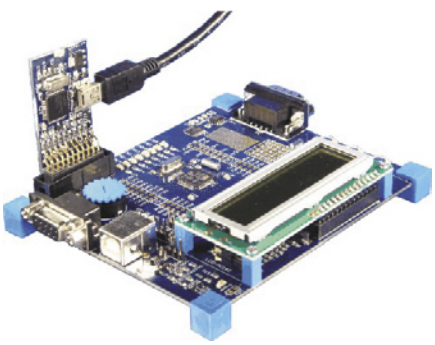
„Śledzenie" ARM-ów w Realu

Serial Wire Viewer

Możliwości wbudowanych we współczesne procesory mechanizmów debugowania są niezwykle istotne dla szybkiego i sprawnego prowadzenia prac nad projektem. Np. procesory ARM implementują wiele zaawansowanych technologii, które zwiększają możliwości wykrywania błędów i testowania oprogramowania w układzie docelowym, i to w czasie rzeczywistym. Najnowszą jest technologia SWV (Serial Wire Viewer), umożliwiająca rdzeniowi procesora wysyłanie „na zewnątrz" w czasie rzeczywistym informacji o stanie mikrokontrolera (Real Time Trace) za pomocą szeregowego, jednoliniowego portu SWO (Serial Wire Output). W artykule przedstawiono dotychczas stosowane mechanizmy debugowania oraz sposób wykorzystania technologii SWV w procesorach z rdzeniem Cortex-M3.

Celem działalności firmy ARM jest opracowanie architektury zaawansowanych, efektywnych i bogato wyposażonych mikrokontrolerów z wbudowanymi odpowiednimi i użytecznymi mechanizmami wspomagającymi uruchamianie aplikacji. Jest to zbieżne z zapotrzebowaniem projektantów na narzędzia umożliwiające testowanie i wykrywanie błędów w sposób jak najbardziej efektywny. Z tego powodu firma ARM rozbudowała moduł debugera EmbeddedICE stosowany dotychczas w rdzeniach ARM7, ARM9, ARM11. Nowe rozwiązanie o nazwie CoreSight oferowane w ARM9, ARM11 i mikrokontrolerach Cortex, łączy w sobie debuger i pamięć śladu. W artykule przedstawiono w jaki sposób na przestrzeni lat w technice mikroprocesorowej możliwości debugowania początkowo rosły, by później ulec wręcz ograniczeniu. Pokazano również, koncentrując się na technice wykorzystania SWV, że firma ARM przywróciła te możliwości w swoich mikrokontrolerach, czyniąc z wbudowanych mechanizmów debugera podstawowy argument za stosowaniem ARM'ów i Cortex'ów.



Fot. 1: Adapter USB-JTAG UlinkME firmy Keil podłączony do złącza JTAG na płytce ewaluacyjnej MCBSTM32

Stare, dobre czasy

W latach 70, gdy korzystano z procesora Intel 8080, to przy uruchamianiu oprogramowania były do dyspozycji działające na innych komputerach symulatory listy instrukcji, często napisane przez samego programistę. Do testowania w układzie docelowym i wykrywania „bug'ów" stosowano metodę „prób i błędów", czyli po prostu po zaprogramowaniu pamięci ROM stwierdzano „działa", czy „nie działa". Później zaczęto stosować emulatory pamięci ROM, które znacznie ułatwiły pracę.

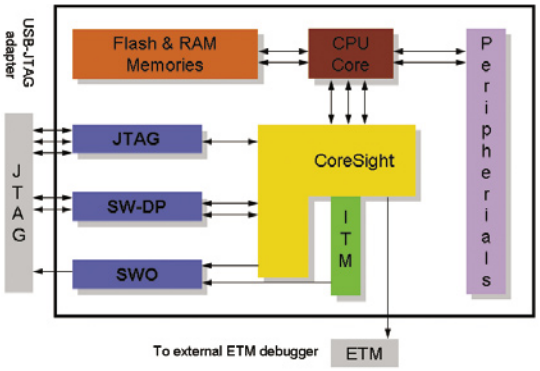
Rewolucja jaką przyniosły mikrokontrolery i mikroprocesory oraz ich zastosowania, podniosła poprzeczkę wymagań co do narzędzi wspomagających uruchamianie oprogramowania. W związku z tym, że programy zmieniły charakter z obliczeniowych na sterujące, istotna stała się możliwość uruchamiania i podglądu działania w rzeczywistym środowisku i czasie. Te założenia spełniały emulatory sprzętowe tzw. ICE (*In Circuit Emulators*). Procesor zastępowany był w układzie docelowym wtykiem sondy emulującej podłączonej do komputera nadrzędnego. Emulatory umożliwiały uruchomienie programu, pracę krokową, zatrzymania na tzw. pułapkach realizowanych sprzętowo oraz próbkowania stanu pamięci, magistral i linii we/wy ze skomplikowanymi systemami wyzwalania i filtrów. Po zatrzymaniu aplikacji można było podejrzeć stan rejestrów, pamięci, portów itp. Oprogramowanie sterujące emulatorów, zależnie od producenta, pozawalało również śledzić realizację programu zarówno na poziomie asemblera, jak i języków wysokiego poziomu. Niestety, były to urządzenia drogie i przez to często niedostępne.

Przy relatywnie niskich prędkościach pracy mikrokontrolerów rodziny 8051 i podobnych, do budowy emulatorów wykorzystywano standardowe elementy elektroniczne oraz tzw. bond-out-chips. Te ostatnie to mikrokontrolery przeznaczone specjalnie do emulatorów, z identycznym do układu docelowego rdzeniem, rozbudowane o dodatkowe linie do przesyłania informacji o stanie wewnętrznym procesora i sterowania nim. Sprzętowe emulatory ICE było bardzo popularne do późnych lat 90-tych i to one wyznaczyły najwyższy standard debugerów w układzie docelowym.

Czym później tym gorzej

Na rynku zaczęły się pojawiać szybkie układy z dużą ilością wyprowadzeń, bardziej efektywną magistralą, większą ilością zintegrowanych funkcji. Ekstremalna prędkość pracy procesorów ograniczyła możliwości dalszego używania emulatorów sprzętowych ICE. Duże ilości wyprowadzeń i nowe obudowy np. typu BGA, uniemożliwiały mechaniczne podłączenie sond. Integracja wewnątrz struktury wszystkich zasobów odcięła dostęp z zewnątrz do wewnętrznych magistral adresowej, danych i sterującej. Wszystko to spowodowało cofnięcie stosowanej technologii uruchamiania układów , przynajmniej o 10 lat, jeśli nie do wczesnych lat monitorów testowych. Zaczęto promować symulatory i dodatkowe programy monitorujące dodawane do aplikacji projektanta pozwalające wydobyć na zewnątrz wewnętrzne dane procesora. Symulatory są efektywnym narzędziem, jednak poza docelowym środowiskiem pracy. Dla celu sprzęgnięcia ich z układem rzeczywistym potrzebują dodatkowo platformy programowo-sprzętowej, która i tak nie odtworzy w 100% rzeczywistych warunków pracy. Z kolei monitory testowe funkcjonowały kosztem czasu pracy procesora i wielkości kodu programu. Zwykle blokowały też używa-

Opracowano na podstawie materiałów firmowych „Real-Time Trace: Serial Wire Viewer" Robert Boys ARM Ltd.



Rys. 2. Bloki funkcjonalne debugera mikrokontrolerów Cortex

ny w aplikacji interfejs szeregowy UART do wydobycia interesujących danych na zewnątrz. Kolejnym problemem stało się to, że w coraz bardziej złożonych systemach sterowania czasu rzeczywistego zatrzymanie CPU nie zawsze jest możliwe co powodowało konkretne problemy z testowaniem i uruchamianiem urządzeń. Wielu błędów nie dawało się też rozpoznać.

Uruchamianie via JTAG

Wychodząc naprzeciw potrzebom rynku, producenci mikrokontrolerów, w tym również ARM, zaczęli w swoich układach integrować bloki funkcjonalne, których jedynym zadaniem było umożliwienie testowania i uruchamiania oprogramowania. Dostęp do tych modułów realizowany był przez interfejs JTAG (*Joint Test Action Group*). Standard JTAG od jakiegoś czasu był implementowany w różnych układach i służył przede wszystkim do testowania pakietów elektronicznych. Dzisiaj JTAG jest obecny w większości mikrokontrolerów i jest wykorzystywany również do uruchamiania oprogramowania w układzie docelowym z poziomu komputera nadrzędnego w konfiguracji pokazanej na **fot. 1**.

Wspomniane zintegrowane moduły debugera via JTAG zapewniają jednak tylko podstawowe funkcje testowe, takie jak uruchomienie, zatrzymanie, praca krokowa, zastawianie pułapek. Niestety większość z zaawansowanych funkcji, które były oferowane przez sprzętowe emulatory ICE, a do których m.in. należało zapisywanie pamięci śladu, została utracona.

Zaistniała realna potrzeba wydobycia na zewnątrz w czasie rzeczywistym informacji o pracy i stanie jednostki centralnej. I to przy akceptowalnych kosztach. Interesujące są przy tym podstawowe dane jak: stan licznika instrukcji, dane zapisywane, dane odczytywane czy zawartość wewnętrznych rejestrów. Firma ARM jako jedna z pierwszych wyszła naprzeciw tym potrzebom i dostarczyła rozwiązanie problemu w postaci zintegrowanego bloku ETM (*Embedded Trace Macrocell*)

Uruchamianie via ETM

W późnych latach 90-tych firma ARM zaprojektowała blok ETM w technologii „system on-chip", który dostarcza na zewnątrz podstawowe dane o pracy i stanie procesora. Dane są przesyłane 4, 8 lub 16 bitową dedykowaną magistralą śladu i przechwytywane przez specjalne, zewnętrzne narzędzie sprzętowe. Urządzenie to, stanowiące substytut emulatora sprzętowego, musi być zbudowane na szybkich układach elektronicznych, aby przechwycić wszystkie dane przy pełnej prędkości pracy rdzenia ARM. Jakkolwiek jest to rozwiązanie efektywne, to istnieje ogromne zapotrzebowanie na tańszą metodę gromadzenia informacji, bez potrzeby wykorzystania drogiego sprzętu. Blok ETM jest opcjonalnym dla producentów mikrokontrolerów ARM i nie wszystkie mikrokontrolery go posiadają.

Uruchamianie via SWV

ARM w swojej nowej rodzinie rdzeni Cortex-M3 zaimplementował nowy blok zintegrowanego debugera – CoreSight. W skład modułu CoreSight wchodzą wcześniej omówione bloki testowe JTAG, ETM oraz dodatkowo SWD (*Serial Wire Debug*) i SWV (*Serial Wire Viewer*). Również w nowym rdzeniu blok ETM występuje jako opcja i nie wszystkie Cortex'y go zawierają. Natomiast wszystkie mikrokontrolery oparte o rdzeń Cortex-M3, oferowane na dzień dzisiejszy przez Luminary Micro i STMicroelectronics, wspierają technologie JTAG, SWD i SWV.

SWD oferuje taką samą funkcjonalność jak JTAG, wykorzystując przy tym mniejszą ilość doprowadzeń – używane są tylko 2, podczas gdy JTAG potrzebował 4…7. Umożliwia to zastosowanie mniejszego gniazda, zajmującego mniej miejsca na płytce. SWD i JTAG umożliwiają uruchomienie, zatrzymanie programu, pracę krokową, zastawienie do 8 pułapek sprzętowych i do 4 „watchpointów". Przykładowy adapter USB-JTAG/SW o nazwie ULINK2 oferowany przez firmę Keil obsługuje oba interfejsy.

SWV jest rozwiązaniem podobnym funkcjonalnie do portu ETM pozwalającym na wysłanie informacji na zewnątrz mikrokontrolera. Wykorzystywane jest przy tym tylko jedno wyprowadzenie, któremu nadano nazwę SWO (*Serial Wire Output*), przy czym sygnały SWD i SWO dzielą to samo gniazdo JTAG.

Co SWV może zrobić dla Ciebie?

Przy pomocy SWV można podejrzeć większość istotnych z punktu widzenia funkcjonowania programu informacji, ponieważ rejestruje on następujące dane:

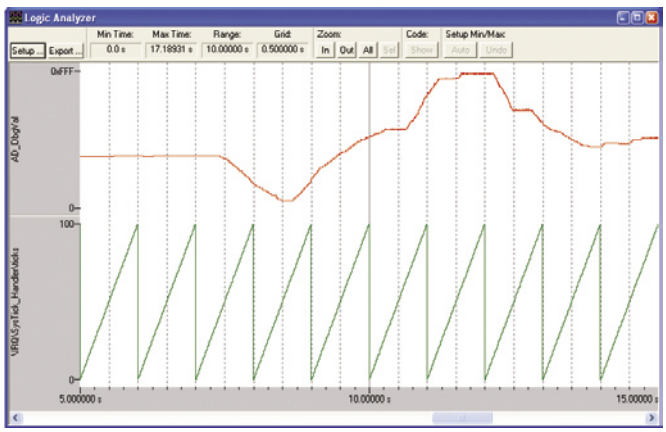
1. wartość licznika instrukcji PC,
2. cykle zapisu i odczytu danych,
3. wartości zmiennych i stany portów,
4. liczniki zdarzeń,
5. wejścia/wyjścia do/z procedur obsługi przerwań wyjątków,
6. czasy i cykle wykonywania instrukcji,

Nie jest przy tym potrzebna żadna nakładka na program aplikacyjny i przez to nie są wymagane żadne dodatkowe cykle instrukcji i tym samym nie są zmieniane rzeczywiste zależności czasowe. Projektant może przy tym sam wysyłać wewnętrzne dane na zewnątrz procesora poprzez wyjście SWO.

Podstawowa różnica pomiędzy SWV i ETM jest taka, iż ETM ma większą prędkość transmisji, jednak SWV dla większości zastosowań jest wystarczający i nie ma potrzeby inwestowania w drogie narzędzia sprzętowe obsługujące mechanizm ETM debugera. Przykładowo pakiet oprogramowania firmy KEIL (marka firmy ARM) RealView MDK (*Microcontroller Development Kit*) zapewnia obsługę SWV bez ponoszenia dodatkowych kosztów. Pakiet MDK wykorzystuje dobrze znane programistom oprogramowanie µVision3 firmy Keil jako środowisko projektowe IDE i tani adapter USB-JTAG. Przyjrzyjmy się bliżej poszczególnym możliwościom debugowania w tym środowisku, odsyłając jednocześnie czytelników po bardziej szczegółowe informacje na stronę *www.keil.com*.

Śledzenie zmiennych programowych

Środowisko µVision monitorując pracę mikrokontrolera może wyświetlać wartości zmiennych na kilka różnych sposobów – w oknie podglądu zmiennych (*Watch Window*), w oknie podglądu obszarów pamięci (*Memory Windows*), jak również w postaci graficznej jako przebieg tworzony na bieżąco w czasie rzeczywistym. Wartości zmiennych przesyłane są przez linie SWO za pomocą bloku SWV, który do swojej pracy nie musi angażować cykli i zasobów CPU, poza trzema wcześniej opisanymi liniami. Zmienne globalne, statyczne i struktury wyświetlane są w prosty i przejrzysty sposób. Nie ma natomiast możliwości wizualizacji zmiennych lokalnych. W tym celu muszą one najpierw zostać przekonwertowane na zmienne globalne lub statyczne. Na **rys. 3** pokazano przykładową



Rys. 3. Wizualizacja zmiennych w analizatorze stanów środowiska µVision

graficzną wizualizację w czasie rzeczywistym zmiennej globalnej `AD_DbgVal` związanej z stanem napięcia na wejściu przetwornika analogowo-cyfrowego oraz zmiennej `Systick`.

Śledzenie rejestrów urządzeń peryferyjnych

SWV umożliwia odczyt w czasie rzeczywistym wartości rejestrów związanych z urządzeniami peryferyjnymi mikrokontrolera. Niezależnie czy wykonywany jest zapis czy odczyt pod danym adresem urządzenia peryferyjnego przez zmienną globalną czy statyczną, zdarzenie jest monitorowane i zapisane w pamięci śladu. Daje to możliwość prezentacji informacji w taki sposób, jak każdej innej zmiennej – w oknie podglądu zmiennych, w oknie podglądu obszaru pamięci lub w postaci wykresu czasowego takiego, jak pokazano na rys. 3.

Śledzenie licznika programu PC

SWV umożliwia odczyt i wyświetlenie wartości licznika programu PC. Jest to użyteczne do śledzenia przebiegu wykonania programu aplikacyjnego. Umożliwia analizę wykonania programu m.in.:

- zdefiniowanie sytuacji, w których program wpada w ślepą pętlę,
- identyfikację instrukcji, która się do tego przyczyniła,
- selekcję czasochłonnych procedur programowych, w których procesor „spędza" swój czas.

Niewątpliwie nawet dedykowany pojedynczy port szeregowy nie jest w stanie przesłać wszystkich wartości licznika PC z oczywistej przyczyny, to jest dużych prędkości z jakimi pracują procesory z rdzeniem ARM. Dlatego wprowadzono opcję stochastycznego próbkowania i śledzenie wartości licznika PC. Jest to często wystarczające do detekcji nieoczekiwanych zachowań aplikacji i analizy przetwarzania programu. Jednocześnie monitorowany jest czas przetwarzania, który może być prezentowany w cyklach lub sekundach. Na **rys. 4** przedstawiono sposób wizualizacji rekordów śladu - licznika instrukcji i czasu przetwarzania.

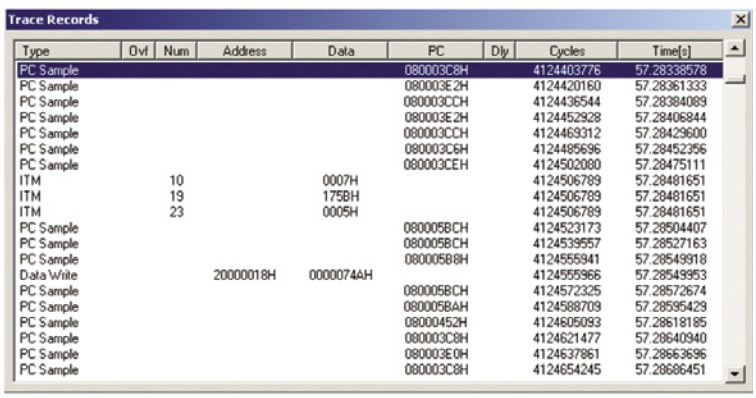
Śledzenie zapisów/odczytów pamięci

Na rekord śladu obrazujący odczyt lub zapis danych składają się typowo: adres instrukcji zapisu/odczytu, wartość zapisywana/odczytywana, adres zmiennej, czas wykonania liczony w cyklach CPU i w sekundach. Na **rys. 5** przedstawiono przykładową wizualizację serii zapisów danych. Istnieje przy tym możliwość filtrowania wyświetlanej informacji poprzez wybór odpowiednich opcji w pojawiającym się oknie.

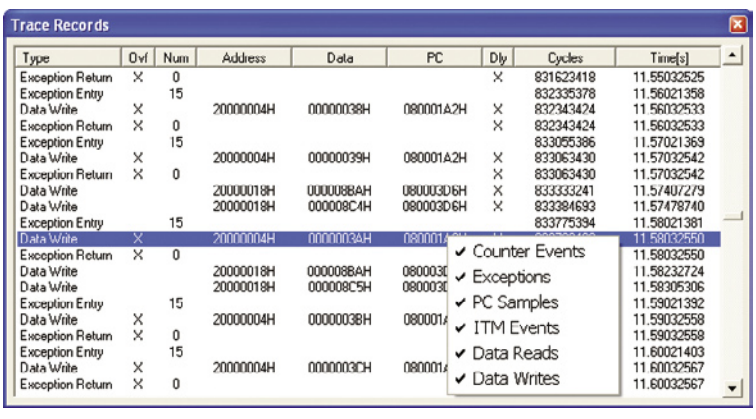
Tryb ITM

Tryb ITM (*Instrumentatnion Trace Macrocell*) umożliwia użytkownikowi zapisanie wybranej zmiennej do portu SWO i wysłanie jej wartości na zewnątrz za pośrednictwem interfejsu SW. Dane te mogą być wyświetlane przez środowisko IDE w postaci rekordów śladu (**rys. 5**) lub w oknie monitorującym interfejs SW (**rys. 6**). Tryb ten w małym stopniu ingeruje w program aplikacyjny. Wymaga bowiem dodania krótkiego kodu do programu aplikacyjnego wpisującego daną zmienną do rejestru SWO.

Tryb ITM jest analogiczny do używanej kiedyś metody wysyłania zmiennych kontrolnych na zewnątrz przez interfejs UART. Nie zajmuje jednak zasobów procesora przydatnych w aplikacji



Rys. 4. Stochastyczna wizualizacja rekordów śladu – licznika instrukcji, czasu przetwarzania i zmiennych zdefiniowanych przez użytkownika w trybie ITM



Rys. 5. Wizualizacja rekordów śladu – odczytów, zapisów i wyjątków

Liczniki zdarzeń

W trakcie uruchamiania użytkownik ma możliwość śledzenia pewnych zdarzeń mających miejsce w trakcie wykonywania programu. Z każdym takim zdarzeniem powiązany jest licznik naliczający ilość jego wystąpień. Rodzaje zdarzeń są predefiniowane przez środowisko IDE. I tak istnieje możliwość śledzenia:

1. ilości cykli zegara jednostki centralnej (32 bit),
2. ilości wykonanych instrukcji (8 bits),
3. ilości cykli instrukcji (8 bit),
4. ilości cykli w trybie uśpienia (8 bit),
5. ilości cykli w obsłudze przerwań (8 bit),
6. ilości cykli operacji zapisu/odczytu.

W każdej chwili przepełnienie powiązanego z danym zdarzeniem licznika, powoduje zmianę wartości licznika zdarzeń w oknie Event Counters. Mechanizm ten może zostać wykorzystany do pomiaru czasu, efektywności pracy systemu itp.

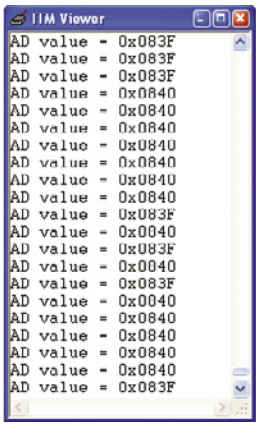
Na **rys. 7** pokazano ile razy każdy z wyżej wymienionych liczników się przepełnił. Liczniki te mogą zostać przy tym wyzerowane poprzez wciśnięcie „0".

Śledzenie wyjątków

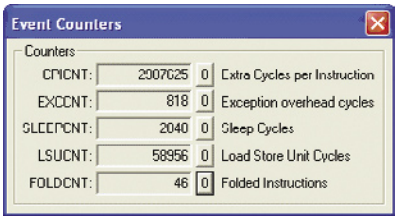
SWV może rejestrować wejścia i wyjścia do/z procedury obsługi wyjątków. W oknie śledzenia wyjątków przedstawiony jest zmierzony czas i numer wyjątku. Na **rys. 8** przestawiono okno Exception Trace, gdzie wyjątek SysTick (15) pojawił się 921 razy. Informacje o wyjątku wyświetlane w tym oknie są również powtarzane w oknie Trace Records pokazanym na rys. 5, gdzie przykładowo są wyświetlone wyjątki o numerach 0 i 15.

Warto, nie warto

Śledzenie programu zarówno z wykorzystaniem mechanizmów ETM jak i SWV znacząco zwiększa efektywność wykrywania błędów w trakcie uruchamiania oprogramowania aplikacyjnego. Występujące przy realizacji dużych projektów problemy, zabierające godziny, dni a nawet tygodnie, często mogą być wykryte w ułamku tego czasu z wykorzystaniem opisanych powyżej funkcji śledzenia w czasie rzeczywistym. Jest to szczególnie istotne w przypadkach, gdy nie wolno zatrzymać programu apli-



Rys. 6. Okno Serial Wire Viewer

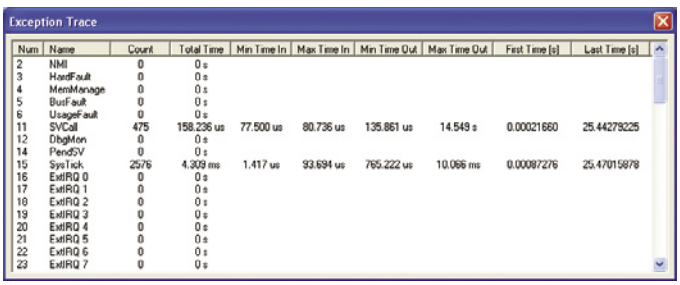


Rys. 7. Okno Event Counters

kacyjnego ze względu na groźbę uszkodzenia urządzenia. Tylko funkcja śladu daje możliwość podglądu bez zatrzymywania programu. Jest to również istotne w sytuacjach, gdy konsekwencje jakiegoś błędu objawią się znacznie później, niż moment wykonania błędnego kodu. Trudno bowiem zidentyfikować błąd po jego spóźnionych objawach. Poniżej zestawiono typowe rodzaje problemów, które można rozwiązać przy pomocy funkcji śladu:

- problemy ze wskaźnikami,
- wykonywanie nielegalnych instrukcji i zapisów danych,
- nadpisywanie kodu – zapisy do pamięci Flash, nieoczekiwane zapisy do rejestrów SFR, błędy obsługi stosu,
- błędne wartości zmiennych, nie zainicjalizowanie zmiennych i tablic, – nadpisywanie zmiennych,
- przepełnienia stosu,
- problemy z protokołami komunikacyjnymi, problemy czasowe,
- problemy z timingiem systemu, zarządzaniem systemem przerwań i procesami,
- niestabilne działanie programu – przysłowiowe „chodzenie w maliny",
- problemy z czasami przetwarzania i koniecznością optymalizacji kodu.

Przy wykorzystywaniu systemów operacyjnych czasu rzeczywistego (RTOS) funkcja śladu pozwala dodatkowo na wyświetlenie w czasie rzeczywistym interesujących informacji dotyczących stanu poszczególnych procesów np. liczby zadań i aktywnych procesów, stanu semaforów itp.



Rys. 8. Okno Exception Trace

Próba niewiele kosztuje

Blok SWV dostarcza tanią metodę wydobywania „on-line" informacji z wnętrza mikrokontrolera. Do wykrywania bardzo złożonych, głęboko ukrytych błędów i do bardzo restrykcyjnego debugowania, nadal na pewno jest i będzie wykorzystywana technologia ETM. Jednak w świetle zwykle ograniczonych finansów i codziennej praktyki nie wymagającej najwyższej jakości środków, technologia SWV jest nie do pobicia. Daje najlepszy stosunek jakości do ceny. Zachęcamy więc – spróbuj, a polubisz.

Wszystko, co jest potrzebne do zapoznania się z tematem to: płytka ewaluacyjna z procesorem STM32 lub procesorem Stellaris firmy Luminary Micro, adapter USB-JTAG Ulink2 lub UlinkME firmy Keil oraz co najmniej darmowe, ewaluacyjne oprogramowanie RealView MDK. MDK bowiem nawet w wersji ewaluacyjnej oferuje dużą grupę przykładów i zapewni przednią „zabawę".

Paweł Adamczyk
WG Electronics Sp. z o.o.