STM32 dla użytkowników 8-bitowców (1)

32-bitowe mikrokontrolery są postrzegane przez konstruktorów jako elementy do bardziej wymagających zadań. Ta opinia jest ugruntowywana przez wyposażenie w stosunkowo duże pamięci programu Flash i pamięci danych oraz w zaawansowane układy peryferyjne. Duża wydajność i znakomite wyposażenie peryferyjne łączyło się dotychczas ze zdecydowanie wyższą ceną w porównaniu z prostszymi mikrokontrolerami 8-bitowymi czy nawet 16-bitowymi. Wynikało to między innymi z tego, że skomplikowany 32-bitowy rdzeń zajmuje dużą powierzchnię na płytce krzemowej i jest droższy w produkcji.

Nowa generacja rdzeni Cortex-M zaprojektowana przez firmę ARM specjalnie do układów sterowania została znacząco uproszczona. Uproszczenie spowodowało, że zadania stawiane w układach wbudowanych mogą być wykonywane szybciej, przy zmniejszonym poborze energii, ale jednoczenie implementacja w krzemie ma mniejsze wymiary i przez to jest tańsza. Z tych powodów, mogły pojawić się mikrokontrolery z wydajnym rdzeniem 32-bitowym w cenie jednostek 8-bitowych. Wykorzystali to oczywiście producenci mikrokontrolerów wietrząc w tym możliwość udziału w olbrzymim rynku prostszych aplikacji dotychczas opanowanym przez mikrokontrolery 8-bitowe.

Posiadanie dobrego produktu to jedno, a sprzedanie go, to zupełnie inna rzecz. Oprócz ceny bardzo ważną rzeczą jest przyzwyczajenie konstruktorów. Jeżeli ktoś swoje urządzenie potrafi zrobić bez problemu na korzystając z 8-bitowca, to dlaczego nagle ma zacząć korzystać z zupełnie innej architektury, narzędzi i rezygnować z wszystkich swoich przyzwyczajeń? Tym bardziej, że producenci 8-bitowców również nie próżnują i ich wyroby są coraz lepiej wyposażone a dzięki nowym technologiom wytwarzania mogą być taktowane szybszymi zegarami i wydajanie oszczędzać energię. Żeby sprzedać nowe rozwiązania trzeba użytkownika jakoś przekonać. Może to być łatwość zakupu elementu, dostępne, tanie i łatwe w zdobyciu płytki ewaluacyjne, dostępne i tanie lub bezpłatne narzędzia software'owe i środowisko IDE. Ważnym elementem kampanii reklamowych jest propagowanie nowych rozwiązań w szkołach i uczelniach, tak by nowo wykształceni konstruktorzy je znali i używali.

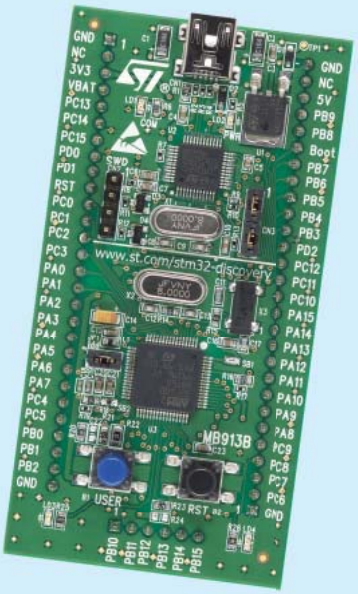
Ja postanowiłem przekonać jak wygląda przejście z architektury 8-bitowej na 32-bitową z perspektywy kogoś kto to robi pierwszy raz. Załóżmy, że do docelowego projektu szukamy taniego mikrokontrolera 32-bitowego z rdzeniem Cortex. Wybór padł na rodzinę STM32F100 produkowaną przez firmę STMicroelectronics. Układy te mają rdzeń Cortex–M3 taktowany z maksymalną częstotliwością 24 MHz. Pamięć programu Flash może mieć maksymalną pojemność 512 kB, a pamięć danych RAM 32 kB.

Tani moduł ewaluacyjny

Najprostszą i wbrew pozorom najtańszą metodą na szybkie i bezproblemowe rozpoczęcie pracy z nowym mikrokontrolerem jest zakup modułu ewaluacyjnego. Jeżeli chcemy wstępnie sprawdzić, czy element rzeczywiście spełni nasze wymagania, to poszukamy modułu taniego, ale odpowiednio wyposażonego. Przez to wyposażenie rozumiem minimum 2 elementy: wbudowany programator/emulator i wyprowadzenie linii portów do złącz, które można łatwo wykorzystać. Programator w tanim module znacznie obniża koszty, bo wciąż nie wiemy czy nowy element spełni oczekiwania i wydatek na drogi programator/emulator jest ryzykowny. Dostępne wyprowadzenia mikrokontrolera znacznie ułatwiają pracę, bo znam tanie moduły z nietypowym złączem do których trzeba dokupić bardzo drogę płytkę rozszerzenia, by cokolwiek z nim zrobić.

Dla STM32 Value Line znalazłem tani (ok. 60 PLN) i spełniający wszystkie zestaw ewaluacyjny STM32 Value Line Discovery (**fotografia 1**).

Zestaw jest podzielony na 2 części: programator/emulator zgodny z ST-Link/V2. To dobra wiadomość, bo jest to narzędzie wspierane przez najpopularniejsze środowiska IDE. ST-Link jest sterowany i zasilany ze złącza USB co dodatkowo upraszcza jego używanie. Programator wbudowany w płytkę ma wyprowadzone sygnały magistrali programującej i można go wykorzystać jako zewnętrzny programator we własnych konstrukcjach po zakończeniu prób. Bardzo rozsądne rozwiązanie za niewielkie pieniądze.



Fotografia 1. Zestaw STM32VLDiscovery

W drugiej części modułu umieszczono mikrokontroler do prób – STM32F100RBT6. Jego wyprowadzenia są połączone z dwoma listwami goldpinów, które są umieszczone na krawędzi płytki. Każde z wyprowadzeń jest czytelnie opisane. Bardzo mi się takie rozwiązanie spodobało, bo bez żadnych pro-

blemów można się dołączyć do linii portów na przykład za pomocą przewodów zakończonych wtyczkami pasującymi do goldpinów. Oprócz mikrokontrolera na płytce są jeszcze umieszczone 2 rezonatory kwarcowe: 8 MHz i 32 kHz („zegarkowy"), 2 diody LED zielona i niebieska połączone z liniami portów, oraz 2 przyciski: jeden do zerowania mikrokontrolera (RST), a drugi połączony z linią portów (USER). Spartańskie wyposażenie modułu tłumaczy jego niską cenę. Jednak prostota jest zdecydowanie tutaj zaletą.

Środowisko IDE

Mamy już moduł z programatorem i dużo zapału. Potrzebujemy teraz środowiska IDE, w którym będzie można pisać programy, potem je kompilować, uruchamiać i na koniec programować pamięć układu. W dokumentacji modułu są polecane:

- uVision 4 produkowane przez firmę Keil,
- Embedded Workbench for ARM produkowany przez firmę IAR,
- TrueSTUDIO produkowana przez firmę Atollic.

Ponieważ należy się spodziewać, że każde z tych środowisk będzie dobre, to wybrałem losowo uViosion 4 firmy Keil. No może nie do końca losowo, bo firma Keil jest częścią firmy ARM – projektanta rdzenia Cortex-M3 i na początek wydała mi się najbardziej wiarygodna. Nie oznacza to ,ze uważam, że pozostałe IDE są w jakiś sposób gorsze. Po prostu musiałem coś wybrać.

IDE uVision4 oprócz środowiska IDE zawiera też wersję ewaluacyjną kompilatora C dla naszego rdzenia. Keil jest znany przede wszystkim z kompilatorów C w tym słynnego już dla 8051 i spodziewałem się, że ten będzie równie dobry. Bezpłatna wersja kompilatora dostarczana wraz z pakietem IDE ma ograniczenie kodu do 32 kB i do celów eksperymentów na pewno wystarczy. Jednak w tym miejscu musimy sobie przypomnieć, że naszym celem jest sprawdzenie czy można bezboleśnie przejść ze stosowania mikrokontrolerów 8-bitowych na 32-bitowe Cortex'y. Dla 8-bitowców Atmela jest do dyspozycji bezpłatny kompilator GCC bez ograniczenia kodu i innych ograniczeń. Użytkownicy Microchipa mają ewaluacyjne wersje kompilatorów wszystkich rodzin łącznie z 32-bitowymi PIC32. Ograniczeniem jest tylko brak silnej optymalizacji kodu, przez co pamięć programu zapełnia się szybciej niż mogłaby. Na tym etapie poszukiwań mamy „tylko" 32 kB albo „aż" 32 kB. Nie mając za sobą kilku praktycznych projektów, które zastępują projekty z 8-bitowcami trudno powiedzieć czy to wystarczy, ale na pewno to ograniczenie da o sobie prędzej czy później znać. Być może znajdzie się inne legalne rozwiązanie, bo kupienie drogiego kompilatora tylko po to, aby zmienić rodzinę, na pewno nie wchodzi w grę. Przypomnijmy, że rozważamy przejście z 8 bitów na 32 w sytuacji, kiedy 8 bitów jeszcze wystarczy, a na przykład chcemy być bardziej nowocześni, bo tak sugerują wszechobecne reklamy.

Wracajmy do tematu. Pakiet uVision 4 można pobrać po uprzednim wypełnieniu ankiety ze strony *www.keil.com*. Instalacja przebiega tak, jak w podobnych programach tego typu i nie wymaga komentarza. Po zakupie płytki i zainstalowaniu IDE mamy już wszystko co potrzebne by zacząć próby.

Płytka ewaluacyjna zapewnia programowanie, zasilanie i taktowanie mikrokontrolera. Spróbujemy napisać najprostszy program zapalający i gaszący diodę LED umieszczoną na STM32 Value Line Discovery. Żeby nawet taki prosty program mógł zadziałać, trzeba zadbać o taktowanie mikrokontrolera. Mamy na płytce rezonator kwarcowy i wydaje się, że nie będzie z tym problemu.

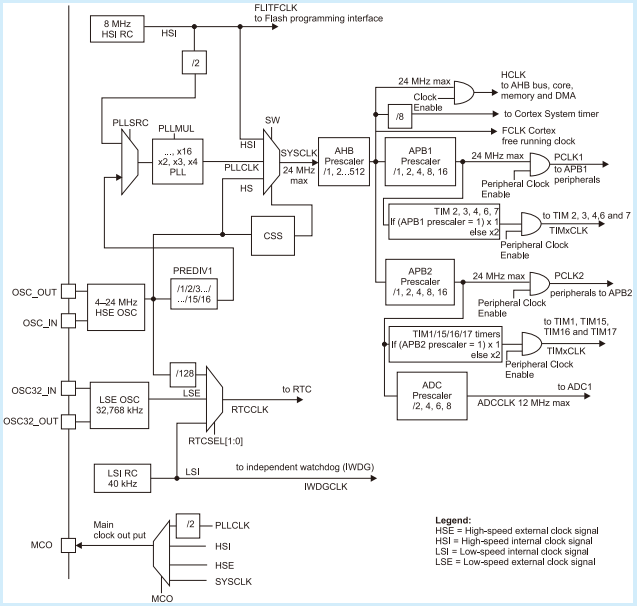
W historycznie pierwszych mikrokontrolerach źródłem sygnału zegarowego był wyłącznie oscylator kwarcowy z zewnętrznym oscylatorem czyli „kwarcem". Żeby mikrokontroler mógł działać, trzeba było dołączyć do niego kwarc i ewentualnie kondensatory ceramiczne o pojemności 10...50 pF, zależnie od wymagań układu. Tak było chociażby w intelowskich 8051. Częstotliwość taktowania klasycznego 8051 zawierała się w granicach 1,2...12 MHz. W mikrokontrolerach PIC16 wprowadzono konfigurację generatora taktującego rdzeń za pomocą bitów konfiguracyjnych umieszczonych w pamięci Flash i programowanych programatorem razem z pamięcią programu. To już był duży postęp, bo można było taktować małymi częstotliwościami np. 32 kHz lub stosować generator RC z zewnętrznymi elementami RC. Wraz z rozwojem technologii zaczęto stosować wewnętrzne generatory RC o wysokiej częstotliwości do taktowania rdzenia i generatory RC małej częstotliwości do celów pomocniczych, np. do taktowania wewnętrznego watchdoga. Wbrew pozorom, to dość ważny krok w technice taktowania. Wewnętrzny generator RC ma zalety: nie wymaga elementów zewnętrznych i przede wszystkim bardzo szybko startuje po włączeniu zasilania, czego nie można powiedzieć o generatorach kwarcowych. Jest idealny w aplikacjach, które nie wymagają dużej stabilności generowanej częstotliwości, czyli tam, gdzie nie jest wymagane dokładne odliczanie czasu lub transmisja asynchroniczna. Wewnętrzny RC jest chętnie stosowany w popularnych 8-bitowcach Atmela i Microchipa, ale opcja dodatkowa dla oscylatora kwarcowego.

Zatem w mikrokontrolerach 8-bitowych najpopularniejszy generator sygnału taktującego to generator kwarcowy o programowanym zakresie częstotliwości albo wewnętrzny generator RC o częstotliwości 4 MHz lub 8 MHz. Taktowanie jest programowane bitami konfiguracyjnymi tak zwanymi fusebit'ami. Oprócz tego, stosuje się wewnętrzne dzielniki dzielące częstotliwość generatora (głównie wewnętrznego RC), tak aby można było taktować mikrokontroler z częstotliwościami niższymi, jeżeli to konieczne. Dzielnik można programować w trakcie pracy mikrokontrolera i dynamicznie zmieniać częstotliwość taktowania, zależnie od potrzeb. Sprawa jest stosunkowo prosta. Zobaczmy teraz, jak to jest w przypadku mikrokontrolera STM32. Wiemy już, że rdzeń może być taktowany maksymalną częstotliwością 24 MHz. Podobnie jak w mikrokontrolerach 8 bitowych, źródłem sygnału może być generator kwarcowy nazwany HSE i wewnętrzny oscylator RC nazwany HSI. Generator HSE wymaga zewnętrznego oscylatora kwarcowego o częstotliwości z zakresu 4...24 MHz. Na pierwszy rzut oka wydaje się, że ten zakres w zupełności wystarczy. Rdzeń może być taktowany z maksymalną częstotliwością 24 MHz i wystarczy dobrać odpowiedni rezonator do naszych wymagań. Na płytce ewaluacyjnej kwarc ma częstotliwość 8 MHz. Czyżby producent płytki zamierzał taktować rdzeń tylko z 1/3 maksymalnej prędkości? Wybór tej częstotliwości jest w pewien sposób związany z rozbudowanym blokiem generowania sygnału zegarowego. W STM32F100 częstotliwość sygnału z gene-

ratora może być dzielona, ale może być też **powielana** w układach PLL (**rysunek 2**).

Ponieważ wewnętrzny generator RC HSI ma również częstotliwość 8 MHz, to można używać go zamiennie z HSE bez konieczności przeprogramowywania dzielników i układów PLL. Trzeba też wiedzieć, że po włączeniu zasilania mikrokontroler staruje z generatorem wewnętrznym i dopiero w wykonywanym programie można się przełączyć na generator kwarcowy. Ponadto, przy problemach z generatorem kwarcowym mikrokontroler przełącza się automatycznie na wewnętrzny RC. To jeszcze jeden argument za tym, by częstotliwość HSE była równa częstotliwości HSI. Jeżeli przypatrzymy się rysunkowi 2 to widać, że jest możliwe ustawienie odmiennych częstotliwości taktowania rdzenia i taktowania magistral układów peryferyjnych. Zaprogramowanie układu generowania zegara z jednej strony jest bardziej skomplikowane niż w 8-bitowych mikrokontrolerach, ale jednocześnie jest bardziej elastyczne. No cóż, coś za coś.

Ważne jest to, że zegar nie jest programowany bitami konfiguracyjnymi i musimy o jego konfigurację zadbać w programie, najlepiej na jego samym początku. Jak to zrobić? Mamy 2 możliwości: albo na podstawie dokumentacji mikrokontrolera zapiszemy wszystkie konieczne rejestry konfiguracyjne (co nie jest w końcu takie trudne, tylko pracochłonne), albo skorzystamy z gotowych procedur standardowej biblioteki *Standard Peripherial Library*. Obie metody są dobre, ale na początek dobrze jest zobaczyć jak robią to doświadczeni użytkownicy. Dlatego skorzystamy z gotowych bibliotek. Zazwyczaj producent mikrokontrolerów lub modułów ewaluacyjnych stara się dostarczyć przykładowe programy . Wykorzystamy to analizując za miesiąc przykład obsługi timera SysTick, pokazujący jak skonfigurować i wykorzystać licznik SysTick oraz zgłaszane przez niego przerwanie.



Rysunek 2. Moduł generowania sygnału zegarowego w STM32F100

Tomasz Jabłoński, EP