Jaszczurki zaciskają pasa

Możliwości energooszczędnych mikrokontrolerów EFM32 w teorii i praktyce (1)

Producenci układów scalonych prześcigają się w coraz to śmielszych i bardziej zaawansowanych technologicznie rozwiązaniach dotyczących oszczędzania energii w aplikacjach. Artykuł ten przedstawia w jaki sposób możemy wydłużyć autonomiczny czas działania konstruowanych przez nas urządzeń zasilanych bateryjnie opartych o mikrokontrolery rodziny EFM32 firmy Energy Micro.

Współczesny świat ceni sobie mobilność. Wymusza tryb życia związany z ciągłym przemieszczaniem się i zarazem pozostawaniem w kontakcie z osobami w różnych częściach globu oraz dostępu do informacji, a w szczególności dostępu do Internetu. Uwidacznia się to znaczącym wzrostem sprzedaży urządzeń mobilnych takich jak netbooki, tablety oraz smartphony. W ostatnich miesiącach można zaobserwować istny „wyścig zbrojeń" zarówno w komputerach mobilnych, jak i systemach operacyjnych je obsługujących. Prawdziwą rewolucją dziejącą się na naszych oczach jest rozwój systemów opartych na „chmurze", dzięki której dostęp do naszych plików możliwy będzie z każdego miejsca z podłączeniem do Internetu. Problemem przestanie być również utrata naszych danych w przypadku uszkodzenia domowego komputera. Jednak każde rozwiązanie ma swoje zalety i wady. Jakie rozwiązania systemowe i sprzętowe zagoszczą na dobre w naszym życiu, okaże się w niedalekiej przyszłości.

Zapotrzebowanie na energię

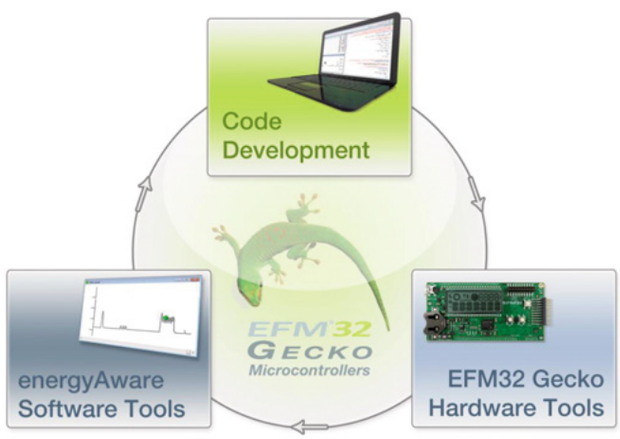
Mobilność jest ściśle związana z wygodą. Chyba nikt nie wyobraża sobie osoby sprawdzającej swoją pocztę internetową z tabletem w ręku oraz wielkim akumulatorem na plecach. Można zaryzykować stwierdzenie, że mobilność urządzenia nie jest tylko możliwością przemieszczania się z nim, lecz również umiejscawianie go tam, gdzie nie ma dostępu do zasilania sieciowego (poleganie na energii z innych źródeł), nie ograniczając jednocześnie jego użyteczności. Tyczy się to nie tylko komputerów przenośnych, ale wszystkich urządzeń i systemów obecnych w naszym środowisku. Między innymi systemów wbudowanych, monitoringu, odtwarzaczy muzycznych, sieci sensorowych, inteligentnych budynków itd. Chociaż w ostatnich latach prężnie rozwija się pozyskiwanie „zielonej energii", to wciąż w większości zastosowań polegać musimy na energii zgromadzonej w akumulatorach. Ponieważ nadal stosunek pojemności akumulatora dostępnego w komercyjnej sprzedaży do jego wymiarów i masy nie jest zadowalający, stąd jesteśmy poniekąd zmuszeni do stosowania innych rozwiązań w celu wydłużenia autonomicznego czasu działania urządzeń i systemów elektronicznych zasilanych bateryjnie.

Elektroniczna ewolucja

Chociaż bateryjne i akumulatorowe źródła energii są nadal wąskim gardłem, nie zahamowały rozwoju elektroniki. Wręcz przeciwnie, taka sytuacja sprawiła, że inżynierowie byli zmuszeni do wymyślania nowych rozwiązań zapewniających coraz to dłuższy czas działania urządzeń. Dzięki temu pomimo częstych rozmów telefonicznych, słuchania muzyki czy przeglądania stron internetowych w telefonie wystarczy naładować akumulator co kilka dni, a nie co kilka godzin. Można spokojnie stwierdzić, że na dzień dzisiejszy bardzo duża część rynku urządzeń elektronicznych jest opanowana przez mikrokontrolery, a w szczególności te z rdzeniami firmy *ARM*. Powstał również specjalny rdzeń do zastosowań w aplikacjach, w których krytyczną kwestią jest niski pobór energii przy zachowaniu wydajności 32-bitowego rdzenia *ARM* – jest nim *Cortex-M3*. Korzysta z niego kilku producentów mikrokontrolerów, między innymi relatywnie nowy gracz na rynku – *Energy Micro* z rodziną układów *EFM32*. „Maskotką" tych mikrokontrolerów jest najprymitywniejsza z jaszczurek – gekon. I tak jak różne gekony na drodze ewolucji przystosowywały się do specyficznych środowisk, tak i specjalnie zaprojektowane przez firmę *Energy Micro* gekony starają się sprostać restrykcyjnym wymogom stawianym przez aplikacje o małym poborze mocy. Na usta z lekkim uśmiechem cisną się dwa słowa – elektroniczna ewolucja.

Scalone laboratorium pomiarowe

Zaprezentowane możliwości oszczędzania energii, w które zostały wyposażone mikrokontrolery *EFM32* (gekony) trudno rzetelnie porównać z rozwiązaniami zastosowanymi w innych mikrokontrolerach przeznaczonych do aplikacji o małym poborze mocy. Dlatego wybór układu dla konkretnego urządzenia i ocena wydajności „gekonów" w stosunku do jego konkurentów pozostaje w gestii konstruktora.

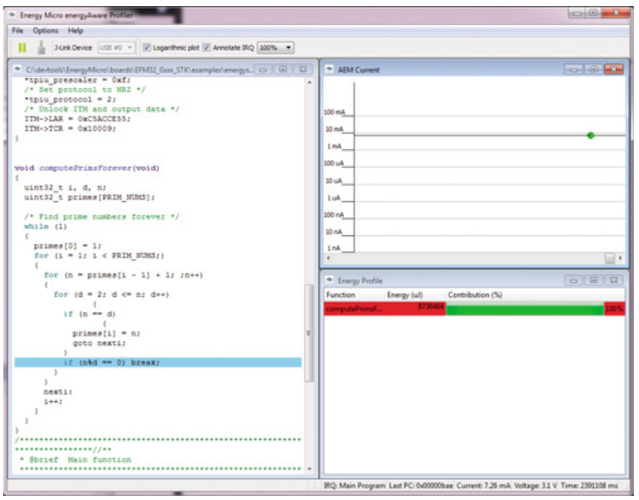


Rysunek 1. Kompletne środowisko do analizy i optymalizacji zużycia energii przez układy EFM32

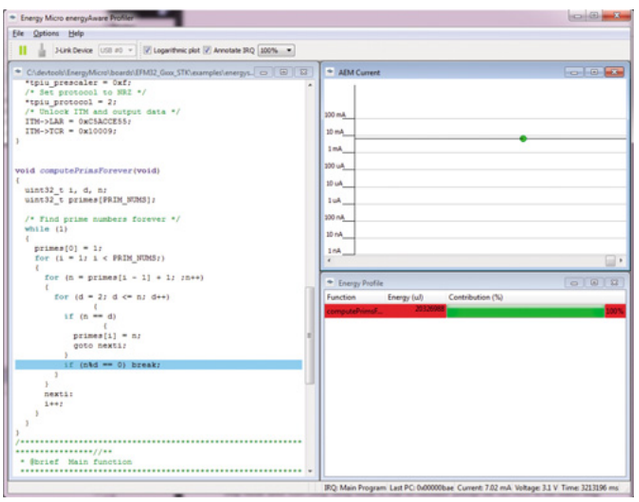
Do zaprezentowania możliwości mikrokontrolerów *EFM32* została użyta płytka testowa *STK* firmy *Energy Micro* z układem *EFM32G890F128* na pokładzie. Jest ona wyposażona między innymi w rezonatory kwarcowe 32,768 kHz i 32 MHz oraz system do nadzorowania poboru prądu *AEM* (*Advanced Energy Monitoring*). Dzięki *AEM* możemy mierzyć w czasie rzeczywistym prądy w zakresie od 100 nA do 50 mA. Dodatkowo dla zobrazowania wyników użyty został program *energyAware Profiler*. Dzięki zastosowaniu takiego zestawu narzędzi oraz środowiska *IDE* mamy do dyspozycji kompletne stanowisko do analizowania i optymalizacji zużycia energii przez nasz układ (**rysunek 1**). Aplikacja *energyAware Profiler*, po wgraniu do niej pliku wynikowego kompilacji stworzonego w oparciu o standard *ELF* (*Executable and Linkable Format*), ma możliwość śledzenia kodu programu równolegle do tworzenia wykresu czasowego poboru prądu przez urządzenie (**rysunek 2**). Dodatkowo program umożliwia sprawdzenie ile każda napisana przez nas funkcja „zużyła" dotychczas prądu oraz jest procentowy udział w całości zużycia przez mikrokontroler energii (**rysunek 3**). Dzięki temu wykrywanie nadmiernego poboru energii i optymalizacja kodu pod tym względem staje się dużo prostsza. Kod pozwalający przedstawić zmniejszenie zużycia prądu w czasie dzięki specjalnym rozwiązaniom zastosowanym w mikrokontrolerach *EFM32* został napisany w oparciu o przykłady i materiały dostępne na stronie *www.energymicro.com* oraz korzysta ze wsparcia bibliotek *CMSIS* (*The Cortex Microcontroller Software Interface Standard*), dzięki czemu lepiej zrozumieć napisane programy, jednak jest to kosztem mniejszego zoptymalizowania kodu pod kątem czasu wykonywania oraz ilości miejsca zajmowanej pamięci *Flash*.

Rysunek 2. Proces tworzenia wynikowego pliku ELF na potrzeby debugowania energetycznego

Testowanie i optymalizacja zużycia energii w czasie rzeczywistym była kiedyś bardzo czasochłonna i wymagała wielu specjalistycznych przyrządów pomiarowych oraz drogiego oprogramowania, co skutkowało dłuższym czasem wprowadzania urządzeń o niskim poborze mocy na rynek lub wprowadzanie urządzeń bez optymalizacji energetycznej. Dzisiaj prawie cała aparatura pomiarowa potrzebna do tego celu mieści się praktycznie w jednym układzie scalonym, a oprogramowanie typu *energyAware* można sobie ściągnąć z Internetu za darmo.

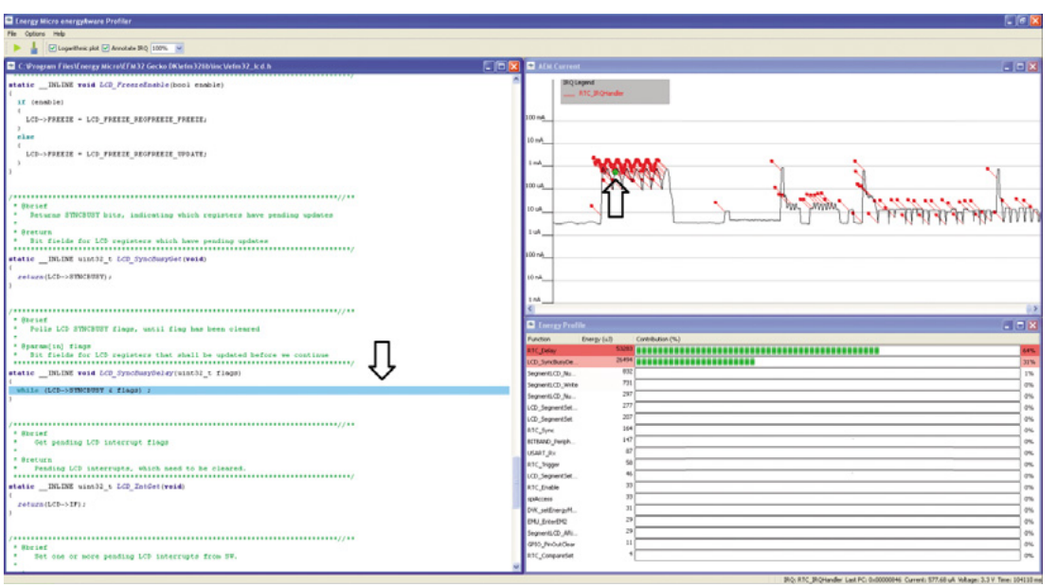


Rysunek 4. Debuggowanie energetyczne układu EFM32 taktowanego rezonatorem kwarcowym 32 MHz



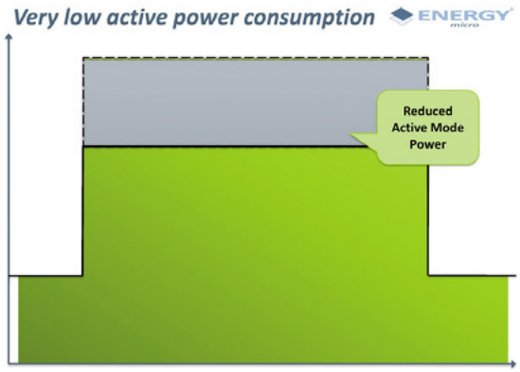
Rysunek 5. Debuggowanie energetyczne układu EFM32 taktowanego rezonatorem kwarcowym 32 MHz przy wyłączonych zegarach peryferyjnych

Arsenał elektronicznego gekona – teoria i praktyka

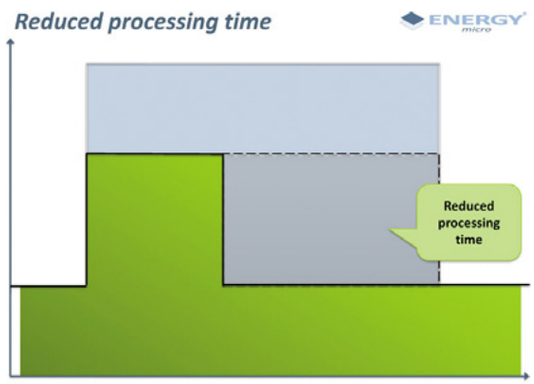


Rysunek 3. Przykładowy wygląd programu energyAware Profiler

Jednym z wyznaczników ilości energii zużywanej podczas pracy procesora jest pobór prądu w przeliczeniu na 1 MHz. Dla zilustrowania tego parametru posłużyliśmy się kodem źródłowym programu wyznaczającego liczby pierwsze. Dzięki odpowiednim ustawieniom mikrokontrolera możemy być pewni nieprzerwanej, ciągłej pracy rdzenia wykonującej program z pamięci *Flash*. Po resecie zostaje załączony zegar z zewnętrznego rezonatora kwarcowego – 32 MHz. Dzięki programowi *energyAware Profiler* odczytujemy zużycie prądu przez



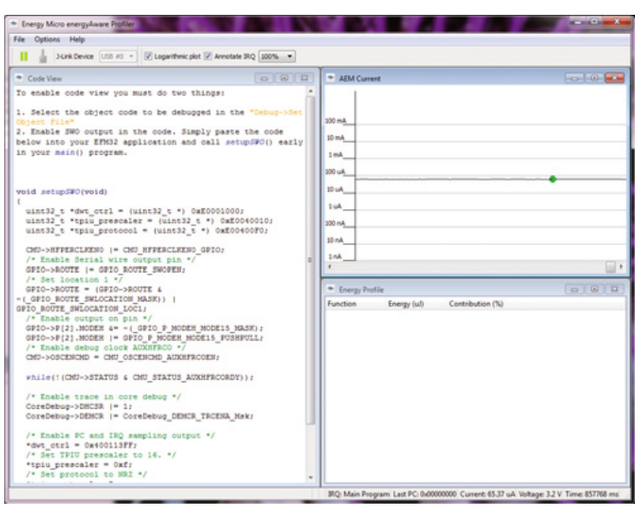
Rysunek 6. Zmniejszone zużycie energii układów EFM32 w stanie aktywnym



Rysunek 7. Krótszy czas przetwarzania układów EFM32 dzięki zastosowaniu rdzenia Cortex-M3

mikrokontroler. Wynosi ono 7,26 mA przy napięciu zasilania 3,1 V (**rysunek 4**). Wykonując proste obliczenia wyznaczamy interesujący nas parametr – zużycie prądu wynosi 227 µA/MHz przy zasilaniu 3,1 V. Całkiem dobry wynik. Czy można jeszcze mniej? Oczywiście!

„Gekony" dzięki elektronicznej ewolucji zostały przystosowane w szereg opcji umożliwiających oszczędność energii. Jedną z nich jest możliwość odłączenia sygnału zegarowego od nieużywanych w danym momencie modułów. A więc nieznacznie modyfikujemy kod, aby tym razem odłączyć zegary od wszystkich niepotrzebnych modułów. Wynik – oszczędziliśmy trochę energii. Teraz pobór prądu wynosi 7,02 mA przy 3,1 V (**rysunek 5**). Co daje 219 µA/MHz. Jest to zużycie energii w stanie aktywnym (**rysunek 6**). Jednak zanim przejdziemy do dalszych zalet „gekonów" trzeba mieć na uwadze jedną rzecz – rodzina układów *EFM32*

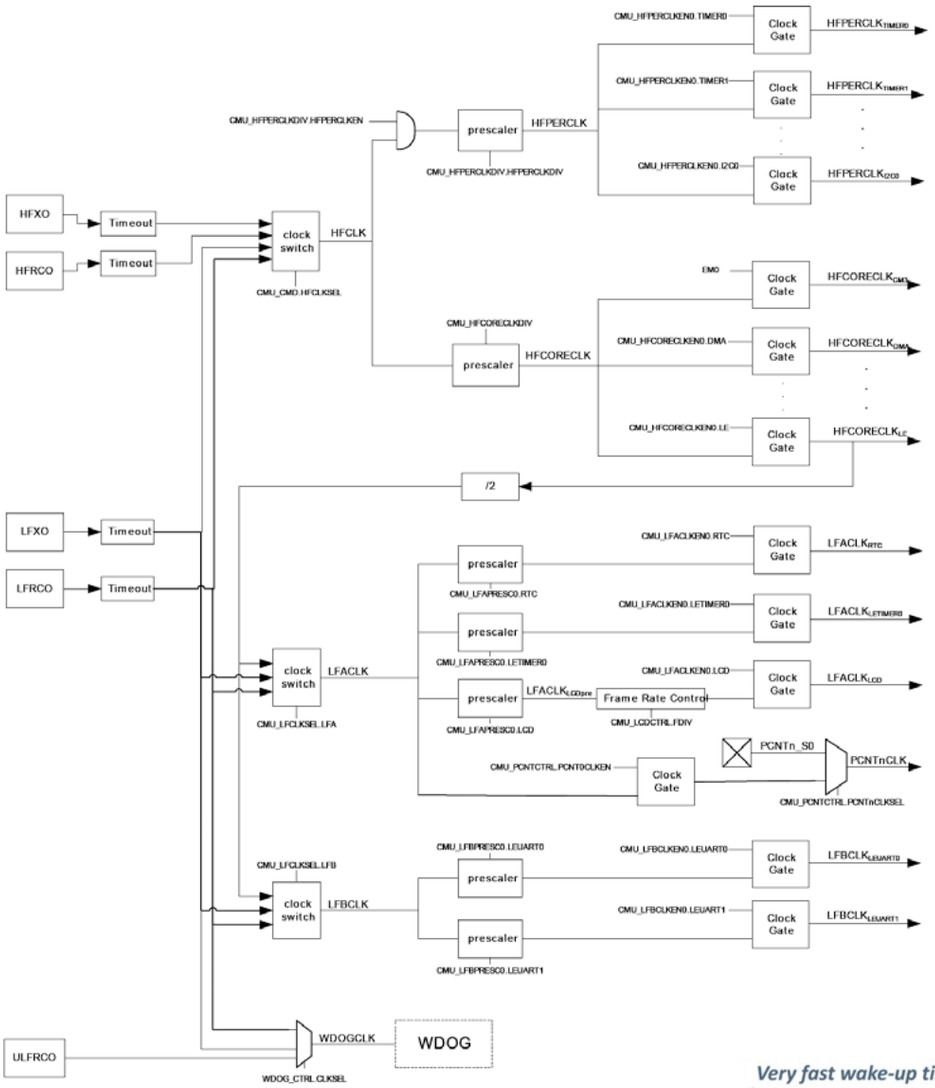


Rysunek 8. Debuggowanie energetyczne układu EFM32 taktowanego rezonatorem kwarcowym 32,768 kHz

nie została stworzona jako maszynka do szybkich obliczeń czy przetwarzania obrazów. Firma *Energy Micro* postawiła na skonstruowanie układu dla aplikacji o niskim poborze mocy, w których większość komponentów znajduje się w trybie uśpienia przez długi okres czasu, a tylko przez małą jego część potrzebne moduły przechodzą w stan aktywny, wykonują swoje zadanie i znowu przechodzą w tryb uśpienia. Dzięki zastosowaniu rdzenia *ARM Cortex-M3* operacje wykonują się szybciej, a zatem określone zadania trwają krócej, co z kolei zmniejsza czas w którym procesor i moduły są stanie aktywnym (**rysunek 7**). Redukcja całościowego poboru energii jest znaczna, co jest priorytetem w takich aplikacjach. Nie oznacza to jednak, że rodziny *EFM32* nie można używać w innych urządzeniach, chociaż w niektórych przypadkach byłoby to porównywalne do sprintera startującego w maratonie.

Można by pokusić się o stwierdzenie, że jeżeli ten mikrokontroler nie jest przeznaczony do szybkich obliczeń w czasie rzeczywistym, to w celu zaoszczędzenia energii możemy zastosować zegar o niskiej częstotliwości. Jest to pewne rozwiązanie, lecz w większości przypadków, jeżeli chodzi o działanie rdzenia, lepiej jest taktować go wysoką częstotliwością, żeby szybciej wykonał zadanie, a następnie przełączył się w tryb uśpienia. Jednak pomysł z obniżeniem częstotliwości taktowania nie jest taki zły. Jeżeli potrzebujemy wykonywać jakieś zadanie tylko co pewien długi okres, np. 1 sekundy, to zamiast taktować rdzeń i licznik kilkoma megahercami, możemy to zrobić np. rezonatorem zegarkowym. Przykład z obliczaniem liczb pierwszych po podłączeniu oscylatora 32,768 kHz pobierał prąd 65,37 µA przy 3,2 V (**rysunek 8**). Pod względem zastosowania i manipulacji różnymi częstotliwościami, architektura rodziny *EFM32* jest dobrze przemyślana. Do układów możemy podpiąć rezonator o dużej częstotliwości (4-32 MHz) oraz o małej częstotliwości (32,768 kHz). Dodatkowo dostępne są wbudowane rezonatory RC: małej częstotliwości (32,768 kHz) oraz programowalny dużej częstotliwości (1-28 MHz). Zegar został poprowadzony osobno do taktowania rdzenia, a osobno do peryferii. Bardzo przydatna jest możliwość wyłączania zegara poszczególnym peryferiom w celu oszczędzania energii. Również osobny zegar został poprowadzony do tak zwanych peryferii niskoenergetycznych *LE* (*Low Energy*). Cała ta sieć przełączanych zegarów daje nam sporo możliwości manipulacji (**rysunek 9**). Jeżeli tylko mądrze zostanie ona wykorzystana, to z pewnością pozwoli na dużą oszczędność energii w mikrokontrolerze.

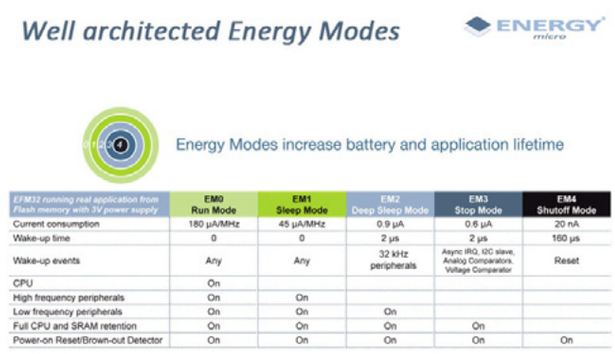
Jak powszechnie wiadomo wszystko w przyrodzie dąży do najniższego stanu energetycznego. Nic więc dziwnego, że inżynierowie podczas projektowania już dawno wprowadzili tryb uśpienia jako standardowa opcja każdego mikrokontrolera. Dzięki niemu jeśli nie jest konieczna ciągła praca układu możemy go uśpić w celu oszczędności energii, a następnie kiedy będzie taka potrzeba obudzić mikrokontroler na różne sposoby – na przykład poprzez przerwanie. Ekipa z *Energy Micro* nie zadowoliła się jednak tylko jednym trybem uśpienia. Postanowili zastosować ich aż cztery. Zatem łącznie z trybem normalnej pracy każdy gekon może znajdować się w jednym z pięciu trybów energetycznych (**rysunek 10**). Z punktu widzenia osoby piszącej oprogramowanie na mikrokontrolery *EFM32*, możemy spojrzeć na te tryby jako na profile energetyczne. Każdy z nich cechuje się zmniejszającą się ilością możliwych do użycia zasobów mikrokontrolera. Przykładowo sam rdzeń procesora dostępny jest tylko w trybie *EM0*, natomiast we wszystkich trybach oprócz ostatniego (*EM5*) utrzymywana jest zawartość pamięci SRAM. Peryferia zostały podzielone na dwie zasadnicze grupy: zwykłe , które można taktować z zegara zarówno nisko- jak i wysokoczęstotliwościowego oraz tak zwane niskoenergetyczne (*LE*), które zostały zaprojektowane w taki sposób, aby można było je taktować niskimi częstotliwościami, ale zarazem wciąż zachowując ich podstawową funkcjonalność, na przykład interfejs *UART* – do prędkości 9600 baud. Dzięki temu w celu dalszej oszczędności energii możemy przejść do takiego trybu w którym pracują tylko peryferia *LE*, a ani procesor, ani zwykłe peryferia nie są dostępne. Takie profile energetyczne można z przymrużeniem oka porównać do faz snu zwierząt, a w szczególności człowieka. Oprócz fazy zasypiania, w której powoli uspokaja się organizm i częstotliwości fal mózgowych są coraz



Rysunek 9. Blokowy schemat sieci zegarów w architekturze układów EFM32

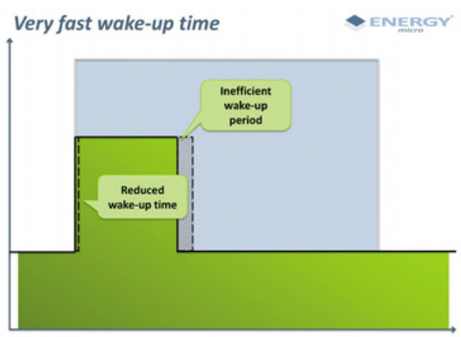
wolniejsze, istnieją jeszcze inne. Przykładowo istnieje faza o wolnych ruchach gałek ocznych (NREM), w której zwolniony jest metabolizm, wolniej pracuje większość organów, nieznacznie spada temperatura ciała (w przypadku człowieka o około 0,5 stopnia), dzięki czemu oszczędzana jest energia podczas snu oraz dodatkowo, organizm regeneruje się. Innej fazie – fazie szybkich ruchów gałek ocznych (REM) – towarzyszy bardzo niezwykłe zjawisko. Jest nim paraliż senny – odłączenie funkcji ruchowych organizmu – możemy go porównać do odłączenia napięcia zasilania i zegarów taktujących od peryferii mikrokontrolera podczas jednego z trybów uśpienia. Ciekawostką jest fakt, że właśnie w tej fazie występują sny i dlatego śniąc o bieganiu nie poruszamy kończynami – w większości przypadków oczywiście. A teraz wracając do tematu rodziny EFM32, dużą uwagę przywiązano również do tego, aby czas jaki potrzebuje procesor do przejścia z poszczególnych trybów uśpienia do normalnej pracy był jak najkrótszy (**rysunek 11**), a prąd pobierany przez konkretne, aktywne układy w odpowiednich profilach uśpienia był możliwie najmniejszy (**rysunek 12**). Dodatkowo, podczas pracy w trybie uśpienia niskoenergetyczne peryferia są w stanie w pewnym stopniu działać autonomicznie bez udziału procesora, dzięki czemu oszczędność energetyczna zwiększa się (**rysunek 13**). Nad całością czuwa specjalny moduł *EMU* (*Energy Management Unit*), który pozwala na przejścia pomiędzy profilami uśpienia i komunikuje się z odpowiednimi podsystemami mikrokontrolera, jak pamięć, kontroler oscylatorów, kontroler napięć czy moduł przerwań (**rysunek 14**).

Rodzina *EFM32* została wyposażona w specjalne, niskoenergetyczne (*LE – LowEnergy*) peryferia: *UART* (*LEUART*) oraz licznik (*LETIMER*). Dzięki nim możliwa jest praca tych peryferii będąc jednocześnie w trybie uśpienia *EM2*, w którym nie jest już możliwe korzystanie z ich zwykłych odpowiedników. Co prawda mają one swoje ograniczenia, jednak jest to kosztem mniejszej

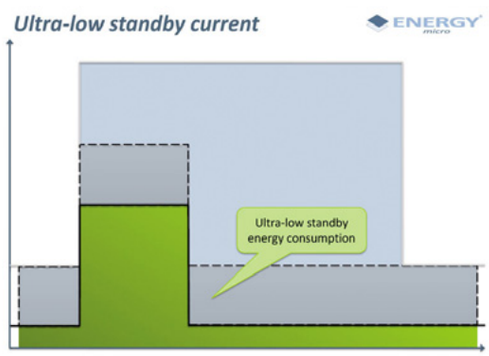


EFM32 running real application from Flash memory with 3V power supply	EM0 Run Mode	EM1 Sleep Mode	EM2 Deep Sleep Mode	EM3 Stop Mode	EM4 Shutoff Mode
Current consumption	180 µA/MHz	45 µA/MHz	0.9 µA	0.6 µA	20 nA
Wake-up time	0	0	2 µs	2 µs	160 µs
Wake-up events	Any	Any	32 kHz peripherals	Async IRQ, I2C slave, Analog Comparators, Voltage Comparator	Reset
CPU	On				
High frequency peripherals	On	On			
Low frequency peripherals	On	On	On		
Full CPU and SRAM retention	On	On	On	On	
Power-on Reset/Brown-out Detector	On	On	On	On	On

Rysunek 10. Zestawienie trybów energetycznych układów EFM32



Rysunek 11. Krótszy czas przejścia wybudzania się układów EFM32 z trybów uśpienia



Rysunek 12. Zmniejszone zużycie energii układów EFM32 w stanach uśpienia

ilości pobieranej energii. Przykładowo, firma *Energy Micro* zapewnia, że moduł *LEUART* pobiera tylko kilka µA podczas aktywnej pracy oraz tylko 150 nA oczekując na nadejście danych. Jednak przejście do trybu *EM2* nie oznacza, że możemy korzystać tylko z peryferii posiadających przedrostek *LE* w swojej nazwie. W tym trybie nadal dostępne są takie moduły jak analogowe komparatory, przetworniki A/C oraz C/A, częściowo interfejs I²C, zegar czasu rzeczywistego i inne. Dzięki wspomaganiu DMA możemy dłużej utrzymywać rdzeń procesora w uśpieniu podczas komunikacji otoczenia z mikrokontrolerem. Nasuwa się zatem pytanie: „a czy nie lepiej byłoby zostawić procesor uśpiony, jeżeli wynik działania jednego peryferium bezpośrednio wywołuje działanie drugiego?". Oczywiście, że byłoby lepiej. Inżynierowie z *Energy Micro* postanowili wykorzystać ten pomysł. Tak powstał moduł autonomicznej komunikacji między peryferiami *PRS* (Peripheral Reflex System). Dzięki zastosowaniu specjalnych, przełączanych kanałów komunikacyjnych pomiędzy modułami mikrokontrolera istnieje możliwość zadziałania przetwornika C/A po zakończeniu konwersji przetwornika A/C, czy zmiana wyjścia pinu układu wywołana licznikiem (**rysunek 15**), a wszystko to bez konieczności interwencji procesora.

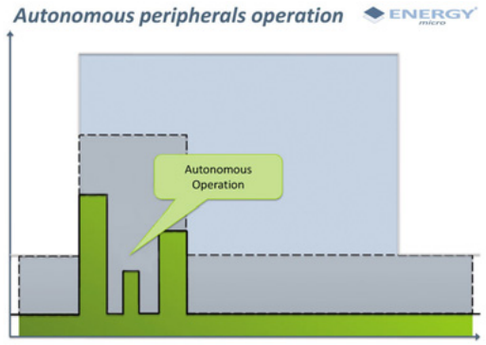
W celu zmniejszenia całościowego poboru prądu mikrokontrolera możemy również podczas projektowania systemu zaplanować niższe zasilanie. Rodzina *EFM32* posiada możliwość pracy przy napięciu zasilania z zakresu 1,8-3,8 V.

Podczas debuggowania jest opcja włączenia wyjścia *SWO* interfejsu *SWD*. Robimy to dodając na początku naszej pętli głównej *main()* kodu wywołania funkcji *SetupSWO();*. Dzięki temu będziemy mieć dostęp do takich rejestrów procesora jak np. *program counter*. Po aktywowaniu tej linii kodu będzie również możliwy podgląd kodu programu w relacji do wykresu zużycia energii w programie *energyAware Profiler*. Należy mieć na uwadze, że jeżeli ta opcja jest włączona, to prąd pobierany przez mikrokontroler będzie większy o kilkaset µA. Pamiętajmy, aby wyłączyć tą opcję, jeżeli nasz kod nie będzie wymagał już debuggowania, dzięki czemu dodatkowy prąd nie będzie pobierany. Dla sprawdzenia tego faktu weźmy pod uwagę początkowy przykład obliczania liczb pierwszych przy wyłączonych zbędnych sygnałach zegarowych. Pobór prądu wynosił 7,02 mA przy 3,1 V. Po włączeniu opcji *SWO* prąd pobór prądu wyniósł 7,58 mA przy 3,1 V (**rysunek 16**), czyli nieco ponad 0,5 mA więcej.

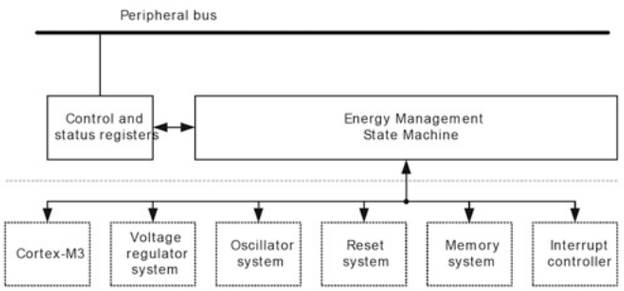
Wiedza podstawą sukcesu

Samo zastosowanie w aplikacji energooszczędnego mikrokontrolera nie zapewni jeszcze bardzo długiego czasu działania urządzenia pracującego na baterii. Podstawą jest zapoznanie się z możliwościami jakie dostarcza dany układ. A takich możliwości mikrokontrolery z rodziny *EFM32* mają wiele. Dzięki dostępnym na stronie *Energy Micro* notom aplikacyjnym oraz wielu przykładowym programom testowym jesteśmy w stanie szybko zrozumieć i nauczyć się korzystać z funkcji oszczędzania energii. Dodatkowo posiłkując się płytkami ewaluacyjnymi na których dostępny jest scalony miernik mocy czasu rzeczywistego (*AEM*) oraz oprogramowaniem *energyAware Profiler* jesteśmy w stanie zobaczyć ile energii oszczędzamy w danym zastosowaniu. Należy również mieć na uwadze fakt, iż projektując urządzenie w którym kluczowym parametrem jest minimalizacja zużycia energii, istnieje potrzeba zastosowanie nie tylko specjalistycznego mikrokontrolera. Również inne komponenty powinny cechować się niskim zużyciem prądu oraz możliwościami uśpienia, aby całościowe zużycie energii było jak najmniejsze. Dopiero poprzez połączenie energooszczędnych układów i przemyślane algorytmy sterowania nimi jesteśmy w stanie osiągnąć 3-4 krotne wydłużenie czasu życia aplikacji zasilanych bateryjnie, którym chwali się producent elektronicznych gekonów – firma *Energy Micro*.

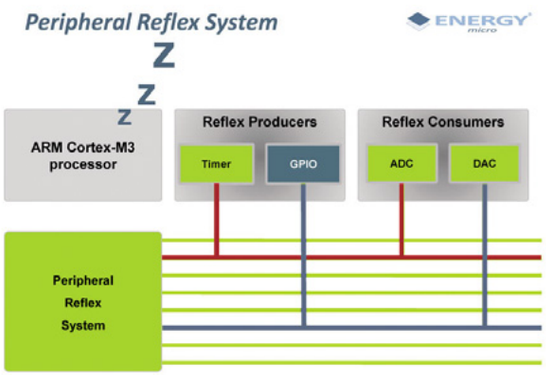
Wojciech Gelmuda
Piotr Bratek
Andrzej Kos
Katedra Elektroniki
Akademia Górniczo-Hutnicza



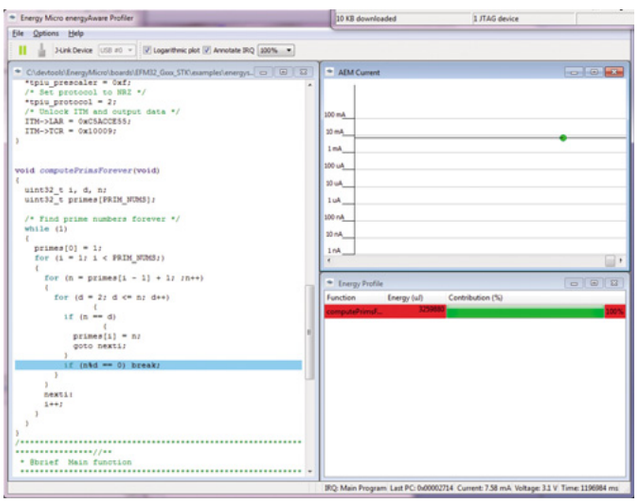
Rysunek 13. Zmniejszone zużycie energii układów EFM32 dzięki pewnej autonomii pracy układów peryferyjnych bez użycia rdzenia



Rysunek 14. Blokowy schemat komunikacji modułu EMU z innymi podsystemami



Rysunek 15. Blokowy schemat komunikacji modułu PRS bez użycia rdzenia



Rysunek 16. Debuggowanie energetyczne układu EFM32 taktowanego rezonatorem kwarcowym 32 MHz przy włączonym wyjściu SWO