STM32 – tryby obniżonego poboru mocy (2)

W poprzedniej części artykułu o trybach obniżonego poboru mocy opisałem ogólnie wszystkie tryby poboru mocy, budowę modułu STM32L Discovery i metody pomiaru bardzo małych prądów. Teraz skupimy się na dokładniejszym opisie dwu wybranych trybów: Sleep Low Power i Standby. Jednak przed tym razem dokładniej opiszę działanie dwóch zasadniczych bloków mikrokontrolera: regulatora napięcia i układu taktowania.

W każdym trybie oszczędzania energii jest używany programowany, wewnętrzny, liniowy regulator napięcia i związane z nim dynamiczne zarządzenie skalowanie napięcia zasilania. Regulator zasila wszystkie układy cyfrowe z wyłączeniem modułu trybu Standby. Dynamiczne skalowanie zasilania jest techniką zarządzania poborem mocy, która pozwala na zwiększanie (overvolting), lub zmniejszanie (undervolting) napięcia zasilania Vcore zależnie od wymagań.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 52617, pass: 30lct328
- pierwsza część artykułu

Regulator napięcia Vcore

Napięcie wyjściowe może być programowane w 3 zakresach do 1,2 V do 1,8 V:

- **Zakres 1**. Napięcie na wyjściu ma wartość 1,8 V dopóty, dopóki na wejściu regulatora napięcie jest wyższe niż 2 V. W tym zakresie napięcia zasilającego można kasować i zapisywać pamięć Flash mikrokontrolera
- **Zakres 2**. Napięcie na wyjściu wynosi 1,5 V. Programowanie pamięci Flash jeszcze jest możliwe, ale wydłuża się czas dostępu.
- **Zakres 3**. Napięcie na wyjściu wynosi 1,2 V. Dostęp do pamięci Flash jest najwolniejszy. Kasowanie i programowanie pamięci nie jest możliwe.

Zakresu 2 i 3 można używać w całym dostępnym zakresie napicia zasilającego, od 1,65 V do 3,6 V.

W **tabeli 1** umieszczono skrócony opis zależności wydajności mikrokontrolera od napięcia wyjściowego regulatora Vcore.

Po zerowaniu mikrokontrolera regulator jest zawsze włączony. Zależnie od trybu oszczędzania energii zastosowanego w aplikacji, pracuje on w 3 różnych trybach: *Main* (MR), *Low Power* (LPR) i *Power Down*.

- W trybie *Run* regulator pracuje w trybie *Main* (MR) i zasila pełną mocą domenę rdzenia (napięcie Vcore), czyli sam rdzeń, pamięci i cyfrowe układy peryferyjne.
- W trybie *Low Power Run* regulator pracuje w trybie *Low Power* (LPR) i zasila z małą mocą domenę rdzenia zachowując zawartość rejestrów i wewnętrznej pamięci RAM.
- W trybie *Sleep* regulator pracuje w trybie *Main* (MR) i zasila pełną mocą domenę rdzenia zachowując zawartość rejestrów i wewnętrznej pamięci RAM.
- W trybie *Low Power Sleep* regulator pracuje w trybie *Low Power* (LPR) i zasila z małą mocą domenę rdzenia zachowując zawartość rejestrów i wewnętrznej pamięci RAM.
- W trybie *Stop* regulator pracuje w trybie *Low Power* (LPR) i zasila z małą mocą domenę rdzenia zachowując zawartość rejestrów i wewnętrznej pamięci RAM.
- W trybie *Standby* regulator jest wyłączony. Zawartość wewnętrznej pamięci RAM i rejestrów nie jest zachowana.

Programowanie regulatora wymaga wykonania następującej sekwencji:

1. Sprawdzenia wartości napięcia Vdd po to, by wyznaczyć, który zakres Vcore jest dostępny (**rysunek 1**).
2. Testowanie bitu VOSF w rejestrze PWR_CSR. Trzeba czekać na wyzerowanie bitu VOSF.
3. Zapisanie bitów VOS[12:11] w rejestrze PWR_CR po to, aby zaprogramować zakres Vcore
   VOS[12:11]= 00 nieużywane
   01 1,8V (zakres1)
   10 1,5V (zakres2)
   ,2V (zakres3)
4. Testowanie bitu VOSF w rejestrze PWR_CSR. Trzeba czekać na wyzerowanie bitu VOSF w rejestrze PWR_CSR.

W programie testowym modułu STM32L Discovery do programowania napięcia regulatora jest używana funkcja biblioteczna z pliku stm32l1xx_pwr.c PWR_VoltageScalingConfig. Zamieszczono ją na **listingu 1**.Ta funkcja zapisuje tylko bity VOS rejestru PWR_CR. Kompletna sekwencja programowania musi kończyć się testowaniem bitu VOSF sygnalizującego ustabilizowanie się zaprogramowanego napięcia. Jest to pokazane na **listingu 2**.

W czasie konfigurowania regulatora system generatora sygnałów zegarowych jest zatrzymywany do czasu

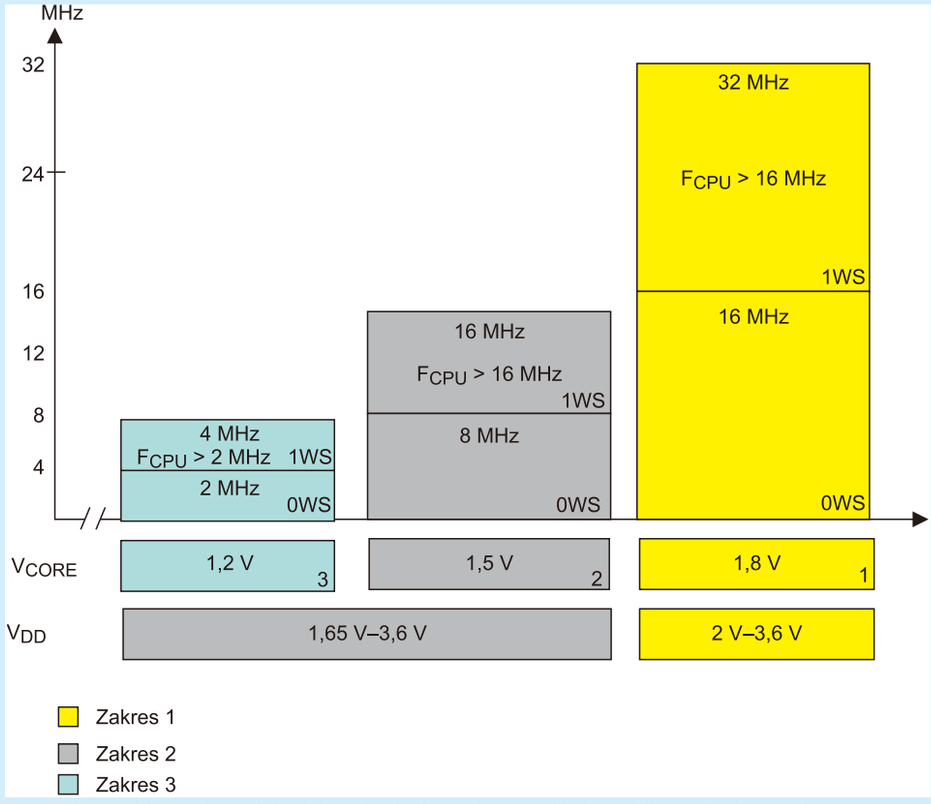
Tabela 1. Zależność wydajności od napięcia zasilającego Vcore

Wydajność CPU	Ograniczenie mocy	Zakres Vcore	Napięcie Vcore	Max. Częstotliwość		Zakres Vdd
				1 WS (*)	0WS	
Wysoka	Niskie	1	1,8	32	16	2,0…3,6
Średnia	Średnie	2	1,5	16	8	1,65…3,6
Niska	Wysokie	3	1,2	4	2	

(*) WS – liczba cykli *wait state*

aż napięcie na wyjściu ustabilizuje się. Świadczy o tym wyzerowanie się bitu VOSF. Trzeba to uwzględnić przy projektowaniu oprogramowania, szczególnie w wypadku sekwencji krytycznych czasowo z wykorzystaniem przerwań lub aplikacji wykorzystujących układy licznikowe.

Napięcie wyjściowe zakresu 1 napięcia Vcore jest poprawne, gdy Vdd jest większe niż 2 V. Jeżeli Vdd spadnie poniżej 2 V, to aplikacja musi zmienić konfigurację mikrokontrolera. Do detekcji obniżenia napięcia jest wykorzystywany blok *PVD Monitor*. Może on wygenerować przerwanie, jeżeli napięcie spadnie poniżej zaprogramowanej wartości. Żeby wykryć obniżenie napięcia poniżej 2,0 V, trzeba w PVD zaprogramować BOR na poziom 2 (typowo 2,26 V). Po wykryciu spadku poniżej 2,0 V musimy przeprogramować regulator. Do dyspozycji są zakresy 2 i 3. Zmian wymaga też konfiguracja systemu taktowania, bo częstotliwość sygnału zegarowego jest ograniczona do 16 MHz dla zakresu 2 i do 4,2 MHz dla zakresu 3 napięć zasilających. Oczywiście, jeśli jest wybrany zakres 2 lub 3, a napięcie zasilania spadnie poniżej 2,0 V, to nie są potrzebne żadne działania.



Rysunek 1. Zależność wydajności w funkcji napięcia zasilania Vdd i napięcia regulatora Vcore

Tryby obniżonego poboru mocy a taktowanie

Wiemy już, że pobór mocy przez mikrokontroler, oprócz wartości napięcia zasilającego, jest ściśle związany z częstotliwością taktowania mikrokontrolera.

Zegar systemowy o nazwie SYSCLK może pochodzić z jednego z czterech głównych źródeł:

- HSI (*High Speed Internal*). Sygnał zegarowy jest generowany przez precyzyjny, kalibrowany, wewnętrzny oscylator RC o częstotliwości 16 MHz. Może on być użyty jako zegar bezpośrednio taktujący rdzeń lub jako źródło zegara dla układu PLL. HSI jest kalibrowany w procesie produkcji z dokładnością 1% przy temperaturze otoczenia +25°C. Użytkownik ma możliwość samodzielnego kalibrowania taktowania mikrokontrolera za pomocą odpowiednich rejestrów.
- HSE (*High Speed External*). Sygnał zegarowy jest generowany przez wbudowany generator kwarcowy z zewnętrznym rezonatorem. Możliwe jest też dołączenie zewnętrznego przebiegu zegarowego na wejście OSC_IN. HSE może być użyty jako zegar bezpośrednio taktujący rdzeń lub jako źródło zegara dla układu PLL. Częstotliwości generowane mają zakres od 1 MHz do 24 MHz.
- Generator PLL. Częstotliwość odniesienia dla generatora PLL może być wytwarzana przez blok HSI lub HSE. Powinna ona mieścić się w zakresie 2…24 MHz. Generator PLL jest stosowany, gdy jest potrzebna większa częstotliwość taktowania niż mogą to zapewnić bezpośrednio generatory HSI (16 MHz) lub

Listing 1. Funkcja programowania zakresu napięć Vcore

```
/* @param  PWR_VoltageScaling: specifies the voltage scaling range. This parameter can be:
  @arg PWR_VoltageScaling_Range1: Voltage Scaling Range 1 (VCORE = 1.8V)
  @arg PWR_VoltageScaling_Range2: Voltage Scaling Range 2 (VCORE = 1.5V)
  @arg PWR_VoltageScaling_Range3: Voltage Scaling Range 3 (VCORE = 1.2V)
  @retval None
*/
void PWR_VoltageScalingConfig(uint32_t PWR_VoltageScaling)
{
  uint32_t tmp = 0;
  /* Check the parameters */
  assert_param(IS_PWR_VOLTAGE_SCALING_RANGE(PWR_VoltageScaling))
  tmp = PWR->CR;
  tmp &= CR_VOS_MASK;
  tmp |= PWR_VoltageScaling;
  PWR->CR = tmp & 0xFFFFFFF3;
}
```

Listing 2. Kompletna sekwencja programowania regulatora.

```
/* ustawienie regulatora na  1.8V */
  PWR_VoltageScalingConfig(PWR_VoltageScaling_Range1);

  /* Czekaj na ustabilizowanie się napięcia */
  while (PWR_GetFlagStatus(PWR_FLAG_VOS) != RESET) ;
```

Tabela 2. Zależność maksymalnej częstotliwości taktowania od napięcia zasilania

Zakresy napięć	Częstotliwość zegara			
	MSI	HSI	HSE	PLL
Zakres1 (1,8V)	4,2MHz	16MHz	HSE 32MHz(zewnętrzny) Lub 24MHz (kwarc)	32MHz (PLLVCOmax=96MHz)
Zakres2 (1,5V)	4,2MHz	16MHz	16MHz	16MHz (PLLVCOmax=48MHz)
Zakres3 (1,2V)	4,2MHz	----	8MHz	4MHz (PLLVCOmax=24MHz)

HSE (maksymalnie 24 MHz). Generator PLL jest również używany do generowania sygnału zegarowego o częstotliwości 48 MHz dla modułu USB.

- MSI (*Multi Speed Internal*). Sygnał zegarowy jest generowany przez wewnętrzny oscylator RC. Jego częstotliwość można programować przez zapisywanie bitów MSIRANGE[2:0] w rejestrze RCC_ICSCR. Dostępnych jest 7 częstotliwości: 4,193 MHz; 2,097 MHz (wartość domyślna); 1,048 MHz; 524,288 kHz; 262,144 kHz; 131,072 kHz; 65,536 kHz. Blok MSI jest wykorzystywany jako zegar systemowy po zerowaniu, wybudzeniu ze stanu *Stop* i w trybie *Standby Low Power*.

Dodatkowo wykorzystywane są dwa generatory sygnału zegarowego LSE i LSI. LSE jest generatorem stabilizowanym za pomocą zewnętrznego oscylatora kwarcowego o częstotliwości 32768 Hz. Jest on głównie stosowany do taktowania zegara RTC. LSE jest włączany poprzez ustawienie bitu LSEON w rejestrze RCC_CSR, a przez testowanie bitu LSERDY w tym samym rejestrze można stwierdzić czy zegar jest stabilny czy nie.

Generator LSI o częstotliwości ok. 37 kHz jest przewidziany do użycia w trybach niskiego poboru mocy Stop i Standby, do taktowania niezależnego modułu *Watchdog* – IWDG. Podobnie jak LSE, można go włączyć i wyłączyć zmieniając stan bitu LSION i testować stabilność przez odczytywanie bitu LSIRDY w rejestrze RCC_CSR.

Maksymalna częstotliwość taktowania mikrokontrolera jest związana z napięciem Vcore zasilającym rdzeń mikrokontrolera. Tę zależność została opisano w **tabeli 2**.

Przełączanie źródła sygnału zegarowego

Wprowadzanie mikrokontrolera w tryb obniżonego poboru mocy wymaga przełączania źródła sygnału zegarowego na przykład z HSI na MSI. Takie przełączenie jest możliwe, gdy generator sygnału zegarowego, który mamy zamiar użyć, jest stabilny lub pętla PLL jest zsynchronizowana. Na **listingu 3** pokazano funkcję *SetHSICLKToMSI*. Jej argument *freq* ustala częstotliwość zegara MSI. Argument *div2* określa czy wyjściowa częstotliwość MSI ma być dzielona przez 2, argument *With_RTC* określa, czy ma pracować zegar RTC.

Przełączenie zegarów można umownie podzielić na 3 etapy. W pierwszym etapie są wykonywane ustawienia dotyczące pamięci Flash. Potem funkcja przełącza regulator na zakres 3 (1,2 V). W trzecim, końcowym etapie jest ustalana częstotliwość. Następnie funkcja *RCC_SYSCLKConfig* przełącza zegar na MSI. Po przełączeniu program czeka na potwierdzenie użycia MSI jako zegara systemowego – funkcja *RCC_GetSYSCLKSource*. Jak widać na **listingu 3**, przełączenie źródła zegara nie jest łatwe i wymaga wykonania szeregu czynności w określonej kolejności.

Po zakończeniu trybu oszczędzania energii ponownie będziemy potrzebowali dużej mocy obliczeniowej i dlatego znowu trzeba przełączyć taktowanie na źródło HSI. Sposób przełączenia pokazano na **listingu 4**.

Wprowadzanie w tryby oszczędzania energii

Tryby oszczędzania energii są wprowadzane przez specjalne instrukcje: WFI i WFE. Wykonanie instrukcji WFI powoduje, że wybudzenie nastąpi po zgłoszeniu przerwania, a instrukcji WFE po wystąpieniu zdarzenia dla trybów obniżonego poboru mocy *Sleep*. W przypadku trybów *Stop* i *Standby* rodzaj instrukcji nie ma specjalnego znaczenia dla sposobu wybudzenia (**tabela 3**).

Przełączenie w tryb Sleep Low Power

Tryb *Sleep Low Power* jest załączany po ustawieniu regulatora napięcia w na niski pobór energii (ustawienie bitu LPSDSR) oraz wykonanie jednej z 2 instrukcji: WFI lub WFE. Wykonanie instrukcja WFE (*wait for interrupt*) wprowadza tryb *Sleep*, a wybudzenie następuje po zgłoszeniu przerwania. Instrukcja WFE (*wait for event*) również wprowadza stan *Sleep* ale wybudzenie następuje po wystąpieniu zdarzenia.

W trybie *Sleep Low Power* nie jest dostępna pamięć Flash, ale można korzystać z pamięci RAM. Należy pamiętać, że również taktowanie jest ograniczone do częstotliwości oferowanych przez generator MSI, a Vcore musi być ustawione na zakres 2.

Zależnie od wartości bitu *SLEEPONEXIT* z rejestru *Cortex-M3 System Control Register* są możliwe 2 sposoby wejścia w *Sleep Low Power*:

- Sleep-now. Jeśli bit *SLPEEPONEXIT* jest wyzerowany, MCU wprowadza tryb *Sleep* natychmiast po wykonaniu instrukcji WFI lub WFE.
- Sleep-on-exit. Jeśli bit *SLEEPONEXIT* jest ustawiony, to MCU wprowadza tryb *Slepp* natychmiast po zakończeniu obsługi przerwania o najniższym priorytecie. Pozwala to na zakończenie

Tabela 3. Wprowadzanie i wybudzanie z trybów obniżonego poboru mocy

Tryb oszczędzania	Wprowadzenie	Wybudzenie	Regulator napięcia
Low Power run	Bity LPSDSR, LPRUN oraz ustawienie taktowania	Wymuszenie napięcia 1,8V na regulatorze	Low Power mode
Sleep	WFI	Każde przerwanie	ON
	WFE	Każde zdarzenie	
Low Power Sleep	Bity LPSDSR oraz instrukcja WFI	Każde przerwanie	Low Power mode
	Bity LPSDSR oraz instrukcja WFE	Każde zdarzenie	
Stop	Bity PDDS, LPSDSR, SLEEPDEEP, oraz instrukcja WFI lub WFE	Przerwanie zewnętrzne na skonfigurowanej linii EXTI	ON, low Power mode (zależnie od PWR_CR)
Standby	Bity PDDS, SLEEPDEEP, oraz instrukcja WFI lub WFE	Narastające zbocze na linii WKUP, RTC alarm, zewnętrzny restart, restart watchdoga.	OFF

obsługi wszystkich wcześniej zgłoszonych przerwań.

Żeby wejść w tryb *Sleep Low Power* trzeba:

- Wyłączyć pamięć Flash używając bitu SLEEP_PD w rejestrze FLASH_ACR. To redukuje pobór energii, ale wydłuża czas wybudzenia.
- Zablokować zbędne i odblokować każdy niezbędny sygnał zegarowy używając rejestrów RCC_APBxENR i RCC_AHBENR.
- Zmniejszyć częstotliwość zegara systemowego – przełączenie HSI -> MSI.
- Ustawić regulator w trybie obniżonego poboru mocy poprzez ustawienie bitu LPSDSR.
- Wykonać instrukcję WFI, lub WFE.

Wyjście z trybu *Sleep Low Power* jest zależne od instrukcji wejścia. Dla instrukcji WFI wybudzenie następuje po zgłoszeniu przerwania. Jeżeli została użyta instrukcja WFE, to MCU jest wybudzane natychmiast po zaistnieniu zdarzenia. Zdarzenie może być generowane po odblokowaniu przerwania od układów peryferyjnych za pomocą rejestrów sterujących peryferiami, ale bez konfigurowania przerwań rejestrami NVIC. Dodatkowo, trzeba odblokować tę możliwość przez ustawienie bitu SEVONPEND w rejestrze *Cortex-M3 System Control*. Kiedy MCU wychodzi z trybu oszczędzania energii, po wykonaniu WFE, trzeba wyzerować wszystkie bity rejestrów związanych ze zgłoszeniem przerwania.

Zdarzenie może być również generowane po skonfigurowaniu linii EXTI w trybie *event mode*. Wtedy nie jest konieczne zerowanie rejestrów po wyjściu z trybu *Sleep Low Power*.

Na **listingu 5** pokazano procedurę wprowadzającą mikrokontroler w tryb Sleep. Rozpoczynamy od wyłączenia modułu PVD (*Programmable Voltage Detector*) odpowiedzialnego za kontrolowanie napięcia zasilania. Może to zapobiec zgłoszeniu przerwania po obniżeniu napięcia przez regulator i niekontrolowanemu wybudzeniu. Potem jest włączany tryb niskiego poboru energii przez regulator napięcia (funkcja *PWR_UltraLowPowerCmd* z argumentem *ENABLE*). Następnie program zapamiętuje rejestry konfiguracyjne RCC i ustawia konfigurację niskiego poboru energii strukturze *sav_RCC*. Jest to wykonywane przez funkcję *Config_RCC* umieszczoną na **listingu 6**. Ograniczeniu pobieranej energii służy też zatrzymanie systemowego licznika *SysTick*. Przed wprowadzeniem trybu *Sleep* jest przeprogramowywany zegar systemowy. Do tego celu została wykorzystana znana już funkcja przełączenia taktowania z zegara HSI na MSI (listing 3).

Generator MSI jest programowany na generowanie najniższej częstotliwości 65,536 kHz, która jest dodatkowo dzielona przez 2. Dlatego ostatecznie zegar systemowy ma częstotliwość ok. 32 kHz. Wprowadzenie w tryb *Sleep* wykonuje funkcja *PWR_EnterSleepMode* pokazana na **listingu 7**. Jej argumentami są: *PWR_Regulator* określający tryb pracy regulatora i *PWR_SLEEPEntry* określający rodzaj instrukcji, z którą zostanie wprowadzony tryb *Steep*.

Tabela 4. Konfiguracja Sleep-now i Sleep-on-exit

Sleep-now-mode	
Wprowadzenie trybu	Regulator napięcia ustawiony w tryb niskiego poboru mocy, pamięc Flash wyłączona WFI, lub WFE kiedy: SLEPPDEEP=0 , oraz SLEPPONEXIT=0
Wyjście z trybu	Regulator napięcia w tryb normalnej pracy , pamięć flash włączona Jeżeli został użyta komenda WFI, to wybudzenie po zgłoszeniu przerwania Jeżeli został użyta komenda WFE, to wybudzenie po zaistnieniu zdarzenia
Sleep-on-exit	
Wprowadzenie trybu	Regulator napięcia ustawiony w tryb niskiego poboru mocy, pamięc Flash wyłączona WFI, kiedy: SLEPPDEEP=0 , oraz SLEPPONEXIT=1
Wyjście z trybu	Przerwanie

Listing 3. Przełączenie zegara z HSI na MSI

```
void SetHSICLKToMSI(uint32_t freq,bool div2,bool With_RTC)
{
/* RCC system reset */
  RCC_DeInit();
  /* Flash bez opóźnienia*/
  FLASH_SetLatency(FLASH_Latency_0);
/* zablokowanie bufora  Prefetch  */
  FLASH_PrefetchBufferCmd(DISABLE);
/* zablkowanie dostepu 64-bitowego */
  FLASH_ReadAccess64Cmd(DISABLE);
/* zablokowanie  FLASH podczas  SLeep  */
  FLASH_SLEEPPowerDownCmd(ENABLE)
/* Odblokowanie  PWR APB1 Clock */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
/*  Wybranie Voltage Range 3 (1.2V) */
  PWR_VoltageScalingConfig(PWR_VoltageScaling_Range3);
/* czekaj aż  Voltage Regulator  jest gotowy */
  while (PWR_GetFlagStatus(PWR_FLAG_VOS) != RESET);
/* Ustawienie częstotliwości   MSI  */
  RCC_MSIRangeConfig(freq);
/* Ustawienie  MSI jako źródło zegara */
  RCC_SYSCLKConfig(RCC_SYSCLKSource_MSI);
/* Czekaj aż  MSI zostanie uznany jako zegar  SYSCLK*/
  while (RCC_GetSYSCLKSource() != 0x00);
/*czy podzielić częstotliwość przez 2*/
  if (div2)
  {
    RCC_HCLKConfig(RCC_SYSCLK_Div2);
  }
  RCC_HSICmd(DISABLE);
/* Wyłączenie zegara  HSE */
  RCC_HSEConfig(RCC_HSE_OFF);
/* Wyłączenie zegara  LSE  */
  if (! With_RTC) RCC_LSEConfig(RCC_LSE_OFF);
/* Wyłączenie zegara  LSI  */
  RCC_LSICmd(DISABLE);
}
```

Listing 4. Przełączenie taktowania na HSI

```
void SetHSICLK(void)
{
/* odblokuj zegar  HSI  */
  RCC_HSICmd(ENABLE);
/*czekaj aż HSI będzie gotowy */
  while (RCC_GetFlagStatus(RCC_FLAG_HSIRDY) == RESET);
/* odblokuj dostęp  64-bitowy */
  FLASH_ReadAccess64Cmd(ENABLE);
/* odblokuj  bufor Prefetch  */
  FLASH_PrefetchBufferCmd(ENABLE);
/* Flash 1 wait state */
  FLASH_SetLatency(FLASH_Latency_1);
/*konfiguruj zegar HSI jako SYSCLK*/
  RCC_SYSCLKConfig(RCC_SYSCLKSource_HSI);
/*czekaj aż HIS bedzie gotowy */
  while (RCC_GetSYSCLKSource() != 0x04);
/*konfiguruj PLL */
  RCC_HCLKConfig(RCC_SYSCLK_Div1);
/* PCLK2 = HCLK */
  RCC_PCLK2Config(RCC_HCLK_Div1);
/* PCLK1 = HCLK */
  RCC_PCLK1Config(RCC_HCLK_Div1);
}
```

PWR_Regulator może mieć predefiniowaną wartość *PWR_Regulator_ON*, czyli tryb *Sleep* z włączonym regulatorem napięcia lub *PWR_Regulator_LowPower*, czyli tryb *Sleep* z regulatorem napięcia w trybie *Low Power*. Dla naszego trybu *Sleep* użyjemy argumentu *PWR_Regulator_LowPower*. Drugi argument – *PWR_Entry_WFI* – oznacza wejście w tryb *Sleep* z instrukcją WFI.

Przełączenie w tryb Standby

Tryb Standby umożliwia uzyskanie najniższego poboru mocy. Jest oparty o mechanizm *deep sleep* zaimplementowany w rdzeniu Cortex-M3, pozwalający na całkowite wyłączenie regulatora napięcia a w konsekwencji napięcia Vcore. Są wtedy wyłączane oscylatory MSI, HSI, HSE oraz generator z pętlą PLL. Jest też niszczona zawartość pamięci RAM i rejestrów z wyjątkiem rejestrów zegara RTC i rejestrów układu Standby. Tryb Standby jest wprowadzany przez wykonanie instrukcji WFI lub WFE, ale przedtem musza być spełnione następujące warunki:

- Ustawienie bitu *SLEEPDEEP* w rejestrze *Cortex-M3 System Control*.
- Ustawienie bitu *PDDS* w rejestrze *Power Control* (*PWR_CR*).
- Wyzerowanie bitu *WUF* w rejestrze *Power Control*/*Status* (*PWR_CSR*).
- Wyzerowanie flagi źródła wybudzenia związanej z modułem RTC: *RTC Alarm A*, *RTC Alarm B*, *RTC Wakeup*, *Tamper* lub *Time–stamp*.

Mikrokontroler wychodzi ze stany *Standby*, gdy wystąpi:

- Sygnał zerowania zewnętrznego (*Reset*) na wyprowadzeniu NRST.
- Sygnał zerowania od układu Watchdoga IWDG (jeżeli jest uruchomiony).
- Narastające zbocze na wyprowadzeniach WKUP (WKUP1, WKUP2, lub WKUP3).
- Alarm z modułu RTC.
- Zdarzenie tamper lub time-stamp.

W trybie Standby większość wyprowadzeń I/O jest ustawiana w trybie wysokiej impedancji z wyjątkiem wyprowadzenia Reset, wyprowadzenia PC13 skonfigurowanego jako WKUP3, tamper, time-stamp, wyjście RTC Alarm lub wyjście kalibracji RTC oraz wyprowadzenia WKUP1 (PA0) i WKUP3 (PE6), jeżeli są skonfigurowane.

Moduł RTC może być tak zaprogramowany, żeby wybudzał mikrokontroler ze stanu Standby sekwencyjnie, co pewien określony czas. Zegar RTC jest najczęściej taktowany oscylatorem LSE o częstotliwości 32,768 kHz stabilizowanej kwarcem. Można go też skonfigurować, by był taktowany wewnętrznym oscylatorem LSI. Do odmierzania czasu rzeczywistego raczej się wtedy nie nadaje, ale jak najbardziej do okresowego wybudzania ze stanu Standby.

Żeby RTC automatycznie wybudzał (Auto-wakeup AWU) mikrokontroler ze stanu Standby za pomocą zdarzenia alarmu, trzeba go skonfigurować przez:

- Odblokowanie przerwania RTC Alarm w rejestrze RTC_CR.
- Skonfigurować RTC, tak aby generował alarm co zadany czas.

Listing 5. Przełączenie w tryb *Sleep Low Power*

```
/* zablokowanie PVD */
PWR_PVDCmd(DISABLE);
/* Odblokowanie  Ultra low power mode */
PWR_UltraLowPowerCmd(ENABLE);
/* zapamiętanie rejestrów konfiguracyjnych RCC */
Config_RCC(&SavRCC);
/*stop  sys tick */
SysTick->CTRL = 0;
/* Swith in MSI 32KHz */
SetHSICLKToMSI(RCC_MSIRange_0,DIV2,NoRTC);
/*wprowadzenie trybu Sleep Low Power*/
PWR_EnterSleepMode(PWR_Regulator_LowPower,PWR_SLEEPEntry_WFI);
```

Listing 6. Zapamiętanie rejestrów konfiguracyjnych RCC i konfiguracja niskiego poboru energii

```
void Config_RCC(RCC_TypeDef *sav_RCC)
{
  /* zapisanie rejestrów  konfiguracyjnych RCC  */
  sav_RCC->AHBENR   = RCC->AHBENR;
  sav_RCC->APB1ENR  = RCC->APB1ENR;
  sav_RCC->APB2ENR  = RCC->APB2ENR;
  sav_RCC->AHBLPENR   = RCC->AHBLPENR;
  sav_RCC->APB1LPENR  = RCC->APB1LPENR;
  sav_RCC->APB2LPENR  = RCC->APB2LPENR;

  /* ustawienie konfiguracji niskiego poboru energii */
  RCC->AHBENR = 0x05; // Ports A and C enable
  RCC->AHBLPENR = 0x05;
  RCC->APB1ENR =  RCC_APB1ENR_PWREN;     // PWR management enable
  RCC->APB2ENR = 0;
}
```

Listing 7. Wprowadzenie w tryb Sleep

```
void PWR_EnterSleepMode(uint32_t PWR_Regulator, uint8_t PWR_SLEEPEntry)
{
  uint32_t tmpreg = 0;
  assert_param(IS_PWR_REGULATOR(PWR_Regulator));
  assert_param(IS_PWR_SLEEP_ENTRY(PWR_SLEEPEntry));
/*wybór regulatora w trybie Sleep
  tmpreg = PWR->CR;
/* zerowanie bitów PDDS i  LPDSR */
  tmpreg &= CR_DS_MASK;
/* ustawienie bitu  LPDSR stosownie do wartości  PWR_Regulator  */
  tmpreg |= PWR_Regulator;
/* zachowanie nowej wartości  */
  PWR->CR = tmpreg;
/* Zerowanie bitu  SLEEPDEEP rejestru Cortex System Control Register */
  SCB->SCR &= (uint32_t)~((uint32_t)SCB_SCR_SLEEPDEEP);
/* Wybranie wejscia w tryb  SLEEP */
  if(PWR_SLEEPEntry == PWR_SLEEPEntry_WFI)
   {
/*Żądanie Wait For Interrupt */
    __WFI();
  }
   else
  {
    /*Żądanie Wait For Event */
    __WFE();
  }
}
```

Listing 8. Wprowadzenie w tryb Standby

```
/* zablokowanie  PVD */
  PWR_PVDCmd(DISABLE);
/* odblokowanie trybu  Ultra low power */
  PWR_UltraLowPowerCmd(ENABLE);
/* konfiguracja wejścia wakeup - PC13*/
  RTC_OutputTypeConfig(RTC_OutputType_PushPull);
  RTC_OutputConfig(RTC_Output_WakeUp,RTC_OutputPolarity_High);
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13  ;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_400KHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
  GPIO_Init( GPIOC, &GPIO_InitStructure);
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource13,GPIO_AF_RTC_AF1) ;
  Config_RCC(&SavRCC);
/*zablokowanie SysTick*/
  SysTick->CTRL = 0;
/*  MSI =  32KHz */
  SetHSICLKToMSI(RCC_MSIRange_0,DIV2,NoRTC);
/*odblokowanie wybudzania WKUP*/
  PWR_WakeUpPinCmd(PWR_WakeUpPin_1,ENABLE);
/*wejście w tryb Standby*/
  PWR_EnterSTANDBYMode();
```

Listing 9. Wprowadzanie w tryb Standby

```
void PWR_EnterSTANDBYMode(void)
{
/* Wyzerowanie flagi  Wakeup  */
  PWR->CR |= PWR_CR_CWUF;
/* Wybranie trybu  STANDBY */
  PWR->CR |= PWR_CR_PDDS;
/* Ustawienie  bitu SLEEPDEEP w  Cortex System Control
Register */
  SCB->SCR |= SCB_SCR_SLEEPDEEP;

  /* Request Wait For Interrupt */
  __WFI();
}
```

Listing 10. Przykład czynności wykonywanych po wybudzeniu

```
/* zegar HSI*/
  SetHSICLK();
  Config_Systick();
  RCC->AHBENR = SavRCC.AHBENR;
   /* regulator na zakres 1*/
  PWR_VoltageScalingConfig(PWR_VoltageScaling_Range1);
  /* Czekaj na ustalenie napięcia */
  while (PWR_GetFlagStatus(PWR_FLAG_VOS) != RESET) ;
/* odtworzenie rejestrów RCC  */
  RCC->APB1ENR = SavRCC.APB1ENR;
  RCC->APB2ENR = SavRCC.APB2ENR;
  RCC->AHBLPENR = SavRCC.AHBLPENR;
  RCC->APB1LPENR = SavRCC.APB1LPENR;
  RCC->APB2LPENR = SavRCC.APB2LPENR;
/* wyłączenie niskiego poboru mocy regulatora */
  PWR_EnterLowPowerRunMode(DISABLE);
  PWR_UltraLowPowerCmd(DISABLE);
/* Odblokowanie Flash */
  FLASH_SLEEPPowerDownCmd(DISABLE);
/* Zerowanie flagi  Wake Up  */
  PWR_ClearFlag(PWR_FLAG_WU);
/* Odblokowanie  PVD */
  PWR_PVDCmd(ENABLE);
```

Konfiguracja wybudzenia za pomocą RTC Tamper wymaga:

- Odblokowania *RTC Time Stamp Interrupt* w rejestrze *RTC_CR* lub *RTC Tamper Interrupt* w rejestrze *RTC_TCR*.
- Skonfigurowania RTC do wykrywania zdarzeń *Tamper* lub *Time Stamp*.

Możliwe jest też wybudzenie przez wygenerowanie zdarzenia *RTC Wakeup*. Trzeba wtedy:

- Odblokować przerwanie *RTC Wakeup Interrupt* w rejestrze *RTC_CR*.
- Skonfigurować RTC do generowania zdarzenia *RTC Wakeup*.

Na **listingu 8** pokazano fragment programu wprowadzający tryb *Standby*. Przed wykonaniem procedury *PWR_EnterSTANDBYMod* (**listing 9**) wprowadzającej tryb **Standby** regulator napięcia jest wprowadzany w stan niskiego poboru mocy, taktowanie przełączane na 32 kHz i jest konfigurowane wejście wybudzania WKUP2 (PC13).

Po wybudzeniu

Po wybudzeniu musimy przywrócić mikrokontroler do pracy z pełną prędkością i odblokować zablokowane funkcje w celu obniżenia poboru energii. Na **listingu 10** zamieszczono przykładową procedurę wykonywaną po wyjściu mikrokontrolera ze stanu obniżonego poboru mocy. Najpierw jest przywracane taktowanie oscylatorem HSI i jest odblokowywany licznik *SysTick*. Potem w regulatorze napięcia ustawia się zakres 1 o najwyższym dostępnym napięciu zasilania. Konfiguracja taktowania jest uzupełniana o odtworzenie rejestrów RCC. Na końcu jest zerowana flaga Wake-up i odblokowywany moduł PVD.

Tomasz Jabłoński, EP