STM8S001J3 (9)

Tryby obniżonego poboru prądu

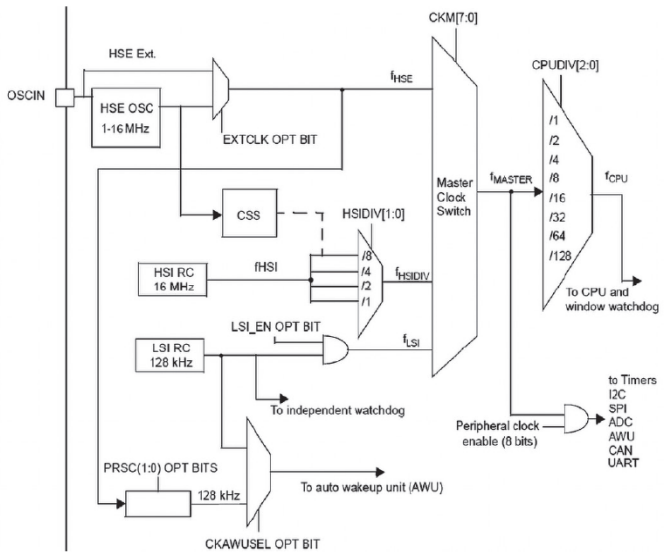
Dziewiąta część z cyklu artykułów dotyczących 8-pinowego układu STM8S001J3 koncentruje się na tematyce związanej z metodami pozwalającymi obniżyć pobór prądu mikrokontrolera. Jak zwykle teorię uzupełniają przykładowe aplikacje.

Optymalne zarządzanie poborem prądu jest zawsze jednym z celów, których osiągnięcie zakłada sobie zespół tworzący urządzenie elektroniczne. Powody, dla których jest to istotny cel, jest wiele.

- Pobór prądu ma kluczowe znaczenie w urządzeniach, dla których źródłem zasilania jest bateria. Im niższy jest pobór prądu, tym dłużej urządzenie może działać bez konieczności ładowania baterii.
- Pobór prądu jest również istotny w kontekście emisji ciepła: niższy pobór prądu oznacza mniejszy stopień nagrzania podzespołów elektronicznych, co niekiedy pozwala uniknąć stosowania rozwiązań chłodzących takich jak radiatory czy wentylatorki.
- Pobór prądu warto ograniczać również z powodów ekologicznych. Urządzenia zużywające mniej prądu są bardziej przyjazne środowisku.

W każdej aplikacji zasadniczo wyróżnić można dwa stany: aktywny, a więc taki, w którym mikrokontroler realizuje przewidziane mu zadania wykorzystując CPU i peryferia, oraz nieaktywny, w którym mikrokontroler nie używa swoich zasobów. W zależności od aplikacji występowanie tych stanów może być nieregularne (a więc ich czas trwania nie jest możliwy z góry do określenia), albo regularne (np. moment przejścia do trybu aktywnego wyznacza układ licznikowy odmierzający czas). Przykład drugiego z opisanych przypadków zilustrowano na **rysunku 1**.

W stanie aktywnym mikrokontroler może ograniczyć zużycie prądu głównie przez zmniejszenie swojej wydajności obliczeniowej, co sprowadza się do zredukowania częstotliwości sygnału taktującego CPU, pamięć i peryferia. Ponadto, peryferia mikrokontrolera, które w danej chwili nie są używane mogą zostać wyłączane wraz z dedykowanymi im sygnałami zegarowymi. W stanie nieaktywnym mikrokontroler może przełączyć się w specjalny tryb pracy, który znacznie ogranicza pobór prądu. W trybie tym zarówno praca CPU, jak też wszystkich sygnałów zegarowych i peryferiów jest zatrzymana, a więc kod aplikacji nie jest wykonywany. Układ wychodzi z tego trybu pracy za pomocą przerwania (zewnętrznego lub wewnętrznego), po czym następuje wznowienie wykonywania kodu aplikacji.



Rysunek 1. Profil aplikacji mikrokontrolera uwzględniający pobór prądu

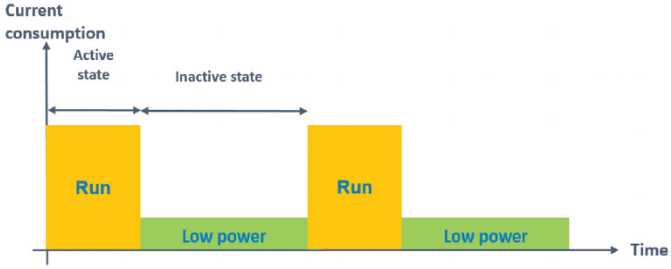
Sposoby ograniczenia poboru prądu układu STM8S001J3 w stanie aktywnym

W stanie aktywnym urządzenia układ STM8S001J3 używa trybu RUN, który pozwala na wykonywanie kodu aplikacji i używanie wszystkich zasobów mikrokontrolera bez żadnych ograniczeń. Zgodnie z tym co napisano we wstępnie, mikrokontroler może zmniejszyć pobór prądu przez redukcję częstotliwości sygnału taktującego CPU, pamięć i peryferia. W przypadku układu STM8S001J3 sygnał ten dostarczać może jeden z bloków: HSI (*High Speed External*) o częstotliwości 16 MHz, HSE (*High Speed Internal*) o częstotliwości 1..16 MHz lub LSI (*Low Speed Internal*) o częstotliwości 128 kHz. Jakie zatem dostępne są możliwości redukcji częstotliwości taktowania?

- W wypadku używania źródła sygnału zegarowego HSE możliwe jest przełączenie się na źródło sygnału zegarowego HSI lub LSI.
- W przypadku używania źródła sygnału zegarowego HSI możliwe jest zmodyfikowanie wartości podzielnika HSIDIV w zakresie 1, 2, 4 i 8. Dodatkowo możliwe jest przełączenie się na źródło sygnału zegarowego LSI.
- W przypadku używania dowolnego źródła sygnału zegarowego (HSE, HSI, LSI) możliwe jest zmodyfikowanie wartości podzielnika CPUDIV w zakresie 1, 2, 4, 8, 16, 32, 64 i 128.

Schemat bloku sygnałów zegarowych pokazujący obrazowo wymienione źródła sygnału zegarowego oraz podzielniki pokazano na **rysunku 2**. Bardziej szczegółowe informacje dotyczące tego zasobu oraz sposobu, w jaki sterować nim w aplikacji można znaleźć w artykule numer 4 z tego cyklu (EP 3/2018). Wartości poboru prądu mikrokontrolera dla kilku wybranych konfiguracji bloku sygnałów zegarowych umieszczono w **tabeli 1**.

Dodatkowo, mikrokontroler STM8S001J3 może ograniczyć pobór prądu przez wyłączenie sygnałów zegarowych taktujących peryferia. Jest to mechanizm PCG (*Peripheral Clock Gating*), który można zastosować dla peryferiów ADC, I²C, AWU, SPI, TIM i UART.



Rysunek 2. Schemat bloku sygnałów zegarowych mikrokontrolera STM8S001J3

Tabela 1. Pobór prądu mikrokontrolera STM8S001J3 w trybie RUN (w warunkach wykonywania kodu z pamięci Flash)

Taktowanie mikrokontrolera	Typowy pobór prądu
fMASTER = HSE = 16 MHz	4,3 mA
fMASTER = HSI = 16 MHz	3,7 mA
fMASTER = HSI/8 = 2 MHz	0,84 mA
fMASTER = LSI (128 kHz)	0,42 mA

Sposoby ograniczenia poboru prądu układu STM8S001J3 w stanie nieaktywnym

Kolejne korzyści pod względem poboru prądu mikrokontroler STM8S001J3 uzyskać może w stanie nieaktywnym. Dostępne są trzy dedykowane tryby pracy: WAIT, HALT oraz ACTIVE HALT.

Mikrokontroler STM8S001J3 przechodzi z trybu RUN do trybu WAIT poprzez wykonanie instrukcji WFI (*Wait For Interrupt*). W efekcie praca CPU zostaje zatrzymana przy jednoczesnej możliwości dalszego działania peryferiów. Zawartość rejestrów oraz pamięci SRAM jest podtrzymana. Mikrokontroler powraca do trybu RUN na skutek przerwania wewnętrznego lub zewnętrznego.

Mikrokontroler STM8S001J3 przechodzi z trybu RUN do trybu HALT poprzez wykonanie instrukcji HALT. W porównaniu do trybu WAIT, w trybie HALT nie tylko praca CPU jest zatrzymana, ale również główny sygnał zegarowy fMASTER oraz wyodrębnione z niego sygnały taktujące peryferia są wyłączone. W rezultacie peryferia nie działają. Zawartość rejestrów oraz pamięci SRAM jest podtrzymana. Mikrokontroler powraca do trybu RUN na skutek przerwania zewnętrznego związanego z portem wejścia/wyjścia:

- skonfigurowanego jako linia GPIO (*General Purpose Input Output*) wytwarzająca przerwanie,
- lub skonfigurowanego jako linia AF (*Alternate Function*) a więc peryferium (np. UART) wytwarzające przerwanie.

W oparciu o tryb pracy HALT powstał dodatkowo tryb ACTIVE HALT. Tryby te różnią się od siebie jedną rzeczą. Wybudzenie mikrokontrolera z trybu ACTIVE HALT następuje nie tylko na skutek przerwania zewnętrznego, ale też wewnętrznego, którego źródłem jest blok AWU (*Auto Wake-Up timer*). AWU jest prostym układem licznikowym, który odmierza czas. Jego schemat blokowy pokazano na **rysunku 3**. Źródłem sygnału taktującego AWU może być albo zegar LSI, albo HSE. W przypadku LSI częstotliwość sygnału to 128 kHz. W przypadku HSE jest to częstotliwość z zakresu 1..16 MHz, przy czym podzielnik bloku AWU redukuje tą wartość do 128 kHz. Oba sygnały (LSI lub zredukowany HSE) podawane są na wejście multipleksera, który wybiera jeden z nich do taktowania AWU. Sygnał na wyjściu multipleksera opcjonalnie może zostać zmierzony przez układ licznikowy TIM1, co w efekcie pozwala na zwiększenie dokładności AWU. Jednocześnie sygnał ten podawany jest na 6-bitowy podzielnik, za którym znajdują się liczniki zliczające impulsy. Wartości liczników porównywane są z wybranym interwałem tworzącym podstawę czasu wybudzania mikrokontrolera. Zakres możliwych do wybrania interwałów to 15 μs...31 s. Gdy na skutek zliczania impulsów sygnału taktującego wartość licznika osiągnie wartość odpowiadającą wybranemu interwałowi czasu, generowane jest przerwanie i następuje wybudzenie mikrokontrolera.

W trybie ACTIVE HALT możliwe jest sterowanie wbudowanym regulatorem napięcia. Gdy regulator jest włączony, mamy do czynienia z trybem FAST ACTIVE HALT, który zapewnia krótszy czas wybudzenia mikrokontrolera, ale jednocześnie wyższy pobór prądu. Z kolei tryb z wyłączonym regulatorem napięcia nosi nazwę SLOW ACTIVE HALT. Pozwala on zmniejszyć pobór prądu do minimum kosztem dłuższego czasu wybudzenia mikrokontrolera.

Listing 1. Wykonany w oparciu o funkcje SPL kod pliku main.c dla aplikacji z trybem HALT

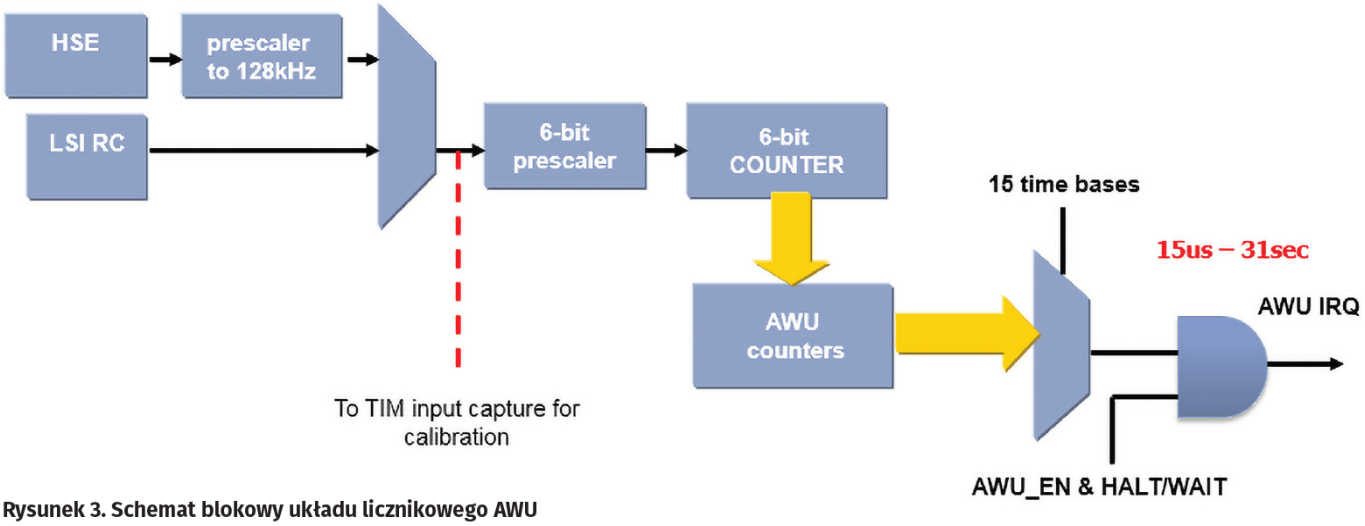
```
#include "stm8s.h"

void delay(unsigned long int how_long);
main()
{
    //delay to avoid irreversible blocking of SWIM
    delay(100000);
    //configuration of unused pins
    GPIOA->DDR |= GPIO_PIN_2;
    GPIOB->DDR |= GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_6 | GPIO_PIN_7;
    GPIOC->DDR |= GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_7;
    GPIOD->DDR |= GPIO_PIN_0 | GPIO_PIN_2 | GPIO_PIN_4 | GPIO_PIN_7;
    GPIOE->DDR |= GPIO_PIN_5;
    GPIOF->DDR |= GPIO_PIN_4;
    //configuration of used pins: PC3 as input (button), PA3 as output (LED)
    GPIO_Init(GPIOC, GPIO_PIN_3, GPIO_MODE_IN_FL_IT);
    GPIO_Init(GPIOA, GPIO_PIN_3, GPIO_MODE_OUT_PP_LOW_FAST);
    //configuration of external interrupt
    EXTI_SetExtIntSensitivity(EXTI_PORT_GPIOC, EXTI_SENSITIVITY_RISE_ONLY);
    //configuration of Flash memory in Power-down mode
    FLASH_SetLowPowerMode(FLASH_LPMODE_POWERDOWN);
    enableInterrupts();
    halt()
    GPIO_WriteHigh(GPIOA, GPIO_PIN_3);
    while (1)
    {
    }
}

void delay(unsigned long int how_long)
{
    unsigned long i;
    for (i = 0; i< how_long; i++)
    {
    }
}
```

Tabela 2. Zestawienie porównawcze trybów pracy mikrokontrolera STM8S001J3 dla stanu nieaktywnego urządzenia

Nazwa trybu pracy	Warunek wejścia	Warunek wyjścia	Stan CPU/ sygnału fMASTER/ peryferiów	Główny regulator napięcia	Pamięć Flash	Czas wybudzenia	Pobór prądu
WAIT	Wywołanie komendy WFI	Przerwanie zewnętrzne lub wewnętrzne	OFF/ON/ON	ON	ON	0,56 µs	1,1 mA @HSE 16 MHz 0,89 mA @HSI 16 MHz 0,4 mA @LSI 128 kHz
HALT	Wywołanie komendy HALT	Przerwanie zewnętrzne	OFF/OFF/OFF	OFF	ON	52 µs	60 µA
					OFF	54 µs	4,5 µA
SLOW ACTIVE HALT		Przerwanie zewnętrzne lub wewnętrzne od AWU	OFF/OFF/OFF oprócz AWU	OFF	ON	48 µs	66 µA
					OFF	50 µs	10 µA
FAST ACTIVE HALT				ON	ON	1 µs	200 µA
					OFF	3 µs	150 µA



Rysunek 3. Schemat blokowy układu licznikowego AWU

We wszystkich trybach HALT (HALT, SLOW ACTIVE HALT, FAST ACTIVE HALT) istnieje możliwość ustawienia pamięci Flash w jeden z dwóch trybów: ON lub OFF (*Power-down*). Podobnie jak w przypadku sterowania regulatorem, konfiguracja pamięci Flash wpływa na czas wybudzenia mikrokontrolera i jego pobór prądu. Pamięć Flash w trybie ON skutkuje krótszym czasem wybudzenia, ale wyższym poborem prądu. Efekt ustawienia pamięci Flash w tryb OFF jest odwrotny.

Funkcje SPL przydatne w obniżaniu poboru prądu mikrokontrolera

Aby w prosty sposób obniżyć pobór prądu mikrokontrolera STM8S001J3, warto w aplikacji użyć bibliotek SPL (*Standard Peripheral Library*) przygotowanych dla mikrokontrolerów z rodziny STM8S. Funkcje przydatne do kontroli bloku zegarowego zostały wymienione i szczegółowo opisane w artykule zamieszczonym w EP 3/2018. Funkcje dedykowane do sterowania układem licznikowym AWU i pamięcią Flash udostępniają odpowiednio pliki *stm8s_awu.h* i *stm8s_awu.c* oraz *stm8s_flash.h* i *stm8s_flash.c*. Zestawienie najważniejszych funkcji zaprezentowano w **tabeli 3**.

Przykładowa aplikacja z trybem HALT i ACTIVE HALT

W celu wykonania przykładowej aplikacji użyte zostało środowisko programistyczne STVD (ST Visual Develop) oraz kompilator Cosmic CXSTM8. Opis tych narzędzi, jak również instrukcja jak stworzyć za ich pomocą szablon nowego projektu wraz z dodaniem bibliotek SPL dostępne są w artykule numer 3 z tej serii (EP 2/2018). Korzystając ze wspomnianego szablonu projektu należy edytować kod pliku *main.c*, w którym umieszczony zostanie kod aplikacji.

Pierwsza przykładowa aplikacja wykorzystuje tryb pracy HALT. Na potrzeby wyjścia z tego trybu i powrotu do trybu RUN port PC3 skonfigurowano w tryb wejściowy z możliwością generowania przerwania. Po wybudzeniu mikrokontroler zmieni stan portu PA3 włączając tym samym diodę LED. W celu zaimplementowania opisanej aplikacji wykonane zostaną następujące kroki w pliku main.c:

- Zostanie utworzona funkcja opóźniająca *delay()*.
- Na początku aplikacji wywołana zostanie funkcja opóźniająca, co przeciwdziała przez kilka sekund ewentualnemu późniejszemu wyłączeniu interfejsu programowania i debugowania SWIM będącemu efektem rekonfiguracji portów.
- Wykonany zostanie kod konfiguracji portów wejścia/wyjścia, które nie są połączone z wyprowadzeniami mikrokontrolera (kod wzięty z noty aplikacyjnej AN5047: *Getting started with the STM8S001J3 microcontroller*).
- Wywołana zostanie funkcja *GPIO_Init()* w celu skonfigurowania portu PC3 jako wejście (z możliwością generowania przerwania) oraz portu PA3 jako wyjście.
- Wywołana zostanie funkcja *EXTI_SetExtIntSensitivity()* ustawiająca aktywne zbocze dla przerwania.
- Wywołana zostanie funkcja *FLASH_SetLowPowerMode()* ustawiająca pamięć Flash w tryb *Power-down*.
- Wywołana zostanie funkcja *enableInterrupts()* włączająca przerwania.
- Wywołana zostanie funkcja *halt()* skutkująca w zmianie trybu pracy mikrokontrolera z RUN na HALT.

Tabela 3. Zestawienie wybranych funkcji z bibliotek dla układu licznikowego AWU, pamięci Flash oraz dodatkowych funkcji do trybów o niskim poborze prądu

Nazwa funkcji	Opis działania funkcji
AWU_DeInit(void)	Konfiguracja domyślna układu licznikowego AWU
AWU_Init(AWU_Timebase_TypeDef AWU_TimeBase)	Konfiguracja interwału czasu, po którym układ licznikowy AWU wybudzi mikrokontroler
AWU_Cmd(FunctionalState NewState)	Włączenie lub wyłączenie układu licznikowego AWU
AWU_LSICalibrationConfig(uint32_t LSIFreqHz)	Uaktualnienie układu licznikowego AWU zmierzoną wartością sygnału LSI
AWU_IdleModeEnable(void)	Włączenie trybu oszczędzania energii w układzie licznikowym AWU
AWU_GetFlagStatus(void)	Odczytanie stanu wybranej flagi układu licznikowego AWU
FLASH_SetLowPowerMode(FLASH_LPMode_TypeDef FLASH_LPMode)	Ustawienie trybu pracy pamięci Flash podczas trybu HALT i ACTIVE HALT
enableInterrupts()	Aktywowanie przerwań
halt()	Wejście w tryb HALT lub ACTIVE HALT
wfi()	Wejście w tryb WAIT

- Po wybudzeniu mikrokontrolera (np. przez wciśnięcie przycisku dołączonego do pinu PC3): wywołana zostanie funkcja *GPIO_WriteHigh()*, która włączy diodę LED dołączoną do pinu PA3.

Kod zgodny z zaprezentowanym opisem pokazano na **listingu 1**.

Druga aplikacja wykorzystuje tryb pracy ACTIVE HALT. W zaprezentowanym przykładzie układ licznikowy AWU wybudzi mikrokontroler z trybu ACIVE HALT po 12 sekundach. Analogicznie do poprzedniej aplikacji, tutaj również po wybudzeniu port PA3 zmieni stan diody LED. W celu zaimplementowania opisanej aplikacji wykonane zostaną następujące kroki w pliku main.c:

- Zostanie utworzona funkcja opóźniająca *delay()*.
- Na początku aplikacji wywołana zostanie funkcja opóźniająca, co przeciwdziała przez kilka sekund ewentualnemu późniejszemu wyłączeniu interfejsu programowania i debugowania SWIM będącemu efektem rekonfiguracji portów.
- Wykonany zostanie kod konfiguracji portów wejścia/wyjścia, które nie są połączone z wyprowadzeniami mikrokontrolera (kod wzięty z noty aplikacyjnej AN5047: *Getting started with the STM8S001J3 microcontroller*).
- Wywołana zostanie funkcja *GPIO_Init()* w celu skonfigurowania portu PA3 jako wyjście.
- Wywołana zostanie funkcja *FLASH_SetLowPowerMode()* ustawiająca pamięć Flash w tryb *Power-down*.
- Ustawiony zostanie bit SWUAH w rejestrze ICKR w celu wyłączenia regulatora napięcia w trybie ACTIVE HALT.
- Wywołana zostanie funkcja *AWU_DeInit()* w celu domyślnego skonfigurowania układu licznikowego AWU.
- Wywołana zostanie funkcja *AWU_Init()* w celu ustawienia 12-sekundowego interwału wybudzania mikrokontrolera przez układ licznikowy AWU.
- Wywołana zostanie funkcja *AWU_Cmd()* włączająca układ licznikowy AWU.
- Wywołana zostanie funkcja *enableInterrupts()* włączająca przerwania.
- Wywołana zostanie funkcja *halt()* skutkująca w zmianie trybu pracy mikrokontrolera z RUN na ACIVE HALT.
- Po wybudzeniu mikrokontrolera: wywołana zostanie funkcja *GPIO_WriteHigh()*, która włączy diodę LED.

Kod zgodny z zaprezentowanym opisem pokazano na **listingu 2**.

Listing 2. Wykonany w oparciu o funkcje SPL kod pliku main.c dla aplikacji z trybem ACTIVE HALT

```
#include "stm8s.h"

void delay(unsigned long int how_long);
main()
{
    //delay to avoid irreversible blocking of SWIM
    delay(100000);
    //configuration of unused pins
    GPIOA->DDR |= GPIO_PIN_2;
    GPIOB->DDR |= GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_6 | GPIO_PIN_7;
    GPIOC->DDR |= GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_7;
    GPIOD->DDR |= GPIO_PIN_0 | GPIO_PIN_2 | GPIO_PIN_4 | GPIO_PIN_7;
    GPIOE->DDR |= GPIO_PIN_5;
    GPIOF->DDR |= GPIO_PIN_4;
    //GPIO configuration
    GPIO_Init(GPIOA, GPIO_PIN_3, GPIO_MODE_OUT_PP_LOW_FAST);
    //Consumption optimisations: Flash and regulator
    FLASH_SetLowPowerMode(FLASH_LPMODE_POWERDOWN);
    CLK->ICKR |= CLK_ICKR_SWUAH;
    //AWU configuration
    AWU_DeInit();
    AWU_Init(AWU_TIMEBASE_12S);
    AWU_Cmd(ENABLE);
    enableInterrupts();
    halt();
    GPIO_WriteHigh(GPIOA, GPIO_PIN_3);
    while (1)
    {
    }
}

void delay(unsigned long int how_long)
{
    unsigned long int i;
    for (i = 0; i< how_long; i++)
    {
    }
}
```

Podsumowanie

W artykule przekazano informacje o rozwiązaniach pozwalających obniżyć pobór prądu układu STM8S001J3. W stanie aktywnym urządzenia mikrokontroler wykorzystuje tryb RUN i może obniżyć pobór prądu kosztem wydajności obliczeniowej (niższa częstotliwość sygnału taktującego) i poprzez wyłączanie sygnałów zegarowych dla peryferiów. Z kolei w stanie nieaktywnym mikrokontroler może wejść w tryb WAIT, HALT lub ACTIVE HALT. Ponadto w trybie HALT i ACTIVE HALT możliwe jest wyłączenie pamięci Flash, a tryb ACTIVE HALT może dodatkowo sterować regulatorem napięcia. Opis wszystkich zaprezentowanych rozwiązań uzupełniono o przykładowe aplikacje.

Osoby chcące dowiedzieć się bardziej szczegółowych informacji powinny sięgnąć do dokumentacji technicznej producenta. Krótki opis trybów pracy mikrokontrolera, układu licznikowego AWU oraz dane odnośnie pobieranego prądu i czasu wybudzenia można znaleźć w nocie katalogowej mikrokontrolera (*datasheet*). Z kolei schematy oraz szczegółowy opis wszystkich funkcjonalności i rejestrów wymienionych zasobów znajdują się w podręczniku użytkownika mikrokontrolera (*user manual* RM0016). Dodatkowo aplikacja testowa dostępna jest w dedykowanym podkatalogu bibliotek SPL: …\STM8S_StdPeriph_Lib\Project\STM8S_StdPeriph_Examples\AWU.

Szymon Panecki
szymon.panecki@st.com