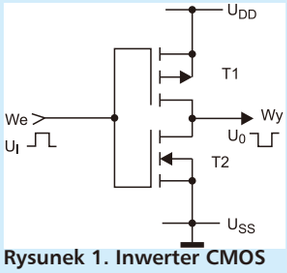
STM32 – tryby obniżonego poboru mocy

Temat obniżonego poboru mocy pobieranej przez mikrokontrolery pojawia się co jakiś czas w różnych publikacjach. Obecnie stał się bardziej „medialny" za sprawą coraz bardziej powszechnego wykorzystywania samartfonów i tabletów. Są to urządzenia, od których wymaga się coraz większej wydajności przy zasilaniu bateryjnym. Wydajność wiąże się z bezpośrednio z szybkością działania wbudowanych mikroprocesorów, ale jak wiadomo – im jest wyższa częstotliwość taktowania mikroprocesora, tym więcej mocy pobiera on ze źródła zasilania.

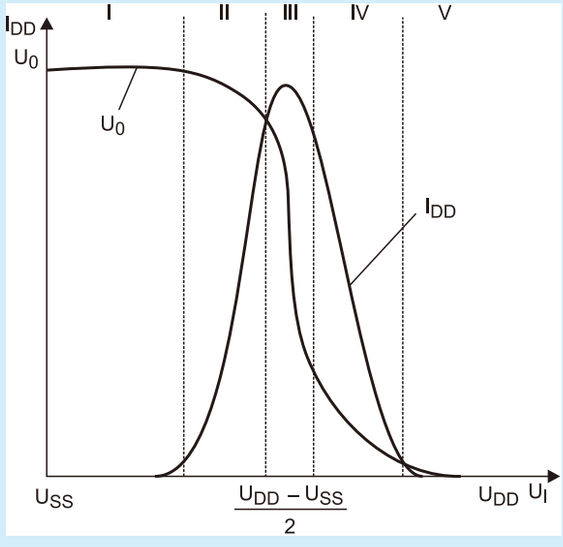
Można próbować pogodzić większe zapotrzebowanie na moc z czasem pracy z baterii poprzez zwiększanie jej pojemności. Jednak postęp w konstruowaniu nowych ogniw jest niezbyt szybki. Mamy do dyspozycji ogniwa wynalezione wiele lat temu. Na przykład najnowsze ogniwa litowo jonowe zostały wynalezione w firmie Sony na początku lat 90-tych XX wieku. Pomimo usilnych prób nie bardzo można znacząco zwiększyć pojemność baterii bez zwiększania ich ciężaru i wyzmiarów, a to niezbyt dobre rozwiązanie dla urządzeń przenośnych. Pozostaje próbować zagwarantować oczekiwaną wydajność mikroprocesora i jego otoczenia przy ograniczaniu poboru mocy. Cała nadzieja w postępie technologicznym i pomysłowości projektantów układów cyfrowych.

Niemal wszystkie układy cyfrowe są wykonywane w technologii CMOS. Głównym elementem w tej technologii jest komplementarna para tranzystorów – jeden z kanałem N, drugi z kanałem P (**rysunek 1**). Taka para tworzy logiczny inwerter. Jeżeli napięcie wejściowe jest odpowiednio niskie (poziom L), to tranzystor z kanałem N jest zablokowany, a tranzystor z kanałem P przewodzi. Napięcie wyjściowe jest prawie równe napięciu zasilania Udd. Przy napięciu wejściowym zbliżonym do poziomu logicznego wysokiego sytuacja jest odwrotna i na wyjściu występuje napięcie równe w przybliżeniu Uss. W stanach ustalonych prąd pobierany przez inwerter jest bardzo mały, bo jeden z tranzystorów jest zablokowany. Kluczowy dla poboru mocy jest stan przejściowy.



Rysunek 1. Inwerter CMOS

Na **rysunku 2** pokazano charakterystykę przejściową inwertera podzieloną na 5 obszarów. Obszar I oraz V to stany, w których jeden z tranzystorów nie przewodzi. Jednak, gdy napięcie wejściowe Ui zaczyna narastać, oba tranzystory przechodzą w stan pracy liniowej i ze źródła zasilania zaczyna płynąc prąd. Ten prąd osiąga najwyższą wartość w obszarze III. Z charakterystyki pokazanej na rys. 2 wynika jeden wniosek: im częściej będą następowały zmiany stanów inwertera, tym więcej prądu będzie on pobierał. Takie twierdzenie zgadza się z ogólnie znaną zależnością zwiększania poboru mocy przy zwiększaniu częstotliwości taktowania.



Rysunek 2. Charakterystyka przejściowa inwertera CMOS

Z częstotliwością taktowania wiąże się też problem przeładowywania pojemności pasożytniczych połączeń pomiędzy tranzystorami i pojemności samych tranzystorów. Im większe pojemności i częstotliwość, tym jest większa potrzebna energia do ich przeładowania. Tu ratunkiem jest technologia wykonywania układów scalonych. Obecnie wykonywane tranzystory i połączenia pomiędzy nimi są coraz mniejsze, a co za tym idzie – również pojemności pasożytnicze.

Czy można coś z tym zrobić? Można – stosując zasadę „pracuj na pełnych obrotach, kiedy jest to niezbędne, ale odpoczywaj, jeżeli tylko możesz, kiedy nie masz zbyt wiele albo nic do zrobienia". Stosując różne tryby oszczędzania energii można zaoszczędzić jej wiele w trybach czuwania. Daje to bardzo dobre rezultaty, szczególnie w urządzeniach konsumenckich pracujących w trybie ciągłym (np. smartfony). Nawet najbardziej zagorzały fan SMS-owania i harcowania po facebooku kiedyś musi spać lub jeść.

Jeżeli popatrzymy na krzywą poboru prądu z rys. 2, to widać, że jego wartość maksymalna zależy od napięcia poziomu wysokiego, czyli od napięcia zasilania. Z tej zależności wynika prosty wniosek: im niższe napięcie zasilania układu, tym mniejszy pobór mocy. Każdy, kto śledzi rozwój układów cyfrowych, a szczególnie mikroprocesorowych, zna tendencję do obniżania napięcia zasilania. Kiedyś standardem było napięcie +5 V – były nim zasilane układy TTL i mikroprocesory wykonywane w technologii N-MOS, potem zasilano nim układy CMOS. Od

Tabela 1. Pobór prądu w trybie Run.

Warunki pomiaru		Fhclk	Typowo	Maksymalnie			jednostka
				55°C	85°C	105°C	
fhse=fhclk do 8 MHz fhse=1/2 fhclk powyżej 8 MHz (PLL ON)	Zakres 3 Vcore=1,2V VOS[1:0]=11	1MHz	270	400	400	400	µA
		2MHz	470	600	600	600	
		4MHz	890	1025	1025	1025	mA
	Zakres 2 Vcore=1,5V VOS[1:0]=10	4MHz	1	1,3	1,3	1,3	
		8MHz	2	2,5	2,5	2,5	
		16MHz	3,9	5	5	5	
	Zakres1 Vcore=1,8V VOS[1:0]=01	8MHz	2,16	3	3	3	
		16MHz	4,8	5,5	5,5	5,5	
		32MHz	9,6	11	11	11	
Zegar HSI (16 MHz)	Zakres2 Vcore=1,5V	16MHz	4	5	5	5	
	Zakres 1 Vcore=1,8V	32MHz	9,4	11	11	11	
Zegar MSI 65 kHz	Zakres 3 Vcore=1,2V	65kHz	50	85	90	100	µA
Zegar MSI 524 kHz		524kHz	150	185	190	200	
Zegar MSI 4,2 MHz		4,2MHz	900	1000	1000	1000	

jakiegoś czasu układy cyfrowe – a w tym mikrokontrolery – są zasilane napięciem +3,3 V. Ale to napięcie może być obniżane w strukturze układu do 1,8 V lub mniej. Tym obniżonym napięciem jest zasilany najbardziej energożerny blok – rdzeń. Jak dowiemy się dalej, nie są to jedyne manipulacje napięciem zasilającym. Zaawansowane mechanizmy obniżania poboru mocy potrafią dynamicznie dostosowywać go do bieżących potrzeb. Użytkownicy mikrokontrolerów w aplikacjach przemysłowych muszą pamiętać o tym, że im niższe napięcie zasilania, tym mniejsza odporność na zburzenia. Stosowanie układów o obniżonym napięciu zasilania wymaga szczególnej staranności przy projektowaniu układów zasilania.

STM32L152RBT6

Nowoczesne, zoptymalizowane pod kątem oszczędzania energii mikrokontrolery dają konstruktorowi wiele narzędzi do efektywnego obniżenia poboru mocy ze źródła zasilania. Jednym z przykładów takiego mikrokontrolera jest 32-bitowy STM32L152RBT6 z rdzeniem Cortex M-3. Oprócz wielu trybów oszczędzania energii, ma on możliwość zasilania napięciem z zakresu 1,65…3,6 V. Kod 152RBT6 oznacza wbudowany sterownik wyświetlacza LCD, obudowę LQFP o 64 wyprowadzeniach, 128 kB pamięci programu Flash i przemysłowy zakres temperatury pracy -40…+85°C. Programista, oprócz dużej pamięci Flash, ma do dyspozycji 16 kB RAM, 4 kB pamięci EEPROM i 80-bajtowy rejestr backup. Układ jest dobrze wyposażony w peryferia analogowe: 12-bitowy A/C, 12-bitowy D/C z buforem wyjściowym oraz podwójny komparator. Mogące one pracować już od napięcia zasilania +1,8 V, z ograniczeniem prędkości przy niskim napięciu zasilającym. Imponująca jest lista wbudowanych interfejsów komunikacyjnych: USB, 2×SPI, 3×USART i 2×I²C. Dziesięć wbudowanych timerów 16-bitowych, w tym część z kanałami IC/OC/PWM, powinno zaspokoić zapotrzebowanie nawet bardzo rozbudowanych aplikacji.

Mikrokontroler może być taktowany zegarem o częstotliwości zmienianej w szerokim zakresie – od 32 kHz do 32 MHz. Główny sygnał zegarowy może być dostarczany z 3 głównych źródeł:

- Zegar HSE o częstotliwości 1…24 MHz z oscylatora z zewnętrznym kwarcem. Sygnał z tego zegara może być podawany na układ PLL.
- Wewnętrzny, kalibrowany programowo oscylator RC HSI o częstotliwości 16 MHz. Jego sygnał może być podawany na układ PLL.

Tabela 2. Pobór mocy dla trybu Sleep

Warunki pomiaru: kod wykonywany z pamięci Flash lub RAM		Fhclk	Typowo	Maksymalnie			Jednostka
				55°C	85°C	105°C	
HSE=16MHz PLL ON i fhclk<16 MHz	Zakres 3 Vcore=1,2 V VOS[1:0]=11	1 MHz	80	140	140	140	µA
		2 MHz	150	210	210	210	
		4 MHz	280	330	330	330	
	Zakres 2 Vcore=1,5 V VOS[1:0]=10	4 MHz	280	400	400	400	
		8 MHz	450	550	550	550	
		16 MHz	900	1050	1050	1050	
	Zakres1 Vcore=1,8 V VOS[1:0]=01	8 MHz	550	650	650	650	
		16 MHz	1050	1200	1200	1200	
		32 MHz	2300	2500	2500	2500	
Zegar HSI (16 MHz)	Zakres2 Vcore=1,5 V	16 MHz	1000	1100	1100	1100	
	Zakres 1 Vcore=1,8 V	32MHz	2300	2500	2500	2500	
Zegar MSI 65 kHz	Zakres 3 Vcore=1,2 V	65kHz	30	50	50	50	
Zegar MSI 524 kHz		524kHz	50	70	70	70	
Zegar MSI 4,2 MHz		4,2MHz	200	240	240	240	

Tabela 3. Pobór mocy w trybie Low Power run

Warunki pomiaru			Typowo	Maksymalnie	Jednostka
Wszystkie układy peryferyjne wyłączone, Vdd = 1,65…3,6V	MSI 65 kHz fHCLK=32 kHz	TA=-40°C…+25°C	9	12	µA
		TA=85°C	17,5	24	
		TA=105°C	31	46	
	MSI 65 kHz fHCLK=65 kHz	TA=-40°C…+25°C	14	17	
		TA=85°C	22	29	
		TA=105°C	35	51	
	MSI 131 kHz fHCLK=131 kHz	TA=-40°C…+25°C	37	42	
		TA=85°C	37	42	
		TA=105°C	48	61	

Tabela 4. Pobór mocy w trybie low Power sleep

Warunki pomiaru			Typowo	Maksymalnie	Jednostka
Wszystkie układy peryferyjne wyłączone, Vdd = 1,65…3,6V	MSI 65kHz fHCLK=32 kHz	TA=-40°C…+25°C	17,5	20	µA
		TA=85°C	22	27	
		TA=105°C	31	39	
	MSI 65 kHz fHCLK=65 kHz	TA=-40°C…+25°C	18	26	
		TA=85°C	23	28	
		TA=105°C	31	41	
	MSI 131 kHz fHCLK=131 kHz	TA=-40°C…+25°C	22	30	
		TA=85°C	26	34	
		TA=105°C	34	35	

Tabela 5. Pobór prądu w trybie Standby

Warunki pomiaru			Typowo	Maksymalnie	Jednostka
Z pracującym modułem RTC	RTC taktowany przez LSI	TA=-40°C…+25°C	1,1	1,8	µA
		TA=85°C	1,87	3	
		TA=105°C	2,78	5	
	RTC taktowany przez LSE	TA=-40°C…+25°C	1,33	2,9	
		TA=85°C	2,01	4,3	
		TA=105°C	3,27	6,3	
Z Wyłączonym modułem RTC	Wyłączony watchdog i LSI	TA=-40°C…+25°C	0,3	0,55	
		TA=85°C	1	1,7	
		TA=105°C	2,5	4	

Tabela 6. Ograniczenia funkcjonalności z zależności od napięcia zasilania

Napięcie zasilania	Zakres dynamicznej zmiany napięcia zasilającego	Moduły DAC i ADC	Moduł USB	Linie I/O
Vdd=1,65…1,8V	Zakres 2 lub zakres 3	Nie działa	Nie działa	Ograniczona prędkość
Vdd=1,8…2V	Zakres 2 lub zakres 3	Prędkość konwersji do 500Ksps	Nie działa	Ograniczona prędkość
Vdd=2,0V…2,4	Zakresy 1, 2 lub 3	Prędkość konwersji do 500Ksps	Działa	Pełna prędkość działania
Vdd=2,4….3,6V	Zakresy 1, 2 lub 3	Prędkość konwersji do 1Msps	Działa	Pełna prędkość działania

- Wewnętrzny, kalibrowany programowo oscylator RC MSI, który może generować 7 częstotliwości: 65,6 kHz, 131 kHz, 262 kHz, 524 kHz, 1,05 MHz, 2,1 MHz i 4,2 MHz. Kiedy jest dostępny zewnętrzny sygnał o częstotliwości 32768 Hz, to MSI można kalibrować programowo z tolerancją ±0,5%.

Do taktowania zegara RTC i modułu LCD wykorzystuje się dwa źródła częstotliwości o bardzo małym poborze prądu: wspomniany, kwarcowy oscylator o częstotliwości 32768 Hz nazywany LSE i wewnętrzny oscylator RC o częstotliwości 37 kHz nazywany LSI, typowo wykorzystywany do taktowania wewnętrznego watchdoga. Ponieważ mikrokontroler ma wbudowany interfejs USB, do jego taktowania jest przeznaczony specjalny układ z wbudowanym blokiem PLL, gwarantujący taktowanie z częstotliwością 48 MHz.

Na uwagę z punktu widzenia poboru mocy zasługuje moduł zegara RTC. Oprócz pełnienia głównej funkcji zegara/kalendarza, RTC może generować cykliczne przerwania i alarmy do wybudzania mikrokontrolera z trybów *Stop* lub *Standby*. Można w ten sposób wybudzać mikrokontroler w cyklach czasowych od 120 µs do 36 godzin. Sam RTC w trybie oszczędzania Stop pobiera bardzo mały prąd, jeżeli jest taktowany oscylatorem LSI: 1,2 µA przy zasilaniu 1,8 V i 1,4 µA przy zasilaniu 3,0 V.

Na przykładzie mikrokontrolera STM32L152RBT6 pokażę możliwości optymalizowania poboru mocy w nowoczesnym, wydajnym mikrokontrolerze.

Optymalizacja napięcia zasilania

Rodzina mikrokontrolerów STM32L15xxx wspiera mechanizm dynamicznej zmiany napięcia zasilania w celu

optymalizacji poboru prądu. Regulację wykonuje wbudowany w strukturę układu, programowany regulator LDO. Napięcie zasilania może być ustawiane w zależności od maksymalnej częstotliwości taktowania i jest podzielone na 3 zakresy:

- **Zakres 1**. Napięcie zasilania jest w granicach 2,0…3,6 V. Mikrokontroler może pracować z maksymalną częstotliwością do 32 MHz.
- **Zakres 2**. Pełny zakres napięcia zasilania. Mikrokontroler jest taktowany z częstotliwością do 1 MHz.
- **Zakres 3**. Pełny zakres napięcia zasilania. Mikrokontroler jest taktowany częstotliwością do 4 MHz generowaną przez wewnętrzny oscylator RC multispeed MSI.

W **tabeli 1** umieszczono zestawienie poboru prądu w trybie pracy *Run*.

Tryby oszczędzania energii

Oszczędzanie energii jest zawsze ko,mpromisem pomiędzy jak najniższym poborem mocy, szybkością (czasem) startu do normalnej pracy i dostępnymi źródłami wybudzenia. W trybach najniższego poboru jest zatrzymywany zegar taktujący rdzeniem i układami peryferyjnymi. Wyjście z tego stanu wymaga czasu potrzebnego do wystartowania generatora i ustabilizowania jego drgań. Zatrzymanie taktowania peryferii ogranicza również liczbę źródeł zdolnych do wybudzenia układu.

W mikrokontrolerze z rodziny STM32L152 jest dostępnych 7 trybów oszczędzania energii:

- Sleep mode. W tym trybie jest zatrzymywany tylko rdzeń (CPU). Wszystkie układy peryferyjne pracują normalnie i mogą wybudzić CPU po zgłoszeniu przerwania (**tabela 2**).
- *Low Power Run*. Ten tryb jest wprowadzany, gdy mikrokontroler jest taktowany wewnętrznym zegarem RC (MSI) o częstotliwości minimum 65 kHz. Program jest wykonywany z pamięci Flash lub RAM, a wewnętrzny regulator napięcia jest ustawiony na niski pobór mocy (*Low Power Mode*). W tym trybie częstotliwość taktowania i dostępność części układów peryferyjnych jest ograniczana (**tabela 3**).
- *Low Power Sleep Mode*. Ten tryb jest aktywny po wprowadzeniu trybu *sleep* i jednoczesnym ustawieniu regulatora napięcia na tryb niskiego poboru energii. W Low Power sleep mode są ograniczenia częstotliwości taktowania i ograniczenia w stosowaniu układów peryferyjnych. Wybudzenie z tego trybu następuje po wykryciu przerwania lub zdarzenia (**tabela 4**).
- *Tryb Stop z RTC*. W tym trybie jest podtrzymywana zawartość pamięci RAM, rejestrów CPU i działa zegar RTC. Wszystkie sygnały zegarowe taktujące rdzeń są wyłączone. Nie pracują układy PLL, MSI RC, HSI RC i HSE oraz generator kwarcowy. Zegar LSE lub LSI pracuje, a układ zasilania jest ustawiony w tryb Low Power. Mikrokontroler może być wybudzony z trybu Stop przez wejście przerwania zewnętrznego EXTI przez 8 µs. Źródłem tego przerwania może być jedna z 16 linii zewnętrznego przerwania, zdarzenie od komparatorów 1, lub 2, oraz zdarzenia od timera RTC i modułu USB.
- *Tryb Stop bez zegara RTC*. Działa jak *Stop z RTC*, tylko jest wyłączony zegar RTC – nie pracuje oscylator LSE lub LSI. Źródła wybudzeń są również takie same, z wyłączeniem zdarzeń od zegara RTC.
- *Tryb Standby z zegarem RTC*. W tym trybie rdzeń mikrokontrolera oraz wewnętrzny regulator napięcia są wyłączone – rdzeń nie jest zasilony. Nie pracują układy PLL, MSI RC, HSI RC i HSE oraz generator kwarcowy, ale zegar LSE lub LSI pracuje, bo pracuje moduł RTC. Po wejściu w *Standby* zawartość pamięci RAM i rejestrów CPU jest niszczona, z wyjątkiem rejestrów sprzętowego modułu *Standby*. Mikrokontroler jest wybudzany z trybu Standby przez czas 60 µs poprzez zewnętrzny sygnał *Reset* (wyprowadzenie NRST), restart wymuszony przez watchdog (IWDG), narastającym zboczem na 3 pinach WKUP lub zdarzeniami od timera RTC.
- *Tryb Standby bez zegara RTC*. W tym trybie rdzeń mikrokontrolera oraz wewnętrzny regulator napięcia są wyłączone. Nie pracują układy PLL, MSI RC, HSI RC i HSE oraz generator kwarcowy. Zegar LSE lub LSI nie pracuje, bo nie pracuje moduł RTC. Po wprowadzeniu w *Standby* zawartość pamięci RAM i rejestrów CPU jest niszczona, z wyjątkiem rejestrów sprzętowego modułu *Standby*. Mikrokontroler jest wybudzany z trybu Standby przez czas 60 µs dzięki zewnętrznego sygnałowi *Reset* (wyprowadzenie NRST), restart wymuszony przez watchdog (IWDG), narastającym zboczem na 3 pinach WKUP (tabela 5).

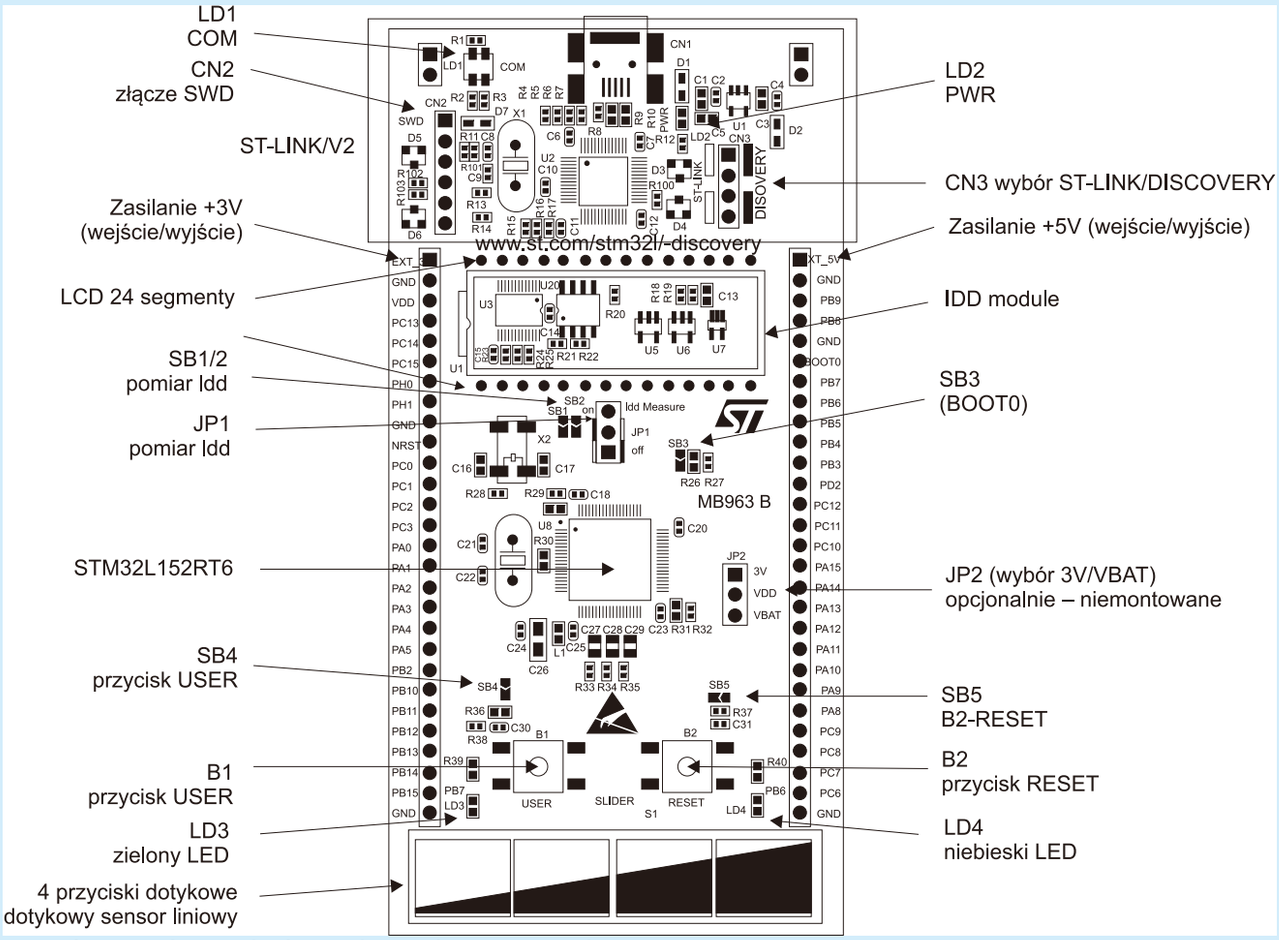
Jak wspomniałem, tryby obniżonego poboru mocy mogą powodować ograniczenia w pracy układów peryferyjnych. W **tabeli 6** umieszczono zestawienie ograniczeń działania modułów analogowych DAC i ADC, interfejsu USB i portów I/O w zależności od napięcia zasilania w trybach 1, 2 i 3.

STM32L Discovery

Wybór mikrokontrolera tego typu nie był przypadkowy, ponieważ producent – firma STM – oferuje tani zestaw ewaluacyjny *STM32L Discovery* zawierający *STM32L152RBT6* i kilka dodatkowych układów przydatnych do testowania niskiego poboru mocy. Na płytce (**fotografia 3**) umieszczono:

- Mikrokontroler STM32L152RBT6.
- Moduł linkera/debugera ST-LINK/V2. Ten moduł może być zastosowany do programowania dowolnego mikrokontrolera przez dodatkowe złącze SWD.
- Układ zasilania z możliwością zasilania z portu USB, oraz z zewnętrznego źródła o napięciu +5 V, lub +3,3 V.
- Układ pomiary prądu zasilania Idd.
- Wyświetlacz LCD (bez własnego sterownika).

Fotografia 3. Moduł STM32L Discovery



Rysunek 4. Rozmieszczenie elementów na płytce STM32L Discovery

- Cztery diody LED.
- Dwa przyciski: Reset i User.
- Liniowy sensor dotykowy.
- Cztery przyciski dotykowe.
- Listwy goldpinów połączone z wyprowadzeniami mikrokontrolera.

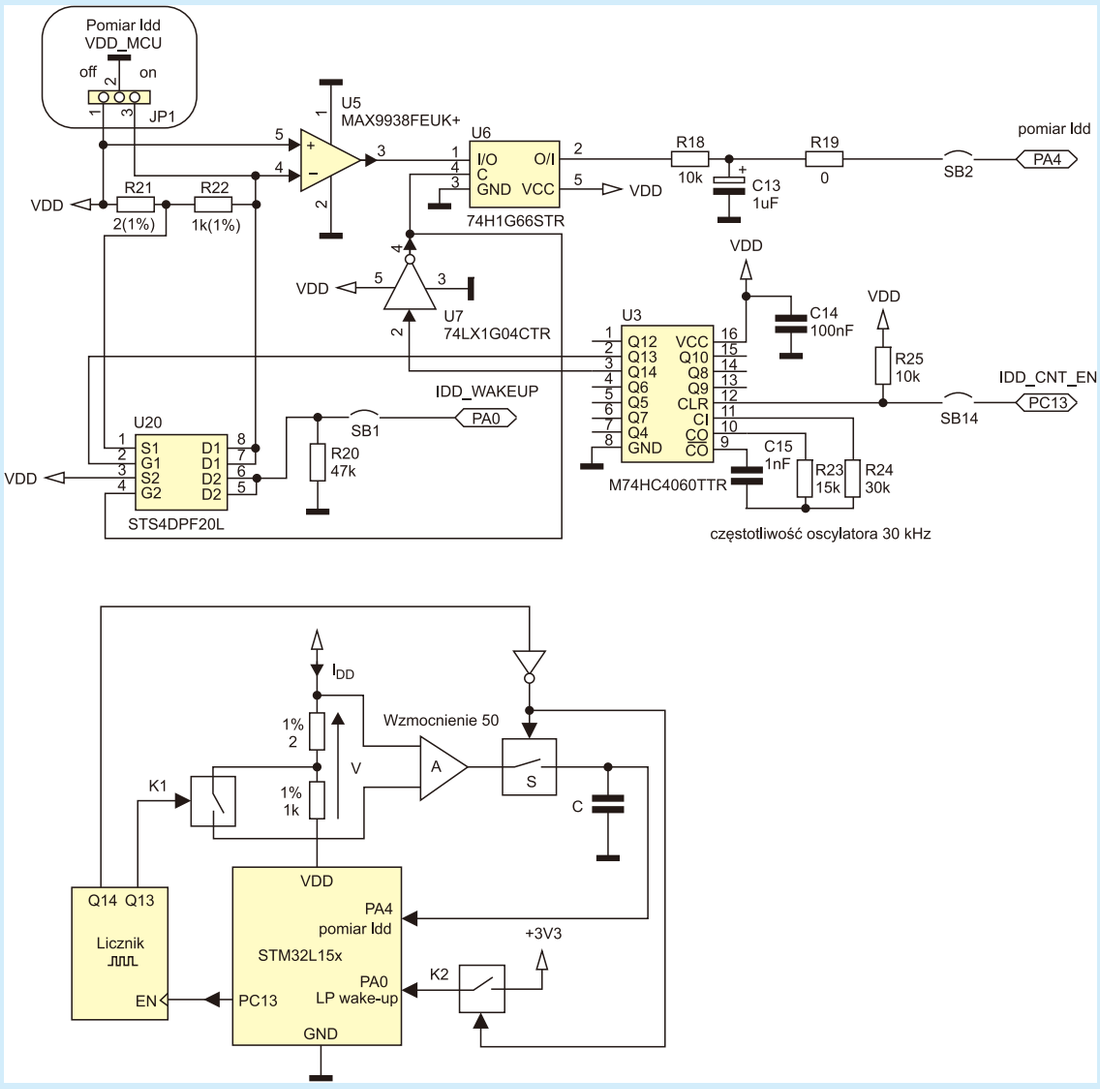
Układ do pomiaru natężenia prądu zasilania

Podczas testowania trybów obniżonego poboru mocy najbardziej naturalnym jest pomiar prądu pobieranego przez mikrokontroler. Można do tego celu wykorzystać multimetr, jednak konstruktorzy *STM32L Discovery* postanowili wbudować układ pomiarowy na płytce, aby natężenie mierzonego prądu mogło być wyświetlane na wyświetlaczu LCD. To na pewno podnosi atrakcyjność modułu, ale kosztem rozbudowy części sprzętowej. Wiąże się to też z rozbudowanym programem testowym, który musi mierzyć prąd i wyświetlać jego wartość. Jeżeli jednak mamy gotowy, zaprojektowany i uruchomiony moduł do pomiaru z wszystkimi działającymi obwodami sprzętowymi i przygotowanym przez producenta programem testowym, to nic nie stoi na przeszkodzie, aby z tego skorzystać. Schemat układu pomiaru prądu pokazano na **rysunku 5**.

Układ do pomiaru prądu można włączać i wyłączać zworką JP1. Kiedy jest ona w położeniu ON, to mikrokontroler jest zasilany przez układ pomiaru prądu. Położenie zworki w pozycji OFF powoduje ominięcie układu pomiaru. Po całkowitym usunięciu zworki, pobór prądu można mierzyć amperomierzem włączonym pomiędzy piny 1 i 2 złącza JP1.

Układ pomiarowy prądu pracuje w 2 zakresach: „niskim” – do 60 µA i „wysokim” – do 30 mA. Pomiar w zakresie „wysokim” jest wykonywany przez mierzenie spadku napięcia na rezystorze R21 (2 Ω/1%). To napięcie jest wzmacniane 50 razy przez wzmacniacz operacyjny U5 (MAX9938). W układzie pomiarowym rezystor R22 jest zwierany przez tranzystor FET1 układu U20 sterowany stanem wyjścia Q13 licznika 74HC4060. Poziom wysoki na IDD_CNT_EN powoduje zerowanie licznika i Q13 jest ustawiane. Wzmocnione napięcie jest podawane przez dwukierunkowy klucz U5 na wejście przetwornika A/C mikrokontrolera.

Pomiar w zakresie „niskim” jest bardziej skomplikowany. Bocznikiem pomiarowym jest rezystor R22 o oporności 1 kΩ i tolerancji 1%. Cykl pomiaru rozpoczyna się po wymuszeniu na wyjściu IDD_CNT_EN poziomu niskiego. Pojawia się on na wejściu CLR licznika 74HC4060 (układ U3) i zaczyna on zliczać impulsy o częstotliwości ok. 30 kHz. Jednocześnie wyjście Q13 układu U3 zostaje wyzerowane na 150 ms dołączając poprzez klucz U20 rezystor R22 do układu pomiaru prądu. Zakres pomiarowy jest ustalany w ten sposób na 60 µA. Po 150 ms potrzebnych na ustabilizowanie się prądu pobieranego przez mikrokontroler R22 jest zwierany przez tranzystor T1 układu U20 przez kolejne 150 ms. Po upływie tego czasu klucz analogowy U5 jest rozwierany (zmiana poziomu na Q14), a napięcie odpowiadające zmierzonemu prądowi jest „zapamiętywane” w kondensatorze C13 i może być mierzone przez przetwornik A/C mikrokon-



Rysunek 5. Układ pomiaru prądu Idd

trolera. Jednocześnie zmiana poziomu na Q14 powoduje ustawienie wejścia WAKE_UP mikrokontrolera i jego przejście w tryb *Run*. Na **rysunku 6** pokazano zależności czasowe w trakcie pomiaru prądu na niskim zakresie pomiarowym

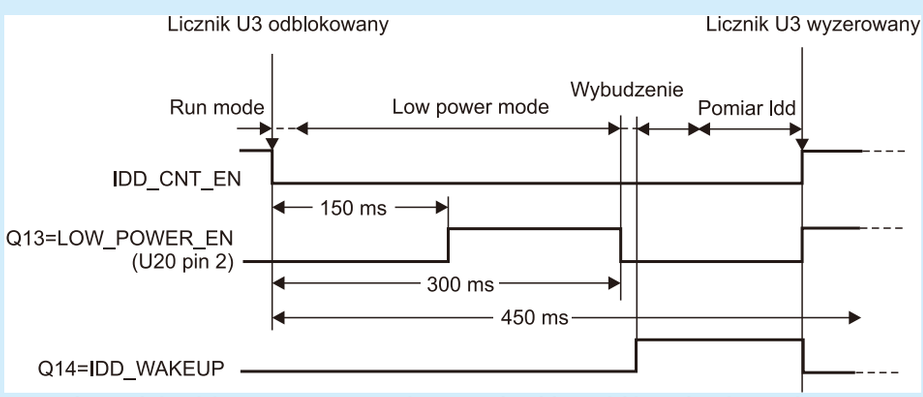
Program testowy

Program testowy może pracować w 3 trybach: Demo, pomiaru prądu bias, test fabryczny. Pierwszy tryb jest uruchamiany automatycznie po włączeniu zasilania modułu *STM32L Discovery*. Pozostałe dwa muszą być uruchomione przez użytkownika. Aby pomiary prądu były dokładne, zaleca się przed wykonaniem zasadniczych pomiarów zmierzenie i zapisanie prądu bias.

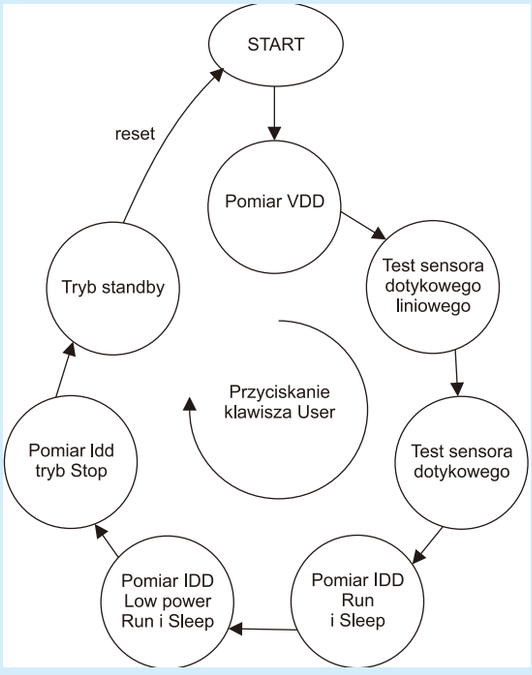
Tryb Demo Mode

Przeznaczony do demonstracji możliwości sensorów dotykowych i pomiaru prądu Idd. W fabrycznie skonfigurowanym module, po dołączeniu do portu USB, na ekranie wyświetlacza LCD pojawia się tekst powitalny, a potem wartość napięcia zasilania mikrokontrolera. Ta faza testu jest również sygnalizowana naprzemiennym miganiem diod LD3 (zielona) i LD4 (niebieska). Kolejna funkcje tego testu muszą być wybierane przez użytkownika poprzez sekwencyjne przyciskanie przycisku USER (**rysunek 7**).

Po pierwszym naciśnięciu przycisku *User* można przetestować działanie liniowego sensora dotykowego umieszczonego przy dolnej krawędzi płytki. Na ekranie



Rysunek 6. Zależności czasowe przy pomiarze prądu Idd na niskim zakresie pomiarowym



Rysunek 7. Fazy działania programu testowego

Listing 1 Wprowadzenie trybu RUN z oscylatorem MSI

```
case MCU_RUN:
/* włączenie generatoraMSI */
  SetHSICLKToMSI(RCC_MSIRange_6,NoDIV2,NoRTC);
  Config_RCC(&SavRCC);
  SysTick->CTRL = 0;
  RCC->APB1ENR = 0;
/* opóźnienie wymagane przy pomiarze prądu  */
  for (i=0;i<0xffff;i++) {
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
    __NOP();  __NOP();    __NOP();  __NOP();
  }
  break;
```

Listing 2. Przełączenie taktowania z HSI na LSI

```
void SetHSICLKToMSI(uint32_t freq,bool div2,bool With_
RTC)
{
/* zerowanie systemu RCC */
  RCC_DeInit();
/* program z pamięci Flash*/
  FLASH_SetLatency(FLASH_Latency_0);
/* wyłącz działanie bufora  Prefetch Buffer */
  FLASH_PrefetchBufferCmd(DISABLE);
/* zablokuj dostęp 64 bitowy */
  FLASH_ReadAccess64Cmd(DISABLE);
/* wyłącz Flash podczas trybu  SLeep  */
  FLASH_SLEEPPowerDownCmd(ENABLE);
/*włącz  PWR APB1 Clock */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
/* wybierz zakres napięć zasilających  Voltage Range 3
(1.2V) */
  PWR_VoltageScalingConfig(PWR_VoltageScaling_Range3);
/*czekaj aż regulator napięcia będzie gotowy*/
  while (PWR_GetFlagStatus(PWR_FLAG_VOS) != RESET);
/* ustawienie czestotliwości  */
  RCC_MSIRangeConfig(freq);
/* ustawieni e zegara MSI jako zegara systemowego */
  RCC_SYSCLKConfig(RCC_SYSCLKSource_MSI);
/* czekaj aż MSI zostanie ustawiony jako zegar systemowy
*/
  while (RCC_GetSYSCLKSource() != 0x00);
/* podział częstotliwości przez 2 */
if (div2) RCC_HCLKConfig(RCC_SYSCLK_Div2);
  RCC_HSICmd(DISABLE);
/* zablokowanie zegara HSE */
  RCC_HSEConfig(RCC_HSE_OFF);
/* Bez RTC - zablokowanie zegara  LSE */
  if (! With_RTC) RCC_LSEConfig(RCC_LSE_OFF);
/* Zablokowanie zegara LSI */
  RCC_LSICmd(DISABLE);
}
```

LCD początkowo jest wyświetlany znak „%”. Po dotknięciu pola czujnika i w trakcie przesuwania palcem od lewej do prawej po polu sensora zmienia się wyświetlana wartość położenia palca, od 0% do 100%. Kolejne przyciśnięcie *User* umożliwia wywołanie procedur testowania czterech „przycisków” dotykowych. Na ekranie LCD są wyświetlane cztery 0. Po dotknięciu pola przycisku jest wypełniana cyfra zero odpowiadająca danemu przyciskowi. Obie procedury testujące funkcje dotykowe działają bardzo dobrze i mogą być wykorzystane we własnych aplikacjach.

Kolejną funkcją włączaną sekwencyjnie klawiszem *User* jest naprzemienny pomiar prądu w trybach *Run* i *Sleep*. Mój moduł zmierzył pobór 0,86 mA w trybie *Run* i 0,29 mA w trybie *Sleep*. Kolejne pomiary dotyczą naprzemiennego natężenia prądu pobieranego w trybie *Low Power Run* i *Low Power Sleep*. Tu zmierzone wartości (jak się można było spodziewać) są dużo mniejsze. W trybie *Low Power Run* jest pobierany prąd 7,57 µA, a w trybie *Low Power Sleep* 3,88 µA. Ostatni pomiar, to pobór prądu w trybie *Stop z zegarem RTC* – pobór 1,12 µA i *Stop bez zegara RTC* – pobór 0,05 µA.

Wyniki pomiarów są zgodne z oczekiwaniami. Jednak dla programisty – konstruktora dużo ciekawsze jest poznanie, jak w praktyce wprowadza się mikrokontroler w poszczególne tryby. Program testowy modułu zawiera funkcję uint16_t ADC_Icc_Test(uint8_t Mcu_State) wykonującą pomiary w poszczególnych trybach oszczędzania energii. Co oczywiste, aby zmierzyć coś w tym czy innym trybie, trzeba najpierw mikrokontroler w ten tryb wprowadzić. Tryb oszczędzania energii jest określony argumentem *Mcu_State*. Na **listingu 1** pokazano fragment wprowadzający mikrokontroler w tryb *Run* z taktowaniem z generatora MSI. Wejście w ten tryb polega głównie na włączeniu generatora za pomocą funkcji SetHSICLKToMSI. Częstotliwość generatora jest określana przez argument RCC_MSIrange_6. Sygnał z MSI nie jest dzielony przez 2. Potem jest odliczane opóźnienie potrzebne do ustabilizowania się pobieranego prądu po zmianie taktowania.

Funkcję SetHSICLKToMSI pokazano na **listingu 2**. Jej argumentami są: częstotliwość oscylatora (*freq*), włączenie/wyłączenie podziału częstotliwości przez 2 (*div2*) oraz włączenie/wyłączenie zegara RTC (*With_RTC*).

Przełączenie źródła sygnału taktującego nie jest banalne. Trzeba w odpowiedniej kolejności wykonać szereg czynności. Większość z nich jest wykonywana przez funkcje biblioteczne lub gotowe funkcje napisane specjalnie dla potrzeb tego programu testowego. Nam pozostaje tylko cieszyć się, że nie trzeba ich tworzyć samodzielnie. Oprócz przełączenia taktowania jest tu za pomocą funkcji PWR_VoltageScalingConfig wybierany tryb pracy regulatora (**listing 3**). Podobne jest wykonywane włączenie trybu *Sleep* – **listing 4**. Najpierw program przełącza taktowanie na MSI, a potem za pomocą funkcji PWR_EnterSleepMode włącza regulator napięcia i wprowadza mikrokontroler w tryb Sleep, co pokazano na **listingu 5**. Funkcje wprowadzenia w tryby *Stop* i *Standby* zamieszczono na **listingu 6** i **listingu 7**.

Jak już wiemy z opisu metody pomiaru prądu, po przejściu w tryb oszczędzania energii ładuje się kondensator C13 (rys. 5), a po zakończeniu pomiaru jest otwierany klucz analogowy i przetwornik A/C może zmierzyć to napięcie, proporcjonalne do pobieranego prądu. Przy

mierzeniu napięcia i wyświetlaniu na wyświetlaczu LCD mikrokontroler jest przełączany na taktowanie generatorem HSI, a napięcie zasilające jest przełączane w tryb 1. Można wtedy używać bloku przetwornika A/C bez żadnych ograniczeń. Fragment programu wykonywanego po wybudzeniu z trybów oszczędzania energii pokazano na **listingu 8**. Zasadnicza część programu zawierająca procedury wprowadzania w tryby oszczędzania energii i procedury pomiaru energii jest umieszczona w pliku icc_measure.c, w tym główna funkcja uint16_t ADC_Icc_Test(uint8_t Mcu_State). Można je sobie przeanalizować otwierając projekt programu testowego z plikami źródłowymi. Pomocne w analizowaniu programów będą na pewno schematy blokowe wprowadzania w tryb *Low Power*, który jest pokazany na **rysunku 9** i *Standby Mode* na **rysunku 10**. Warto tu zwrócić uwagę na to, że oprócz standardowych czynności związanych z trybami oszczędzania energii, czyli włączenie i zaprogramowanie regulatora napięcia zasilania, programowania taktowania i wprowadzania trybu oszczędzania przez zapisywanie rejestru SLEEPDEEP, trzeba też zwrócić uwagę na to, co będzie się działo z innymi elementami układu. Aby ograniczenie prądu miało sens, trzeba wyłączyć moduł wyświetlacza LCD i tak ustawić porty GPIO, aby pobierały jak najmniej prądu. Po wprowadzeniu trybów oszczędzania energii większość linii GPIO jest konfigurowana jako wejścia analogowe, bo w takiej konfiguracji pobierają najmniej prądu. Wyjątkiem są linie potrzebne do działania aplikacji, czyli sterujące diodami LD3 i LD4, wejście przetwornika PA4, wejście wybudzenia PA0 i wyjście PC13 sterujące działaniem zewnętrznego licznika U3 w układzie pomiaru prądu. Poza tym, jest niezbędne skonfigurowanie układu przerwań, bo zgłoszenie przerwania zewnętrznego na linii PA0 powoduje wybudzenie mikrokontrolera.

Listing 3. Zmiana trybu pracy regulatora napięcia (skalowanie napięcia)

```
void PWR_VoltageScalingConfig(uint32_t PWR_VoltageScaling)
{
  uint32_t tmp = 0;
/* zmien tryb pracy regulatora*/
  assert_param(IS_PWR_VOLTAGE_SCALING_RANGE(PWR_VoltageScaling));
  tmp = PWR->CR;
  tmp &= CR_VOS_MASK;
  tmp |= PWR_VoltageScaling;
  PWR->CR = tmp & 0xFFFFFFF3;
}
```

Listing 4. Wprowadzenie w tryb Sleep (fragment funkcji uint16_t ADC_Icc_Test(uint8_t Mcu_State))

```
/* SLEEP mode : pomiar z  MSI 4 MHz bez  RTC */
case MCU_SLEEP:
//przełącz zegar na MSI
  SetHSICLKToMSI(RCC_MSIRange_6,NoDIV2,NoRTC);
  Config_RCC(&SavRCC);
  Config_Systick_50ms();
  Delay(1);
//Wprowadzenie w tryb Sleep
  PWR_EnterSleepMode(PWR_Regulator_ON,PWR_SLEEPEntry_WFI);
  break;
```

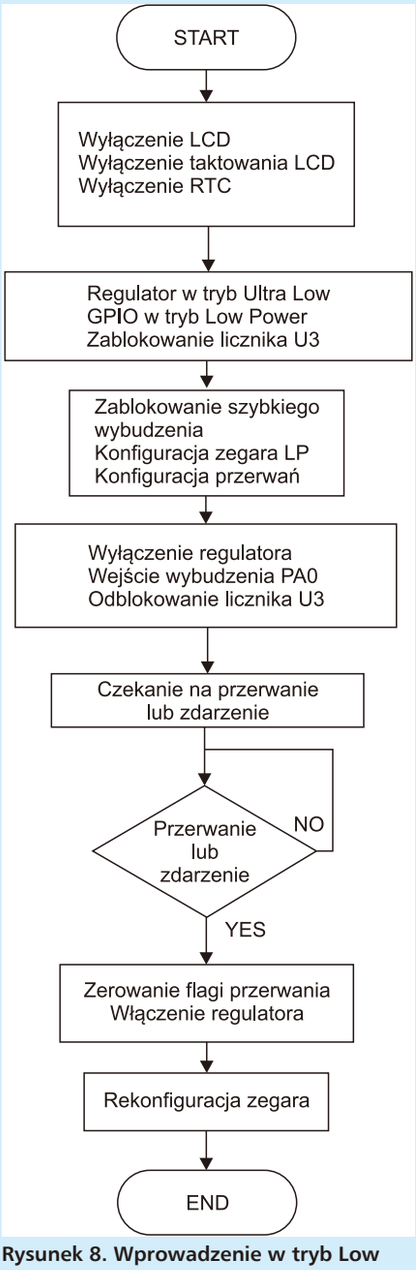
Listing 5. Wprowadzenie w tryb Sleep

```
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 stanu regulatora w trybie  Sleep*/
  tmpreg = PWR->CR;
/* zerowanie bitów  PDDS I  LPDSR */
  tmpreg &= CR_DS_MASK;
/*ustawienie bitów LPDSR zaleznie od wartosci  PWR_Regulator */
  tmpreg |= PWR_Regulator;
/* zapisanie nowej wartosci */
  PWR->CR = tmpreg;
/* zerowanie bitu SLEEPDEEP w rejestrze Cortex System Control Register */
  SCB->SCR &= (uint32_t)~((uint32_t)SCB_SCR_SLEEPDEEP);
/* wprowadzenie trybu Sleep*/
  if(PWR_SLEEPEntry == PWR_SLEEPEntry_WFI)
  {
/* żądanie czekania na przerwanie */
    __WFI();
  }
  else
  {
/* żądanie czekania na zdarzenie ( Event) */
    __WFE();
  }
}
```

Listing 6. Wprowadzenie w tryb Stop

```
 void PWR_EnterSTOPMode(uint32_t PWR_Regulator, uint8_t PWR_STOPEntry)
{
  uint32_t tmpreg = 0;
  assert_param(IS_PWR_REGULATOR(PWR_Regulator));
  assert_param(IS_PWR_STOP_ENTRY(PWR_STOPEntry));
/* Ustawienie regulatora w tryb STOP */
  tmpreg = PWR->CR;
  /* Kasowanie bitów PDDS i LPDSR */
  tmpreg &= CR_DS_MASK;
/* ustawienie bitów  LPDSR zależnie od wartości PWR_Regulator  */
  tmpreg |= PWR_Regulator;
/* zapisz nową wartość */
  PWR->CR = tmpreg;
/* ustaw bit  SLEEPDEEP w rejestrze  Cortex System Control */
  SCB->SCR |= SCB_SCR_SLEEPDEEP;
/* wybranie wejscia w tryb Select STOP*/
  if(PWR_STOPEntry == PWR_STOPEntry_WFI)
  {
/* żądanie czekania na przerwanie */
    __WFI();
  }
  else
  {
    /* żądanie czekania na zdarzenie */
    __WFE();
  }
/*zerowanie bitu  SLEEPDEEP w rejestrze  Cortex System Control */
  SCB->SCR &= (uint32_t)~((uint32_t)SCB_SCR_SLEEPDEEP);
}
```

Pomiar i zapisanie wejściowego prądu polaryzacyjnego (bias). Jak już wiemy pomiar prądu pobieranego przez mikrokontroler polega na mierzeniu spadku napięcia na rezystorze włączonym szeregowo w obwód źródła zasilania. Taki sposób pomiaru wymaga, żeby spadek



Rysunek 8. Wprowadzenie w tryb Low Power

na rezystorze pomiarowym nie był duży, bo zmniejsza się wtedy wartość napięcia zasilania co może nie być korzystne a nawet możliwe. Dlatego układ pomiarowy zawiera wzmacniacz operacyjny wzmacniający spadek napięcia 50-krotnie. Ponieważ mamy tu do czynienia z pomiarami bardzo małych prądów, to należy zmierzyć i uwzględnić wejściowy prąd polaryzacyjny wzmacniacza. Program testowy ma wbudowaną procedurę umożliwiającą taki pomiar. Żeby ją uruchomić, trzeba wykonać następujące czynności:

- Wyłączyć zasilanie modułu.
- Zworkę JP1 ustawić w położeniu OFF (wyłączenie pomiaru prądu Idd).
- Nacisnąć przycisk User i kiedy jest przyciśnięty włączyć zasilanie modułu.
- Na wyświetlaczu powinien wyświetlić się komunikat „Bias Current".
- Po zwolnieniu przycisku *User* na wyświetlaczu pojawi się wartość zmierzonego prądu.
- Procedura pomiaru i wyświetlania prądu jest wykonywana w nieskończonej pętli. Powrót do pomiaru prądu Idd jest możliwy po restarcie mikrokontrolera.

Po wykonaniu tych czynności prąd zostanie zmierzony i zapisany w pamięci nieulotnej. Na **listingu 9** pokazano procedurę pomiaru i wyświetlania prądu *bias*. Sam pomiar jest wykonywany za pomocą znanej nam już procedury ADC_Icc_Test, mierzącej prąd w różnych trybach oszczędzania energii. Do pomiaru prądu *bias* wykorzystuje się pomiar w trybie *Stop bez zegara RTC*. Jednak tu nie mierzymy poboru prądu przez mikrokontroler, bo JP1 jest w pozycji OFF i zasilanie mikrokontrolera jest dołączone bezpośrednio do źródła zasilania, a nie przez rezystor pomiarowy. Na wejście nieodwracające jest podane napięcie zasilania, a na wejście odwracające napięcie zasilania przez rezystor 1 kΩ. I to właśnie spadek napięcia na tym rezystorze odpowiada wejściowemu prądowi polaryzacji wzmacniacza pomiarowego.

Listing 7 wprowadzenie trybu Standby

```
void PWR_EnterSTANDBYMode(void)
{
/* zeruj znacznik Clear Wakeup  */
  PWR->CR |= PWR_CR_CWUF;
/* wybierz tryb  STANDBY*/
  PWR->CR |= PWR_CR_PDDS;
/*Ustaw bit SLEEPDEEP w rejestrze  Cortex
System Control  */
  SCB->SCR |= SCB_SCR_SLEEPDEEP;
/* żądanie czekania na przerwanie */
  __WFI();
}
```

Listing 8. Fragment funkcji uint16_t ADC_Icc_Test(uint8_t Mcu_State)) mierzącej napięcie na kondensatorze C13

```
SetHSICLK();  //taktowanie zegarem HSI
Config_Systick(); //skonfigurowanie licznika sys
tick
RCC->AHBENR = SavRCC.AHBENR;
//włączenie trybu 1 dla regulatora napięcia
PWR_VoltageScalingConfig(PWR_VoltageScaling_
Range1);
/* czekaj aż regulator będzie gotowy*/
while (PWR_GetFlagStatus(PWR_FLAG_VOS) !=
RESET) ;
/* zmierz przetwornikiem ADC napięcie na
kondensatorze C13 */
adc_measure = Current_Measurement();
```

Procedura pomiaru ADC_Icc_Test jest wywołana z argumentem MCU_STOP_NoRTC. Umożliwia to po pierwsze, pomiar prądu w ogóle, a po drugie powoduje, że mikrokontroler w czasie pomiaru jest w trybie *Stop z wyłączonym RTC* i nie zakłóca pomiaru prądu bias. W moim module testowym zmierzone natężenie prądu miało wartość 0,24 μA. Dla przypomnienia, pobór prądu w trybie *Stop z zegarem RTC* wynosił 1,12 μA, a *Stop bez zegara RTC* – 0,05 μA. Widać, że wartości bias mają znaczący wpływ na dokładność pomiaru i powinny być uwzględnione przy korekcji pomiaru.

Procedura samotestująca. Fabryczny program testowy został wyposażony w dodatkową procedurę samotestującą nazwaną *Manufacturing Test*. Jest ona wykonywana, jeżeli w trakcie w działa trybu *Demo Mode* przyciśniemy przycisk *User* przez co najmniej 3 sekundy. Ten test jest przeznaczony do sprawdzenia poprawności skonfigurowania modułu i poprawności działania wszystkich obwodów sprzętowych potrzebnych do prawidłowego działania *STM32L Discovery* z programem testowania trybów ograniczanego poboru energii.

Manufacturig Test sprawdza działanie oscylatora LSE (*Low Speed External*), wartość napięcia zasilania, prąd Idd w trybach *Run* i *Low Power* oraz prąd polaryzujący *bias*. Przejście testu bez błędu daje pewność, że pomiary prowadzone w trybie *Demo Mode* i *Bias Current Record* są wykonywane prawidłowo. Zasadnicza część procedury testowej, to wywołanie 5 funkcji kolejno testujących wybrane parametry pracy modułu, co pokazano na **listingu 10**. Jak wygląda funkcja testująca napięcie zasilające test_Vdd pokazano na **listingu 11**. Jej głównym elementem jest pomiar napięcia i porównanie czy mieści się w zadanych granicach.

Konfigurowanie układów peryferyjnych

Żeby zestaw STM32L Discovery mógł poprawnie wykonywać testy trybów oszczędzania energii muszą być prawidłowo skonfigurowane układy peryferyjne:

- Przetwornik ADC do pomiaru prądu Idd, prądu Bias i napięcia zasilania. Rozdzielczość przetwornika jest ustawiana na 12 bitów, pomiar pojedynczy i czas próbkowania 192 cykle.

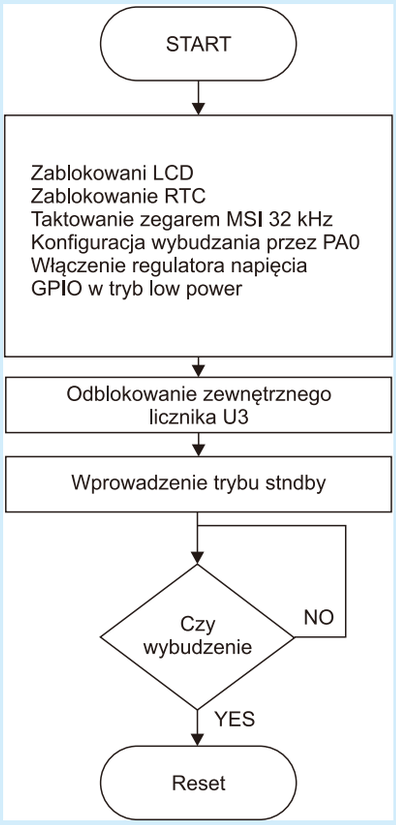
Listing 9 pomiar i zapamiętanie prądu Bias wzmacniacza pomiarowego

```
void Bias_measurement(void)
{
  float Current;
  uint16_t MeasurINT;
/* wyświetlanie ,że aplikacja jest w trybie pomiaru BIAS CURRENT */
  LCD_GLASS_ScrollSentence("      ** BIAS CURRENT ** JP1 OFF
**",1,SCROLL_SPEED);
/* Pomiar wejściowego prądu polaryzującego */
  MeasurINT = ADC_Icc_Test(MCU_STOP_NoRTC);
/* konwersja do wartości wyrażonej w  uA*/
  Current = MeasurINT * Vdd_appli()/ADC_CONV;
  Current *= 20L;

/*wyświetlenie wartości prądu */
  display_MuAmp((uint32_t)Current);
/* odblokowanie dostępu do pamięci EEPROM*/
  DATA_EEPROM_Unlock();
/* zapis zmierzonego prądu do pamięci eeprom*/
  DATA_EEPROM_FastProgramByte((uint32_t)&Bias_Current, MeasurINT) ;
/* zablokowanie dostępu do pamięci eeprom */
  DATA_EEPROM_Lock();
/* nieskończona pętla pomiaru I wyświetlania prądu Bias*/
  while (1)
  {
    /* pomiar prądu */
    MeasurINT = ADC_Icc_Test(MCU_STOP_NoRTC);
    /* konwersja do wartości wyrażonej w  uA */
    Current = MeasurINT * Vdd_appli()/ADC_CONV;
    Current *= 20L;
    /* wyświetlenie wartości prądu */
    display_MuAmp((uint32_t)Current);
    Delay(800) ;
  }
}
```

Listing 10. Fragment funkcji auto_test

```
test_vdd();
test_icc_Run();
test_Bias();
test_icc_STOP();
test_icc_STBY();
```



Rysunek 9. Wprowadzenie w tryb Standby

- Porty GPIO. Linie portów są wykorzystywane do podłączenia przycisku User (PB1), jako wejście wybudzenia (wejście PA0), sterują diodami LED (PB7 – zielona i PB6 niebieska). Do obsługi elementów dotykowych (sensora liniowego i przycisków) są wykorzystywane linie PA6, PA7, PC4, PC5, PB0 i PB1.
- Sterownik LCD. Wbudowany sterownik matrycy LCD jest używany do sterownia wyświetlacza LCD i przy wprowadzeniu trybu oszczędzania jest wyłączany programowo.
- Układ zegara. Używany w trybach oszczędzania energii zegar MSI jest wyłączany w trybie pracy *Low Power Run* i taktowanie przejmuje zegar LSE, aż do wybudzenia układu.

Podsumowanie

Przyznam, ze moduł STM32L Discovery jest w pewnym sensie wyjątkowy. Przy stosunkowo niskiej cenie (ok. 60 złotych) ma dość rozbudowane możliwości sprzętowe, rzadko spotykane w modułach tej klasy cenowej. Pomysł, by mierzyć rzeczywisty pobór prądu mikrokontrolera zoptymalizowanego do pracy w trybach oszczędzania energii uważam za bardzo trafiony. Co prawda, na początku myślałem, że to trochę „na wyrost", bo przecież prąd można zmierzyć multimetrem, ale po kilku testach zmieniłem zdanie. Stosunkowo prosty i pomysłowo zaaplikowany układ po-

Listing 11 funkcja testowania napięcia zasilania Vdd

```
void test_vdd(void)
{
  uint16_t vdd_test;
  uint16_t Message[6];

  /* wyświetlenie nazwy testu */
  LCD_GLASS_DisplayString(" VDD ");
  DELAY;
  /* pomiar napięcia zasilającego */
  vdd_test = (int)Vref_measure();
  DELAY;
  /* sprawdzenie czy napięcie zasilające miesci sie w zadanych granicach */
  if ((vdd_test>VCC_MAX) || (vdd_test<VCC_MIN))
  {
    /* jezeli nie, to program wchodzi w nieskończoną pętlę – wyjście tylko przez reset */
    while(1)
    {
      /* wyswietlenie komunikatu o błędzie*/
      LCD_GLASS_ScrollSentence("VDD ERROR ",1,SCROLL_SPEED);
      DELAY;
      convert_into_char (vdd_test, Message);
      Message[5] = 'V';
      Message[4] = ' ';
      Message[1] |= DOT;
      Message[0] = ' ';
      LCD_GLASS_DisplayStrDeci(Message);
      DELAY;
      DELAY;
    }
  }
  /* napięcia zasilania poprawne – wyświetl OK*/
  LCD_GLASS_DisplayString("VDD OK");
  DELAY ;
}
```

miarowy daje natychmiastowe wyniki, a pomiar bardzo małych prądów multimetrem też nie jest banalny i może wprowadzać duże błędy.

O oszczędzaniu energii przez mikrokontrolery pisze się wiele. Producenci, a raczej ich działy marketingu, prześcigają się w pokazywaniu swoich produktów jako super oszczędnych energetycznie pokazując nam coraz bardziej nieprawdopodobnie małe pobory prądu. Jednak możliwości to jedno, a w praktyce może być różnie. Aplikacje układów energooszczędnych nie należą do łatwych. Trzeba mieć spore doświadczenie, aby zaprojektowane urządzenie rzeczywiście pobierało mały prąd. Wykorzystanie możliwości energooszczędnych mikrokontrolerów wymaga dobrej znajomości wszystkich układów mających wpływ na pobór prądu: bloku taktowania, regulatora napięcia, ale też układów peryferyjnych. Poza tym, trzeba optymalizować energetycznie otoczenie mikrokontrolera, czyli pozostałe układy aplikacji.

Niezbędne jest poznanie zasad, kiedy można taki tryb załączyć i kiedy z niego wyjść lub przełączyć się w inny tryb, tak aby nasza aplikacja z jednej strony była energooszczędna, a z drugiej mogła wykorzystać zalety szybkiego, wydajnego, 32-bitowego rdzenia mikrokontrolera.

Dla konstruktora praktyka zestaw: moduł *STM32L Discovery* i firmowy program demonstracyjny to kopalnia wiedzy. Analizowanie procedur daje szybki, praktyczny wgląd w metody programowania. W jakiej kolejności i jak zaprogramować układ taktowania, regulacji napięcia, jak skonfigurować układ wybudzenia i jak w końcu wprowadzić mikrokontroler w żądany tryb oszczędzania. Tę wiedzę można zdobyć studiując noty katalogowe mikrokontrolera, opisy rejestrów i bardzo szczegółowo zasadę działania poszczególnych bloków funkcjonalnych. Na pewno zajmie to sporo czasu, a jak wiadomo – czas programisty jest drogi. Dużo szybciej i na pewno łatwiej można się tego nauczyć i zacząć stosować wykorzystując gotowe procedury programu testowego.

STM32L Discovery to jednak więcej niż tylko tryby oszczędzania, chociaż do tego jest przeznaczony ten moduł. Konstruktor może niejako przy okazji poznać jak można sterować matrycą wyświetlacza LCD. Dla mnie szczególnie interesująca była doskonale działająca demonstracja działania paneli dotykowych. Przyjęta przez producentów strategia, by oferować tanie i w miarę możliwości dobrze wyposażone moduły ewaluacyjne z dobrze przygotowanymi programami testowymi, jest bardzo dobrym sposobem na szybki start w dziedzinach wymagających sporej wiedzy. Tryby oszczędzania energii i *STM32L Discovery* to potwierdzają.

Tomasz Jabłoński, EP