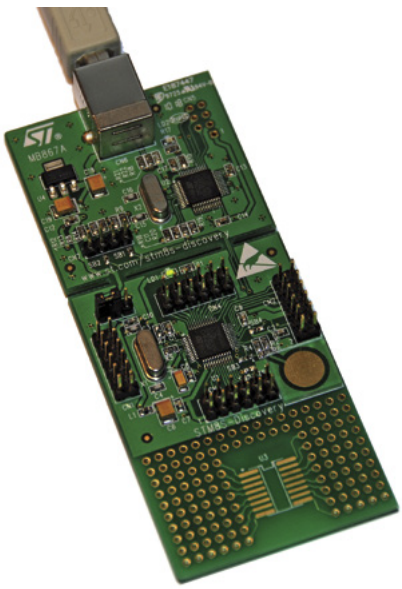
Mikrokontrolery STM8S – pierwsze kroki (1)

Jednym z podstawowych elementów każdego systemu wbudowanego jest układ sterujący i nadzorujący jego pracę, którym zazwyczaj jest mikrokontroler. Niebywała różnorodność tworzonych aplikacji sprawia jednak, że nie w każdym systemie musi się znaleźć bogato wyposażony w bloki peryferyjne układ o dużej mocy obliczeniowej. Często wystarczający jest mikrokontroler znacznie prostszy, a przy tym tańszy. Stąd, mimo rosnącej popularności mikrokontrolerów 32-bitowych, rynek układów 8-bitowych jest nadal bardzo szeroki. Co więcej, pojawiają się na nim nowe rodziny mikrokontrolerów. Jedną z nich, wprowadzoną w 2008 r., są układy STM8 oferowane przez ST Microelectronics. W dwóch częściach artykułu przedstawione zostaną cztery przykładowe programy pokazujące podstawowe zagadnienia związane z programowaniem tych układów.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 16344, pass: 52qof741
• kod źródłowy

Rodzina mikrokontrolerów STM8 powstała jako następca rodziny ST7, a w jej skład wchodzą 4 grupy układów: STM8S – główna linia mikrokontrolerów STM8, STM8L – wersje o małym poborze mocy, STM8A – wersje przeznaczone dla przemysłu samochodowego i STM8T – wersje dedykowane do zastosowania w urządzeniach sterowanych dotykowo. Podstawowym elementem tych mikrokontrolerów jest nowoczesny, szybki rdzeń przetwarzający instrukcje z wykorzystaniem 3-etapowego potoku. Układy wykorzystują architekturę harwardzką, a wśród najważniejszych cech wyróżnić należy między innymi realizowane sprzętowo operacje dzielenia i mnożenia, do 6 KB pamięci operacyjnej SRAM, do 128 KB pamięci Flash, do 2 KB wbudowanej pamięci EEPROM o trwałości ponad 300tys cykli zapisu, obsługę 32 przerwań, wyposażenie w 8- i 16-bitowe licznik uniwersalne, 10-bitowy przetwornik A/C, układy UART, I²C i SPI oraz tryby obniżonego poboru energii.



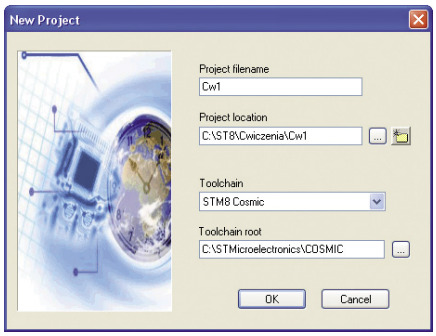
Fotografia 1. Zestaw STM8S Discovery

Poznawanie możliwości mikrokontrolera zwykle najwygodniej jest rozpocząć od zakupu gotowego zestawu uruchomieniowego. W przypadku układów STM8 wybór nie jest zbyt duży i na polskim rynku sprowadza się w zasadzie tylko do zestawów firmowych ST Microelectronics. Oprócz rozbudowanych i drogich można jednak na szczęście znaleźć także proste i tanie (około 45zł) płytki serii Discovery. Przedstawione w artykule przykłady uruchamiane były na jednym z takich zestawów – STM8S Discovery (**fotografia 1**). Jego głównym elementem jest mikrokontroler STM8S105C6. Oprócz elementów bezpośrednio związanych z uruchomieniem mikrokontrolera (m.in. oscylator zewnętrzny i elementy zasilające) na płytce znajdują się tylko złącza szpilkowe, na które wyprowadzono wszystkie linie obudowy układu, jedna dioda oraz pole dotykowe. Dodatkowo, w dolnej części płytki znajduje się kilka pól lutowniczych pozwalających uzupełnić zestaw o własne układy. Ponadto, w skład zestawu wchodzi programator USB ST-Link pracujący w standardzie SWIM. Ciekawostką jest możliwość odłamania fragmentu płytki zawierającej programator od reszty zestawu. Możemy go wówczas wykorzystywać niezależnie do programowania innych procesorów serii STM8.

Opisywane w artykule programy tworzone były w języku C w darmowym środowisku *ST Visual Develop* (STVD) w wersji 4.3.1. Środowisko STVD nie zawiera kompilatora i linkera, ale pozwala wykorzystać narzędzia firm Raisonace lub Cosmic. Przy tworzeniu programów wykorzystano drugą z możliwości. Oprócz środowiska wykorzystane zostały także gotowe biblioteki stworzone przez ST Microelectronics. Pierwszą z nich jest *STM8S/A Standard Peripheral Library* ułatwiająca korzystanie z układów peryferyjnych mikrokontrolera. Z kolei drugą – biblioteka *STM8 Touch Sensing Library* pozwalająca w łatwy sposób wykorzystać znajdujące się na płytce uruchomieniowej pole dotykowe. Podczas instalacji narzędzi firmy Cosmic należy dokonać rejestracji, a wykorzystać można je dopiero po otrzymaniu pliku licencyjnego, co trwa około 1 dnia, ponieważ proces licencjonowania nie jest zautomatyzowany. Otrzymany e-mailem plik licencyjny należy umieścić w podkatalogu *License* w strukturze katalogów narzędzi Cosmic. Pobierając pliki

Tabela 1. Źródła plików potrzebnych przy tworzeniu programów dla STM8S

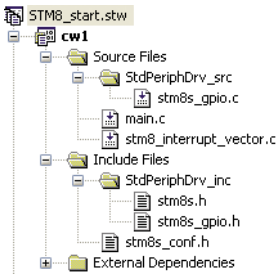
ST Toolset (ZIP)
http://www.st.com/internet/evalboard/product/210567.jsp
STM8S/A Standard Peripheral Library (ZIP) STM8S and STM8A Microcontroller Families – Reference Manual (PDF) STM8S105 – Datasheet (PDF) AN2869: Guidelines for designing touch sensing applications (PDF)
http://www.st.com/internet/mcu/product/215105.jsp
STM8 Touch Sensing Library (ZIP) STM8 Touch Sensing Library (PDF)
http://www.st.com/internet/mcu/product/251122.jsp
STM8S Discovery User Manual (PDF)
http://www.st.com/internet/evalboard/product/247087.jsp
Cosmic STM8 Cross Development Tools (ZIP)
http://www.cosmic-software.com/stm8.php



Rysunek 2. Okno konfiguracyjne nowego projektu

Listing 1. Kod źródłowy programu 1 – plik main.c

```
#include „stm8s.h”

int main(void) {
  volatile unsigned int i;

  GPIO_DeInit(GPIOD); //od-konfigurowanie GPIO

  //Konfiguracja GPIO D0
  GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_LOW_FAST);

  //Glowna pelta programu
  while(1) {
    //Zmiana stanu LED na przeciwny
    GPIO_WriteReverse(GPIOD, GPIO_PIN_0);
    for (i = 0; i < 50000; i++);
  }
}
```



Rysunek 3. Struktura plików w programie 1

oprogramowania i bibliotek, warto przy okazji pobrać także kilka plików dokumentacji. W **tabeli 1** zestawiono adresy internetowe, pod którymi znaleźć można potrzebne pliki.

Program 1 – migająca dioda

Pierwszym programem, który zostanie utworzony, będzie mikrokontrolerowy odpowiednik klasycznego „hello world”, czyli program migający diodą LED. Tworzenie projektu należy rozpocząć od uruchomienia STVD i wybrania z menu *File* polecenia *New Workspace*, a następnie wybrania w oknie dialogowym elementu *Create Workspace and Project*. Rozpocznie się wówczas procedura tworzenia nowej przestrzeni roboczej, która zawierać może jeden lub kilka projektów. W oknie, które się pojawi, należy określić nazwę przestrzeni i jej lokalizację. W kolejnym oknie, dotyczącym tworzenia projektu (**rysunek 2**), należy podać jego nazwę i lokalizację, a następnie wybrać rodzaj kompilatora (w naszym przypadku będzie to *STM8 Cosmic*) i lokalizację jego katalogu głównego. Kolejnym etapem jest wybranie rodzaju mikrokontrolera, dla którego pisać będziemy programy – w naszym przypadku należy wybrać STM8S105C6.

Po wybraniu typu układu utworzony zostanie pusty projekt zawierający dwa pliki: *main.c* oraz *stm8_interrupt_vector.c*. Pierwszy z plików jest głównym plikiem z kodem naszego programu, zaś drugi – służy do powiązania przerwań z procedurami ich obsługi.

Do projektu należy jeszcze dodać biblioteki obsługi układów peryferyjnych. W widoku struktury projektu w panelu przestrzeni roboczej (*Workspace*), w grupie *Source Files* należy utworzyć podgrupę o nazwie *StdPeriphDrv_src*, a w grupie *Include Files* podgrupę *StdPeriphDrv_inc*. Ponieważ pierwszy projekt będzie bardzo prosty, do grupy *StdPeriphDrv_src* wystarczy dodać tylko plik *C:\ \ST8\stm8_stdperiph_lib\STM8S_StdPeriph_Lib_V2.1.0\Libraries\STM8S_StdPeriph_Driver\src\stm8s_gpio.c*

Z kolei do *StdPeriphDrv_inc* wystarczy dodać z katalogu *C:\ST8\stm8_stdperiph_lib\STM8S_StdPeriph_Lib_V2.1.0\Libraries\STM8S_StdPeriph_Driver\inc* pliki *stm8s.h* oraz *stm8s_gpio.h*. Ponadto, z katalogu *C:\ST8\stm8_stdperiph_lib\STM8S_StdPeriph_Lib_V2.1.0\Project\STM8S_StdPeriph_Template* należy do katalogu własnego projektu skopiować plik *stm8s_conf.h*.

Na **rysunku 3** pokazano strukturę plików dla programu 1.

Ostatnim etapem konfiguracji projektu jest poinformowanie bibliotek, że wykorzystywany będzie mikrokontroler serii STM8S105. W tym celu należy kliknąć prawym klawiszem myszy na nazwę projektu w oknie przestrzeni roboczej i wybrać polecenie *Settings*…. W oknie, które się pojawi, należy wybrać zakładkę *C Compiler* i w polu *Preprocessor Definitions* wpisać STM8S105 (**rysunek 4**). Operację tę należy powtórzyć zarówno dla kompilacji typu *Debug*, jak i *Release* (wybór możliwy jest z listy rozwijalnej w lewym górnym rogu okna).

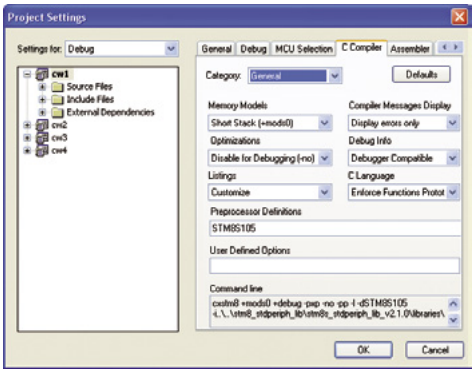
Naszym pierwszym zadaniem będzie zamigotanie diodą. Jest ona podłączona do linii 0 portu GPIOD. Aby można było w programie skorzystać z bibliotek dostarczonych przez STM, należy dołączyć na początku pliku *main.c* plik nagłówkowy *stm8s.h*. Dalej, na początku funkcji *main*( ) konieczne będzie zdefiniowanie zmiennej *i*, która posłuży jako licznik pętli opóźniającej. Zmienna ta powinna być ulotna (*volatile*), aby kompilator nie poddawał poleceń wykorzystujących tę zmienną optymalizacji. W przeciwnym wypadku pętla opóźniająca mogłaby zostać usunięta, gdyż nie zawiera w sobie żadnego użytecznego kodu. Pierwszą operacją, którą należy przeprowadzić w funkcji *main*( ), jest deinicjalizacja portu GPIOD, aby mieć pewność, że przyjmie on wyłącznie ustawienia, które podane zostaną w kolejnym kroku. Służy do tego funkcja *GPIO_DeInit*( ). Następnie można skonfigurować linię GPOID0 jako wyjście komplementarne (*push-pull*). Dodatkowo wybiera się także maksymalną częstotliwość zmian stanu danej linii (dostępne są częstotliwości 2 i 10 MHz) oraz aktywny stan logiczny linii (wysoki lub niski). Do zapisu ustawień konfiguracyjnych służy funkcja *GPIO_Init*( ). Wewnątrz pętli głównej pozostaje jeszcze dodać polecenie zmiany stanu linii D0 na przeciwny (funkcja *GPIO_WriteReverse*( )) oraz umieścić pętlę opóźniającą.

Gotowy program należy skompilować (klawisz F7). Jeśli kompilacja się powiedzie, program zostanie umieszczony w pamięci mikrokontrolera po przejściu do trybu śledzenia wykonania programu. Najpierw konieczne jest jednak ustawienie typu debuggera. W tym celu należy wybrać z menu *Debug instrument* polecenie *Target Settings* i z listy rozwijalnej wybrać *Swim ST-Link*. Po zatwierdzeniu ustawień można przejść do trybu debuggera (polecenie *Debug > Strat Debugging*…). Uruchomienie programu następuje po wydaniu plecenie *Debug > Run*…. Jeśli wszystkie kroki zostały wykonane poprawnie, zielona dioda LED na zestawie uruchomieniowym powinna zacząć migać. Kod programu podany jest na **listingu 1**.

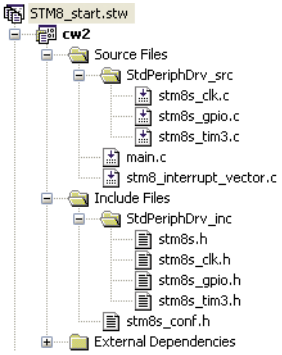
Program 2 – konfiguracja sygnałów taktujących i licznika

Naszym zadaniem będzie teraz napisanie programu, który będzie sterował diodą w taki sposób, że będzie ona włączana co 1 s na 250 ms. Do odliczania czasu wykorzystany zostanie licznik TIM3, a dodatkowo zmienimy częstotliwość pracy mikrokontrolera na 16 MHz.

Procedura tworzenia projektu będzie taka sama jak poprzednio, z tym że dodatkowo należy do niego dołączyć bibliotekę *stm8s_clk* zawierającą funkcje konfigurujące sygnały



Rysunek 4. Okno konfiguracyjne projektu – zakładka ustawień kompilatora



Rysunek 5. Struktura plików w programie 2

taktujące oraz *stm8s_tim3* zawierającą funkcje obsługi licznika. Na **rysunku 5** pokazano strukturę plików dla programu 2. Przy okazji warto zwrócić uwagę, że w przypadku, gdy nowy projekt dodawany jest jako kolejny do już istniejącej przestrzeni roboczej, należy podczas pracy pamiętać o przełączaniu się pomiędzy projektami. Wyboru dokonuje się w menu *Project > Set Active Project…*. Niezależnie od tego, które pliki są w danym momencie otwarte i edytowane, kompilowany jest tylko projekt ustawiony jako aktywny.

W poprzednim programie kod źródłowy zawierał tylko polecenia dotyczące konfiguracji portu GPIO oraz zmiany jego stanu. Można zauważyć, że nigdzie nie występują polecenia związane np. z konfiguracją częstotliwości taktowania mikrokontrolera. Jest tak, ponieważ wykorzystana została konfiguracja standardowa, w której układ taktowany jest z wewnętrznego oscylatora HSI o częstotliwości 16 MHz dzielonej przez 8 (czyli 2 MHz). Oprócz taktowania sygnałem pochodzącym z HSI możliwe jest także taktowanie z oscylatora wewnętrznego (LSI) o częstotliwości 128 kHz oraz z zewnętrznego oscylatora HSE, którego częstotliwość może zawierać się w przedziale od 1 do 24 MHz. Możliwe jest także bezpośrednie podanie sygnału zegarowego o częstotliwości do 24 MHz (HSE Ext.). Uproszczony schemat drzewa sygnałów taktujących pokazany jest na **rysunku 6**.

Na płytce zestawu uruchomieniowego umieszczony jest zewnętrzny oscylator o częstotliwości 16 MHz (oscylator HSE). Aby zmienić źródło głównego sygnału taktującego (fMASTER), należy ten oscylator najpierw włączyć (funkcja *CLK_HSECmd*( )), a następnie odczekać na stabilizację generowanego przez niego sygnału. Kolejnym krokiem jest ustawienie wartości dzielnika (*prescalera*) częstotliwości, który pozwala wybrać docelową częstotliwość sygnału fCPU taktującego rdzeń mikrokontrolera. Aby uzyskać maksymalną szybkość pracy układu, należy wybrać wartość 1. Dostępne są również dzielniki o wartościach kolejnych potęg 2, aż do 128. Do wyboru dzielnika służy funkcja *CLK_SYSCLKConfig*( ). Ostatnim etapem jest przełączenie źródła taktowania poprzez wywołanie funkcji *CLK_ClockSwitchConfig*( ). Należy przy tym podać tryb przełączenia,

Listing 2. Kod źródłowy programu 2 – plik *main.c.*

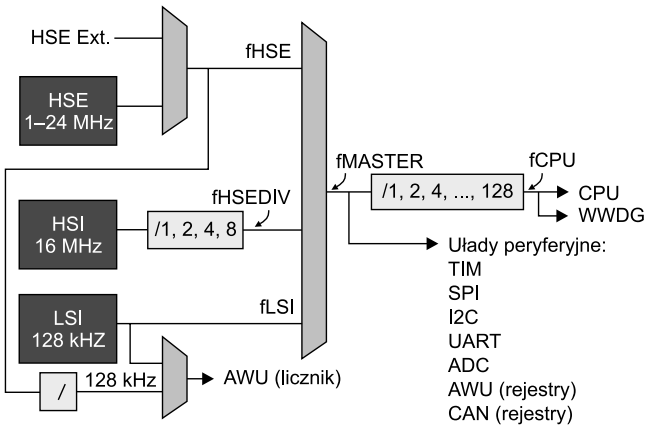
```
#include „stm8s.h”

void CLK_Config(void);
void GPIO_Config(void);
void TIM_Config(void);

int main(void) {
  CLK_Config();
  GPIO_Config();
  TIM_Config();

  //Glowna pelta programu
  while(1) {
    //LED miga co 1 sekunde i wlaczona jest przez 1/4 sekundy
    if (TIM3_GetCounter() < 16250) {
      GPIO_WriteLow(GPIOD, GPIO_PIN_0);
    } else {
      GPIO_WriteHigh(GPIOD, GPIO_PIN_0);
    }
 }
}


void CLK_Config(){
  //Konfiguracja sygnalow zegarowych
  //wlaczenie HSE
  CLK_HSECmd(ENABLE);
  while (CLK_GetFlagStatus(CLK_FLAG_HSERDY)==RESET);

  //dzielnik dla zegara fCPU
  CLK_SYSCLKConfig(CLK_PRESCALER_CPUDIV1);

  //przelaczenie zrodla taktowania na HSE=16MHz
  //(tryb automatyczny, przerwanie od przelaczenia wylaczone,
  //poprzednie zrodlo zegara wylaczone)
  CLK_ClockSwitchConfig(CLK_SWITCHMODE_AUTO,
                        CLK_SOURCE_HSE,
                        DISABLE,
                        CLK_CURRENTCLOCKSTATE_DISABLE);
}


void GPIO_Config(){
  //Konfiguracja portow GPIO
  GPIO_DeInit(GPIOD); //de-konfiguracja GPIO

  //Konfiguracja GPIO D0
  GPIO_Init(GPIOD, GPIO_PIN_0, GPIO_MODE_OUT_PP_LOW_FAST);
}


void TIM_Config(){
  //Konfiguracja licznikow TIM
  TIM3_DeInit(); //de-konfiguracja licznika TIM3

  //Ustawienie prescalera TIM3, dopuszalne potegi 2,
  //16MHz/256=62500 taktow/s, stad okres = 62500 (1 sekunda)
  TIM3_TimeBaseInit(TIM3_PRESCALER_256, 62500);

  //wlacz TIM3
  TIM3_Cmd(ENABLE);
}
```

wskazać nowe źródło sygnału, określić, czy po przełączeniu ma zostać zgłoszone przerwanie oraz czy dotychczasowe źródło sygnału ma pozostać włączone. Tryb przełączenia może być automatyczny lub ręczny. W pierwszym z nich układ przełączający sam decyduje o momencie przełączenia i wykonuje odpowiednią procedurę automatycznie. W trybie ręcznym programista może dokładnie określić moment przełączenia. Z kolei wybór określający stan dotychczasowego źródła sygnału taktującego po przełączeniu pozwala podtrzymać aktywność źródła, jeśli na przykład w programie przełączenie pomiędzy źródłami będzie wykonywane wielokrotnie. Będzie ono wówczas szybsze.

Zastosowana w poprzednim programie pętla opóźniająca pozwoliła na realizację zadania, jakim było miganie diodą, jednak odliczanie czasu było jedynie przybliżone. Rdzeń STM8 wykorzystuje przetwarzanie potokowe, co z jednej strony pozwala znaczne zwiększyć jego wydajność, ale z drugiej m.in. sprawia, że czas trwania poszczególnych instrukcji nie zawsze jest taki sam. Dodatkowo, jeśli w systemie wykorzystywane są przerwania, one również mogą wydłużyć czas trwania zwykłej pętli opóźniającej. Z tego powodu, jeśli w systemie konieczne jest dokładne odmierzanie czasu, skorzystać należy z liczników. Mikrokontrolery STM8S105 wyposażone są w 3 liczniki 16-bitowe i jeden 8-bitowy.

Najbardziej zaawansowanym i oferującym najwięcej funkcji jest licznik 16-bitowy TIM1. Pozwala on zliczać w górę, w dół oraz na zmianę w obu kierunkach. Dodatkowo wyposażony jest w licznik powtórzeń, który pozwala określić, po ilu pełnych cyklach pracy licznik zgłosi zdarzenie przepełnienia. Częstotliwość taktowania licznika może być regulowana dzięki dzielnikowi, który pozwala podzielić sygnał fMASTER przez dowolną wartość z zakresu 1…65536. Oprócz taktowania sygnałem wewnętrznym licznik może być także taktowany sygnałem pochodzącym z zewnątrz. Kolejną cechą licznika są 4 kanały, które pozwalają m.in. na reagowanie na osiągnięcie zadanych

w nich wartości (bez przerywania pracy licznika). Umożliwia to m.in. generowanie sygnału PWM, pomiar parametrów sygnału PWM, pracę w trybie pojedynczych impulsów, sterowanie silnikami prądu stałego oraz współpracę z enkoderami. Zliczanie może być wyzwalane programowo lub sprzętowo, poprzez sygnał na wejściu ETR. Licznik może również zgłaszać przerwania będące wynikiem zdarzeń takich jak: przeładowanie licznika, wyzwolenie zliczania, osiągniecie wartości zadanej dla któregoś z kanałów oraz zewnętrzne zatrzymanie pracy licznika.



Rysunek 6. Uproszczony schemat drzewa sygnałów taktujących mikrokontrolerów STM8S

Liczniki TIM2 i TIM3 również są 16-bitowe, ale ich możliwości są mniejsze. Po pierwsze, mogą zliczać tylko w górę i nie mają liczników powtórzeń. Po drugie mają tylko 3 (TIM2) i 2 (TIM3) kanały. Ogranicza to więc ich wykorzystanie do zliczania, porównywania wartości z zadaną w jednym z kanałów oraz generowania sygnału PWM. Ponadto, przy wyborze sygnału taktującego dostępne są tylko wartości dzielników będące potęgami 2 z zakresu 1…32768.

Ostatni z liczników, TIM4, jest licznikiem 8-bitowym zliczającym w górę. Nie został on wyposażony w kanały, stąd możliwości jego wykorzystania sprowadzają się tylko do odliczania czasu. Wybierając sygnał taktujący ten licznik, można korzystać z dzielników będących potęgami 2 z zakresu 1…128.

Pisanie programu rozpoczniemy od pliku *main.c*, gdzie na jego początku należy dołączyć poleceniem *#include* plik nagłówkowy *stm8s.h*. W przeciwieństwie do programu 1, gdzie cały kod umieszczony został w funkcji *main*( ), podzielimy wykonywane operacje na funkcje. Funkcja *CLK_Config*( ) będzie zawierać polecenia związane z konfiguracją sygnałów taktujących, funkcja *GPIO_Config*( ) – polecenia związane z konfiguracją portu GPIOD, a funkcja *TIM_Config*( ) – polecenia związane z konfiguracją licznika TIM3. Przed funkcją *main*( ) należy umieścić prototypy tych funkcji, a ich kod – na końcu pliku *main.c*. W funkcji *CLK_Config*( ) należy: włączyć oscylator HSE oraz odczekać, aż będzie on gotowy do pracy, wybrać dzielnik częstotliwości dla sygnału fCPU (w naszym przypadku będzie on miał wartość 1) oraz wywołać procedurę przełączenia źródła sygnału taktującego na sygnał pochodzący z HSE. Należy dodać, że w przypadku mikrokontrolerów STM8 standardowo, po starcie układu, sygnały taktujące poszczególne bloki peryferyjne są włączone, nie ma więc konieczności ich aktywacji. Można je natomiast wyłączyć w celu obniżenia poboru energii, gdy któregoś z bloków w tworzonym systemie nie wykorzystujemy. W drugiej z funkcji – *GPIO_Config*( ), skonfigurować należy linię 0 portu GPIOD jako wyjście komplementarne. Z kolei w funkcji *TIM3_Config*( ) należy ustawić dzielnik częstotliwości sygnału taktującego licznik, określić liczbę taktów, po której nastąpi jego przeładowanie (czyli określić okres licznika) oraz licznik włączyć. Zgodnie z założeniami, dioda ma być włączana co 1 s. Jeśli wybierzemy dzielnik 256, wówczas na 1 sekundę przypadać będzie 62500 taktów sygnału taktującego. Taką właśnie wartość należy więc podać jako okres licznika.

Gdy wszystkie funkcje konfigurujące są już gotowe, można przejść do tworzenia kodu funkcji *main*( ). Na jej początku napisane przed chwilą funkcje powinny zostać wywołane, a następnie program powinien wejść w nieskończoną pętlę. W jej wnętrzu należy sprawdzać wartość licznika TIM3, do czego służy funkcja *TIM3_GetCounter*( ). Jeśli stan licznika jest mniejszy niż 16250 (co stanowi ¼ z 65000), wówczas diodę LED należy włączyć (funkcja *GPIO_WriteLow*( )), w przeciwnym wypadku – należy ją wyłączyć (funkcja *GPIO_WriteHigh*( )).

Kod programu 2 pokazany został na **listingu 2**.

W niniejszej, pierwszej części artykułu przedstawione zostały podstawowe cechy mikrokontrolerów STM8S, procedura tworzenia projektu w środowisku STVD oraz dwa przykładowe programy demonstrujące sposób obsługi portu GPIO, liczników uniwersalnych oraz konfigurację sygnałów taktujących. W drugiej części, która ukaże się w następnym numerze EP, pokazane zostanie generowanie sygnału PWM, obsługa przerwań oraz pola dotykowego.

Marek Galewski
marg@mech.pg.gda.pl