DfuSe: USB Device Firmware Upgrade

Alternatywna metoda programowania pamięci Flash mikrokontrolerów STM32

Urządzenie umożliwiające skorzystanie z mechanizmu DFU pełni dwie zamienne funkcje: „właściwego" urządzenia USB oraz programatora pamięci Flash ROM. Ponieważ niepraktyczne byłoby, gdyby obydwie te funkcje dostępne były jednocześnie w czasie wykonywania operacji DFU podstawowe funkcje urządzenia nie są dostępne. Nic nie stoi jednak na przeszkodzie, aby w urządzeniu zaimplementować tylko funkcje DFU, które byłyby wykorzystane do programowania pamięci, nawet wtedy, jeśli urządzenie nie jest przeznaczone do pełnienia swoich podstawowych funkcji w powiązaniu z magistralą USB i wykorzystywana jest ona wyłącznie do ładowania programu do pamięci. W niniejszym artykule przedstawiony zostanie sposób, jak wykorzystać mechanizm DFU do ładowania programu do pamięci mikrokontrolerów STM32 wyposażonych w interfejs USB.

Mikrokontrolery STM32 wyposażone w interfejs USB mogą być programowane za pomocą mechanizmu Device Firmware Upgrade. Mechanizm ten został opracowany na potrzeby aktualizacji przez użytkownika oprogramowania urządzeń pracujących na magistrali USB. Przedstawiamy go w artykule.

Przygotowanie kodu bootloadera DFU

Aby możliwe było wgranie programu do pamięci Flash poprzez łącze USB, konieczne jest zaprogramowanie mikrokontrolera programem realizującym funkcję bootloadera DFU. Oczywiście do tego celu należy wykorzystać typową metodę programowania pamięci, np. poprzez złącze JTAG, albo za pomocą systemowego bootloadera ładującego program przez łącze szeregowe. Kod źródłowy sterownika DFU znajduje się w podkatalogu *Examples\ARM\STM3210B–EVAL\STM32F10xUSBLib\USBLib\demos\Device_Firmware_Upgrade* katalogu instalacyjnego środowiska RIDE (typowo *C:\Program Files\Raisonance\Ride*), które publikujemy na CD–EP6/2008B (jest także dostępne na stronie *www.raisonance.com*). W podkatalogu *Project/RIDE* znajduje się plik projektu, który należy otworzyć w środowisku RIDE. Przyjrzyjmy się zawartości pliku *main.c*, a konkretnie fragmentowi funkcji *main*, którego treść przedstawiono na **list. 1**.

Fragment ten jest dla nas najistotniejszą częścią programu sterownika DFU. Do zmiennej *JumpAddress* jest przypisywany adres, pod którym będzie się znajdować ładowany za pomocą mechanizmu DFU program użytkownika, a dokładniej tablica wektorów przerwań. Adres ten musi wskazywać na drugie słowo obszaru kodu programu (czyli wektor zerowania), gdyż w przypadku programów dla rdzenia ARM Cortex–M3 pierwsze słowo pamięci programu zawsze zawiera wartość początkową wskaźnika stosu. Program użytkownika, który będzie ładowany za pomocą DFU musi być odpowiednio przygotowany. Sposób przygotowania programu użytkownika zostanie omówiony w dalszej części artykułu. Po zainicjowaniu

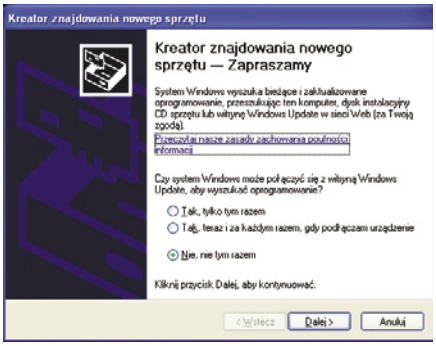
Jako platformę sprzętową autor wykorzystał zestaw STM3210B–EVAL, ale prezentowana aplikacja będzie funkcjonować także w innych środowiskach sprzętowych mikrokontrolerów STM32.

List. 1.

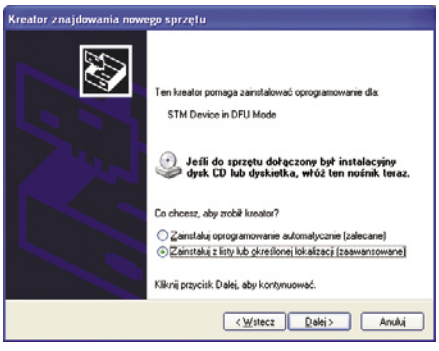
```
JumpAddress = *(vu32*)0x8003804;

DFU_Button_Config();

/* Test if PB.09 level is low (Key push-button on Eval Board pressed) */
if (DFU_Button_Read() != 0x00)
{ /* Test if user code is programmed starting from address 0x8003800 */
  if (((*(vu32*)0x8003800) & 0x2FFF0000 ) == 0x20000000)
  { /* Jump to user application */
    Jump_To_Application = (pFunction) JumpAddress;
    Jump_To_Application();
  }
  /* Otherwise enters DFU mode to allow user to program his application */
}
```

zmiennej *JumpAddress* następuje konfiguracja wyprowadzenia PB9, do którego jest dołączony przycisk uruchamiający sterownik DFU. Bootloader DFU zostanie uruchomiony, jeśli przycisk podłączony do wyprowadzenia PB9 został naciśnięty. W przypadku, gdy przycisk nie został naciśnięty sprawdzana jest wartość pierwszego słowa pamięci programu użytkownika. Jeżeli wartość ta wskazuje poza obszar pamięci RAM, można uznać, że program użytkownika nie został do tej pory załadowany i również zostanie uruchomiony bootloader DFU. Jeśli natomiast wartość słowa odczytanego spod adresu pierwszego słowa programu użytkownika wskazuje na dowolną komórkę pamięci RAM, następuje pobranie wartości drugiego słowa pamięci programu (0x8003804) będącego wektorem zerowania wskazującym na pierwszą instrukcję programu.

W celu otrzymania pliku z kodem wynikowym bootloadera DFU należy przeprowadzić kompilację projektu. Proces kompilacji powinien zakończyć się sukcesem i w katalogu z projektem powinien zostać utworzony plik **.hex*. Plikiem tym należy zaprogramować mikrokontroler za pomocą interfejsu JTAG lub RS232 z wykorzystaniem bootloadera znajdującego się w pamięci systemowej mikrokontrolera.



Rys. 1.



Rys. 2.

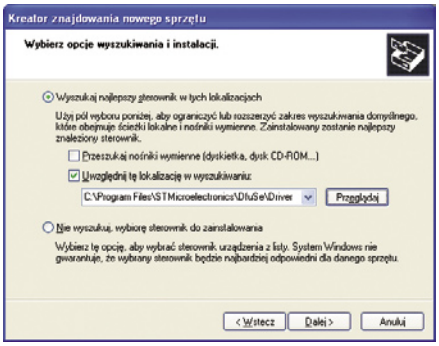
Instalacja oprogramowania

W celu załadowania programu za pomocą mechanizmu DFU należy zainstalować program *DfuSe Demonstrator* (*http://www.st.com/stonline/products/support/micro/files/um0412.zip*, dostępny także na CD–EP6/2008B). Instalacja programu przebiega w typowy dla aplikacji systemu Windows sposób. W katalogu instalacyjnym programu znajduje się podkatalog *Driver* z plikami sterowników dla systemu Windows pozwalającymi na komunikację z mikrokontrolerem pracującym w trybie DFU. Katalog ten należy wskazać instalatorowi sprzętu po wykryciu przez system mikrokontrolera podłączonego za pomocą łącza USB do komputera PC.

Instalacja sprzętu

Kolejnym etapem po zainstalowaniu aplikacji sterującej jest zainstalowanie w systemie Windows sterowników DFU. W tym celu podłączamy kablem USB A–B na przykład płytkę STM3210B–EVAL do wolnego portu USB w komputerze oraz włączamy zasilanie płytki.

System Windows powinien wykryć nowy sprzęt, po czym zostanie uruchomiony *Kreator znajdowania nowego sprzętu* (**rys. 1**), w którego oknie powitalnym zaznaczamy opcję *Nie, nie tym razem* i klikamy przycisk *Dalej*. W następnym oknie zaznaczamy opcję *Zainstaluj z listy lub określonej lokalizacji (zaawansowane)* (**rys. 2**). W kolejnym oknie wskazujemy ścieżkę dostępu do wspomnianego wcześniej katalogu ze sterownikami: *C:\Program Files\STMicroelectronics\DfuSe\Driver* (**rys. 3**). Po ukazaniu się ostrzeżenia



Rys. 3.

informującego, że instalowane sterowniki nie przeszły testów zgodności z systemem Windows XP należy kliknąć przycisk *Mimo to kontynuuj*. Sterowniki powinny zostać pomyślnie zainstalowane.

Po uruchomieniu programu *DfuSe Demonstrator* w polu *Available DFU and compatibile HID devices* na liście rozwijanej powinna być widoczna pozycja *STM device in DFU mode* (**rys. 4**). W oknie zostaną wyświetlone informacje o urządzeniu, takie jak numery *VendorID*, *ProductID* oraz numer wersji oprogramowania. Numery te są istotne i zostaną wykorzystane do przygotowania pliku **.dfu*, dlatego należy je zapisać, bądź zapamiętać.

Oprócz tego zostaną wyświetlone dostępne obszary pamięci do zaprogramowania – w tym przypadku pamięć wewnętrzna mikrokontrolera (*Internal Memory*) oraz zewnętrzna pamięć SPI (*SPI Flash: M25P64*) znajdująca się na płytce zestawu STM3210B–EVAL.

Przygotowanie pliku z kodem programu

Program, który będzie wgrywany poprzez mechanizm DFU musi zostać dostosowany do tego celu poprzez zmianę adresu początku sekcji Flash. W tym celu należy wejść do katalogu *C:\Program Files\Raisonance\Ride\Lib\ARM* i skopiować plik *STM32F103_128K_20K_DEF.ld* pod inną nazwę, np. *STM32F103_128K_20K_DEF_DFU.ld*.

Nowy plik należy otworzyć edytorem tekstowym i dokonać zmiany adresu sekcji Flash z domyślnej:
FLASH (rx) : ORIGIN = 0x8000000, LENGTH = 128K
na odpowiednią dla współpracy z DFU:
FLASH (rx) : ORIGIN = 0x8003800, LENGTH = 114K

Następnie podobnie postępujemy z plikiem *STM32F103_128K_20K_FLASH.ld* zmieniając jego nazwę na *STM32F103_128K_20K_FLASH_DFU.ld* oraz otwieramy go w edytorze tekstowym i zmieniamy wpis:
INCLUDE „STM32F103_128K_20K_DEF.ld"
na właściwy:
INCLUDE „STM32F103_128K_20K_DEF_DFU.ld"

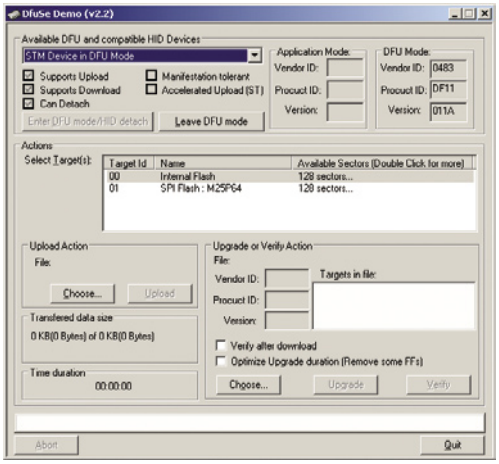
Teraz pozostaje tylko poinformować kompilator o zmianie skryptu linkera. W tym celu w oknie ustawień projektu w sekcji *LD Linker* w kategorii *Scripts* zmieniamy opcję *Use default script file* na *No* oraz w polu *Script file* wskazujemy plik *STM32F103_128K_20K_FLASH_DFU.ld* (**rys. 5**).

Poza zmianą skryptu linkera konieczne jest jeszcze poinformowanie kontrolera przerwań NVIC o aktualnej lokalizacji wektorów przerwań, która obecnie znajduje się pod adresem 0x80003800. W tym celu należy skorzystać z funkcji *NVIC_SetVectorTable*:
NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x3800);

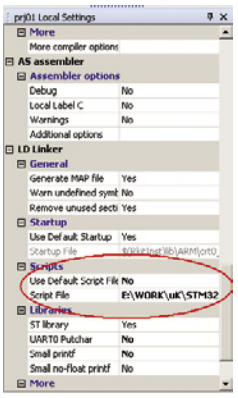
Funkcja ta musi zostać wywołana z programu głównego przed uaktywnieniem jakiegokolwiek przerwania.

Przykładowy kod programu, służącego do sprawdzenia poprawności wszystkich opisanych uprzednio czynności przedstawiono na **list. 2**.

Program ten realizuje bardzo prostą funkcję, miganie diodą LED podłączoną do wyprowadzenia PC6, wystarczającą jednak do sprawdzenia poprawności przebiegu procesu ładowania programu.



Rys. 4.



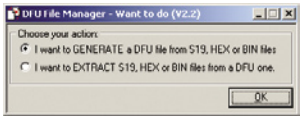
Rys. 5.

Przygotowanie pliku DFU

W celu zaprogramowania pamięci mikrokontrolera z wykorzystaniem mechanizmu DFU należy przygotować plik **.dfu*, który oprócz kodu programu zawiera dodatkowe informacje, jak np. numery VID i PID urządzenia USB, dla którego jest przeznaczony ładowany program oraz numer wersji programu. Dzięki temu nie jest możliwe zaprogramowanie mikrokontrolera programem przeznaczonym dla innego urządzenia, bądź też w starszej wersji od aktualnie znajdującej się w urządzeniu.

Do przygotowania pliku **.dfu* służy program *DFU File Manager*, który został zainstalowany razem z programem *DfuSe Demonstrator*. Po jego uruchomieniu pojawi się okno służące do wyboru czynności, jaką chcemy przeprowadzić na pliku **.dfu* (**rys. 6**). Nas oczywiście interesuje opcja *I want to GENERATE a DFU file from S19, HEX or BIN files*, która jest domyślnie zaznaczona. Po kliknięciu przycisku OK ukaże się okno służące do wprowadzenia danych niezbędnych do przygotowania pliku DFU (**rys. 7**). Są to numery Vendor ID, Product ID oraz numer wersji oprogramowania, ścieżki dostępu do

List. 2.
```
#include „stm32f10x_lib.h"

GPIO_InitTypeDef GPIO_InitStructure;

int main(void)
{
vu32 i;

NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x3800);

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);

GPIO_InitStructure.GPIO_Pin   =  GPIO_Pin_6;
GPIO_InitStructure.GPIO_Speed =  GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode  =  GPIO_Mode_Out_PP;
GPIO_Init(GPIOC, &GPIO_InitStructure);

do{
  GPIO_SetBits(GPIOC, GPIO_Pin_6);
  for(i = 0; i < 0x50000; i++);
  GPIO_ResetBits(GPIOC, GPIO_Pin_6);
  for(i = 0; i < 0x50000; i++);
  }while(1);
return 0;
}
```

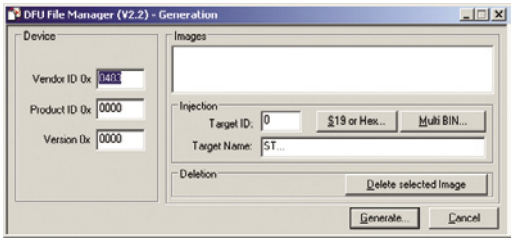


Rys. 6.

pliku z kodem itp. Pole Vendor ID jest domyślnie uzupełnione wartością 0x0483 będącą identyfikatorem firmy STMicroelectronics. W polu Product ID należy wpisać wartość odczytaną z okna głównego programu *DfuSe Demonstrator*. Z kolei numer wersji powinien przyjąć wartość następną w stosunku do numeru aktualnie znajdującego się oprogramowania.

Programowanie pamięci

Po wygenerowaniu pliku DFU nie pozostaje już nam nic innego, jak tylko przesłać go do mikrokontrolera w celu zaprogramowania pamięci programu. W tym celu należy ponownie uruchomić program *DfuSe Demonstrator* i w polu *Upgrade or Verify Action* kliknąć przycisk *Choose* wskazując przygotowany w poprzednim kroku plik **.dfu*. W przypadku pomyślnego załadowania pliku **.dfu* na pasku statusu wyświetlony zostanie tekst *File correctly loaded*. Wyświetlone zostaną również informacje na temat załadowanego pliku: numery VID, PID urządzenia dla którego jest przeznaczony oraz wersji zawartego w nim oprogramowania. Po kliknięciu przycisku *Upgrade* do mikrokontrolera zostanie przesłany kod programu zawartego w pliku **.dfu*. Po wyzeorwaniu mikrokontrolera przesłany program powinien się uruchomić, natomiast urządzenie DFU powinno zniknąć z listy *Available DFU and compatible HID Devices*. Aby ponownie wprowadzić mikrokontroler w tryb DFU należy go wyzerować przy niskim stanie na wyprowadzeniu PA9 (wciśnięty przycisk KEY na płytce zestawu STM3210B–EVAL).



Rys. 7.

Podsumowanie

Wykorzystanie mechanizmu DFU do programowania pamięci mikrokontrolerów STM32 na etapie tworzenia programu jest nieco mozolne ze względu na konieczność ręcznego tworzenia każdorazowo pliku **.dfu*, jednak jest ciekawą alternatywą do klasycznych metod programowania wykorzystujących JTAG czy wbudowany bootloader przez RS-232 pozwala na zaprogramowanie mikrokontrolera bez większych nakładów, np. na programator JTAG pracujący na łączu USB. Zarówno specyfikacja pliku **.dfu*, jak i źródłowa postać programów opisanych w artykule są dostępne w podkatalogu *Sources* katalogu instalacyjnego programu *DfuSe Demonstrator*, w związku z czym można się pokusić o zautomatyzowanie procesu tworzenia pliku **.dfu* z pliku **.hex*.

Radosław Kwiecień, EP
radoslaw.kwiecien@ep.com.pl