Przedmowa

Niniejsza książka przeznaczona jest dla niewtajemniczonych w arkana USB, a w szczególności dla tych, którzy nigdy nie pisali aplikacji wykorzystujących mikrokontroler i interfejs USB, a chcieliby zacząć takie aplikacje tworzyć. Starałem się, aby zainteresowała zarówno hobbystów, jak i studentów kierunków związanych z elektroniką lub informatyką, ale także doświadczonych konstruktorów systemów mikroprocesorowych. Od Czytelnika oczekuję tylko, choćby minimalnego, doświadczenia w programowaniu mikrokontrolerów i znajomości przynajmniej podstaw języka C.

Jeszcze całkiem niedawno układy mikrokontrolerowe z dużymi komputerami łączono głównie za pomocą interfejsu szeregowego RS-232, a czasem równoległego Centronics. Niegdyś każdy komputer osobisty (pecet) był wyposażony w te interfejsy. Obecnie zostały one całkowicie wyparte przez uniwersalny interfejs szeregowy USB (ang. *Universal Serial Bus*). Jeśli potrzebujemy podłączyć do komputera mikrokontroler, który nie obsługuje USB, zwykle stosujemy konwerter USB/RS-232. Zaletą tego rozwiązania jest wsteczna kompatybilność – można używać istniejącego i przetestowanego oprogramowania zarówno po stronie mikrokontrolera, jak i po stronie komputera. Poważną wadą jest niewykorzystywanie wszystkich możliwości, jakie oferuje USB. Można też znaleźć w Internecie całkowicie programowe implementacje USB, symulujące ten interfejs na wejściach-wyjściach ogólnego przeznaczenia, czyli GPIO (ang. *General Purpose Input-Output*). Rozwiązania te obsługują tylko bardzo ograniczony podzbiór wymagań standardu USB, więc formalnie nie są z nim zgodne. Zatem nie są godne polecenia i nie są omawiane w tej książce.

Obecnie mikrokontrolery coraz częściej wyposażane są w układ peryferyjny realizujący sprzętowo interfejs urządzenia USB. Dzięki temu można wyeliminować konwerter (zmniejszając koszt urządzenia) i wykorzystać wszystkie zalety USB. Dodatkowo, jeśli ten układ peryferyjny obsługuje protokół kontrolera (ang. *host*) USB, można do mikrokontrolera podłączać urządzenia USB: mysz, klawiaturę, pamięć Flash itp. Niestety USB jest bardzo skomplikowanym interfejsem. Podstawowy dokument, opisujący wersję 2.0 standardu, ma ponad 600 stron. Każda klasa urządzeń opisana jest w dodatkowym dokumencie, a często w kilku osobnych dokumentach. Do tego dochodzą komentarze do standardu, opisy jego rozszerzeń i erraty. Razem jest to kilka tysięcy stron. USB zaprojektowano, aby ułatwić życie użytkownikom – urządzenie podłącza się do gniazda i już wszystko działa (ang. *plug and play*). Niestety łatwość używania jest okupiona sporym utrapieniem dla programistów. Protokoły USB są bardzo rozbudowane i trzeba się mocno napracować, aby je poprawnie zaimplementować. Obsługa USB od strony mikrokontrolera też przysparza sporo trudności. Aby zaprogramować UART lub USART obsługujący RS-232, wystarczy skonfigurować kilka rejestrów. Układ peryferyjny USB ma co najmniej kilkadziesiąt rejestrów, których funkcje trzeba poznać. Wszystko to

sprawia, że przejście tej pierwszej bariery programistycznej, napisanie pierwszej aplikacji korzystającej z USB jest dla początkującego niezwykle trudne. Celem tej książki jest możliwie bezbolesne wprowadzenie Czytelnika w świat USB.

Na kilku przykładach pokazuję, jak zaprogramować USB w mikrokontrolerze STM32 z rdzeniem ARM Cortex-M3 lub Cortex-M4. Przykłady starałem się napisać możliwie ogólnie. Jeśli w przyszłości pojawią się inne modele STM32 (również z rdzeniem Cortex-M0) z interfejsem USB, to przykłady te dadzą się na nie przenieść bez większych problemów. Wyraźnie rozdzielam warstwą zależną od sprzętu od warstwy protokołu. Warstwa protokołu, z uwagi na swoją niezależność od sprzętu, daje się bardzo łatwo zaadaptować na inne mikrokontrolery. Książka ta nie aspiruje do bycia wyczerpującym kompendium wiedzy o USB. Prezentowane przykłady nie demonstrują wszystkich możliwości USB. Z uwagi na ogrom tego, co zawiera standard, jest to po prostu niemożliwe. Opisuję głównie te zagadnienia, które mogą przydać się przy programowaniu mikrokontrolerów, w nadziei że staną się one inspiracją do własnych udanych projektów. Zainteresowany Czytelnik z pewnością zechce kiedyś poznać więcej szczegółów działania USB i zajrzy do standardu, co – jak myślę – po przeczytaniu tej książki będzie łatwiejsze. W efekcie książka ta jest bardziej o USB niż o mikrokontrolerach STM32, które dały pierwotny pretekst do jej napisania.

Do omówienia przykładów wybrałem kilka modeli STM32, tak aby reprezentowały one możliwie pełne spektrum dostępnych obecnie układów z tej rodziny mikrokontrolerów. Przykłady uruchamiałem na modelach STM32F103, STM32F107, STM32F207, STM32F407 i STM32L152, stosując zestawy STM3220G-EVAL, ZL29ARM, ZL30ARM, ZL31ARM oraz moduły STM32F4-Discovery, STM32L--Discovery, MMstm32F103Vx-0-0-0, a także specjalnie zaprojektowaną na potrzeby tej książki płytkę nazywaną dalej gadżetem. Dzięki elastycznemu sposobowi konfigurowania układów peryferyjnych (zarówno wewnętrznych bloków mikrokontrolera, jak i układów zewnętrznych) przykładowe programy mogą być z łatwością przeniesione na inne modele STM32 wyposażone w interfejs USB oraz inne płytki.

W rozdziale 1 opisuję architekturę i podstawowe protokoły USB. Koncentruję się głównie na wersji 2.0, która jest obsługiwana przez układy peryferyjne USB zastosowane w STM32. Jest to jedyny „teoretyczny" rozdział w książce, tzn. niezawierający żadnego przykładu. Materiał przedstawiony we wszystkie pozostałych rozdziałach ilustruję przykładami programów. Jak wspomniałem wyżej, oficjalna dokumentacja USB jest bardzo rozbudowana, a wiele jej fragmentów przeznaczonych jest głównie dla osób, które chcą projektować sprzęt realizujący interfejs USB. Programista lub konstruktor układów mikroprocesorowych, chcący tworzyć aplikacje korzystające z USB, nie potrzebuje aż tak szczegółowej dokumentacji. Przystępując do pisania tej książki, odczułem brak dobrego i zwartego kompendium na temat USB (zwłaszcza w języku polskim) przydatnego programistom i konstruktorom układów mikroprocesorowych. Celem pierwszego rozdziału jest właśnie choćby częściowe wypełnienie tej luki.

W rozdziale 2 pokazuję warianty sprzętu, którego używałem do testowania przykładowych programów. Przedstawiam stosowaną konwencję nazywania plików

źródłowych. Opisuję, jak skompilować przykłady. Sporo miejsca poświęcam też ogólnym rozważaniom, jak poprawnie pisać programy do wielu wariantów sprzętu. Omawiam, jak dostosować projekty do dowolnej konfiguracji sprzętu. Rozdział 2 zawiera ponadto opis pomocniczych funkcji wykorzystywanych w przykładach. Rozdział ten kończy się projektem, który demonstruje konfigurowanie wariantu sprzętu i taktowania mikrokontrolera oraz obsługę błędów. Celem tego projektu jest przedstawienie procedur niezwiązanych bezpośrednio z USB, ale stosowanych we wszystkich następnych projektach. Ponadto przykład ten umożliwia szybkie przetestowanie zestawu narzędzi programistycznych (ang. *toolchain*) i sprzętu (płytka prototypowa, programator, adapter JTAG).

W rozdziale 3 przedstawiam trzy projekty prostych urządzeń USB pełnej szybkości (ang. *full speed*), demonstrujące wszystkie cztery, zdefiniowane w standardzie USB, tryby przesyłana danych (ang. *control*, *interrupt*, *bulk*, *isochronous*). Każde z tych urządzeń można podłączyć do komputera osobistego pracującego pod kontrolą systemu Windows lub Linux. Od strony programistycznej omawiam tylko warstwę aplikacji, odkładając dokładny opis implementacji protokołu USB do następnego rozdziału. Projekt 1 demonstruje przesyłanie pilne (ang. *interrupt transfer*). Urządzenie jest rozpoznawane jako mysz. Do zmiany położenia wskaźnika myszy i klikania jej przyciskami służą dżojstik i przyciski monostabilne zainstalowane na płytce prototypowej lub akcelerometr MEMS. Projekt 2 demonstruje przesyłanie masowe (ang. *bulk transfer*). Urządzenie jest rozpoznawane jako wirtualny port szeregowy, czyli COM w Windows, a /dev/ttyACM w Linuksie. Można się z nim skomunikować za pomocą popularnych programów: HyperTerminal (Windows), PuTTY (Windows), Minicom (Linux). Wpisując odpowiednie polecenie, można sterować diodami świecącymi podłączonymi do mikrokontrolera. Opcjonalnie, jeśli układ jest wyposażony w LCD, wyświetlane są na nim informacje diagnostyczne. Projekt 3 demonstruje przesyłanie izochroniczne (ang. *isochronous transfer*) strumienia danych audio. Urządzenie jest rozpoznawane jako głośnik, który może być używany przez dowolny program odtwarzający dźwięk. Po podłączeniu urządzenia na ekranie komputera pojawia się ikona głośnika. Kliknięcie tej ikony otwiera panel z kontrolkami regulacji głośności i wyciszania. Opcjonalnie, jeśli układ jest wyposażony w LCD, wyświetlane są na nim informacje diagnostyczne. W każdym z powyższych trzech przykładów pojawiają się też komunikaty przesyłane jako dane sterujące (ang. *control transfer*).

W rozdziale 4 przedstawiam pierwszą część, dotyczącą urządzenia USB, napisanej przeze mnie biblioteki upraszczającej korzystanie z USB na mikrokontrolerach STM32. Wszystkie prezentowane w książce przykłady bazują na tej bibliotece. Omawiam też bibliotekę libusb, która ułatwia pisane aplikacji korzystających z urządzeń USB po stronie komputera osobistego. Rozdział kończy się projektem pokazującym, jak zaimplementować urządzenie USB własnej klasy (ang. *vendor specific*). Program demonstruje przesyłanie danych sterujących, pilnych, masowych i izochronicznych, w obu kierunkach: kontroler-urządzenie i urządzenie-kontroler.

Rozdział 5 poświęcony jest zarządzaniu zasilaniem interfejsu USB. Omawiam w nim, zamieszczone w standardzie, wymagania dotyczące poboru prądu przez urządzenie USB. Opisuję praktycznie stosowane sposoby realizacji tych wymagań.

Projekt, kończący ten rozdział, demonstruje automatyczne usypianie urządzenia przy braku aktywności na szynie (ang. *auto suspend*), budzenie urządzenia (ang. *wakeup*) i zdalne budzenie kontrolera (ang. *remote wakeup*).

W rozdziale 6 omawiam modyfikacje, jakie trzeba wprowadzić w implementacji urządzenia, aby korzystać z transmisji wysokiej szybkości (ang. *high speed*) wprowadzonej w wersji 2.0 standardu USB. Jako przykładowy projekt proponuję zaimplementowanie zewnętrznej pamięci masowej korzystającej z przesyłania masowego (ang. *bulk*). Urządzenie to jest rozpoznawane jako zewnętrzny dysk.

Dwa ostatnie rozdziały poświęcone są kontrolerom USB. W rozdziale 7 przedstawiam drugą część napisanej przeze mnie biblioteki. Opisuję moduły umożliwiające zaimplementowanie prostego kontrolera USB. Po podłączeniu do niego urządzenia USB na LCD wyświetlane są podstawowe informacje o nim: szybkość transmisji, VID, PID, klasa i podklasa oraz obsługiwany protokół, znaczniki związane z zarządzaniem energią (ang. *self powered*, *remote wakeup*) oraz deklarowany przez urządzenie limit poboru prądu z szyny. Jeśli są dostępne, wyświetlane są nazwa producenta i nazwa urządzenia. Ponadto kontroler ten obsługuje protokół fazy rozruchu (ang. *boot protocol*) urządzeń klasy HID (ang. *Human Interface Device*). Jeśli podłączone urządzenie jest myszą, to wyświetlany jest stan przycisków oraz współrzędne położenia jej wskaźnika. Jeśli podłączone urządzenie jest klawiaturą, to wyświetlana jest informacja o wciśniętych klawiszach. W rozdziale 8 przedstawiam projekt kontrolera obsługującego pamięć masową USB Flash. Po podłączeniu takiego urządzenia przykładowa aplikacja przeprowadza test, który polega na odczytywaniu i zapisywaniu plików.

Wszystkie pliki źródłowe prezentowanych w tej książce projektów oraz inne pliki, które są pomocne przy ich kompilowaniu i testowaniu, znajdują się w skompresowanym archiwum, które można ściągnąć ze strony Wydawnictwa BTC *http://www.btc.pl* lub z mojej strony domowej *http://www.mimuw.edu.pl/~marpe/book*. Dla pełnego zrozumienia przykładów konieczne jest zajrzenie do ich tekstu źródłowego, do czego gorąco zachęcam, tym bardziej że, z uwagi na dużą objętość tych źródeł, w samej książce zamieszczam tylko kluczowe fragmenty programów, przede wszystkim te, które dotyczą bezpośrednio USB. Nie omawiam szczegółów związanych z konfigurowaniem rejestrów peryferyjnych – te można łatwo wyczytać z plików źródłowych.

W książce staram się promować polską terminologię. Nie chciałbym, aby literatura techniczna była zaśmiecana angielskimi terminami lub ich nieudolnymi tłumaczeniami. Dla wielu angielskich terminów istnieją dobre i od dawna ugruntowane, ale być może zapomniane, polskie odpowiedniki. Aby jednak ułatwić posługiwanie się dokumentacją w języku angielskim, przy pierwszym, a czasem i przy kolejnym użyciu terminu, który nie jest powszechnie znany, umieszczam w nawiasach jego angielski odpowiednik.

Marcin Peczarski, Warszawa 2013