Konfiguracja układów FT8Uxx

Układy interfejsu USB firmy FTDI dosyć szybko stały się standardem w amatorskich konstrukcjach. Pozwalają na nawiązanie komunikacji z hostem USB (zazwyczaj komputerem) bez wnikania w szczegóły działania sprzętowo-programowego stosu obsługi magistrali. W najprostszym przypadku możemy je zastosować praktycznie bez żadnych dodatkowych zabiegów, jednak wykorzystanie w pełni możliwości kostek wraz z towarzyszącymi sterownikami oraz bibliotekami wymaga znacznie głębszego zapoznania się z zasadami ich konfiguracji.

Używanie FT8Uxx jako wirtualnego portu szeregowego

1. Zasady ogólne

Jest to najprostszy sposób użycia. Instalując w systemie odpowiedni sterownik (VCP – *virtual COM port*), powodujemy, że FT8Uxx jest widziany jako dodatkowy, zwykły port komunikacji szeregowej. Może być wykorzystywany bez żadnych dodatkowych zabiegów przez istniejące aplikacje. Indywidualna konfiguracja kostki nie jest wymagana, można nawet wcale nie montować pamięci EEPROM. Tego typu rozwiązanie zastosujemy, budując adapter przejściowy USB<->RS232 potrzebny w razie braku w komputerze odpowiedniej liczby portów COM. Musimy jednak mieć na uwadze pewne ograniczenia:

– Pakietowa struktura transmisji danych w magistrali USB może wprowadzać różnego rodzaju opóźnienia. Nie ma to zazwyczaj większego znaczenia dla zwykłych aplikacji wysyłających i odbierających dane, może jednak uniemożliwić pracę programów narzędziowych korzystających z krótkich *time-outów* oraz dodatkowych sygnałów kontrolnych portu (programatory, symulatory itp.).
– Nie będą działać stare programy DOS-owe odwołujące się bezpośrednio do rejestrów i przerwań portu (pod Windows sterownik VCP przejmuje wywołania funkcji WinAPI obsługujących port – użycie portu w programie musi być oparte na tych funkcjach).
– Wiele starszych aplikacji Windows umożliwia wybór portu szeregowego jedynie w ograniczonym zakresie (1 – 4). Wprawdzie sterownik VCP umożliwia dowolną zmianę numeru portu, ale musimy uważać na potencjalne konflikty sprzętowe ze zwykłymi, istniejącymi w maszynie portami COM.

Tab. 1

16,15,14 = 000	podzielnik ułamkowy = 0
16,15,14 = 001	podzielnik ułamkowy = 0,5
16,15,14 = 010	podzielnik ułamkowy = 0,25
16,15,14 = 011	podzielnik ułamkowy = 0,125
16,15,14 = 100	podzielnik ułamkowy = 0,375
16,15,14 = 101	podzielnik ułamkowy = 0,625
16,15,14 = 110	podzielnik ułamkowy = 0,75
16,15,14 = 111	podzielnik ułamkowy = 0,875

2. Ustawianie dowolnej szybkości transmisji

W zamian za te niewielkie niedogodności mamy możliwość znacznie bardziej elastycznego niż w zwykłym porcie ustawienia szybkości transmisji. W kostce FT8U232 źródłem taktowania transmisji jest wewnętrzny sygnał zegarowy 48 MHz zredukowany w preskalerze (z podziałem przez 16) do wartości 3 MHz, która z kolei dzielona jest przez odpowiedni (ustawiany programowo) współczynnik. Dla zwiększenia dokładności dzielnik posiada część całkowitą oraz ułamkową. Dla wersji AM część ułamkowa mogła wynosić: 0,5 – 0,25 – 0,125. Wersja BM jest wyposażona znacznie lepiej: 0,875 – 0,75 – 0,625 – 0,5 – 0,375 – 0,25 – 0,125. Oczywiście nie da się dokładnie ustawić każdej wartości, jednak biorąc pod uwagę dopuszczalne odchyłki szybkości transmisji (±3%), pole manewru jest naprawdę szerokie.

Ustawienie dowolnej szybkości jest zupełnie proste w aplikacji pisanej specjalnie pod kątem użycia FT8U232 (służą do tego opisane dalej funkcje biblioteczne), gorzej ze zwykłymi programami, które nic „nie wiedzą" o dzielnikach i dopuszczają wybór szybkości jedynie spośród typowego dla standardu szeregu. W takim przypadku możemy jednak zastąpić niektóre typowe wartości własnymi.

Sterownik VCP przy konfigurowaniu portu wpisuje do kostki podzielnik odpowiadający wartości *baud rate* wybranej w uruchamianym programie. Odpowiedni szereg wartości podzielników jest zapisany tekstowo w pliku *ftdiport.inf*. Jeśli więc zmienimy wpis – dotychczasowej typowej szybkości będzie odpowiadać całkiem nowa wartość podzielnika (czyli nowa, potrzebna nam szybkość transmisji).

Struktura klucza [*FtdiPort232.HW.AddReg*] ([*FtdiPort232.NT.HW.AddReg*] dla Windows 2000/XP) w pliku jest następująca (dla układu w wersji BM):

```
HKR,,ConfigData,1,11,00,3F,3F,
10,27,00,00,88,13,00,00,C4,09,00,00,E2,04,00,00,
71,02,00,00,38,41,00,00,9C,80,00,00,4E,C0,00,00,
34,00,00,00,1A,00,00,00,0D,00,00,00,06,40,00,00,
03,80,00,00,00,00,00,00,D0,80,00,00
```

W pierwszej linii znajdują się ogólne informacje (w tym flaga wersji układu), następne pozycje zaznaczone na przemian pogrubieniem i kursywą to 4-bajtowe bloki odpowiadające kolejnym szybkościom transmisji. Bajty podane są w kolejności od najmłodszego w zapisie heksadecymalnym – czyli np. pozycja 38,41,00,00 odpowiada 32-bitowej liczbie 0x4138 = *0011* 00 0001 0011 1000. Bity 0 – 13 określają wartość całkowitej części dzielnika – tutaj jest ona równa 1 0011 1000 = 0x138 = 312. Bity 14 – 16 (kursywa) kodują wartość ułamkową jak w **tab. 1**.

– w naszym przypadku wynosi ona 0,5. Wyliczona wartość podzielnika – 312,5 odpowiada szybkości 3000000/312,5 = 9600 baud. Korektę wpisu przeprowadzamy odwrotnie (np. chcąc zamienić ustawienie 9600 baud na 250 000 baud, wyznaczymy dzielnik jako 3000000/250000 = 12.00; część ułamkowa = 0, czyli bity 14-16 = 000, a więc kod szybkości wyniesie liczbowo 0x0c, natomiast

Tab. 2

10,27,00,00	podzielnik = 10000,	szybkość = 300
88,13,00,00	podzielnik = 5000,	szybkość = 600
C4,09,00,00	podzielnik = 2500,	szybkość = 1200
E2,04,00,00	podzielnik = 1250,	szybkość = 2,400
71,02,00,00	podzielnik = 625,	szybkość = 4,800
38,41,00,00	podzielnik = 312,5,	szybkość = 9,600
9C,80,00,00	podzielnik = 156,	szybkość = 19,230
4E,C0,00,00	podzielnik = 78,	szybkość = 38,461
4,00,00,00	podzielnik = 52,	szybkość = 57,692
1A,00,00,00	podzielnik = 26,	szybkość = 115,384
0D,00,00,00	podzielnik = 13,	szybkość = 230,769
06,40,00,00	podzielnik = 6,5,	szybkość = 461,538
03,80,00,00	podzielnik = 3,25,	szybkość = 923,076
00,00,00,00	rezerwa	
D0,80,00,00	podzielnik = 208.25,	szybkość = 14406

zapisany tekstowo – 0C,00,00,00, co wstawiamy zamiast dotychczasowego 38,41,00,00).

Przypisanie pozycji typowym wartościom zawarto w **tab. 2**.

Zwróćmy jeszcze uwagę na to, że wersja AM ma zapisy nie 4-, a 2-bajtowe, do zakodowania 4 wartości ułamkowych wystarczają bity 14 i 15 każdej pozycji. Rodzaj zapisu określa bit 4. w bajcie flag, dla AM jest on równy zero, a początek klucza ma postać: HKR,,ConfigData,1,**01**,00,3F,3F,...).

Tab. 3. Budowa deskryptora urządzenia

Nazwa pola deskryptora urządzenia	FT8Uxx
Całkowita długość (w bajtach) deskryptora (18)	0x12
Kod typu deskryptora = 1	0x1
Numer wersji interfejsu USB (1.10)	0x10,0x01
Klasa urządzenia	0
Podklasa urządzenia	0
Typ protokołu	0
Rozmiar endpointu kontrolnego EP0	0x08
Numer identyfikacyjny wytwórcy – VID (vendor ID)	**0x03,0x04**
Numer identyfikacyjny produktu – PID (product ID)	**0x01,0x60**
Numer wersji	**0x00,0x02**
Indeks nazwy producenta	0x01
Indeks nazwy produktu	0x02
Indeks numeru seryjnego produktu	0x03/ **0x0**(BM)
Liczba dostępnych konfiguracji	0x01

*Uwaga do tabeli
Kolejne cyfry numerów są zapisywane heksadecymalnie w kodzie BCD, młodszy bajt na początku, np.: 0x01,0x60 oznacza PID = 6001; 0x10,0x01 oznacza wersję 1.10 itd.

Powyższe zamiany prawdopodobnie nigdy nie będą potrzebne przy podłączaniu różnych typowych urządzeń. Jednak podczas budowy własnych konstrukcji opartych o mikrokontrolery ich komunikacja z komputerem nareszcie przestaje wymagać skrupulatnego doboru odpowiednich rezonatorów (np. powyższy przykład z 250 kbaud świetnie pasuje do mikrokontrolera AtMega z wewnętrznym oscylatorem 8MHz).

Należy także zaznaczyć, że ustawienia te mają sens jedynie dla kostki 8U232. Układ 8U245 zawsze prowadzi transmisję z maksymalną osiąganą prędkością (wartość *baud rate* jest ignorowana).

Zapisy konfiguracyjne w zewnętrznej pamięci EEPROM

1. Struktura i zawartość deskryptorów układu

Po dołączeniu urządzenia do magistrali USB, host wykrywa je i wykonuje szereg standardowych operacji (zwanych enumeracją), które pozwalają na:
- rozpoznanie rodzaju urządzenia i określenie jego możliwości komunikacyjnych i funkcjonalnych,
- dynamiczne przydzielenie mu indywidualnego adresu na magistrali,
- wybór i załadowanie właściwego sterownika systemowego,
- włączenie wybranej konfiguracji urządzenia.

Pobranie informacji o urządzeniu polega na przesłaniu przez nie bloków danych opisowych, nazywanych zgodnie z przeznaczeniem deskryptorami (*description* – ang. opis). Mają one stałą, ściśle określoną strukturę. Najpierw jest przesyłany ogólny deskryptor urządzenia (*device descriptor*) (**tab. 3**).

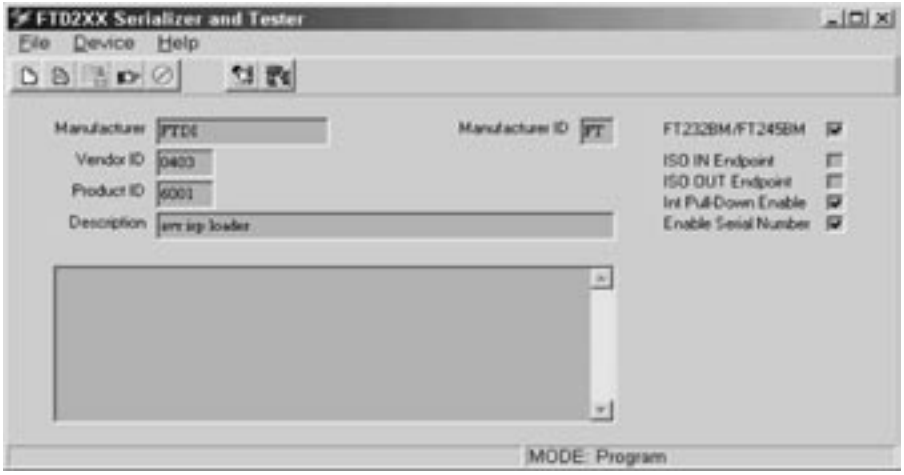
Potem kolejno pobierane są deskryptory konfiguracji, interfejsów, endpointów i opisów tekstowych (**tab. 4**). Ilość tych informacji zależy od stopnia skomplikowania urządzenia. Nazwę „konfiguracja" wprowadzono dla opisu bardziej skomplikowanych przyrządów, które mogą być wykorzystywane i przełączane na kilka różnych sposobów – z różnymi funkcjami, innym poborem prądu czy charakterystykami komunikacji. Prostsze urządzenia – w tym również kostka FT8Uxx – mają tylko jedną dostępną konfigurację. Interfejs z kolei jest wydzielonym zespołem wejściowych i wyjściowych strumieni komunikacyjnych USB (endpointów) przeznaczonych dla wykonywania określonych zadań.

Większość pozycji w deskryptorach FT8Uxx jest stała, natomiast pogrubieniem zaznaczono wartości, które mogą być zmienione przez użytkownika według indywidualnych potrzeb (nie przedstawiono zawartości deskryptorów interfejsu oraz endpointów, gdyż wynika ona z konstrukcji kostki i użytkownik nie ma na nią wpływu). Nowe wartości są pobierane z zewnętrznej pamięci EEPROM, do której wpisujemy je za pomocą programu narzędziowego *Ftd2xxst.exe*. Bez dołączonej pamięci użyte są wartości domyślne (VID = 0403, PID = 6001, producent = FTDI, produkt = USB <-> Serial, zasilanie z magistrali, pobór prądu = 90 mA).

Tab. 4. Budowa deskryptora konfiguracji

Nazwa pola deskryptora konfiguracji	FT8Uxx
Całkowita długość (w bajtach) deskryptora = 9	0x09
Kod typu deskryptora = 2	0x02
Całkowita długość deskryptorów opisu konfiguracji	0x20,0x00
Liczba interfejsów dostępnych w konfiguracji	0x01
Numer identyfikacyjny konfiguracji	0x01
Indeks nazwy konfiguracji	0x0
Atrybuty konfiguracji	**0x80**
Maksymalny prąd pobierany przez konfigurację (w jednostkach 2 mA)	**0x2d (45)**

*Uwaga:
Bajt atrybutów ma następujące znaczenie:
- bity 7 i 6 określają sposób zasilania urządzenia w danej konfiguracji (10 – z magistrali, 01 – zasilanie własne),
- ustawiony bit 5 wskazuje na możliwość zdalnego wybudzania komputera ze stanu uśpienia.



Rys. 1. Okno ustawień podstawowych *Ftd2xxst.exe*

2. Konfigurowanie układu za pomocą Ftd2xxst.exe

Okna dialogowe *Ftd2xxst.exe* pokazane są na **rys. 1** i **2** (szczegółowy opis używania programu znajdziemy w załączonym manualu). Udostępnione są następujące podstawowe opcje konfiguracyjne:
- wybór pomiędzy wersją układu AM/BM;
- określenie nazwy i symbolu producenta;
- ustawienie numeru identyfikacyjnego producenta (VID);
- ustawienie numeru identyfikacyjnego produktu (PID);
- określenie nazwy urządzenia.

Gdy używamy wersji BM, wyświetlone zostają *checkboxy* dodatkowych opcji:
- przełączenie wejścia/wyjścia w tryb *isochronous* (wymaga oddzielnych sterowników),
- włączenie lekkiego podciągania do masy linii I/O w stanie *sleep* (zalecane zwłaszcza przy zasilaniu z magistrali dodatkowych zewnętrznych urządzeń – po wyłączeniu zasilania różne resztkowe napięcia na liniach zostają wyzerowane),
- włączenie zapisu numeru seryjnego do EEPROM-u (wersja AM przy braku EEPROM-u przesyła domyślny numer, co prowadzi do konfliktu przy podłączeniu dwóch lub więcej kostek bez EEPROM-u; BM domyślnie nie wysyła numeru, co eliminuje ten efekt – indeks w deskryptorze jest wyzerowany i przestawiany na 0x03 dopiero po zaznaczeniu opcji *Enable Serial Number*).

Oprócz tego w oddzielnym okienku ustawiamy opcje zaawansowane:
- *Plug & Play* nie dotyczy samej kostki FT8Uxx, ale zbudowanego na jej bazie urządzenia – poprzez zaznaczenie opcji informujemy Windows, że nasz przyrząd obsługuje systemowe wywołania *PnP* i powinien być nimi objęty,
- numer seryjny, który możemy ustawić ręcznie albo automatycznie (tryb automatyczny zabezpiecza przed przypadkowym powtórzeniem),
- rodzaj zasilania (z magistrali lub samodzielne) – to ustawienie zmienia wpis do deskryptora, co pozwala na jednorazowe ustalenie trybu zasilania podczas enumeracji. Układ FT8U232 jest dodatkowo wyposażony w wejście PWRCTL, które umożliwia dynamiczną zmianę trybu zasilania bez zerowania kostki – sterownik odczytuje odpowiednią informację cyklicznymi poleceniami GET_STATUS),
- zewnętrzne wybudzanie hosta z trybu uśpienia (poprzez wejście RI FT8U232 albo wejście SI/WU FT8U245),
- maksymalny przewidywany pobór prądu (wpisujemy bezpośrednio w mA – program samoczynnie przelicza na jednostki używane w deskryptorze).

*3. Edycja plików *.inf*

Jeśli wpiszemy do EEPROM-u nowe, zmienione identyfikatory VID lub PID, musimy nanieść odpowiednie poprawki również w plikach *.inf. Windows wykorzystuje identyfikatory pobrane z urządzenia podczas enumeracji do wyszukania i załadowania odpowiedniego sterownika. Przy braku ich zgodności system nie będzie w stanie nawiązać z urządzeniem komunikacji.

Tryb pracy jako VCP wykorzystuje pliki *ftdibus.inf* oraz *ftdiport.inf*. Odnajdujemy w nich wszystkie linie z odwołaniami do identyfikatorów (VID_0403&PID_6001) i wykonujemy odpowiednie korekty. Należy też zmienić zapis w plikach deinstalatorów (*ftdiunin.ini* lub *ftdiun2k.ini*). Dodatkowo możemy zmienić opisy wykorzystywane przez Windows przy ładowaniu sterownika. W pliku *ftdibus.inf* linie:

```
HKLM,%WINUN%\FTDICOMM , „DisplayName”,,”FTDI USB-to-Serial Converter Drivers”
HKLM,%WINUN%\FTDICOMM , „DisplayName”,,”FTDI USB Serial Converter Drivers”
```

decydują o opisie w okienku *Dodaj/Usuń programy*, zaś linia:

```
USB\VID_0403&PID_6001.DeviceDesc=”USB High Speed Serial Converter”
```

o opisie urządzenia w kluczu *Uniwersalne Kontrolery Magistrali Szeregowej* w Managerze Urządzeń.

W pliku *ftdiport.inf* linia:

```
VID_0403&PID_6001.DeviceDesc=”USB Serial Port”
```

określa opis w kluczu *Porty Com i Lpt* Managera Urządzeń.

Przy użyciu sterownika Direct Driver zmiany wykonujemy tylko w plikach *ftd2xx.inf* oraz *ftd2xxun.ini*.

Podobnie jak poprzednio możemy też skorygować opisy używane przez Windows:

```
HKLM,%WINUN%\FTD2XX , „DisplayName-”,,”FTDI FTD2XX USB Drivers”
USB\VID_0403&PID_6001.DeviceDesc=”FTDI FT8U2XX Device” .
```

W praktyce amatorskiej zmiana VID/PID będzie następować raczej rzadko – możemy z powodzeniem używać domyślnych ustawień FTDI. Jedynym typowym przypadkiem wymagającym korekty może być konieczność użycia jednocześnie urządzeń wykorzystujących zarówno tryb pracy VCP (np. uniwersalna przejściówka USB <-> RS 232), jak i tryb oparty o firmowy interfejs API (np. różne przyrządy warsztatowe). Oba tryby korzystają z różnych sterowników (*VCP* oraz *Direct Driver*), które nie mogą być jednocześnie załadowane, jeśli opi-

Rys. 2. Okno ustawień dodatkowych *Ftd2xxst.exe*

sane są tym samym zestawem identyfikatorów (na stronie FTDI znajdziemy także opis problemu z użyciem *Direct Driver* pod Windows XP + Service Pack 1 – wymagana jest zmiana PID, ale do tego FTDI oferuje oddzielny niewielki program konfigurujący).

W zastosowaniach komercyjnych wymagane jest stosowanie własnego identyfikatora VID – przydział identyfikatorów dla producentów jest ujęty w znormalizowane formy organizacyjne i podlega odpowiednim opłatom.

Konfiguracja parametrów pracy układu z poziomu współpracującej aplikacji

1. Interfejsy API dla programowania komunikacji z użyciem FT8Uxx.

Jak wspomniano wyżej, FT8Uxx mogą być zainstalowane jako dodatkowe porty COM, używane w tradycyjny sposób przez istniejące oprogramowania. Tryb ten nie pozwala jednak na korzystanie z wielu dodatkowych właściwości oferowanych przez kostki (chociaż kolejne wersje sterowników sukcesywnie rozszerzają zakres ustawień). Jeśli więc samodzielnie opracowujemy aplikacje, w których wystąpi komunikacja z FT8Uxx, warto stosować sterownik *Direct Driver* i współpracującą z nim bibliotekę *Ftd2xx.dll* zawierającą zestaw potrzebnych funkcji.

Funkcje biblioteczne *Ftd2xx.dll* podzielone zostały na kilka grup:
- *Classic Interface* – najstarszy, złożony z prostych w obsłudze funkcji interfejs komunikacyjny,
- *FT-Win32 API* – najnowszy zbiór funkcji dopasowanych nazwami, działaniem i sposobem wywołania do systemowych funkcji API Windows – pozwala to w łatwy sposób zmodyfikować kody źródłowe istniejących aplikacji (te dwa interfejsy powinny być stosowane rozdzielnie),
- obsługa dostępu do pamięci EEPROM – pozwala na wykorzystanie dla własnych potrzeb wolnej (nieużywanej przez wpisy konfiguracyjne) przestrzeni dołączonej kostki EEPROM,
- obsługa rozszerzonych właściwości wersji BM układów.

Dokładny opis interfejsów programowych znajdziemy w obszernym manualu dostępnym na stronie FTDI.

2. Programowe ustawianie parametrów komunikacji USB

Za pomocą interfejsów *Ftd2xx.dll* możemy dopasować parametry połączenia USB do niektórych, szczególnych wymogów transmisyjnych.

a) Przy wymianie danych z urządzeniem poprzez niewielkie, często powtarzane pakiety (z takimi rozwiązaniami mamy bardzo często do czynienia w warsztacie elektronicznym przy obsłudze różnego rodzaju przyrządów opartych na mikrokontrolerach) może dojść do spowolnienia transmisji poprzez timeout odczytu niepełnego bufora *bulk*. Bufor ma rozmiar 64B i jest odsyłany do hosta po całkowitym wypełnieniu lub po upłynięciu timeoutu, gdy nie jest załadowany do końca (co może często zachodzić w omawianym przypadku komunikacji). Domyślna (i niezmienna dla wersji AM) wartość timeoutu wynosi 16 ms. Wersja BM pozwala na zmianę w zakresie 1-255 ms. Zmianę można wykonać tylko programowo, wykorzystując funkcję:
FT_SetLatencyTimer (FT_HANDLE *ftHandle*, UCHAR *ucTimer*)
(ftHandle jest systemowym uchwytem urządzenia określonym przy jego prawidłowym otwarciu).

b) Rozmiar pakietów, w jakie sterownik organizuje przesyłane dane powinien być dopasowany do szybkości przesyłu. Duże pakiety przy małych szybkościach będą powodować nierównomierny przepływ danych (sterownik oczekuje na kompletację pakietu, która przy wolnej transmisji zabiera relatywnie dużo czasu). Rozmiar możemy zmieniać w zakresie 64B – 16 kB (jako wielokrotność 64B) za pomocą funkcji:
FT_SetUSBParameters (FT_HANDLE *ftHandle*, DWORD *dwInTransferSize*, DWORD *dwOutTransferSize*)
(obecna wersja ustawia tylko pakiet odbiorczy, argument dla nadawczego jest ignorowany).

3. Bit Bang – dodatkowy tryb pracy układów FT8Uxx BM

Bit Bang to zupełnie oddzielny tryb pracy kostki, całkowicie niezależny od standardowych funkcji transmisyjnych. Każda linia może być ustawiona jako wejściowa lub wyjściowa. Dane są wysyłane i odbierane bajtami zgodnie z przyjętą szybkością transmisji, przy czym:
- wysłanie bajtu oznacza ustawienie linii wyjściowych zgodnie z wartościami odpowiadających bitów,
- odbiór bajtu oznacza odczyt rzeczywistego stanu linii wejściowych (dla linii wyjściowych odczytywane jest ich ostatnie ustawienie).

Przykładowo nadanie n bajtów z szybkością 1000 baud spowoduje co 1 ms ustawienie na liniach wyjściowych po kolei n wartości. Taki sam odbiór oznacza pobranie przez hosta n wartości linii wejściowych skanowanych co 1 ms.

Przypisanie linii do bitów danej jest podane w **tab. 5**.

Podczas przerw w nadawaniu linie wyjściowe pozostają w ostatnio ustawionym stanie. Tryb jest obsługiwany za pomocą funkcji:
FT_SetBitMode (FT_HANDLE *ftHandle*, UCHAR *ucMask*, UCHAR *ucEnable*)
gdzie:
- ftHandle – uchwyt uzyskany przy otwarciu urządzenia,
- ucMask – przypisanie linii: bit = 0 ustawia odpowiadającą linię jako wejście, bit = 1 – jako wyjście,
- ucEnable – włączenie (bit B0 = 1) lub wyłączenie (bit B0 = 0) trybu *Bit Bang*

oraz dodatkowo przez funkcję natychmiastowego (niezależnego od normalnego strumienia odbieranych danych) odczytu aktualnego stanu wejść:
FT_GetBitMode (FT_HANDLE *ftHandle*, PUCHAR pucData)
gdzie pucData jest wskaźnikiem na zmienną typu *unsigned char*, która przyjmuje odebraną wartość.

Bit Bang nadaje się świetnie do sprzętowo-programowej realizacji rozmaitych magistral komunikacyjnych (musimy jednak cały czas mieć na uwadze ograniczenia wynikające z pakietowej struktury przesyłu USB).

Wszelkie szczegółowe materiały i opisy znajdziemy w notach aplikacyjnych na stronie producenta www.ftdichip.com. Stamtąd też warto pobierać najnowsze uaktualnienia sterowników i bibliotek, których możliwości są wciąż rozszerzane.

Jerzy Szczesiul
jerzy.szczesiul@ep.com.pl

Tab. 5. Przypisanie linii do bitów danej

FT8U245 BM	FT8U232 BM	Bit Bang data bit
Data 0	TxD	B0
Data 1	RxD	B1
Data 2	RTS	B2
Data 3	CTS	B3
Data 4	DTR	B4
Data 5	DSR	B5
Data 6	DCD	B6
Data 7	RI	B7