USB - Universal Serial Bus

Opis standardu oraz przykładowa implementacja
w układzie Cypress PsoC 5 CY8C58LP

KATEDRA
SYSTEMÓW
MIKROELEKTRONICZNYCH

inż. Dominik Marszk

Standard komunikacji przewodowej opracowywany przez konsorcjum USB (założone przez Compaq, DEC, IBM, Intel, Microsoft, NEC i Nortel).

Definiuje sprzętową oraz logiczną budowę interfejsu pomiędzy hostem a urządzeniami.

Standard USB jest wciąż rozwijany.

Zakłada łatwe i bezproblemowe łączenie ze sobą urządzeń (Plug & Play) implementujących standard.

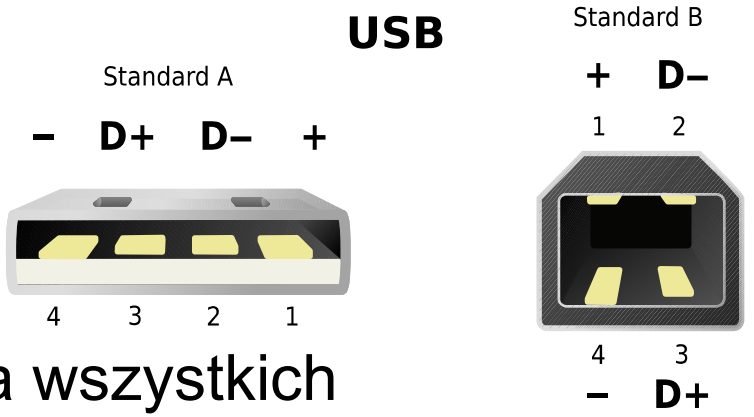
Wydane dotąd wersje

- USB 1.x (1996)
  - Low Speed (LS) – 1.5 Mbps (tolerancja zegara ± 1.5%)
  - Full Speed (FS) – 12 Mbps (tolerancja zegara ± 0.25%)
- USB 2.0 (2000)
  - High Speed (HS) – 480 Mbps (tolerancja zegara ± 0.05%)
- USB 3.0 (2008)
  - SuperSpeed (SS) – 5 Gbps
- USB 3.1 (2013)
  - SuperSpeed+ (SS+) - 10 Gbps

Szybkość transmisji to nie szybkość Host-Device a sumaryczna przepustowość kontrolera, współdzielona między wszystkimi urządzeniami!

Fizyczna budowa interfejsu

- Dla USB 1.x oraz 2.0 urządzenia są standardowo połączone 4 liniami: V_{BUS} (+5V), D-, D+ oraz GND.
- USB 3.0 wprowadza dodatkową masę GND_DRAIN pomagającą ekranować kabel, oraz 2 dodatkowe pary przewodów – SSTx oraz SSRx (full-duplex).
- Komunikacja odbywa się poprzez różnicową parę skręconych ze sobą przewodów. Impedancja charakterystyczna pary $R_L = 90\Omega \pm 15\%$.
- Low/Full Speed nie używa terminacji. High Speed używa terminacji 45Ω do GND lub 90Ω różnicowo.
- Host ma na sztywno 15kΩ pull-down na wszystkich liniach danych (SE0 dłużej niż 10ms - reset)



Identyfikacja trybu i kodowanie

- Różnicowe „1” na drucie to D+ na poziomie 2.8V-3.6V i D- na 0V-0.3V.
- Różnicowe „0” na drucie to D- na poziomie 2.8V-3.6V i D+ na 0V-0.3V.
- HS używa niższych poziomów – odpowiednio 0.36V – 0.44V i -0.01V – 0.01V.
- Urządzenia Low Speed mają pull-up 1.5kΩ do 3.3V na D-.
  Full Speed używa pull-upa na D+.
- Wszystkie urządzenia High Speed i SuperSpeed najpierw działają w trybie Full Speed i po resecie próbują wynegocjować tryb HS poprzez wysłanie symbolu K, host wspierający żądany tryb odpowiada co najmniej trzykrotnym powtórzeniem stanów JK (chirping). Pull-up jest odłączany w trybie HS. Tryb SS to rozszerzenie trybu HS o 2 dodatkowe pary różnicowe (negocjowane oddzielnie i opcjonalnie).
- Stan „idle” na drucie to J. Stan przeciwny to K. Z pull-upów wynika, że dla LS J = „0” na drucie a dla FS J = „1” na drucie.
- Kodowanie NRZI (logiczne „0” powoduje przejście stanu, „1” podtrzymanie go), bit stuffing („0” po każdym ciągu sześciu „1”).
- Transmisja wyłącznie w formie pakietów.
- Nie tylko różnicowe sygnały – SE0 jest używane do resetu oraz do oznaczenia końca pakietu.

Transmisja

- Wszystkie transfery inicjowane są przez hosta.
- Pakiety grupowane są na transakcje, składające się z:
  - Pakietu typu Token,
  - Opcjonalnego pakietu z danymi,
  - Pakietu statusu (handshake).
- Kierunek ustawienia bitów to LSB-first.
- Każdy pakiet składa się z następujących pól:

  - Sync - 8-bit dla LS i FS, 32-bit dla HS. Ciąg '0', ostatni bit '1'.
  - ID pakietu (PID) – 4-bit, identyfikuje typ pakietu, wysyłany jest dwukrotnie – drugi raz zanegowany.
  - Payload (zależnie od typu pakietu)
  - CRC – 16-bit dla danych, 5-bit dla tokenów, nieużywany dla innych.
  - End of Packet (EOP) – SE0 na 2 szerokości bitu + jeden stan J.
  -

Typy pakietów

- Token:
  - Typ IN/OUT/SETUP.
  - W miejscu payloadu mają:
    - ADDR – 7-bit, adres urządzenia. Dozwolone adresy to 1-127.
    - ENDP – 4-bit, numer endpoint'u.
- Data:
  - Typ DATA0/DATA1 (dla HS jeszcze DATA2 i MDATA).
  - W miejscu payloadu jest n-bajtów danych. Dla LS max 8. Dla FS max 1023. Dla HS max 1024.
- Handshake:
  - Typ ACK, NAK, STALL, NYET (tylko HS).
- Start of Frame – specjalny token, payload to Frame Number (zwiększany co 1ms). Wysyłany przez hosta co 1ms ± 500ns dla FS i 125µs ± 0.0625 µs dla HS.

Group	PID Value	Packet Identifier
Token	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	SETUP Token
Data	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake	0010	ACK Handshake
	1010	NAK Handshake
	1110	STALL Handshake
	0110	NYET (No Response Yet)
Special	1100	PREamble
	1100	ERR
	1000	Split
	0100	Ping

Typy trasmisji

- Transakcje:
  - OUT – host wysyła dane, urządzenie potwierdza.
  - IN – host żąda danych, urządzenie je wysyła a host potwierdza.
  - SETUP – host wysyła dane analogicznie jak w OUT, ale payload pakietu DATA ma zawsze 8 bajtów.
- Rodzaje transferu danych (jak i endpointów):
  - Bulk – IN/OUT, sprawdzanie błędów, rozmiar pakietów 8/16/32/64 bajtów (FS) lub 512 (HS). Używane np. w nośnikach danych.
  - Interrupt – zwykle IN, periodyczne transmisje danych wywoływane przez hosta. Rozmiar pakietów 1-8 (LS), 1-64 (FS) lub 1-1024 (HS) bajtów. Używane np. w urządzeniach wskazujących.
  - Isochronous – gwarantowana przepływność, ale bez pakietów handshake. Bez retransmisji. Dopuszcza się błędy. Używane np. w urządzeniach strumieniujących.
  - Control – zawsze używany jest EP0, zaczyna się od etapu SETUP, potem opcjonalnie następuje etap DATA, kończy się etapem STATUS (transakcja DATA z pakietem DATA1 zerowej długości).

Enumeracja

- Po wykryciu urządzenia host resetuje je, negocjuje prędkość a następnie enumeruje urządzenie, inicjując transfer typu control żądając początku jego deskryptora. Urządzenie odpowiada pierwszymi 8 bajtami deskryptora, zawierają one:
  - Rozmiar i typ deskryptora,
  - Specyfikację USB wspieraną przez urządzenie,
  - Klasę urządzenia,
  - Podklasę,
  - Kod protokołu,
  - Maksymalny rozmiar pakietu dla EP0.
- Następnie dla USB 1.x następuje reset urządzenia (USB 2.0 go nie wymaga), przydzielenie mu adresu i odczytanie wszystkich deskryptorów oraz konfiguracji urządzenia.

Deskryptory i typowe klasy urządzeń

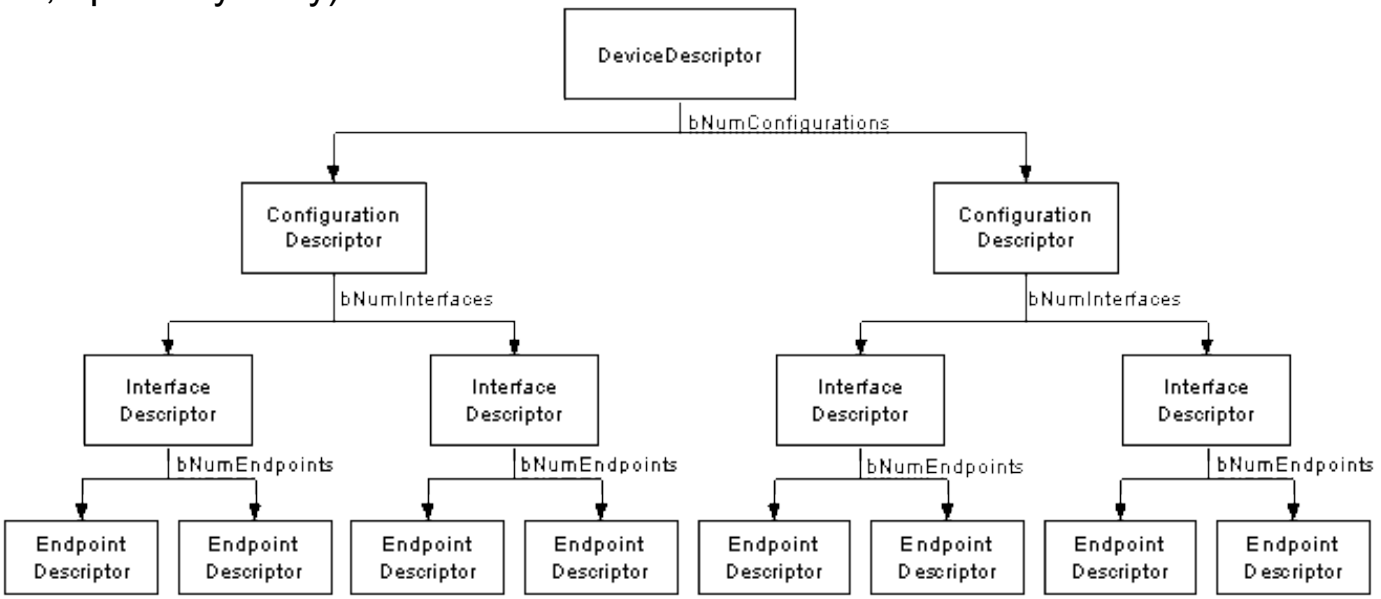
Specyfikacja USB definiuje wiele standardowych klas urządzeń:

- Audio, CDC-Control, HID, Physical, Image, Printer, Mass Storage, Hub, CDC-Data, Smart Card, Content Security, Video, Personal Healthcare, Audio/Video Devices, Billboard Device, Diagnostic Device, Wireless Controller.

Dodatkowo zdefiniowane są specjalne rodzaje klas:

- From Interface Descriptor, Misc, Application Specific, Vendor Specific

Systemy takie jak Windows/Linux mają wbudowane sterowniki dla większości popularnych klas. Umożliwia to podłączenie urządzenia i używanie go bez potrzeby instalacji specjalnych sterowników (pendrive, mysz, klawiatura, aparat cyfrowy)



Przykładowa topologia mikrofonu USB

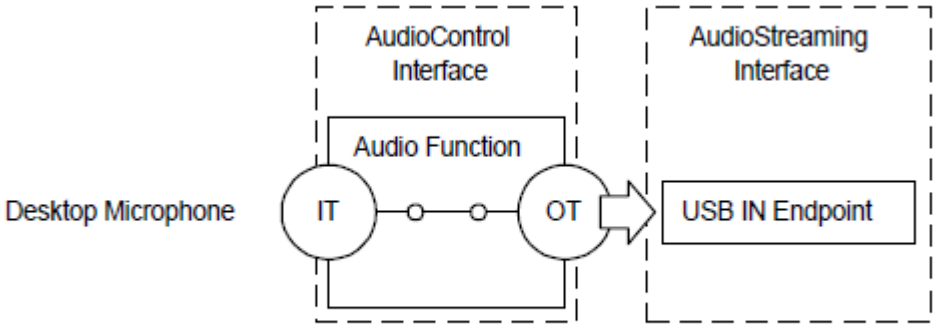


Figure B-1: USB Microphone Topology

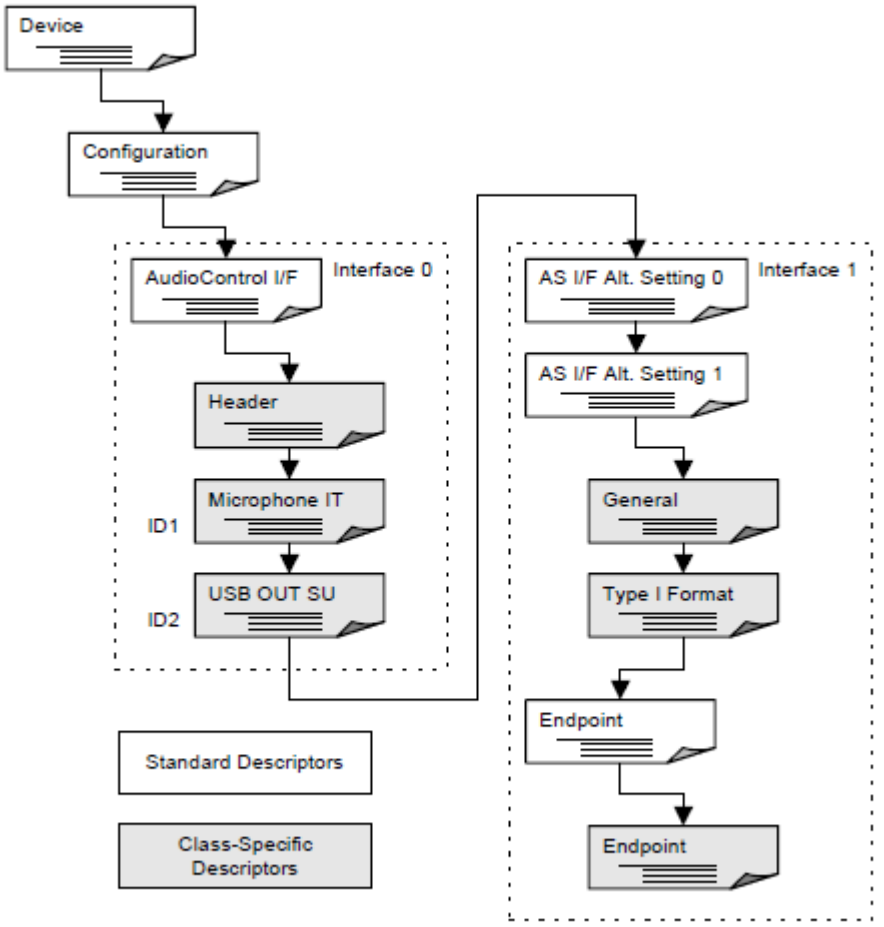


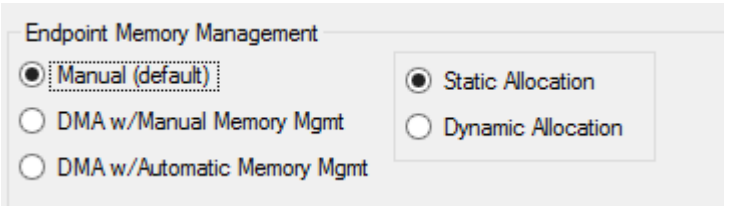
Figure B-2: USB Microphone Descriptor Hierarchy

Sterownik USBFS

Aby USB działało należy poprawnie skonfigurować zegary.

Type	Name	Domain	Desired Frequency	Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on Reset	
System	Digital Signal	DIGITAL	? MHz	? MHz	±0	–	0	☐	
System	XTAL 32kHz	DIGITAL	32.768 kHz	? MHz	±0	–	0	☐	
System	XTAL	DIGITAL	24.000 MHz	? MHz	±0	–	0	☐	
System	ILO	DIGITAL	? MHz	100.000 kHz	-55, +100	–	0	☑	
System	BUS_CLK (CPU)	DIGITAL	? MHz	24.000 MHz	±0.25	–	1	☑	MASTER_CLK
System	MASTER_CLK	DIGITAL	? MHz	24.000 MHz	±0.25	–	1	☑	PLL_OUT
System	PLL_OUT	DIGITAL	24.000 MHz	24.000 MHz	±0.25	–	0	☑	IMO
System	IMO	DIGITAL	24.000 MHz	24.000 MHz	±0.25	–	0	☑	
System	USB_CLK	DIGITAL	48.000 MHz	48.000 MHz	±0.25	–	1	☑	IMOx2
Local	ADC_theACLK	DIGITAL	11.368 MHz	12.000 MHz	±0.25	–	2	☑	Auto: MASTER_CLK
Local	USB_Clock_vbus	DIGITAL	? MHz	24.000 MHz	±0.25	–	0	☑	BUS_CLK

USBFS posiada 3 tryby obsługi danych na endpointach.



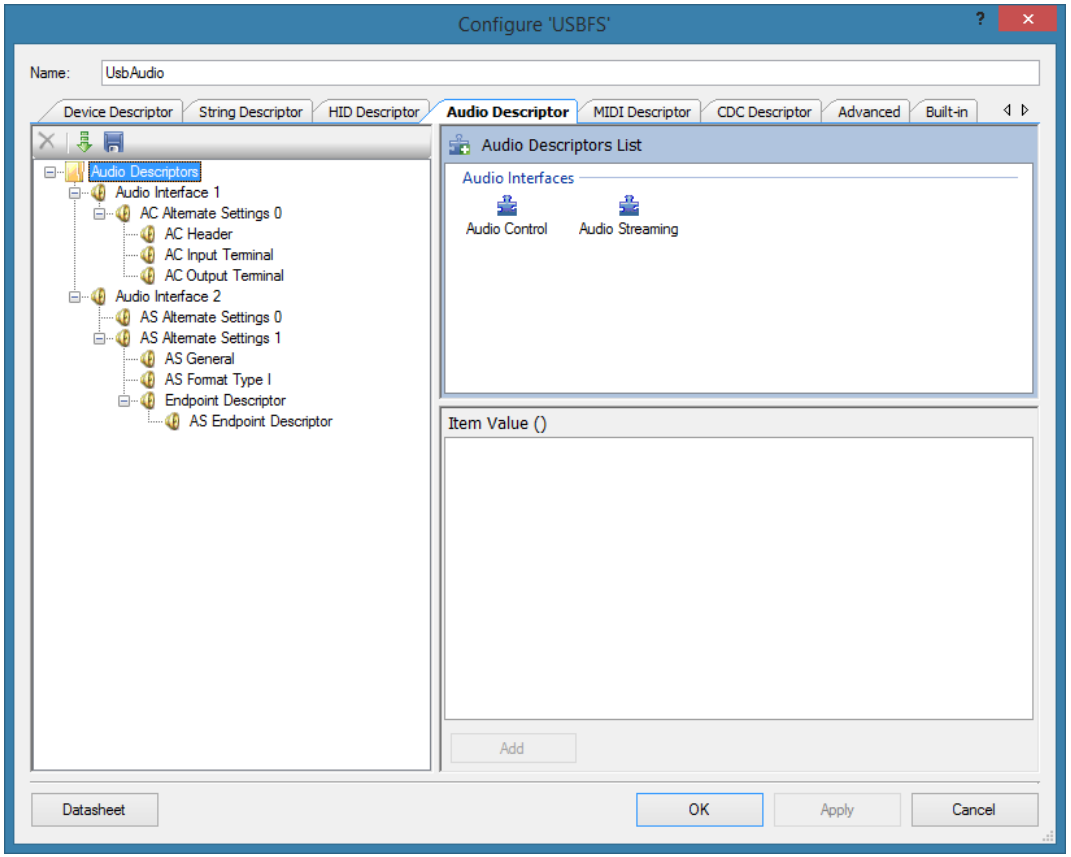
Przykładowa implementacja urządzenia klasy Audio

PSoC Creator posiada wygodny interfejs edycji deskryptorów.

Implementację należy przeprowadzić zgodnie ze specyfikacją urządzeń USB Audio aby Windows był w stanie kontrolować je przy pomocy wbudowanego sterownika.

Przydatne dokumenty:

- USB Audio Devices
- USB Audio Terminal Types
- USB Audio Data Formats



Back-end

Inicjowany funkcją USB_Start(DeviceId, mode);

Stan enumeracji zwraca funkcja USB_GetConfiguration();

Główna pętla programu powinna monitorować stan enumeracji oraz aktualnie aktywną funkcję urządzenia. Transfery DMA mogą być inicjowane z funkcji main lub z przerwania.

```
for(;;)
{
    if(USB_IsConfigurationChanged() != 0u)
    {
        /* Check the active Alternative setting when configuration changed */
        if((USB_GetConfiguration() != 0u) && (USB_GetInterfaceSetting(AUDIO_INTERFACE) != 0u))
        {   /* Alternate 1 with AudioStreaming interface has been set */
            Timer_Start();
            audio_on = 1;
            lcd_update();
        }
        else    /* Zero Alternate setting for mute */
        {
            audio_on = 0;
            lcd_update();
            Timer_Stop();
        }
    }
    if (USB_GetEPState(USB_EP2) == USB_IN_BUFFER_EMPTY)
    {
        if(selectedBuf == 1)
        {
            led_1_Write(1);
            USB_LoadInEP(USB_EP2, (uint8*)dmaBuf1, SMP_PER_PACKET * sizeof(dmaBuf2[0]));
            selectedBuf = 2;
            led_1_Write(0);
            index = get_samples(dmaBuf2, SMP_PER_PACKET, index, freq, SMP_FREQ);
        }
        else
        {
            led_2_Write(1);
            USB_LoadInEP(USB_EP2, (uint8*)dmaBuf2, SMP_PER_PACKET * sizeof(dmaBuf2[0]));
            selectedBuf = 1;
            led_2_Write(0);
            index = get_samples(dmaBuf1, SMP_PER_PACKET, index, freq, SMP_FREQ);
        }
    }
    if(newFreq != freq)
    {
        freq = newFreq;
        lcd_update();
    }
```

Więcej informacji

Powyższa prezentacja zawiera tylko ogólne informacje. Aby zaimplementować konkretne urządzenie należy bliżej zapoznać się ze specyfikacją standardu:

- http://www.usb.org/
- http://www.usbmadesimple.co.uk/index.html
- http://www.beyondlogic.org/usbnutshell/
- http://en.wikipedia.org/wiki/USB