Politechnika Łódzka
Instytut Elektroniki

Programowanie Mikrokontrolerów

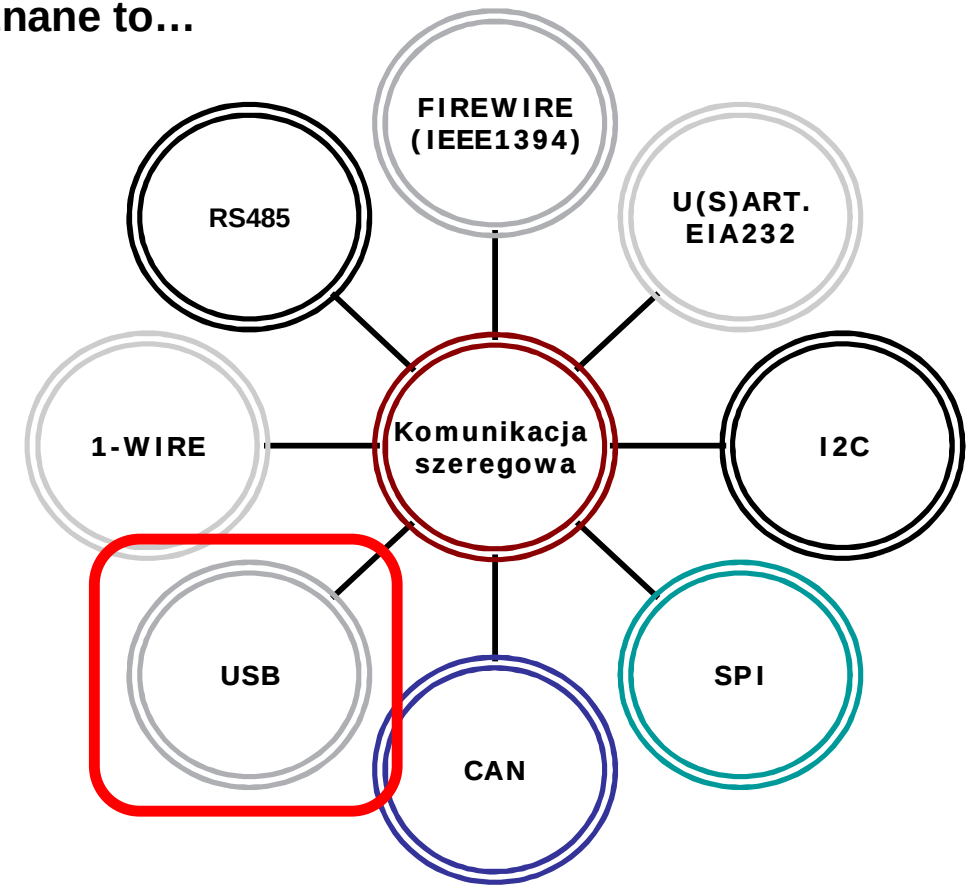
USB – *Universal Serial Bus*

mgr inż. Paweł Poryzała
Zakład Elektroniki Medycznej

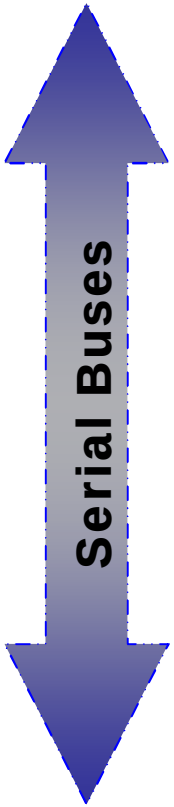
Komunikacja szeregowa

- Jakie znamy typy komunikacji szeregowej?

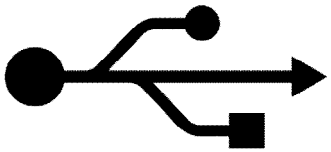
Prawdopodobnie najbardziej znane to…



USB – zagadnienia



- Asynchroniczna, szeregowa, różnicowa transmisja danych w kodzie NRZI, *half-duplex*,
- Cztery przewody (D+, D-, VBUS, GND),
- Ustandaryzowane konektory,
- Maksymalnie 127 urządzeń, automatyczna detekcja oraz korekcja błędów,
- Zarządzana centralnie przez jednego HOSTa
- Obsługa *plug'n'play* oraz *hot swapp,*
- Identyfikacja urządzeń poprzez parę PID/VID (Product ID/Vendor ID)
- Prędkości transmisji od 1,5 Mbit/s do 480 Mbit/s
- Oryginalnie zaproponowany w 1996 roku przez grupę firm: Compaq, Digital Equipment, IBM, Intel, Microsoft, and Northern Telecom



Dlaczego mówimy o USB

- Proste porównanie…

	USB	Serial	Parallel
Industry Standard	Yes	Yes	No
Bandwidth	12 Mbps	115 Kbps	115 KBps EPP/ECP - 3 MBps
Number of Devices	127 devices on a single USB bus	Limited to the number of ports available on the computer.	Limited to the number of ports available on the computer.
Bus Power	Yes, can provide up to 500 mA at 5V	No	No
Cable Length Limit	5 m / 16.4 ft	3 m / 10 ft	1.8 m / 6 ft
Plug'n'Play	Yes	No	No
Hot Swapable	Yes	No	No

http://www.totalphase.com/

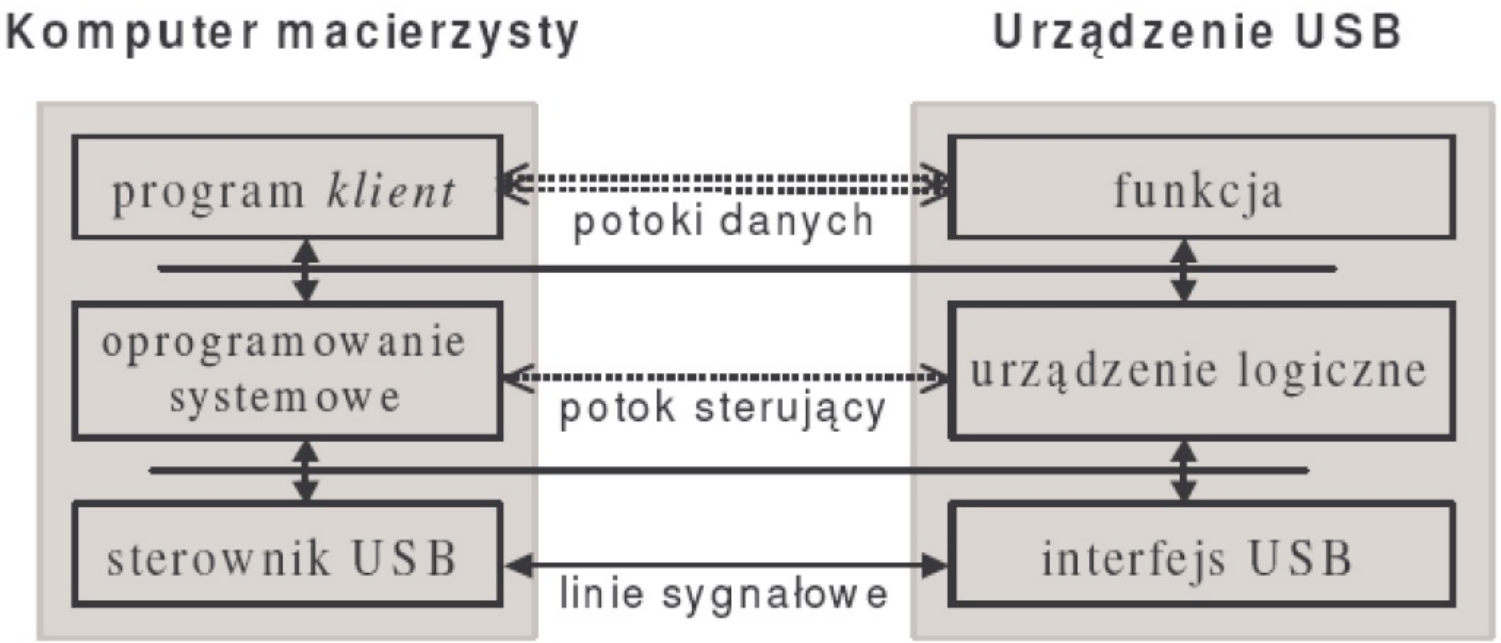
Wady USB

- Prędkość 480 Mbit/s nie robi już dziś ogromnego wrażenia – np. IEEE-1394b to 3.2 Gbit/s
- Odległość komunikacji do 5 metrów (max. 30m z zastosowaniem 5-ciu urządzeń HUB) – RS485, IEEE-1394b lub Ethernet oferują znaczenie więcej,
- Brak możliwości połączeń peer-to-peer (host-host bądź device-device), zawsze wymaga jest konfiguracja host-device
  - problem ten częściowo eliminuje USB On-The-Go – ma funckcje device oraz ograniczone funkcje host lub kabel komunikacji host-host
- Brak komunikacji rozgłoszeniowej (Broadcast) – występuje ona w IEEE-1394 bądź Ethernet
- Konieczność posiadania Vendor ID będąc producentem urządzeń (ok. 1500 USD)

USB-IF – *USB Implementers Forum*

- *USB Implementers Forum:*
  - http://www.usb.org/home
- Zajmuje się standaryzacją oraz promocją:
  - USB
  - Wireless USB
  - USB On-The-Go

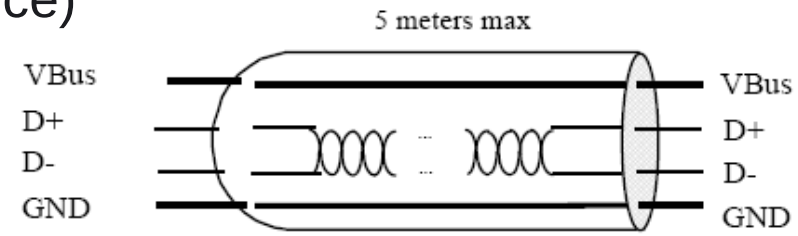
Struktura warstw USB



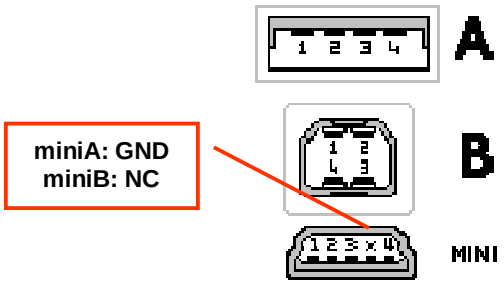
- warstwa funkcjonalna
- warstwa logiczna
- warstwa fizyczna

Standard elektryczny

- Cztery przewody:
  - D+ oraz D- - różnicowo przesyłane dane (STP), kodowanie NRZI (*Non Return to Zero Invert*) co pozwala na wygodną synchronizację urządzeń po obu stronach (Host oraz Device)
  - GND oraz +5V – zasilanie



- Każde urządzenie posiada:
  - „upstream connection to Host” w przypadku Device, gniazdo USB B lub mini-B
  - „downstream connection to Device” w przypadku Hosta, gniazdo USB A



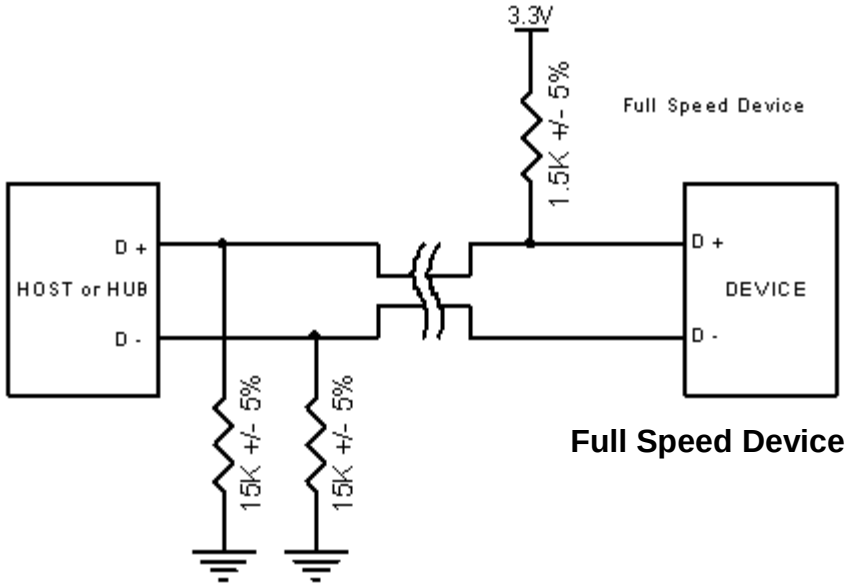
Pin	Kolor	Funkcja
1	RED	V_{BUS}(+5V)
2	WHITE	D-
3	GREEN	D+
4	BLACK	GND

Zdefiniowane prędkości transmisji

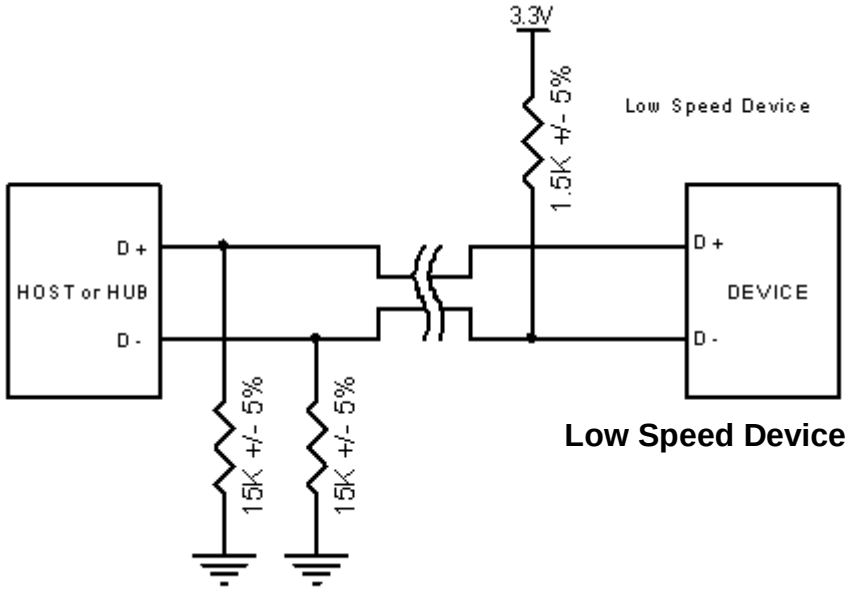
- USB Speeds:
  - Low Speed: 1,5 Mbit/s (USB 1.0)
  - Full Speed: 12 Mbit/s (USB 1.1)
  - Hi-Speed: 480 Mbit/s (USB 2.0)
  - SuperSpeed: 4,8 Gbit/s (USB 3.0)
- Całkowicie kontrolowana przez hosta (tylko jeden host na magistrali, standard USB OTG wprowadza protokoły negocjacji hosta)

Identyfikacja prędkości transmisji urządzenia

- Identyfikacja poprzez odpowiednie podciągnięcie linii D+ lub D- do +3.3V.
- Wykorzystywane do wykrywania obecności urządzenia na magistrali.



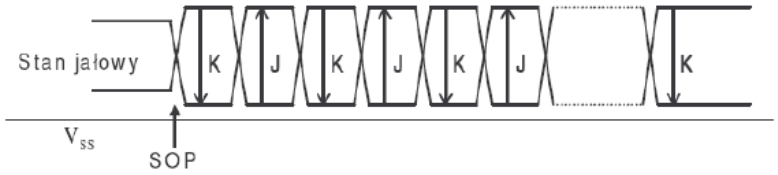
Full Speed Device



Low Speed Device

Urządzenia High Speed Device rozpoczynają jako podłączone w trybie Full Speed lecz potem zmienia częstotliwość nośną i odłącza rezystor PullUp.

Standard elektryczny



Stan jałowy magistrali USB

Transmisja:	z dużą szybkością	z pełną szybkością	z małą szybkością
Linia (D+)	GND	> 2,7V	< 0,8 V
Linia (D-)	GND	< 0,8 V	> 2,7V

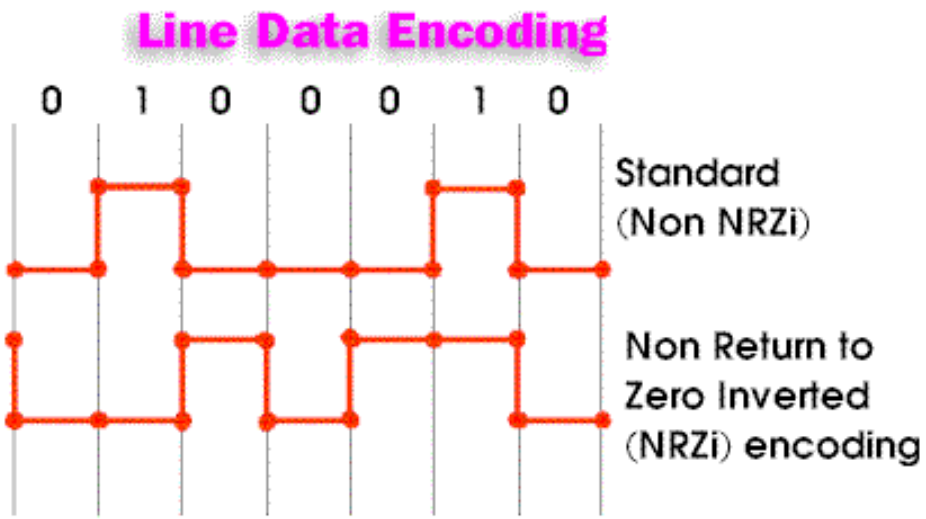
Bus State	Data State		
	Low Speed	Full Speed	Speed
Differential 0	J	K	K
Differential 1	K	J	J

- Stany K oraz J definiowane są różnie w różnych trybach pracy jako różnicowe 0 lub 1:
  - Różnicowe 0 – D_+ LOW, D_- HIGH
  - Różnicowe 1 – D_+ HIGH, D_- LOW

- Dodatkowe warunki na magistrali – jak Start-of-Packet, End-of-Packet, Disconnect, Connect, Resume, Reset etc.

Kodowanie NRZI - *Non Return to Zero Invert*

- Konwencja kodowania:
  - zmiana stanu na zboczu zegara oznacza przesyłane „logiczne 0”,
  - brak zmiany stanu na zboczu zegara oznacza przesyłane „logiczne 1”

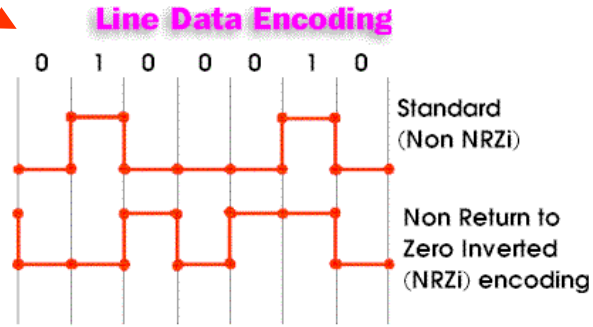


A więc odwrotnie
jak w kodowaniu NRZ.

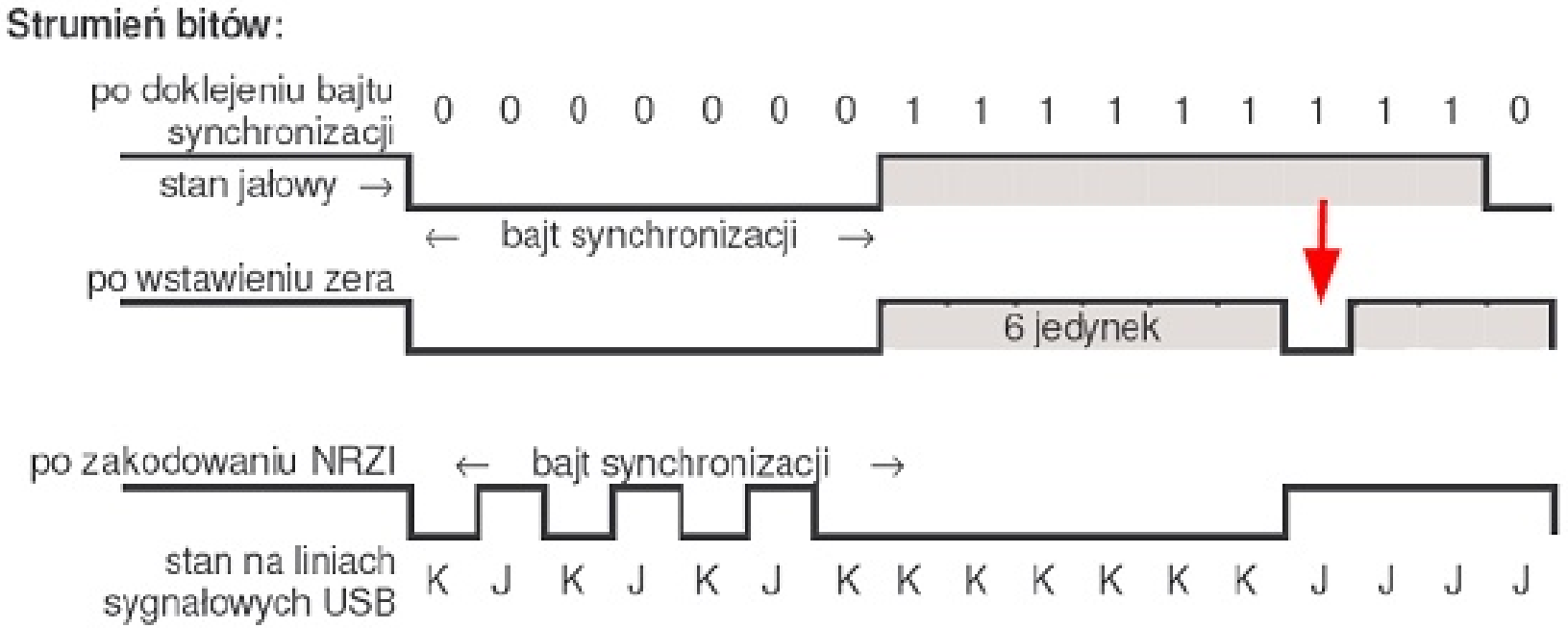
Synchronizacja danych

- Znane (nam) sposoby:
  - Dodatkowe bity START / STOP synchronizujące każdy bajt – wada???
  - Sygnał zegara przesyłany równolegle z danymi – wada???

- W USB stosuje się:
  - Bit Stuffing – odbiornik synchronizuje się na zboczach – kiedy pojawiają się zbocza w ciągu danych kodowanych NRZI???
  - SYNC Field – na początku każdego pakietu (low & full speed SYNC Field: 8 bitów: KJKJKJKK)



Synchronizacja danych



Strumień bitów po wstawieniu zera i zakodowaniu NRZI

- Bit Stuffing teoretycznie zwiększa ilość przesyłanych danych o 17%, w praktyce jest to jednak znacznie mniej (dla danych losowych ok. 0,8% - co 125 bit jest dodany).

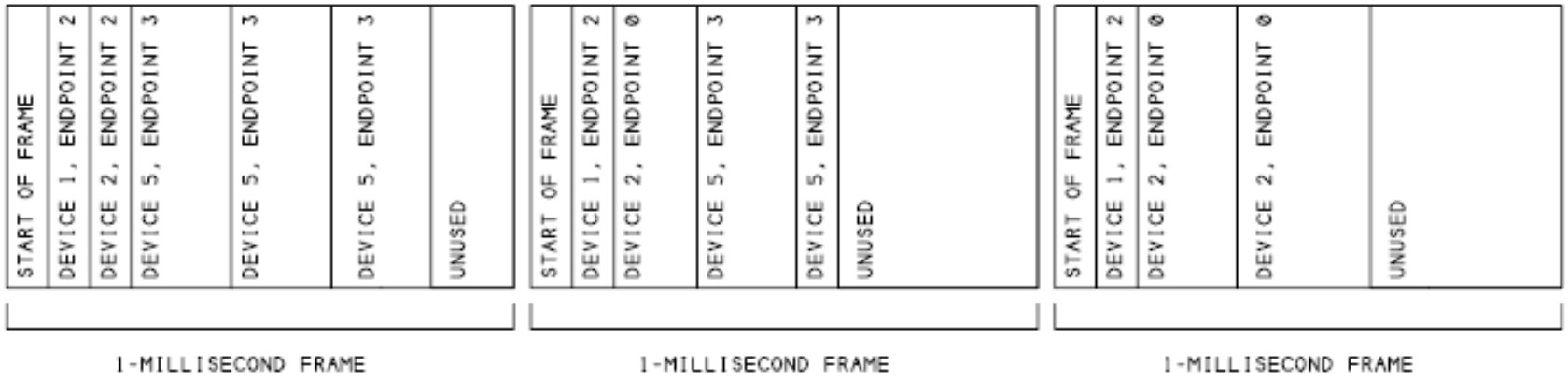
Dariusz Makowski, TM, DMCS

Zasilanie urządzeń z magistrali USB

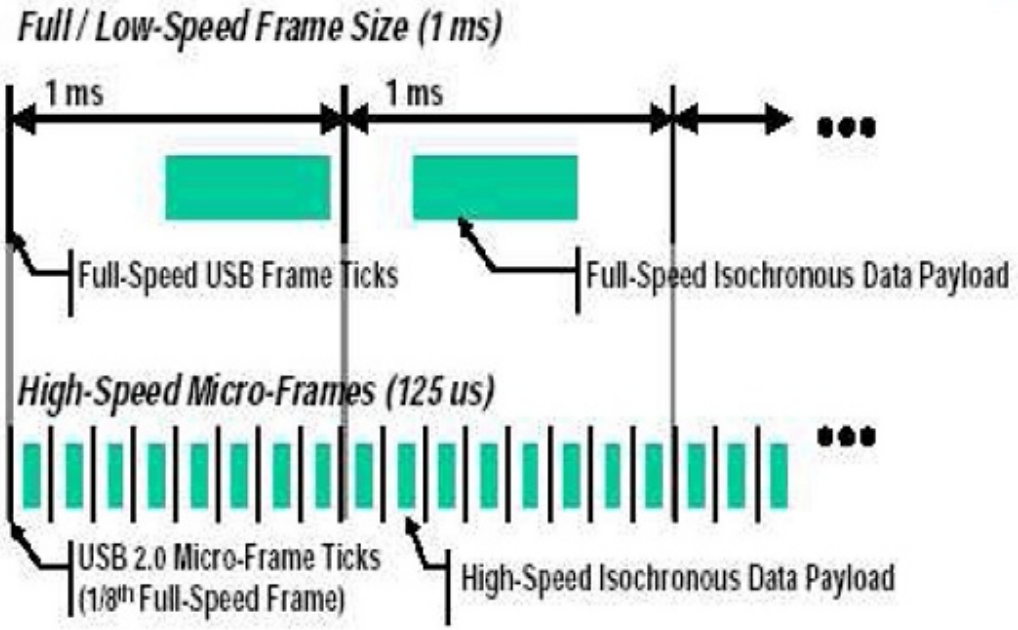
- Zasilanie USB
  - Urządzenie określa maksymalny prąd jaki pobierać może z szyny zasilającej USB (z dokładnością 2mA) podczas konfiguracji. Wartość ta nie może zostać przekroczona, nawet w przypadku zaniku zasilania zewnętrznego urządzenia.
- Klasy urządzeń USB:
  - Low-power bus powered device (max. 100mA, 4,4 .. 5,25V)
  - High-power bus powered device (max. 100mA w trakcie konfiguracji, po zakończeniu max. 500mA, 4,75 .. 5,25V)
  - Self-powered device (max. 100mA, reszta z zasilania zewnętrznego)
- Pozostałe ograniczenia (np. *inrush current*, any decoupling capacitors < 10uF, suspend current < 500uA (on 1 unit load = 100mA))

Ramki USB

- Host centralnie zarządza wymianą danych dzieląc je na:
  - ramki (po 1ms) w przypadku urządzeń low- i full-speed,
  - mikro-ramki (po 125us) w przypadku urządzeń high-speed.

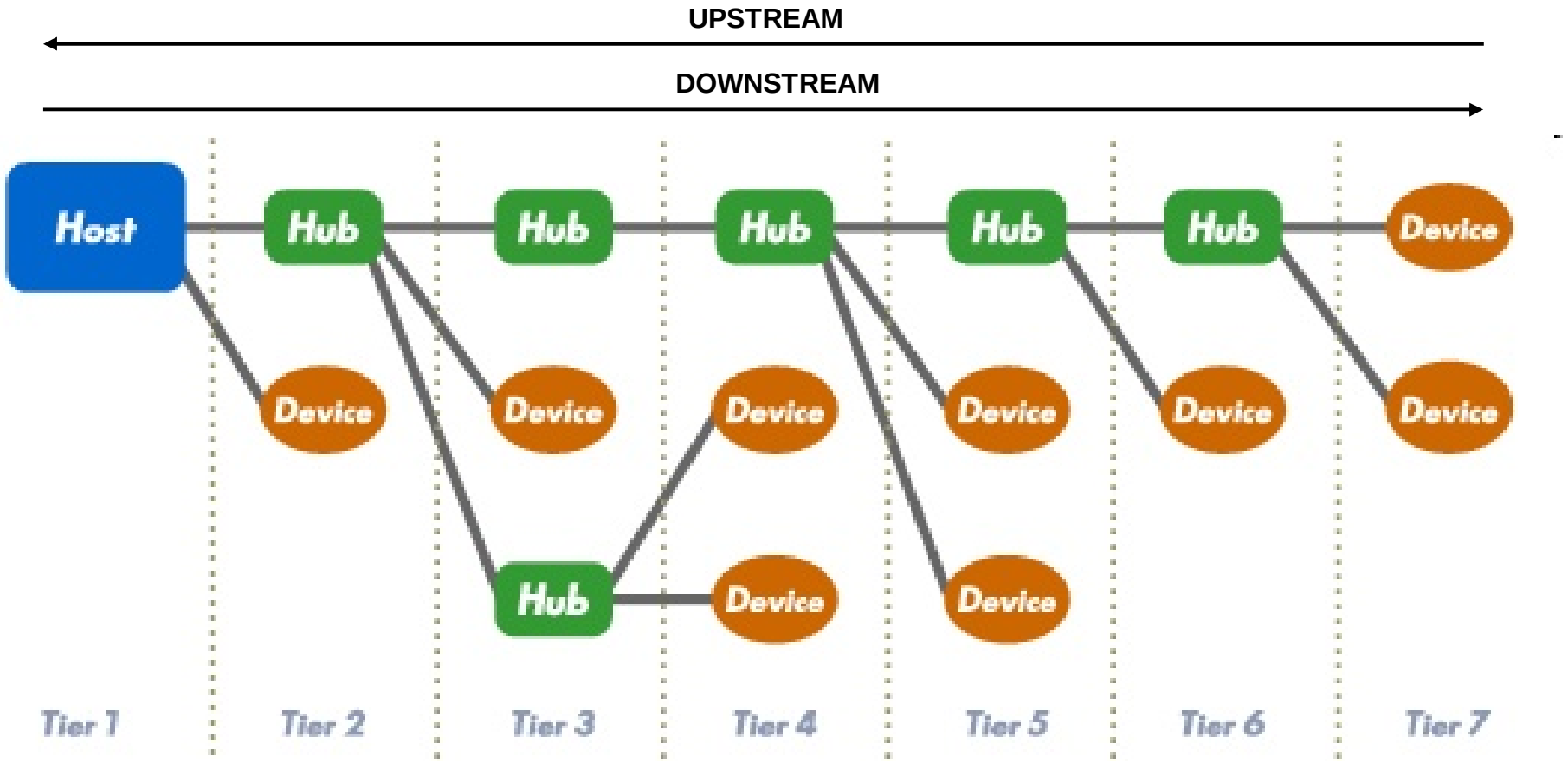


Ramki USB



- Każdy transfer to jedna lub więcej transakcji
- Zależnie od ilości danych transakcja może składać się z jednej bądź kilku ramek,
- Każda transakcja rozpoczyna się od podania adresu urządzenia, wskazania Endpoint danej transakcji oraz określenia jej kierunku.

Topologia USB



max. 127 urządzeń podłączonych na jednej magistrali HOSTa,
max. 7 poziomów (a więc nie więcej niż 5xHUB w szeregu)

Obowiązki urządzeń HOST oraz DEVICE

- HOST
  - Wykrywanie urządzeń podłączonych do magistrali,
  - Centralne zarządzanie wymianą danych,
  - Kontrola błędów,
  - Zasilanie urządzeń DEVICE,
  - Wymiana danych z urządzeniami DEVICE
- DEVICE
  - Rozpoznanie komunikacji z kierowanej do urządzenia,
  - Dostosowanie się do rozkazów HOSTa,
  - Kontrola błędów,
  - Zarządzenie energią,
  - Wymiana danych z urządzeniem HOST

Protokół wymiany danych

- RS232 – protokół wymiany danych nie jest ustalony (dowolnie stosuje się własne protokoły),
- USB – ma kilka warstw odpowiedzialnych za organizację transmisji danych,
- Kompletna transakcja USB składa się z:
  - Token Packet (Header defining what it expects to follow)
  - Optional Data Packet, (Containing the payload)
  - Status Packet (Used to acknowledge transactions and to provide a means of error correction)

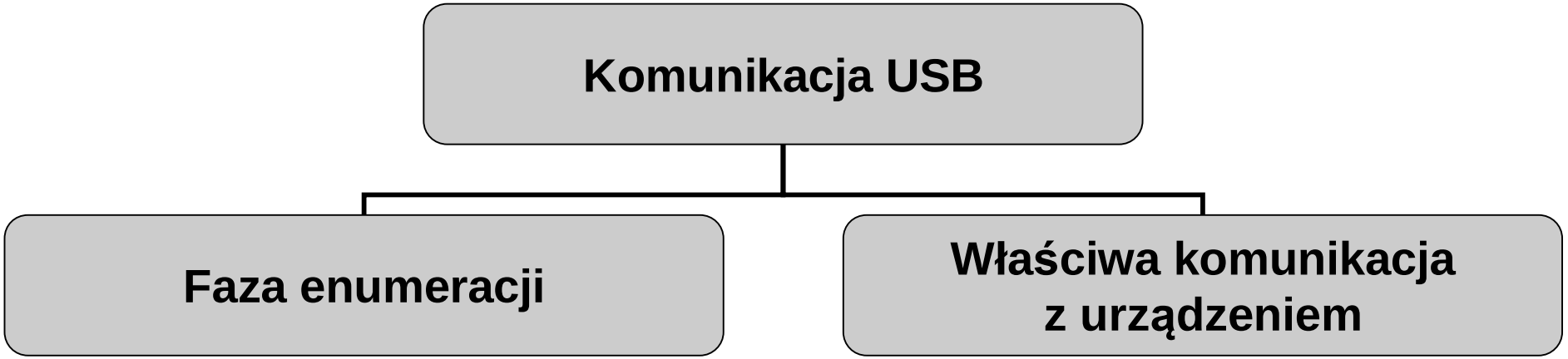
Klasy urządzeń

- Standard definiuje klasy urządzeń
- Idea ta umożliwia specyfikację dla każdej z klas minimalnej funkcjonalności oraz cech – urządzenia tej samej klasy mogą używać tych samych sterowników, np.:
  - Human Interface Device: mysz, klawiatura, joystick,
  - Mass Storage Device: dysk USB, pendrive, karta pamięci, telefon komórkowy
- Wyjście poza daną klasę implikuje konieczność pisania własnych driverów urządzenia

Klasy urządzeń

- Klasy USB *Device Working Group* wspierane przez MS Windows:
  - Bluetooth class
  - Chip/smart card interface devices (CCID)
  - Hub class
  - Human interface device (HID)
  - Mass storage class (MSC)
  - Printing class
  - Scanning/imaging (PTP)
  - Media Transfer (MTP)
  - USB Audio class
  - Modem class (CDC)
  - Video class (UVC)

Komunikacja USB

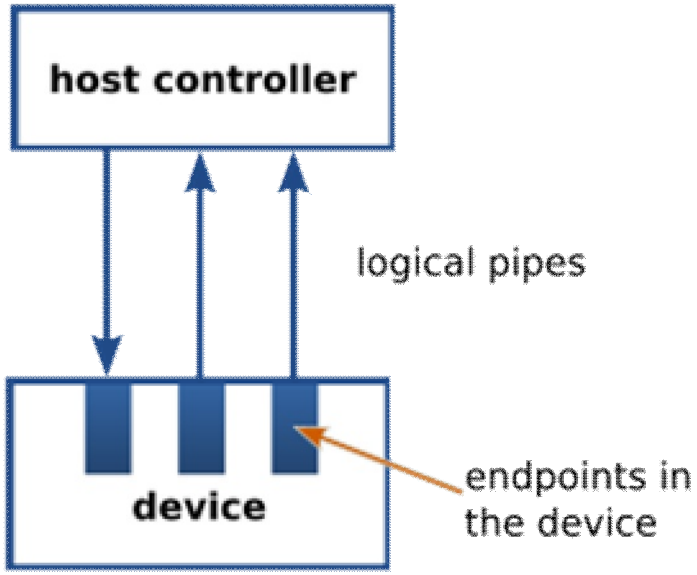


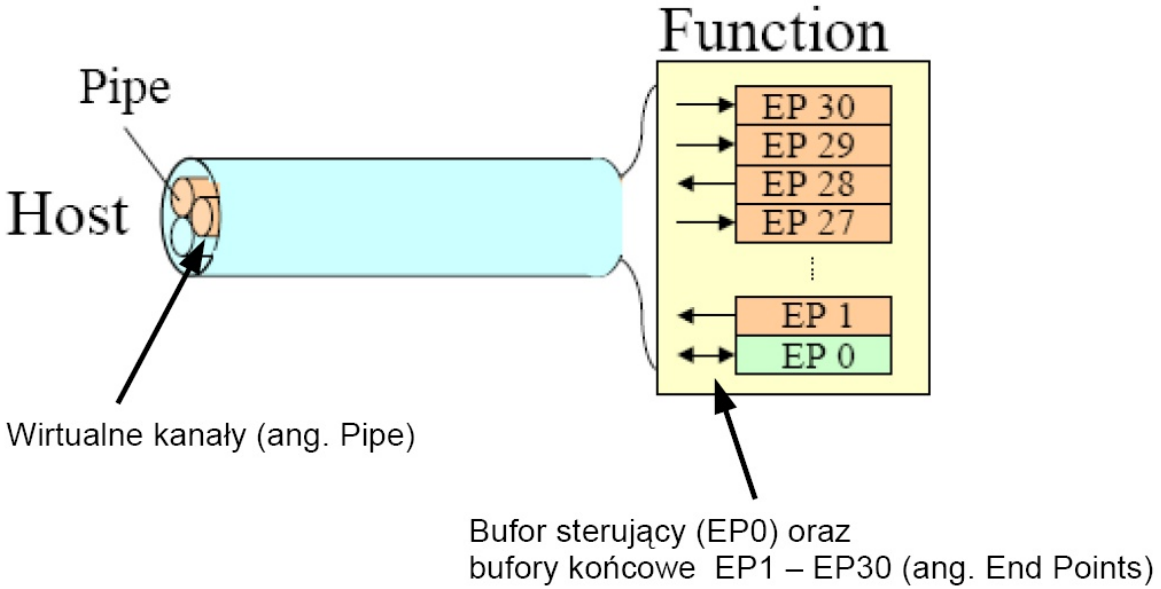
- series of standard requests from the host,
- device must identify each request, return requested information, and take other actions specified by the requests

- host has exchanged enumeration information with the device and a device driver has been assigned and loaded
- application communications can begin

Protokół wymiany danych

- Uproszczony model wymiany danych w standardzie USB





- Jak HOST rozpoznaje urządzenie, jak je konfiguruje, jak wygląda protokół???

EP - *Endpoint*

- ENDPOINT (EP) - *a uniquely addressable portion of a USB device that is the source or sink of information in a communication flow between the host and device*
- Podstawowa jednostka komunikacji w standardzie USB ->
  - endpoint is a buffer that stores multiple bytes. Typically the endpoint is a block of data memory or a register in the controller chip
- Wszelkie transmisje USB odbywają się z lub do konkretnego EP urządzenia w wirtualnych kanałach (Logical Pipe) tworzonych pomiędzy urządzeniem HOST a odpowiednimi EPs urządzeń,
- EP jest jednokierunkowym nadajnikiem bądź odbiornikiem danych, wyjątkiem jest Control Endpoint – EP 0
- Istnieją różne typy EP (w zależności od przyjętego typu transferu danych).

EP - *Endpoint*

- Adres EP składa się z numeru (0 .. 15) oraz kierunku (**określony z perspektywy HOSTa**)
  - EP 1IN – dane wysyłane do urządzenia HOST przez urządzenie podrzędne,
  - EP 2OUT – dane wysyłane przez urządzenie HOST do urządzenia podrzędnego.
- Control EP (zawsze EP 0) – musi umożliwiać transfery w dwu kierunkach (para IN oraz OUT EPs o wspólnym numerze),
- Ilość EPs:
  - full- lub high-speed:
    - 30 dodatkowych EPs: adresy EP 1 .. 15, w kierunkach IN oraz OUT
  - low-speed:
    - 2 dodatkowe EPs: EP 1IN & EP 1OUT lub EP 1IN & EP 2IN.

EP - *Endpoint*

- Każda wymiana danych poprzedzona jest informacją o numerze EP, kierunku wymiany danych oraz o ew. transakcji Setup (kontrolnej)

Transaction Type	Source of Data	Types of Transfers that Use this Transaction Type	Contents
IN	device	all	data or status information
OUT	host	all	data or status information
Setup	host	control	a request

LP - *Logical Pipe*

- Logical Pipe – logiczne połączenie pomiędzy urządzeniem HOST oraz konkretnym EP (może być kilka LPs), nawiązywane na etapie enumeracji.
- Zestaw parametrów opisujących Logical Pipe:
  - Wielkość pasma
  - Typ transferu danych
  - Kierunek transmisji
  - Maksymalna wielkość pakietu / bufora danych.
- *Default Control Pipe* – dwukierunkowe połączenie nawiązane z EP 0 IN oraz EP 0 OUT (Control EP) do transferu danych sterujących oraz na etapie konfiguracji urządzenia.
- Dwa rodzaje Logical Pipes:
  - Stream Pipe
  - Message Pipe

Strategie transferu danych

- USB zaprojektowano dla różnych urządzeń, przesyłających dane z różną prędkością, o różnym charakterze, z różnym „czasem reakcji” oraz kontrolą błędów.
- Istnieją cztery strategie transferu danych:
  - Control Transfers
  - Bulk Transfers
  - Interrupt Transfers
  - Isochronous Transfers

Strategie transferu danych

- Control
  - używany na etapie konfiguracji urządzenia
- Interrupt
  - transmisje odbywają się w ściśle określonych odstępach czasu (host w określonych odstępach odpytuje urządzenie o ew. nowe dane); używane w urządzeniach wejściowych (np. klawiatur, mysz),
- Isochronous
  - nieustanne, okresowe transfery krytycznych czasowo danych, bez kontroli błędów, spore pasmo danych; używane w urządzeniach multimedialnych audio / wideo,
- Bulk
  - do transmisji dużych ilości danych, jednocześnie z najniższym priorytetem (jeśli magistrala jest zajęta transmisje są opóźniane), ścisła kontrola błędów transmisji (CRC oraz retransmisja); używane w urządzeniach klasy Mass Storage Device oraz np. skanerach.

Strategie transferu danych

Type	Important attributes	Max size LS	Max size FS	Max size HS	Examples
Interrupt	Quality + time	8	64	3072	Mouse, keyboard
Bulk	Quality	-	64	512	Printer, scanner
Isochronous	time	-	1023	3072	Audio, video
Control	Quality + time	8	64	64	System control

Dariusz Makowski, TM, DMCS

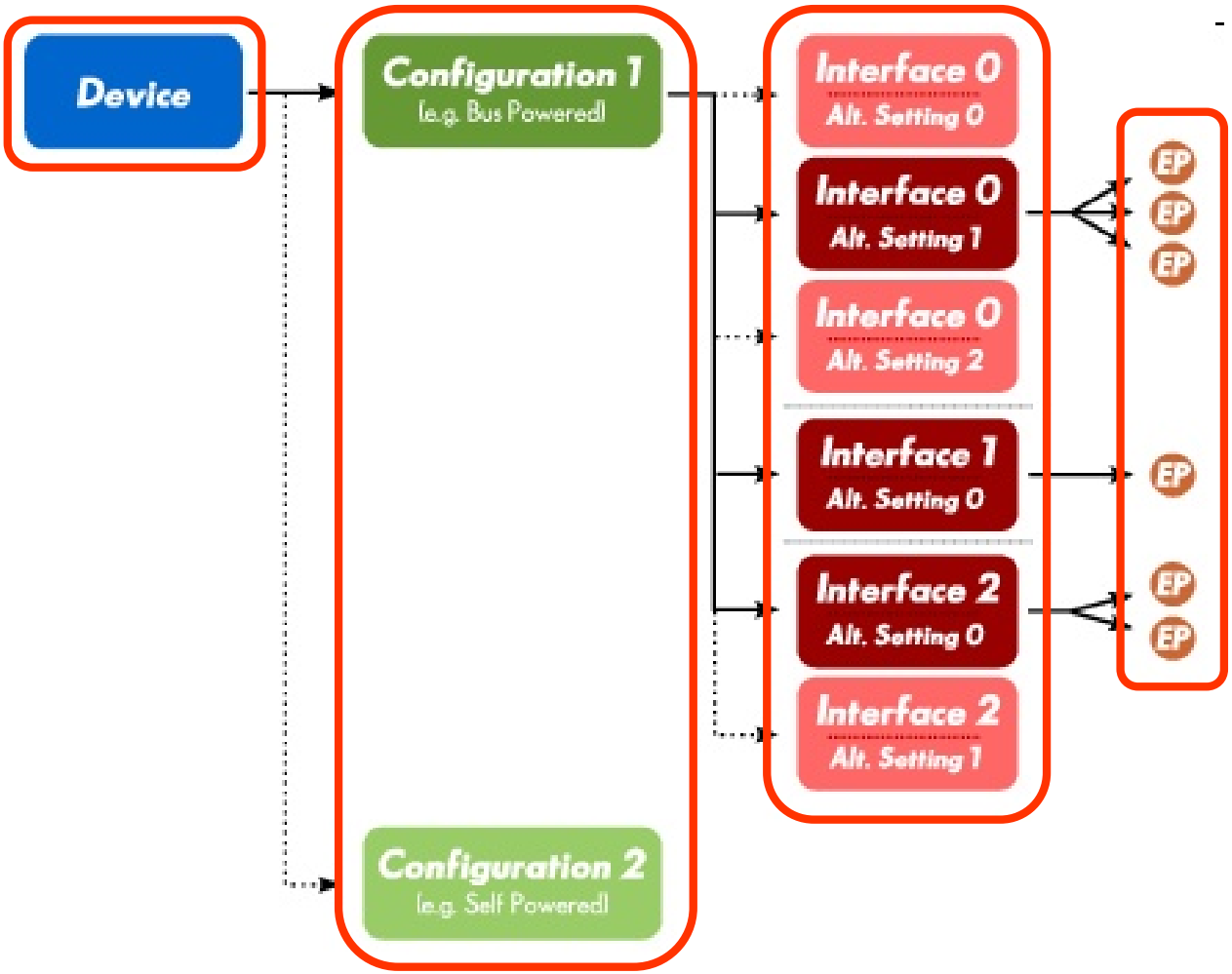
Enumeracja urządzeń

- Enumeracja następuje po podłączeniu urządzenia do HOSTa – ten kolejno:
  - rozpoznaje obecność oraz typ (prędkość) urządzenia,
  - resetuje podłączone urządzenie,
  - odpytuje o deskryptory urządzenia,
  - nadaje unikalny, 7-bitowy adres na magistrali,
  - odpytuje o deskryptory konfiguracji i interfejsów,
  - ładuje odpowiedni sterownik,
  - wybiera konfiguracje urządzenia

Enumeracja urządzeń

- W wyniku enumeracji Host przypisuje urządzeniom indywidualne adresy oraz ustala podstawowe parametry transmisji:
  - Adres urządzenia w przestrzeni USB,
  - Rodzaj transferu,
  - Kierunek transmisji danych,
  - Rozmiar przesyłanych pakietów,
  - Szybkość transmisji,
  - Adresy buforów używanych przez sterowniki urządzenia,
  - Prąd pobierany przez urządzenie.

Deskryptory urządzeń



- Każde urządzenie posiada:
  - pojedynczy **deskryptor urządzenia** (definiuje możliwe konfiguracje)
  - **deskryptory konfiguracji** (tylko jeden aktywny w danym momencie) – każdy definiuje własne interfejsy
  - **deskryptory interfejsów** (z różnymi ustawieniami, tylko jedne aktywne w danym momencie) – każdy aktywny definiuje własne ***Endpoints*** (przez deskryptory)

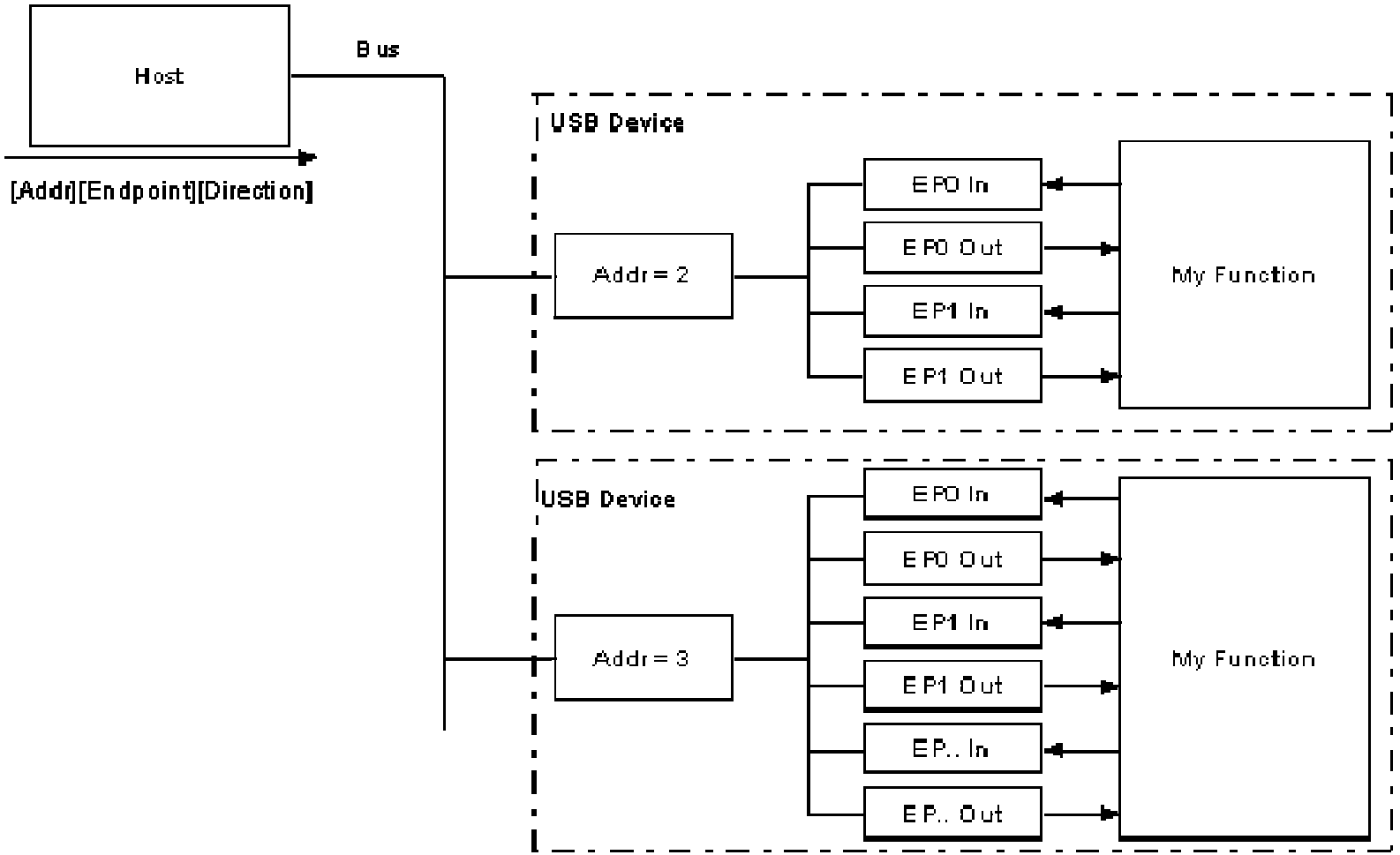
Deskryptory urządzeń

- ***Device Descriptor***: Each USB device can only have a single Device Descriptor. This descriptor contains information that applies globally to the device, such as serial number, vendor ID, product ID, etc. The device descriptor also has information about the device class. The host PC can use this information to help determine what driver to load for the device.
- ***Configuration Descriptor***: A device descriptor can have one or more configuration descriptors. Each of these descriptors defines how the device is powered (e.g. bus powered or self powered), the maximum power consumption, and what interfaces are available in this particular setup. The host can choose whether to read just the configuration descriptor or the entire heirarchy (configuration, interfaces, and alternate interfaces) at once.

Deskryptory urządzeń

- ***Interface Descriptor***: A configuration descriptor defines one or more interface descriptors. Each interface number can be subdivided into multiple alternate interfaces that help more finely modify the characteristics of a device. The host PC selects particular alternate interface depending on what functions it wishes to access. The interface also has class information which the host PC can use to determine what driver to use.
- ***Endpoint Descriptor***: An interface descriptor defines one or more endpoints. The endpoint descriptor is the last leaf in the configuration hierarchy and it defines the bandwidth requirements, transfer type, and transfer direction of an endpoint. For transfer direction, an endpoint is either a source (IN) or sink (OUT) of the USB device.
- ***String Descriptor***: Some of the configuration descriptors mentioned above can include a string descriptor index number. The host PC can then request the unicode encoded string for a specified index. This provides the host with human readable information about the device, including strings for manufacturer name, product name, and serial number.

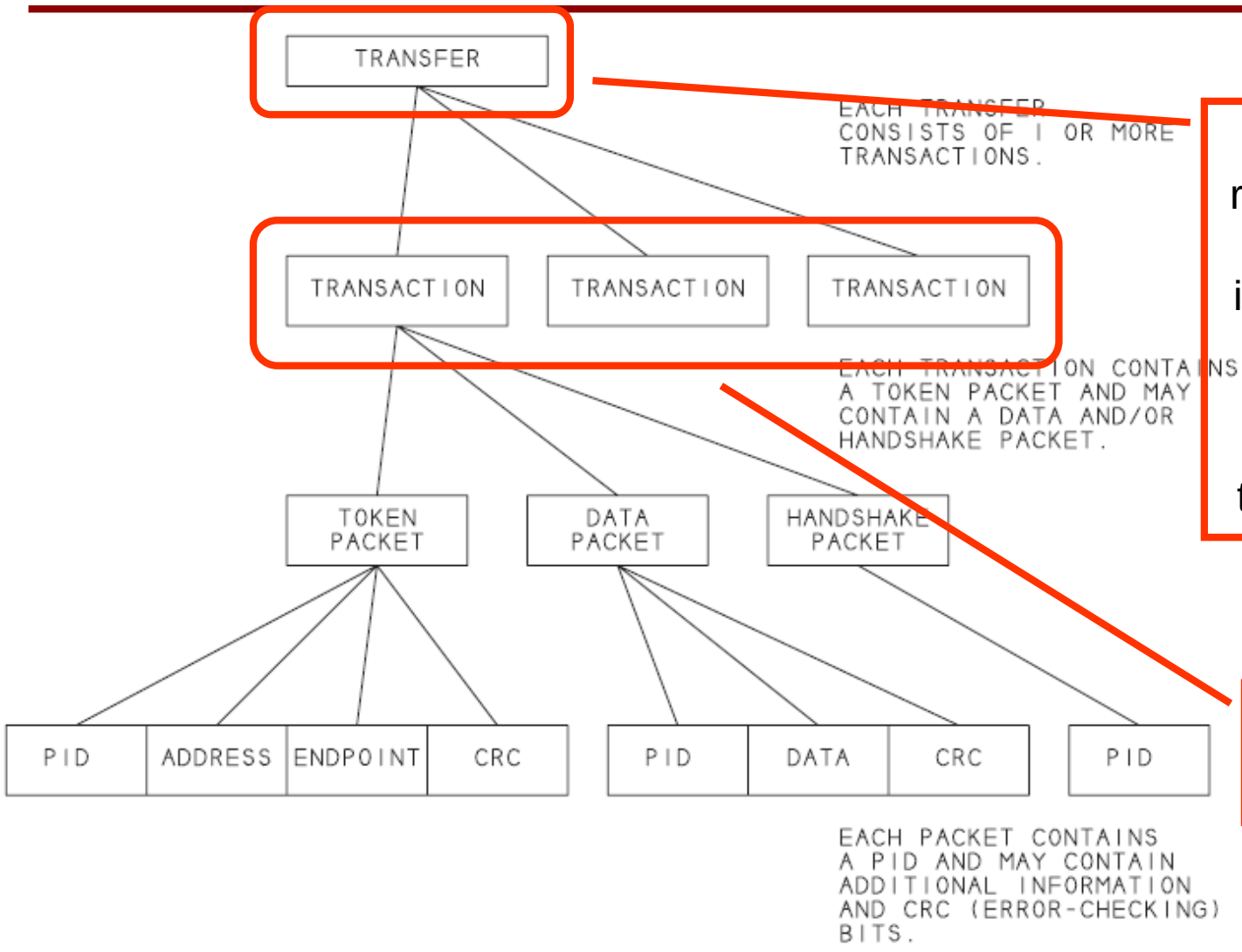
Magistrala USB po enumeracji



Transfer danych

- transfer is a the process of making and carrying out a communication request
- transaction is the delivery of service to an endpoint (either the host’s sending information to the device, or the host’s requesting and receiving information from the device)

Transfer danych



Może trwać przez kilka ramek, ale musi zakończyć się nie zakłócony (zakaz innej komunikacji w trakcie transakcji). Wniosek – urządzenia muszą b. szybko reagować na transfery urządzenia Host.

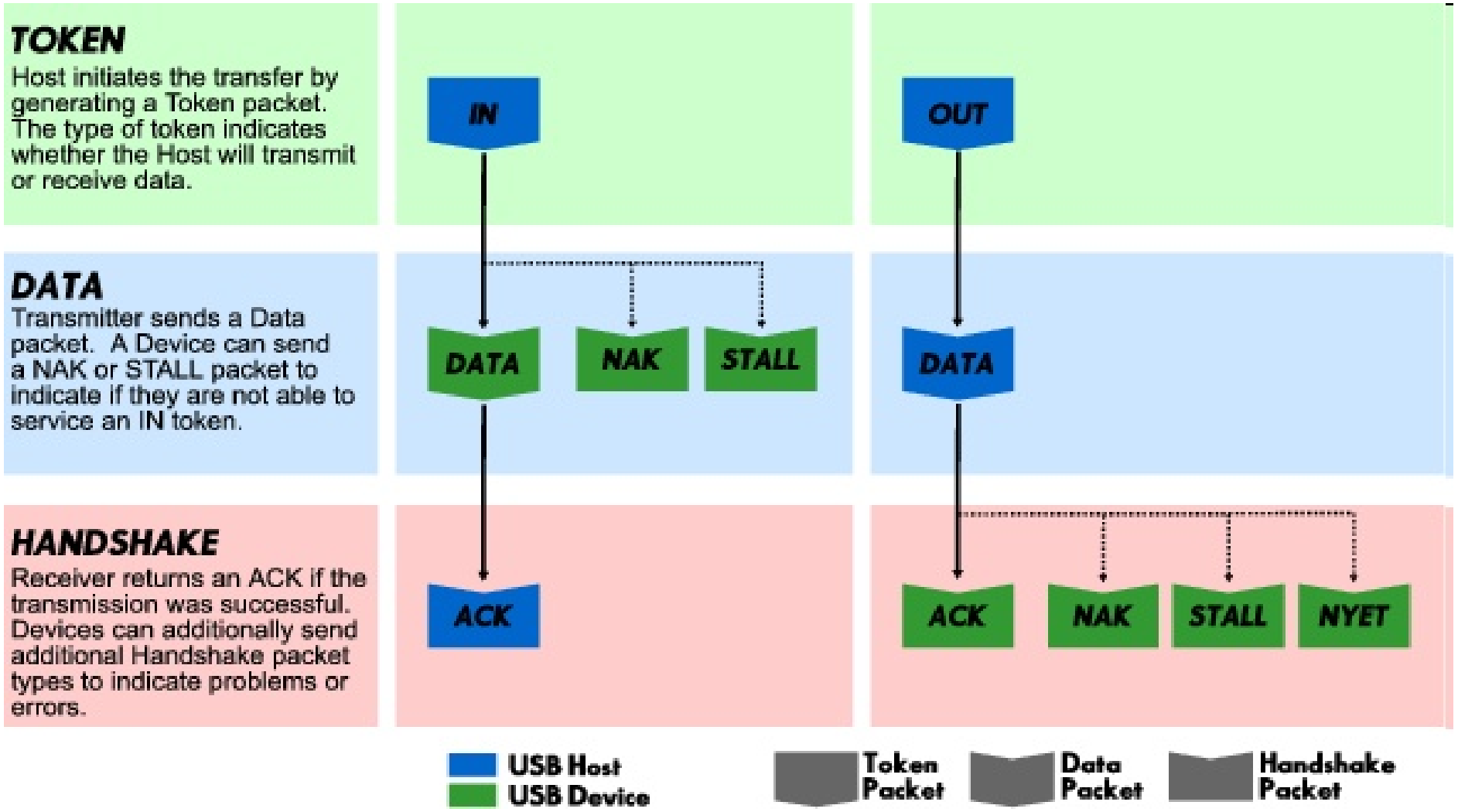
Różnego typu – IN, OUT bądź SETUP

Fazy transakcji – TOKEN, DATA & HANDSHAKE

- Transakcje składają się z maksymalnie 3 faz:
  - TOKEN,
  - DATA,
  - HANDSHAKE
- Każda faza to jeden lub dwa pakiety.

Transfer Type		**Transactions**	**Phases (packets). Each downstream, low-speed packet is also preceded by a PRE packet.**
Control	Setup Stage	One transaction	Token
			Data
			Handshake
	Data Stage	Zero or more transactions (IN or OUT)	Token
			Data
			Handshake
	Status Stage	One transaction (opposite direction of transaction(s) in the Data stage or IN if there is no Data stage)	Token
			Data
			Handshake
Bulk		One or more transactions (IN or OUT)	Token
			Data
			Handshake
Interrupt		One or more transactions (IN or OUT)	Token
			Data
			Handshake
Isochronous		One or more transactions (IN or OUT)	Token
			Data

Fazy transakcji – TOKEN, DATA & HANDSHAKE



Transmisja danych – TOKEN, DATA & HANDSHAKE

- Transmisja kontrolowana z poziomu urządzenia HOST.
- Faza pierwsza (TOKEN):
  - HOST wysyła *TOKEN packet* do urządzenia o konkretnym adresie oraz EP,
  - TOKEN może być typu:
    - *IN* – HOST oczekuje danych od konkretnego ADDR/EP
    - *OUT* – HOST będzie wysyłał dane do konkretnego ADDR/EP
    - *SETUP* – HOST będzie wysyłał dane konfiguracyjne do urządzenia

Transmisja danych – TOKEN, DATA & HANDSHAKE

- Faza druga (DATA):
  - Wymiana danych w pakietach, ew. informacja NAK lub STALL, gdy urządzenie nie jest w stanie obsłużyć otrzymanego tokena IN
- Faza trzecia (HANDSHAKE):
  - Strona odbiorcza wysyła ACK, NAK lub STALL aby zakończyć wymianę danych.

- Opisany schemat dotyczy wszystkich typów transmisji poza *Isochronous* (nie ma tam fazy handshake).

Pakiety USB

- Pakiety USB:
  - Token Packets
  - Data Packets
  - Handshake Packets
  - Start Of Frame Packets

Składać mogą się z następujących pól...

Name	SIze (bits)	Packet Types	Purpose
SYNC	8	all	Start-of-packet and synchronization
PID	8	all	Identify the packet type
Address	7	IN, OUT, Setup	Identify the function address
Endpoint	4	IN, OUT, Setup	Identify the endpoint
Frame Number	11	SOF	Identify the frame
Data	0 to 8192 (1024 bytes) for 2.0 hardware; 0 to 8184 (1023 bytes) for 1.x hardware	Data0, Data1	Data
CRC	5 or 16	IN, OUT, Setup, Data0, Data1	Detect errors

Pakiety USB

- Wysyłane od LSB do MSB
- Każdy zaczyna się polem SYNC, które pełni rolę znacznika SOP (Start Of Packet), dodatkowo zapewnia synchronizację odbiornika do przesyłanych danych
  - 8-bit w przypadku urządzeń *low/full speed,*
  - 32-bity w przypadku urządzeń *high speed*.
- Pole Packet ID - PID (8-bit) definiuje typ pakietu (4-bity w wersji prostej oraz kolejne 4 w wersji zanegowanej)

Pakiety USB

- Pole ADDR – 7-bit, umożliwia zaadresowanie do 127 urządzeń
  - 0x00: adres specjalny, odpowiadają na niego wszystkie nie-zainicjalizowane urządzenia
- Pole ENDP – 4-bity, umożliwia zaadresowanie do 16 Endpoints
  - Ograniczenie w przypadku urządzeń *low speed* – max. 4 EP (2 additional endpoints on top of the default, control pipe)

Pakiety USB

- Pole CRC – liczone dla *payload*,
  - 5-bit dla pakietów z grupy TOKEN
  - 16-bit dla pakietów z grupy DATA

SOF - pakiet początku ramki (mikroramki):

Pakiety zapowiedzi: IN, OUT, SETUP:

Pakiety sterujące

Wielomiany generujące sumy kontrolne CRC

$$G(x) = x^5 + x^2 + 1.$$

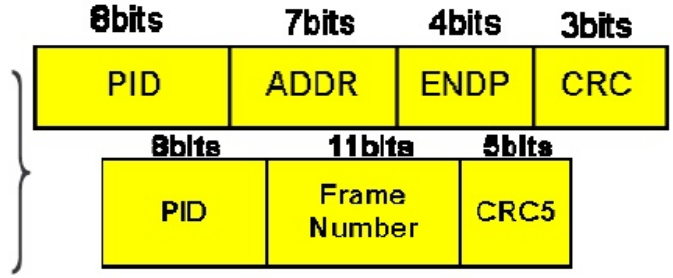
Pakiet danych:

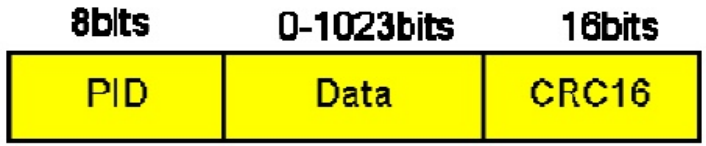
Ramka danych

$$G(x) = x^{16} + x^{15} + x^2 + 1.$$

Typy pakietów USB

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



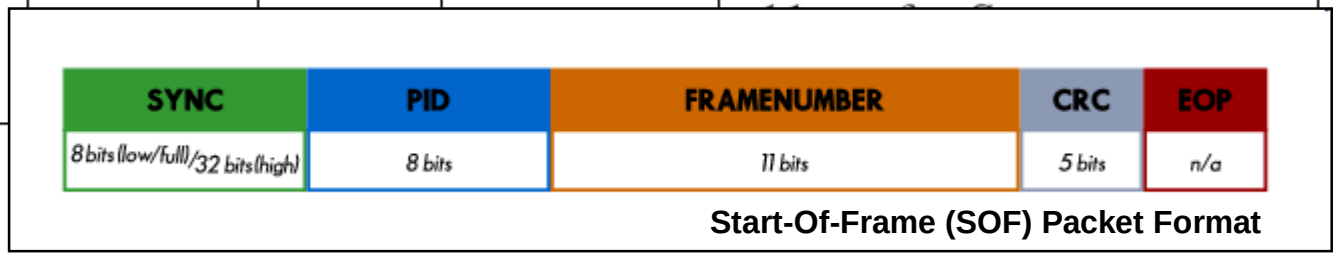


Dariusz Makowski, TM, DMCS

Pakiety TOKEN

Packet Type	PID Name	Value	Transfer types used in	Source	Bus Speed	Description
Token (identifies transaction type)	OUT	0001	all	host	all	Device and endpoint address for OUT (host-to-device) transaction.
	IN	1001	all	host	all	Device and endpoint address for IN
	SOF	0101				
	SETUP	1101				

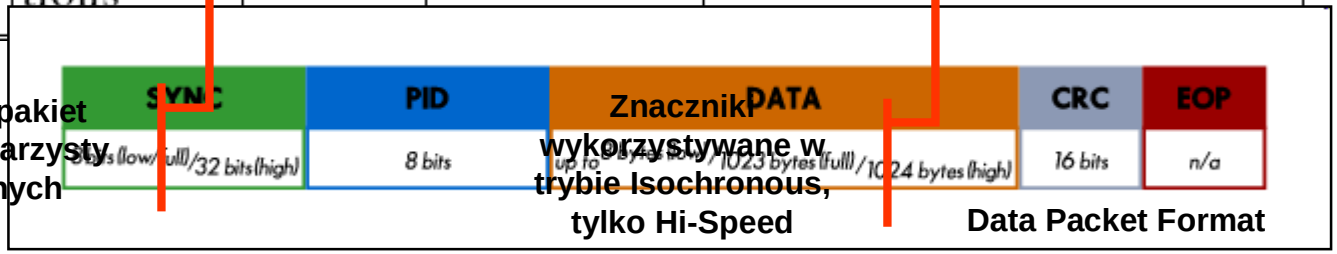
Token Packet Format



Start-Of-Frame (SOF) Packet Format

Pakiety DATA

Packet Type	PID Name	Value	Transfer types used in	Source	Bus Speed	Description
Data (carries data or status code)	DATA0	0011	all	host, device	all	Data toggle, data PID sequencing
	DATA1	1011	all	host, device	all	Data toggle, data PID sequencing
	DATA2	0111	isoch.	host, device	high	Data PID sequencing
	MDATA	1111	isoch., split transac-tions	host, device	high	Data PID sequencing



Data Packet Format

Poprzedza pakiet danych, parzysty pakiet danych

Poprzedza pakiet danych, nieparzysty pakiet danych

Znaczniki wykorzystywane w trybie Isochronous, tylko Hi-Speed

Pakiety HANDSHAKE

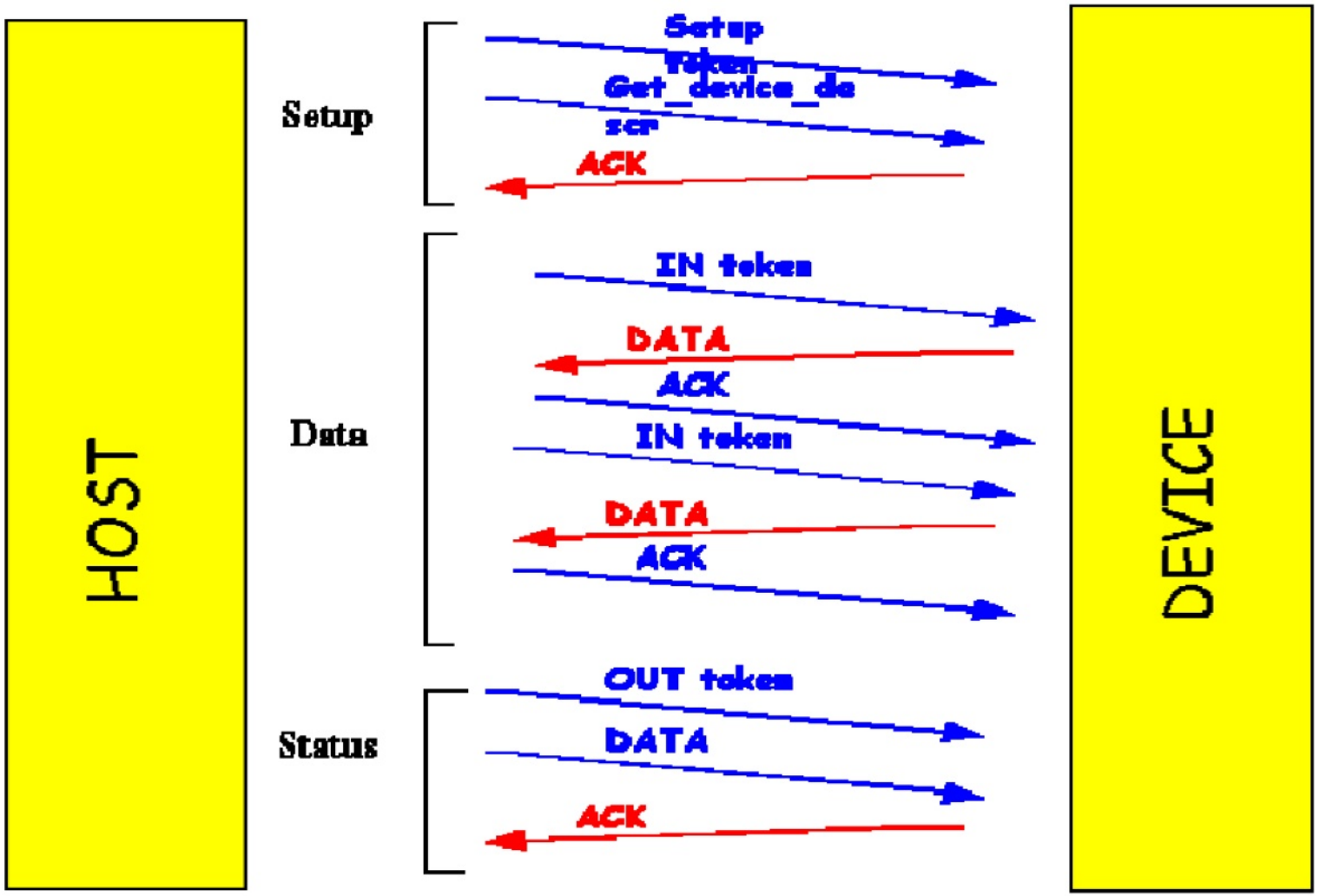
Packet Type	PID Name	Value	Transfer types used in	Source	Bus Speed	Description
Handshake (carries status code)	ACK	0010	all	host, device	all	Receiver accepts error-free data packet.
	NAK	1010	control, bulk, interrupt	device	all	Receiver can't accept data or sender can't send data or has no data to transmit.
	STALL	1110	control, bulk, interrupt	device	all	A control request isn't supported or the endpoint is halted.
	NYET	0110	control Write, bulk OUT, split transactions	device	high	Device accepts error-free data packet but isn't yet ready for another or

Handshake Packet Format

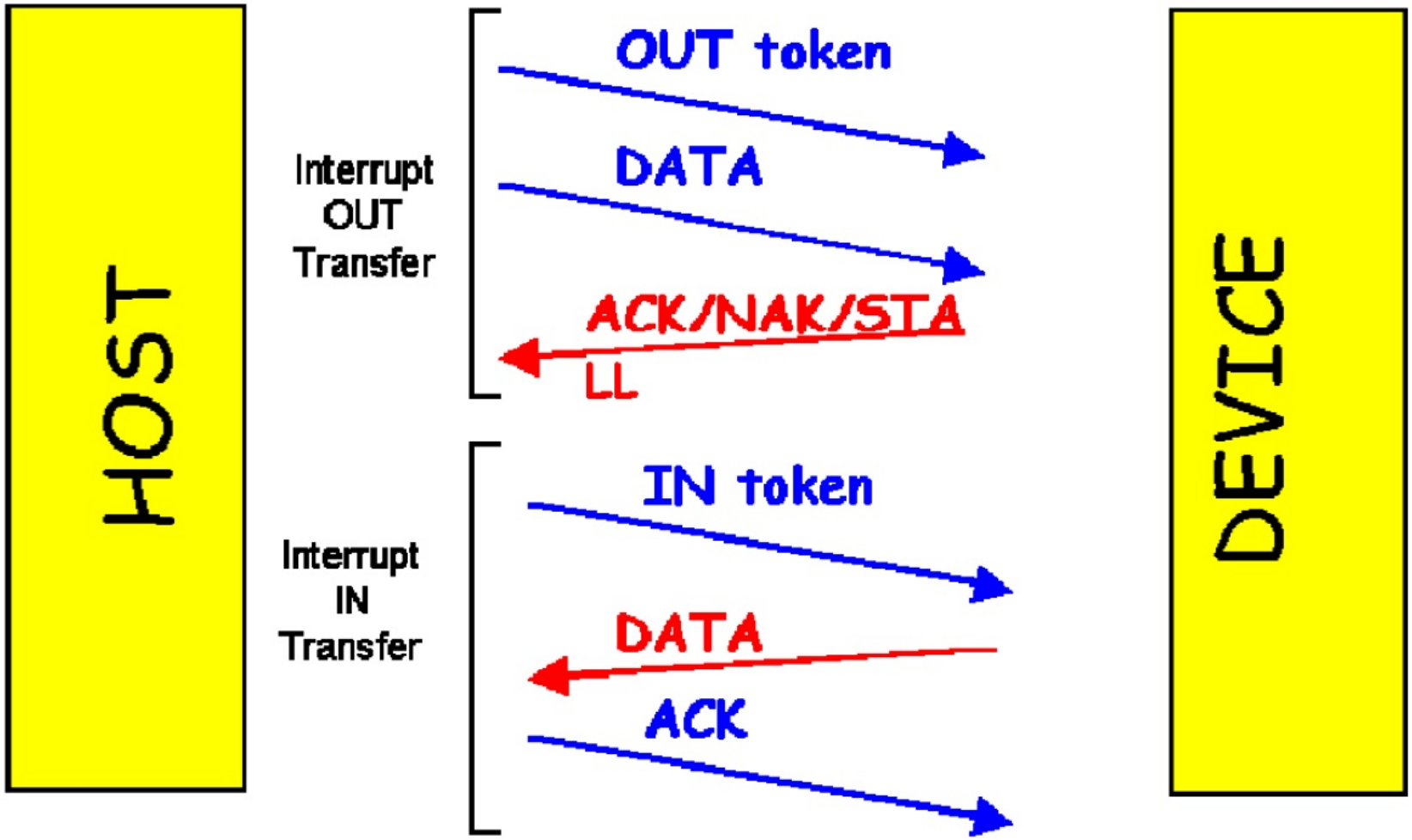
Pakiety SPECIAL

Packet Type	PID Name	Value	Transfer types used in	Source	Bus Speed	Description
Special	PRE	1100	control, interrupt	host	full	Preamble issued by host to indicate that the next packet is low speed.
	ERR	1100	all	hub	high	Returned by a hub to report a low- or full-speed error in a split transaction.
	SPLIT	1000	all	host	high	Precedes a token packet to indicate a split transaction.
	PING	0100	control Write, bulk OUT	host	high	Busy check for bulk OUT and control Write data transactions after NYET.
	reserved	0000	–	–	–	For future use.

Transfer Setup (sterujący)

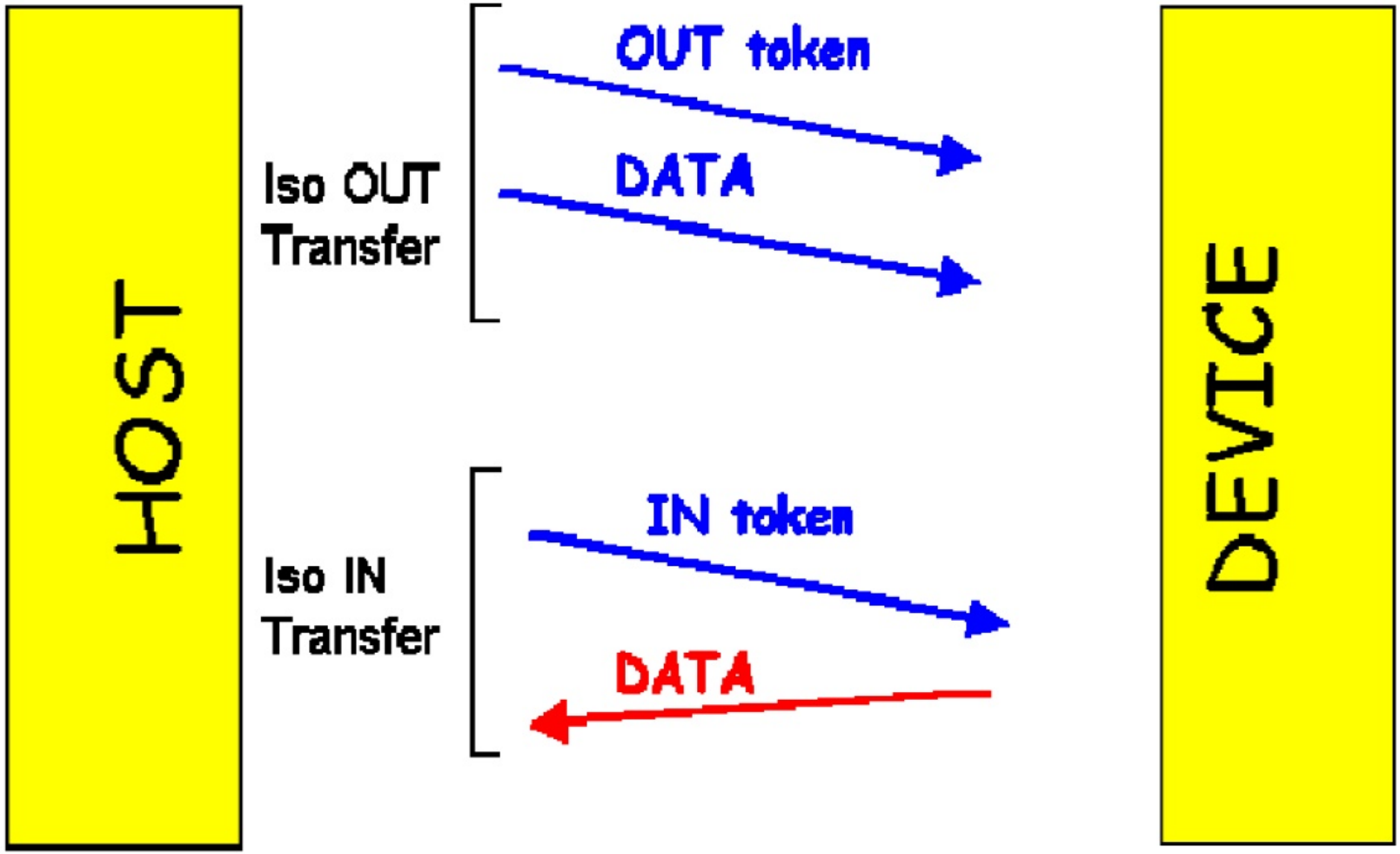


Transfer Interrupt i Isochronous (przerwaniowy i izochroniczy)



Dariusz Makowski, TM, DMCS

Transfer Bulk (masowy)



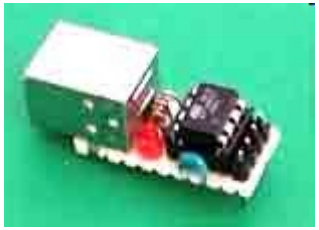
Dariusz Makowski, TM, DMCS

Projektowanie urządzenia USB

- Mikrokontroler z interfejsem USB, zewnętrzny układ kontrolera USB bądź emulacja programowa,
- Odpowiedni firmware w projektowanym urządzeniu obsługujący komunikację USB,
- Firmware „właściwy”,
- Sterownik urządzenia dla urządzenia HOST, który umożliwia komunikację,
- Aplikacja współpracująca ze sterownikiem – jeśli jest wymagana.

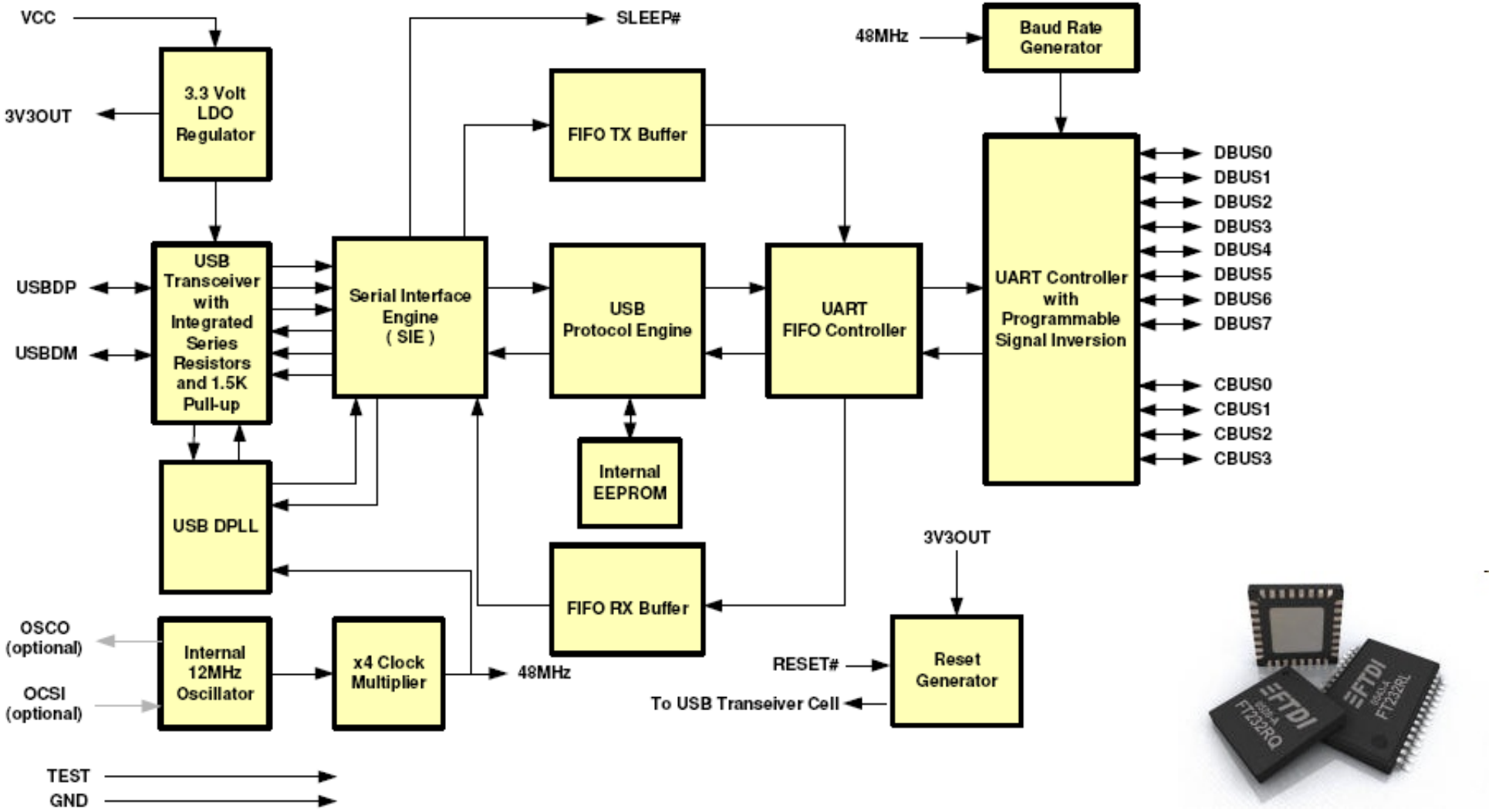
Wykorzystanie USB

- OBDEV (AVR-USB: software-only implementation of a low-speed USB)
- AVR-CDC (USB-RS232C Interface using Communication Device Class)

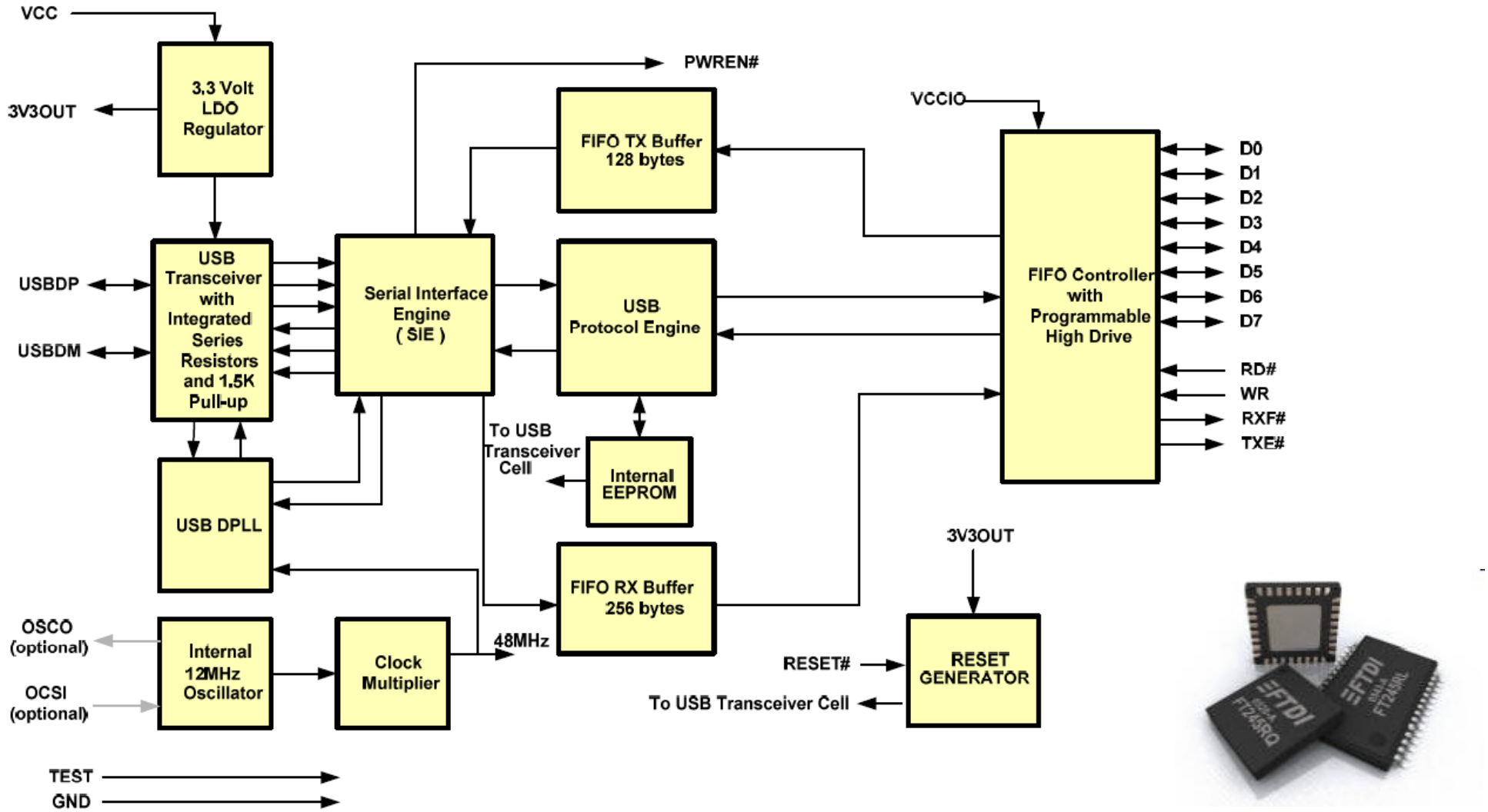


- ISP1583 - Hi-Speed USB peripheral controller
- PDIUSBD12 – Full-Speed USB peripheral controller with parallel buss
- MAX3420 / MAX3421 – Hi-Speed USB device / device and host controller
- FT232R / FT245R - Single chip USB to asynchronous serial data transfer / parallel FIFO bidirectional data transfer interface

FT232R


VCC
3V3OUT
3.3 Volt LDO Regulator
SLEEP#
48MHz
Baud Rate Generator
FIFO TX Buffer
USBDP
USBDM
USB Transceiver with Integrated Series Resistors and 1.5K Pull-up
Serial Interface Engine (SIE)
USB Protocol Engine
UART FIFO Controller
UART Controller with Programmable Signal Inversion
DBUS0
DBUS1
DBUS2
DBUS3
DBUS4
DBUS5
DBUS6
DBUS7
CBUS0
CBUS1
CBUS2
CBUS3
Internal EEPROM
USB DPLL
3V3OUT
FIFO RX Buffer
OSCO (optional)
OSCI (optional)
Internal 12MHz Oscillator
x4 Clock Multiplier
48MHz
RESET#
To USB Transeiver Cell
Reset Generator
TEST
GND

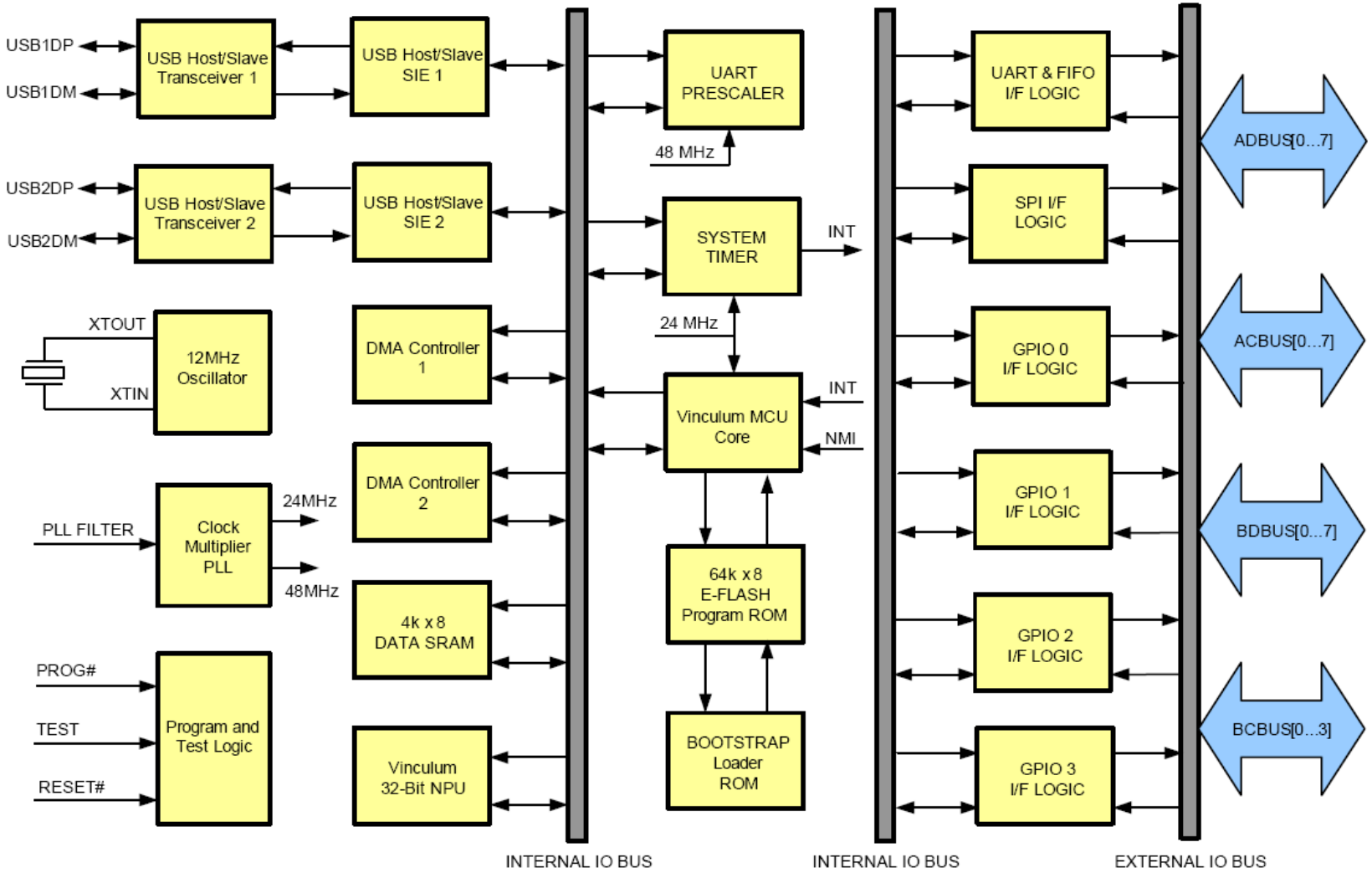
FT245R


VCC
3.3 Volt LDO Regulator
3V3OUT
PWREN#
FIFO TX Buffer 128 bytes
VCCIO
USB Transceiver with Integrated Series Resistors and 1.5K Pull-up
USBDP
USBDM
Serial Interface Engine (SIE)
USB Protocol Engine
FIFO Controller with Programmable High Drive
D0
D1
D2
D3
D4
D5
D6
D7
RD#
WR
RXF#
TXE#
To USB Transceiver Cell
Internal EEPROM
USB DPLL
3V3OUT
FIFO RX Buffer 256 bytes
OSCO (optional)
OSCI (optional)
Internal 12MHz Oscillator
Clock Multiplier
48MHz
RESET#
RESET GENERATOR
To USB Transceiver Cell
TEST
GND

FT232R & FT245R

- Virtual COM Port (VCP) drivers and direct (D2XX) drivers
- The VCP driver emulates a standard PC serial port such that the USB device may be communicated with as a standard RS232 device. The D2XX driver allows direct access to a USB device via a DLL interface.

Vinculum VNC1L



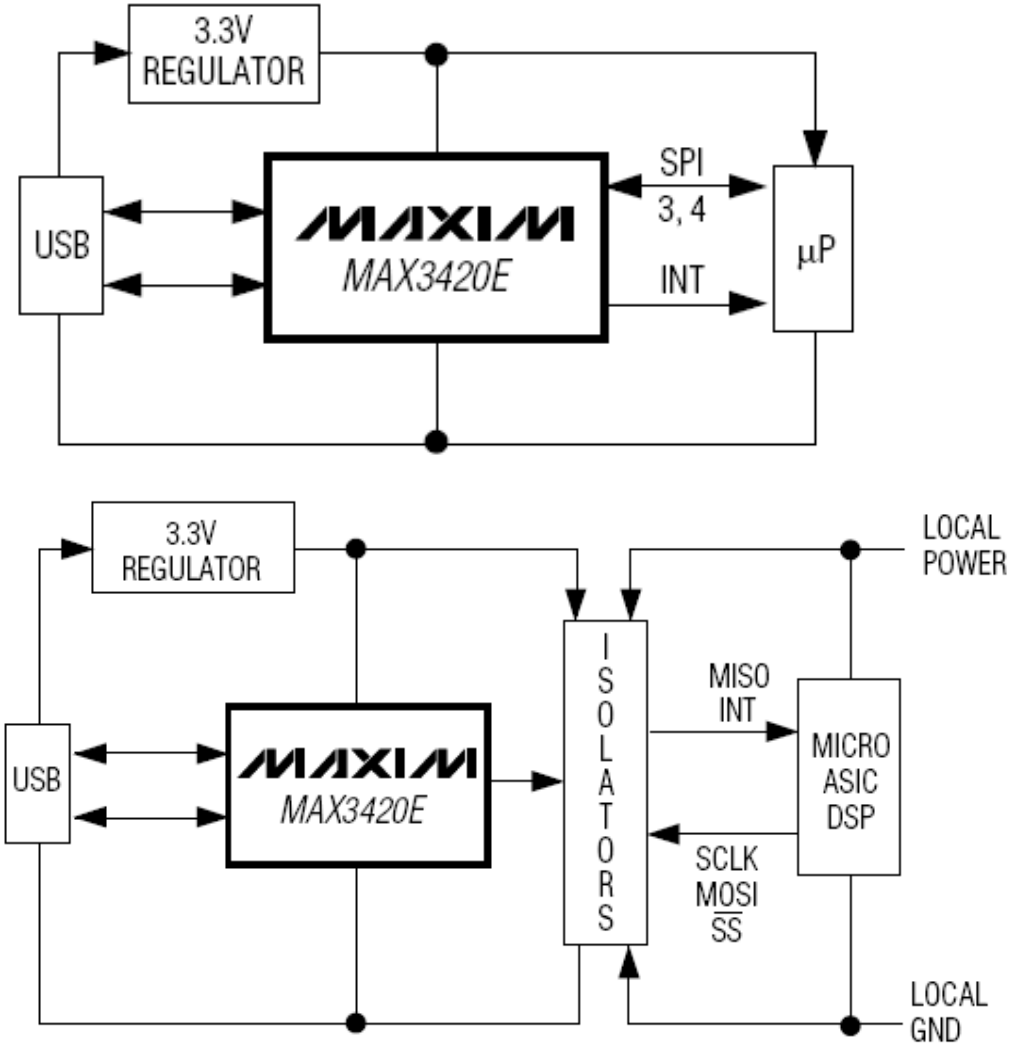
Vinculum VNC1L

- Single chip embedded dual USB Host Controller IC
  - 8/32 bit Vinculum-MCU Core
  - Dual DMA controllers for hardware acceleration
  - 64k Embedded Flash Program Memory, 4k internal Data SRAM
  - 2 x USB 2.0 Low/Full-speed Host/Slave Ports
  - UART, SPI and Parallel FIFO interfaces
  - Integrated firmware allows read from and write to FAT format USB Flash keys,
  - Up to 28 GPIO pins depending on configuration
  - 3.3V operation with 5V safe inputs
  - Low power operation (25mA running/2mA standby)
  - Inbuilt FTDI firmware easily updated in the field (over UART or USB)
  - LQFP-48 RoHS compliant package
  - Multi-processor configuration capable

Vinculum VNC1L

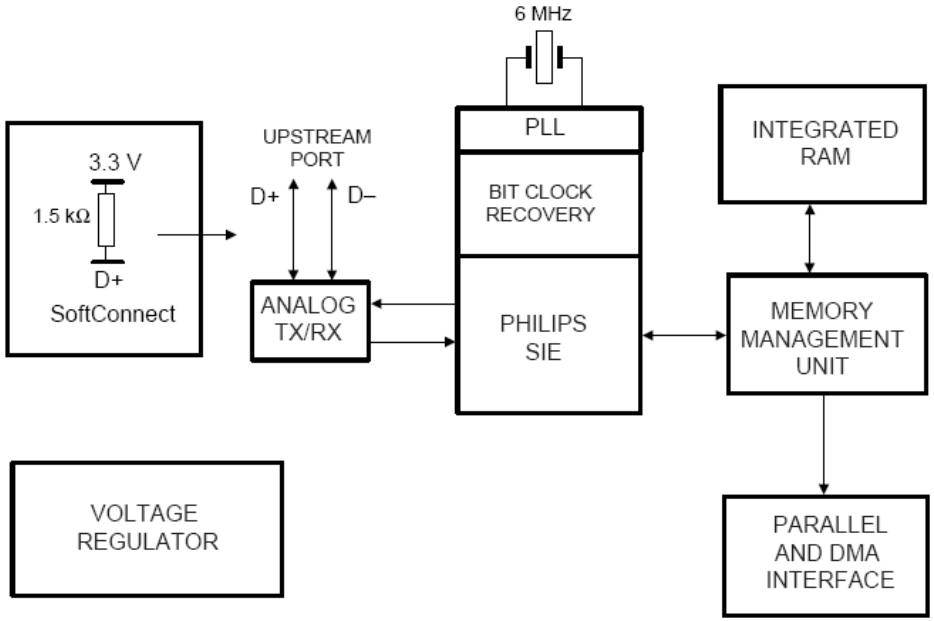
- Typical Applications
  - Add USB host capability to embeddedproducts.
  - Interface USB Flash drive to MCU/PLD/FPGA.
  - USB Flash drive to USB Flash drive file transfer interface.
  - Digital camera to USB Flash drive or other USB slave device interface.
  - PDA to USB Flash driver or other USB slave device interface.
  - MP3 Player to USB Flash drive or other USB slave device interface.
  - USB MP3 Player to USB MP3 Player.
  - Mobile phone to USB Flash drive or other USB slave device interface.
  - GPS to mobile phone interface.
  - Instrumentation USB Flash drive or other USB slave de
  - Data-logger USB Flash drive or other USB slave devic
  - Set Top Box - USB device interface.
  - GPS tracker with USB Flash disk storage.

MAX3420E & MAX3421



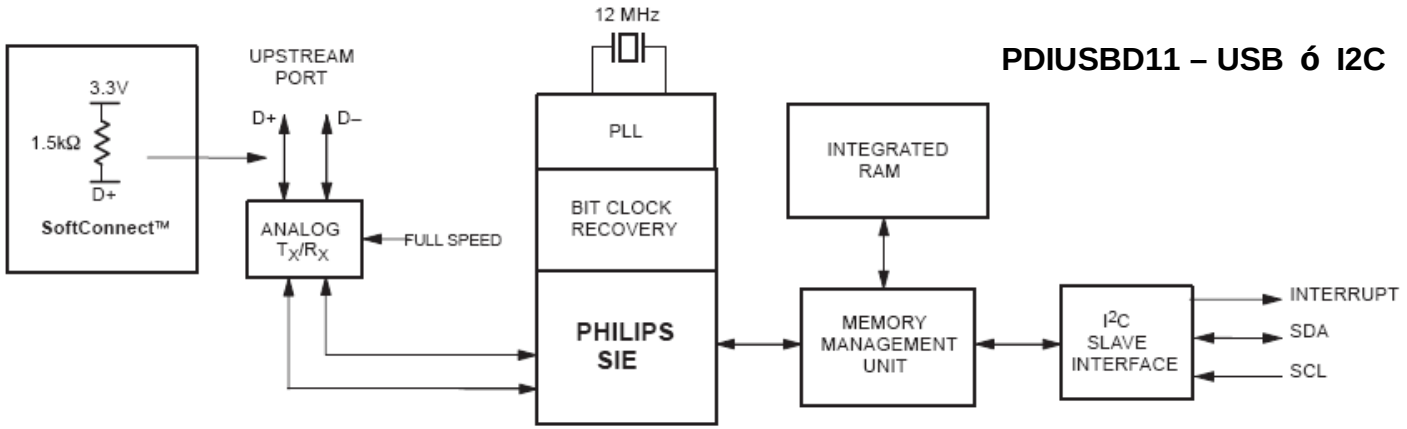
- Microprocessor-Independent USB 2.0 (Full-Speed Operation) Solution
- Firmware/Hardware Control of an Internal D+ Pullup Resistor
- Programmable 3- or 4-Wire 26MHz SPI Interface
- Internal Comparator Detects VBUS for Self-Powered Applications
- Interrupt Output Pin (Level or Programmable Edge) Allows Polled or Interrupt-Driven SPI Interface
- Intelligent USB Serial-Interface Engine (SIE)
  - Automatically Handles USB Flow Control and Double Buffering
  - Handles Low-Level USB Signaling Details
  - Contains Timers for USB Time-Sensitive Operations So SPI Master Does Not Need to Time Events
- Built-In Endpoint FIFOs
  - EP0: CONTROL (64 Bytes)
  - EP1: OUT, Bulk or Interrupt, 2 x 64 Bytes (Double-Buffered)
  - EP2: IN, Bulk or Interrupt, 2 x 64 Bytes (Double-Buffered)
  - EP3: IN, Bulk or Interrupt (64 Bytes)
- Four General-Purpose Inputs and Four General-Purpose Outputs

PDIUSBD11, PDIUSBD12



PDIUSBD12 – USB ó PARALLEL

PDIUSBD11 – USB ó I2C



Mikrokontrolery ze zintegrowanym USB

- Cypress CY7C68013A
  - USB 2.0 High-Speed
  - Rozbudowane 8051, pamięć ładowana z USB, EEPROM, 16kB pamięci programu RAM
- Microchip PIC16C745
  - 8k pamięci programu,
  - Low Speed 1Mbps USB, 6 EP
- Freescale 68HC705JB4
  - HC05 Core, 3584 bytes ROM, 176 bytes RAM
  - 1,5 Mbps Low Speed peripheral
  - 1 Control, 2 Interrupt EP
- Atmel AVR AT90USB1286
  - 128 kB pamięci programu, 8kB SRAM,
  - USB 2.0, Full-Speed, Control EP0 & 6 EPs with IN, OUT directions – Bulk, Interrupt & Isochronous Transfers
- Atmel AT91S128
  - 128kB pamięci programu, 32kB SRAM
  - USB Full Speed (12 Mbps) peripheral port

Więcej o USB

- Wikipedia EN:
  - http://en.wikipedia.org/wiki/USB
- Total Phase: http://www.totalphase.com
- Beyond Logic: USB in a NutShell
  - http://www.beyondlogic.org/usbnutshell/usb1.htm
- USB Made Simple
  - http://www.usbmadesimple.co.uk/
- USB Central: http://www.lvr.com/usb.htm

- AVR-USB:
  - http://www.obdev.at/products/avrusb/index.html
- AVR-CDC:
  - http://www.recursion.jp/avrcdc/

Politechnika Łódzka

Instytut Elektroniki