

# Interfejsy w Systemach Komputerowych - ULTIMATE

SonMati  
Ervelan  
Doxus

25 stycznia 2015

# Pytania i odpowiedzi

## 1 RS-232

### Prawda/Fałsz

- RS-232 jest portem przeznaczonym do synchronicznej transmisji znakowej. Generator taktu odpowiedzialny za wyprowadzanie znaków typowo ustawiany jest na: 1200bd, 2400bd, 4800bd, 9600bd, 19200bd.

*RS-232 jest portem przeznaczonym do asynchronicznej transmisji znakowej. Da się sztucznie stworzyć synchroniczną transmisję.*

- Linie kontrolne w interfejsie RS-232 to: DTR, DSR, RTS, CTS, RI, DCD. Pary DTR/DSR i RTS/CTS wykorzystywane są do realizacji handshake'u w połączeniach bezmodemowych.

*Tak, te pary linii mogą być wykorzystywane do handshake'a podczas gdy RxD i TxD zajmują się przesyłem danych.*

- Transakcja w systemie MODBUS składa się z zapytania (query) wysyłanego przez stację Slave i odpowiedzi odsyłanej przez stację Master.

*Jest odwrotnie - zapytanie wysyła Master, a odpowiedź odsyła Slave.*

- W trybie transmisji ASCII znacznikiem początku ramki jest znak ':', a końca ramki para znaków CR LF. W trybie transmisji RTU znacznikiem początku ramki jest znak 'Ctrl-A', a końca para znaków CTRL-Y CTRL-Z.

*Zdanie jest poprawne dla ASCII. Dla RTU, znacznikiem początku i końca ramki jest przerwa o długości minimum 4T, gdzie T jest czasem trwania jednego znaku.*

- Standard RS-232 transmituje znaki synchronicznie, bity w znakach [asynchronicznie]

*Ostatnie słowo ucięte, więc spekuluję że tak właśnie było napisane. To nieprawda, jest odwrotnie.*

- Standard RS-422 pozwala na osiągnięcie szybkości 10MBodów na odległość 100m.

*IMO pozwala, na slajdzie 12 jest napisane że 10 Mbd przy zasięgu DO 100m - czyli 100m chyba też.*

- Liniami kontrolnymi w RS-232 nie są linie TxD, RxD, SG.

*Owszem, TxD i RxD są liniami danych, a SG to po prostu masa.*

- System MODBUS składa się z faz zapytania i odpowiedzi.

*Tak właśnie jest.*

- W systemie MODBUS
  - Obowiązuje master/slave.

*Peunie, a w dodatku Slave'ów może być wielu.*

- Prędkości transmisji wynoszą od 1200 do 19200bd.

*Jak najbardziej.*

- Ramka w ASCII może mieć format 7N2 (lub np. 7E1, 7O1).

*Tak, patrz warstwa fizyczna MODBUS.*

- Ramka w RTU może mieć format 8N2 \*(lub np. 8E1, 8O1).

*Tak, patrz warstwa fizyczna MODBUS.*

- W trybie transmisji RTU jest kontrola błędów CRC.

*Tak, jest elementem budowy ramki RTU.*

- Bit kontrolny w RS-232 zależy od bitu danych i bitu stopu.

*Bit kontrolny służy do kontroli parzystości/nieparzystości, nie ma związku z bitem stopu.*

- Za pomocą RS-232 możemy połączyć ze sobą 2 stacje DCE  
*Połączyć możemy dwie stacje DTE, lub DTE z DCE. Dwie stacje DCE łączą się za pomocą łącza telefonicznego.*
- W MODBUS kontrola błędów jest realizowana za pomocą LRC lub CRC.  
*Tak, LRC wykorzystywane jest w trybie ASCII, CRC w trybie RTU.*
- Do portu RS 485 można podłączyć tylko jedno urządzenie, ale za to obsługiwać go z dużo większą szybkością i na większą odległość niż jest to możliwe w przypadku interfejsu RS 232.  
*Można podłączyć do 32 stacji.*
- Format ramki w protokole Modbus jest następujący: znacznik początku ramki, adres urządzenia slave, adres mastera, pole danych, znacznik końca ramki.  
*Opis nie pasuje ani do trybu ASCII, ani RTU*
- RS 232 jest portem przeznaczonym dla asynchronicznej transmisji znakowej, realizowanej zazwyczaj w trybie dupleksowym, czyli dwukierunkowej transmisji niejednoczenej (naprzemiennej)  
*Tryb dupleksowy jest równoczesny, to półduplexowy jest niejednoczesny.*
- W interfejsie RS 232 linie TxD i RxD służą do transmisji znaków, natomiast DTR, RTS to wyjścia kontrolne, a DSR, CTS, RI i DCD to wejścia kontrolne.  
*Indeed*
- Multipleksowanie urządzeń ze znakowym portem asynchronicznym pozwala na ich kontrolę poprzez jeden port RS-232.  
*Żeby kontrolować kilka urządzeń z jednego portu potrzebny jest koncentrator. Jeśli "używanie koncentratora" równe się "multipleksowanie", to PRAWDA.*
- Węzeł podrzędny w systemie MODBUS po wykryciu błędu w komunikacie wysyła potwierdzenie negatywne do węzła nadzorującego.  
*W odpowiedzi pole to jest wykorzystywane do pozytywnego lub negatywnego potwierdzenia wykonania poleceń.*
- Czy w trybie ASCII systemu MODBUS każdy bajt wysyłany jest jako znak z przedziału 0x00, 0xFF?  
*Bajt dzielimy na 2 części i wysyłamy jako 2 znaki z przedziału 0-9 i Ah-Fh*

## 2 USB

### Prawda/Fałsz

- Kontrola urządzenia USB odbywa się poprzez zapisy komunikatów do bufora o numerze 0 i odczytanie informacji statusowych z bufora o numerze 0.  
*Zgadza się.*
- W przypadku błędu transmisji każda transakcja USB jest powtarzana, ponieważ niedopuszczalne jest przekazywanie danych przekłamanych.  
*Transakcje izochroniczne nie są powtarzane w przypadku błędu transmisji.*
- Hub nie dopuszcza ruchu full speed do portów, do których są podłączone urządzenia low speed.  
*Tak, urządzenie lowspeed blokuje możliwość włączenia fullspeed na swoim porcie.*  
*Urządzenie lowspeed jest obsługiwane z prędkością lowspeed, ale nie ma to wpływu na prędkość pozostałych urządzeń w hubie. Pakiety wysyłane do takiego urządzenia są po prostu poprzedzane specjalną preambułą*
- Reset portu USB polega na rekonfiguracji hosta, po której host zapisuje tablicę deskryptorów do urządzenia podłączonego do tego portu.  
*Reset portu USB polega na rekonfiguracji urządzenia. W następcej procedurze enumeracji między innymi dochodzi do odczytu tablicy deskryptorów z urządzenia przez host.*  
*Reset polega na przywróceniu urządzenia do stanu domyślnego. Potem host rozpoczyna konfigurację urządzenia, najpierw odczytując deskryptor urządzenia, potem nadając mu unikatowy adres, a następnie odczytując tablicę deskryptorów, dzięki którym wie, jak ma to urządzenie skonfigurować.*

- Typowa transakcja USB składa się z pakietów żądania i odpowiedzi, z których każdy potwierdzany jest osobnym potwierdzeniem.

*Typowa transakcja USB składa się z pakietów token, data i handshake. Transakcje izochroniczne nie są potwierdzane.*

- W systemie USB urządzenia zgłaszą żądania do hosta, który je kolejkuje i następnie obsługuje w kolejności pojawiania się zgłoszenia.

*Urządzenia nie zgłaszą żądań, tylko są odpytywane przez hosta. Host nie tworzy jednej kolejki, tylko w miarę możliwości stara się obsługiwać wszystkie urządzenia jednocześnie, równomiernie, zapobiegając zawłaszczeniu.*

- W USB można połączyć kaskadowo do 5 hubów, korzystających z zasilania magistralowego

*Podłączyć je można tylko korzystając z zasilania zewnętrznego lub hybrydowego. Przy zasilaniu magistralowym zabraknie zasilania już na drugim hubie. Co więcej, należy mieć na uwadze maksymalne dopuszczalne opóźnienie sygnału, które przy przejściu przez 5 hubów jest osiągane - 350ns. Urządzenia podpięte do 5'tego huba mogą nie działać poprawnie.*

- Mechanizm data toggle w USB służy do przywracania synchronizacji pomiędzy hostem i urządzeniem, utracone na skutek wystąpienia błędów w pakietach danych.

*Mechanizm data toggle przywraca synchronizację pomiędzy hostem i urządzeniem na skutek błędu w potwierdzeniu odsyłanym przez odbiorcę.*

*Transakcja jest powtarzana w przypadku błędego odebrania pakietu danych (odbiorca wysyła wówczas pakiet NAK handshake). Gdy wystąpiło błędne odebranie potwierdzenia ACK handshake przez nadawcę, dane wysypane są jeszcze raz (wówczas odbiorca je odrzuca i wysyła pakiet ACK handshake)*

- Host kontroler USB komunikuje się z interfejsem magistrali USB urządzenia peryferyjnego za pomocą fizycznego kanału komunikacyjnego.

*Tak, używamy kabla.*

- Kamera internetowa może przesyłać obraz do komputera za pomocą transferu izochronicznego z szybkością LowSpeed w interfejsie USB.

*Z tabelki można wyczytać, że dla transferu izochronicznego nie można wykorzystać szybkości LowSpeed.*

- Pakiety USB przesyłane z szybkością LowSpeed muszą być poprzedzone pakietem preambuły

*Tak, jest on charakterystyczny dla pakietów przesyłanych z szybkością LowSpeed*

- Urządzenie peryferyjne USB 2.0 może być podłączone do host kontrolera za pośrednictwem maksymalnie sześciu hubów.

*Aby spełnić normę (ograniczenie czasowe oczekiwania na odpowiedź), można połączyć za pośrednictwem maksymalnie 5 hubów.*

- Pole PID w pakiecie USB zabezpieczone jest 16-bitową sumą kontrolną CRC.

*Pole PID zabezpieczone jest 4-bitowym polem kontroli, będącym prostą negacją bitów pola PID.*

- Do portu dolnego huba podłączane mogą być tylko wtyki USB typu B.

*Tylko wtyki typu A.*

- Transakcja dzielona w USB 1.1 składa się z dwóch części: SSPLIT i CSPLIT.

*Takie czary dopiero w USB 2.0*

- W przypadku połączenia USB HighSpeed wykonywane jest podparcie linii D- do Vcc za pośrednictwem rezystora 1,5k.

*Po podłączeniu urządzenia High Speed wpierw jest ono identyfikowane jako Full Speed, więc wykonywane jest podparcie linii D+ do Vcc za pośrednictwem rezystora 1,5k. Następnie, poprzez chirp ("ćwierkanie") host i urządzenie ustalają, czy możliwa jest komunikacja w trybie High Speed. Jeśli tak, usuwane jest podparcie przez rezistor, a obwód zamknięty jest terminatorem.*

- W kodowaniu NRZI co sześć jedynek jest wstawiany bit synchronizacji "0".

*Pomieszczone pojęcia. W kodowaniu NRZI nie występuje dodawanie bitu synchronizacji. Proces ten nazywa się bit stuffing. Zdanie było by poprawne, gdyby brzmiało np. W kodowaniu NRZI z bit stuffingiem co sześć.*

- Transakcje kontrolna i przerwaniowa w USB 1.1 są transakcjami aperiodycznymi z gwarantowanym pasmem w ramach jednej mikroramki.  
*Transakcja kontrola jest transakcją aperiodyczną. Transakcja przerwaniowa jest transakcją periodyczną.*
- W kontrolerze OHC transakcje izochroniczne są porządkowane/kolejkowane w drzewo/strukturę drzewiastą.  
*Tak, OHC wykorzystuje strukturę drzewa, a UHC tablicę wskaźników (listę podwieszaną).*
- Standard USB 2.0 wymaga skręconych, ekranowanych kabli.  
*Well, High speed all the way, więc wymaga*
- Transfer kontrolny i przerwaniowy są transferami aperiodycznymi.  
*Było podobne pytanie. Transfer kontrolny jest aperiodyczny, transfer przerwaniowy jest periodyczny.*
- Wielowarstwowa architektura USB 2.0 składa się z 3 warstw.  
*Tak - warstwa interfejsu magistrali USB, warstwa urządzenia USB, warstwa funkcji urządzenia*
- W porcie USB dane są dzielone na transakcje.  
*Dane w ramce są dzielone na transakcje, więc tak*
- Hub podłączony do portu USB ma obciążalność 100uA.  
*Hub podłączony do portu USB bez własnego zasilania (zasilanie magistralowe) ma obciążalność dla portów dolnych do 100mA na port (maksymalną 400mA na cały hub). Hub z zasilaniem zewnętrznym lub hibrydowym ma obciążalność do 500mA na port.*
- W systemie USB do mechanizmów kontroli danych należą:
  - Przełączanie pakietów danych  
*Tzw. Data Toggle*
  - Wykrywanie braku aktywności na linii danych *Tak, po 3 ms braku aktywności USB przechodzi w tryb zmniejszonego poboru prądu.*
  - Zabezpieczenie znacznika SOF lub EOF  
*Reakcją jest natomiast objęte wystąpienie fałszywego znacznika końca pakietu (false EOP)*
  - kodowanie LRC  
*Pakiety zabezpieczone są kodowaniem CRC.*
- Wydajność dolnego portu (USB 2.0) wynosi 500mA.  
*Nie wiadomo. Zasilany Hub może wystawić te 500mA, ale niezasilany już tylko 100mA*
- USB 2.0 ma parę przewodów ekranowanych.  
*Taki upgrade.*
- W kodowaniu NZR wstawia się dodatkowe bity synchroniczne.  
*Dodatkowe bity synchroniczne wstawia się w kodowaniu NRZI*
- Urządzenie USB 2.0 może zasygnalizować swoją niegotowość do zapisu danych z szybkością High-Speed wysyłając pakiet PING-NYET.  
*Wychodzi na to, że niegotowość zgłasza samym NYET? Pyta – PING, odpowiada (niegotowość) NYET. I tak cały czas, chyba że dostanie ACK. ACK – wykonanie transakcji OUT. NYET – host kontynuuje wysyłanie zapytań PING*
- W systemie deskryptorów urządzenia USB może wystąpić kilka deskryptorów urządzenia, konfiguracji, interfejsów i punktów końcowych.  
*Deskryptor urządzenia może być jeden. Innych – konfiguracji, interfejsu, końcowych może być więcej.*
- Hub USB ma przerwaniowy punkt końcowy, który wykorzystuje do powiadomiania hosta o podłączeniu urządzenia USB do któregoś z jego portów dolnych.  
*Chyba.*  
*Tak, jest to tzw. Hub change endpoint (punkt końcowy "zmiany w hubie")*
- Na wierzchołku wielopoziomowego, hierarchicznego układu deskryptorów USB znajduje się deskryptor konfiguracji. Na szczycie znajduje się pojedynczy deskryptor urządzenia.  
*Na szczycie znajduje się pojedynczy deskryptor urządzenia.*

- Transfer masowy I izochroniczny USB 1.1 są przykładami transferów aperiodycznych z zagwarantowanym pasmem w ramach jednej mikroramki.  
*Izochroniczny jest periodyczny, masowy nie ma zagwarantowanego pasa (wg tabelki z prędkościami)*
- W deskryptorze konfiguracji USB jest jakiś pole statusowe, które mówi o maksymalnym poborze prądu. Dla wartości 50 urządzenie pobiera 50mA.  
*Pole to jest tak skonstruowane, żeby wartość zmieściła się w jednym bajcie, ze skokiem co 2mA. Dlatego urządzenie, które zgłasza, że 50 może zasywać maksymalnie 100mA.*
- Uszeregowanie transakcji w USB nie zależy od implementacji kontrolera.  
*W OHC przerwaniowe są w strukturze drzewa, a w UCH listy podwieszanej, co ma wpływ na uszeregowanie (do sprawdzenia).*
- Host może zasygnalizować chęć zapisu danych do urządzenia wysyłając pakiet NYET do urządzenia USB 2.0, które z kolei odpowiada pakietem PING jeśli jest gotowe do zapisu.  
*To host posyła PING - zapytanie, czy urządzenie jest gotowe do zapisu. Te odsyła ACK - gotowe, lub NYET - jeszcze nie.*

### 3 IEEE 1394 Firewire

#### Prawda/Fałsz

- Po resecie w systemie FireWire (IEEE1394) wykonywane są procedury TREEID i SELFID .  
*Po resecie następuje „TREEID” odpowiedzialne za ustalenie węzła głównego a później „SELFID” odpowiedzialne za rozesłanie adresów do poszczególnych portów.*
- Transakcja dzielona IEEE1394 umożliwia wykorzystanie magistrali przez inne węzły po zakończeniu subakcji żądania (request), a przed wysłaniem odpowiedzi (response).  
*Tak, pomiędzy żądaniem a odpowiedzą magistrala jest wolna i można ją wykorzystać*
- Transakcje asynchroniczne IEEE1394 są uprzywilejowane w stosunku do transakcji izochronicznych i w związku z tym zawsze wykonywane są na początku cyklu.  
*Jest odwrotnie. To izochroniczne są uprzywilejowane nad asynchronicznymi.*
- Urządzenie IEEE 1394 będące konsumentem zasilania może posiadać co najwyżej jedno 6-kontaktowe gniazdo IEEE 1394.  
*Tak po prostu jest.*
- Numery węzłów IEEE 1394 nadawane są podczas procedury samoidentyfikacji na podstawie wartości wewnętrznych liczników odebranych pakietów SelfID.  
*Tak, SELFID przypisuje każdemu węzłowi unikatowy identyfikator pełniący rolę adresu, a następnie rozsyła je w formie pakietów selfid z każdego portu do wszystkich pozostałych węzłów.*
- Pakiet potwierdzenia odbioru asynchronicznego pakietu żądania zapisu bloku danych w pierwszej fazie transakcji asynchronicznej IEEE 1394 zawiera sumę kontrolną w formie parzystości.  
*Tak, zawiera kod potwierdzenia i parzystość - patrz budowa pakietów*
- W IEEE 1394 pakiet nowego cyklu (SCP) jest wysyłany ZAWSZE co 125us.  
*Izochroniczne owszem, ale asynchroniczne w ramach interwału równych szans, którego długość zależy od liczby węzłów asynchronicznych jednocześnie żądających dostępu do łącza.*
- IEEE 1394 posiada osobne pary ekranowanych przewodów (dla) TPA i TPB.  
*Tak wynika z przekroju budowy*
- Pole adresowe w IEEE1394 składa się z numeru magistrali (10b), numeru węzła (6b) i adresu w węźle (48b).  
*Wszystko się zgadza.*
- W systemie IEEE1394 węzeł A o szybkości S100 połączono z węzłem B o szybkości S200 za pośrednictwem węzła C o szybkości S400, co zwiększyło szybkość transmisji pomiędzy węzłami A i B w stosunku do ich połączenia bezpośredniego.  
*Nie, węzeł A wciąż nadaje z szybkością S100, a teraz dodatkowo musi przejść przez węzeł C*

- W interfejsie IEEE1394 przerwa pomiędzy subakcjami transakcji asynchronicznych jest mniejsza od przerwy pomiędzy transakcjami izochronicznymi, co powoduje ich uprzywilejowanie podczas arbitrażu.  
*Izochroniczne mają pierwszeństwo, to raz. Dodatkowo przerwa pomiędzy izochronicznymi zazwyczaj jest krótsza od tej pomiędzy asynchronicznymi.*
- Biorący udział w arbitrażu asynchronicznym węzeł A (dotyczy interfejsu IEEE 1394) nie może uzyskać dostępu do łącza, bo zdominował je asynchroniczny węzeł B, który ciągle wygrywa arbitraż.  
*Kolejność przydziału zależy od położenia węzła w drzewie systemu. Węzły położone bliżej korzenia uzyskują dostęp przed węzłami bardziej oddalonymi. Łączny czas wykonania subakcji przez wszystkie węzły nazywa się fairness interval. W tym czasie każdy węzeł uzyska dostęp do magistrali, aby wykonać subakcję. Dostęp do magistrali węzeł zarządzający przekazuje "rotacyjnie". Następna subakcja będzie mogła być wykonana w kolejnym interwale równych szans.*
- Odbiorca transakcji asynchronicznej wygrał arbitraż I odesłał pakiet odpowiedzi. Inicjator transakcji odebrał odpowiedź I oczekuje na wygranie arbitrażu w celu odesłania pakietu potwierdzenia. *Nie, istnieje coś takiego jak dostęp natychmiastowy - potwierdzenie wysyła się za pośrednictwem warstwy PHY, która nie rywalizuje o dostęp do magistrali.*
- W przypadku transakcji dołączanych (interfejs IEEE 1394) nie trzeba rywalizować o dostęp do magistrali w celu odesłania odpowiedzi.  
*Transakcja dołączana dostarcza dane przy potwierdzeniu, a ono nie wymaga arbitrażu.*
- W interfejsie IEEE 1394 szansa wygrania arbitrażu wzrasta wraz ze wzrostem odległości (mierzone liczbą węzłów pośredniczących) węzła ubiegającego się o dostęp do magistrali od korzenia.  
*Jest na odwrót – bliżej korzenia, większe szanse.*
- Wartość przerw i ograniczeń czasowych w interfejsie IEEE 1394 są “na sztywno” określone przez standard i nie mogą być korygowane.  
*Przerwa pomiędzy subakcjami transakcji asynchronicznej może być regulowana.*
- W przykładowym interfejsie IEEE 1394 występują tylko transakcje asynchroniczne. Jeżeli uczestnikiem (inicjatorem lub odbiorcą) transakcji asynchronicznej jest korzeń, to zawsze wygra on arbitraż jako pierwszy.  
*Zgodnie z zasadą "Bliżej korzenia, większe szanse" korzeń wygrywa wszystko. Jednak występują transakcje izochroniczne i asynchroniczne.*
- Kontroler cyklu musi być korzeniem w topologii IEEE1394, bo musi wysyłać sygnał CSP.  
*Chyba.*
- Węzeł w IEEE1394, który zainicjalizował transakcję asynchroniczną może być odbiorcą transakcji zainicjalizowanej przez inny węzeł w tym samym Interwale Równych Szans.  
*Raczej tak, węzły asynchroniczne do wykonania transakcji nie wymagają alokacji pasma (subakcja). Also - zarządzanie magistralą stara się umożliwić jednoczesne jej wykorzystanie przez różne transfery.*

## 4 IEEE-488 i SCPI

### Prawda/Fałsz

- GPIB (IEEE-488) jest interfejsem równoległym, opartym na 8-bitowej, 2 kierunkowej magistrali danych i 8 sygnałach sterujących: REN, IFC, ATN, SRQ, EOI, NRFD, NDAC, DAV  
*Tak, bity wysyła się ósemkami, stąd m. in. podaje się prędkość w bajtach na sekundę*
- SCPI to język programowania na bazie języka C wyposażony w biblioteki funkcji sterujących urządzeniami pomiarowo-kontrolnymi.  
*SCPI jest językiem kontroli urządzeo (i nie bazuje na C).*
- Znak ‘:’ w rozkazach SCPI reprezentuje przejście pomiędzy poziomami w rozgałęzionej strukturze subsystemu, natomiast prefiks ‘\*’ oznacza rozkaz wspólny.  
*Tak, dwukropki służą do precyzowania zapytania, gwiazdka jako nagłówek komunikatu wspólnego.*

- System statusowy urządzenia SCPI składa się tylko z jednego, 8-bitowego rejestru, w którym bit B6 jest zgłoszeniem żądania obsługi.  
*Składa się z minimum dwóch rejestrów, których układ jest wielopoziomowy (hierarchiczny).*
- Kontrola szeregową I kontrola równoległa to mechanizmy automatycznego wykrywania urządzeń podłączonych do systemu IEEE 488.  
*Kontrola szeregową I równoległą służą do identyfikacji urządzeń zgłaszających żądanie obsługi.*
- Maska związana z bajtem statusowym SCPI służy do blokowania ustawiania wybranych bitów bajtu statusowego.  
*maską związaną z bitem statusowym jest register maski żądania obsługi, który odpowiada za selekcję bitów powodujących zgłoszenie żądania, ale nie ma on wpływu na stan samych bitów w bajcie statusowym. Bajt statusowy jest rejestrem zbiorczym swoich rejestrów nadzorczych, więc jego wartość zależy od wartości tamtych rejestrów i ich masek.*

## 5 Inne

### Prawda/Fałsz

- Interfejsy USB, IEEE1394 oraz RS-232 udostępniają zasilanie systemowe i mają mechanizmy zarządzania zasilaniem.  
*USB i IEEE-1394 owszem, ale nie RS-232.*

## Opracowanie materiałów

## 1 RS-232 – szeregowy port znakowy

## 1.1 Co to jest?

Standard RS-232 został wprowadzony w 1962 r. w celu normalizacji interfejsu pomiędzy *urządzeniem końcowym dla danych* (DTE - Data Terminal Equipment), a *urządzeniem komunikacyjnym* (DCE - Data Communication Equipment). Na zajęciach zajmujemy się tak naprawdę zrewindowaną wersją standardu: RS-232C, wprowadzoną w 1969 roku.

RS-232C umożliwia przesył danych na niewielkie odległości - do 15 metrów - oraz niewielką szybkość - do 20 kb/s - przez niesymetryczne łącze.

## 1.2 Charakterystyka interfejsu RS-232

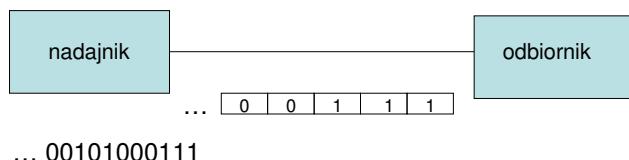
Łącze szeregowe przeznaczone do asynchronicznej transmisji znakowej realizowanej zazwyczaj w trybie półdupleksowym.

### 1.2.1 Transmisja danych

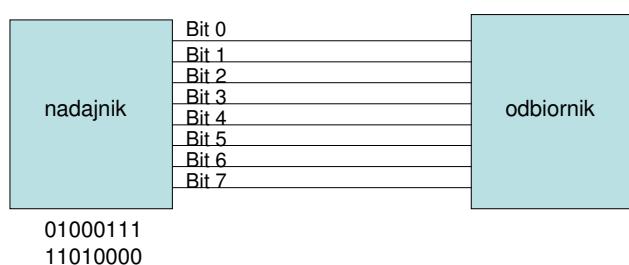
Szeregową asynchroniczną transmisję znakową w trybie półduplexowym (praktycznie tylko w tym trybie, ale mogą być inne). CIEKAWOSTKA: ma budowę duplexową.

### 1.2.2 Rodzaje transmisi

- Szeregową - sekwenncyjne przesyłanie bitów w ustalonej kolejności (od LSD lub MSB) po jednej linii transmisyjnej.



- Równoległa - przesyłanie bitów słowa po przyporządkowanej każdemu bitowi linii transmisyjnej (bity przesyłane równolegle, słowa przesyłane szeregowo).

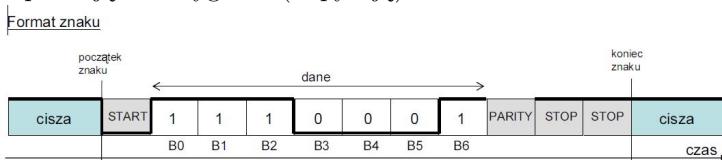


### 1.2.3 Definicje danych

- "0" - 0V
- "1" - 12V
- Czas transmisji jednego bitu  $T$  - stały, nie większy niż czas propagacji.
- $\frac{1}{T}$  - liczba bitów przesyłana w jednostce czasu. Standardowe wartości 110, 150, 300, 600, 1200, 2400 ... 19200 [b/s]
- Impulsy rozeznające - sprawdzają stan bitów odebranych (następują co  $T$ , które narzuca nadawca).

### 1.2.4 Jednostka informacyjna - znak

Jednostka informacji o ścisłe określonym formacie. Odbiorca dysponuje impulsami próbującymi, które rozpoznają stan sygnału (odpytują).



### 1.2.5 Przekazywanie konfiguracji

Aby nadawca i odbiorca mogli się porozumieć i interpretować znaki w ten sam sposób, muszą zostać tak samo skonfigurowane. Innymi słowy, muszą posiadać ten sam takt nadawania.

Metody:

- dodatkowe łącze
- jako element konfiguracji (wykorzystanie generatorów kwarcowych, synchronizm częstotliwościowy)

Nominalne położenie impulsu = ok.  $\frac{1}{2} \times T$  - pośrodku, największe bezpieczeństwo próbkowania. Służą do tego układy korekcji fazy impulsu - liczniki, zliczające liczbę impulsów na wejściu i dają 1 na wyjściu.

### 1.2.6 Definicja znaku

- **START** - bit kontrolny, znacznik początku (SOF - Start Of Frame) - jałowy z punktu widzenia przesyłanej informacji i służący jedynie w celu synchronizacji. START zapewnia  $\frac{n}{2}$  jako stan licznika (fazy impulsu).
- **DANE** - 7-8 bitów (kiedyś też 5-6, obecnie już nieużywane), które są treścią znaku, począwszy od bitu najmniej znaczącego (LSB - least significant bit). Tym bitem jest B0.
- **PARITY** – bit kontroli poprawności znaku, służy jako zabezpieczenie informacji. Może, ale nie musi występować. Jednak decyzja o jego występowaniu ma charakter globalny - dotyczy każdego znaku w danej transmisji. Jego stan określa zasadę:
  - Kontrola parzystości (*Even parity*) - polega na sprawdzeniu liczby jedynek na polu danych i ustawieniu bitu kontrolnego na "1" w przypadku nieparzystej liczby jedynek lub na "0" w przypadku parzystej (uzupełnienie do parzystości).
  - Kontrola nieparzystości (*Odd parity*) - polega na sprawdzeniu liczby zer na polu danych i ustawieniu bitu kontrolnego na "1" w przypadku nieparzystej liczby zer lub na "0" w przeciwnym przypadku.
  - Brak kontroli (*None*)

Ten bit kontroli pozwala wykryć przekłamanie w transmisji danych pod warunkiem, że liczba przekłamań jest nieparzysta.

- **STOP** – 1 lub 2 bity kontrolne, znacznik końca znaku.

### 1.2.7 Konwencja nazewnicza rodzajów transmisji

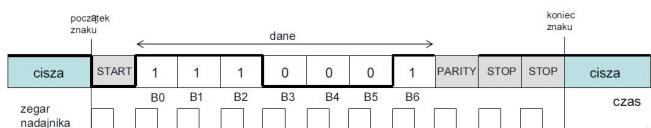
[Ilość bitów danych][Rodzaj kontroli][Liczba bitów stopu]  
Przykłady:

- 7E2 - 7 bitów danych, kontrola parzystości, 2 bity stopu (10 bitów + START = 11 bitów)
- 8O1 - 8 bitów danych, kontrola nieparzystości, 1 bit stopu (11 bitów)
- 8N2 - 8 bitów danych, brak kontroli, 2 bity stopu (11 bitów)

### 1.2.8 Rodzaje transmisji

- **Synchroniczna** - elementy informacji wysypane w takt zegara nadajnika. W ten sposób przesyłane są bity w ramach pojedynczej jednostki informacyjnej.
- **Asynchroniczna** - wysyłanie elementów informacji niesynchronizowane zegarem nadajnika. W ten sposób są wysyłane poszczególne jednostki - ich wprowadzanie nie jest sygnalizowany żadnym sygnałem, więc odstęp między nimi jest dowolny.  
Czas trwania bitu nazywa się *odstępem jednostkowym* i oznaczamy go  $t_b$ . Jego odwrotność ( $f = \frac{1}{t_b}$ ) określa szybkość transmisji w bodach, gdzie 1 [bd] = 1 [bit/s].

### 1.2.9 Transmisja w RS-232



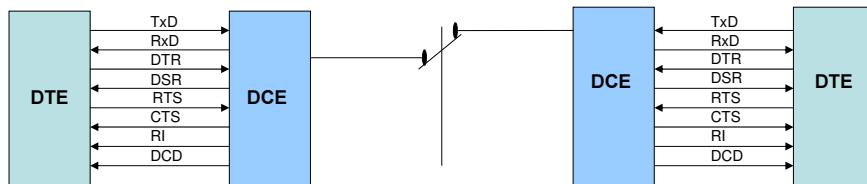
- Synchroniczne wysyłanie bitów
- Asynchroniczne wysyłanie znaków
  - Polega na wysyłaniu pojedynczych znaków, które mają ścisłe określony format.
  - Brak sygnału zegarowego określającego momenty wysyłania znaków.
  - Odstępy między znakami nieokreślone.

### 1.2.10 Tryby transmisji

- **Simpleksowa** - jednokierunkowa, z nadajnika do odbiornika.
- **Półduplekksowa (HDX)** - dwukierunkowa, niejednoczesna (w danej chwili czasu jedno urządzenie jest nadajnikiem, a drugie odbiornikiem). Zakłada istnienie tylko jednej linii transmisyjnej. Wymaga konfiguracji (informacja, kto kiedy nadaje).
- **Dupleksowa (FDX)** - dwukierunkowa, jednocożesna (w danej chwili czasu oba urządzenia mogą spełniać rolę nadajnika lub odbiornika). Brak konieczności sprawdzania czy łącze jest wolne oraz mechanizmu rezerwacji łącza.

### 1.3 Komunikacja DTE-DCE - sygnały w porcie RS-232

Komunikacja dwóch stacji DTE przez komutowane łącze telefoniczne.



Urządzenia				
DTE	Data Terminal Equipment	Komputer		
DCE	Data Communication Equipment	Modem		
Linie (sygnały)				
Skrót	Nazwa	Znaczenie	Przeznaczenie	Kierunek
TxD	Transmitted Data	Dane nadawane	Linia danych	Wyjście
RxD	Received Data	Dane odbierane	Linia danych	Wejście
DTR	Data Terminal Ready	Gotowość DTE	Linia kontrolna	Wyjście
DSR	Data Set Ready	Gotowość DCE	Linia kontrolna	Wejście
RTS	Request to Send	Żądanie nadawania	Linia kontrolna	Wyjście
CTS	Clear To Send	Zgoda na nadawanie	Linia kontrolna	Wejście
RI	Ring Indicator	Wskaźnik wywołania	Linia kontrolna	Wejście
DCD	Data Carrier Detected	Wykrycie nośnej	Linia kontrolna	Wejście
SG	Signal Ground	Masa sygnałowa	Masa	—

#### 1.3.1 Fazy pracy układu

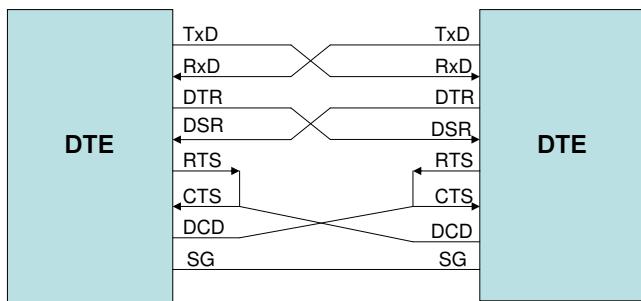
- Tryb nawiązywania połączenia
- Tryb transmisji danych (wtedy nas interesują dupleksy i inne)

#### 1.3.2 Linie w złączu RS-232

- Linie danych: TxD, RxD
- Linie kontrolne: DTR, DSR, RTS, CTS, RI, DCD

## 1.4 Połączenie bezmodemowe DTE-DTE

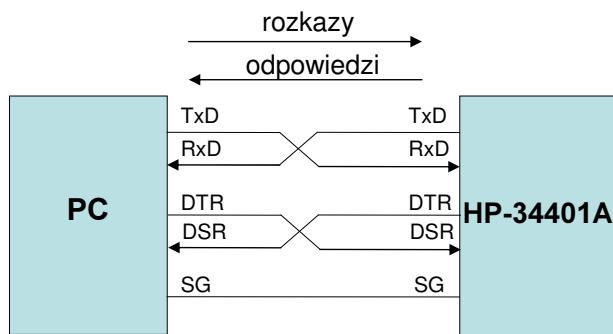
Przykład połączenia dla transmisji dupleksowej.



- PG, SG - masa
- TxD, RxD - dane
- RTS, CTS, DCD, DSR, DTR - sterowanie

## 1.5 Kontrola transmisji: handshake i protokół XON/XOFF

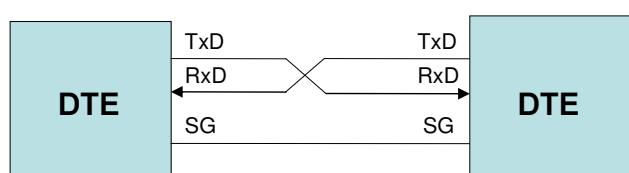
### 1.5.1 Handshake



- DTR = 1 - zgoda na nadawanie
- DTR = 0 - brak zgody na nadawanie
- DTR informuje, czy bufor jest zapełniony. DSR sprawdza go u partnera przed wysłaniem dalszych danych.
- Analogiczna sytuacja, kiedy podłączone są RTS i CTS zamiast DTR i DSR. RTS wystawia informację, CTS sprawdza.

### 1.5.2 Protokół XON/XOFF

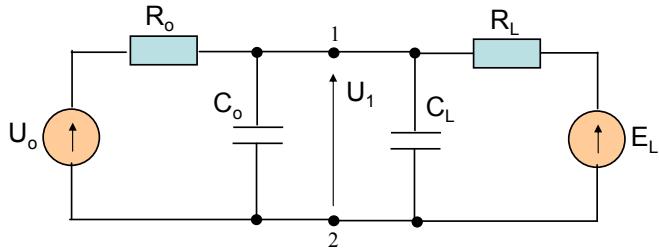
Występuje przy wymianie informacji w trybie dupleksowym. Umożliwia blokowanie i odblokowywanie transmisji danych. Np. drukarka - gdy skończy się papier w trakcie drukowania, przesył jest blokowany. Gdy użytkownik uzupełni papier, transmisja jest wznowiana. Taki protokół XON/OFF nazywany jest programowym (software XON/OFF). Rozwiążanie hardware to transmisja półduplexowa za pośrednictwem sygnałów w kanale wtórnym.



- XON – ASCII 19 (CTRL-S)
- XOFF – ASCII 17 (CTRL-Q)

## 1.6 Parametry elektryczne sygnałów

Poniżej przedstawiono schemat "obwodu stykowego" złożonego ze źródła sygnału, toru transmisyjnego i odbiornika. Parametry zdefiniowano przy założeniu, że szybkość transmisji nie przekracza 20 kbd.



Model obwodu transmisyjnego

### Poziomy sygnałów

$ U_o  < 25 \text{ V}$	Sygnały danych:
$3 \text{ k} \Omega < R_o < 7 \text{ k} \Omega$	-15 V < U <sub>1</sub> < -3 V 1 logiczna
$I_{zwarcia} < 0,5 \text{ A}$	+3 V < U <sub>1</sub> < +15 V 0 logiczne
$ E_L  < 2 \text{ V}$	
$C_o + C_L < 2500 \text{ pF}$	Sygnały sterujące:
Zmiana U <sub>1</sub> < 30 V/us	-15 V < U <sub>1</sub> < -3 V 0 logiczne
Jeden nadajnik - 1 odbiornik	+3 V < U <sub>1</sub> < +15 V 1 logiczna

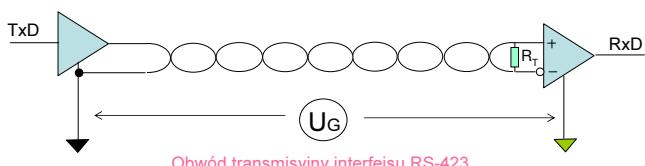
Na rysunku powyżej: kiloomy oraz mikrosekundy.

Wada: jest to obwód represyjny, da się go silnie zakłócić poprzez różnicę potencjałów pomiędzy masami.

## 1.7 Standardy RS-423, RS-422, RS-485

Niesymetryczna przesyłanie danych w RS-232C ogranicza szybkość i odległość transmisji, a ponadto nie jest zabezpieczone przed zakłóceniami zewnętrznymi. Aby to polepszyć wymyślono inne standardy.

### 1.7.1 RS-423A



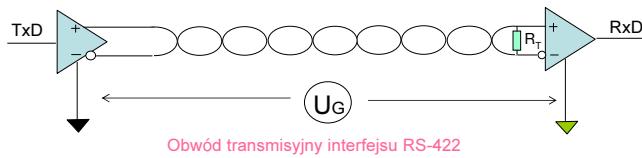
- szybkość do 100 kbd (przy zasięgu do 30 m)
- zasięg do 1200 m (przy szybkości do 3 kbd)

Standard RS-423A określa elektryczną charakterystykę napięciowego obwodu transmisyjnego złożonego z niesymetrycznego nadajnika oraz symetrycznego (różnicowego) odbiornika. Takie obwody stosuje się do przesyłania sygnałów binarnych pomiędzy DTE i DCE, które reprezentują dane lub funkcje sterujące. Zastosowanie różnicowego obciążenia pozwala na znaczne zmniejszenie wpływu napięcia wspólnego  $U_G$  powstającego na wskutek różnicy potencjałów masy nadajnika i odbiornika, jak również przesłuchów między nimi.

Standard wymaga aby dla każdego kierunku transmisji istniał przynajmniej jeden niezależny przewód powrotny.

Typowa prędkość wynosi 100 kb/s przy odległości do 30 m.

### 1.7.2 RS-422A



- szybkość do 10 Mbd (przy zasięgu do 100 m)
- zasięg do 1200 m (przy szybkości 100 kbd)

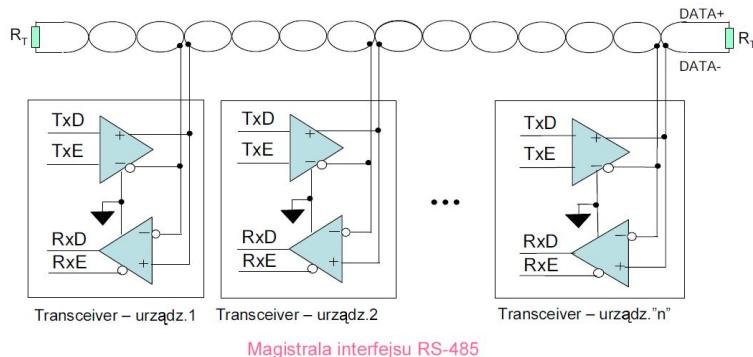
Wykorzystuje pełną symetryzację łącza, zapewnia szybką transmisję w obecności zakłóceń. Standardy RS-423 oraz RS-485 określają symetryczny, zrównoważony system transmisji danych złożony z:

- różnicowego nadajnika
- dwuprzewodowego zrównoważonego toru przesyłowego
- odbiornika o różnicowym obwodzie wejściowym.

Standard RS-422A nie wprowadza ograniczeń na minimalną i maksymalną częstotliwość, a jedynie na zależność między szybkością zmian sygnału, a czasem trwania bitu.

### 1.7.3 RS-485A

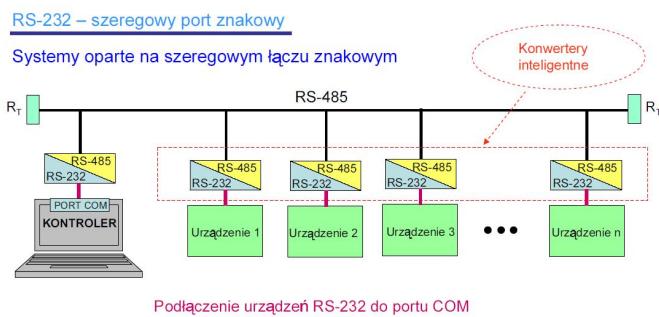
Standard RS-485A jest rozwinięciem RS-422. Łącze RS-485A jest również zrównoważone i symetryczne, przy czym dopuszcza się nie tylko wiele odbiorników, ale i wiele nadajników podłączonych do jednej linii. Nadajniki muszą być trójstanowe.



## 1.8 Systemy komunikacyjne oparte na łączu znakowym

### 1.8.1 System oparty na szeregowym łączu znakowym

Podłączenie urządzenia RS-232 do portu COM z pom. int. RS-485



$R_T$  - Rezystory zabezpieczające przed niekorzystnym odbiciem fali (tzw. Terminatory).

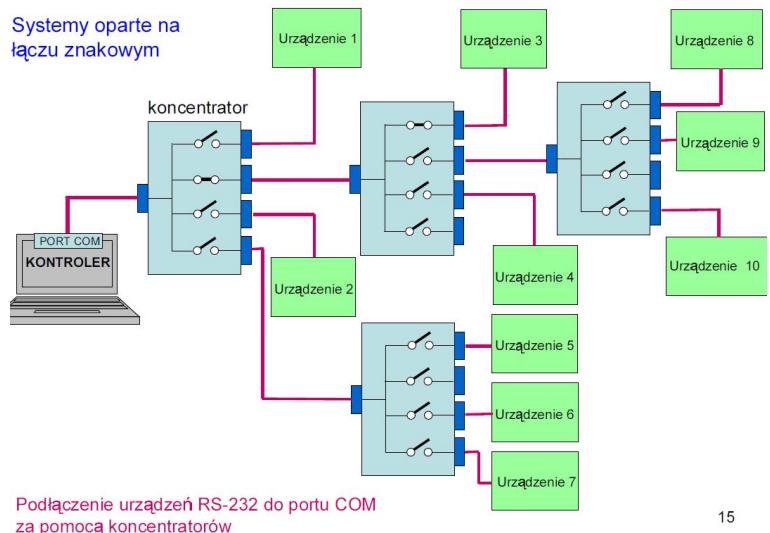
**Problem:** dostęp do magistrali kontrolera i urządzeń systemu

**Rozwiążanie:** Implementacja protokołu komunikacyjnego (warstwa łącza danych). Komputer zarządza innymi urządzeniami w całym systemie. Do zbudowania tego wystarczają proste przejściówka do zmian sygnałów.

#### Komunikacja:

- Selekcja urządzenia kontrolującego (*master*) - generuje on rozgłoszenie (*broadcast*) do wszystkich urządzeń i zbiera dane.
- Selekcja urządzenia odbierającego - konieczna gdy wiele urządzeń chce przesłać odpowiedź do mastera, co może powodować konflikt.
  - nadanie identyfikatorów (adresacja urządzeń)
  - zastosowanie przejściówek - są inteligentne i odpowiadają za dostęp do urządzenia.

### 1.8.2 System oparty na łączu znakowym



Koncentrator zawiera 4 klucze portu RS. Dostęp jest tylko do jednego wyjścia naraz.

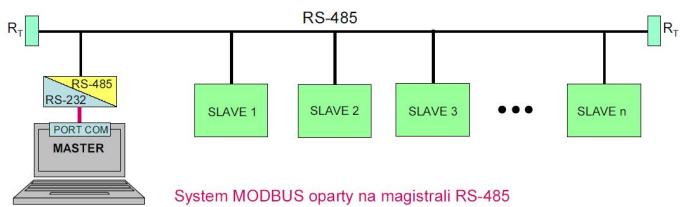
**Kaskadowe połączenie** - przełącznik podłączony do przełącznika. Pojawia się problem wyboru drogi do urządzenia, która musi być znana. Koncentratory muszą mieć informacje o **mapie urządzeń**.

### 1.9 System MODBUS

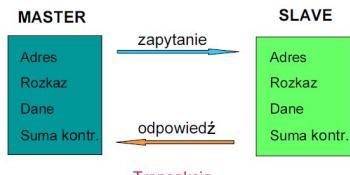
Interfejs MODBUS został opracowany w firmie Modicon i jest przyjętym standardem dla asynchronicznej, znakowej wymiany informacji pomiędzy urządzeniami systemów pomiarowo-kontrolnych.

#### 1.9.1 Charakterystyka

- Reguła dostępu do łącza na zasadzie Master-Slave.
- Zabezpieczenie przesyłanych komunikatów przed błędami
- Potwierdzenie wykonania rozkazów zdalnych i sygnalizacja błędów
- Mechanizmy zabezpieczające przed zawieszeniem systemu
- Wykorzystanie asynchronicznej transmisji znakowej zgodnej z RS-232C



System MODBUS oparty na magistrali RS-485



16

### 1.9.2 Transakcja

Jedno urządzenie może inicjować transakcje (*master*), a pozostałe (*slave*) odpowiadają jedynie na zapytania mastera. Transakcja składa się z polecenia (*query*) wysyłanego z master do slave oraz z odpowiedzi (*response*) przesyłanej ze slave do master. Odpowiedź zawiera dane żądane przez master lub potwierdzenie realizacji jego połączenia. Wykrycie końca kończy fazę w której następuje przekazanie łącza masterowi.

### 1.9.3 Format wiadomości

Dotyczy zarówno polecień jednostki nadzorcowej, jak i odpowiedzi podległych.

- Adres
- Kod funkcji reprezentujący rozkaz, pierwszy bit rozkazu oznacza jego rodzaj. 0 - normalny, 1 - szczególny.
- Dane
- Kontrola błędów (dla pracy w warunkach przemysłowych)

W przypadku odpowiedzi odpowiednio w polach znajdują się:

- Adres (swój, slave'a, do kontroli poprawności)
- Pole potwierdzenia realizacji rozkazu
- Dane żądane przez master
- Kontrola błędów

### 1.9.4 Rodzaje transakcji

- **Adresowana** - przeznaczona dla pojedynczej jednostki slave
- **Rozgłoszeniowa (broadcast)** - wysyłana do wszystkich jednostek podległych. Na ten rodzaj polecenia jednostki nie przesyłają odpowiedzi.

### 1.9.5 Rodzaje odpowiedzi

- **Normalna** - w przypadku poprawnego wykonania polecenia.
- **Szczególna** - jeżeli slave wykryje błąd przy odbiorze wiadomości lub nie jest w stanie wykonać polecenia, to przygotowuje specjalny komunikat o wystąpieniu błędu i przesyła jako odpowiedź. W przypadku tej wiadomości jest ona **powiększona** o 128 - miejsce na kod błędu.

### 1.9.6 Parametry protokołu

- Reguła dostępu do łącza: Master-Slave
- Zakres adresów: 1 - 247 (identyfikatory slave'ów)
- Adres rozgłoszeniowy: 0, rozpoznawany przez wszystkie slave'y
- Kontrola błędów: LRC/CRC, ograniczenie czasowe odpowiedzi
- Wymagana ciągłość przesyłania znaków w ramce

### 1.9.7 Ramka w systemie MODBUS

W systemie MODBUS wiadomości są zorganizowane w ramki o określonym początku i końcu. Umożliwia do odbiornikowi odrzucenie ramek niekompletnych i sygnalizację błędów.

### 1.9.8 Rodzaje transmisji ramek

- ASCII
- RTU

### 1.9.9 Ramka w trybie ASCII

Każdy bajt wiadomości przesyłany jest w postaci dwóch znaków ASCII. Zaletą tego rozwiązania jest to, że pozwala na długie odstępy między znakami (1 s) bez powodowania błędów.

#### Format znaku

Format znaku						
SOF	EOF					
”:”	ADRES 1 bajt	ROZKAZ 1 bajt	DANE n bajtów	LRC 1 bajt	CR	LF
1 znak	2 znaki	2 znaki	2n znaków	2 znaki	1 znak	1 znak

- System kodowania: heksadecymalny, znaki ASCII 0-9, A-F. Jeden znak heksadecymalny zawarty jest w każdym znaku ASCII wiadomości.
- Jednostka informacyjna: ograniczona znakami start (na początku) i stop (na końcu), 10-bitowa.
- Znacznikiem początku ramki jest znak dwukropka (”:” - ASCII 3Ah).
- Dopuszczalne znaki dla pozostałych pól (poza znacznikiem końca ramki) to 0-9, A-F.
- Pole funkcji: dwa znaki w trybie ASCII
- Podsumowując: wykorzystujemy 2 znaki heksadecymalne do przesyłu 1go znaku ASCII. Dzielimy ten na dwie części i przesyłamy w dwóch pakietach po 10 bitów.
- Urządzenie po wykryciu znacznika początku sprawdza czy pole adresowe zawiera jego własny adres. Jeżeli tak, to odczytuje zawartość pola funkcji i pola danych.
- Pole kontrolne LRC (*Longitudinal Redundancy Check*), 1-bajtowe, zabezpiecza część informacyjną. Sumuje część informacyjną bajtu i uzupełnia do 2.
- Ramka kończy się przesłaniem dwóch znaków: CR i LF.
- Ramki muszą być przesyłane w postaci ciągłej, tzn. odstęp między kolejnymi znakami tworzącymi ramkę nie może być większy niż 1 s. W przeciwnym razie urządzenie sygnalizuje błąd.

Stosowane jest zabezpieczenie części informacyjnej ramki kodem LRC .

### 1.9.10 Ramka w trybie RTU

SOF	ADRES 1 bajt	ROZKAZ 1 bajt	DANE n bajtów	CRC 2 bajty	EOF
$\geq 4 \times T$	1 znak	1 znak	n znaków	2 znaki	$\geq 4 \times T$

W trybie RTU wiadomości zaczynają się odstępem czasowym trwającym minimum  $3.5 \times (\text{czas trwania pojedynczego znaku})$ , w którym panuje cisza na łączu (można to zrealizować np. przez odmierzanie czasu trwania znaku przy zadanej na łączu szybkości bodowej).

- Pierwszym polem informacyjnym jest adres urządzenia
- Dopuszczalne znaki w ramach pól ramki: 0-9, A-F
- Zakres kodów operacji: 1 - 255
- Urządzenia stale monitorują magistralę. Jak adres odebrany w wiadomości zgadza się z ich właściwym, to lecą dalej.
- Ramkę kończy przerwa czasowa trwająca co najmniej  $3.5 \times (\text{długość znaku})$
- W przypadku gdy nowa wiadomość pojawia się przed upływem niezbędnej przerwy to będzie ona potraktowana jako kontynuacja poprzedniej wiadomości. Doprowadzi to do błędu sumy kontrolnej.
- Ramki muszą być przesyłane w postaci ciągłej, tzn. odstęp między kolejnymi znakami tworzącymi ramkę nie może być większy niż  $1.5 \times (\text{długość znaku})$ .
- Przekroczenie odstępu powoduje uznanie ramki za niekompletną i błędna.
- Kontrola danych typu CRC - 2-bajtowe, silniejsze niż LRC.

### 1.9.11 Warstwa fizyczna

- Asynchroniczna transmisja znakowa
- Formaty znaków
  - Tryb ASCII: 7E1, 7O1, 7N2
  - Tryb RTU: 8E1, 8O1, 8N2
- Szybkość: od 1200 bd do 19200 bd
- Rodzaj łącza:
  - Magistrala RS-485
  - Multipleksowany RS-232
- Rodzaj transmisji (zależny od łącza):
  - różnicowa dla RS-485
  - odniesiona do masy dla RS-232

## 2 USB – Uniwersalny interfejs szeregowy

### 2.1 Zalety

- Podłączenie dużej liczby różnorodnych urządzeń
- Automatyczne wykrywanie włączenia urządzenia do systemu oraz jego odłączenia
- Przeprowadzanie wszystkich operacji konfiguracyjnych, w tym instalacji sterownika, bez udziału użytkownika
- Szybka transmisja
- Zasilanie urządzenia bezpośrednio z portu

### 2.2 Parametry

Dla USB 1.1 oraz 2.0

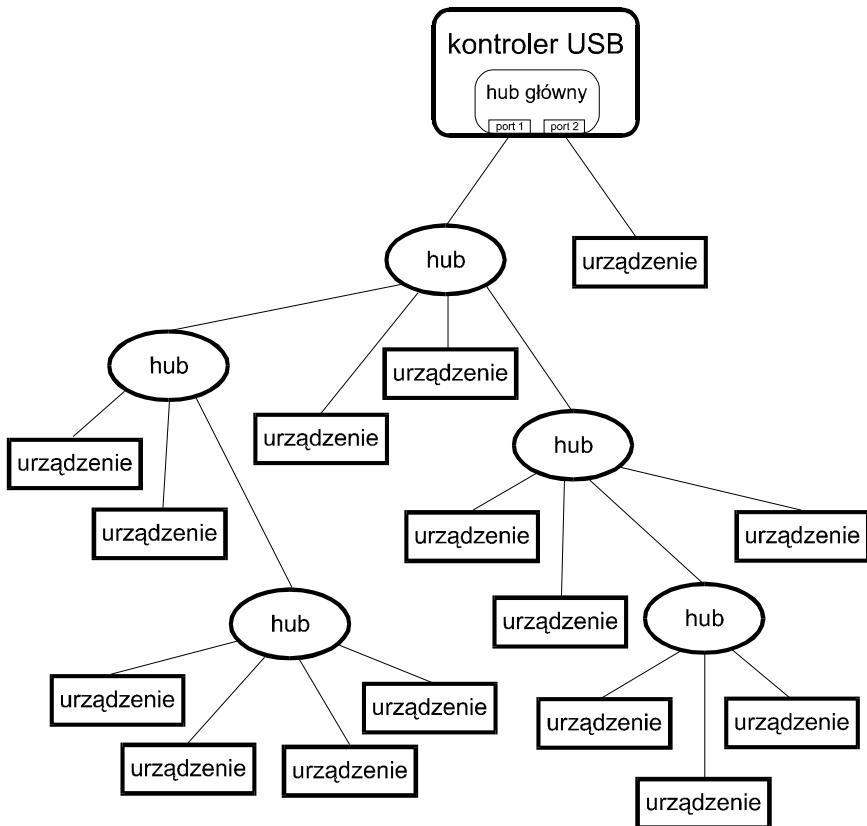
- Szybkość transmisji: 12 Mb/s (1.1), 480 Mb/s (2.0)
- Złożoność kontrolera w stacji host: 10000 bramek
- Złożoność kontrolera w urządzeniu: 1500 do 2000 bramek

### 2.3 Charakterystyka systemu USB

#### 2.3.1 Podstawowe właściwości interfejsu USB

- **Gorące podłączenie** - włączanie/wyłączanie urządzeń bez wyłączania systemu oraz konfiguracja bez udziału użytkownika.
- **Jeden typ złącza** - złącze typu A w koncentratorze i B w urządzeniu (dwa kontakty linii danych oraz dwa zasilania).
- **Duża liczba podłączanych urządzeń** - moduły komunikacyjne, jak hub czy koncentrator, posiadają kilka portów USB oraz umożliwiają łączenie kaskadowe w celu rozszerzenia liczby portów do podłączanie urządzeń. W ten sposób można uzyskać wielopoziomową gwiazdzistą strukturę. Ostatecznie do USB można podłączyć maksymalnie **127** urządzeń.
- **Różne szybkości transmisji**
  - Mała (*low speed*): 1,5 Mb/s
  - Pełna (*full speed*): 12 Mb/s
  - Wysoka (*high speed*): 480 Mb/s
- **Zasilanie** - USB posiada własny system dystrybucji zasilania: minimalnie dostarcza 4.7 V/100 mA (zagwarantowane przez magistralę), maksymalnie 500 mA (przy zasilaniu zewnętrznym lub hybrydowym). Przy braku aktywności na magistrali przez 3s urządzenie przechodzi w tryb zmniejszonego poboru prądu (SUSPEND), gdzie mogą pobierać maksymalnie 500  $\mu$ A. Wyjście z tego trybu jest możliwe poprzez inicjatywę hosta bądź urządzenia (*remote wake-up*)
- **Protokół komunikacyjny, detekcja błędów** - obowiązuje złożony, pakietowy protokół k. z kontrolą poprawności przesyłu. Żądania przesłania danych (*transfer*) dzielone są na transakcje, które z kolei złożone są z pakietów: kontrolnego, danych i potwierdzenia. Pakiety są zabezpieczone sumą kontrolną. W przypadku wykrycia błędu jest możliwe powtórzenie transakcji.

- **Transfery USB** - 4 typy transferów:
  - Kontrolny (*control transfer*)
  - Masowy (*bulk transfer*)
  - Przerwaniowy (*interrupt transfer*)
  - Izochroniczny (*isochronous transfer*)
- **Niski koszt** - nieduża złożoność układów interfejsowych
- **Schemat blokowy systemu USB**



### 2.3.2 Środowisko sygnałowe

Łącze transmisyjne oparte jest na obwodzie różnicowym. Różnicowy nadajnik jest połączony z różnicowym odbiornikiem przez parę przewodów.

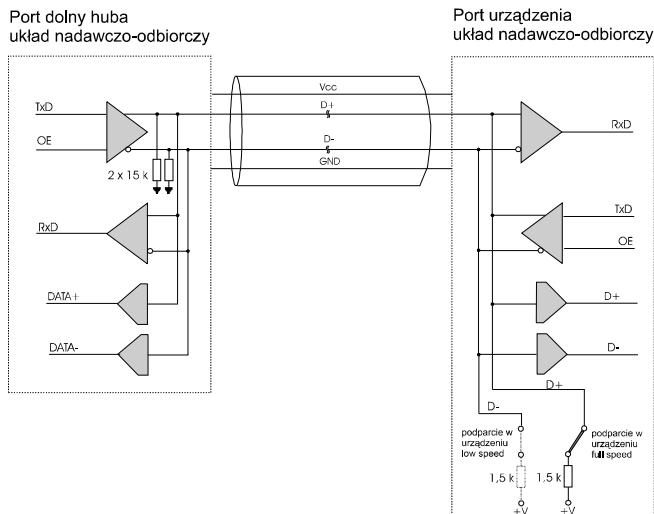
W przypadku transmisji wolnej wystarcza para nieskręcana i nieekranowana, natomiast transmisja pełna lub wysoka wymaga pary skręcanej i ekranowanej.

Zasilanie jest przekazywane przewodami  $V_{cc}(+5V)$  oraz  $GND$  (masa).

Trójstanowy nadajnik można zablokować sygnałem  $OE$ , co oznacza blokadę portu. Wówczas wyjście nie-odwracające ( $D+$ ) i odwracające ( $D-$ ) przy zablokowanym nadajniku zostają na potencjale masy (patrz rezystory 15k).

Wzmacniacze odniesione do masy monitorują napięcie na liniach  $D+$  i  $D-$ .

## Obwód transmisyjny w standardzie USB

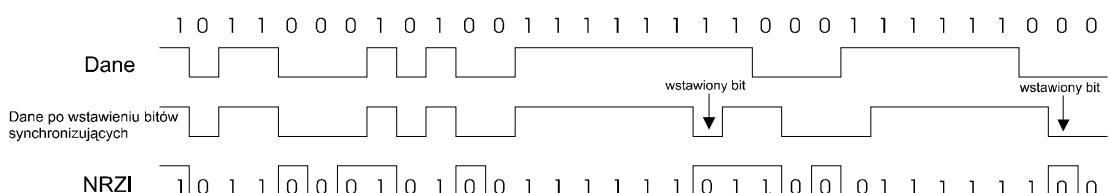


### Stany magistrali USB

- Logiczne „1” w obwodzie różnicowym  $(D+) - (D-) > 200 \text{ mV}$
- Logiczne „0” w obwodzie różnicowym  $(D+) - (D-) < -200 \text{ mV}$
- Plus 9 innych stanów

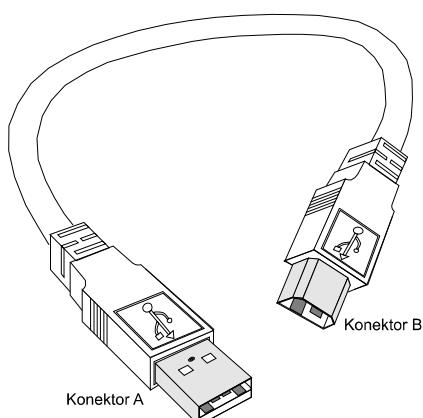
### Kodowanie bitów w systemie USB

W systemie USB przyjęto kodowanie NRZI (*Non Return to Zero Inverted*) ze wstawianiem bitu synchronizującego (tzw. *bit stuffing*). Na początku każdego bitu o wartości logicznej zero następuje zmiana sygnału, natomiast po każdym 6 kolejnych bitach o wartości jeden wstawiany jest bit o wartości zero. Umożliwia to prawidłowe poznawanie bitów w sygnale odebranym bez potrzeby przesyłania sygnału zegarowego nadajnika. Bitы wstawione są usuwane po rozpoznaniu stanu bitów w odebranym sygnale.



### 2.3.3 Środowisko fizyczne

#### Podstawowy kabel do podłączenia urządzenia USB



W standardzie USB wyróżnia się dwa rodzaje złączy

- Konektor A – strona portu dolnego (hub), „od strony urządzenia”
- Konektor B – strona portu górnego (urządzenie), „od strony hosta”

Złącza USB posiadają 4 zaciski: po dwa dla magistrali danych i zasilania. W samym złączu kontakty zasilania są nieco wysunięte w przód przed kontakty danych, a więc po włożeniu najpierw jest podłączone zasilanie, a potem dane.

Nr kontaktu	Nazwa sygnału	Kolor przewodu w kablu
1	Vcc	Czerwony
2	+DATA	Biały
3	-DATA	Zielony
4	GND	Czarny

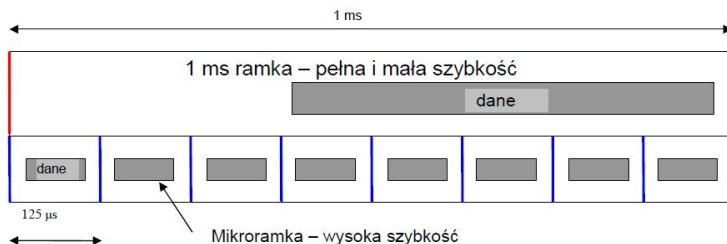
### 2.3.4 Rodzaje kabli

- Kabel przeznaczony dla urządzeń „low speed”
  - Nieekranowany
  - Zawiera dwie, nieskręcane pary przewodów: dla danych (28 AWG) i zasilania (20-28 AWG)
- Kabel przeznaczony dla urządzeń „full speed i high speed”
  - Zawiera ekranowaną parę skręcaną (28 AWG) dla danych
  - I nieekranowaną parę (20-28 AWG) dla zasilania
- Czas propagacji sygnału w kablu: < 30 ns przy przesyłce z częstotliwością 1-16 MHz. Wymagany dla urządzeń małej i pełnej szybkości. Czas propagacji wpływa na długosc kabla:
  - 5 m dla 6ns/m
  - 3 m dla 10ns/m

### 2.3.5 Ramki i mikro ramki

W systemie USB dane są przekazywane:

- W ramkach o czasie trwania 1 ms dla małej i pełnej szybkości transmisji.
- W mikroramkach o czasie trwania 125  $\mu$ s dla wysokiej szybkości transmisji.



#### Ramka 1 ms

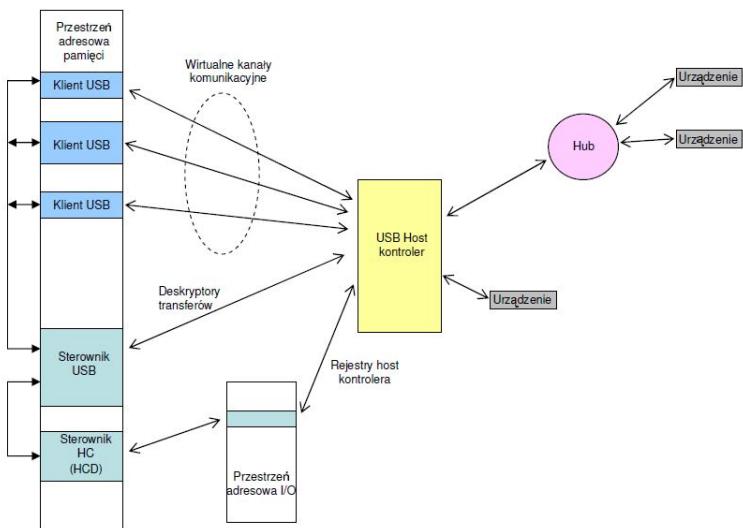
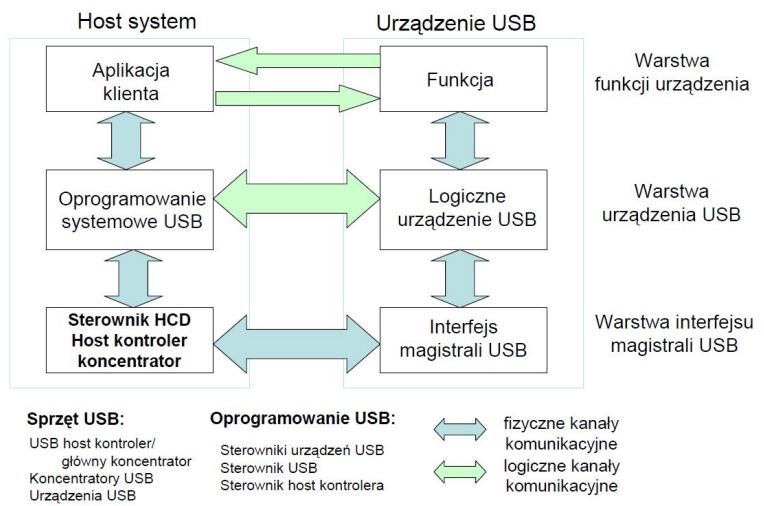
- Czas pomiędzy dwoma kolejnymi pakietami SOF (*Start Of Frame*) nazywany jest ramką. Początek ramki wyznacza pakiet SOF zawierający 11 bitów danych i 5 bitów CRC. Dane w pakiecie SOF reprezentują kolejny numer ramki. Licznik ramek przepelnia się co 2048 ms.
- Pełna szybkość transmisji
  - Maksymalny teoretyczny rozmiar ramki w bitach:  $1 \text{ ms} \times 1/\text{12 Mhz} = 12000 \text{ bitów}$
  - Powyższy rozmiar ramki w bajtach:  $12000 : 8 = 1500$ .
  - Pakiety kontrolne zajmują ok. 300 bajtów.
  - Praktyczna góra granica liczby transmitowanych w ramce: 1200 bajtów.
- Mała szybkość transmisji
  - Max rozmiar ramki w bitach:  $1 \text{ ms} \times 1/\text{1,5 Mhz} = 1500 \text{ bitów}$
  - Max rozmiar ramki w bajtach:  $1500 : 8 = 187$

### Mikroramka $125 \mu s$

- Czas pomiędzy dwoma kolejnymi pakietami SOF wysyłanymi przez high speed huba nazywa się mikroramką.
- Mikroramka trwa  $125 \mu s$
- Na 1 ramkę przypada 8 mikroramek
- Wysoka prędkość transmisji - szybkość transmisji w mikroramce wynosi 480 Mhz.

### 2.3.6 Model komunikacyjny

Warstwowy model komunikacyjny przedstawiający odpowiadające sobie elementy sprzętowo-programowe po stronie hosta i urządzenia.



### 2.3.7 Transfery USB

Wyróżniono 4 typy transferów przeznaczone dla różnych urządzeń z różnymi wymaganiami komunikacyjnymi. Każdy wymaga innego kanału komunikacyjnego (*pipe*).

- **Masowy** (*bulk transfer*) - time delivery accuracy + quality delivery accuracy ("jak najlepiej w miarę możliwości").

Przeznaczony do komunikacji z urządzeniami, do których zapisuje się lub z których odczytuje duże ilości danych, jednak czas przekazywania danych oraz regularność ich dostarczania nie są krytyczne. Ważna jest kontrola przekazywanych danych, błędy transmisji muszą zostać wykryte i skorygowane poprzez żadanie ponownego przesłania błędного bloku danych.

Nie ma zagwarantowanego pasma, transmisja ta jest realizowana, jeżeli inne transfery nie wykorzystają przydzielonych im zasobów.

Może być realizowany tylko z pełną lub wysoką szybkością.

Wymagane jest, aby pakiet danych miał stałą wartość (*MaxPacketSize*): 8, 16, 32 lub 64. Wszystkie kolejne pakiety danych muszą mieć tę stałą wartość z wyjątkiem ostatniego, który sygnalizuje koniec transferu. Przykłady: drukarka, skaner.

- **Przerwaniowy** (*interrupt transfer*) - quality delivery accuracy.

Do komunikacji z urządzeniami, do których zapisuje się lub z których odczytuje się niewielkie bloki danych, jednak trzeba robić to regularnie, w ustalonych odstępach czasowych.

Parametry transferu zawarte są w deskryptorze pobieranym na etapie konfiguracji urządzenia.

Ma zagwarantowane pasmo, może być wykorzystany do obsługi urządzeń low speed i full speed.

Rozmiar pakietu danych zależy od wybranej szybkości: dla low speed jest to 8, dla full speed to maksymalnie 64. Wszystkie kolejne pakiety danych muszą mieć tę stałą wartość z wyjątkiem ostatniego, który sygnalizuje koniec transferu.

Przykłady: mysz, klawiatura.

- **Izochroniczny** (*isochronous transfer*) - time delivery accuracy.

Przeznaczony do komunikacji z urządzeniami, do których zapisuje się lub z których odczytuje duże ilości danych, przy czym krytyczna jest regularność dostarczania danych.

Nie jest ważna kontrola przekazywania danych, nie jest dopuszczalne ponowne wysłanie bloku danych.

Dla komunikacji w pełnej lub wysokiej szybkości. Parametry transferu USB odczytuje z deskryptora urządzenia.

Graniczna wartość parametru *MaxPacketSize* to 1023 bajty na ramkę.

Przykład: odtwarzanie muzyki w aparaturze audio.

- **Kontrolny** (*control transfer*) - time delivery accuracy + quality delivery accuracy.

Musi być obsługiwany przez każde urządzenie USB, ponieważ jest przeznaczony do jego konfiguracji i nadzoru. W ramce zarezerwowano stosowane pasmo dla realizacji transferów kontrolnych (10% czasu trwania ramki).

Obsługuje wszystkie 3 rodzaje prędkości transferu.

Dzieli się na 3 etapy:

- Przekazanie rozkazu - 8-bitowe pole danych udostępnia informację o liczbie bajtów danych przesyłanych w etapie przekazani danych, o ile ten występuje.
- Przekazanie danych - pakiety danych mogą mieć rozmiar nie większy niż 64 bajty (parametr *MaxPacketSize*).
- Przekazanie statusu.

## Podział i podsumowanie transferów

Cechy i parametry	Kontrolny	Masowy	Przerwaniowy	Izochroniczny
Low speed (abajt/ms)	24 (3 transakcje 8 B)	Nie dozwolony	0,8 (8 B na 10 ms)	nie dozwolony
Full speed (abajt/ms)	832 (13 tr. 64 B na ramkę)	1216 (19 tr. 64 B na ramkę)	64 (1 tr. 64 B na ramkę)	1023 (1 tr. 1023 B na ramkę)
High speed (abajt/ms)	15872 (31 tr. 64 B na mikroramkę)	53248 (13 tr. 512 B na mikroramkę)	24576 (3 tr. 64 na mikroramkę)	24576 (3 tr. 64 B na mikroramkę)
Kierunek przepływu danych	Odczyt/zapis	Odczyt/zapis	Odczyt/zapis	Odczyt/zapis
Zarezerwowane pasmo dla transferów w ramach ramki	10% dla low i full, 20% dla high	Brak rezerwacji	90% dla low i full, 80% dla high	90% dla low i full, 80% dla high
Kontrola błędów transmisji	Tak	Tak	Tak	Nie
Komunikat/Strumień	Komunikat	Strumień	Strumień	Strumień
Określony rytm (periodyczność)	Aperiodyczny	Aperiodyczny	Periodyczny	Periodyczny

- Transfery strumieniowe (*stream transfer*), przekazywanie danych bajt po bajcie.
- Transfery komunikatowe (*message transfer*), struktura komunikatów i ich znaczenie określa standard.

### 2.3.8 Zarządzenie magistralą USB

Wszystkie transfery USB są inicjowane i kontrolowane przez hosta. Host usiłuje jednocześnie realizować transfery wielu urządzeń tak, aby jedno urządzenie nie blokowało dostępu.

Niektóre transfery mają priorytety. Transferom izochronicznym i przerwaniowym nadano 90% pasma, a kontrolnym 10%. Oznacza to, że transfer masowy nie jest gwarantowany - jeżeli pozostałe 3 rodzaje zajmą maksymalny możliwy czas, to transfer masowy się nie odbędzie.

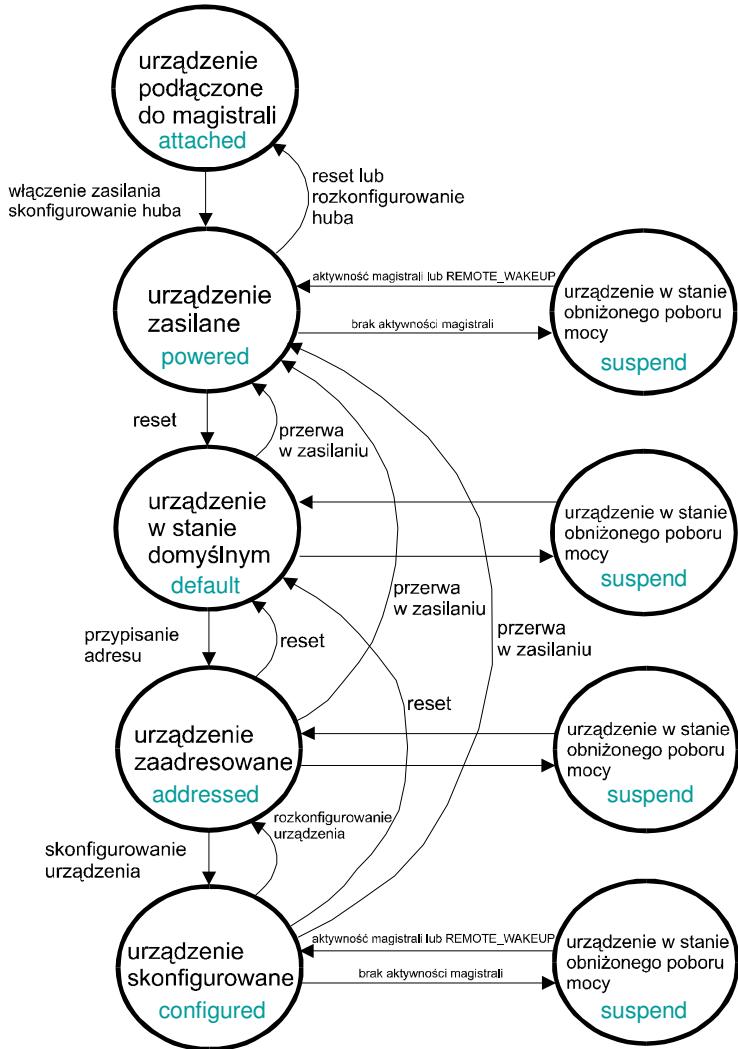
W praktyce jednak zdarza się to rzadko i 3 transfery nie wykorzystują pełnego pasma, co pozwala kontrolerowi USB na przydzielenie niewykorzystanego miejsca transferom masowym.

Kontroler dąży do jak najlepszego wykorzystania miejsca w ramce, w które może wstawić nawet wiele transakcji danego transferu masowego.

Działanie kontrolera oparte jest na zasadzie "usilnych starań" (*best efforts*), aby jak najsprawniej (w jak najkrótszym czasie) zrealizować wszystkie żądane transfery. Jednak jeżeli w ramce 1 ms brakuje miejsca na transferu masowego to musi on czekać.

### 2.3.9 Stany urządzenia USB

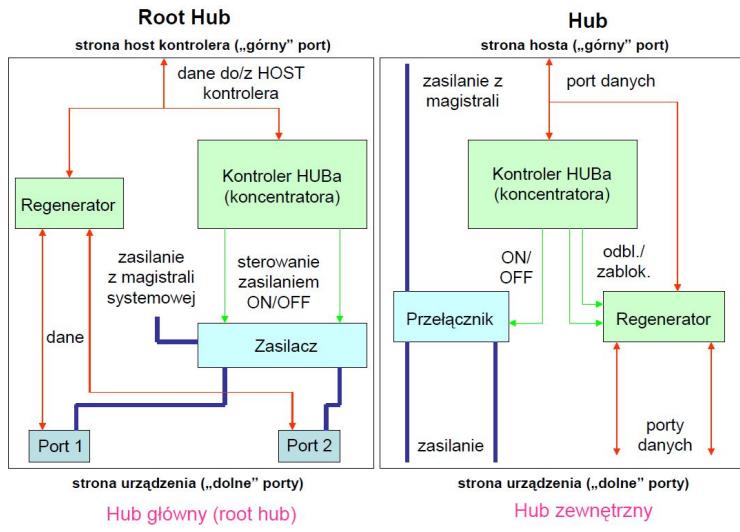
Zanim urządzenie USB zostanie skonfigurowane i udostępnione aplikacjom musi przejść przez kilka stanów przedstawionych na rysunku



### **2.3.10 Hub w systemie USB**

Huby to urządzenia pośredniczące w komunikacji z urządzeniami końcowymi.

- Zwiększą liczbę portów dostępnych dla urządzeń oraz możliwe jest ich kaskadowe łączenie.
  - Port huba od strony hosta nazywa się górnym (*upper stream port*), natomiast porty od strony urządzeń dolnym (*down stream port*).
  - Hub przeważnie posiada 4 porty dolne, rzadko kiedy inną liczbę jak np. 7.
  - Hub nie tylko przekazuje dane, ale pośredniczy również w dystrybucji zasilania do urządzeń.
  - Huby zgodne ze standardem 1.x są zdolne do komunikacji z małą lub pełną szybkością. Obowiązuje zasada, że do urządzeń low speed nie wolno kierować ruchu full speed. Hub potrafi wykryć jakie urządzenie jest podłączone do danego portu dolnego i wie, kiedy należy go zablokować.
  - Transfery low speed są poprzedzone specjalną preambułą, których zadaniem jest odblokowanie portów dolnych do których podłączone są urządzenia low speed.



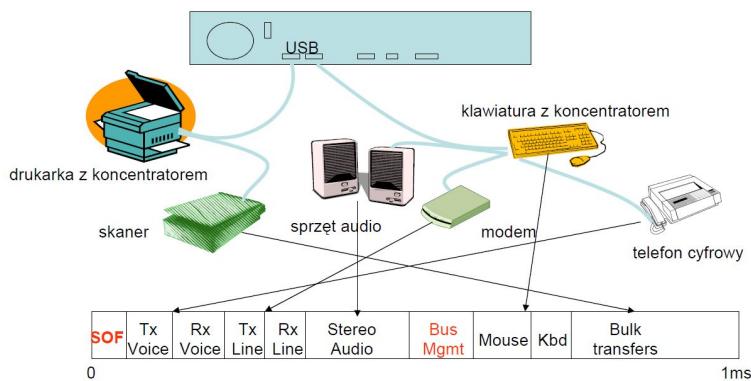
## 2.4 Protokół komunikacyjny

Transakcje USB zbudowane są z pakietów, przy czym typowa transakcja zawiera:

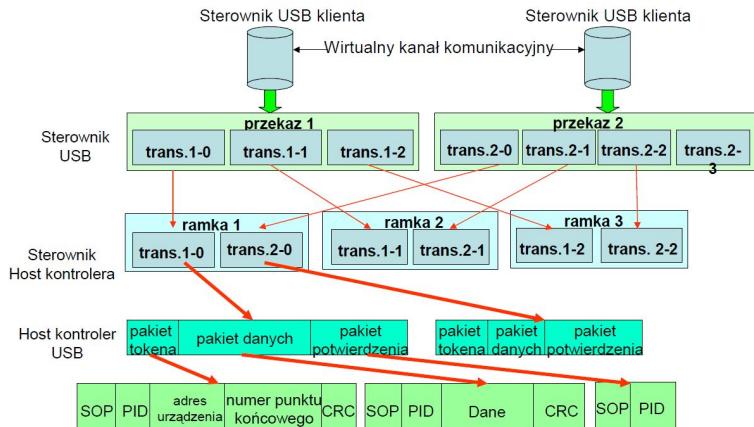
- Pakiet tokena (*token packet*) - jest elementem kontrolnym transakcji i informuje o jej rodzaju: kontrolna lub przeznaczona do zapisywania danych.
  - Pakiet danych (*data packet*) - pole przeznaczone dla danych zapisywanych do urządzenia lub odczytywanych z urządzenia.
  - Pakiet potwierdzenia (*handshake packet*) - informuje jednostkę nadającą o odebraniu danych lub instrukcji sterującej przez odbiorcę.

#### 2.4.1 Właściwości

- Komunikacja z urządzeniami USB za pomocą ramek o długości maksymalnie 1 ms.



- Pakietowa struktura komunikatów



Struktura bloków komunikacyjnych w USB (kolejkowanie odbywa się na poziomie sterownika host kontrolera)

#### 2.4.2 Format i typy pakietów USB

##### Format i typy pakietów USB

SOP Sekwencja synchronizacyjna	PID Identyfikator pakietu	Pole kontrolne identyfikator apakietu	Pole informacyjne pakietu	Suma kontrolna CRC zabezpieczająca pole informacyjne	EOP Znacznik końca pakietu
-----------------------------------	------------------------------	--	------------------------------	---	-------------------------------

Format pakietu

Typ pakietu	PID
Token In	1001b
Token Out	0001b
Token Setup	1101b
Token SOF	0101b
Data 0	0011b
Data 1	1011b
Handshake ACK	0010b
Handshake NAK	1010b
Handshake STALL	1110b
Preamble	0011b

Kody PID (USB 1.1)

18

##### Pakiety szczególne:

Stan jasowy	Sekwencja synchronizacyjna	PID preambuły (0011)	Pole kontroli PID (*100)	Cpoźnienie na odblokowanie portu	Sekwencja synchronizacyjna (początek pakietu low speed)
		pakiet preambuły przesyłany z pełną szybkością		początek pakietu przesyłanego z małą szybkością	

Pakiet preambuły poprzedzający pakiet przesyłany z małą szybkością

#### 2.4.3 Reakcja na błędy

Wykrywanie błędów i kontrola transmisji

- Kontrola poprawności pakietów (zabezpieczenie pola PID sumą CRC)

- W przypadku pakietu Token - 5-bitowa suma
- W przypadku pakietu Data - 16-bitowa suma

Wykrycie błędu powoduje odrzucenie pakietu.

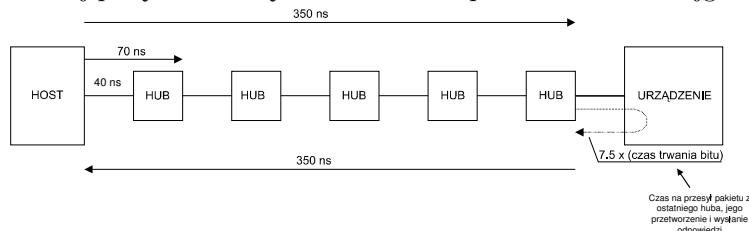
- Reakcja na fałszywy znacznik końca pakietu (*false EOP*)
- Ograniczenie czasowe oczekiwania na odpowiedź
- Przełączanie kolejnych pakietów danych (mechanizm *data toggle*)
- Wykrywanie transakcji występujących po zakończeniu ramki (tzw. „paplanie” – *babble*)
- Wykrywanie braku aktywności magistrali (LOA – *Loss Of Activity*)

Transfer kontrolny, przerwaniowy i masowy wysyłają pakiet ponownie, jeżeli jest on błędem, nie informując odbiorcy o tym. Transfer izochroniczny nie.

#### 2.4.4 Czas obiegu (round trip delay)

Ograniczenie czasowe oczekiwania na odpowiedź. Jest to podstawowy mechanizm informowania nadawcy o niepoprawnym przekazaniu pakietu. Ograniczenie czasowe nie może być mniejsze od  $16^*$ (czas trwania bitu), ani większe od  $18^*$ (czas trwania bitu).

Poniżej połaczenie urządzenia z hostem przez 5 hubów – najgorszy przypadek pod względem czasowym.



Jest on podstawa do rozważań na temat ograniczeń tego czasu.

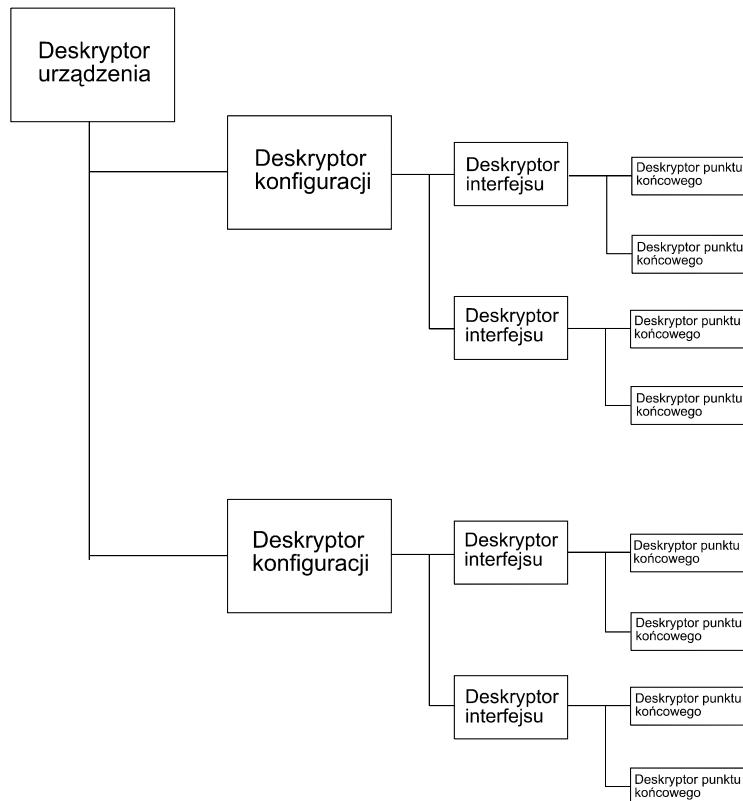
- Round trip delay:  $2 \times 350 \text{ ns} = 700 \text{ ns}$
- Czas trwania 1 bitu dla full speed:  $1/12 \text{ MHz} = 83 \text{ ns}$
- Round trip delay w bitach dla full speed:  $700 \text{ ns}/83 \text{ ns} = 8,5 \text{ bitu}$
- Timeout oczekiwania na odpowiedź:  $7,5 \text{ bitu} (\text{specyfikacja}) + 8,5 \text{ bitu} = 16 \text{ bitów}$

#### 2.5 Deskryptory w urządzeniach USB

W każdym urządzeniu USB znajduje się pełna informacja o sposobie komunikacji z urządzeniem, udostępniana podczas procesu enumeracji.

Informacja ta jest przechowywana w deskryptorach - tablicach o ścisłe określonej strukturze.

##### 2.5.1 Hierarchiczna struktura deskryptorów w urządzeniu USB



## 2.5.2 Rodzaje deskryptorów

- Deskryptor urządzenia - m.in. liczba konfiguracji dostępnych w urządzeniu. (+ wersja USB, klasa, protokół, ID, wytwórcza, nazwa produktu, nr seryjny)
- Deskryptor konfiguracji - opisuje każdą konfigurację z osobna. Zawiera informację o liczbie interfejsów przypisanych do danej konfiguracji oraz atrybuty zasilania, **maksymalny pobór prądu z krokiem co 2 mA.**
- Deskryptor interfejsu - posiada go każdy interfejs, który określa m.in. liczbę punktów końcowych z nim związanych.
- Deskryptor punktu końcowego - charakteryzuje każdy punkt końcowy. + adres, rodzaj transferu (*bulk* itp.), odstęp czasowy odpytywania dla izochronicznego (1) i przerwaniowego (1-255)
- Deskryptor łańcuchowy - zawiera kod określający język tekstu lub sam właściwy tekst.

W danym urządzeniu może wystąpić maksymalnie jeden deskryptor urządzenia, oraz wiele deskryptorów innych rodzajów. Każdy deskryptor może posiadać parę pomniejszych deskryptorów.

## 2.6 Wykrywanie i konfiguracja urządzeń

Cechą USB jest automatyczne wykrywanie urządzeń połączenia zasilania w systemie lub podłączeniu urządzenia do systemu. Procedura enumeracji zajmuje się rozpoznaniem urządzenia, sprawdzeniem czy komunikacja jest możliwa, przydzieleniem adresu, konfiguracją i instalacją sterownika.

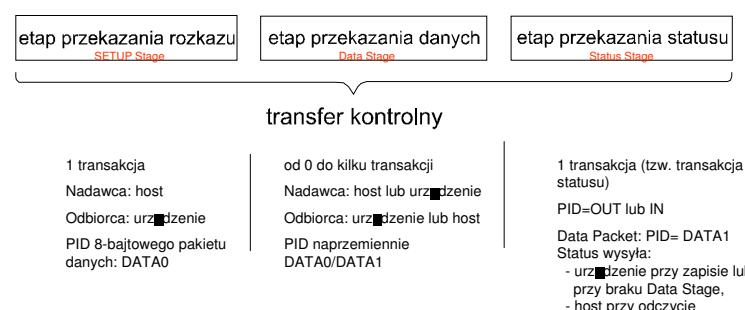
### 2.6.1 Procedura enumeracji urządzenia

1. Automatyczne wykrycie urządzenia - powoduje ustawienie bitu w rejestrze statusu huba, który odpowiada temu portowi.
2. Odblokowanie portu i generacja RESETU - przejście do stanu domyślnego.
3. Odczyt deskryptora urządzenia - skierowanie stosownego rozkazu do punktu końcowego 0 o adresie 0. Jest tylko jedno takie urządzenie w systemie - te, które właśnie podlega procesowi enumeracji.
4. Przypisanie adresu - unikalny adres dla urządzenia
5. Odczyt pozostałych deskryptorów - jeżeli istnieją
6. Konfiguracja urządzenia
7. Instalacja sterownika klienta
8. Urządzenie jest dostępne dla aplikacji

## 2.7 Kontrola urządzenia – transfer kontrolny

Transfer kontrolny służy do sterowania urządzeniem. Urządzenie USB otwiera kontrolny kanał komunikacyjny pomiędzy hostem a punktem końcowym 0 w urządzeniu.

### 2.7.1 Etapy transferu kontrolnego



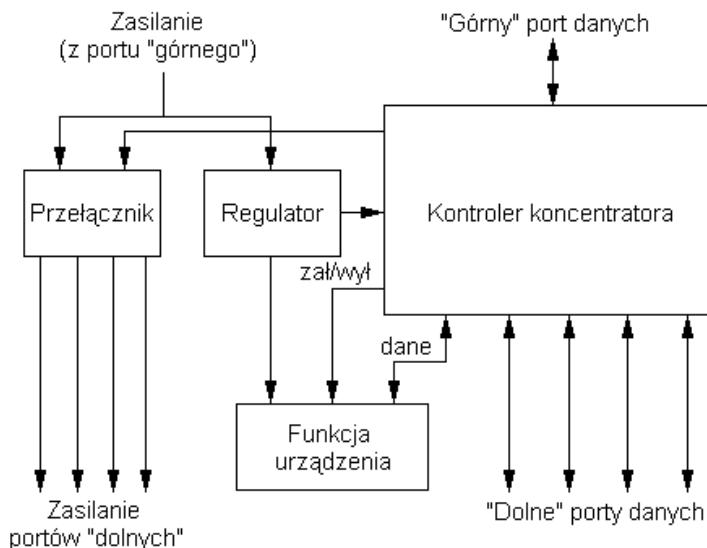
- Przekazanie rozkazu (*Setup Stage*) - jak w każdym pakiecie Token, następne pola zawierają adres urządzenia oraz punktu końcowego. Kolejnym elementem jest 8-bajtowy pakiet danych typu Data 0, który specyfikuje rozkaz. Transakcję kończy pakiet *handshake*.
- Przekazanie danych (*Data Stage*) - jeżeli 8 bajtów z powyższego nie wystarcza, wtedy następuje etap przekazania danych. Liczbę bajtów do przesłania określa słowo *Length* w polu transakcji Setup. Jeżeli ta liczba nie przekracza wartości *MaxPacketSize* to do przekazania danych wystarcza jedna transakcja, w przeciwnym razie odpowiednio dzieli się na więcej transakcji (sufit(*Length* / MPC)). Kodowanie:
  - Token packet IN (odczyt z urządzenia) / OUT (zapis).
  - Data packet DATA0 (jeśli są następne, to naprzemiennie DATA1/DATA0)
  - Handshake
- Przekazanie statusu (*Status Stage*) - potwierdzenie wykonania transferu lub poinformowanie o niepowodzeniu w realizacji. Normalnie występuje do zakończenia transakcji, ale host może rozpoczęć ją wcześniej, w Data Stage (wtedy DS jest przerywane).

## 2.8 Hub USB

### 2.8.1 Co to jest?

Standard USB definiuje oddzielną klasę urządzeń: klasa HUB. Jest to tak ważna część tego systemu, że została zdefiniowana w standardzie USB, a nie we własnej oddzielnej specyfikacji. Typowy hub posiada jeden deskryptor.

### 2.8.2 HUB zasilany z magistrali



### 2.8.3 Proces konfiguracyjny huba

- Odczyt standardowych deskryptorów urządzenia
- Przypisanie hubowi adresu
- Załączenie zasilania na porty, co jest niezbędne do detekcji podłączonych do portu urządzeń
- Odczyt punktu końcowego, "zmiany w hubie", w celu wykrycia urządzeń podłączonych do portów
- Odblokowanie portu w celu umożliwienia dostępu do urządzenia

#### 2.8.4 Punkt końcowy huba "zmiany w hubie"

- Cyklicznie odczytywany, umożliwia monitorowanie zmian na portach dolnych huba, co z kolei umożliwia wykrywanie dołączania i usuwania urządzeń.
- Odpływanie odbywa się przez wykorzystanie kanału przerwaniowego.
- Hub change point*: informuje o wystąpieniu:
  - Zmiany w zasilaniu lub nadmiernym obciążeniu prądowym huba
  - Zmiany na jednym lub kilku portach dolnych spowodowanej dołączeniem lub odłączeniem urządzenia.
- Odczyt *Hub change point* zwraca bajt statusowy huba. Jeżeli hub posiada więcej niż 7 portów dolnych - zwracane są dwa bajty.
- Jeżeli wszystkie bity w punkcie końcowym są ustawione na 0 (brak zmian), hub nie odsyła bajtu statusowego.
- Nie mylić z punktem końcowym 0 przez który wykonywana jest konfiguracja i kontrola.

#### 2.8.5 Działanie

Numer bitu	Znaczenie dla wartości 1
0	Wykrycie zmiany zasilania lub przełączenia huba
1	Wykrycie zmiany na porcie dolnym 1
2	Wykrycie zmiany na porcie dolnym 2
3	Wykrycie zmiany na porcie dolnym 3
4	Wykrycie zmiany na porcie dolnym 4
5	Wykrycie zmiany na porcie dolnym 5
6	Wykrycie zmiany na porcie dolnym 6
7	Wykrycie zmiany na porcie dolnym 7

Bajt statusowy huba

Jeżeli w bajcie statusowym bit0= 1, to host rozkazem GET\_HUB\_STATUS odczytuje dwa 16 bitowe rejesty stanu:

- status huba (*hub status*),
- zmiana statusu huba (*hub status change*)

Numer bitu	Znaczenie
0	Stan zasilania lokalnego: 0 – zasilanie lokalne jest poprawne 1 – zanik lokalnego napięcia zasilania
1	Sygnalizacja przełączenia prądowego: 0 – działanie huba normalne 1 – przełączenie prądowe huba
15 – 2	Bity zarezerwowane, ustawione na 0

Status huba

Jeżeli w bajcie statusowym bit0= 1, to host rozkazem GET\_HUB\_STATUS odczytuje dwa 16 bitowe rejesty stanu:

- status huba (*hub status*),
- zmiana statusu huba (*hub status change*)

• status huba (*hub status*),  
• zmiana statusu huba (*hub status change*)

Numer bitu	Znaczenie
0	Stan zasilania lokalnego: 0 – zasilanie lokalne jest poprawne 1 – zanik lokalnego napięcia zasilania
1	Sygnalizacja przełączenia prądowego: 0 – działanie huba normalne 1 – przełączenie prądowe huba
15 – 2	Bity zarezerwowane, ustawione na 0

Status huba

Numer bitu	Znaczenie
0	Zmiana zasilania lokalnego: 0 – nie wystąpiła zmiana zasilania lokalnego 1 – wystąpiła zmiana zasilania lokalnego
1	Zmiana przełączenia prądowego: 0 – układ kontroli nie sygnalizował wystąpienia przełączenia 1 – układ kontroli sygnalizował wystąpienie przełączenia
15 – 2	Bity zarezerwowane, ustawione na 0

Rejestr zmian statusu huba

Numer bitu	Znaczenie
0	Zmiana zasilania lokalnego: 0 – nie wystąpiła zmiana zasilania lokalnego 1 – wystąpiła zmiana zasilania lokalnego
1	Zmiana przełączenia prądowego: 0 – układ kontroli nie sygnalizował wystąpienia przełączenia 1 – układ kontroli sygnalizował wystąpienie przełączenia
15 – 2	Bity zarezerwowane, ustawione na 0

Rejestr zmian statusu huba

#### 2.9 Zasilanie urządzeń w systemie USB

USB jest również systemem dystrybucji i zarządzania zasilaniem urządzeń.

##### 2.9.1 Możliwe zasilania urządzeń USB

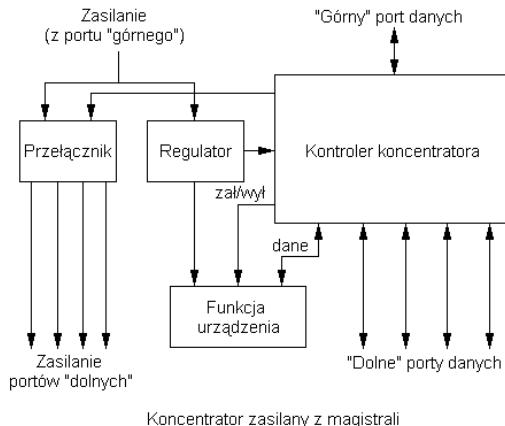
- Urządzenie zasilane z magistrali
- Urządzenie zasilane z własnego źródła
- Urządzenie zasilane częściowo z magistrali i własnego źródła

##### 2.9.2 Zasilanie hubów i pozostałych urządzeń



Dopuszczalne spadki napięcia zasilania.

### 2.9.3 Urządzenie zasilane z magistrali

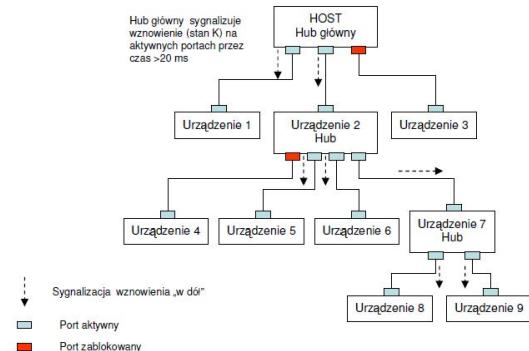


Dla portu zasilanego z magistrali USB i podłączonego do portu o obciążalności 500 mA, 100 mA jest zarezerwowane na zasilanie jego układu. Oznacza to, że 400 mA pozostaje dla jego portów dolnych. Ponieważ każdy port musi mieć zapewnione minimum 100 mA, taki hub może mieć nie więcej niż 4 porty dolne (stąd pewnie przyjęty standard).

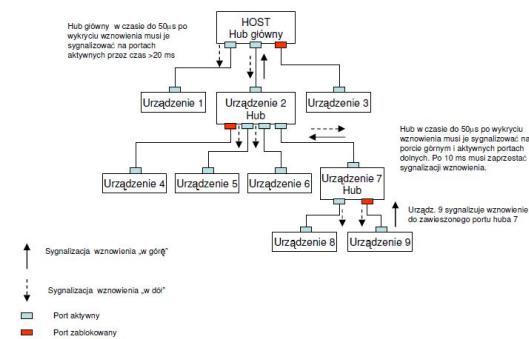
### 2.9.4 Zarządzanie zasilaniem

- Wprowadzenie w **stan zawieszenia** pracy systemu – *suspend*. Pozostając w zawieszeniu urządzenie zachowuje swoją konfigurację, co pozwala jego wznowienie (*resume*) do normalnego działania.
  - **Globalne** (*global suspend*) - rozkaz SetPortFeature (PORT\_SUSPEND) adresowany do huba głównego, zawiesza cały system
  - **Częściowe** (*selective suspend*) - rozkaz SetPortFeature (PORT\_SUSPEND) adresowany do huba zewnętrznego, w którym wybrany port (lub część systemu) ma zostać zawieszony. W ten sposób HUB zawiesza tylko siebie i poniższe HUBy/urządzenia.
  - Zawieszone porty nie propagują ruchu „w dół” oraz nie przekazują „w górę” żadnych sygnałów od urządzeń do nich podłączonych.
  - Urządzenia w stanie zawieszenia zachowują swój stan, co umożliwia wznowienie ich pracy bez ponownej konfiguracji.
  - Hub w stanie zawieszenia dodatkowo blokuje wszystkie nadajniki, zatrzymuje wewnętrzne zegary i zachowuje stan wszystkich portów dolnych.
  - Urządzenie w stanie zawieszenie pobiera prąd  $< 500[\mu A]$
  - Urządzenie automatycznie przechodzi do stanu *SUSPEND* po wykryciu braku aktywności magistrali przez 3 ms. (USB wysyła ramki co 1 ms)
    - \* Urządzenia pełnej szybkości wykorzystują pakiety SOF
    - \* Urządzenia małej szybkości wykorzystują generację przejścia ze stanu jałowego do stanu K w odstępach czasu nie większych niż 3 ms.
- **Wznowienie** pracy systemu - *resume*
  - Globalne
  - Wake-up
  - Wznowienie pracy urządzenia może nastąpić

- \* Z inicjatywy kontrolera - jako wznowienie po zawieszeniu globalnym lub częściowym



- \* Z inicjatywy urządzenia - po wystąpieniu zdarzenia wymagającego obsługi („budzenie” - *Wakeup*)



- Wznowienie po zawieszeniu globalnym - rozpoczęta hub główny wykonując rozkaz wznowienia pracy systemu. Oznacza gotowość wszystkich hubów i urządzeń do wykonywania transakcji.
- Wznowienie po częściowym zawieszeniu - może wykonać:
  - \* Host rozkazem *ClearPortFeature* (PORT SUSPEND) adresowanym do huba w którym znajduje się zawieszony port.
  - \* Urządzenie podłączone do zawszezonego portu poprzez *Wakeup*

## 2.10 Rozwiązania host kontrolerów

### 2.10.1 Co to jest?

Host kontrolera (*Host Controller Driver*) przetwarza poszczególne IRP na transakcje, tworzy listę transakcji w ramach ramki, przygotowuje scenariusz wykonania każdej transakcji i kontroluje jej wykonanie.

### 2.10.2 Rozwiązania

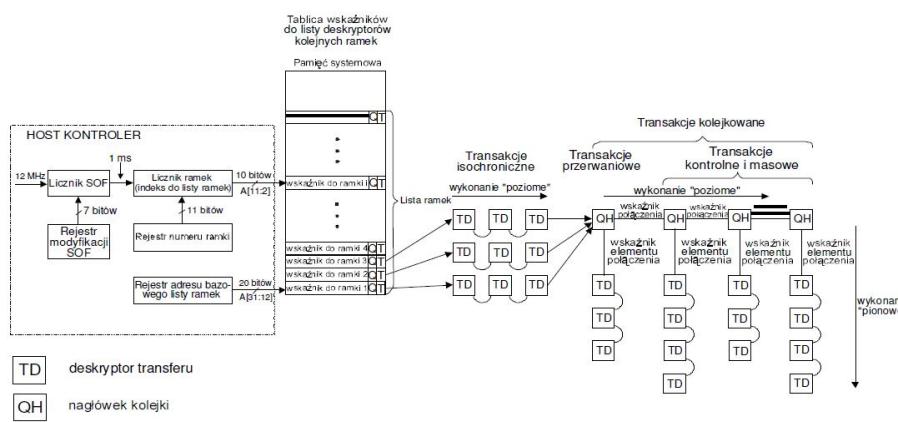
Opracowano dwa rozwiązania host kontrolera:

- Uniwersalny host kontroler - *Universal Host Controller* (Intel)
- Otwarty host kontroler – *Open Host Controller* (Compaq, Microsoft, National Semiconductor)

Są do siebie bardzo podobne, różnią się tylko sposobem obsługi oraz współpracą z hubem głównym.

### 2.10.3 Uniwersalny host kontroler (UHC)

## Zasada działania kontrolera UHC



#### Zasada działania kontrolera UHC

#### Ogólna postać deskryptora transferu (TD)

R = bity zarezerwowane

- Bity do odczytu/zapisu przez host kontroler
  - Bity tylko do odczytu przez host kontroler

#### Ogólna postać nagłówka kolejki (QH)

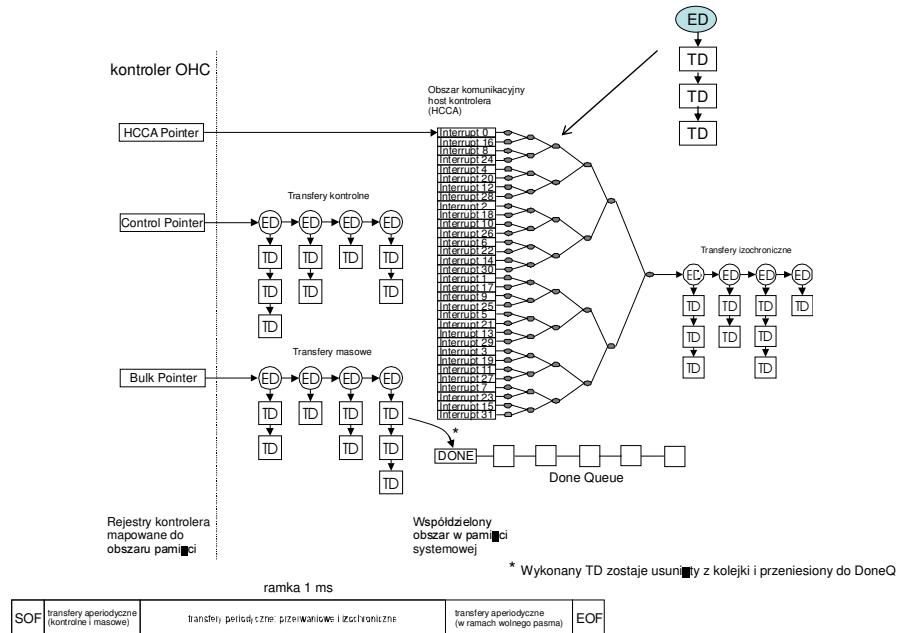
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Wskaznik do nagłówka kolejki	0 0 Q T	00-03h
	Wskaznik do elementu kolejki	0 R Q T	04-07h

R = bity zarezerwowane

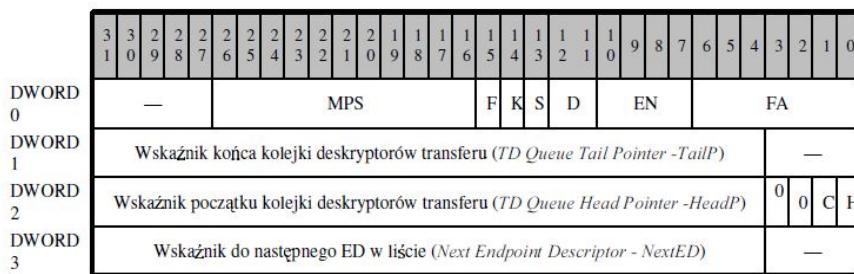
- Bity do odczytu/zapisu przez host kontroler
  - Bity tylko do odczytu przez host kontroler

#### 2.10.4 Otwarty host kontroler (OHC)

### Zasada działania kontrolera OHC



## Format deskryptora punktu końcowego (ED)



### Ogólna postać deskryptora transferu (TD)



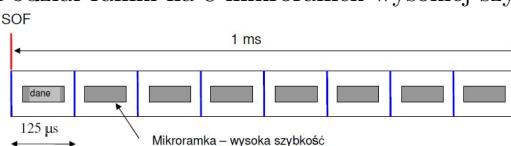
## 2.11 USB 2.0 - rozszerzenie standardu

### 2.11.1 Ważniejsze elementy wprowadzone w USB 2.0

- Wysoka szybkość transmisji (high speed) - 480 Mb/s
- Protokół PING-NYET
- Transakcja SPLIT
- Komunikacja z szerokopasmowym punktem izochronicznym
- Nowe typy pakietów

### 2.11.2 Wysoka szybkość transmisji

Podział ramki na 8 mikroramek wysokiej szybkości



- Mikroramka trwa  $125 \mu s$
- Na 1 ramkę przypada 8 mikroramek
- Wysoka szybkość transmisji - w mikroramce wynosi 480 Mhz.

### 2.11.3 Protokół PING-NYET

Potwierdzenie NYET (*NOT YET*) dla urządzeń *high speed*.

**Problem:** przy zapisie do urządzenia, jeżeli nie jest ono „gotowe na dane” potwierdzenie negatywne przychodzi dopiero po pakiecie danych – strata czasu.

**Rozwiązań:**

TOKEN PING – zapytanie urządzenia, czy jest gotowe do przyjęcia danych.

Możliwe odpowiedzi i reakcja hosta:

- ACK – wykonanie transakcji OUT
- NYET – host kontynuuje wysyłanie zapyta PING

**Korzyści:** lepsze wykorzystanie magistrali (PING jest krótki).

### 2.11.4 Transakcja SPLIT

**Problem:** Host i hub wysokiej szybkości komunikują się z urządzeniem małej lub pełnej szybkości. Szybkie przesyłanie danych pomiędzy hostem i hubem oraz wolne pomiędzy hubem i urządzeniem – konieczność buforowania danych w hubie.

**Rozwiązań:**

Transakcja dzielona, złożona z dwóch części:

- SSPLIT (*Start Split* - rozpoczęcie transakcji dzielonej)
- CSPLIT (*Complete Split* – zakończenie transakcji dzielonej).
- Pomiędzy tymi dwoma elementami transakcji dzielonej mogą być wykonywane inne transakcje z wysoką szybkością.

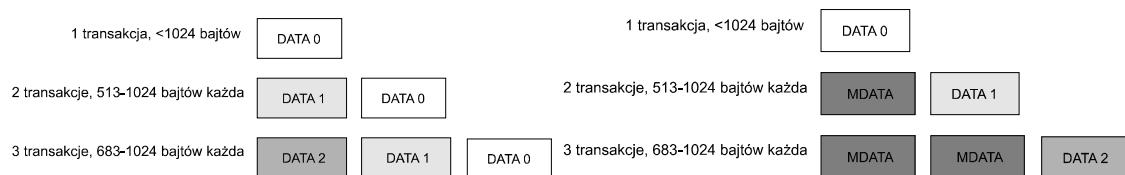
### 2.11.5 Komunikacja z szerokopasmowym punktem izochronicznym

Komunikacja z „normalnym” izochronicznym punktem końcowym zakłada jedną transakcję na ramę lub mikroramkę.

W przypadku punktów szerokopasmowych, obsługiwanych tylko przez kanały *high speed*, istnieje możliwość przekazania w jednej mikroramce większej ilości danych wykonując bezpośrednio po sobie od jednej do trzech transakcji.

Dane w takiej sekwencji transakcji muszą być oznaczone, przy czym nie wystarczą już „znaczniki” Data 0 i Data 1, dlatego wprowadzono dwa kolejne typy pakietów danych: **Data 2** i **MData**.

Pakiet **Data 2** wykorzystywany jest przy odczycie danych z urządzenia, natomiast **MData** i **Data 2** przy zapisie do urządzenia.



Dozwolone sekwencje pakietów danych  
przy odczycie szerokopasmowym

Dozwolone sekwencje pakietów danych  
przy zapisie szerokopasmowym

### 2.11.6 Specjalne pakiety TOKEN wprowadzone w USB 2.0

Kody pakietów specjalnych zgodnie z USB 2.0

Typ pakietu	PID
Pakiety danych	
Data 2	0111b
MData	1111b
Pakiety potwierdzenia	
NYET	0110b
Pakiety specjalne	
PRE	1100b
ERR	1100b
SPLIT	1000b
PING	0100b
Reserved	0000b

### 3 IEEE-488 and SCPI standards

Standard IEEE-488, ogłoszony w 1973 roku, określa organizację systemów ATE (*Automatic Test Equipment*), przeznaczonych do akwizycji danych i sterowania urządzeń w obiektach skupionych lub rozproszonych. Te mają zastosowanie w testowaniu wyborów, pomiarach laboratoryjnych, sterowaniu procesami przemysłowymi itp.

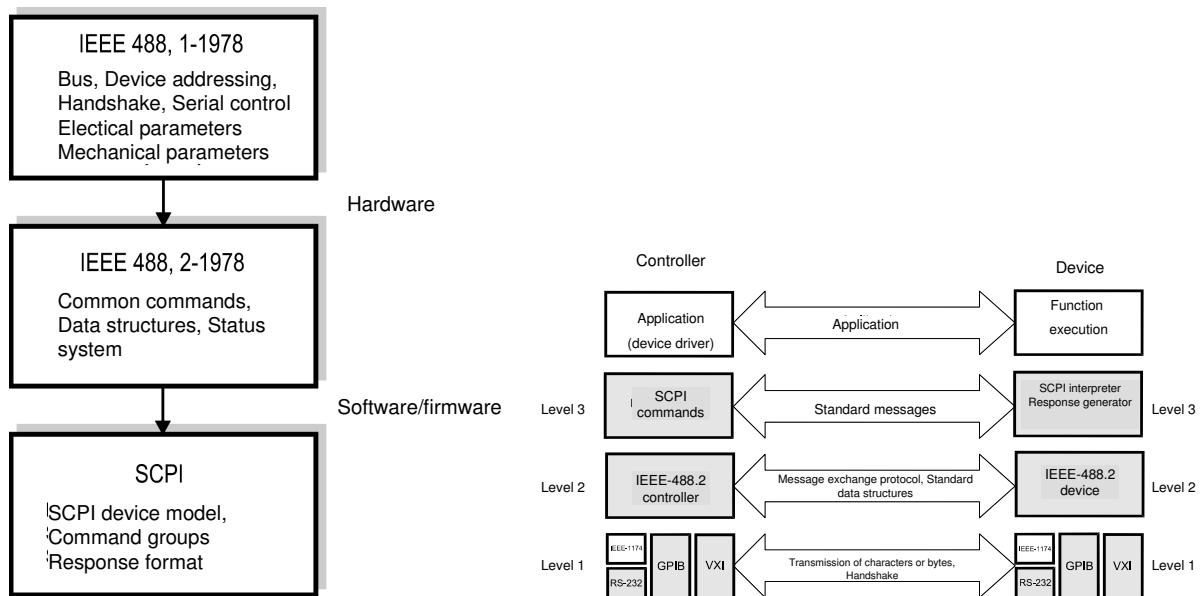
Urządzenia systemu ATE wyposażone są w układ komunikacyjny (interfejs), który umożliwia wymianę danych za pomocą łącza interfejsowego. Pracą systemu zarządza kontroler, który jest nadziedny w stosunku do pozostałych urządzeń, które pełnią rolę przyrządów wykonawczych.

Standard IEEE-488 znany jest również jako GPIB (*General Purpose Interface Bus*). Określa on dokładnie właściwości elektryczne i mechaniczne interfejsu, jego protokoły i funkcje, jednak nie normuje przesyłanych treści. Tym zajmuje się uzupełnienie standardu w postaci IEEE-488.2 (1973) oraz SCPI (1990).

Protokół systemu ATE można podzielić na 3 poziomy:

- Poziom 1, złożony z
  - Warstwa fizyczna (GPIB), określa łącze transmisyjne i sposób przesyłania jednostek
  - Warstwa łącząca danych, dostarcza reguły dostępu do łączą, sposób adresacji itp.
- Poziom 2, IEEE-488.2 kontroler < – > IEEE-488.2 urządzenie  
Standard IEEE-488.2 określa protokół wymiany komunikatów, syntaktykę komunikatów i strukturę danych.
- Poziom 3, rozkazy SCPI < – > Interpretacja rozkazów, generacja odpowiedzi  
SCPI normuje treść komunikatów przesyłanych pomiędzy kontrolerem a urządzeniem.
- Poziom 4, Aplikacja < – > Wykonanie funkcji urządzenia (ale to już nas raczej nie interesuje).

Ilustracja rozwoju standardu IEEE-488 oraz model komunikacyjny systemu ATE



### 3.1 IEEE-488 (GPIB)

Standard GPIB ogłoszono w 1979 roku, jest definicją warstwy fizycznej oraz łącza danych dla systemów pomiarowo-kontrolnych, przeznaczonych do:

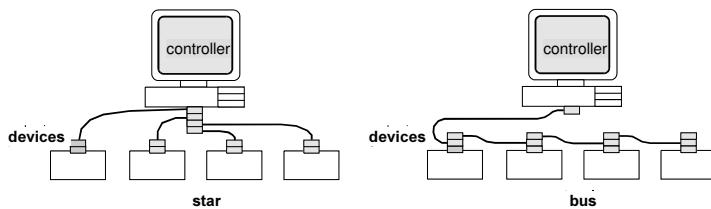
- Testowania wyborów
- Pomiarów laboratoryjnych i przemysłowych

#### 3.1.1 Interface profile

Standard ten określa budowę i działanie części komunikacyjnej systemu opartego na magistrali, do której można:

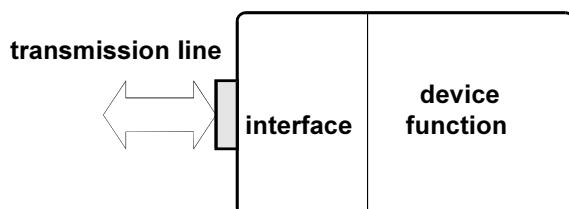
- Podłączyć do 15 urządzeń,
- Rozłożonych na niewielkim obszarze (zasięg do 20 m),
- Maksymalna odległość między kolejnymi urządzeniami to 2 m,
- Wymieniających między sobą dane rzędu 300 do 500 kB / s

Komunikacją zarządza kontroler, który konfiguruje system do komunikacji, obsługuje zgłoszenia żądania obsługi i wykonuje podstawowe operacje. **Dwie podstawowe konfiguracje systemu GPIB**



#### Podział urządzenia systemowego na część interfejsową i urządzenie właściwe

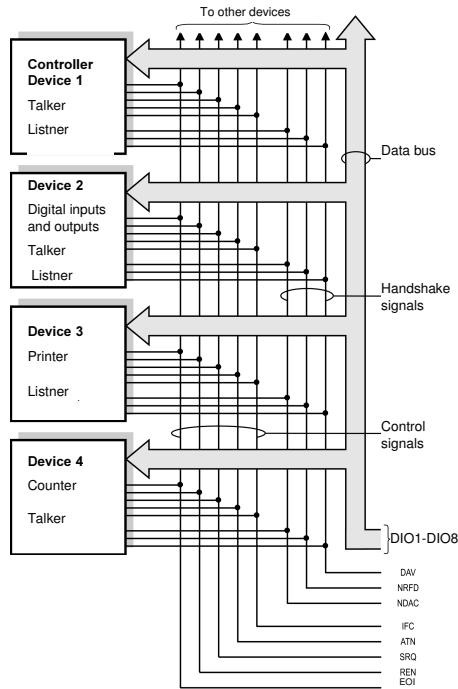
Można tak podzielić każde urządzenie wykonawcze. Część interfejsowa to zdolność urządzenia do wymiany komunikatów z innymi elementami systemu, a urządzenie właściwe wykonuje podstawowe funkcje urządzenia.



### 3.1.2 Magistrala

#### Właściwości

- Magistrala zawiera 16 linii
  - 8 dla transmisji danych
  - 8 dla sygnałów sterujących
- Jest to łącze równoległe, umożliwia przesłanie wszystkich bitów bajtu
- DIO1-DIO8 – dwukierunkowa magistrala danych
- 5 sygnałów kontrolnych:
  - ATN (*Attention* - wyjście), określa tryb pracy magistrali danych
    - \* ATN=0 - transmisja bajtów danych
    - \* ATN=1 - transmisja instrukcji sterujących (adresów lub rozkazów)
  - REN (*Remote Enable* - wyjście), 1 oznacza, że system jest zarządzany zdalnie przez kontroler
  - IFC (*Interface clear* - wyjście), 1 powoduje wyzerowanie części interfejsowych urządzeń
  - SRQ (*Service request* - wejście), 1 oznacza, że w systemie jest co najmniej jedno urządzenie żądające obsługi
  - EOI (*End Of Identity* - wyjście or wejście), zależny od stanu linii ATN
    - \* EOI=1 razem z ATN=0 (EOI=1 ^ ATN=0) - realizacja znacznika końca bloku danych
    - \* EOI=1 razem z ATN=1 (EOI=1 ^ ATN=1) - kontrola równoległa
- 3 sygnały sterujące (*Byte handshake signals*)
  - NRFD (*Not Ready For Data* - wyjście or wejście) - jest co najmniej jeden odbiorce niegotowy do przyjęcia danych
  - DAV (*Data Valid* - wyjście or wejście) - bajt znajdujący się na magistrali jest ważny i może zostać odczytany
  - NDAC (*Not Data Accepted* - wyjście or wejście) - przyjęcia bajtu na magistrali nie zostało potwierdzone przez wszystkich odbiorców



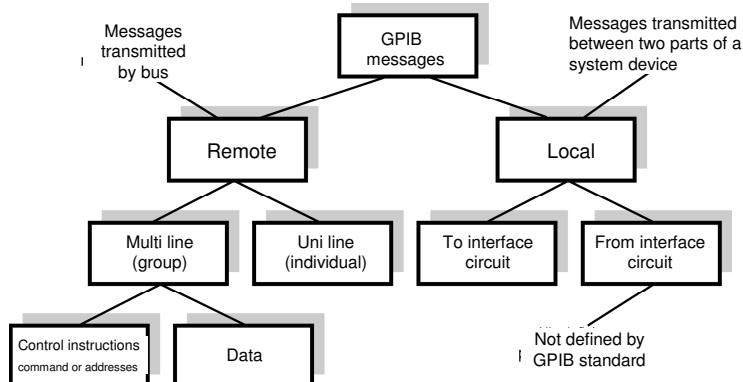
Rysunek 1: Reprezentacja magistrali

### 3.1.3 Funkcje interfejsowe

Każda funkcja reprezentuje pewną określoną zdolność urządzenia jako jednostki systemu GPIB. Ciekawostka: każda funkcja implementowana jest tylko w tych urządzeniach, które jej potrzebują.

- SH (*Source Handshake*) - inicjator transmisji, dla urządzeń mogących nadawać dane
- AH (*Acceptor Handshake*) - akceptor transmisji, dla urządzeń mogących odbierać dane
- T (*Talker*) - nadawca, umożliwienie wprowadzenia bajtu na magistralę danych
- L (*Listener*) - odbiorca, umożliwienie odebranie bajtu z magistrali danych
- SR (*Service Request*) - zgłoszenie żądania obsługi
- DC (*Device Clear*) - zerowanie urządzenia, umożliwienie zdalnego zerowania (reset)
- DT (*Device Trigger*) - wyzwalanie urządzenia, umożliwienie zdalnego wyzwolenia operacji (pomiaru)
- RL (*Remote/Local*) - zdalne / lokalne, umożliwienie przełączania trybu urządzenia
- PP (*Parallel Poll*) - kontrola równoległa, wszystkie linie danych mogą odbierać żądania. Każde urządzenie ma przypisaną linię danych.
- C (*Controller*) - kontroler, umożliwienie zdalnego zarządzania systemem GBIB i przekazywanie zarządzania innym kontrolerom - w systemie może być wiele kontrolerów, jednak tylko jeden może być aktywny, reszta musi być uśpiona.

### 3.1.4 Podział komunikatów



komunikaty dzielimy na zdalne(przesyłane po magistrali) i lokalne (wymieniane między częścią interfejsową, a urządzeniem właściwym).

Komunikaty zdalne dzielimy na:

- wieloliniowe: przenoszone przez magistrale danych. Tutaj wyróżniamy:
  - instrukcje sterujące (adresy lub rozkazy)
  - dane
- jednoliniowe: przenoszone przez wydzieloną linię sterującą

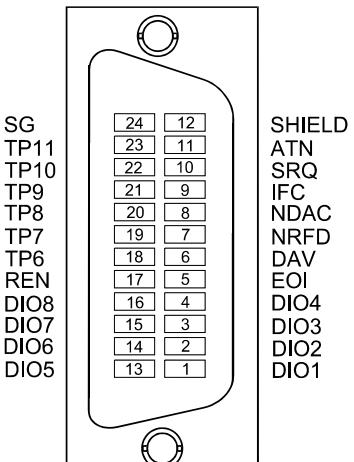
### Podział funkcjonalny komunikatów:

- UC (*Universal Commands*) - rozkazy uniwersalne, przeznaczone dla wszystkich urządzeń w systemie
- AC (*Addressed Commands*) - rozkazy adresowane, przeznaczona tylko na konkretnych urządzeń
- AD (*Addresses*) - adresy, komunikaty umożliwiające wyznaczenie urządzenia nadającego i urządzeń odbierających
- HS (*Handshake*) - komunikaty synchronizacji, kontrolujące transfer bajtu przez magistralę danych
- DD (*Device Depended (data)*) - komunikaty zależne od urządzenia, ich znaczenie jest zależne od przyrządu
- ST (*Status*) - komunikaty statusowe, udostępniają informacje o stanie urządzenia
- SE (*Secondary*) - komunikaty wtórne, adres lub rozkaz wtórny

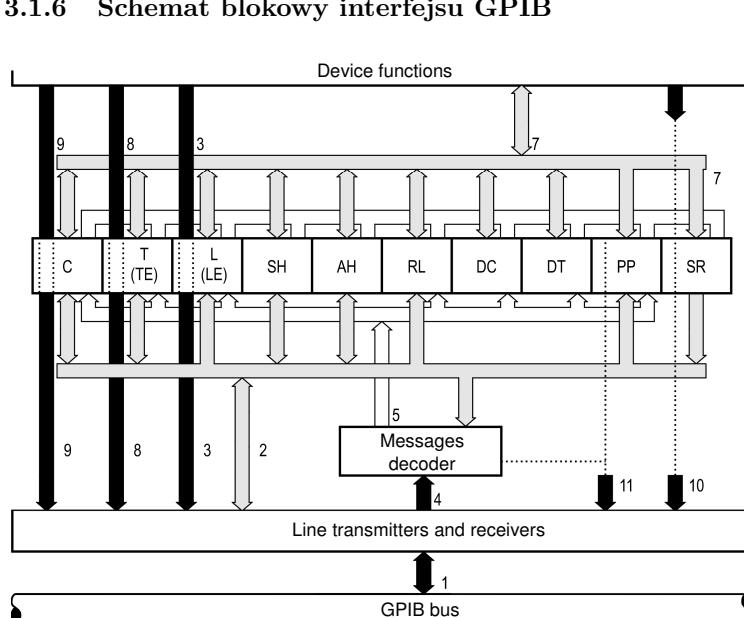
### 3.1.5 Złącze interfejsu (connector)

W systemie GPIB zastosowano 24-stykowe złącze.

- Wszystkie sygnały mają wspólny potencjał odniesienia (masę SG).
- Linie sterujące: DAV, NRFD, NDAC, IFC, SRQ oraz ATN są wykonane w postaci skręcanej pary przewodów, w których przewód powrotny jest połączony odpowiednio do punktów TP1, TP2, ... TP11.
- Zacisk SHIELD stanowi kontakt do podłączenia ekranu kabla.



Rysunek 2: Złącze



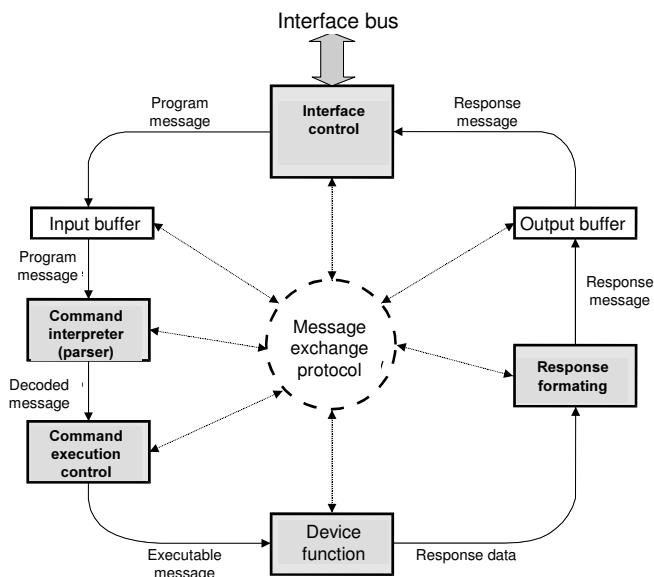
## 3.2 Standard IEEE-488.2

Standard ten określa:

- Protokół wymiany komunikatów pomiędzy kontrolerem, a urządzeniem na poziomie warstwy 2
- Synchronizację działania kontrolera i urządzenia
- Strukturę systemu statusowego w urządzeniu i sposób jego obsługi
- Zestaw tzw. rozkazów wspólnych

### 3.2.1 Schemat funkcjonalny urządzenia

Ilustracja przepływu danych i ich przetwarzania



### 3.2.2 Protokół wymiany komunikatów

Jego zadaniem jest zapewnienie poprawnego transferu wiadomości pomiędzy kontrolerem, a urządzeniem i określenie reakcji urządzenia na naruszenia reguł protokołu.

Ogółem komunikacja pomiędzy kontrolerem a urządzeniem polega na:

- wysyłaniu do urządzenia komunikatów programujących, które można podzielić na rozkazy bez odpowiedzi i zapytania
- odbieraniu odpowiedzi z urządzenia na wysłane zapytania

#### Zasady komunikacji kontrolera z urządzeniem (*Device responsibility*)

1. Rozkazy są wykonywane przez urządzenie zgodnie z kolejnością ich wprowadzenia do przyrządu (poza ścisłe określonymi wyjątkami)
2. Urządzenie jest zobowiązane odesłać odpowiedź tylko na wcześniej odebrane zapytanie (przez kontroler)
3. Odpowiedzi na zapytania są odsyłane w kolejności odpowiadającej odebranym zapytaniom

### Reguły protokołu komunikacyjnego (*Controller responsibility*)

1. Kontroler nie może odebrać odpowiedzi na zadane zapytanie, zanim nie zakończy rozpoczętego komunikatu programowego.
2. Kontroler nie może wysłać kolejnego komunikatu programowego, jeżeli nie odebrał w całości wszystkich odpowiedzi na zadane wcześniej zapytania.
3. Kontroler nie może podejmować próby odczytu odpowiedzi z urządzenia, jeżeli wcześniej nie wysłał żadnego zapytania.
4. Zapytanie powodujące odesłanie odpowiedzi o nieokreślonym rozmiarze musi być umieszczone jako ostatnie zapytanie w komunikacie programowym (zawierającym kilka zapytań).
5. Istnieje niebezpieczeństwo zablokowania urządzenia w przypadku wprowadzenia do urządzenia komunikatu programowego zawierającego kilka zapytań.

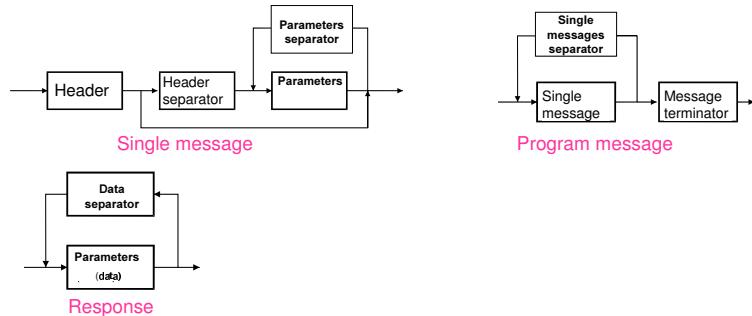
## 3.3 Standard SCPI

SCPI (*Standard Commands for Programmable Instruments*), normalizuje treści komunikatów przepływających pomiędzy kontrolerem, a urządzeniem. Można go uważać za język kontroli, określono w nim budowę i reguły syntaktyczne komunikatów programowych wysyłanych do urządzenia i odpowiedzi z urządzenia.

Wydodzi się z normy IEEE-488.2.

### 3.3.1 Budowa komunikatów

Komunikaty składają się z jednego lub kilku komunikatów jednostkowych. Na końcu komunikatu programowego musi wystąpić znacznik terminujący.



#### Komunikat programowy:

- **Nagłówek:** tekst określający znaczenie komunikatu. Może to być pojedyncze słowo kluczowe lub ciąg słów oddzielonych znakiem dwukropka (':', 3Ah)
- **Separator nagłówka:** występuje pomiędzy nagłówkiem, a pierwszym parametrem jako jeden lub kilka znaków niedrukowalnych ASCII z zakresu od 0 do 31 (dec), oprócz NL, 0Ah. Najczęściej jest to jedna lub kilka spacji.
- **Parametry:** określają wartości związane z rozkazem. Separatorem pomiędzy parametrami jest znak przecinka (',', 2Bh), przy czym białe znaki są ignorowane.
- **Separator komunikatów jednostkowych:** jest to znak średnika (';', 3Bh)
- **Terminator komunikatu:** znacznik terminujący, który jest wymagany do zakończenia komunikatu. Jest to znak LF (NL, 0Ah)

#### Odpowiedź:

- Zawierają tylko dane,
- a reszta jak w powyższym

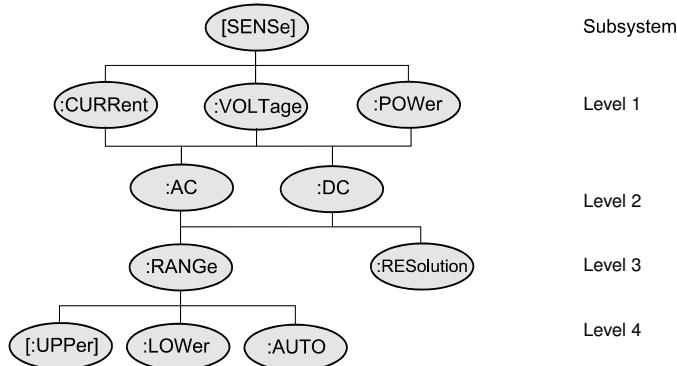
## Zapytanie:

- Nagłówki zapytań mają na końcu znak pytajnika ('?', 3Fh)

## Komunikaty wspólne:

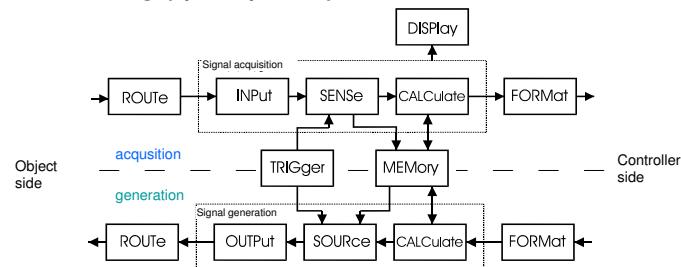
- Posiadają prefiks w postaci znaku gwiazdki ('\*', 2Ah)

**Drzewiasta struktura rozkazów SCPI** (jest to fragment drzewa podsystemu [SENSe]):



### 3.3.2 Model urządzenia w standardzie SCPI

Schemat blokowy ilustrujący przepływ sygnału i danych w urządzeniu. Poszczególne bloki reprezentują podstawowe funkcje urządzenia, a ich nazwy są jednocześnie nazwami grup rozkazów (podsystemów), które obsługują daną funkcjonalność.



Wyróżnia się dwie części:

- akwizycję sygnału ("czujnik")
- generację sygnału ("źródło")

### 3.3.3 Podsystemy w standardzie SCPI

Podsystem to grupa funkcjonalności. W każdym konkretnym urządzeniu występują tylko te, które są mu potrzebne.

Owych podsystemów jest kilkanaście, tutaj są przedstawione najistotniejsze:

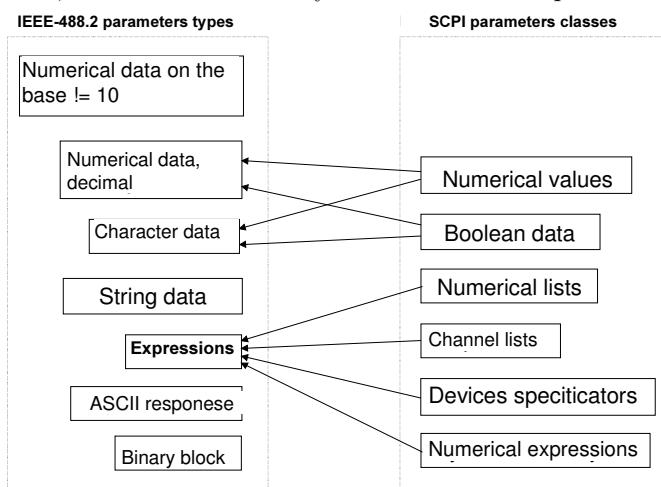
- **CALCulate** - odpowiedzialny za matematyczne przetwarzanie wyników i danych po akwizycji sygnału lub przed generacją sygnału
- **INput** - przekształcanie sygnału (wzmocnieniem, filtracją itp.) przed właściwą konwersją dokonywaną w bloku SENSE
- **OUTput** - kontrola przekształtników sygnału wyjściowego (po generacji przez blok SOURCE)
- **ROUTe** - odpowiedzialny za podłączenie sygnału wejściowego do układu przekształcającego sygnał lub wyjściowego do zacisków wyjściowych przyrządu
- **SENSe** - kontrola podstawowego modułu akwizycji sygnału (bloku wykonującego konwersję analogowo-cyfrową)

- **SOURce** - kontrola modułu generacji sygnału
- **SYStem** - podsystem umożliwiający "utrzymanie" przyrządu (kontrolę pomocniczych parametrów)

Wyżej wymienione słowa kluczowe mogą występować zarówno w postaci rozwinietej (np. **Calculate**), jak i skróconej (**CALC**).

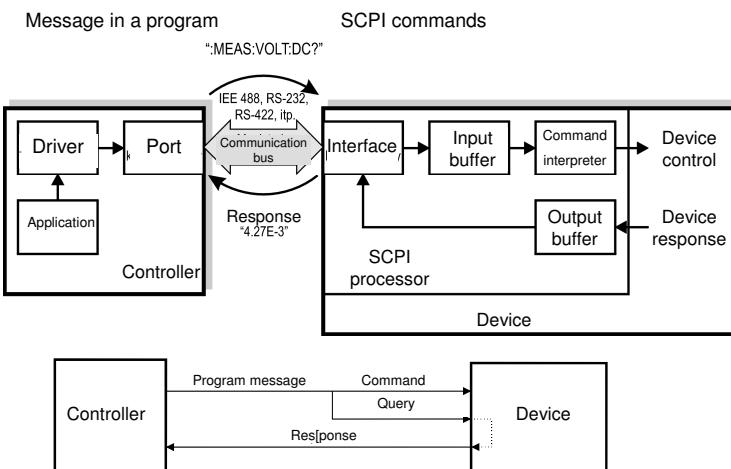
### 3.3.4 Parametry

Typy parametrów w komunikatach programowych i odpowiedziach zostały określone w standardzie IEEE-488.2, natomiast w SCPI wyróżniono kilka klas parametrów w ramach wybranych typów IEEE-488.2.



### 3.3.5 Kontrola urządzeń

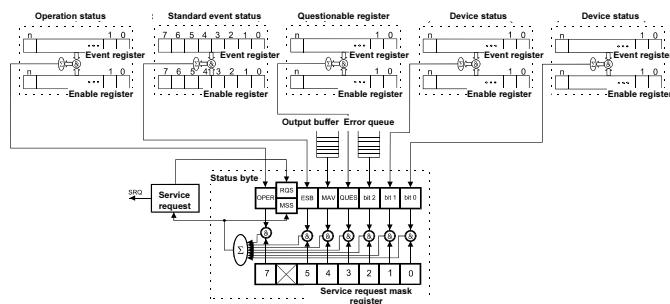
Urządzenie kontrolowane przez rozkazy SCPI



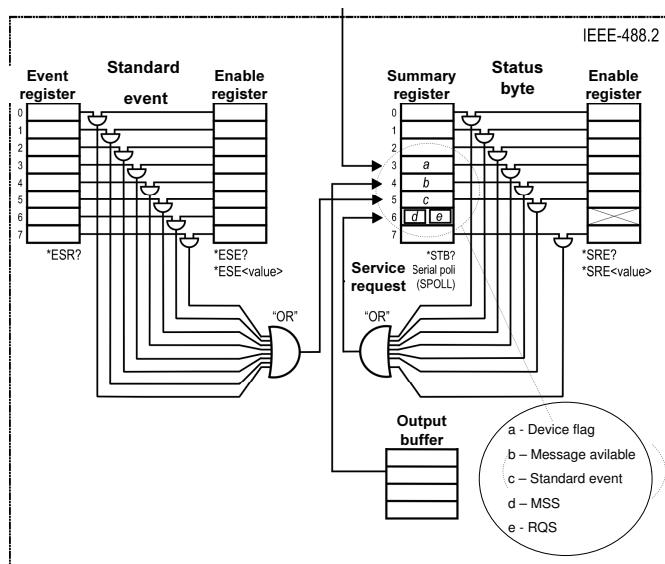
### 3.3.6 System rejestrów statusu

Urządzenie przechowuje informację o stanie w tzw. SRS, który ma strukturę hierarchiczną, złożoną z 5 grup rejestrów statusowych. Każda z nich składa się z rejestrów zdarzeń i rejestrów maski.

1. Bajt statusowy - umieszczona w najwyższym poziomie hierarchii, udostępnia uogólnioną informację o stanie urządzenia.
2. Rejestr zdarzeń standardowych - 8-bitowy rejestr na drugim poziomie hierarchii, informuje o wystąpieniu zdarzeń uważanych za "typowe" dla urządzenia SCPI.
3. Rejestr znaczników urządzenia - 16-bitowy rejestr na drugim poziomie hierarchii, informuje o "jałości" wyników pomiaru i sprawności ważniejszych układów w przyrządzie.
4. Rejestr operacji - 16-bitowy rejestr na drugim poziomie hierarchii informujący o stanie wykonywanych przez przyrząd operacji.
5. Rejestr statusu zależny od urządzenia - specyficzny dla urządzenia, liczba i znaczenie bitów są od niego zależne.



Minimal SCPI status system

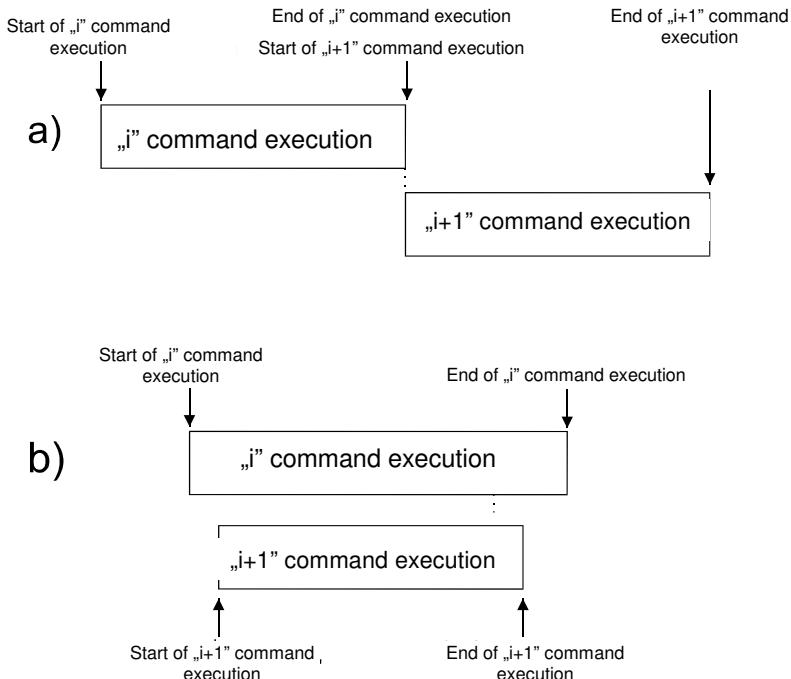


### 3.3.7 Synchronizacja urządzeń z aplikacją w standardzie SCPI

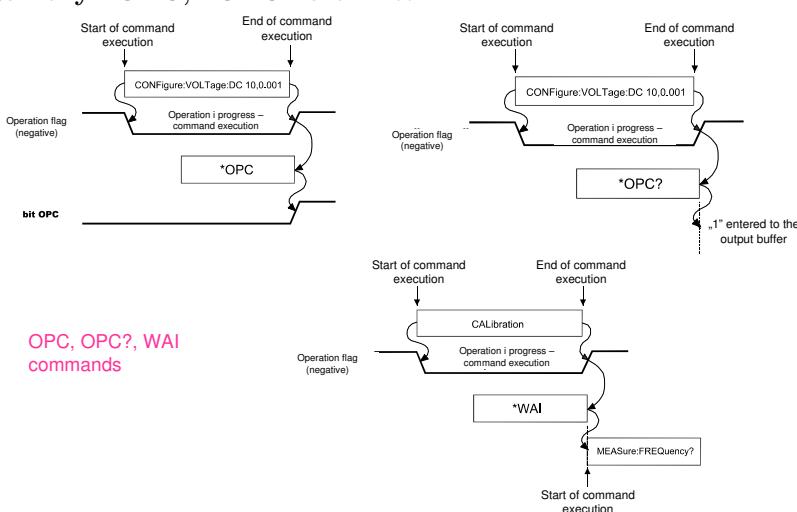
Zadaniem synchronizacji jest zapewnienie właściwej sekwencji operacji urządzeń w systemie. W urządzeniach SCPI synchronizacja aplikacji sterującej z operacjami wykonywanymi przez urządzenie oparta jest na rozkazach \*OPC, \*OPC? oraz \*WAI.

**Rodzaje rozkazów** Dzielimy je na:

- Sekwencyjne - wykonywane jeden po drugim w kolejności wprowadzania ich do przyrządu.
- Nakładające się - możliwe wykonywanie wielu rozkazów jednocześnie.



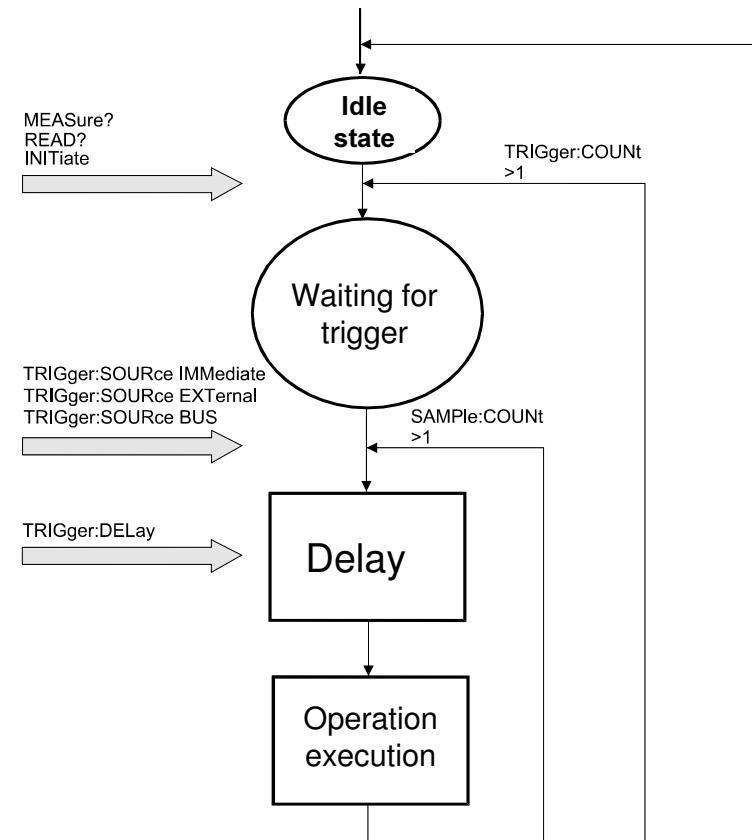
### Rozkazy \*OPC, \*OPC? oraz \*WAI



### 3.3.8 Wyzwalanie operacji (device trigger)

Standard SCPI umożliwia uzależnienie wyzwolenia od spełnienia szeregu warunków i wystąpienia różnych zdarzeń (elastyczność).

Typowy układ wyzwalania operacji (trigger circuit in HP-34401A multimeter)



## 4 IEEE-1394 (FireWire)

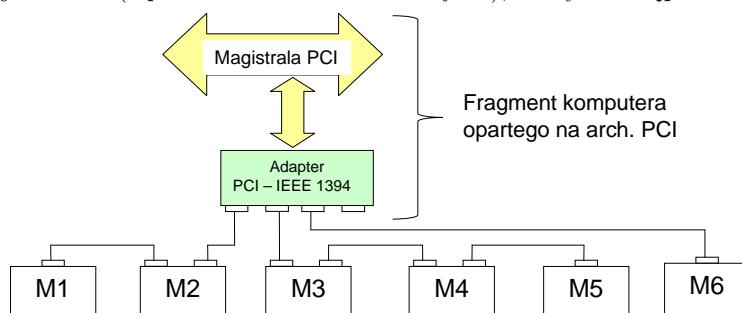
Cel projektantów FireWire: stworzenie interfejsu opartego na łączu szeregowym, który mógłby zastąpić równoległy port SCSI. Chodziło o zaprojektowanie bardzo szybkiej magistrali, dogodnej do transferu dużej ilości danych wymienianych z urządzeniami pamięci masowych oraz danych multimedialnych.

### 4.1 Możliwości interfejsu IEEE 1394a

- Podłączenie do komputera dużej liczby różnorodnych urządzeń
- Wykrywanie włączenia urządzenia do systemu oraz jego odłączenia (*hot plugging*)
- Automatyczna konfiguracja systemu do komunikacji
  - określenie topologii systemu
  - samoidentyfikacja urządzeń
  - wyznaczenie szybkości transmisji
  - konfiguracja urządzeń do komunikacji
- Szybka transmisja danych pod kontrolą pakietowego protokołu komunikacyjnego, oparta na transakcjach asynchronicznych i izochronicznych
- Rywalizacyjny dostęp urządzeń do łącza pod kontrolą procedury arbitrażowej
- Zasilanie urządzenia z systemu

### 4.2 Topologia i interfejs IEEE 1394a

Układem umożliwiającym podłączenie komputera opartego na PCI do magistrali szeregowej IEEE 1394 jest most (*Open Host Controller Interface*), który udostępnia zazwyczaj kilka portów IEEE 1394.



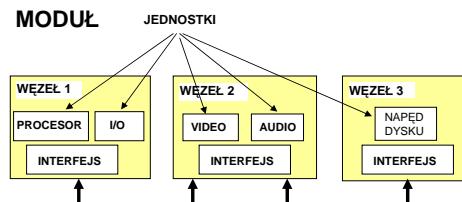
Rys.1. Podłączenie urządzeń do portów IEEE 1394

- Topologia rozgałęzionej gwiazdy.
- Struktura: rozgałęziona magistrala (*daisy chain*).
- Poszczególne urządzenia, zwane węzłami, rywalizują ze sobą o dostęp do łącza w celu przesłania porcji danych (pakietu). Pakiety przesyłane są co  $125 \mu s$
- **Urządzenia z jednym portem stanowią zakończenie gałęzi** (liście), urządzenia z wieloma portami umożliwiają rozbudowę gałęzi przez podłączenie kolejnego urządzenia. Liściem może też być urządzenie wieloportowe, w którym wykorzystany jest tylko jeden port.
- **Komunikacja pomiędzy portami oparta jest na zasadzie punkt-punkt** (*point to point*), co dla urządzenia wieloportowego oznacza odbiór i detekcję całego pakietu przez jeden port, a następnie jego resynchronizację do lokalnego zegara i retransmisję do pozostałych portów.
- **System nie posiada hosta**, komunikacja pomiędzy urządzeniami odbywa się bez interwencji stacji nadzędnej. Ten rodzaj komunikacji określamy jako P2P (*peer to peer*)

- Konfiguracja urządzeń następuje dynamicznie, bez wykorzystania kontrolera systemu. Interfejsy urządzeń mają wbudowane funkcje konfiguracyjne, które umożliwiają określenie topologii systemu, parametryzację portów urządzeń oraz przekazanie informacji o urządzeniu. Odbywa się po każdym resecie.

### 4.3 Architektura węzła

System IEEE 1394 ma topologię rozgałęzionej gwiazdy i oparty jest na transferach bezpośrednich między



Rys. 2. Urządzenie interfejsu Firewire

urządzeniami (brak hosta w systemie).

Ważniejsze terminy stosowane w IEEE 1394:

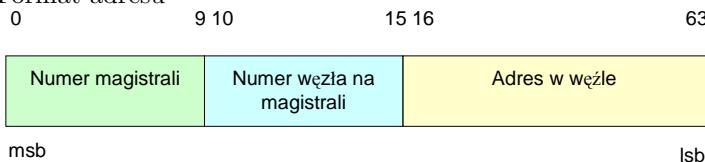
- moduł (*module*): fizyczne urządzenie podłączone do magistrali, które może zawierać kilka węzłów
- węzeł (*node*): logiczne urządzenie wewnętrzne modułu
- jednostka (*unit*): element węzła o określonej funkcjonalności (np. procesor, pamięć, I/O)

### 4.4 Przestrzeń adresowa i jej podział

Przestrzeń adresowa jest zgodna z IEEE Std 1212-1991 Control and Status Register (CSR) Architecture.

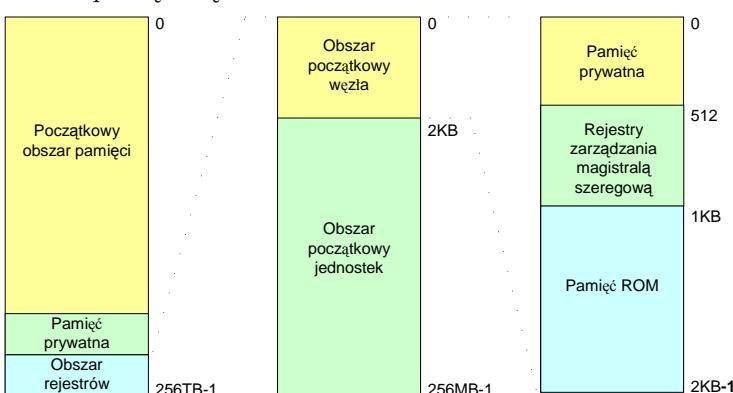
- Adres zawiera 64 bity podzielone na:
  - Numer magistrali (*Bus Number*), 10 najstarszych bitów: Bus 0 do Bus 1023
  - Numer węzła w ramach magistrali (*Node Number*), 6 bitów: Node 0 do Node 63
  - Pole adresowe w węźle (*Node Address*), 48 bitów.

- Format adresu



Rys. 3. Adresacja w systemie IEEE-1394

- Podział pamięci węzła



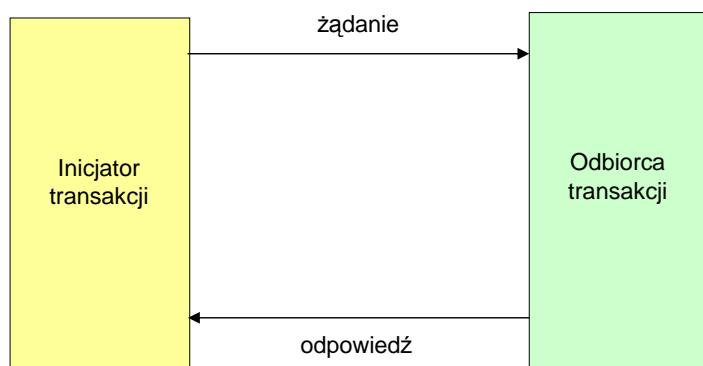
Rys.4. Podział pamięci węzła

## 4.5 Transfery i transakcje

Zasady komunikacji:

- Transfery są żądaniami przesłania danych.
- Transfery dzielone są na transakcje (w celu umożliwienia podziału pasma pomiędzy węzłami).
- Wyróżnia się transakcje:
  - **Asynchroniczne** - wymagają potwierdzenia (odpowiedzi) i w przypadku błędów są powtarzane. Mogą być wykonywane cyklicznie lub aperiodyczne.
  - **Izochroniczne** - nie są potwierdzane. Nie kontroluje się poprawności przesłania danych, a jedynie zapewnia stałego rytm ich dostarczania.

### 4.5.1 Transakcje asynchroniczne



Rys. 5. Model transakcji asynchronicznej

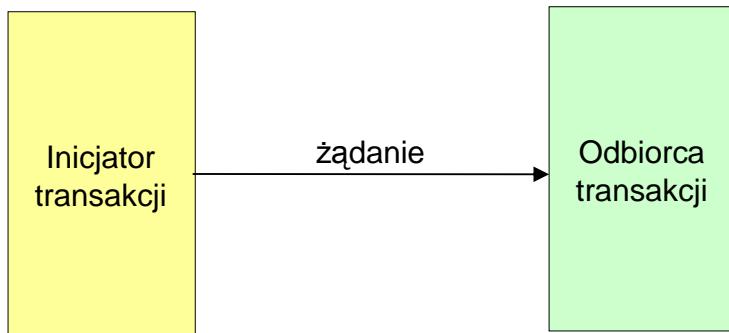
Rozpoczęcie transakcji nie jest synchronizowane z odbiorcą. Wyróżniono 3 rodzaje transakcji asynchronicznych:

- Odczyt (*read*) - pobranie danych od odbiorcy
- Zapis (*write*) - przekazanie danych do odbiorcy
- Zamknięcie (*lock*) - odczyt danych, ich modyfikacja i ponowny zapis

Każda transakcję rozpoczyna inicjator transakcji (*requester*) i potwierdza odbiorca transakcji (*responder*). Zatem transakcja dzielona jest na dwie części:

- Fazę przekazania żądania (adres, rozkaz i dane)
- Fazę przekazania odpowiedzi (przy zapisie - status wykonania, przy odczytce - dane)

#### 4.5.2 Transakcje izochroniczne



Rys.6. Model transakcji izochronicznej

- Transfer izochroniczny zapewnia regularne dostarczanie danych bez sprawdzania ich ważności.
- Dane dostarczane są cyklicznie w odstępach  $125 \mu s$ , bez potwierdzenia ich odbioru przez urządzenie odbierające.
- Życie danych izochronicznych jest wyjątkowo krótkie, a ich uszkodzenia nie mają trwałych konsekwencji.
- Przykładem mogą być dane cyfrowe reprezentujące dźwięk przesyłane do głośników.

### 4.6 Konfiguracja urządzeń do komunikacji, współdzielenie magistrali

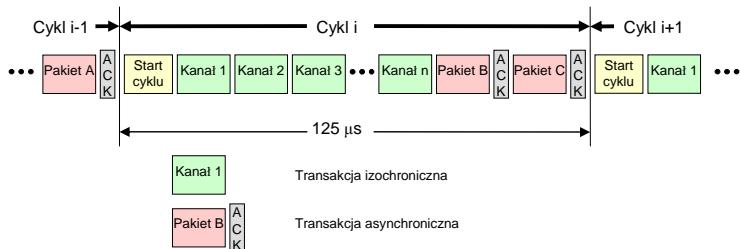
#### 4.6.1 Konfiguracja

Po każdym włączeniu zasilania lub zmianie węzłów w systemie (dołączenie lub odłączenie urządzenia) następuje automatyczna konfiguracja magistrali, której rezultatem jest przygotowanie węzłów do działania.

Po przeprowadzeniu konfiguracji węzły mogą rywalizować o dostęp do magistrali (arbitraż), a po jego uzyskaniu przekazywać dane.

#### 4.6.2 Współdzielenie magistrali

Transakcje asynchroniczne i izochroniczne współdzielą magistralę, ponieważ zarządzanie magistralą „stara się” umożliwić jednoczesne (quasi) jej wykorzystanie przez różne transfery.

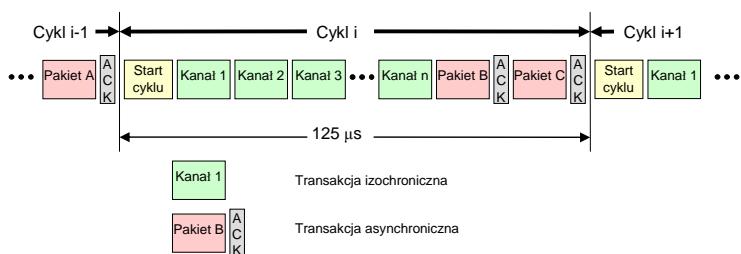


Rys. 10. Współdzielenie pasma przez transakcje izochroniczne i asynchroniczne

## 4.7 Model komunikacyjny

Podstawowe funkcje warstw protokołu komunikacyjnego:

- **Warstwa zarządzania magistralą** - konfiguracja i zarządzanie węzłem
- **Warstwa transakcji** - świadczy usługi dla transferów asynchronicznych realizując protokół żądanie-odpowiedź określony przez architekturę CSR. Wspiera 3 rodzaje operacji: odczyt, zapis i zamknięcie.
- **Warstwa połączeniowa** - wykonuje transakcję generując sekwencję pakietów w stacji nadzędnej i odbierającej.
- **Warstwa fizyczna** - zapewnia elektryczny i mechaniczny interfejs szeregowej magistrali oraz wykonuje arbitraż w celu udostępnienia łącza tylko jednemu nadawcy (ochrona przed kolizją).



Rys. 10. Współdzielenie pasma przez transakcje izochroniczne i asynchroniczne

### 4.7.1 Warstwa zarządzania magistralą

Zarządzanie magistralą obejmuje automatyczną konfigurację węzła do komunikacji oraz, opcjonalnie, zarządzanie dystrybucją zasilania.

Zarządzanie globalne obejmuje:

- przypisanie numeru kanału i pasma transferom izochronicznym
- kontrolę odstępów czasowych wykonywania transakcji izochronicznych
- kontrolę zasilania węzłów
- ”dostrajanie” magistrali w celu polepszenia wydajności
- świadczenie usług dla innych węzłów, jak np. ustawienie maksymalnej szybkości transmisji pomiędzy węzłami

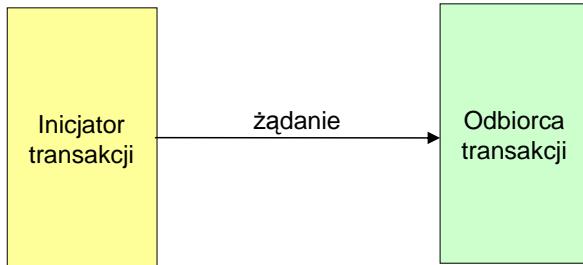
### 4.7.2 Warstwa transakcji

Świadczy usługi dla transferów asynchronicznych, które umożliwiają:

- wykonanie odczytu z urządzenia (*read*)
- wykonanie zapisu danych do urządzenia (*write*)
- wykonanie modyfikacji danych w urządzeniu (*lock*)

Inicjatorem transakcji jest warstwa aplikacji, która zleca jej wykonanie warstwie transakcji. Model wykonania transakcji wyróżnia inicjatora (*requester*) oraz odbiorcę (*responder*), a także dzieli transakcję na dwie operacje: wysłanie żądania i odebranie odpowiedzi.

Transakcja asynchroniczna z potwierdzeniem żądania i odpowiedzi (pakietów).



Rys.6. Model transakcji izochronicznej

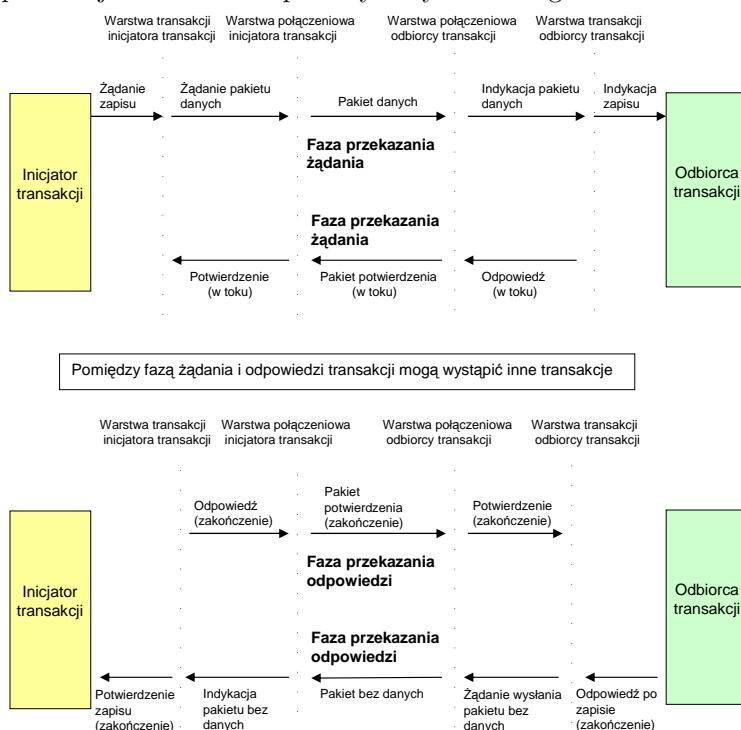
#### 4.7.3 Warstwa połączeniowa

Oparta na modelu żądanie-odpowiedź.

- Dla transferów asynchronicznych
  - Warstwa połączeniowa węzła inicjującego transakcję dokonuje translacji żądania wykonania transakcji na pakiety przekazywane do warstwy fizycznej.
  - Warstwa połączeniowa węzła odbierającego dokonuje operacji odwrotnej - przekształca odczytane z warstwy fizycznej pakiety i przekazuje je do warstwy transakcji.
- Dla transferów izochronicznych jest podobnie, tylko żądanie wykonania transakcji (w węźle inicjującym) przekazywane jest do warstwy połączeniowej bezpośrednio przez sterownik programowy, a pakiety w warstwie połączeniowej u odbiorcy przekazywane są bezpośrednio do sterownika programowego.

Warstwa połączeniowa wykonuje również transakcje specjalne:

- Transakcja dzielona (*Split Transaction*) - rozpoczęcie kolejnej transakcji przed zakończeniem poprzedniej. Pozwala na lepsze wykorzystanie magistrali i niezablokowanie jej.



Rys. 8. Działanie warstwy połączeniowej podczas transakcji dzielonej

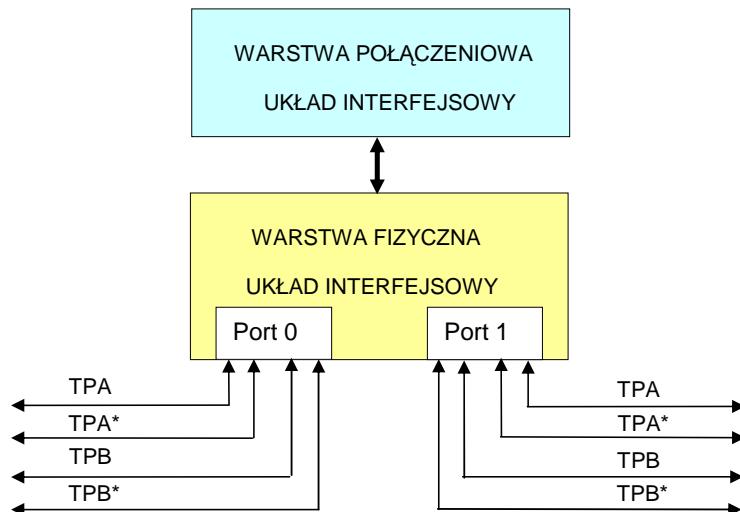
- Transakcja dołączana (concatenated transaction) - umożliwia wykorzystanie magistrali przez inne węzły podczas gdy odbiorca transakcji opóźnia odesłanie odpowiedzi. Kiedy odpowiedź będzie już gotowa, odbiorca musi na drodze arbitrażu uzyskać dostęp do łącza (co wydłuża transakcję). Jeżeli jednak odbiorca jest zdolny do szybkiego udzielania odpowiedzi, może ją dołączyć do pakietu potwierdzenia wysyłanego w fazie przekazania żądania), co sprawia, że odbiorca transakcji nie musi rywalizować o dostęp do magistrali. Jest to transakcja dołączana.
- Transakcja zespolona (Unified Transaction) - to samo co transakcja dołączana, tylko ograniczona do operacji zapisu. Umożliwia wyłączenie odpowiedzi do potwierdzenia zamykającego fazę przekazania żądania. Sprawia to, że faza przekazania odpowiedzi jest niepotrzebna.



Rys. 9. Transakcja zespolona

#### 4.7.4 Warstwa fizyczna

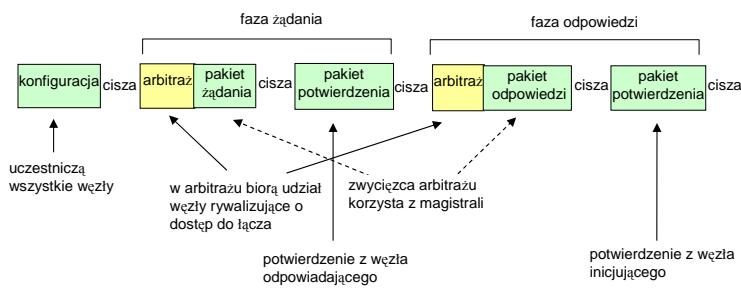
Prosty węzeł systemu IEEE 1394 z dwoma portami i liniami sygnałowymi magistrali.



Rys. 12. Porty i linie sygnałowe w węźle

Podstawowe operacje realizowane na liniach sygnałowych:

- **Konfiguracja** - wykonywana po włączeniu zasilania oraz dołączeniu urządzenia do magistrali bądź odłączeniu z niej. Przygotowuje (wszystkie) węzły systemu do wymiany informacji.
- **Arbitraż** - konieczny jeżeli w toku jest kilka transakcji realizowanych przez różne węzły. Rywalizują one o dostęp do magistrali, a konflikt rozstrzyga procedura arbitrażowa. Konieczny przed przekazywaniem pakietu żądania i odpowiedzi, niekonieczny dla pakietu potwierdzenia.



Rys. 13. Wykorzystanie magistrali przez operacje na liniach sygnałowych

- Przesył danych - prawo tego urządzenia, kto wywalczyło dostęp do magistrali. Dane kodowane są w zapisie NRZ (*Non Return to Zero*).

### Szybkość transmisji

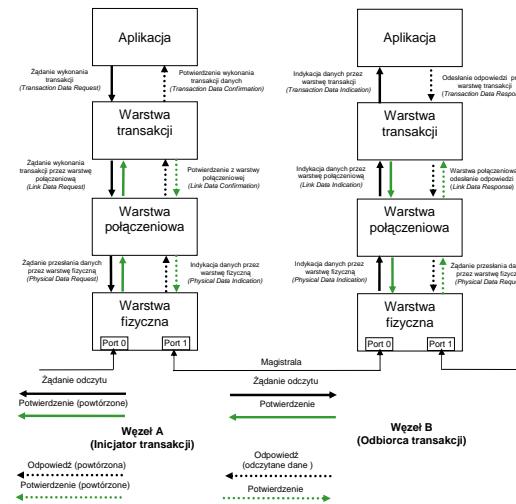
Szybkość transmisji w Mb/s	Maksymalna liczba bajtów danych w pakiecie
<b>Transfery asynchroniczne</b>	
100	512
200	1024
400	2048
800	4096
1600	8192
3200	16384
<b>Transfery izochroniczne</b>	
100	1024
200	2048
400	4096
800	8192
1600	16384
3200	32768

Minimum musi być spełnione! Szybkość transmisji może być większa, ale nie musi. Prędkości dla omawianych standardów:

- Dla IEEE 1394: S100
- Dla IEEE 1394a: S100, S200, S400
- Większe szybkości istnieją dla nowszych standardów

## Transakcja asynchroniczna

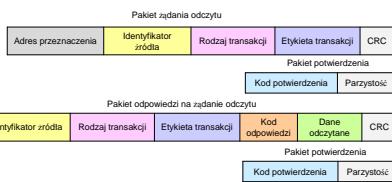
Jest to transakcja odczytu pomiędzy węzłami A i B (aplikacja w węźle B udostępnia dane tej w A).



Rys.14. Przykład asynchronicznej transakcji odczytu realizowanej pomiędzy węzłami



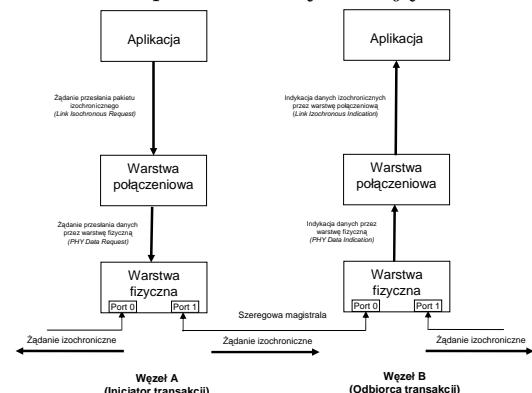
Rys.15. Struktura pakietów dla operacji zapisu



Rys.16. Struktura pakietów dla operacji odczytu

## Transakcja izochroniczna

Wykonywane pomiędzy aplikacją, a warstwą połączeniową, z pominięciem warstwy transakcji. Brak potwierdzania pakietów. Często mają charakter rozgłoszenia.



Rys.17. Przykład transakcji izochronicznej

### Pakiety w transakcjach izochronicznych.

Transakcje izochroniczne są jednokierunkowe i ograniczają się tylko do przekazania danych przez węzeł inicjujący transakcję. Odbiorca nie odsyła żadnej odpowiedzi. Znaczenie pól typowej transakcji izochronicznej:

Izochroniczny pakiet żądania

Numer kanału	Rodzaj transakcji	Dane	CRC
--------------	-------------------	------	-----

Rys.18. Struktura pakietu izochronicznego

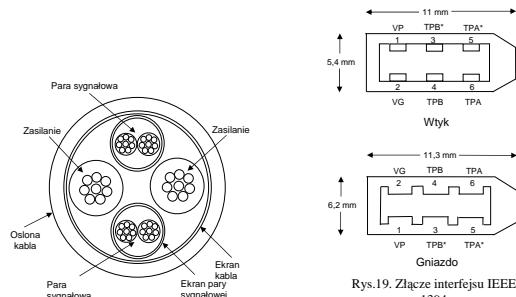
- Numer kanału (Channel Number) – węzły uczestniczące w transakcji izochronicznej wykorzystują numer kanału jako adres,
- Rodzaj transakcji(Transaction Type) – definiuje rodzaj przesyłanego pakietu jako „izochroniczny”.
- Dane (Data) – przekazywane dane,
- CRC – suma kontrolna.

## 4.8 Interfejs fizyczny

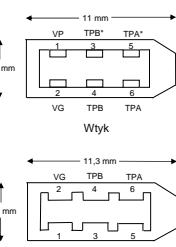
Obejmuje złącza, okablowanie oraz układ elektryczny, spełnia funkcje takie jak:

- nadawanie i odbiór sygnałów
- generacja i rozpoznawanie stanów interfejsu
- sygnalizacja maksymalnej szybkości transmisji akceptowanej przez dany port

W skład interfejsu fizycznego wchodzi również układ dystrybucji zasilania (umożliwia zasilanie węzłów, które nie mają własnych jednostek zasilających).



Rys.20. Przekrój kabla interfejsu IEEE 1394



Rys.19. Złącze interfejsu IEEE 1394

Tabela 2. Organizacja wyrowadzeń złącza IEEE 1394

Numer kontaktu	Nazwa sygnału	Opis
1	VP	Zasilanie od 8 do 40 V, zacisk +
2	VG	Masa zasilania
3	TPB*	Para skręcana B przeznaczona do: - transmisji sygnału, - odbioru strobu
4	TPB	
5	TPA*	Para skręcana A przeznaczona do: - transmisji strobu, - odbioru danych
6	TPA	

W złączach zaciski zasilania są trochę wysunięte do przodu, co sprawia, że przy wdzióbywaniu najpierw doprowadzane jest zasilanie, a potem pojawia się kontakt na liniach sygnałowych.

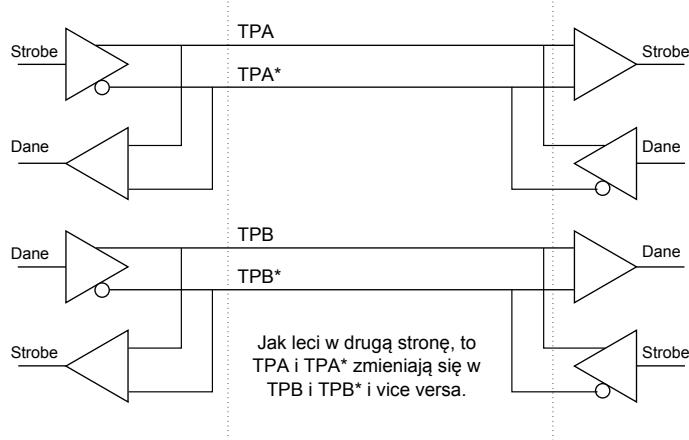
## 4.9 Interfejs sygnałowy

Interfejs sygnałowy jest bardzo rozbudowy, ponieważ służy nie tylko do transmisji i odbioru danych, ale także do realizacji funkcji arbitrażowych.

**Sygnał różnicowy** - napięcie pomiędzy TPA/TPA\*, a TPB/TPB\*.

**Stan z** - oznacza stan trafny do wykrycia (nieokreślony). Jego pojawienie się na linii odczytywane jest jako cisza.

### Uproszczony schemat układów interfejsowych



### Odczytywanie stanu na linii połączeniowej

Stan nadajnika nr 1	Stan nadajnika nr 2	Stan odczytywany na linii
1	1	1
1	0	Z
1	Z	1
0	1	Z
0	0	0
0	Z	0
Z	1	1
Z	0	0
Z	Z	Z

- TPA/TPA\* - linia do przekazywania Strobe
- TPB/TBP\* - linia do transmisji danych

Kodowanie bitów odbywa się w zapisach RZ (*Return to Zero*) oraz NRZ (*Non Return to Zero*). Poniżej odtworzony zegar w interfejsie IEEE 1394. Zegar w węźle odbierającym jest przesunięty czasowo w stosunku do sygnałów Data i Strobe na wyjściu węzła nadającego, ponieważ sygnały ulegają opóźnieniom podczas transmisji w kablu.

Przyładowy ciąg danych	1	0	0	1	1	1	0	1	0	0
Data										
Strobe	0	0	1	1	0	1	1	1	1	0
Data ⊕ Strobe										

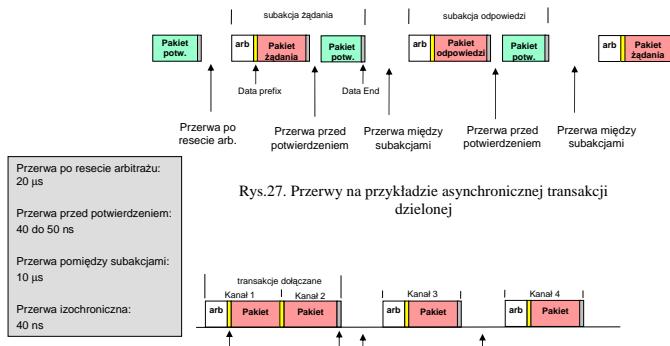
Rys.26. Odtworzenie zegara nadawcy z sygnałów Data i Strobe

Zasada: w wyniku sumowania XOR bitów Data i Strobe uzyskać ciąg występujących na przemian 0 i 1, czemu odpowiada sygnał impulsowy o współczynniku wypełnienia 0.5 i półokresie odpowiadającym bitowej szybkości transmisji.

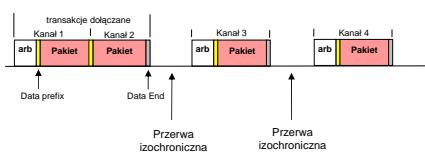
**Sygnalizacja arbitrażu** obejmuje 5 funkcji:

- reset magistrali
- identyfikacja drzewa
- identyfikacja węzła
- normalny arbitraż
- pakiety startowy i końcowy
- sterowanie stanem portu (od wersji 1394a)

**Przerwy czasowe** - pomiędzy poszczególnymi fazami składającymi się na transakcję potrzebne są przerwy czasowe podczas których magistrala pozostaje w stanie jałowym.



Rys.27. Przerwy na przykładzie asynchronicznej transakcji dzielonej



Rys.28. Przerwy pomiędzy transakcjami izochronicznymi

- **Przerwa przed potwierdzeniem** - czas pomiędzy zakończeniem asynchronicznego pakietu (żądania lub odpowiedzi), a rozpoczęciem transmisji pakietu potwierdzenia.
- **Przerwa izochroniczna** - przeznaczona dla kanałów izochronicznych, zapewnia stan jałowy przed rozpoczęciem arbitrażu węzłów izochronicznych. krótsza niż przerwa dla kanałów asynchronicznych.
- **przerwa pomiędzy subakcjami** - dla kanałów asynchronicznych, zapewnia stan jałowy przed rozpoczęciem arbitrażu węzłów izochronicznych. Nie występuje między żądaniem a odpowiedzią w przypadku transmisji dołączanej.
- **Przerwa po resecie arbitrażu** - poprzedza "Interwał równych szans".

#### 4.10 Detekcja szybkości transmisji

Ze względu na szybkość transmisji wyróżnia się 3 rodzaje portów:

- transmisja tylko z szybkością podstawową S100
- transmisja z szybkością S100 lub S200
- transmisja z szybkością S100, S200 lub S400

Gdzie

- $S100 = 98.304 \text{ Mb/s}$
- $S200 = 196.608 \text{ Mb/s}$
- $S400 = 393.216 \text{ Mb/s}$

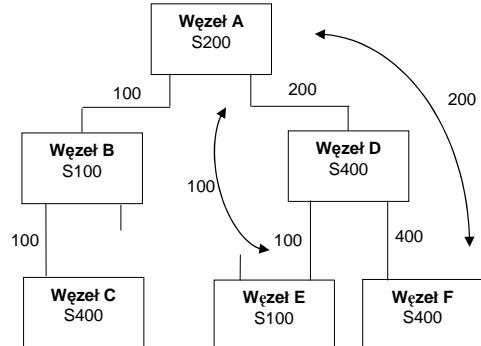
Tylko szybkość S100 jest obligatoryjna dla węzłów. Pozostałe dwa mogą, ale muszą być dostępne. Jeżeli jednak są, to wymagane są układy sygnalizacji szybkości transmisji.

Informacja o możliwych szybkościach transmisji każdego węzła przekazywana jest do innych węzłów rezydujących na magistrali podczas konfiguracji (w szczególności informacja o maksymalnej szybkości transmisji przekazywana jest do węzła podłączonego do tego samego segmentu kabla).

Zasadą jest nieprzekazywanie przez węzeł pakietów otrzymanywanych z daną szybkością do podłączonych do niego węzłów o mniejszej szybkości.

Węzeł może mieć kilka portów, przez niektóre porty może być połączony z węzłami o szybkości większej od podstawowej, a przez inne z węzłami o szybkości podstawowej (zatem możliwa jest sytuacja, gdy węzeł odbiera pakiet z szybkością większą od podstawowej, przekazywany przez inny "szybki" węzeł, ale

nie będzie go mógł przekazać do węzłów "wolniejszych").



Rys.29. Wpływ topologii systemu na szybkość transmisji

## 4.11 Dystrybucja zasilania

Węzły w systemie IEEE 1394 mogą być źródłem zasilania dla innych węzłów, korzystać z zasilania przez inne węzły lub też wykorzystywać zasilanie własne bez udostępniania go innym węzłom.

### 4.11.1 Klasa zasilania

Węzły wysyłają informację o zasilaniu w ramach pakietu „samoidentyfikacji” (*self-ID*) podczas procesu konfiguracyjnego. Informacja ta przedstawia zapotrzebowanie na zasilanie lub możliwość dostarczenia zasilania.

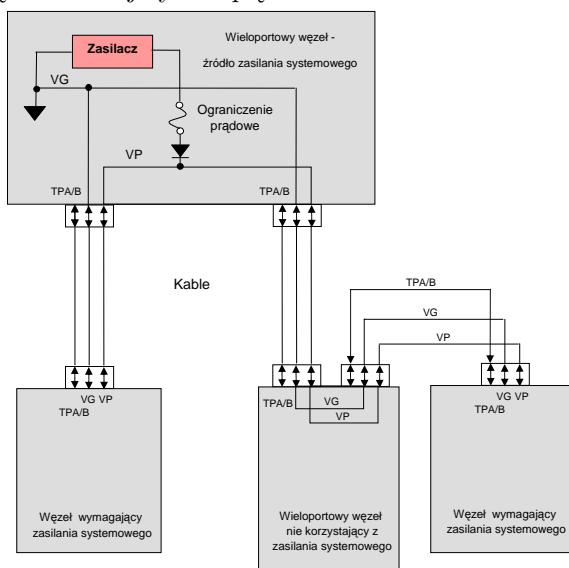
Informacje o zasilaniu przekazywane są na 3 bitowym polu **pwr** w pakiecie self-ID.

1	0	Phy_ID	0	L	Gap_cnt	sp	del	c	<b>pwr</b>	P0	P1	P2	i	m
---	---	--------	---	---	---------	----	-----	---	------------	----	----	----	---	---

Rys.30. Pole *pwr* w pakiecie self-ID do przekazywania informacji o zasilaniu

### 4.11.2 Dystrybucja zasilania

Jeden lub kilka węzłów może jednocześnie być źródłem zasilania. Ponieważ napięcia zasilania udostępniane przez węzły mogą się różnić konieczna jest dioda zapobiegająca przepływowi prądu zasilania do węzła o mniejszym napięciu.



Rys.31. Dystrybucja zasilania

## 4.12 Arbitraż

W FireWire węzły rywalizują o prawo do nadawania w ramach arbitrażu. Arbitraż realizowany jest w okresach przerw w transmisji (*gaps*) i polega na:

- Zagwarantowaniu pasma dla kanałów izochronicznych,
- Równoprawnej rywalizacji w przypadku kanałów asynchronicznych.

Arbitraż rozpoczyna się po rozpoznania przez węzeł okresu jałowego magistrali, co oznacza koniec poprzedniej transmisji. Czasy trwania jałowych stanów magistrali są różne dla transakcji izochronicznych i asynchronicznych i wynoszą:

- Przerwa po resecie arbitrażu:  $20 \mu s$
- Przerwa pomiędzy subakcjami:  $10 \mu s$
- Przerwa przed potwierdzeniem: 40 ns
- Przerwa izochroniczna - od 40 ns (min) do 50 ns (max),
- Przerwa pomiędzy subakcjami transakcji asynchronicznej i przerwa przed potwierdzeniem mogą być „regulowane” (aby arbitraż rozpoczynał się jak najszybciej).

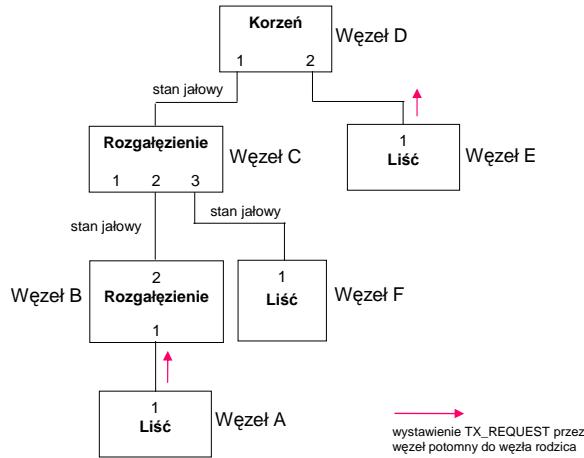
Sposób sygnalizowania arbitrażu (protokół sygnalizacji arbitrażu) jest taki sam dla transakcji asynchronicznych i izochronicznych.

### 4.12.1 Rodzaje arbitrażu

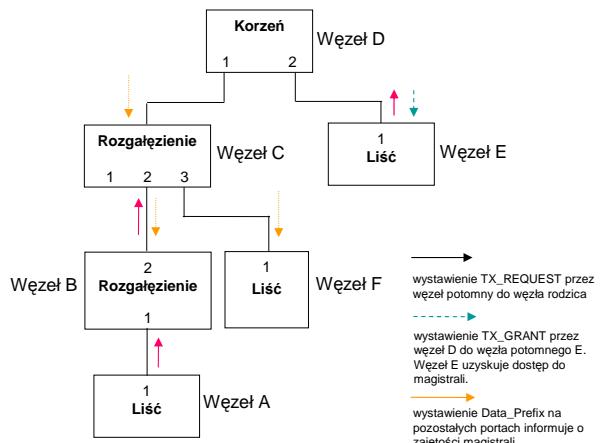
Typ pakietu, który jest przygotowywany do transmisji, determinuje rodzaj żądania wykorzystywanego przez warstwę połączeniową. Są 4 rodzaje:

- arbitraż oparty na równej szansie dostępu do łącza (*fair arbitration service*) stosowany przy transmisji pakietu asynchronicznego
- arbitraż uwzględniający priorytety (*priority arbitration service*) stosowany przy transmisji pakietu cycle strat lub pakietu asynchronicznego o wyższym priorytecie
- arbitraż natychmiastowy (*immediate arbitration service*) stosowany przy transmisji pakietu potwierdzenia
- arbitraż izochroniczny (*isochronous arbitration service*) stosowany przy transmisji pakietu izochronicznego

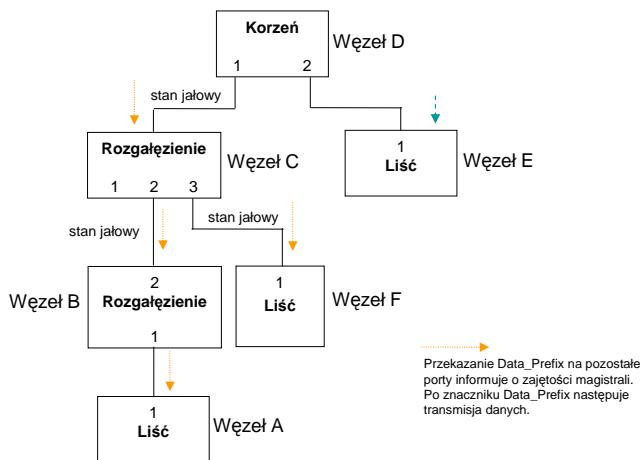
#### 4.12.2 Przykład arbitrażu



Rys.32 a. Dwa węzły A i E żądają dostępu do magistrali (rozpoczęcie arbitrażu)



Rys.32 b. Sygnalizacja arbitrażu dociera do korzenia. Korzeń na port 2 wystawia TX\_GRANT



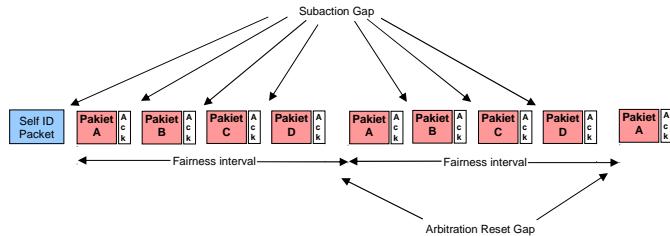
Rys.32 c. Usunięcie TX\_REQUEST kończy arbitraż.  
Węzeł E może rozpoczęć transmisję danych po znaczniku Data\_Prefix.

#### 4.12.3 Arbitraż asynchroniczny

##### Zasada równych szans

- Węzły asynchroniczne do wykonania transakcji nie wymagają alokacji pasma.
- Działanie węzłów asynchronicznych polega na rotacyjnym przekazywaniu priorytetu (*fairness interval* - „interwał równych szans”).
- Węzły asynchroniczne, które mają rozpoczęte transakcje uzyskują dostęp do magistrali w celu wykonania asynchronicznej subakcji (przesłania jednego pakietu).
- Kolejność przydziału zależy od położenia węzła w drzewie systemu. Węzły położone bliżej korzenia uzyskają dostęp przed węzłami bardziej oddalonymi.
- Łączny czas wykonania subakcji przez wszystkie węzły nazywa się *fairness interval*.
- W tym czasie każdy węzeł uzyska dostęp do magistrali, aby wykonać subakcję.
- Dostęp do magistrali węzeł zarządzający przekazuje „rotacyjnie”.

- Następna subakcja będzie mogła być wykonana w kolejnym interwale równych szans.



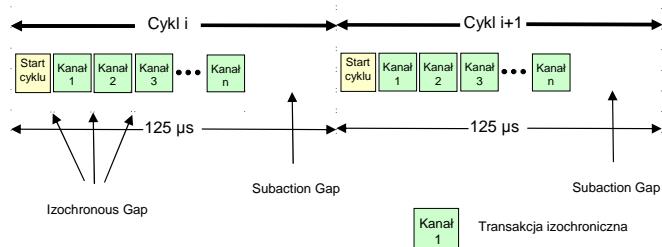
Rys.33. Znaczenie przerw w asynchronicznym arbitrażu

### Odesłanie potwierdzenia, dostęp natychmiastowy

- Węzeł, który zainicjował asynchroniczną transakcję może sam być odbiorcą transakcji zainicjowanej przez inny węzeł.
- Warstwa połączeniowa węzła musi potwierdzić odbiór pakietu poprzez odesłanie potwierdzenia korzystając z usługi *Immediate Arbitration Service (IAS)* warstwy PHY.
- Warstwa PHY po otrzymaniu żądania usługi *IAS* nie rywalizuje o dostęp do magistrali i bezpośrednio po wykryciu przerwy *Acknowledge Gap* (0.04 do 0.05  $\mu s$ ) sygnalizuje warstwie LINK, że jest gotowa do przesłania pakietu potwierdzenia.
- Warstwa połączeniowa przekazuje pakiet potwierdzenia do portu za pośrednictwem warstwy PHY.

#### 4.12.4 Arbitraż izochroniczny

- Arbitraż izochroniczny rozpoczyna się natychmiast po rozgłoszeniu pakietu *Cycle Start*, traktowanego jako znacznik początku cyklu.
- Po czasie 0.04 ms węzły izochroniczne pragnące uzyskać dostęp do magistrali rozpoczynają rywalizację o łącze.
- Węzeł, który wygra arbitraż wykonuje transakcję, po której magistrala powraca do stanu jałowego. Stan jałowy magistrali trwający przez 0.04 ms traktowany jest jako przerwa izochroniczna (*Isochronous Gap*), po której rozpoczyna się arbitraż kolejnych węzłów izochronicznych zgłaszających żądanie transmisji.
- Przerwa izochroniczna trwa tyle samo, co *Acknowledge Gap* dla transakcji asynchronicznych i jest znacznie mniejsza od *Subaction Gap*. W ten sposób żaden węzeł asynchroniczny nie uzyska dostępu do magistrali zanim wszystkie węzły izochroniczne nie wykonają swoich transakcji.
- Cykl izochroniczny trwa 125  $\mu s$ . Poszczególne węzły izochroniczne w ramach cyklu mają przydzielone pasmo (tzw. kanał), które jest częścią interwału 125  $\mu s$ .
- Za przydział pasma odpowiedzialny jest węzeł zarządzający zasobami izochronicznymi. Jeżeli węzły izochroniczne nie wykorzystują pełnych 125 ms, w ramach wolnego czasu realizowane są transakcje asynchroniczne.

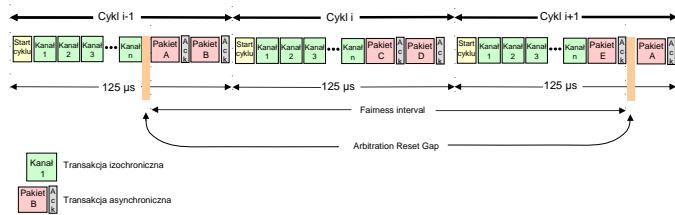


Rys. 34. Arbitraż transakcji izochronicznych

Pakiet *Cycle Start* generowany jest co 125  $\mu s$  przez węzeł będący korzeniem systemu.

#### 4.12.5 Łączenie arbitraży izochronicznego i asynchronousznego

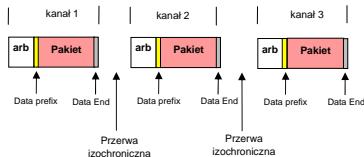
- W ramach wolnego miejsca w interwale 125  $\mu s$  niewykorzystanego przez transakcje izochroniczne wykonywane są transakcje asynchronousne.
- 80% cyklu przydzielone jest transakcjom izochronicznym, a tylko 20% transakcjom asynchronousnym.
- Transakcje izochroniczne wykonywane są w kolejnych 125  $\mu s$  cyklach (wartość nominalna), natomiast transakcje asynchronousne w ramach interwału równych szans, którego długość zależy od liczby węzłów asynchronousnych jednocześnie żądających dostępu do łączza.



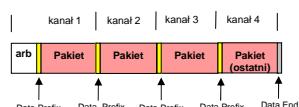
Rys. 35. Arbitraż transakcji izochronicznych i asynchronousnych

### 4.13 Pakiety

- **Pakiety izochroniczne** - Pakiety izochroniczne transmitowane są jako grupowe lub rozgłoszeniowe do jednego lub więcej węzłów.
- Transakcje izochroniczne:



Rys.37. Normalne transakcje izochroniczne



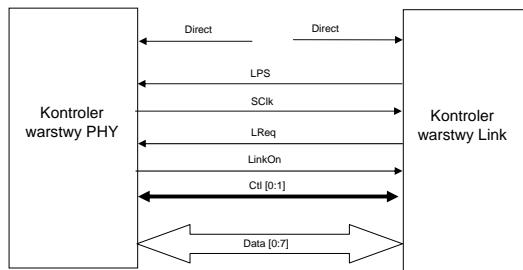
Rys.38. Izochroniczne transakcje dołączane wysyłane z jednego węzła

- **Pakiety warstwy fizycznej**

- Standard IEEE 1394-1995 definiuje 3 pakiety PHY:
  - \* Self Identification (Self-ID)
  - \* Link-On
  - \* PHY Configuration
- Wersja 1394a rozszerza zestaw o pakiety:
  - \* Ping
  - \* Remote Access
  - \* Remote Reply
  - \* Remote Command
  - \* Resume
- Elementem procesu konfigurującego system IEEE-1394 do komunikacji jest nadanie każdemu węzlowi unikatowego identyfikatora oraz poinformowanie pozostałych węzłów systemu o możliwościach transmisyjnych danego węzła.

- Te operacje wykonywane są za pomocą pakietów Self-ID rozgłoszanych przez węzły bezpośrednio po resecie i określaniu topologii systemu (identyfikacji drzewa).

## 4.14 Interfejs LINK-PHY



Rys.43. Interfejs pomiędzy warstwami Link i PHY

## 4.15 Ponowienie transakcji

Wyróżniono dwa sposoby ponowienia transmisji pakietu:

- jednofazowa repetycja (*single phase retry*),
- dwufazowa repetycja (*dual phase retry*).

Obsługa błędów jest zadaniem warstw transakcji lub aplikacji, przy czym obowiązują tu następujące zasady:

- Reakcja na przekłamanie pakietu przez węzeł odbiorcy (przekłamanie pakietu przychodzącego). Błąd wykrywa warstwa liniowa węzła odbierającego pakiet i zawiadamia o tym warstwę transakcji. Warstwa transakcji nie przekazuje uszkodzonego pakietu do aplikacji, natomiast wysyła stosowne potwierdzenie negatywne do węzła będącego źródłem pakietu.
- Reakcja na powiadomienie o przekłamaniu pakietu przez węzeł nadawcy. Węzeł będący źródłem pakietu dowiaduje się o jego przekłamaniu po odebraniu potwierdzenia negatywnego. Warstwa aplikacji tego węzła jest odpowiedzialna za ewentualne powtórzenie transakcji.

## 4.16 Konfiguracja magistrali

### 4.16.1 Reset magistrali

**Przyczyny resetu:**

- włączenie nowego urządzenia do systemu,
- odłączenie urządzenia,
- włączenie/wyłączenie zasilania urządzenia oraz błędy zasilania, jak np. jego chwilowe zaniki. (chodzi tu o zasilanie warstwy fizycznej węzła; włączenie/wyłączenie zasilania warstw połączeniowej i wyższych nie jest przyczyną resetu)
- inicjalizacja programowa, na żądanie aplikacji.

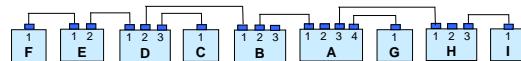
Reset jest „rozgłoszany” do wszystkich węzłów systemu.

Rezultatem Resetu jest:

- Zwolnienie magistrali (przejście wszystkich segmentów do stanu jałowego);
- Inicjalizacja niektórych rejestrów warstwy PHY oraz pewnych pól w wybranych rejestrach warstwy PHY, co prowadzi do usunięcia wszystkich ustawień konfiguracyjnych portu (w szczególności identyfikatorów *parent* lub *child*, które informują o topologii systemu oraz identyfikatorów węzła);

- Inicjalizacja wybranych rejestrów CSR dokonywana przez warstwę LINK (LINK jest powiadamiana o resecie przez PHY).

#### 4.16.2 Identyfikacja drzewa systemu (Tree\_ID)

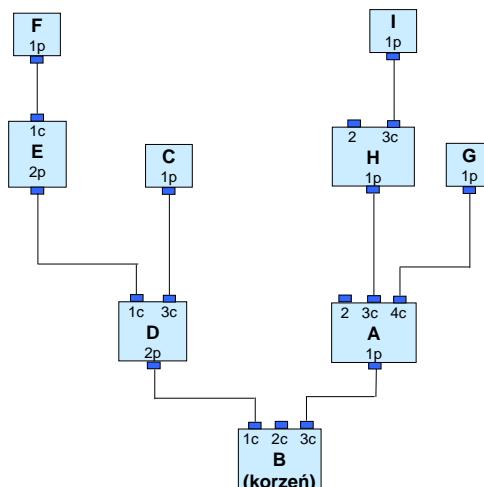


Rys.45. Przykładowy system bezpośrednio po resecie

Identyfikacja drzewa oparta jest na stanach interfejsu określanych przez połączone ze sobą porty, a wykrywanych przez komparatory arbitrażowe.

W procesie wykorzystywane są następujące stany interfejsu:

- Parent\_Notify: STPA=0, STPB = Z, czyli 0Z
- Child\_Notify: STPA=1, STPB = Z, czyli 01



Rys.46. Rezultat działania procedury Tree\_ID

#### 4.16.3 Samoidentyfikacja (Self\_ID)

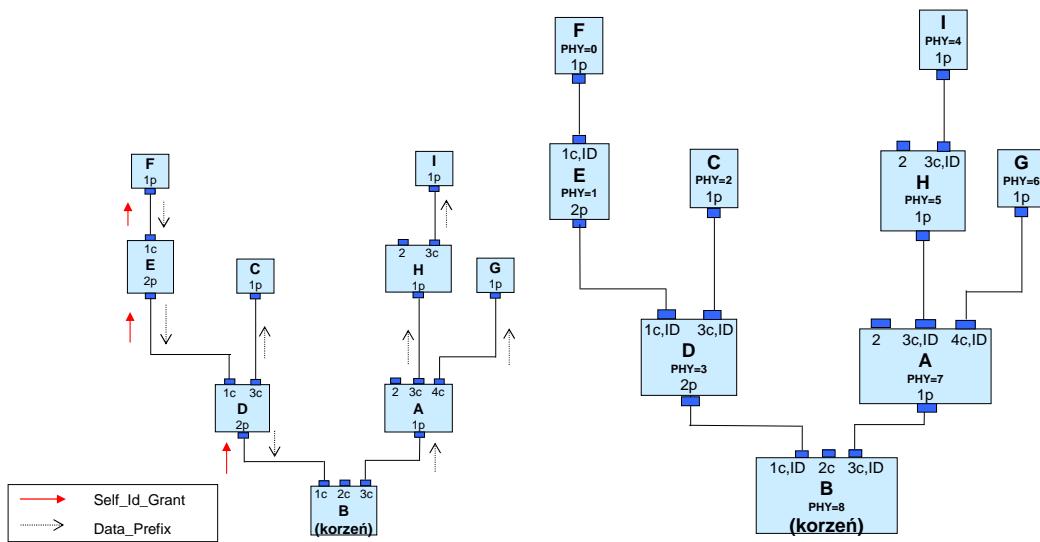
Samoidentyfikacja oparta jest na stanach interfejsu określanych przez połączone ze sobą porty, a wykrywanych przez komparatory arbitrażowe.

W procesie wykorzystywane są następujące stany interfejsu:

- Self\_Id\_Grant (Arb\_Grant): STPA=Z, STPB =0, czyli Z0
- Data\_Prefix: STPA=0, STPB =1, czyli 01
- Identification\_Done: STPA=1, STPB =Z, czyli 1Z

**Rezultatem procedury Self.Id jest:**

- przypisanie każdemu węźlowi unikatowego identyfikatora pełniącego rolę adresu,
- rozesłanie pakietów self.id z każdego portu do wszystkich pozostałych węzłów, które zawierają informacje o możliwościach i wymaganiach komunikacyjnych danego węzła,
- przypisanie każdemu portowi znacznika ID (zidentyfikowany) oznaczającego, że w segmencie magistrali, z którego korzysta dany port została określona szybkość transmisji.



Rys.47. Rozpoczęcie procesu Self\_Id

Rys.48. System po zakończeniu procesu Self\_Id

## 4.17 Zarządzanie zasilaniem

### 4.17.1 Najważniejsze cechy systemu zasilania

:

- Urządzenia mogą posiadać zasilanie własne (lokalne) lub korzystać z zasilania systemowego. Możliwe jest udostępnianie zasilania (nieregulowane napięcie z zakresu 8 do 40 VDC) innym urządzeniom za pośrednictwem magistrali systemowej. Część układów urządzenia może być zasilana ze źródła lokalnego, a inne wykorzystywać zasilanie systemowe.
- Zasilanie lokalne powinno być separowane za pośrednictwem diody od przewodów dystrybucji zasilania systemowego.
- Każde urządzenie musi informować o zapotrzebowaniu na zasilanie lub możliwości jego udostępnienia innym urządzeniom za pośrednictwem pakietu Self\_ID przekazywanego w procesie konfiguracji systemu.
- Bezpośrednio po resecie, urządzeniu wolno pobierać z zasilania systemowego do 1 W. Po zakończeniu konfiguracji systemu, kiedy wiadomo, jaka moc jest dostępna oraz jakie jest zapotrzebowanie urządzeń na zasilanie, możliwe jest przydzielenie urządzeniu większej mocy. Kalkulację mocy w systemie przeprowadza Zarządcy magistrali. Pod nieobecność Zarządcy magistrali, Zarządcy zasobów izochronicznych jest odpowiedzialny za rozesłanie pakietów Link\_ON włączających zasilanie w węzłach, które korzystają z zasilania systemowego.
- Do przekazywania zasilania służy 6 kontaktowe złącze z zaciskami VP i VG.

### 4.17.2 W dokumencie Power Distribution wyróżniono

:

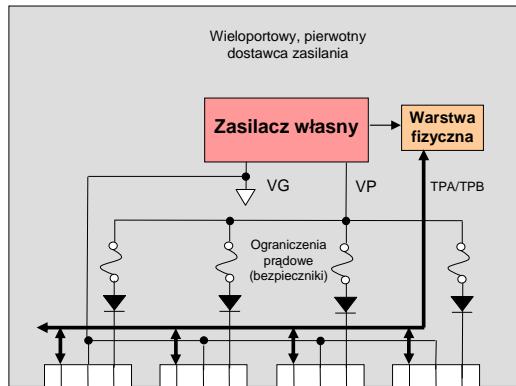
- dostawcę zasilania (*Power Provider*)
- alternatywnego dostawcę zasilania (*Alternate Power Provider*)
- konsumenta zasilania (*Power Consumer*)
- węzłów z własnym źródłem zasilania

#### 4.17.3 Dostawca zasilania

Dostawca zasilania udostępnia innym węzłom zasilanie pochodzące z własnego źródła wykorzystując do tego celu 6 kontaktowy kąt na każdym porcie.

Napięcie na zaciskach zasilania w portach dostawcy jest stałe i nieregulowane (zakres 22 do 33 V) oraz nie zależy od żadnych zdarzeń na magistrali sygnałowej.

Dostawcy zasilania deklarują klasę zasilania 1,2 lub 3 w zależności od mocy, którą są w stanie dostarczyć.



Rys.49. Dostawca zasilania w systemie FireWire

#### 4.17.4 Alternatywny dostawca zasilania

Dostawca alternatywny (DA) jest podklassą dostawcy zasilania.

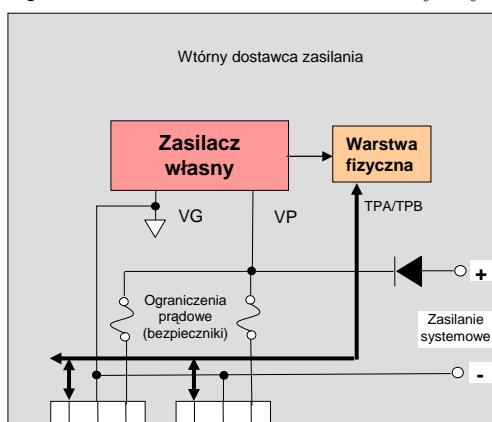
DA dostarcza mniejszą moc od dostawcy pierwotnego, i obowiązują go mniejsze restrykcje.

DA ze złączy 6 kontaktowych.

Zaleca się żeby DA posiadał 2 porty.

W przypadku awarii zasilania własnego, dostawca alternatywny może korzystać z zasilania systemowego w celu podtrzymania funkcjonowania warstwy fizycznej i przekazywania ruchu pomiędzy portami.

W pakiecie Self\_ID dostawca alternatywny raportuje klasę zasilania 4



Rys.50. Dwuportowy, alternatywny dostawca zasilania w systemie FireWire

#### 4.17.5 Konsument zasilania

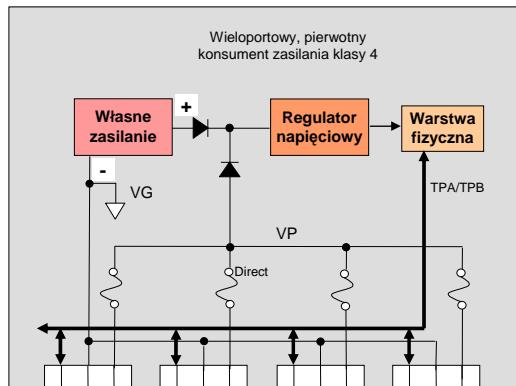
Konsument zasilania (KZ) wyposażony jest tylko w jeden port 6 kontaktowy, przez który pobiera prąd zasilania (pozostałe porty, o ile istnieją, muszą być 4 kontaktowe).

KZ raportuje w pakiecie Self\_ID klasę zasilania 4,6 lub 7.

Konsument zasilania mogą pobierać prąd zasilania z systemu lub zachowywać się „neutralnie”, tzn. wykorzystywać własne źródło zasilania nie udostępniając go innym urządzeniom.

Od konsumenta, który korzysta z własnego źródła zasilania wymaga się, aby przekazywał zasilanie systemowe pomiędzy swoimi portami, a w przypadku węzłów wieloportowych, wymaga się również obecności

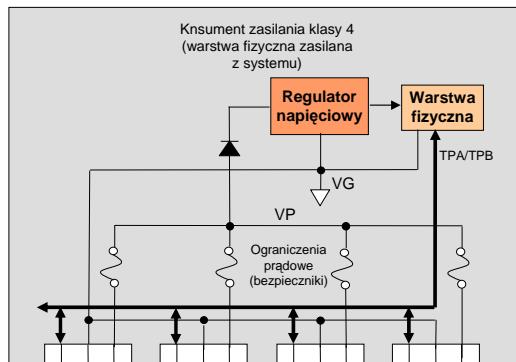
ograniczenia prądowego na każdym porcie. Przekazywanie zasilania pomiędzy portami musi być blokowane, gdy węzeł ma wyłączone zasilanie warstwy fizycznej.



Rys.51. Konsument zasilania klasy 4

Najprostszym konsumentem zasilania jest węzeł, którego warstwa fizyczna zasilana jest bezpośrednio z magistrali.

Układy wykonujące funkcje wyższych warstw protokołowych korzystają na ogół z zasilania własnego.



Rys.52. Konsument zasilania klasy 4 z zasilaniem warstwy fizycznej z magistrali

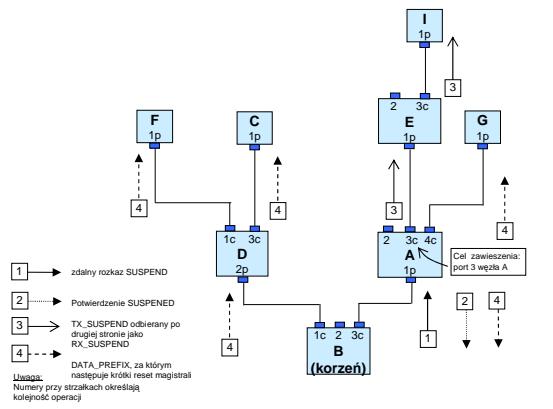
#### 4.17.6 Zawieszenie portu

Zawieszenie portu (*port suspend*) powoduje blokadę ruchu w porcie, a w konsekwencji również blokadę ruchu w całej gałęzi (domenie urządzeń) obsługiwanej przez ten port.

Zawieszane urządzenie wymaga tylko nieznacznej mocy do podtrzymywania swojej konfiguracji. Ze stanu zawieszenia można powrócić do normalnej aktywności realizując tzw. wznowienie (*resume*).

Sposoby zawieszenia portu:

- stosownym rozkazem zdalnym lub lokalnym,
- poprzez detekcję stanu RX\_SUSPEND interfejsu,
- w konsekwencji wykrycia stanu RX\_SUSPEND na innym porcie tego samego węzła,
- poprzez detekcję stanu RX\_DISABLE\_NOTIFY interfejsu.



Rys.53. Zdalne zawieszenie portu

#### 4.17.7 Wznowienie

Wznowienie (*Resume*) jest powrotem portu do normalnego działania po uprzednim zawieszeniu. Wyróżniono 3 sposoby wznowienia:

- globalne, realizowane poprzez rozmówczenie pakietu wznowienia do wszystkich węzłów
- selektywne, realizowane za pomocą zdalnego pakietu PHY, kierowane do konkretnego, zablokowanego lub zawieszzonego portu
- z inicjatywy węzła, w którym znajduje się zawieszony port, po wystąpieniu zdarzenia, którego obsługa wymaga przywrócenia komunikacji na zawieszonym porcie

## 5 Bluetooth

### 5.1 Właściwości

- Bezprzewodowy system komunikacyjny
- Krótki zasięg zestawiony ”ad-hoc”
- *Pikonet*: sieć personalna (WPAN) oparta o master-slave
  - Można wyróżnić slave'y aktywne i zaparkowane
  - Dwa rodzaje połączeń: z aktywnym i zaparkowanym
  - Połączenie z zaparkowanym slave'ie nie wymienia informacji, ale utrzymuje informacje o urządzeniu i aktualizuje jego status (np. jak urządzenie wyjdzie poza zasięg pikonetu to nie przestaje być uwzględniane).
- Możliwości:
  - Współpraca osobistych urządzeń użytkownika
  - Podłączenie urządzeń do sieci (przez punkty dostępowe np.)
  - Automatycznie się kompliluje, wykonuje procedury konfiguracyjne przygotowujące system do komunikacji.
  - Automatyczne:
    - \* Wyszukiwanie urządzeń (*inquiry*) - stwierdzanie, że w zasięgu znalazło się nowe urządzenie i wykonuje odpowiednią konfigurację by włączyć je do pikonetu. Może to być wykonywane zarówno przez mastera, jak i slave (?).
    - \* Podłączenie do mastera (*paging*) - jak pojawia się zgoda na bycie wykrytym, tworzy połączenie fizyczne pozwalające na przesył danych między M-S.
    - \* Wykrywanie usług
- Bezpieczeństwo komunikacji: uwierzytelnianie urządzeń i szyfrowanie (*pairing*)
- Może współpracować w środowiskach silnie zakłóconych (dla systemów w bliskim sąsiedztwie i działających na tym samym paśmie (chyba))

### 5.2 Konfiguracja

1. Wyszukiwanie urządzeń (*inquiry*)
  - Master cyklicznie wysyła komunikat *inquiry*. Istnieje możliwość wyszukiwania urządzeń tylko określonego typu.
  - Urządzenia wykrywalne po rozpoznaniu *inquiry* odsyłają kilkakrotnie unikatowy adres BT, ale dopiero po odczekaniu losowego czasu. Niewykrywalne po prostu ignorują ten komunikat.
  - Format:
    - MASTER *Inquiry B* → urządzenie BT - rozgłoszenie (*broadcast*)
    - MASTER *Adres ← R* urządzenie BT - losowo opóźniona odpowiedź
  - Jak urządzenie nie zostaje wykryte, to brak odpowiedzi.
2. Podłączenia urządzeń (*paging*)
  - *Paging* wykonywany jest sekwencyjnie na wyznaczonych do tego urządzeniach. Służy m.in. do synchronizacji urządzeń w zakresie **czsu, fazy, aktualnego przeskoku częstotliwości**.
  - Urządzenie po wykonaniu pagingu jest w stanie CONNECTION (podłączenie do mastera) i staje się członkiem pikonetu.
  - Format:
    - MASTER *Page →* urządzenie BT
    - MASTER *Connect B ←* urządzenie BT
  - Niewykryte urządzenie pozostaje poza zasięgiem.

### 5.3 Przeskoki częstotliwości

- Łącze zostaje podzielone na sloty o czasie trwania  $625\mu s$
- Przesyłanie informacji parami:
  - Przesyłanie  $M \rightarrow S$  odbywa się w szczelinach parzystych
  - Szczeliny nieparzyste zajęte są na przekazanie odpowiedzi
- Brak informacji o takcie między urządzeniami - master i slave potrzebują oboje zegara do wyznaczania czasu szczelelin.
  - W czasie pagingu pojawia się ta informacja - przekazana jako jeden z parametrów pagingu (slave dostosowuje się do mastera)
  - każdy pakiet na bieżąco przekazuje informacje o takcie - **w każdym slocie obowiązuje inną częstotliwość**
- Master określa sekwencję przeskoków podczas normalnej transmisji
- Slave określa sekwencję przeskoków podczas pagingu
- Pasmo ISM: 2.4 do 2.483 GHz
- Ograniczone dostępne moce: 1; 2.5; 100 [mW].

### 5.4 Parametry

#### 5.4.1 Moc nadajników BT

- Klasa 1: 100 mW (20 dBm), zasięg  $\approx 100$  m
- Klasa 2: 2.5 mW (4 dBm), zasięg  $\approx 10$  m
- Klasa 3: 1 mW (0 dBm), zasięg  $\approx 1$  m

#### 5.4.2 Modulacja

GFSK (Gaussian Frequency Shift Keying)

- Stan logiczny 1 - przesunięcie częstotliwości o 115 kHz w górę względem nośnej
- Stan logiczny 0 - przesunięcie częstotliwości o 115 kHz w dół względem nośnej

#### 5.4.3 Szybkość bitowa

- 1 Mb/s dla GFSK (możliwa do osiągnięcia szybkość „średnia”: 723,4 kb/s)
- 2 Mb/s dla pi/4-QPSK
- 3 Mb/s dla 8-DPSK

### 5.5 Adres urządzenia

#### 5.5.1 Format

Lower Address Part	Non-significant Address Part	Upper Address Part
24 bity	16 bity	8 bitów

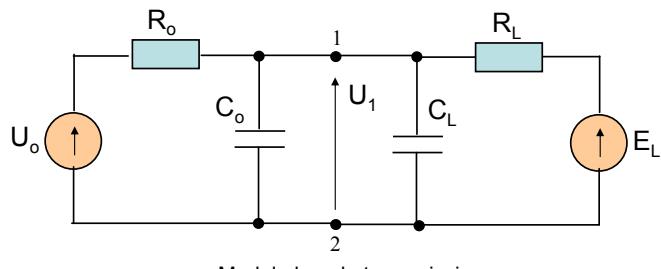
Łącznie 48 bitów.

## 5.5.2 Właściwości

- Adres BT jest unikalny globalnie
- Oprócz adresu BT wyróżnia się:
  - AM\_ADDR - adres aktywny w pikonecie
  - 2 adresy urządzenia zaparkowanego:
    - \* PM\_ADDR
    - \* AM\_ADDR

## 5.6 Transmisja

### 5.6.1 Transmisja synchroniczna

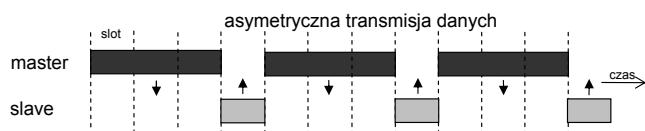


#### Poziomy sygnałów

$ U_o  < 25 \text{ V}$	Sygnały danych:
$3 \text{ k} \Omega < R_o < 7 \text{ k} \Omega$	$-15 \text{ V} < U_1 < -3 \text{ V}$ 1 logiczna
$I_{zwarcia} < 0,5 \text{ A}$	$+3 \text{ V} < U_1 < +15 \text{ V}$ 0 logiczne
$ E_L  < 2 \text{ V}$	Sygnały sterujące:
$C_o + C_L < 2500 \text{ pF}$	$-15 \text{ V} < U_1 < -3 \text{ V}$ 0 logiczne
Zmiana $U_1 < 30 \text{ V/us}$	$+3 \text{ V} < U_1 < +15 \text{ V}$ 1 logiczna
<b>Jeden nadajnik - 1 odbiornik</b>	

Dla każdego pakietu jest przewidziany dokładnie taki sam przedział czasu

### 5.6.2 Transmisja asynchroniczna



Przedział czasowy obejmuje więcej niż 1 pakiet, maksymalnie do 5 pakietów.

## 5.7 Rodzaje danych i łącz

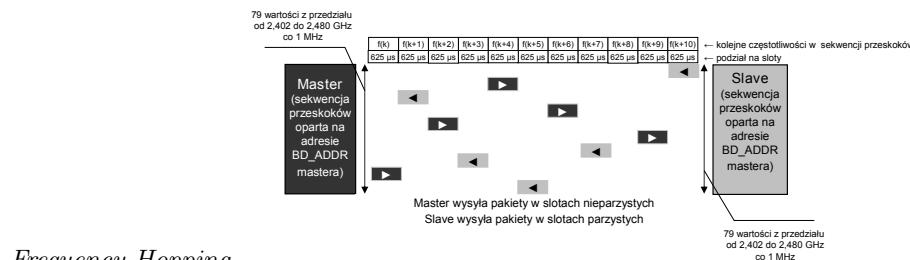
### 5.7.1 Rodzaje danych transmitowanych w BT

- Asynchroniczne
- Synchroniczne
- Izochroniczne

### 5.7.2 Rodzaje łącz

- Łącze ACL (bezpołączniowe - *Connectionless*)
  - przekazywanie rozkazów zarządzania łączem fizycznym (ACL-C)
  - przekazywanie asynchronicznych danych użytkownika (ACL-U)
- Łącze SCO (połączniowe - *Connection Oriented*)
  - Cykliczne przekazywanie danych w zarezerwowanych szczelinach czasowych (stały odstęp między kolejnymi pakietami danego transferu)
  - Liczba kolejnych szczelin czasowych przydzielonych na transmisję jednego pakietu może wynosić 1, 3 lub 5 (zależy od rodzaju pakietu wybranego dla danego łączka).
- Łącze eSCO (połączniowe - *Connection Oriented*)
  - podobne do SCO
  - możliwość retransmisji uszkodzonego pakietu (w tzw. oknie retransmisji)

## 5.8 Przeskoki częstotliwości



Frequency Hopping

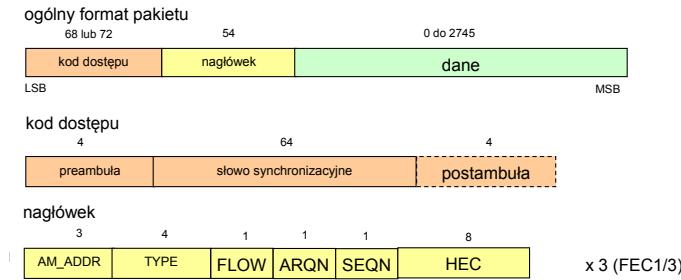
### 5.8.1 Właściwości

- Występują pomiędzy szczelinami czasowymi
- Sekwencja przeskoków musi być znana wszystkim urządzeniom w zasięgu pikonetu
- sekwencja przeskoków oparta na adresie BD\_ADDR mastera
- 79 wartości z przedziału od 2.402 do 2.480 GHz co 1 MHz
- Zastosowanie pseudolosowych sekwencji przeskoków - zmniejsza to ryzyko zakłóceń.
- Im krótsze tym szybsza jest retransmisja informacji.

### 5.8.2 Adaptacyjne przeskoki częstotliwości

- Polega na wykluczaniu częstotliwości (zakresów pasm) urządzeń uszkodzonych, takich, które często przekłamują pakiety. Np. zakłócanych przez inną pobliską sieć pikonet, albo źródło sygnału wi-fi.

## 5.9 Pakiety



Rys. 9. Standardowy format pakietu  
(rozmiary pól w bitach)

### 5.9.1 Nagłówek

Zabezpieczony przez x3 FEC1/3, czyli bity są trzykrotnie duplikowane. (*tryplikowane? XD*)

### 5.9.2 Kod dostępu

Preambuła: 0101 lub 1010

Słowo synchronizacyjne:

- generowane jest na podstawie 24-bitowego LAP
- długość: 64 bity

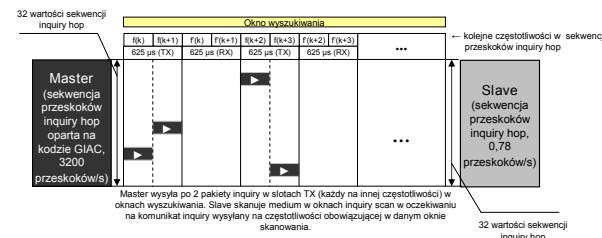
Postambuła: 0101 lub 1010

Pakiety ID nie mają pola postambuły

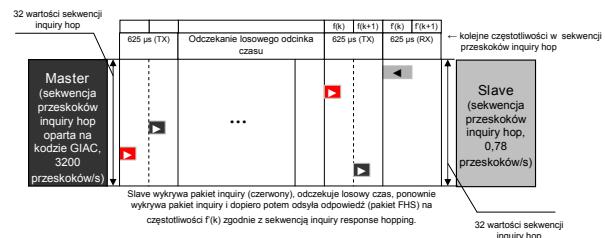
## 5.10 Zegar

- CLKN - natywny (własny) urządzenia (*native clock*)
- CLKE - przybliżony (*estimated clock*)
- CLK - systemowy oparty na zegarze natywnym mastera (*master clock*)

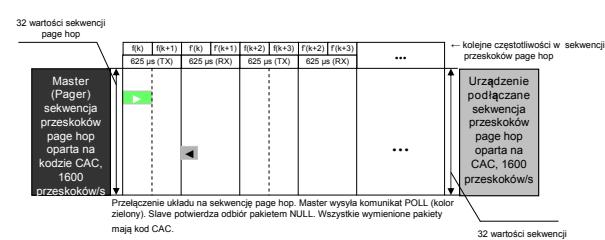
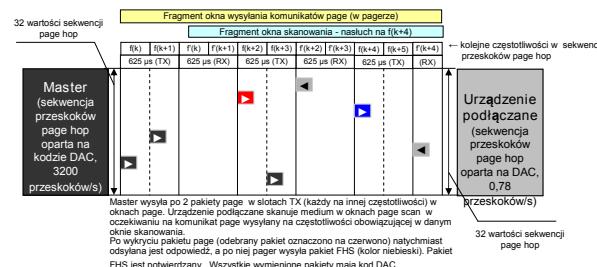
## 5.11 Inquiry i Paging



Rys. 13. Generacja komunikatów inquiry



Rys. 14. Wykrycie inquiry i odeslanie odpowiedzi



Generuje komunikaty *inquiry*. Jak urządzenie odbierze komunikat, to ma obowiązek zwrócenia odpowiedzi. W stałych odstępach czasu wysyła odpowiedź kilkukrotnie - wynika to z tego, że Master może być zajęty innymi zadaniami, okno wyszukiwań otwiera tylko raz na jakiś czas. Slave może cały wysiłek poświęcić na to, by Master-senpai go zauważał.

Rezultatem jest wykrycie urządzenia i pobranie pewnych parametrów, jeżeli master jest chce.

Paging: każdy kolejny pakiet jest szyfrowany innym kluczem.

## Literatura

- [1] Wojciech Mielczarek, *USB Uniwersalny Interfejs Szeregowy*, Helion, Gliwice 2005
- [2] Wojciech Mielczarek, *Komputerowe systemy pomiarowe, standardy IEE-288.2 i SCPI*, Wydawnictwo Politechniki Śląskiej, Gliwice 2002
- [3] Wojciech Mielczarek, *Szeregowy interfejs cyfrowy FireWire*, Wydawnictwo Politechniki Śląskiej , Gliwice 2010
- [4] Wojciech Mielczarek, *Szeregowe interfejsy cyfrowe*, Helion, Gliwice 1997