INTERFAȚA USB

1. Scopul lucrării

Scopul principal al lucrării este prezentarea magistralei USB şi a interfeței care se bazează pe această magistrală. Lucrarea prezintă noțiunile de bază ale interfeței USB, conectorii şi cablurile utilizate, tipurile de transfer, modelul comunicației, protocolul magistralei, structura pachetelor, descriptorii USB şi comenzile USB. De asemenea, lucrarea introduce clasa dispozitivelor HID şi descrie modul în care se poate realiza comunicația între calculator şi dispozitivele din această clasă.

2. Considerații teoretice

2.1. Prezentare generală a magistralei USB

Elaborarea magistralei USB (*Universal Serial Bus*) a început în anul 1995 de către un grup de firme care cuprindea *Compaq*, *Digital*, *IBM*, *Intel*, *Microsoft*, *NEC* şi *Northern Telecom*. Aceste firme au fost reunite în asociația *USB Implementers Forum* (http://www.usb.org), care a publicat prima versiune a specificațiilor USB. Această asociație, care a fost extinsă cu un număr mare de firme, continuă să actualizeze standardele pentru magistrala USB, pentru controlerele acestei magistrale şi pentru diferitele categorii de periferice prevăzute cu această magistrală.

Unul din scopurile elaborării magistralei USB a fost simplificarea interconexiunilor dintre calculator şi periferice, prin reducerea numărului de cabluri care se conectează la calculator şi utilizarea aceluiaşi tip de conector pentru diferite categorii de periferice. Într-un sistem conținând o magistrală USB, diferitele periferice se pot conecta în serie sau într-o topologie sub formă de stea pe mai multe nivele, un singur periferic fiind conectat la un port USB al calculatorului gazdă. Un alt aspect care s-a avut în vedere la elaborarea magistralei USB a fost asigurarea unei rate de transfer mai ridicate decât ratele de transfer permise de porturile seriale şi paralele. Deşi la primele versiuni ale magistralei USB rata maximă de transfer era de numai 12 Mbiți/s, această rată a crescut în mod semnificativ (până la 480 Mbiți/s) la versiunea 2.0 a magistralei USB. De asemenea, s-a urmărit ca perifericele să poată fi adăugate în mod simplu la calculator, fără deschiderea carcasei acestuia, fără oprirea tensiunii de alimentare şi fără reîncărcarea sistemului de operare.

Calculatorul prevăzut cu o magistrală USB detectează adăugarea unui nou periferic şi determină în mod automat care sunt resursele necesare acestuia, inclusiv driverul software şi rata de transfer. Unul din perifericele calculatorului, de exemplu, tastatura sau monitorul, se conectează la conectorul USB al calculatorului. Alte periferice se conectează la un distribuitor ("*hub*") aflat în cadrul tastaturii sau monitorului, fiind posibilă realizarea unor conexiuni sub formă de arbore. Perifericele se pot afla la o distanță de până la 5 m unele față de altele sau față de un distribuitor. În total, se pot conecta până la 127 de periferice USB la un calculator, acestea fiind alimentate cu o tensiune de +5 V prin cablul USB. Toate perifericele USB utilizează un conector standard, eliminându-se necesitatea utilizării unor conectori diferiți pentru fiecare tip de periferic.

Versiunea 1.0 a standardului USB a fost publicată în anul 1996, aceasta fiind urmată de versiunea 1.1, adoptată în anul 1998. Rata de transfer maximă specificată de aceste versiuni este de 12 Mbiți/s. Această rată de transfer este suficientă pentru periferice cum sunt unitatea de disc flexibil, telefonul sau difuzorul digital (prevăzut cu un convertor digital-analogic). Pentru periferice lente, cum este tastatura, mouse-ul, creionul optic sau tableta grafică, a fost prevăzut un canal de viteză redusă, cu rata de transfer de 1,5 Mbiți/s. Versiunea curentă a standardului USB este 2.0, care a fost publicată în anul 2000. Această versiune permite creşterea ratei de transfer cu un factor de 40 față de versiunea 1.1, de la 12 Mbiți/s până la 480 Mbiți/s. Această extensie a versiunii 1.1 a specificațiilor USB presupune

utilizarea aceloraşi cabluri, conectori şi interfeţe software. Utilizatorii pot beneficia însă de o gamă suplimentară de periferice cu performanţe ridicate, de exemplu, camere video digitale, scanere, imprimante, adaptoare ISDN, unităţi de discuri magnetice şi unităţi de discuri optice.

Deşi toate calculatoarele actuale sunt echipate cu porturi USB, acestea nu au înlocuit conectorii standard ai calculatoarelor. Monitorul, tastatura, mouse-ul, porturile paralele şi cele seriale utilizează, în continuare, conectori separaţi. Un motiv ar putea fi arhitectura mai complexă a magistralei USB, care trebuie să permită conectarea unor tipuri variate de periferice. Un alt motiv ar fi că perifericele care trebuie să conţină un distribuitor, cum este monitorul sau tastatura, devin mai complexe şi mai costisitoare.

Elementele principale ale unui sistem care utilizează magistrala USB sunt dispozitivele USB, cablurile USB şi programele de sistem. Dispozitivele de pe magistrala USB sunt conectate fizic la calculatorul gazdă utilizând o topologie sub formă de stea pe mai multe nivele, după cum se ilustrează în Figura 11.1.

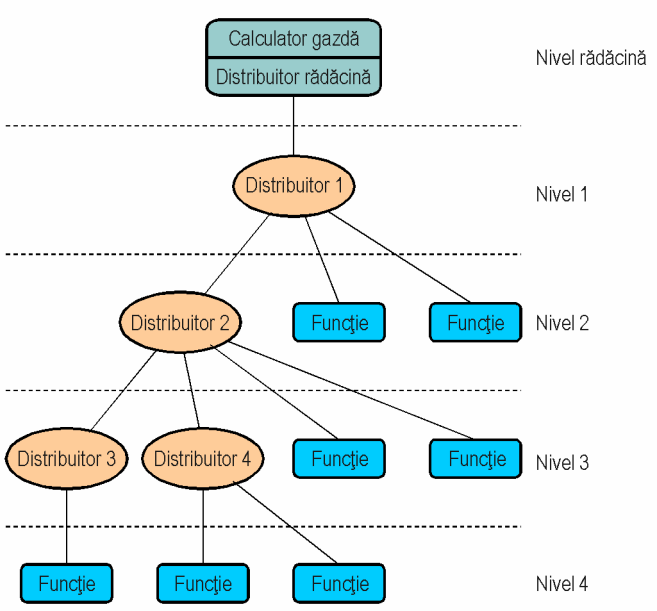


Figura 11.1. Topologia magistralei USB.

Există două categorii de dispozitive USB: distribuitoare şi funcţii. Un *distribuitor* reprezintă o clasă specială de dispozitiv USB, care asigură puncte de conectare suplimentare pentru alte dispozitive USB. Aceste puncte de conectare se numesc porturi. Calculatorul gazdă conţine un distribuitor rădăcină, prin care asigură unul sau mai multe puncte de conectare. În plus, acest distribuitor conţine controlerul magistralei USB. Fiecare magistrală USB are un singur controler de magistrală.

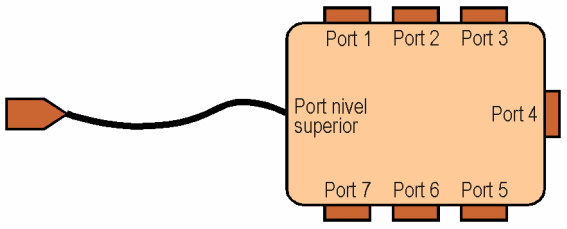


Figura 11.2. Porturile unui distribuitor USB tipic.

Figura 11.2 prezintă un distribuitor USB tipic. Unul din porturile distribuitorului permite conectarea la calculatorul gazdă sau la un distribuitor de pe nivelul superior al topologiei. Fiecare din celelalte şapte porturi permit conectarea la un distribuitor sau la o funcţie de pe nivelul inferior. Distribuitoarele pot fi conectate în cascadă până la cinci nivele. Distribuitorul recunoaşte conectarea dinamică a unui periferic şi asigură o putere de cel puţin 0,5 W pentru fiecare periferic în timpul iniţializării. Sub controlul programului de sistem, distribuitorul poate asigura o putere suplimentară

pentru funcționarea perifericelor, până la 2,5 W (un curent de 0,5 A). Unele periferice, cum este tastatura, mouse-ul sau creionul optic, pot fi alimentate numai cu tensiunea furnizată de cablul magistralei, în timp ce altele pot avea o sursă proprie de alimentare.

Un distribuitor constă din două părți: un controler şi un repetor. Controlerul conține registre de interfață pentru comunicația cu calculatorul gazdă. Comenzile de stare şi de control permit calculatorului gazdă configurarea distribuitorului, monitorizarea şi controlul porturilor sale. Repetorul este un comutator controlat prin protocol între portul de nivel superior şi porturile de nivel inferior. De asemenea, repetorul monitorizează semnalele de pe porturi şi gestionează tranzacțiile care îi sunt adresate. Toate celelalte tranzacții sunt repetate la dispozitivele ataşate. Fiecare port de nivel inferior poate fi validat individual şi poate fi conectat la dispozitive cu viteză ridicată sau cu viteză redusă. Porturile cu viteză redusă sunt izolate de semnalele cu viteză ridicată.

Figura 11.3 ilustrează modul în care distribuitoarele USB asigură conectarea într-un sistem de calcul tipic.

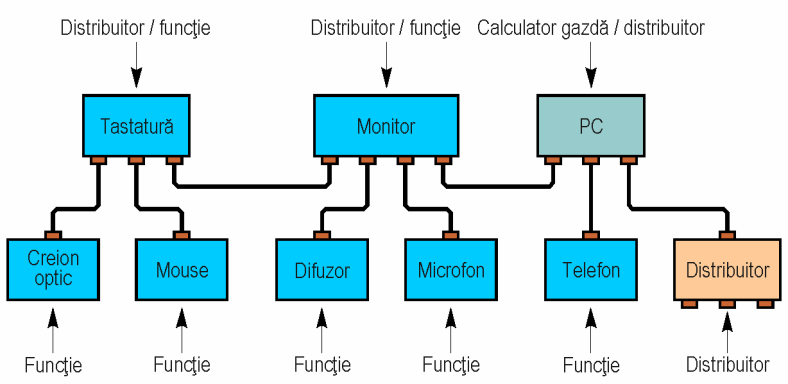


Figura 11.3. Utilizarea distribuitoarelor USB într-un sistem de calcul tipic.

O *funcție* este un dispozitiv USB care poate transmite sau recepționa date sau informații de control pe magistrală. Acest dispozitiv trebuie să răspundă la cererile de tranzacție transmise de calculatorul gazdă. O funcție este implementată în mod obişnuit ca un periferic separat conectat printr-un cablu la un port al unui distribuitor. Un singur dispozitiv fizic poate conține însă funcții multiple. De exemplu, o tastatură şi un dispozitiv de trasare pot fi combinate într-un singur dispozitiv fizic. În cadrul unui asemenea dispozitiv compus, funcțiile individuale sunt ataşate la un distribuitor, iar acest distribuitor intern este conectat la magistrala USB.

Fiecare funcție conține informații de configurație care descriu posibilitățile sale şi resursele necesare. Aceste informații sunt transmise calculatorului gazdă ca răspuns la o tranzacție de control. Înaintea utilizării unei funcții, aceasta trebuie configurată de calculatorul gazdă. Această configurare presupune alocarea unei lățimi de bandă în cadrul magistralei USB şi selectarea opțiunilor specifice de configurație.

Programele de sistem asigură o tratare uniformă a sistemului de I/E de către toate programele de aplicații. Prin ascunderea detaliilor de implementare hardware, se asigură portabilitatea programelor de aplicații. Programele de sistem gestionează conectarea şi deconectarea dinamică a perifericelor. Faza de conectare, numită *enumerare*, implică comunicarea cu perifericele pentru determinarea driverului de dispozitiv care trebuie încărcat şi asignarea unei adrese unice fiecărui periferic. În timpul funcționării, calculatorul inițiază tranzacții cu anumite periferice. Informațiile sunt transmise pe magistrala USB sub forma unor pachete, care sunt recepționate de toate perifericele. Pachetele conțin o adresă a perifericului destinație; doar acest periferic va accepta o anumită tranzacție şi va răspunde în mod corespunzător.

Specificațiile magistralei USB 2.0 (numită şi *Hi-Speed* USB) descriu o magistrală cu performanțe îmbunătățite. Conectoarele şi cablurile conforme cu specificațiile USB 1.1 permit obținerea ratelor de transfer mai ridicate ale magistralei USB 2.0 fără nici o modificare. Rata de transfer maximă care se poate obține este de 480 Mbiți/s.

Perifericele USB 2.0 cu viteze de transfer superioare sunt conectate la un distribuitor USB 2.0. Un distribuitor USB 2.0 acceptă tranzacții de viteză ridicată şi furnizează datele cu ratele corespunzătoare perifericelor USB 2.0 şi perifericelor USB 1.1. Vitezele de transfer ridicate sunt

negociate cu fiecare periferic, iar dacă un periferic nu permite o viteză ridicată, legătura cu acest periferic va funcționa la viteza mai redusă de 12 Mbiți/s sau 1,5 Mbiți/s. Aceasta implică o complexitate mai ridicată a distribuitoarelor şi necesitatea memorării temporare a datelor recepționate. Un distribuitor USB 2.0 va avea porturi de ieşire pentru transferuri cu viteză ridicată şi porturi de ieşire pentru transferuri cu viteză redusă.

Programele de sistem recunosc posibilitățile avansate ale perifericelor USB 2.0, astfel încât pot optimiza performanțele. Aceste programe pot detecta configurațiile de conectare care nu sunt optime. O asemenea configurație se obține, de exemplu, prin conectarea unui periferic USB 2.0 la un distribuitor USB 1.1, caz în care perifericul USB 2.0 va funcționa la viteza corespunzătoare versiunii USB 1.1. De asemenea, dacă într-un lanț de periferice USB 2.0 se conectează un distribuitor USB 1.1, acesta va limita viteza întregului sistem. Programele de sistem pot recomanda utilizatorului o configurație optimă de conectare a perifericelor.

Controlerele magistralei USB, aflate pe placa de bază a calculatorului gazdă, au propriile specificații. În cazul versiunii 1.1 a magistralei USB, existau două specificații pentru aceste controlere. Prima dintre ele, *Universal Host Controller Interface* (UHCI), a fost elaborată de firma *Intel* şi permitea simplificarea circuitelor, partea mai complexă fiind cea de software. A doua specificație, *Open Host Controller Interface* (OHCI), a fost elaborată de firmele *Compaq*, *Microsoft* şi *National Semiconductor*, această specificație permițând simplificarea programelor, partea mai complexă fiind cea de hardware. Odată cu introducerea versiunii 2.0 a magistralei USB, a fost necesară elaborarea unei noi specificații pentru controlerele de magistrală. Această specificație, numită *Enhanced Host Controller Interface* (EHCI), a fost elaborată de mai multe firme, printre care *Intel*, *Compaq*, *NEC*, *Microsoft* şi *Lucent Technologies*.

Există specificații separate pentru diferite categorii (clase) de periferice USB. O clasă USB reprezintă un grup de periferice sau interfețe cu atribute sau servicii similare. De exemplu, două periferice sau interfețe sunt plasate în aceeaşi clasă dacă utilizează şiruri de date cu acelaşi format pentru comunicația cu calculatorul gazdă. Dintre clasele de periferice USB se amintesc următoarele: memorii de masă, monitoare, dispozitive de interacțiune cu utilizatorul (HID – *Human Interface Device*), imprimante, dispozitive audio, dispozitive de comunicație (modemuri, telefoane analogice şi digitale, adaptoare de rețea), dispozitive de captare a imaginilor fixe (camere digitale).

2.2. Conectori şi cabluri

La interfața USB se utilizează patru tipuri de conectori: două tipuri de fişe, amplasate la capetele unui cablu USB, şi două tipuri de mufe, amplasate în cadrul unui distribuitor sau periferic. Fişele şi mufele pot fi de tip A sau de tip B.

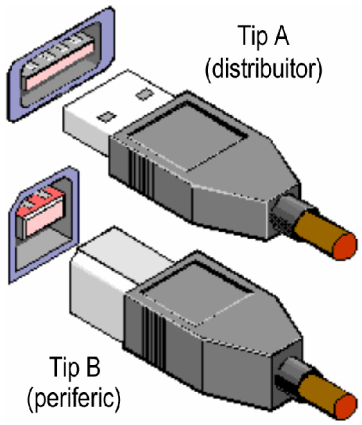


Figura 11.4. Fişe şi mufe utilizate la interfața USB.

Distribuitoarele (de exemplu, cele din calculator) conțin o mufă rectangulară cu patru pini de tip A. Perifericele se conectează la o asemenea mufă dintr-un distribuitor utilizând o fişă rectangulară de tip A (Figura 11.4). Toate cablurile care sunt ataşate permanent la periferice conțin o fişă de tip A. De obicei, perifericele se conectează însă printr-un cablu detaşabil. Aceste periferice conțin o mufă pătrată de tip B, iar cablul care conectează aceste periferice la un distribuitor conține o fişă de tip B la

capătul care se conectează la periferic şi o fişă de tip A la capătul care se conectează la distribuitor. În acest fel, nu este posibilă conectarea incorectă a cablului.

Specificaţiile USB 2.0 au fost modificate ulterior publicării acestora pentru a include o fişă şi o mufă de tip B de dimensiuni mai reduse. Aceşti conectori, numiţi *mini-B*, conţin cinci contacte şi se utilizează pentru echipamentele mobile de dimensiuni reduse, cum sunt calculatoarele PDA (*Personal Digital Assistant*) sau telefoanele mobile. Echipamentele respective conţin o mufă de tip mini-B, iar cablurile utilizate pentru conectarea acestor echipamente la un calculator PC conţin o fişă de tip mini-B la un capăt şi o fişă de tip A la celălalt capăt. Figura 11.5 ilustrează o fişă de tip mini-B alăturată cu o fişă de tip A.

Figura 11.5. Fişă USB de tip mini-B alăturată cu o fişă de tip A.

Specificaţiile USB "On-The-Go" (OTG), care au fost elaborate ca o extensie a specificaţiilor USB 2.0 pentru conectarea directă a unor echipamente mobile, fără utilizarea unui calculator PC, conţin descrierea unor fişe mini-A, mufe mini-A şi mufe mini-AB. Utilizarea acestor conectori este necesară deoarece la conectarea directă a unor echipamente mobile unul din echipamente va avea rolul unui calculator gazdă. Specificaţiile USB OTG descriu şi diferite tipuri de cabluri care utilizează conectori de dimensiuni reduse sau o combinaţie între un conector de dimensiuni reduse şi unul de dimensiuni normale.

Pentru transferul semnalelor şi a tensiunii de alimentare pe magistrala USB, se utilizează un cablu cu patru fire, ilustrat în Figura 11.6. Semnalele diferenţiale de date se transmit pe liniile *D*+ şi *D*–, formate din două fire răsucite. Semnalul de ceas este transmis codificat împreună cu datele. Codificarea utilizată este numită NRZI (*Non Return to Zero Invert*). În cazul acestei metode, biţii de 1 şi 0 sunt reprezentaţi prin tensiuni opuse şi alternante înalte şi joase, fără a exista revenirea la tensiunea de referinţă (zero) între biţii codificaţi. Sunt inseraţi biţi suplimentari pentru a asigura tranziţii suficiente ale semnalelor transmise, în scopul asigurării sincronizării. Fiecare pachet de date este precedat de un câmp de sincronizare pentru a permite receptorilor sincronizarea ceasurilor de recepţie.

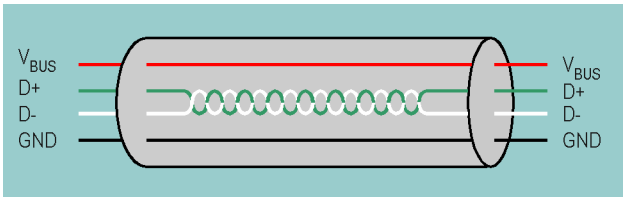


Figura 11.6. Cablul USB.

Cablul USB transmite şi tensiunea de alimentare pentru periferice pe liniile V_{BUS} şi *GND*. Tensiunea pe linia V_{BUS} este de +5 V la sursă. Pentru a asigura nivele de tensiune garantate la intrarea perifericelor şi o impedanţă de terminare corespunzătoare, se utilizează terminatori la fiecare capăt al cablului. Terminatorii permit, de asemenea, detectarea conectării sau deconectării unui periferic şi diferenţierea între perifericele cu viteză ridicată şi cele cu viteză redusă. La capătul cablului din partea distribuitorului, se utilizează terminatori formaţi din rezistenţe de 15 KΩ, prin intermediul cărora liniile *D*+ şi *D*– ale cablului sunt conectate la masă. La capătul cablului din partea perifericului, se utilizează ca terminator o rezistenţă de 1,5 KΩ, prin intermediul căreia una din liniile *D*+ sau *D*– este

conectată la o tensiune cuprinsă între 3 V şi 3,6 V. În cazul unui periferic USB cu viteză normală (12 Mbiți/s), rezistența este conectată la linia *D*+, iar în cazul unui periferic cu viteză redusă, rezistența este conectată la linia *D*–. Pentru un periferic cu viteză ridicată, valoarea rezistenței de la capătul cablului din partea perifericului este de 90 Ω.

Există două tipuri de cabluri USB. Primul tip se utilizează pentru comunicațiile de viteză redusă (1,5 Mbiți/s). Acest tip de cablu trebuie să fie ataşat permanent de echipamentul care îl utilizează. Al doilea tip de cablu este destinat pentru comunicațiile cu viteza de 12 Mbiți/s şi 480 Mbiți/s. Acest tip de cablu poate fi ataşat permanent de echipament sau poate fi detaşabil. Deosebirea dintre cele două tipuri de cabluri este că, în cazul cablului pentru viteza redusă, nu este necesară utilizarea unor fire răsucite (standardul recomandă însă utilizarea unor asemenea fire). Un cablu detaşabil trebuie să fie terminat cu o fişă de tip A la un capăt şi cu o fişă de tip B sau mini-B la celălalt capăt. Standardul USB specifică diferite caracteristici electrice ale cablurilor utilizate pentru comunicațiile de viteză ridicată.

Pentru identificarea simplă a firelor din cablurile USB, standardul specifică utilizarea culorilor din Tabelul 11.1 pentru aceste fire. Acest tabel indică şi asignarea pinilor conectorilor la semnalele magistralei USB.

Tabelul 11.1. Asignarea pinilor conectorilor la semnalele magistralei USB şi culorile firelor cablurilor.

Nr. pin	Semnal	Culoare
1	V_{BUS}	Roşu
2	D-	Alb
3	D+	Verde
4	GND	Negru

Observație

- Nu este posibilă interconectarea a două calculatoare printr-un cablu USB obişnuit. Chiar dacă s-ar utiliza un cablu cu doi conectori de tip A, prin interconectarea a două calculatoare ar exista două controlere USB într-un sistem, ceea ce nu este permis. Există însă cabluri speciale care conțin o punte USB sub forma unui circuit integrat, prin intermediul căruia este posibilă comunicația între cele două calculatoare gazdă.

2.3. Tipuri de transfer

Arhitectura USB permite patru tipuri de transferuri de date: de control, de întrerupere, de date voluminoase şi izocrone.

Transferurile *de control* se utilizează de driverele calculatorului gazdă pentru configurarea dispozitivelor care sunt ataşate la sistem. Alte drivere pot utiliza transferuri de control în moduri specifice implementării.

Transferurile *de întrerupere* se utilizează pentru date cu volum redus. Transferul acestor date poate fi solicitat de un dispozitiv în orice moment, iar rata de transfer pe magistrala USB nu poate fi mai redusă decât cea specificată de dispozitiv. Datele pentru care se utilizează transferurile de întrerupere constau din notificarea unor evenimente, din caractere sau coordonate care sunt organizate pe unul sau mai mulți octeți. Un exemplu îl reprezintă coordonatele de la un dispozitiv indicator (mouse, tabletă grafică). Datele interactive pot avea anumite limite ale timpului de răspuns care trebuie asigurate de magistrala USB.

Transferurile *de date voluminoase* ("*bulk*") se utilizează cu periferice cum sunt memorii de masă, imprimante sau scanere. Aceste date sunt secvențiale. Fiabilitatea transferurilor este asigurată la nivel hardware prin utilizarea unui cod detector de erori şi reluarea unui transfer cu erori de un număr de ori. Rata de transfer în cazul acestor transferuri poate varia în funcție de alte activități de pe magistrală.

Transferurile *izocrone* (*isos* – egal, *chronos* – timp) se utilizează pentru datele care trebuie furnizate cu o anumită rată de transfer constantă şi a căror sincronizare trebuie garantată. Izocron are semnificația "cu durată egală" sau "care apare la intervale regulate". Datele izocrone sunt generate în timp real şi trebuie furnizate cu rata cu care sunt recepționate pentru a păstra sincronizarea lor. Pe

lângă rata de transfer impusă, pentru datele izocrone trebuie respectată şi întârzierea maximă cu care acestea sunt furnizate. Furnizarea la timp a datelor izocrone este asigurată cu preţul unor pierderi potenţiale în şirul de date. Cu alte cuvinte, erorile de transmisie nu sunt corectate prin mecanisme hardware, de exemplu, prin retransmiterea lor. În concluzie, transferurile izocrone se caracterizează prin furnizarea la timp a datelor şi prin lipsa retransmiterii lor în cazul unor erori, deoarece datele întârziate nu mai sunt utile. Spre deosebire de transferurile izocrone, transferurile asincrone se caracterizează prin faptul că fiabilitatea transmiterii datelor este mai importantă decât asigurarea sincronizării. Pentru aceasta se utilizează retransmiterea datelor în cazul unor erori, chiar dacă apar întârzieri din această cauză.

Un exemplu tipic de date izocrone este reprezentat de imaginile video. Dacă rata de transfer a acestor şiruri de date nu este respectată, va avea loc pierderea unor date datorită depăşirii capacităţii bufferelor. Chiar dacă datele sunt furnizate de magistrala USB cu rata adecvată, întârzierile introduse de programe pot afecta negativ aplicaţiile care utilizează aceste date, cum sunt cele pentru video-conferinţe.

Şirurilor de date izocrone li se alocă o porţiune dedicată a lăţimii de bandă a magistralei USB. De asemenea, această magistrală este proiectată pentru o întârziere minimă a transferurilor de date izocrone.

2.4. Modelul comunicaţiei USB

Un sistem USB permite comunicaţia dintre un program de aplicaţie (client) rulat pe un calculator gazdă şi unul sau mai multe dispozitive USB conectate la acest calculator. Un dispozitiv fizic USB conţine o interfaţă cu magistrala USB, un dispozitiv logic USB şi o funcţie. O funcţie USB poate avea cerinţe diferite ale fluxului de comunicaţie pentru diferite interacţiuni între acea funcţie şi calculatorul gazdă. Prin separarea diferitelor fluxuri de comunicaţie cu o funcţie USB, sistemul USB asigură o utilizare mai eficientă a magistralei USB. Fluxurile de comunicaţie utilizează magistrala USB pentru realizarea comunicaţiei între programul de aplicaţie şi funcţia USB. Fiecare flux de comunicaţie se termină la un *punct terminal* ("*endpoint*") dintr-un dispozitiv USB.

Un punct terminal este o parte a unui dispozitiv USB, reprezentând punctul de sfârşit al fluxului de comunicaţie dintre calculator şi dispozitiv. Fiecare dispozitiv logic USB este format din mai multe puncte terminale independente. Fiecare dispozitiv logic are o adresă unică, care este asignată de sistem în momentul conectării dispozitivului la magistrala USB. Fiecare punct terminal dintr-un dispozitiv USB este identificat printr-un număr, care este atribuit la proiectarea dispozitivului. De asemenea, fiecare punct terminal are o anumită direcţie a fluxului de date: intrare (IN), pentru transferul datelor de la dispozitivul USB la calculatorul gazdă, sau ieşire (OUT), pentru transferul datelor de la calculatorul gazdă la dispozitivul USB. Combinaţia dintre adresa dispozitivului, numărul punctului terminal şi direcţia acestuia permite identificarea unică a fiecărui punct terminal.

Punctele terminale conţin buffere de intrare sau de ieşire prin intermediul cărora se realizează comunicaţia între calculatorul gazdă şi dispozitivul USB. De exemplu, dacă programul de aplicaţie rulat pe calculatorul gazdă transmite un pachet adresat unui anumit dispozitiv USB, prin intermediul driverului dispozitivului respectiv, acest pachet va fi depus în bufferul unui punct terminal de ieşire al dispozitivului. Ulterior, controlerul dispozitivului va citi din buffer pachetul recepţionat. Dacă dispozitivul trebuie să transmită date la calculatorul gazdă, nu poate depune datele direct pe magistrală, deoarece toate tranzacţiile sunt iniţiate de calculatorul gazdă. Dispozitivul va depune datele în bufferul unui punct terminal de intrare, aceste date fiind transferate la calculatorul gazdă atunci când acesta va transmite un pachet de intrare prin care va solicita transferul unor date, dacă acestea sunt disponibile.

Fiecare dispozitiv trebuie să conţină cel puţin punctele terminale cu numărul 0, atât cel de intrare, cât şi cel de ieşire. Aceste puncte terminale se utilizează pentru transferul informaţiilor de control şi de stare în timpul fazei de enumerare şi în timpul în care dispozitivul este operaţional şi conectat la magistrala USB. Dispozitivele cu viteză redusă (1,5 Mbiţi/s) pot conţine doar două puncte terminale suplimentare, pe lângă cele două cu numărul 0. Dispozitivele cu viteză normală (12 Mbiţi/s) sau ridicată pot conţine un număr maxim de 15 puncte terminale de intrare şi 15 puncte terminale de ieşire, pe lângă cele două puncte terminale cu numărul 0.

O colecție a unor puncte terminale dintr-un dispozitiv USB implementează o *interfață*. O asemenea interfață indică clasa dispozitivului USB, iar această clasă va determina driverul de dispozitiv generic care va fi utilizat de sistemul de operare pentru comunicația cu punctele terminale care implementează interfața respectivă.

Comunicația dintre programul de aplicație de pe calculatorul gazdă şi un punct terminal dintr-un dispozitiv USB se realizează printr-o conexiune logică numită *conductă* ("*pipe*"). O asemenea conductă reprezintă o legătură între un buffer din memoria calculatorului gazdă şi un punct terminal din dispozitivul USB. Fiecărei conducte i se asociază anumite informații, cum sunt: rata de transfer necesară, tipul serviciului de transfer şi caracteristicile punctului terminal asociat, ca direcția şi dimensiunea bufferului.

Conductele pot fi de tip şir sau de tip mesaj. În cazul conductelor de tip şir, datele nu au o structură definită de specificațiile USB. Datele sunt transferate secvențial şi au o direcție predefinită, de intrare sau de ieşire. Conductele de tip şir permit transferuri de întrerupere, de date voluminoase sau izocrone. Aceste conducte sunt controlate fie de calculatorul gazdă, fie de dispozitivul USB. În cazul conductelor de tip mesaj, datele au o structură definită de specificațiile USB. Cu toate acestea, conținutul datelor transferate într-o conductă de tip mesaj nu este interpretat de controlerul magistralei USB. Aceste conducte sunt controlate de calculatorul gazdă şi permit doar transferuri de control, în ambele direcții.

Comunicația între punctele terminale de intrare şi de ieşire cu numărul 0 şi calculatorul gazdă se realizează printr-o conductă specială, numită *conductă implicită de control*. Această conductă de tip mesaj este disponibilă imediat după conectarea şi resetarea dispozitivului USB, asigurând o legătură bidirecțională pentru interogarea dispozitivului de către programele de sistem ale calculatorului gazdă şi transmiterea informațiilor de configurație către calculator. După configurarea dispozitivului USB, vor fi disponibile alte conducte necesare pentru transferurile de date, conducta implicită de control fiind utilizată în continuare de programele de sistem ale calculatorului gazdă pentru transferurile de control.

2.5. Protocolul USB

Similar cu alte interfețe mai recente, interfața USB utilizează un protocol bazat pe pachete. Toate transferurile sunt inițiate de controlerul USB al calculatorului gazdă. Tranzacțiile de pe magistrală implică transmisia a patru tipuri de pachete:

- Pachet antet (simbol – "*token*");
- Pachet de date;
- Pachet de confirmare ("*handshake*");
- Pachet special.

Fiecare tranzacție începe în momentul în care controlerul USB transmite, pe baza unei planificări, un pachet antet care descrie tipul tranzacției, direcția acesteia, adresa dispozitivului USB şi numărul punctului terminal. Sursa tranzacției transmite apoi un pachet de date conținând datele care trebuie transferate, sau poate indica faptul că nu are date de transmis prin faptul că pachetul de date nu conține informații utile. Destinația răspunde, în general, cu un pachet de confirmare indicând dacă transferul s-a efectuat cu succes sau dacă punctul terminal nu este disponibil.

2.5.1. Câmpurile pachetelor USB

Principalele câmpuri ale pachetelor USB sunt descrise în continuare.

Câmpul de sincronizare

Toate pachetele încep cu un câmp de sincronizare (SYNC), care este utilizat de circuitele receptorului pentru sincronizarea cu ceasul transmițătorului. Câmpul de sincronizare conține un număr de şase tranziții succesive de la valoarea 1 la valoarea 0 sau invers, urmate de un marcaj de doi biți care indică sfârşitul câmpului de sincronizare.

Câmpul de identificare al pachetului

Câmpul de identificare al pachetului (PID) urmează imediat după câmpul de sincronizare. Câmpul PID conţine patru biţi care indică tipul pachetului şi patru biţi de control care confirmă acurateţea biţilor care conţin tipul pachetului. Biţii de control conţin complementul faţă de 1 al biţilor care reprezintă tipul pachetului. Formatul câmpului PID este ilustrat în continuare, unde $nPID_i$ reprezintă complementul faţă de 1 al bitului PID_i.

7	6	5	4	3	2	1	0
$nPID_3$	$nPID_2$	$nPID_1$	$nPID_0$	PID_3	PID_2	PID_1	PID_0

Calculatorul gazdă şi toate funcţiile USB decodifică complet toţi biţii câmpului PID. Dacă un câmp PID este recepţionat cu valori incorecte ale biţilor de control sau cu valori nedefinite ale tipului pachetului, se presupune că a fost recepţionat eronat şi restul pachetului este ignorat de receptor.

Tabelul 11.2 indică diferitele tipuri de pachete, codificarea şi descrierea acestora. Pentru simplitate, pachetele speciale nu sunt detaliate.

Tabelul 11.2. Codificarea şi descrierea diferitelor tipuri de pachete USB.

Tip pachet	Subtip pachet	PID [3..0]	Descriere
Antet (Token)	OUT	0001	Adresa şi numărul punctului terminal pentru o tranzacţie de ieşire
	IN	1001	Adresa şi numărul punctului terminal pentru o tranzacţie de intrare
	SOF	0101	Indicator de început al cadrului şi numărul cadrului
	SETUP	1101	Adresa şi numărul punctului terminal pentru o tranzacţie de control în faza de setare
Date	DATA0	0011	Identificator pentru pachetele de date cu număr par
	DATA1	1011	Identificator pentru pachetele de date cu număr impar
	DATA2	0111	Identificator pentru pachetele de date la tranzacţiile izocrone de intrare de viteză ridicată şi cu lăţimea de bandă ridicată
	MDATA	1111	Identificator pentru pachetele de date la tranzacţiile izocrone de ieşire de viteză ridicată şi cu lăţimea de bandă ridicată
Confirmare	ACK	0010	Confirmarea recepţionării fără erori a pachetului de date
	NAK	1010	Dispozitivul receptor nu poate primi date sau dispozitivul transmiţător nu poate transmite date
	STALL	1110	Punctul terminal este oprit
	NYET	0110	Nu s-a recepţionat încă un răspuns de la receptor
Special		XY00	Identificator al unui pachet special; XY poate fi 01, 10 sau 11

Câmpul de adresă

Câmpul de adresă (ADDR) specifică adresa funcţiei USB care este sursa sau destinaţia unui pachet de date. Acest câmp are o lungime de 7 biţi, permiţând specificarea a până la 128 de adrese. Fiecare adresă defineşte o singură funcţie. Adresa 0 este rezervată ca adresă implicită şi nu poate fi asignată în mod explicit unei funcţii. La pornirea şi resetarea unei funcţii, adresa acesteia va avea valoarea implicită 0. Calculatorul trebuie să seteze adresa funcţiei în timpul procesului de enumerare.

Câmpul punctului terminal

Câmpul punctului terminal (ENDP) permite o adresare mai flexibilă a funcţiilor cu mai multe puncte terminale. Acest câmp are o lungime de 4 biţi, ceea ce permite adresarea a până la 16 puncte

terminale. Dispozitivele cu viteză redusă pot avea însă doar două puncte terminale suplimentare pe lângă punctul terminal cu numărul 0.

Câmpul de date

Câmpul de date poate conține între zero şi 1024 de octeți, în funcție de tipul transferului. Biții de date din cadrul fiecărui octet sunt transmişi pe magistrală începând cu bitul cel mai puțin semnificativ.

Câmpurile de control ciclic redundant

Aceste câmpuri conțin codurile de control ciclic redundant (CRC) utilizate pentru verificarea integrității diferitelor câmpuri din pachetele de antet şi de date. Câmpul PID nu este inclus în calculul codului CRC. Codurile CRC pentru pachetele de antet şi de date asigură detectarea tuturor erorilor de un bit şi de doi biți. În cazul în care codul CRC calculat la recepție diferă de codul transmis într-un câmp CRC, receptorul va ignora câmpurile protejate şi, în majoritatea cazurilor, întregul pachet. Standardul USB specifică polinoamele generatoare utilizate pentru calculul codurilor CRC. Pentru pachetele de antet se utilizează un câmp CRC de 5 biți (CRC5), iar pentru pachetele de date se utilizează un câmp CRC de 16 biți (CRC16).

Câmpul de sfârşit al pachetului

Acest câmp (EOP) indică sfârşitul unui pachet prin valoarea 0 pe durata corespunzătoare a doi biți, urmată de valoarea 1 pe durata corespunzătoare unui bit.

2.5.2. Formatul pachetelor USB

În continuare se prezintă formatul pachetelor de antet, SOF, de date şi de confirmare.

Pachete de antet

Aceste pachete sunt transmise doar de calculatorul gazdă. Structura unui pachet de antet este ilustrată în Figura 11.7.

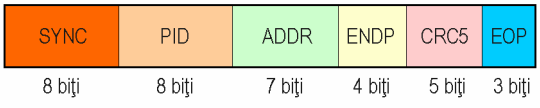


Figura 11.7. Structura unui pachet USB de antet.

Câmpul de identificare al pachetului, PID, poate specifica un pachet de antet cu subtipul IN, OUT sau SETUP. Pachetele cu subtipul IN sau OUT informează dispozitivul USB asupra direcției transferului care urmează: intrare (citire de către calculatorul gazdă), respectiv ieşire (scriere de către calculatorul gazdă). Un pachet cu subtipul SETUP se utilizează la începutul transferurilor de control. În cazul pachetelor cu subtipul OUT sau SETUP, câmpurile ADDR şi ENDP identifică în mod unic punctul terminal care va recepționa următorul pachet de date. În cazul unui pachet cu subtipul IN, câmpurile ADDR şi ENDP identifică punctul terminal care va transmite un pachet de date. Câmpul CRC5 conține codul CRC pentru câmpurile ADDR şi ENDP.

Pachete SOF

Pentru sincronizarea întregului sistem USB, calculatorul gazdă transmite câte un pachet SOF (*Start-of-Frame*) la fiecare interval de timp corespunzător începutului unui cadru sau micro-cadru. Un cadru reprezintă un interval de timp de 1 ms ± 0,0005 ms şi este definit pentru magistrala USB cu viteza normală (12 Mbiți/s). Un micro-cadru reprezintă un interval de timp de 125 μs ± 0,0625 μs şi este definit pentru magistrala USB cu viteza ridicată (480 Mbiți/s). Un pachet SOF constă dintr-un câmp de sincronizare, un câmp PID şi un câmp de 11 biți reprezentând numărul cadrului, după cum se ilustrează în Figura 11.8. În cazul magistralei USB cu viteza ridicată, numărul cadrului va fi acelaşi pentru opt pachete SOF consecutive, pe durata unei perioade de 1 ms.

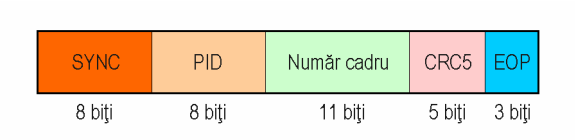


Figura 11.8. Structura unui pachet USB SOF.

Toate funcțiile şi distribuitoarele USB recepționează pachetele SOF. Recepția acestor pachete nu va determina generarea unui pachet de confirmare din partea receptorului.

Pachete de date

Informațiile propriu-zise sunt transmise pe magistrala USB în pachetele de date. Un pachet de date constă din câmpul de sincronizare SYNC, un câmp de identificare al pachetului PID, un câmp de date, un câmp CRC de 16 biți şi câmpul de sfârşit al pachetului EOP (Figura 11.9). Codul CRC se calculează numai pentru câmpul de date. Datele sunt transmise într-un număr întreg de octeți. Pentru dispozitivele cu viteză redusă, lungimea maximă a câmpului de date este de 8 octeți. Pentru dispozitivele cu viteză normală (12 Mbiți/s), lungimea maximă a câmpului de date este de 1023 octeți, iar pentru cele de viteză ridicată (480 Mbiți/s), lungimea maximă este de 1024 octeți.

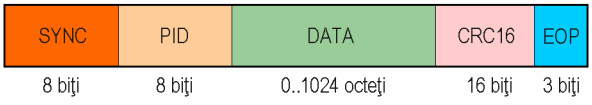


Figura 11.9. Structura unui pachet USB de date.

Pachete de confirmare

Pachetele de confirmare constau doar din câmpul de sincronizare SYNC, un câmp de identificare al pachetului PID şi câmpul de sfârşit al pachetului EOP (Figura 11.10). Aceste pachete se utilizează pentru a raporta starea unei tranzacții de date prin subtipul returnat în câmpul PID. Subtipul unui pachet de confirmare poate fi ACK (*Acknowledge*), NAK (*Negative Acknowledge*), STALL (*Stall*) sau NYET (*No Response Yet*). Aceste subtipuri sunt descrise în Tabelul 11.2.

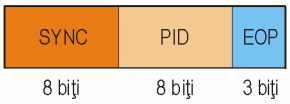


Figura 11.10. Structura unui pachet USB de confirmare.

2.6. Descriptori USB

Dispozitivele USB au o ierarhie de descriptori prin care raportează atributele lor calculatorului gazdă. Un descriptor reprezintă o structură de date cu un format bine definit de standardul USB. Fiecare descriptor începe cu un octet care conține numărul total de octeți ai descriptorului, urmat de un octet care indică tipul descriptorului. Pe lângă descriptorii standard, care sunt specificați de standardul USB, dispozitivele USB pot returna şi descriptori specifici unei clase de dispozitiv sau unui producător.

Există următoarele tipuri principale de descriptori standard:

- Descriptori de dispozitiv;
- Descriptori de configurație;
- Descriptori de interfață;
- Descriptori ai punctelor terminale;
- Descriptori de tip şir de caractere.

Ierarhia de descriptori are ca rădăcină la nivelul superior descriptorul de dispozitiv. La nivelul următor se află descriptorii de configurație, existând câte un asemenea descriptor pentru fiecare

configurație a dispozitivului. Pentru fiecare configurație pot exista unul sau mai mulți descriptori de interfață, în funcție de numărul de interfețe ale configurației respective. În sfârşit, pentru fiecare punct terminal al fiecărei interfețe există câte un descriptor al punctului terminal (Figura 11.11).

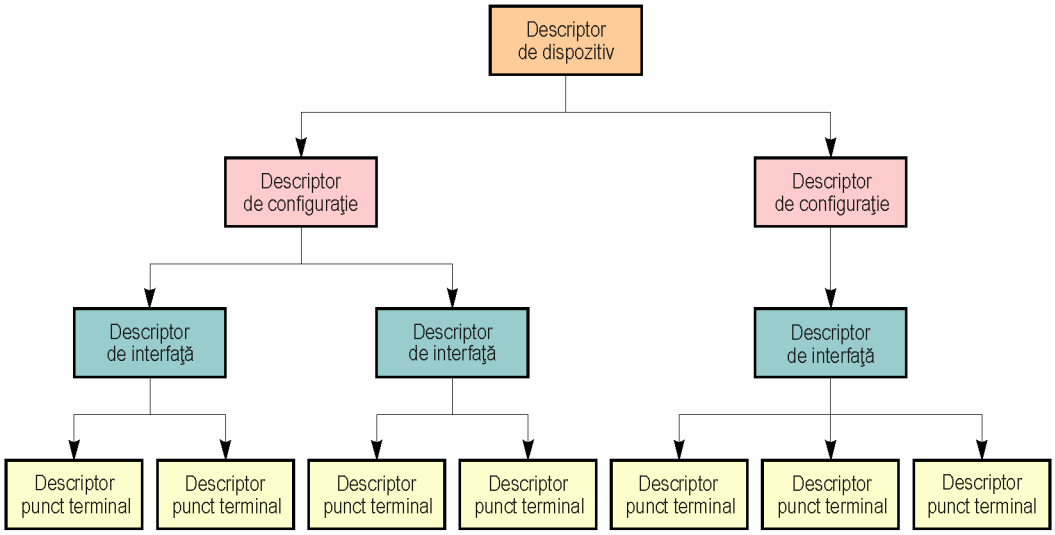


Figura 11.11. Ierarhia de descriptori ai unui dispozitiv USB.

Descriptorul de dispozitiv conține informații generale despre dispozitivul USB, aceste informații fiind valabile pentru toate configurațiile dispozitivului. Un dispozitiv USB poate avea un singur descriptor de dispozitiv. Printre informațiile pe care le conține un descriptor de dispozitiv, se amintesc următoarele: revizia standardului USB cu care este conform dispozitivul, clasa din care face parte dispozitivul, subclasa dispozitivului, identificatorul producătorului (asignat de organizația USB-IF), identificatorul produsului (asignat de producător), numărul de configurații posibile ale dispozitivului.

Un dispozitiv USB poate avea mai multe configurații, caracterizate prin atribute diferite. Un descriptor de configurație caracterizează o anumită configurație a dispozitivului. Un descriptor de configurație conține informații ca numărul de interfețe ale configurației, dacă dispozitivul este alimentat de la magistrala USB sau dacă este alimentat de la o sursă proprie şi curentul maxim consumat de dispozitiv în cazul în care acesta este alimentat de la magistrala USB.

În faza de enumerare, calculatorul gazdă citește descriptorii de configurație, pe baza cărora validează o anumită configurație a dispozitivului. La un moment dat, poate fi validată o singură configurație. De exemplu, un dispozitiv poate avea două configurații, prima corespunzătoare alimentării de la magistrala USB şi a doua corespunzătoare alimentării de la o sursă separată. Dacă dispozitivul este conectat la un calculator alimentat de la rețea, driverul dispozitivului poate selecta prima configurație, permițând alimentarea dispozitivului fără conectarea acestuia la o sursă separată. Dacă dispozitivul este conectat la un calculator portabil, driverul poate valida a doua configurație, care necesită conectarea dispozitivului la o sursă proprie.

Un descriptor de interfață caracterizează un set de puncte terminale din cadrul unei configurații, care sunt grupate într-o interfață. De exemplu, în cazul unui dispozitiv multifuncțional conținând un fax, un scaner şi o imprimantă, unul din descriptorii de interfață poate caracteriza funcția de fax a dispozitivului, un al doilea descriptor de interfață poate caracteriza funcția de scaner, iar un al treilea descriptor de interfață poate caracteriza funcția de imprimantă. Pentru un dispozitiv, pot fi validate simultan mai multe configurații de interfață. Un descriptor de interfață nu este accesibil direct, ci este returnat ca parte a unui descriptor de configurație.

O interfață poate conține setări alternative, care permit modificarea punctelor terminale asociate interfeței sau a caracteristicilor acestora după configurarea dispozitivului. În mod implicit, pentru o interfață este selectată setarea alternativă cu numărul zero. Setările alternative permit modificarea unei interfețe a dispozitivului în timp ce alte interfețe rămân în funcțiune, ceea ce este mai avantajos decât utilizarea unor configurații diferite. Dacă o configurație conține setări alternative

pentru una sau mai multe din interfețele sale, în descriptorul de configurație trebuie să se includă un descriptor de interfață separat pentru fiecare setare, fiecare asemenea descriptor fiind urmat de descriptorii punctelor terminale corespunzătoare.

Un descriptor de interfață conține informații ca numărul interfeței respective, numărul setării alternative a interfeței, numărul punctelor terminale utilizate de interfață, codul clasei şi al subclasei (asignate de organizația USB-IF). Numărul punctelor terminale utilizate de interfață nu include punctul terminal zero.

Fiecare punct terminal utilizat de o interfață are câte un descriptor propriu. În principiu, acest descriptor conține informațiile necesare pentru determinarea cerințelor lățimii de bandă a punctului terminal. Descriptorul unui punct terminal conține informații ca numărul (adresa) punctului terminal, direcția acestuia, tipul transferului utilizat pentru comunicarea cu punctul terminal, dimensiunea maximă a pachetelor pe care le poate transmite sau recepționa punctul terminal şi intervalul de timp după care calculatorul gazdă va interoga punctul terminal în vederea transferurilor de date. Descriptorii punctelor terminale nu sunt accesibili direct, fiind returnați în cadrul unui descriptor de configurație. Punctul terminal zero nu are un descriptor propriu.

Descriptorii de tip şir de caractere sunt opționali. Aceşti descriptori furnizează informații despre dispozitivul USB într-o formă care poate fi afişată direct. Referirea la descriptorii de tip şir de caractere se realizează prin valori de index din cadrul descriptorilor de dispozitiv, de configurație şi de interfață. În cazul în care pentru un dispozitiv USB nu există descriptori de tip şir de caractere, toate referințele la asemenea descriptori trebuie setate la zero în cadrul descriptorilor de dispozitiv, de configurație şi de interfață.

Descriptorii de tip şir de caractere au o structură standard. Primul octet din fiecare descriptor indică lungimea în octeți a descriptorului, iar al doilea octet indică tipul descriptorului. În cadrul unui descriptor, şirul de caractere începe de la deplasamentul 2. Fiecare şir de caractere este codificat în formatul UNICODE, în modul definit de Consorțiul Unicode (http://www.unicode.com). Şirurile de caractere nu sunt terminate cu un octet NULL. Lungimea unui şir se poate determina ca fiind L–2, unde L este lungimea în octeți a descriptorului. Descriptorii pot conține şiruri de caractere în diferite limbi. Atunci când se solicită un descriptor de tip şir de caractere, trebuie să se solicite limba dorită printr-un identificator al limbii (LANGID) de 16 biți.

2.7. Procesul de enumerare

Atunci când un dispozitiv USB este conectat la magistrala USB sau este deconectat de la magistrală, calculatorul gazdă execută un proces numit *enumerare* pentru a determina modificările apărute în configurația sistemului USB. La conectarea unui dispozitiv USB la magistrală, se execută următoarele operații:

1. Distribuitorul la care s-a conectat dispozitivul detectează conectarea acestuia prin intermediul rezistenței utilizate ca terminator al magistralei de la capătul dispozitivului. Distribuitorul informează calculatorul gazdă asupra modificării apărute. În acest moment, dispozitivul USB este alimentat de la magistrală şi portul la care este conectat este dezactivat.
2. Calculatorul determină tipul modificării apărute şi portul la care a apărut modificarea prin interogarea distribuitorului.
3. Calculatorul aşteaptă un timp de minim 100 ms pentru ca tensiunea de alimentare a dispozitivului să devină stabilă, după care transmite o comandă de validare a portului şi o comandă de resetare a acestuia. În cazul unui dispozitiv cu viteză ridicată (480 Mbiți/s), acesta inițiază un protocol electric special pentru stabilirea unei legături cu această viteză. Dacă acest protocol electric nu este inițiat sau nu se termină cu succes, comunicația se va realiza cu viteza normală (12 Mbiți/s).
4. După terminarea procedurii de resetare, portul este validat, iar dispozitivul se află în starea implicită şi poate absorbi un curent de maxim 100 mA de pe linia V_{BUS} a magistralei. Dispozitivul va răspunde la tranzacțiile cu adresa implicită zero.
5. Calculatorul solicită descriptorul de dispozitiv, iar dispozitivul transmite acest descriptor prin intermediul conductei implicite.

6. Calculatorul asignează o adresă unică dispozitivului.
7. Calculatorul solicită dispozitivului descriptorii de configurație, iar dispozitivul transmite calculatorului aceşti descriptori.
8. Pe baza informațiilor de configurație, calculatorul asignează o anumită configurație dispozitivului. În acest moment, punctele terminale ale dispozitivului sunt configurate conform caracteristicilor specificate în descriptorii acestora. Dispozitivul este pregătit pentru funcționare şi poate absorbi de pe linia V_{BUS} a magistralei curentul specificat pentru configurația selectată.

Observație

- Detaliile operațiilor executate în timpul procesului de enumerare pot varia în funcție de sistemul de operare.

Atunci când dispozitivul USB este deconectat de la un port USB, distribuitorul informează calculatorul gazdă asupra modificării apărute. Calculatorul dezactivează portul respectiv şi îşi actualizează informațiile despre topologia magistralei USB.

2.8. Comenzi USB

Fiecare dispozitiv USB trebuie să răspundă la comenzile USB transmise de calculatorul gazdă. Comenzile sunt transmise de calculator pe conducta implicită de control, iar răspunsul este transmis de dispozitiv pe aceeaşi conductă. Specificațiile USB definesc o serie de comenzi standard care trebuie implementate de fiecare dispozitiv USB. În plus, pot exista comenzi specifice pentru diferite clase de dispozitive. De asemenea, producătorii dispozitivelor pot defini comenzi proprii.

Transmiterea şi execuția unei comenzi USB poate necesita două sau trei faze de transfer. În prima fază, calculatorul transmite comanda şi parametrii acesteia într-un pachet SETUP, utilizând un transfer de control. În faza a doua, care este opțională, se transferă date de la calculator la dispozitivul USB sau invers. În faza a treia, se transferă informații de stare de la dispozitivul USB la calculator sau invers. Direcția transferului din faza de stare este inversă direcției transferului din faza de date. Dacă faza de date lipseşte, în faza de stare direcția transferului este de la dispozitivul USB la calculator.

Fiecare pachet SETUP conține opt octeți. Structura unui pachet SETUP este ilustrată în Tabelul 11.3. Dimensiunea câmpurilor este indicată în octeți. Biții din cadrul octetului care descrie caracteristicile comenzii sunt indicați prin b7..b0.

Tabelul 11.3. Structura unui pachet SETUP.

Offset	Câmp	Dimensiune	Descriere
0	bmRequestType	1	Caracteristici ale comenzii: b7: Direcția transferului de date 0 = De la calculator la dispozitiv 1 = De la dispozitiv la calculator b6..b5: Tipul comenzii 0 = Standard 1 = Specific clasei 2 = Definit de producător 3 = Rezervat b4..b0: Destinația comenzii 0 = Dispozitiv 1 = Interfață 2 = Punct terminal 3 = Altă destinație 4..31 = Rezervat
1	bRequest	1	Codul comenzii
2	wValue	2	Parametru; variază în funcție de comandă
4	wIndex	2	Parametru; în mod tipic, numărul interfeței sau al punctului terminal
6	wLength	2	Numărul octeților de transferat în faza de date

Comenzile USB pot fi destinate dispozitivelor, interfețelor sau punctelor terminale. În funcție de destinație, aceeaşi comandă poate avea efecte diferite.

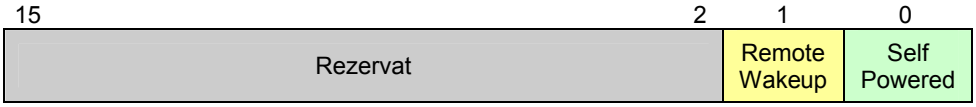
2.8.1. Comenzi standard destinate dispozitivelor

Comenzile standard care sunt destinate dispozitivelor sunt prezentate sintetic în Tabelul 11.4. Coloana Data indică datele transferate în faza de date.

Tabelul 11.4. Comenzi USB standard destinate dispozitivelor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1000 0000	GET_STATUS (0x00)	0	0	2	Stare dispozitiv
0000 0000	CLEAR_FEATURE (0x01)	Selector caracteristică	0	0	–
0000 0000	SET_FEATURE (0x03)	Selector caracteristică	0	0	–
0000 0000	SET_ADDRESS (0x05)	Adresă dispozitiv	0	0	–
1000 0000	GET_DESCRIPTOR (0x06)	Tip şi index descriptor	0 sau LANGID	Lungime descriptor	Descriptor
0000 0000	SET_DESCRIPTOR (0x07)	Tip şi index descriptor	0 sau LANGID	Lungime descriptor	Descriptor
1000 0000	GET_CONFIGURATION (0x08)	0	0	1	Număr configurație
0000 0000	SET_CONFIGURATION (0x09)	Număr configurație	0	0	–

Comanda *Get_Status* destinată unui dispozitiv USB returnează un cuvânt de stare de la dispozitiv la calculator în timpul fazei de date. Structura datelor returnate este ilustrată mai jos.



Bitul Self Powered indică prin valoarea 1 faptul că dispozitivul este alimentat de la o sursă proprie. Dacă acest bit este 0, dispozitivul este alimentat de la magistrala USB. Bitul Self Powered nu poate fi modificat prin comenzile *Set_Feature* sau *Clear_Feature*. Bitul Remote Wakeup indică prin valoarea 1 faptul că dispozitivul are caracteristica *Remote Wakeup* validată. Atunci când această caracteristică este validată, dispozitivul poate determina trecerea calculatorului gazdă din starea inactivă (*Suspend*) în starea activă. Atunci când bitul Remote Wakeup este 0, caracteristica *Remote Wakeup* a dispozitivului este invalidată. Bitul Remote Wakeup poate fi modificat prin comenzile *Set_Feature* şi *Clear_Feature* utilizând selectorul DEVICE_REMOTE_WAKEUP (cu valoarea 0x01).

Comenzile *Clear_Feature* şi *Set_Feature* destinate unui dispozitiv USB permit invalidarea, respectiv validarea unor caracteristici ale dispozitivului. Câmpul wValue al comenzilor trebuie să conțină selectorul caracteristicii respective. Caracteristicile care pot fi invalidate sau validate sunt *Remote Wakeup*, cu selectorul DEVICE_REMOTE_WAKEUP (0x01), şi *Test Mode*, cu selectorul TEST_MODE (0x02). Caracteristica *Test Mode* este implementată doar de dispozitivele USB cu viteză ridicată şi de distribuitoare, permițând trecerea într-un anumit mod de test pentru efectuarea diferitelor teste de conformitate cu standardul USB la nivelul interfeței electrice. Caracteristica *Test Mode* nu poate fi invalidată prin comanda *Clear_Feature*.

Comanda *Set_Address* este utilizată în timpul procesului de enumerare pentru asignarea unei adrese unice dispozitivului USB. Adresa trebuie specificată în câmpul wValue şi poate avea valoarea maximă 127. Adresa dispozitivului va fi setată doar după terminarea cu succes a fazei de stare a comenzii *Set_Address*.

Comanda *Get_Descriptor* returnează un descriptor specificat, dacă acesta există. Câmpul wValue trebuie să specifice tipul descriptorului în octetul superior şi indexul descriptorului în octetul

inferior. Tipul descriptorului poate specifica un descriptor de dispozitiv (0x01), un descriptor de configurație (0x02) sau un descriptor de tip şir de caractere (0x03). Indexul descriptorului selectează un anumit descriptor în cazul în care dispozitivul implementează mai mulți descriptori de acelaşi tip. Acest index se utilizează numai pentru un descriptor de configurație sau de tip şir de caractere. Pentru un descriptor de dispozitiv, indexul trebuie setat la 0. În cazul descriptorilor de tip şir de caractere, câmpul wIndex trebuie să specifice identificatorul limbii (LANGID), iar în cazul altor descriptori acest câmp trebuie setat la 0. Câmpul wLength trebuie să conțină numărul de octeți care vor fi returnați. Dacă descriptorul are o lungime mai mare decât valoarea din câmpul wLength, dispozitivul returnează doar numărul specificat de octeți.

Dacă în câmpul wValue al comenzii *Get_Descriptor* se specifică un descriptor de configurație, dispozitivul va returna descriptorul de configurație, toți descriptorii de interfață pentru acea configurație şi toți descriptorii punctelor terminale pentru toate interfețele. După descriptorul de configurație, se va returna primul descriptor de interfață. După acest descriptor, vor urma descriptorii punctelor terminale pentru prima interfață. Dacă există alte interfețe, se va returna descriptorul lor de interfață şi descriptorii punctelor terminale ai acestora.

Comanda *Set_Descriptor* este opțională şi se poate utiliza pentru actualizarea unor descriptori existenți sau adăugarea unor noi descriptori. Semnificația câmpurilor comenzii este aceeaşi ca şi la comanda *Get_Descriptor*.

Comanda *Get_Configuration* returnează numărul configurației curente a dispozitivului. Valoarea este returnată într-un octet în timpul fazei de date. Dacă valoarea returnată este 0, dispozitivul nu este configurat.

Comanda *Set_Configuration* permite validarea unei anumite configurații a dispozitivului. Numărul configurației dorite trebuie specificat în octetul inferior al câmpului wValue. Acest număr trebuie să corespundă cu numărul unei configurații dintr-un descriptor de configurație. Octetul superior al câmpului wValue este rezervat.

2.8.2. Comenzi standard destinate interfețelor

În Tabelul 11.5 se prezintă comenzile standard care sunt destinate interfețelor.

Tabelul 11.5. Comenzi USB standard destinate interfețelor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1000 0001	GET_STATUS (0x00)	0	Interfață	2	Stare interfață
0000 0001	CLEAR_FEATURE (0x01)	Selector caracteristică	Interfață	0	–
0000 0001	SET_FEATURE (0x03)	Selector caracteristică	Interfață	0	–
1000 0001	GET_INTERFACE (0x0A)	0	Interfață	1	Setare alternativă
0000 0001	SET_INTERFACE (0x0B)	Setare alternativă	Interfață	0	–

Pentru toate comenzile USB destinate interfețelor, câmpul wIndex trebuie să conțină în octetul inferior numărul interfeței la care se referă comanda.

Comanda *Get_Status* destinată unei interfețe returnează doi octeți cu valoarea 0 în timpul fazei de date. Aceşti octeți sunt rezervați pentru versiunile viitoare ale standardului USB.

Comenzile *Clear_Feature* şi *Set_Feature* destinate unei interfețe permit invalidarea, respectiv validarea unor caracteristici ale interfeței. Versiunea curentă a standardului USB nu permite invalidarea sau validarea nici uneia din caracteristicile interfețelor.

Comanda *Get_Interface* returnează setarea alternativă selectată pentru interfața specificată. Setarea alternativă este returnată într-un octet în timpul fazei de date. Principiul setărilor alternative a fost descris în secțiunea 2.6.

Comanda *Set_Interface* permite selectarea unei setări alternative pentru interfața specificată. Câmpul wValue trebuie să conțină setarea alternativă care se doreşte selectată.

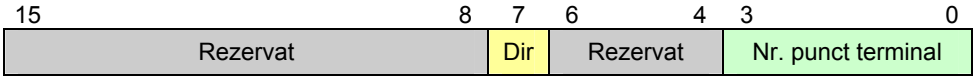
2.8.3. Comenzi standard destinate punctelor terminale

Comenzile standard care sunt destinate punctelor terminale sunt prezentate în Tabelul 11.6.

Tabelul 11.6. Comenzi USB standard destinate punctelor terminale.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1000 0010	GET_STATUS (0x00)	0	Punct terminal	2	Stare punct terminal
0000 0010	CLEAR_FEATURE (0x01)	Selector caracteristică	Punct terminal	0	–
0000 0010	SET_FEATURE (0x03)	Selector caracteristică	Punct terminal	0	–
1000 0010	SYNCH_FRAME (0x0C)	0	Punct terminal	2	Număr cadru

În cazul comenzilor USB destinate punctelor terminale, câmpul wIndex trebuie să conțină numărul punctului terminal la care se referă comanda şi direcția acestuia. Formatul câmpului wIndex este ilustrat mai jos.



Bitul Dir indică prin valoarea 0 un punct terminal de ieşire şi prin valoarea 1 un punct terminal de intrare. În cazul unui punct terminal de control, bitul Dir trebuie setat la 0. Biții 3..0 specifică numărul punctului terminal.

Comanda *Get_Status* destinată unui punct terminal returnează doi octeți de stare în timpul fazei de date. Bitul 0 al octetului inferior returnat indică prin valoarea 1 faptul că punctul terminal este oprit (are caracteristica *Halt* setată la 1).

Comenzile *Clear_Feature* şi *Set_Feature* destinate unui punct terminal permit invalidarea, respectiv validarea unor caracteristici ale punctului terminal respectiv. Câmpul wValue al comenzilor trebuie să conțină selectorul caracteristicii respective. Standardul USB curent permite invalidarea (resetarea la 0) şi validarea (setarea la 1) unei singure caracteristici care se referă la un punct terminal, şi anume, caracteristica *Halt*, pentru care trebuie să se utilizeze selectorul ENDPOINT_HALT (0x00). Această caracteristică va fi resetată la 0 în mod automat după o comandă *Set_Configuration* sau *Set_Interface*.

Comanda *Synch_Frame* determină returnarea de către punctul terminal căruia îi este destinată a numărului unui cadru de sincronizare. Atunci când un punct terminal permite transferuri izocrone, poate fi necesar ca dimensiunea transferurilor să varieze conform unui model repetitiv. Prin această comandă, punctul terminal returnează calculatorului gazdă numărul cadrului de la care începe modelul repetitiv.

2.9. Clasa de dispozitive USB HID

2.9.1. Prezentare generală a clasei de dispozitive HID

Clasa de dispozitive HID (*Human Interface Device*) constă în principal din dispozitive utilizate de operatori pentru controlul funcționării unui sistem de calcul. Exemple tipice de dispozitive din această clasă sunt următoarele:

- Tastaturi şi dispozitive indicatoare: mouse, trackball;
- Comenzi de pe panouri frontale: comutatoare, butoane, glisoare;
- Comenzi aflate pe dispozitive cum sunt telefoane, jocuri sau simulatoare: volane, pedale;
- Dispozitive de afişare: diode electroluminiscente, afişaje alfanumerice;
- Instrumente medicale: aparate cu ultrasunete;
- Dispozitive care nu necesită intervenție umană, dar pot transmite date într-un format similar cu dispozitivele din clasa HID: cititoare pentru coduri de bare, termometre, aparate de măsură.

Clasa HID constă deci dintr-o categorie largă de dispozitive, cu caracteristici variate. În această clasă se pot încadra şi o serie de dispozitive care nu sunt prevăzute cu o interfaţă umană. Prin utilizarea modelului clasei HID, este posibilă comunicarea unitară între calculator şi diferite dispozitive, ceea ce permite realizarea mai simplă a unor aplicaţii diverse. Diferitele sisteme de operare şi, în particular, sistemele de operare *Windows*, dispun de drivere pentru dispozitivele din clasa HID, care se pot utiliza pentru comunicaţia cu aceste dispozitive. Utilizarea acestor drivere este avantajoasă, deoarece astfel se elimină necesitatea scrierii unor drivere pentru comunicaţia cu dispozitivele respective.

Ca orice dispozitiv USB, un dispozitiv din clasa HID poate fi sursa sau destinaţia unei tranzacţii în fiecare cadru (1 ms) sau microcadru (125 μs). Un transfer reprezintă mai multe tranzacţii care creează un set de date cu o anumită structură şi o anumită semnificaţie pentru dispozitiv. În cazul dispozitivelor din clasa HID, un transfer se numeşte *raport*. Datele din cadrul rapoartelor au o anumită structură, care este specificată în descriptorii de raport (secţiunea 2.9.2).

Există trei tipuri de rapoarte: de intrare, de ieşire şi de caracteristici. Un raport de intrare este transmis de un dispozitiv şi conţine date destinate aplicaţiilor rulate pe calculator. Asemenea date sunt, de exemplu, coordonatele *x* şi *y* de la un dispozitiv indicator. Un raport de ieşire este transmis de calculator unui dispozitiv şi conţine date de la aplicaţii care sunt destinate unor comenzi sau afişaje. Un raport de caracteristici conţine date destinate unui dispozitiv sau date reprezentând starea unui dispozitiv. Spre deosebire de datele din rapoartele de intrare sau de ieşire, datele dintr-un raport de caracteristici sunt destinate utilizării de către utilitarele de configurare ale dispozitivelor şi nu de către aplicaţii. De exemplu, valoarea ratei de repetare a unei taste poate fi o dată dintr-un raport de caracteristici.

Anumite dispozitive pot avea mai multe structuri de rapoarte. De exemplu, o tastatură cu un dispozitiv indicator integrat poate raporta în mod independent datele referitoare la tastele apăsate şi cele referitoare la coordonate prin acelaşi punct terminal. Pentru diferenţierea acestor structuri, se utilizează un identificator al raportului, care este o valoare de un octet care precede fiecare raport. Pentru exemplul anterior, identificatorul permite driverului clasei de dispozitive HID să deosebească datele referitoare la taste de cele referitoare la coordonate.

2.9.2. Descriptori specifici pentru clasa HID

Pe lângă descriptorii standard utilizaţi pentru toate clasele de dispozitive USB, există descriptori specifici utilizaţi pentru clasa de dispozitive HID. Astfel de descriptori sunt descriptorul HID, descriptorul de raport şi descriptorul fizic. Figura 11.12 ilustrează încadrarea acestor descriptori în ierarhia descriptorilor USB standard.

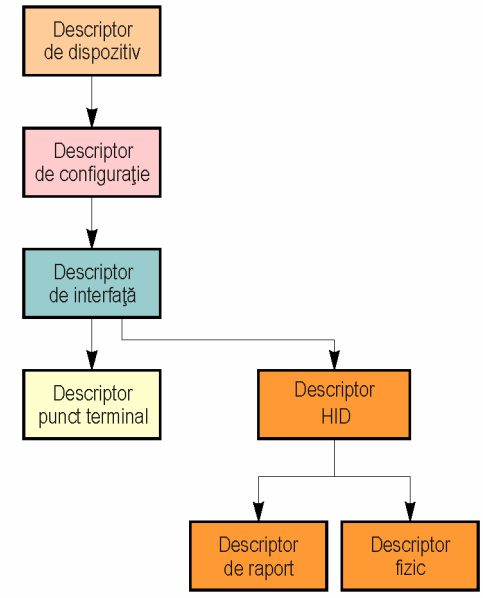


Figura 11.12. Încadrarea descriptorilor specifici clasei HID în ierarhia descriptorilor USB standard.

Un descriptor HID conţine numărul, lungimea şi tipul descriptorilor subordonaţi specifici clasei HID pentru un dispozitiv. Numărul descriptorilor subordonaţi descriptorului HID trebuie să fie cel puţin 1, deoarece trebuie să existe întotdeauna un descriptor de raport. Descriptorul HID mai conţine informaţii ca versiunea specificaţiilor clasei HID cu care este conform dispozitivul şi codul ţării în care este produs dispozitivul.

Un descriptor de raport conţine o serie de articole care descriu dimensiunea şi structura rapoartelor de date. Un asemenea descriptor furnizează informaţii despre datele raportate de dispozitiv pentru fiecare din funcţiile sale şi despre datele care sunt destinate diferitelor funcţii ale dispozitivului. Astfel de informaţii sunt dimensiunile datelor, dacă datele sunt absolute sau relative, valorile minime şi maxime ale datelor individuale. De asemenea, un descriptor de raport indică natura datelor din rapoartele de date, de exemplu, dacă ele reprezintă coordonatele *x* şi *y*.

Descriptorii fizici sunt opţionali. Aceştia furnizează informaţii despre partea sau părţile corpului uman care se utilizează pentru activarea comenzilor dispozitivului. De asemenea, un descriptor fizic poate conţine valori care cuantifică efortul care trebuie depus de utilizator pentru activarea diferitelor comenzi.

2.9.3. Comenzi USB specifice pentru clasa HID

Dispozitivele din clasa HID trebuie să implementeze comenzile USB standard *Get_Descriptor* şi *Set_Descriptor*. Pe lângă aceste comenzi, dispozitivele din clasa HID pot implementa diferite comenzi specifice acestei clase. Aceste comenzi iniţiază tranzacţii prin care calculatorul gazdă poate determina posibilităţile şi starea unui dispozitiv şi poate seta starea unor comenzi ale dispozitivului. Principalele comenzi specifice clasei HID sunt *Get_Report* şi *Set_Report*. Implementarea comenzii *Get_Report* este obligatorie.

Tabelul 11.7 prezintă comenzile *Get_Report* şi *Set_Report*.

Tabelul 11.7. Comenzile *Get_Report* şi *Set_Report* specifice dispozitivelor din clasa HID.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1010 0001	GET_REPORT (0x01)	Tip raport şi ID raport	Interfaţă	Lungime raport	Raport
0010 0001	SET_REPORT (0x09)	Tip raport şi ID raport	Interfaţă	Lungime raport	Raport

Comanda *Get_Report* permite calculatorului recepţionarea unui raport de la un dispozitiv prin conducta implicită de control. Câmpul wValue trebuie să conţină tipul raportului în octetul superior şi identificatorul raportului (ID raport) în octetul inferior. Tipul raportului specifică un raport de intrare (0x01), un raport de ieşire (0x02), sau un raport de caracteristici (0x03). Dacă nu se utilizează identificatori ai rapoartelor, octetul inferior al câmpului wValue trebuie setat la 0.

Comanda *Get_Report* este utilă în momentul iniţializării dispozitivelor pentru determinarea stării acestora. Comanda nu este destinată pentru a interoga starea dispozitivelor la intervale regulate. Pentru rapoartele de intrare repetate, trebuie să se utilizeze o conductă de întrerupere de intrare. În mod opţional, pentru rapoartele de ieşire se poate utiliza o conductă de întrerupere de ieşire (secţiunea 2.9.4).

Comanda *Set_Report* permite calculatorului să transmită un raport la un dispozitiv pentru setarea stării unor comenzi. Semnificaţia câmpurilor acestei comenzi este aceeaşi ca şi pentru comanda *Get_Report*.

2.9.4. Interfaţa cu dispozitivele din clasa HID

Un dispozitiv din clasa HID comunică cu driverul acestei clase fie prin conducta implicită de control, fie printr-o conductă de întrerupere. Dispozitivul utilizează conducta implicită de control pentru următoarele operaţii:

- Recepţionarea comenzilor USB transmise de calculator şi transmiterea răspunsului la aceste comenzi;

- Transmiterea datelor atunci când dispozitivul este interogat de driverul clasei HID prin comanda *Get_Report*;
- Recepționarea datelor de la calculator.

Driverul clasei HID utilizează o conductă de întrerupere pentru următoarele operații:

- Recepționarea unor date de la dispozitiv în mod asincron (date care nu au fost solicitate în mod explicit);
- Transmiterea cu o întârziere redusă a unor date la dispozitiv.

Utilizarea unei conducte de întrerupere de ieşire este opțională. Dacă un dispozitiv inițializează un punct terminal de întrerupere cu direcția de ieşire, atunci rapoartele de ieşire sunt transmise de calculator la dispozitiv prin această conductă. Dacă nu este disponibil un punct terminal de întrerupere cu direcția de ieşire, atunci rapoartele de ieşire sunt transmise de calculator prin punctul terminal de control, utilizând comenzi *Set_Report*.

2.10. Comunicația cu dispozitivele din clasa HID

La conectarea unui dispozitiv din clasa HID la un port USB al calculatorului, sistemul de operare execută procesul de enumerare, în modul descris în secțiunea 2.7. Acest proces nu implică utilizarea unor drivere specifice clasei HID sau specifice dispozitivului. La începutul procesului de enumerare, driverul USB solicită dispozitivului transmiterea descriptorului de dispozitiv, după care preia descriptorul şi asignează o adresă unică dispozitivului. În continuare, driverul USB solicită dispozitivului transmiterea descriptorului de configurație şi preia acest descriptor (care include descriptorii de interfață şi descriptorii punctelor terminale). Din descriptorul de interfață, driverul USB determină clasa dispozitivului, iar dacă această clasă este HID, driverul USB predă controlul driverului acestei clase. Dacă descriptorul de interfață nu specifică o anumită clasă, atunci driverul USB selectează un driver corespunzător pentru dispozitiv pe baza identificatorului producătorului şi al produsului, sau se solicită specificarea unui driver specific dispozitivului.

Dacă ce dispozitivul este recunoscut de sistemul de operare ca un dispozitiv HID, se pot utiliza funcțiile de sistem pentru scrierea unei aplicații de comunicație cu dispozitivul. Aceste funcții utilizează driverul dispozitivelor din clasa HID, astfel încât nu va fi necesară scrierea unui driver specific dispozitivului. În această secțiune se descriu operațiile necesare pentru comunicația cu un dispozitiv din clasa HID în cazul sistemelor de operare *Windows*. Aceste operații sunt diferite pentru alte sisteme de operare, cum sunt *Linux* sau *MacOS*.

Înaintea comunicației propriu-zise cu un dispozitiv din clasa HID, este necesară stabilirea comunicației cu acest dispozitiv, ceea ce presupune mai multe etape. Aceste etape sunt descrise în continuare.

1. Se apelează funcția *HidD_GetHidGuid* pentru a obține identificatorul unic global GUID (*Globally Unique Identifier*) pentru dispozitivele din clasa HID. Parametrul acestei funcții este un pointer la o structură de tip `_GUID`, care reprezintă un buffer în care va fi returnat identificatorul GUID. Această funcție este declarată în fişierul hidsdi.h, care face parte din pachetul de dezvoltare Microsoft DDK (*Driver Development Kit*). Pentru linkeditarea statică este necesară utilizarea fişierului hid.lib.
2. Se apelează funcția de sistem *SetupDiGetClassDevs* pentru a obține un set de informații despre toate dispozitivele din clasa HID care sunt ataşate la sistem. Funcția returnează un indicator (variabilă de tip `HANDLE`) la setul de informații despre lista de dispozitive. Unul din parametrii de apel ai acestei funcții este un pointer la identificatorul global GUID care a fost obținut în etapa anterioară. Această funcție este declarată în fişierul setupapi.h, iar pentru linkeditarea statică este necesară utilizarea fişierului setupapi.lib.
3. Se apelează funcția de sistem *SetupDiEnumDeviceInterfaces* pentru a obține informații despre interfața unui dispozitiv din lista dispozitivelor din clasa HID. Unul din parametrii acestei funcții este indicatorul la setul de informații obținut în etapa 2, iar un alt parametru este pointerul la identificatorul global GUID. Dispozitivul despre a cărui interfață se solicită

informații este specificat printr-un parametru reprezentând indexul în lista de dispozitive. Funcția completează o structură de tip `SP_DEVICE_INTERFACE_DATA` cu informațiile despre interfața dispozitivului şi returnează un pointer la această structură într-un parametru de ieşire. Înaintea apelului funcției, trebuie să se completeze membrul `cbSize` al structurii cu `sizeof (SP_DEVICE_INTERFACE_DATA)`. În caz de succes, funcția returnează o valoare diferită de zero. Această funcție este declarată în fişierul setupapi.h.

4. Dacă apelul funcției din etapa 3 s-a realizat cu succes, se apelează funcția de sistem *SetupDiGetDeviceInterfaceDetail* pentru a afla informații detaliate despre interfața dispozitivului selectat în etapa 3. Unul din parametrii acestei funcții este indicatorul la setul de informații, care a fost obținut în etapa 2. Un alt parametru este pointerul la structura cu informațiile despre interfața dispozitivului, care a fost obținut prin apelul funcției din etapa 3. Funcția completează o structură de tip `SP_DEVICE_INTERFACE_DETAIL_DATA` cu calea de acces la dispozitivul selectat şi returnează un pointer la această structură într-un parametru de ieşire. Structura conține doi membri, `cbSize` şi `DevicePath`, al doilea membru fiind un şir de caractere de lungime variabilă terminat cu un octet zero. Înaintea apelului funcției, membrul `cbSize` trebuie completat cu `sizeof (SP_DEVICE_INTERFACE_DETAIL_DATA)`, care reprezintă dimensiunea părții fixe a structurii, şi nu dimensiunea şirului de lungime variabilă. Membrul `DevicePath` va fi completat de funcția apelată cu calea de acces la dispozitiv, cale care va putea fi transmisă ca parametru pentru apelul funcției *CreateFile*. Un alt parametru trebuie să specifice dimensiunea totală a structurii amintite. În caz de succes, funcția returnează o valoare diferită de zero. Această funcție este declarată în fişierul setupapi.h.
5. Se apelează funcția de sistem *CreateFile* pentru a deschide comunicația cu dispozitivul, utilizând calea de acces care a fost obținută în etapa 4.
6. Dacă indicatorul returnat de funcția *CreateFile* nu este valid, se continuă cu etapa 8. Dacă această funcție returnează un indicator valid de fişier, se determină şirul de identificare al dispozitivului din clasa HID prin apelul funcției *HidD_GetProductString*. Parametrii acestei funcții sunt indicatorul returnat de funcția *CreateFile*, pointerul la un buffer alocat de utilizator în care funcția va depune şirul de identificare şi lungimea bufferului alocat. Această funcție este declarată în fişierul hidsdi.h, care face parte din pachetul de dezvoltare Microsoft DDK.
7. Se compară şirul de identificare obținut în etapa 6 cu şirul de identificare al dispozitivului cu care trebuie să se realizeze comunicația. Trebuie să se țină cont că şirul de identificare al dispozitivului este reprezentat în codul Unicode. Dacă cele două şiruri sunt diferite, se închide fişierul deschis în etapa 5 prin apelarea funcției de sistem *CloseHandle* şi se continuă cu etapa 8. Dacă şirurile sunt identice, se apelează funcția *SetupDiDestroyDeviceInfoList* pentru eliberarea memoriei alocate pentru informațiile despre dispozitive. În acest moment, operația de stabilire a comunicației cu dispozitivul este terminată.
8. Se incrementează indexul dispozitivului şi, dacă acesta nu a ajuns la o anumită valoare maximă, se revine la etapa 3 pentru a obține informații despre interfața următorului dispozitiv. Valoarea maximă poate fi setată, de exemplu, la 20.

După stabilirea comunicației cu dispozitivul, aplicația poate prelua rapoarte de intrare de la dispozitiv prin apelul funcției *ReadFile* sau *ReadFileEx* şi poate transmite rapoarte de ieşire la dispozitiv prin apelul funcției *WriteFile*. Pentru apelul acestor funcții, se utilizează indicatorul de fişier returnat de funcția *CreateFile*.

Funcțiile *ReadFile* şi *ReadFileEx* preiau rapoartele de intrare de la dispozitiv utilizând transferuri de întrerupere. Aceasta înseamnă că dispozitivul trebuie să transmită aceste rapoarte utilizând transferuri de întrerupere printr-un punct terminal de intrare. Primul octet din fiecare raport reprezintă identificatorul raportului. Începând cu sistemul de operare *Windows* XP, este posibilă preluarea rapoartelor de intrare utilizând transferuri de control. Pentru aceasta se poate utiliza funcția *HidD_GetInputReport*, care va transmite dispozitivului o comandă *Get_Report*.

Funcția *WriteFile* transmite rapoartele de ieşire prin comenzi *Set_Report*. Dacă dispozitivul nu are un punct terminal de întrerupere cu direcția de ieşire, funcția *WriteFile* utilizează transferuri de control, în caz contrar utilizând transferuri de întrerupere. Primul octet din fiecare raport reprezintă identificatorul raportului. Începând cu sistemul de operare *Windows* XP, se poate utiliza funcția *HidD_SetOutputReport* pentru transmiterea unui raport de ieşire printr-un transfer de control.

3. Desfăşurarea lucrării

3.1. Răspundeți la următoarele întrebări:

a. Care este rolul terminatorilor utilizați la magistrala USB ?
b. De ce nu este posibilă conectarea a două calculatoare printr-un cablu USB obişnuit ?
c. Care este deosebirea dintre transferurile asincrone şi transferurile izocrone pe magistrala USB?
d. Ce reprezintă punctele terminale şi cum se realizează comunicația între calculatorul gazdă şi un dispozitiv USB prin intermediul punctelor terminale ?
e. Ce informații conține un descriptor de dispozitiv şi un descriptor de configurație ?
f. Care sunt avantajele şi dezavantajele utilizării modelului clasei HID pentru comunicația cu perifericele ?

3.2. Creați o aplicație de tip consolă. Scrieți o funcție pentru stabilirea comunicației între calculator şi placa de dezvoltare CP-JR51USB prin interfața USB. Placa este configurată ca un dispozitiv HID, şirul de identificare al acestuia fiind "Butoane si leduri JR51USB". Funcția trebuie să returneze valoarea booleană TRUE în cazul în care comunicația cu placa a fost stabilită şi valoarea FALSE în caz contrar. Pentru scrierea funcției, parcurgeți etapele descrise în secțiunea 2.10, cu următoarele observații:

- Consultați biblioteca MSDN (http://msdn.microsoft.com/library/) pentru scrierea apelurilor de funcții.
- Copiați în directorul proiectului fişierele din arhiva Hid.zip, disponibilă pe pagina laboratorului. Includeți fişierele setupapi.h şi hidsdi.h în proiectul aplicației şi adăugați directivele `#include` corespunzătoare în fişierul sursă. În setările proiectului, adăugați fişierele setupapi.lib şi hid.lib în lista modulelor de bibliotecă (*Project* → *Settings* → *Link*).
- Pentru apelul funcției *HidDGetHidGuid*, adăugați următoarele linii în fişierul sursă:

```
struct _GUID GUID;
HidD_GetHidGuid (&GUID);
```

- Pentru apelul funcției *SetupDiGetClassDevs*, adăugați următoarele linii:

```
HANDLE PnPHandle;
PnPHandle = SetupDiGetClassDevs (&GUID, NULL, NULL,
            DIGCF_PRESENT | DIGCF_INTERFACEDEVICE);
```

  Afişați un mesaj de eroare dacă funcția returnează valoarea `INVALID_HANDLE_VALUE`.

- Definiți atributele de securitate necesare pentru apelul funcției *CreateFile* prin următoarele linii:

```
SECURITY_ATTRIBUTES SecurityAttributes;
SecurityAttributes.nLength = sizeof (SECURITY_ATTRIBUTES);
SecurityAttributes.lpSecurityDescriptor = NULL;
SecurityAttributes.bInheritHandle = FALSE;
```

- La apelul funcției *CreateFile*, setați parametrul pentru modul de acces la `GENERIC_READ | GENERIC_WRITE`, iar parametrul pentru modul de partajare la `FILE_SHARE_READ | FILE_SHARE_WRITE`. Afişați un mesaj de eroare dacă funcția *CreateFile* returnează valoarea `INVALID_HANDLE_VALUE`.
- Pentru apelul funcției *HidD_GetProductString*, adăugați următoarele linii:

```
char cBuffer [256];
HidD_GetProductString (hFile, cBuffer, sizeof (cBuffer));
```

unde *hFile* este indicatorul returnat de funcția *CreateFile*.

După scrierea funcției, adăugați apelul acestei funcții în programul principal. Rulați programul fără ca placa de dezvoltare CP-JR51USB să fie conectată la calculator. Conectați placa de dezvoltare la calculator printr-un cablu USB, alimentați placa cu o tensiune de 9 V şi poziționați comutatorul USB ON/OFF pe poziția USB ON. Verificați dacă placa este recunoscută de calculator ca un dispozitiv din clasa HID. Rulați din nou programul şi verificați dacă a fost posibilă stabilirea comunicației cu placa de dezvoltare.

3.3. Scrieți un program pentru setarea stării diodelor LED ale plăcii de dezvoltare CP-JR51USB prin interfața USB. Utilizați funcția scrisă la aplicația 3.2 pentru stabilirea comunicației cu placa de dezvoltare. Apelați funcția *WriteFile* pentru transmiterea unui raport de ieşire de doi octeți, primul octet fiind identificatorul raportului (0x00), iar al doilea octetul de comandă al celor patru diode LED de pe placă. Placa va prelua acest octet printr-o comandă *Set_Report* pe conducta implicită de control. În octetul de comandă, asignarea biților este următoarea:

- Bitul 3: dioda LED roşie;
- Bitul 2: dioda LED galbenă;
- Bitul 1: dioda LED verde;
- Bitul 0: dioda LED albastră.

Pentru aprinderea unei diode, bitul corespunzător trebuie setat la 1. Programul va solicita introducerea valorii octetului de comandă de la tastatură.

3.4. Scrieți un program pentru citirea stării butoanelor de pe placa de dezvoltare CP-JR51USB prin interfața USB. Utilizați funcția scrisă la aplicația 3.2 pentru stabilirea comunicației cu placa de dezvoltare. Apelați funcția *ReadFile* pentru preluarea unui raport de intrare de doi octeți, primul octet fiind identificatorul raportului (0x00), iar al doilea octetul de stare al butoanelor de pe placă. Placa transmite în mod repetat octetul de stare utilizând transferuri de întrerupere. În octetul de stare, asignarea biților este următoarea:

- Bitul 2: butonul INT0;
- Bitul 1: butonul de sub dioda LED roşie;
- Bitul 0: butonul de lângă afişajul cu 7 segmente.

Valoarea 0 a unui bit indică starea apăsată a butonului corespunzător. Programul va afişa pe ecran valoarea octetului de stare.

Bibliografie

[1] Allman, S., “Using the HID class eases the job of writing USB device drivers”, EDN, September 19, 2002, http://www.edn.com/contents/images/243218.pdf.

[2] Axelson, J., *USB Complete*, Third Edition, Lakeview Research, http://www.lvr.com/usbc.htm.

[3] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips, “Universal Serial Bus Specification”, Revision 2.0, 2000, http://www.usb.org/developers/docs/usb_20_02212005.zip.

[4] Hyde, J., “Learning USB by Doing”, Intel Corp., http://www.devasys.com/PD11x/JHWP.pdf.

[5] Microsoft Corp., Microsoft Driver Development Kit for Windows, 2005.

[6] Microsoft Corp., MSDN Library, 2005, http://msdn.microsoft.com/library/.

[7] Peacock, C., “On-The-Go Supplement – Point-to-Point Connectivity for USB”, Beyond Logic, 2005, http://www.beyondlogic.org/usb/otghost.htm.

[8] Peacock, C., “USB in a Nutshell – Making Sense of the USB Standard”, Beyond Logic, 2005, http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf.

[9] Rosch, W. L., *Hardware Bible*, Sixth Edition, Que Publishing, 2003.

[10] USB Implementers Forum, Inc., "A Technical Introduction to USB 2.0", http://www.usb.org/developers/usb20/developers/whitepapers/usb_20g.pdf.

[11] USB Implementers Forum, Inc., "Device Class Definition for Human Interface Devices (HID)", Version 1.11, 2001, http://www.usb.org/developers/devclass_docs/HID1_11.pdf.

[12] USB Implementers Forum, Inc., "HID Usage Tables", Version 1.12, 2005, http://www.usb.org/developers/devclass_docs/Hut1_12.pdf.

[13] USB Implementers Forum, Inc., "Introduction to USB On-The-Go", http://www.usb.org/developers/onthego/USB_OTG_Intro.pdf.

[14] USB Implementers Forum, Inc., "On-The-Go Supplement to the USB 2.0 Specification", Revision 1.0a, 2003, http://www.usb.org/developers/docs/usb_20_02212005.zip.

[15] USB Implementers Forum, Inc., "USB 2.0 Specification Engineering Change Notice (ECN) #1: Mini-B Connector", 2000, http://www.usb.org/developers/docs/ecn1.pdf.

[16] USB Implementers Forum, Inc., "USB Frequently Asked Questions", http://www.usb.org/developers/usbfaq/.

[17] USB Implementers Forum, Inc., "USB On-The-Go", http://www.usb.org/developers/onthego/.

[18] USB Implementers Forum, Inc., "USB Specification Expanding, Boosting Performance Up to 40 Times Beyond Current Capability", http://www.usb.org/developers/usb20/backgrounder/.