**Úvod.**

Textová USB vysielačka sú dve USB zariadenia, ktoré umožňujú bezdrôtovú textovú komunikáciu medzi dvomi PC na vzdialenosť až niekoľkých kilometrov pomocou jednoduchého programu. Toto použitie je však treba považovať len za príklad, pretože jedno zo zariadení sa dá ľahko upraviť aj k meraniu, regulácii či riadeniu procesov. Dáta a riadiace povely budú potom sprostredkovávané bezdrôtovo z PC do programu, upravenému na tento účel.
USB textová vysielačka pozostáva z bezdrôtového modulu RFM69W pracujúceho v pásme 433 MHz so širokou škálou nastavení a jednočipového mikrokontroléra PIC16F1454, obsahujúceho port USB 2.0. Mikrokontrolér sprostredkováva komunikáciu medzi PC a bezdrôtovým modulom a tiež zabezpečuje spoľahlivosť prenosu v prípade stratených dátových paketov napr. v dôsledku rušenia alebo slabého signálu.

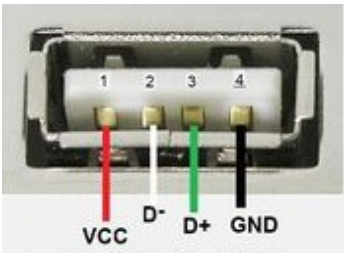
**Program v PIC16F1454**

Program pre PIC16F1454 je napísaný v assebbleri a pozostáva zo štyroch častí:

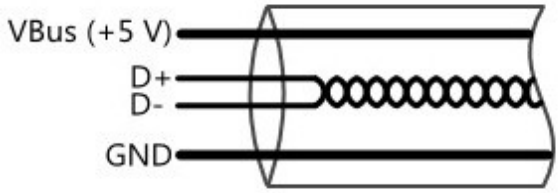
1. Obsluha USB komunikácie
2. Obsluha bezdrôtového modulu RFM69W
3. Sprostredkovanie odosielaných a prijímaných dat medzi PC a modulom RFM69W
4. Zabezpečenie spoľahlivosti bezdrôtovej komunikácie

**Obsluha USB komunikácie**

USB 2.0 port počítača pozostáva zo štyroch pinov:



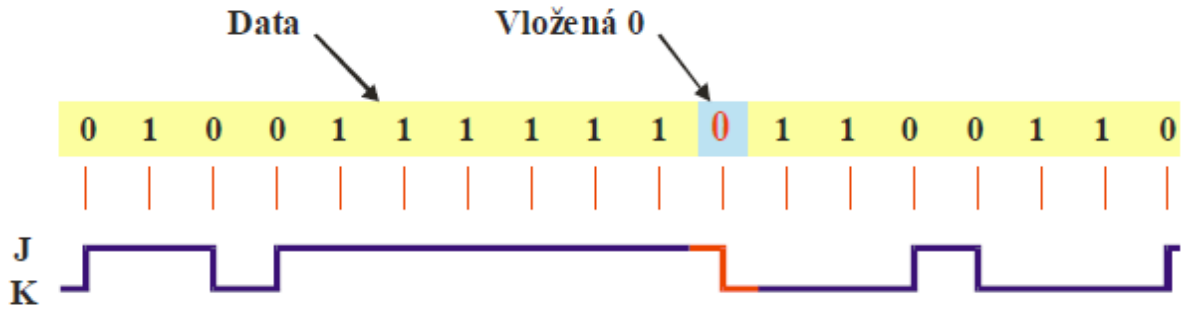

Medzi pinmi VCC+ a GND- sa nachádza napätie 5V, ktoré môže USB zariadenie použiť na svoje napájanie, štandardne s odberom prúdu max. 100mA. Je možné požiadať aj o viac, najviac však 500mA. Pri vyššom prúdovom odbere ako 500mA musí zariadenie použiť svoje vlastné napájanie.
Samotná obojsmerná sériová komunikácia prebieha pomocou striedania polarity napätia +/- medzi pinmi D+ a D- s napätím väčším, ako 200mV. (Pozor, toto je napätie medzi pinmi D+,D-, nie napätie pinov voči zemi (GND). Napätie voči zemi môže dosahovať úrovne až 3,3V) Takýto spôsob prenosu dat zabezpečuje vysokú odolnosť proti rušeniu, pretože v každej chvíli tečie ten istý prúd jedným vodičom tam a druhým späť alebo naopak, čím sa vyruší vonkajšie magnetické pole linky a teda aj citlivosť na magnetické rušenie z okolia. V prípade USB kábla však musia byť oba vodiče linky ešte skrútené.




- VBus – napájanie (červená)
- D+ – dáta (zelená)
- D- – dáta (biela)
- GND – zem (čierna)

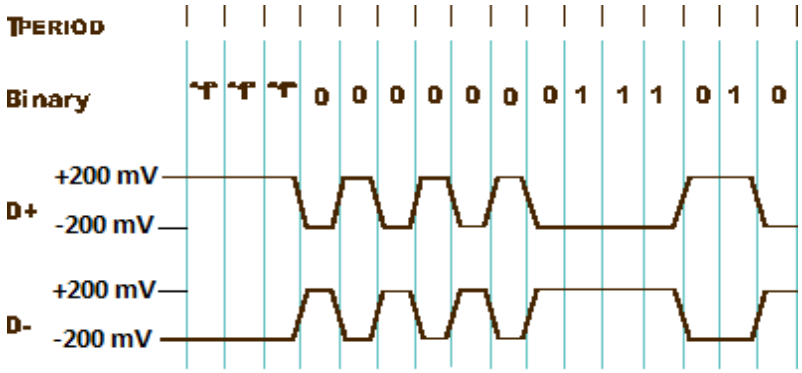
Spôsob, akým sa dáta elektricky prenášajú, sa volá NRZI (non-return-to-zero inverted), čiže zmena polarity medzi D+, D- znamená podľa dohody logickú 0 a stav bez zmeny polarity logickú 1. Pretože logické jedničky sa prenášajú bez zmeny polarity (čiže počas ich prenosu sa na linke nič nedeje), musí byť definovaná aj prenosová rýchlosť, aby príjemca vedel, koľko jedničiek má v danom časovom úseku považovať za platné. V prípade, že by sa vyskytla v dátach príliš dlhá rada jednotiek za sebou, narastá pravdepodobnosť chyby pri odmeriavaní času (lebo žiadne „hodiny“ nie sú dokonalé) mohlo by sa stať, že príjemca vyhodnotí o jednu

jednotku naviac alebo menej, takže by vznikla by chyba pri prenose. Preto sa rozhodlo, že po súvislom slede šiestich jednotiek sa vloží jedna nula (ktorú si potom príjemca po prijatí správy odstráni). Táto metóda zabezpečenia sa volá „bit stuffing“.




Z obrázku pekne vidíme, že v prípade dlhého radu jednotiek, zostáva napätie na linke na rovnakej úrovni čiže bez zmeny.

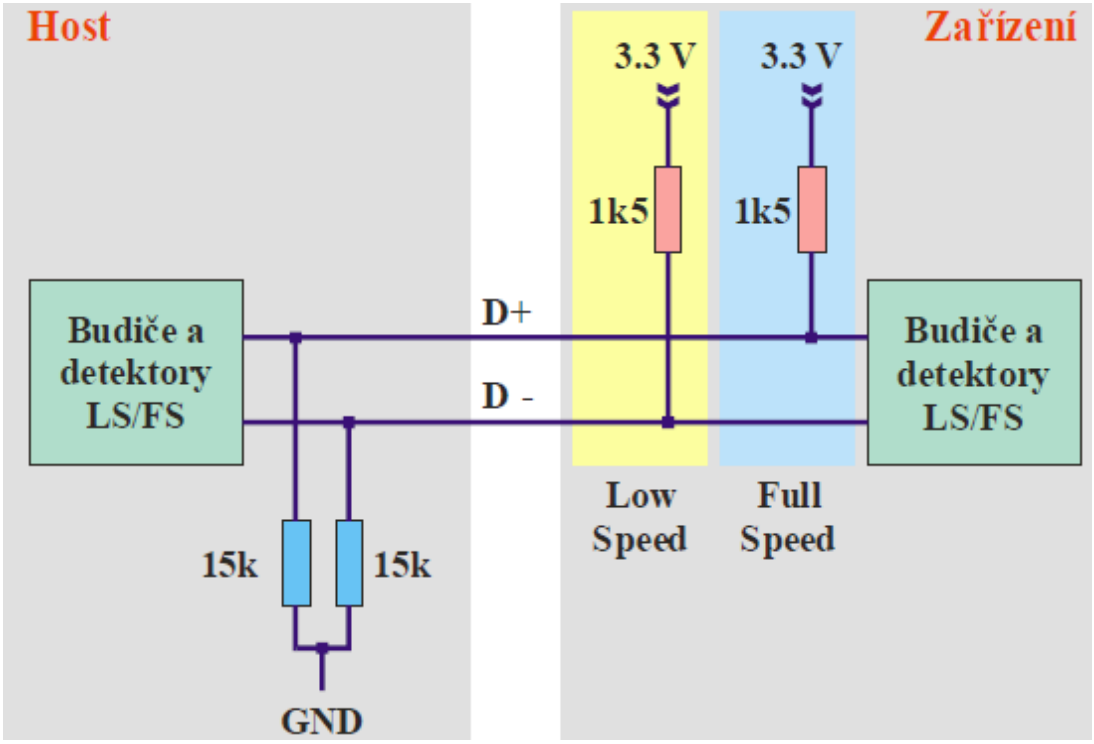
Dáta sa po USB linke posielajú ako už býva zvykom, po paketoch. Na začiatku každého paketu je séria siedmich núl zakončená ôsmou jednotkou. (8 bitov)




Na obrázku vidno, že to vyvolalo súvislé a pravidelné striedanie polarity na linke medzi D+ a D- rýchlosťou, akou sa budú prenášať dáta. Príjemca si na tomto úseku zosynchronizuje svoje hodiny, ktorými odmeriava čas. Tým sa zabezpečí, že zvyšok paketu bude prijatý presne a nenastanú chyby kvôli nepresnému meraniu času bitových úsekov. Štandard USB 2.0 používa tri prenosové rýchlosti:

- Low speed (LS) – 1,5 Mbit/s
- Full speed (FS) – 12 Mbit/s
- High speed (HS) – 480 Mbit/s

O tom, akou prenosovou rýchlosťou sa bude komunikovať rozhoduje zariadenie, ktoré pripojíme k počítaču (hostovi) a to konkrétne pripojením napätia 3,3 V na jeden z pinov D+ alebo D- cez odpor 1,5 kOhm voči zemi GND. Keď zariadenie pripojí svoj odpor na pin D+ oznámi tým počítaču, že sa bude komunikovať rýchlosťou Full speed (12 Mbit/s). Keď pripojí odpor na pin D-, počítač bude vedieť, že má komunikovať rýchlosťou Low speed (1,5 Mbit/s). Pripojením jedného z týchto dvoch odporov je to pre PC zároveň signál, že na USB port bolo pripojené nejaké nové zariadenie.




Od chvíle, keď je pripojený jeden z týchto odporov, počítač vie o zariadení a začne sa ho dopytovať na rôzne informácie. Napr. čo je to za typ a podobne. Deje sa tak pomocou jednoduchých čísel začlenených do paketov, kde každé číslo niečo znamená. Zariadenie potom počítaču odpovedá (opäť len číslami). O tom, čo ktoré číslo znamená, v akom poradí, štruktúre a akej situácii sa majú posielať, bolo už vopred rozhodnuté vývojármi USB zbernice. Kompletný zoznam, popisy a postupy je možné dozvedieť sa na ich stránkach www.usb.org

Existuje niekoľko typov paketov. Paket vždy začína sériou siedmych núl ako som už spomínal, zakončených ôsmou jednotkou, kvôli synchronizácii hodín príjemcu. Tento bajt sa preto volá **Sync**. Po ňom vždy nasleduje bajt PID (Packet ID), čiže identifikačné číslo paketu. Z neho sa príjemca dozvie, čo je to za typ paketu a tiež aj smer prenosu – tj. či sa od zariadenia v nasledujúcom pakete požaduje vysielanie alebo má naopak prijímať – tj., či host od zariadenia dáta požaduje alebo naopak mu nejaké dáta posiela.. USB komunikáciu vždy začína host (počítač). Zariadenia nikdy nesmú svojvoľne vysielať bez vyzvania hostom. **Počítač je teda v celej komunikácii master.** Ďalšie bajty paketu sa už rôznia podľa toho o aký typ ide. Rozoznávame tieto 4 typy paketov:

| | | | | | | |
|---|---|---|---|---|---|---|
| Token | Sync | PID | Addr. | Endp. | CRC5 | EOP |
| Data | Sync | PID | Data | CRC16 | EOP | |
| Handshake | Sync | PID | EOP | | | |
| SOF | Sync | PID | Frame number | | CRC5 | EOP |

- Sync – To už som vysvetlil. Slúži na synchronizáciu hodín zariadenia s hostom. V prípade Low a Full speed sa používa 8 bitov.
- PID – Pakcet ID, to som tiež už vysvetlil: identifikačné číslo paketu. Určuje typ paketu.
- Addr. – Address (adresa) zariadenia
- Endp. – Číslo Endpointu
- Data – Dáta paketu
- Frame number – 11bitové poradové číslo rámca
- CRC5 – kontrolný súčet
- CRC16 – kontrolný súčet
- EOP – End of packet, je znamenie konca paketu o dĺžke trvania 3 bitov, kde počas prvých dvoch bitov zmizne napätie medzi linkami D+ a D- (tomuto stavu sa hovorí SE0) a v treťom bite opäť nabehne

napätie do normálneho kľudového stavu

**Token pakety:**
Token pakety posiela vždy PC a obsahujú v sebe inštrukciu pre zariadenie, čo bude nasledovať . Ako som spomenul, žiadne USB zariadenie pripojené k PC si nesmie vysielať len tak samo od seba kedy sa mu zachce, ale musí počkať na výzvu od PC. Na to slúžia práve token pakety. Existujú 3 druhy token paketov: SETUP, IN, OUT.

*SETUP token paket:*
Slúži na nastavenie alebo zistenie parametrov zariadenia. Ako každý, obsahuje i tento na začiatku bajt Sync. Jeho význam som už vysvetlil. (Synchronizácia hodín.) Ďalej nasleduje bajt PID, čiže identifikačné číslo paketu. V prípade SETUP token paketu má toto číslo hodnotu 0xB4. (0x... znamená, že je to zápis v 16-kovej sústave. Samotné číslo je len B4.)

| Sync | SETUP | ADDR | ENDP | CRC5 | EOP |
|---|---|---|---|---|---|
| 00000001 | 0xB4 | 0x00 | 0x0 | 0x08 | 001 |

Ďalší bajt paketu je adresa zariadenia. SETUP token paket s nulovou adresou posiela počítač len na začiatku, keď bolo zariadenie práve zasunuté do USB portu. Keď mu bude pridelená adresa, už tam nebude 0x00, ale číslo adresy – napr. 0x02. Počítač každému pripojenému zariadeniu k USB zbernici priradí adresu na ktorej bude komunikovať. Najprv má ale hodnotu 0x00, pretože zariadenie sme ešte len pripojili k PC. Ďalší bajt paketu je číslo Endpointu, ktorého sa bude týkať komunikácia. Endpoint je pamäť, ktorú má zariadenie v sebe vyhradenú pre ukladanie prichádzajúcich dát, a/alebo na odosielanie dát. Takýchto pamäťových priestorov (Endpointov) môže byť viac. Minimálne však aspoň jeden s poradovým číslom 0x00, pretože inak by nebolo kam ukladať a/alebo odkiaľ odosielať dáta. Ďalší bajt paketu je kontrolný súčet CRC vypočítaný z predošlých dát. Slúži na zabezpečenie spoľahlivosti prenosu. Príjemca si z prijatých dat vypočíta nezávisle na tom svoj vlastný kontrolný súčet a porovná s tým prijatým v pakete. Ich hodnoty sa musia zhodovať. Ak sa nezhodujú znamená to, že pri prenose nastala chyba. V takom prípade dá príjemca povel odosielateľovi na opakovanie dát. Posledný úsek paketu sa volá EOP (End Of Paket) = koniec paketu. To už nie je bajt, ale len 3 bity, ktoré majú vždy hodnotu 001. Je to znamenie pre príjemcu, že tu paket končí. Toto znamenie je zvláštne tým, že z linky D+/D- zmizne napätie po dobu trvania dvoch bitov. (To sú tie prvé dve nuly.) Už som spomínal, že na linke medzi pinmi D+ a D- je počas prenosu dat vždy napätie 200 mV. Akurát sa mení jeho polarita. Keď ale toto napätie z linky zmizne, je to znamenie pre príjemcu, že paket je už kompletný a nastal jeho koniec. (Tento stav sa volá SE0.) Hneď potom napätie znovu nabehne do východzieho (kľudového) stavu. (To je ten tretí bit s hodnotou 1.) Linka je potom znovu pripravená na prenos ďalších dat.

*OUT token paket:*
Tento token paket posiela počítač vtedy, keď má pre zariadenie nejaké dáta. Preto sa volá OUT(put) – výstupný. Smer prenosu sa vždy určuje z pohľadu počítača, nie zariadenia. Počítač je totiž v celom procese USB komunikácie master. Všetky bajty tohto paketu majú rovnaký význam, ako v prípade SETUP token paketu. Jediný rozdiel je hodnota identifikačného čísla PID, ktorá je v tomto prípade 0x87. Podľa nej zariadenie pozná, že sa je to OUT token paket – čo inými slovami znamená, že v ďalšej komunikácii budú nasledovať dáta určené pre zariadenie.

| Sync | OUT | ADDR | ENDP | CRC5 | EOP |
|---|---|---|---|---|---|
| 00000001 | 0x87 | 0x00 | 0x0 | 0x08 | 001 |

*IN token paket:*
Tento token paket posiela počítač vtedy, keď požaduje od zariadenia nejaké dáta. Preto sa volá IN(put) – vstupný. Smer prenosu sa vždy určuje z pohľadu počítača, nie zariadenia. Takže je vstupný smerom do PC. Všetky bajty tohto paketu majú rovnaký význam, ako v prípade SETUP alebo OUT token paketu. Rozdielna je len hodnota identifikačného čísla PID, ktorá je 0x96. Podľa nej zariadenie zistí, že sa sa jedná o IN token paket – čo inými slovami znamená, že v ďalšej komunikácii sa budú dáta od zariadenia vyžadovať. (Zariadenie bude posielať svoje dáta smerom do počítača.)

| Sync | IN | ADDR | ENDP | CRC5 | EOP |
|---|---|---|---|---|---|
| 00000001 | 0x96 | 0x00 | 0x0 | 0x8 | 001 |

**Data pakety:**
Dátový paket môže posielať jak počítač tak zariadenie a slúži na samotný prenos dát. Neobsahuje adresu ani číslo endpointu. To bolo úlohou token paketu, ktorý počítač poslal zariadeniu pred ním. Jeho štruktúra je nasledovná:

| Sync | DATA0 | DATA | CRC16 | EOP |
|---|---|---|---|---|
| 00000001 | 0xC3 | 80 06 00 02 00 00 FF 00 | 0x9725 | 001 |

Alebo:

| Sync | DATA1 | DATA | CRC16 | EOP |
|---|---|---|---|---|
| 00000001 | 0xD2 | 09 02 19 00 01 01 00 60 | 0x5020 | 001 |

Vidíme, že dátové pakety sú dva. Dôvodom je lepšie zabezpečenie spoľahlivosti prenosu. Keď totiž treba posielať viac dát, rozdelených do viacerých dátových paketov za sebou, posielajú sa striedavo pakety DATA0, DATA1, DATA0, DATA1, atď.... Ak jeden vypadne kvôli chybe prenosu, príjemca zistí porušenie tejto pravidelnosti striedania a oznámi odosielateľovi chybu. Ten potom dáta opakuje.
Ako všetky pakety, i tieto dátové, začínajú bajtom Sync. Jeho význam som už vysvetlil. (Synchronizácia hodín príjemcu podľa odosielateľa.) Ďalej nasleduje bajt PID, čiže identifikačné číslo paketu. V prípade DATA0 paketu má toto číslo hodnotu 0xC3 a v prípade DATA1 paketu hodnotu 0xD2. Podľa nich príjemca rozlíši, že sa jedná o dátové pakety a tiež o ktorý konkrétne. Ďalej v pakete nasleduje oblasť, ktorá sa volá DATA. Schválne píšem „oblasť“, lebo už to nie je len jeden bajt, ale minimálne 8 bajtov. Môže ich byť aj viac. Zariadenie to však musí počítaču oznámiť pri prvotnom nastavení, koľko bajtov má oblasť DATA obsahovať. Na obrázku sú písané v 16-kovej sústave, len tam chýba pred číslami 0x.., aby zápis vyzeral krajšie a prehľadnejšie. Bajty v tejto oblasti sú to najhlavnejšie, o čo v celom prenose ide. Jedná sa o hlavné dáta. Každé číslo tam niečo znamená. O tom čo, rozhodli vývojári USB zbernice a implementovali do ovládačov operačného systému. Naše zariadenie, ktoré v projekte popisujem sa muselo tomu ich rozhodnutiu prispôsobiť, aby správne komunikovalo cez USB zbernicu. Význam čísel a ich postupnosť odosielania popíšem neskôr. Je rôzny v závislosti na druhu zariadenia – napr. myš, klávesnica, dátové úložisko, atď.. Na začiatku, pri prvom zasunutí do USB portu je však prvých pár dátových štruktúr spoločných pre všetky USB zariadenia.
Ďalšia oblasť paketu je kontrolný súčet CRC, ktorý má v tomto prípade 2 bajty, pretože množstvo dat, z ktorého sa vypočítavá je väčší ako v prípade ostatných typov paketov. Ako funguje CRC som už vysvetľoval. Na konci paketu je sekvencia EOP (End Of Paket) – koniec paketu zložená z 3 bitov. Jej význam a fungovanie som tiež už vysvetlil.

**Handshake pakety.**
Tento paket posiela jak počítač tak zariadenie, v závislosti od smeru prenosu. Slúži na potvrdenie alebo odmietnutie prijatia predošlých paketov. Preto býva posielaný na konci paketovej sekvencie. Je najkratší zo všetkých paketov. Začína ako tradične bajtom sync. Pokračuje bajtom PID (identifikačným číslom paketu), ktorý má v prípade ACK (potvrdenie prijatých dat) hodnotu 0x4B. Posledná je už tradične známa sekvencia EOP, označujúca koniec paketu. Viac toho netreba. Keď príjemca pošle odosielateľovi paket ACK, odosielateľ podľa čísla 0x4B okamžite spozná, že príjemca všetky pakety v sekvencii prijal správne.

| Sync | ACK | EOP |
|---|---|---|
| 00000001 | 0x4B | 001 |

Príjemca (zariadenie) však môže dáta aj odmietnuť. Vtedy pošle ako odpoveď na prijímané dáta, paket NAK, ktorého identifikačné číslo je 0x5A. Ak zariadenie pošle tento paket vtedy, keď počítač od neho dáta požaduje

(tj obdržal od PC IN token paket) – znamená to, že zariadenie nemá pre počítač žiadne dáta k odoslaniu.

| Sync | NAK | EOP |
|---|---|---|
| 00000001 | 0x5A | 001 |

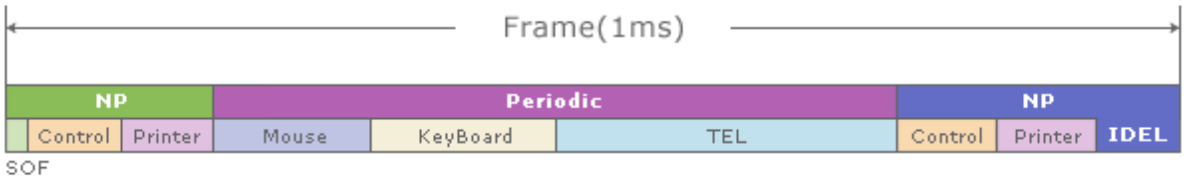
Paket STALL je s identifikačným číslom 0x78 sa posiela sa v prípade odmietnutia požiadavky (chybový stav zariadenia).

| Sync | STALL | EOP |
|---|---|---|
| 00000001 | 0x78 | 001 |

**SOF paket:**
Tento paket posiela počítač vždy na začiatku každého rámca. Obsahuje poradové číslo rámca. Rámec je časový úsek trvajúci 1 milisekundu (v prípade Full a Low speed komunikácie). Do tohto časového rámca je umiestnená jedna dátová transakcia. Ďalšia bude až v ďalšej milisekunde. V prípade Full speed komunikácie ktorú používam v tomto projekte (12 Mbit/s) sa do rámca zmestí 12000 bitových intervalov, čiže 1500 bajtov. Jedna transakcia ale obsahuje podstatne menej bajtov, než 1500, takže do rámca sa zmestia desiatky transakcii. Pre jedno zariadenie tam však počítač iniciuje len jednu dátovú transakciu. Zvyšné časové miesto je ponechané pre iné zariadenia, ktoré sa môžu vyskytnúť na USB zbernici. Transakcie pre jednotlivé zariadenia sa potom ukladajú jedna vedľa druhej do tohto časového rámca.
Prvý bajt paketu SOF je už známy Sync. Druhý bajt je identifikačné číslo, ktoré je v prípade SOF paketu 0xA5. Ďalšia oblasť sa volá „Frame“. Je to 11 bitové počítadlo, ktoré počíta poradové číslo rámca od 0 do 2047. Potom pretečie a počíta opäť od nuly, lebo do 11 bitového čísla sa viac nezmestí:

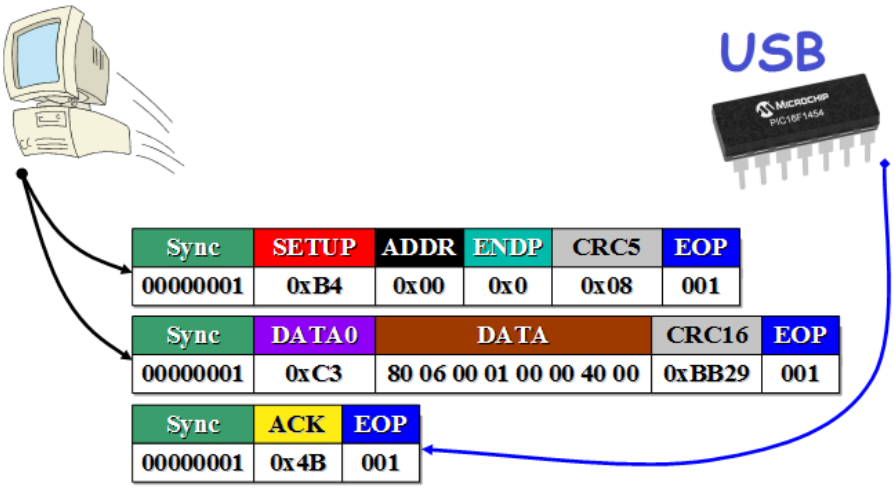
| Sync | SOF | Frame# | CRC5 | EOP |
|---|---|---|---|---|
| 00000001 | 0xA5 | 0x153 | 0x03 | 001 |

SOF paket v našom projekte nevyužívame. Používa sa pre synchronizáciu napr. pri izochrónnom prenose dat. Ale to nie je náš prípad.




Toľko o jednotlivých druhoch paketov a ich význame... Teraz pristúpim k popisu, ako skutočne prebieha komunikácia na USB zbernici.

**Dátové transakcie.**
Jedna dátová transakcia obsahuje viac paketových sekvencii. Paketové sekvencie sa skladajú z paketov. Pakety sa skladajú z bajtov. Bajty sa skladajú s bitov. A bity sú elektricky kódované tak, ako som popísal na začiatku. Niektoré pakety posiela počítač, niektoré posiela zariadenie. Každú paketovú sekvenciu začína vždy počítač. Zariadenie nemôže vysielať kedy sa mu zachce. Musí najskôr počkať na token paket od PC. Ukážeme si na reálnom príklade, ako prebieha pridelenie adresy zariadeniu po tom, čo sme ho zasunuli do USB portu a ako potom prebieha následná výmena dat. Už som spomínal, že najskôr sa komunikuje na adrese 0x00 - a do endpointu (pamäte) 0x00. Takže úplne na začiatku sa posiela:

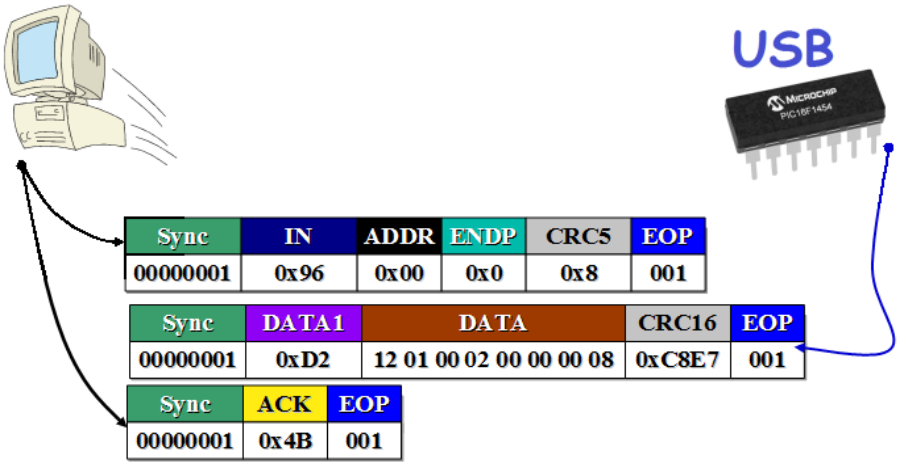



Počítač najprv poslal SETUP token paket s adresou 0x00 a označením endpointu 0x00, pre ktorý sú dáta určené. Druhý paket je dátový typu DATA0. V oblasti DATA tohoto paketu, je číslami popísaná požiadavka, čo počítač od zariadenia chce. Každé číslo má svoj význam. Čísel je 8 a sú zapísané v 16-kovej sústave. Tu je popis, čo tie čísla znamenajú (začínam zľava doprava):

*Setup data paket:*

- **80 –** Požadovaný smer prenosu je od zariadenia do PC,. (Tj. počítač žiada dáta.) V tomto bajte má každý bit svoj význam. Zapnutý je však len bit 7, čo po prepočte do 16-kovej sústavy dáva číslo 80 (binárne: 1000_0000) . Ostatné bity sú nulové. Jednotka sa podobá na písmeno I (Input) Preto to tak spravili.
- **06** – Žiadosť. Hodnota 06 znamená, že sa žiadajú dáta. Presnejšie povedané, celá jedna sada čísel. (tzv. „deskriptor“).
- **00** – Index deskriptora. (Tj. Poradové číslo ak ich je viac k dispozícii.)
- **01** – Požadovaná je sada čísel (deskriptor) o zariadení“ (tzv. device descriptor.)
- **00 00** - Tieto dva bajty sú spolu. V tomto prípade sa jedná o ID jazyka, ale v prípade iných požiadaviek môžu mať iný význam.
- **40 00** – Toto je požadované množstvo bajtov. Tak, ako v predošlom prípade, aj toto sú 2 bajty spolu, čiže hodnota 16 je bitová. **Lenže pozor!** Všetky čísla v oblasti DATA, sa ukladajú zprava doľava, čiže číslo treba zložiť dohromady opačne: **00 40** Keď si ho teraz na kalkulačke prepočítame do desiatkovej sústavy zistíme, že je požadovaných 64 bajtov. Táto hodnota je ale v tejto prvej paketovej sekvencii skôr symbolická, pretože sa ešte nenadviazalo spojenie so zariadením. Až neskôr bude označovať skutočný počet požadovaných bajtov.

Tak... To sme vyčerpali všetky čísla a ich význam v tejto prvej paketovej sekvencii z PC. Ešte chýba spomenúť posledný tretí, ACK paket. Poslalo ho zariadenie a ním potvrdzuje počítaču, že dáta prijalo. -Čo sme teda zistili? Počítač od nás žiada dátovú štruktúru o zariadení. Bol jej vymyslený aj názov: **Device deskriptor.** Táto štruktúra nie je nič iné, len ďalšia podobná sada čísel. Existuje viac druhov deskriptorov. Každý z nich je identifikovaný identifikačným číslom ID umiestneným na druhej pozícii deskriptora. Líšia sa i svojou dĺžkou a významom jednotlivých čísel. Konkrétne device deskriptor má dĺžku 18 bajtov. Nemôžeme ho však teraz poslať celý (tj. postupne po paketoch za sebou), lebo zariadeniu ešte nebola pridelená adresa. Najprv sa pošle len prvých 8 bajtov. Až po pridelení adresy sa bude dať poslať celý. Takže pokračujeme: Zariadenie odpovedá počítaču na jeho požiadavku a posiela prvých 8 bajtov Device deskriptora:




Ako vidíme, počítač najprv poslal paket IN. To znamená, že PC očakáva dáta od zariadenia (Input). Smer prenosu sa vždy určuje z pohľadu PC, lebo on je v celej komunikácii master. Očakáva teda Input - čiže príchod dat do PC. Zatiaľ stále na adrese 0x00 a z endpointu (pamäti zariadenia) číslo 0x00.

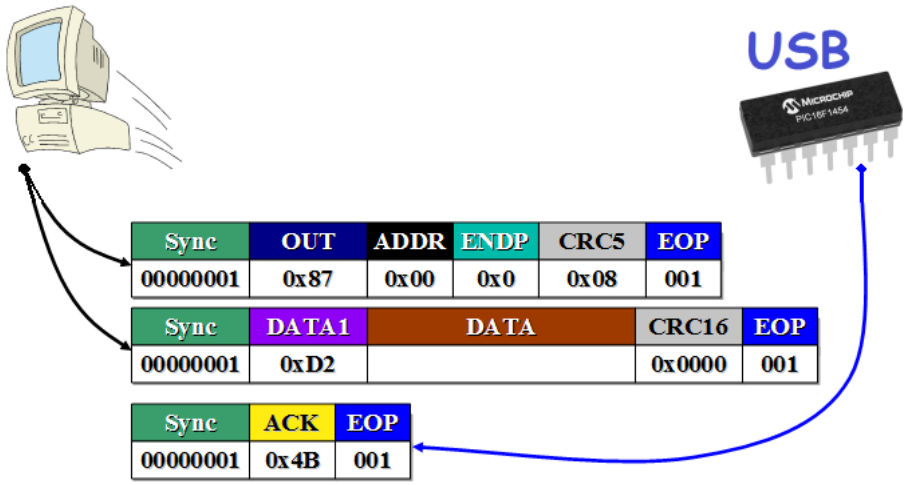
Druhý paket je dátový typu DATA1, lebo predošlý bol DATA0. (Už som spomínal, že oba typy sa pri prenose musia striedať. Je to tak spravené kvôli lepšiemu zabezpečeniu spoľahlivosti prenosu). Tu je význam jednotlivých čísel:

*Device descriptor (prvých 8 bajtov):*

- **12** – Dĺžka device deskriptora v bajtoch. Číslo je v16-kovej sústave, čiže dĺžka je 18 bajtov.
- **01** – Identifikačné číslo deskriptora. (01 = device deskriptor.)
- **00 02** – Verzia USB 02.00 (Dva bajty sú spolu. Ale musia sa otočiť, lebo číta sa zprava doľava. )
- **00** – Číslo triedy zariadenia. (Existujú rôzne triedy USB zariadení a každá má svoje číslo.)
- **00** – Číslo podtriedy zariadenia.
- **00** – Protokol zariadenia
- **08** – Maximálna dĺžka dátového paketu v bajtoch pre endpoint 0. (Jedná sa o dĺžku toho hnedého okienka dátového paketu.)

Deskriptor ale ešte nie je celý. -Dĺžka dátového paketu je defaultne nastavená na max. 8 bajtov, preto sa poslala len časť, tj. prvých 8 bajtov device deskripora. Treba poslať ešte ďalších 10, lebo deskriptor má celkovú dĺžku 18 bajtov. O tom nám hovorí hneď to prvé číslo 0x12 - čiže 18 v desiatkovej sústave. Ono oznamuje celkovú dĺžku deskriptora 18 bajtov. To sa ale stane až nabudúce. Najskôr musí počítač prideliť zariadeniu adresu. Potom sa deskriptor odvysiela celý. –Device deskriptor obsahuje i údaj o nastavení dĺžky paketu (viď posledné číslo). Môžeme si ju v prípade potreby prenastaviť. Štandardne je však nastavená na 08 bajtov. Keby sme chceli požiadať o viac, napíšeme napr. 09 a tým oznámime počítaču, že od teraz sa bude používať 9 bajtov v pakete. Aj zariadenie si ale potom bude musieť vyhradiť miesto v pamäti (endpointe 0) pre 9 bajtov.

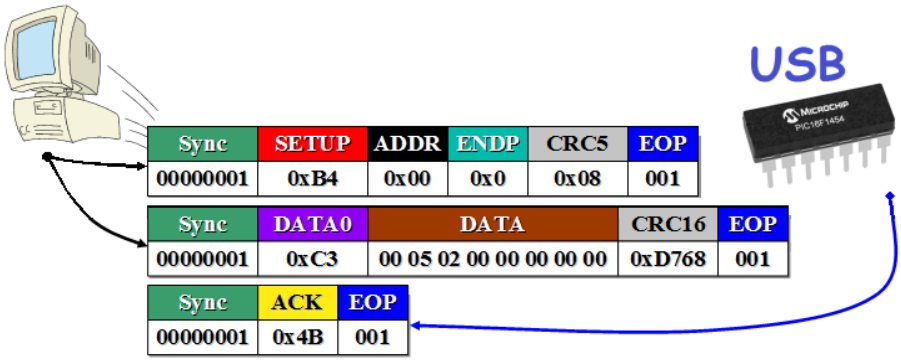
Na záver celej dátovej transakcie počítač ešte pošle potvrdenie, že všetko bolo prijaté správne. Stane sa tak poslaním prázdneho dátového paketu (tj. Dátového paketu s prázdnou DATA časťou):



Tým je prvá dátová transakcia skončená. -Aj táto posledná sekvencia paketov začína tým, že počítač vyzve zariadenie, aby čosi robilo. Zariadenie dovtedy len čakalo. PC mu najprv poslal OUT token paket. To znamená, že budú nasledovať dáta, doručované od počítača k zariadeniu. (Na adrese 0x00, pre endpoint č.0x00.) Druhý paket je už dátový typu DATA1. Tu vidíme porušenie predošlého striedania DATA0, DATA1, DATA0. (Naposledy bol totiž tiež posielaný dátový paket typu DATA1.) -Prečo? Lebo prázdny potvrdzujúci dátový paket **musí byť vždy typu DATA1** bez ohľadu na to, kde skončilo predošlé striedanie DATA paketov.

Teraz nasleduje pridelenie adresy, na ktoré sme čakali. Počítač najskôr vyšle signál RESET. Čo to je za signál? Je to vypnutie napätia na linke. Na linke spadne napätie do nuly po dobu viac ako 10 milisekúnd. Príjemca tento stav vyhodnotí ako signál RESET. (SE0)


Potom napätie opäť nabehne a počítač začne novú dátovú transakciu. Každá nová dátová transakcia začína vždy paketom SETUP. (A končí poslaním prázdneho DATA paketu.) Rovnako teda aj táto. (Stále ešte s adresou 0x00 a pre endpoint 0x00):

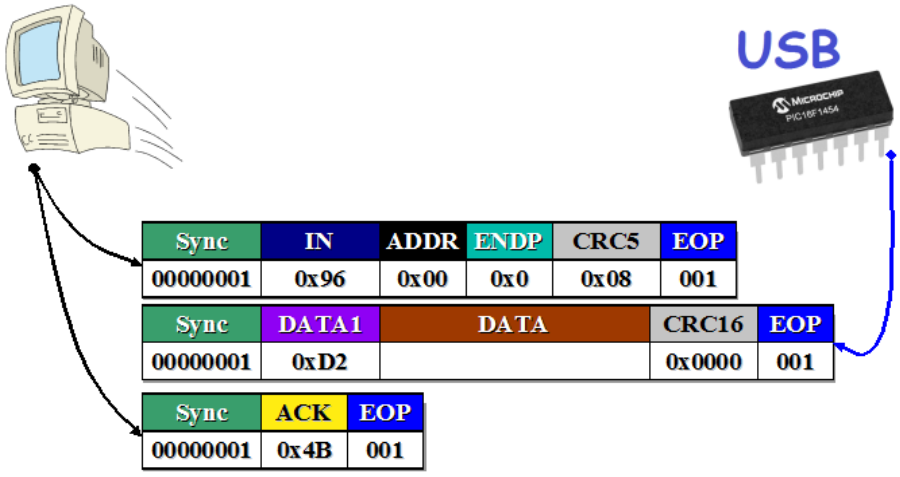



Druhý paket poslaný z PC je dátový typu DATA0. (Striedanie dátových paketov vždy začína typom DATA0 a vždy ho posiela počítač hneď za SETUP paketom. V ňom sa dozvieme, čo vlastne počítač od nás žiada.) V tomto prípade sú dôležité len prvé 3 čísla. Zvyšok sú nuly a sú bez významu. Preto ich ani neuvádzam. Tu je popis ich významu:

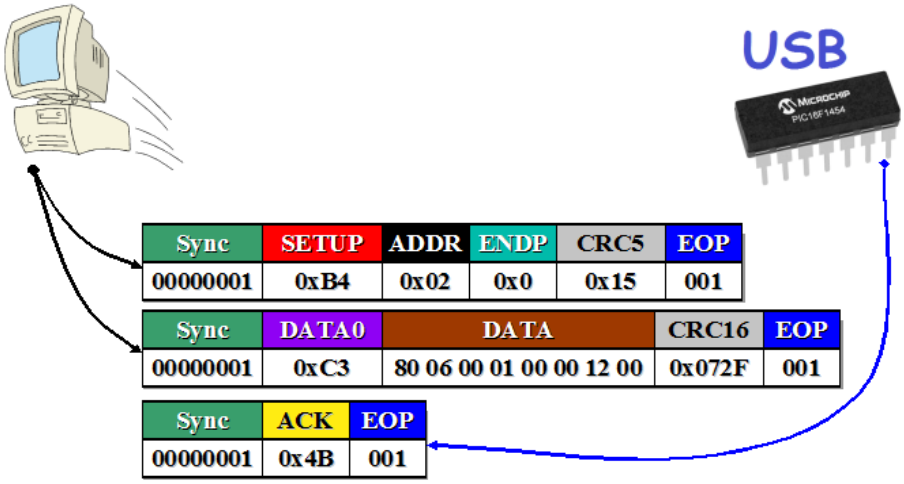
*Setup data paket:*

- **00** – Smer prenosu je od PC k zariadeniu.
- **05** – Jedná sa o nastavenie adresy.
- **02** – Adresa pridelená zariadeniu je 2

Tak a máme adresu. Počítač nášmu zariadeniu pridelil adresu 2. Od teraz teda budeme komunikovať na adrese 2. Všetky token pakety posielané z PC, budú s adresou 2. Pakety s inými adresami musíme ignorovať, pretože sú určené pre iné prípadné zariadenia pripojené k USB zbernici.
Už zostáva len potvrdiť prijatie tejto transakcie pomocou prázdneho dátového paketu (zatiaľ ešte stále na adrese 0):




Tým je dátová transakcia skončená. -Aj tu najprv PC poslal výzvu v podobe vstupného IN token paketu. IN znamená, že PC čaká na dáta od zariadenia. Zariadenie potom pošle prázdny dátový paket typu DATA1, a je potvrdené. Tj. koniec dátovej transakcie.
Hneď za tým PC začne novú dátovú transakciu. -Ako obvykle poslaním SETUP token paketu.:



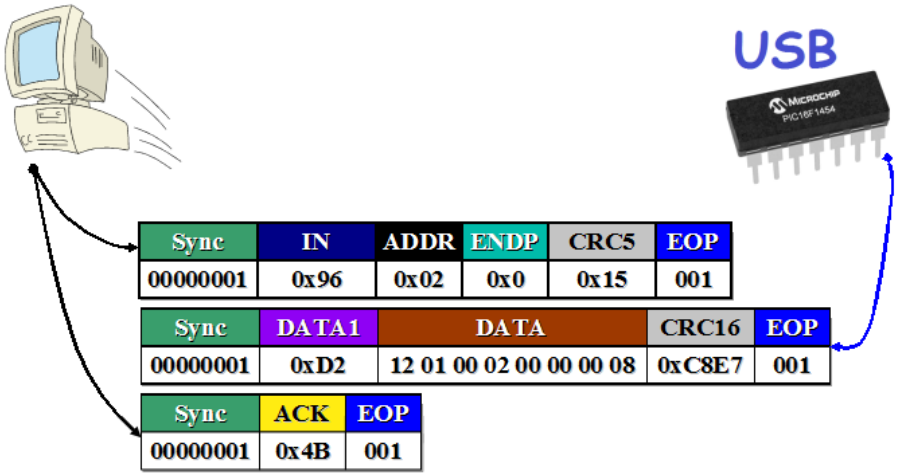

Tento raz ho však už poslal s adresou 2, ktorá bola pridelená zariadeniu. Všetky ďalšie token pakety odteraz adresované nášmu zariadeniu, budú už posielané s touto adresou. Pakety s inou adresou musí zariadenie ignorovať. -Po SETUP token pakete nasleduje ako obvykle dátový paket typu DATA0, z ktorého sa dozvieme, čo po nás počítač chce. Tu je význam jednotlivých čísel:

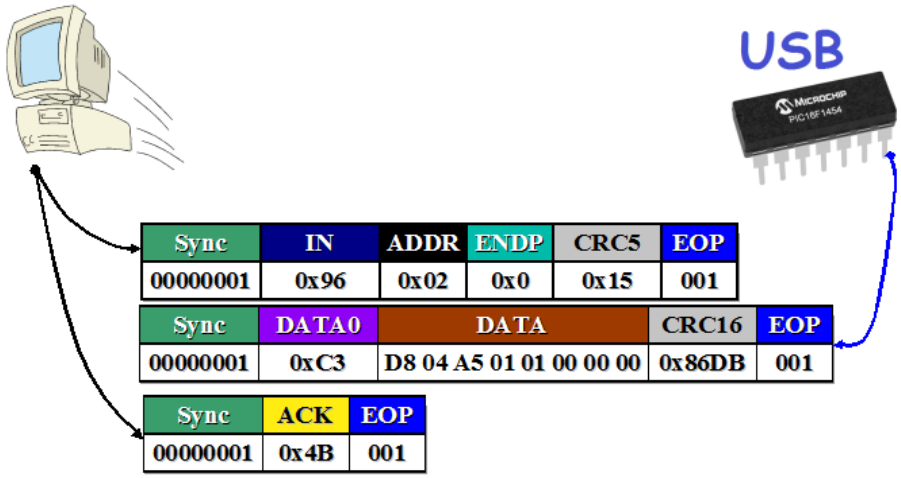
*Setup data paket:*

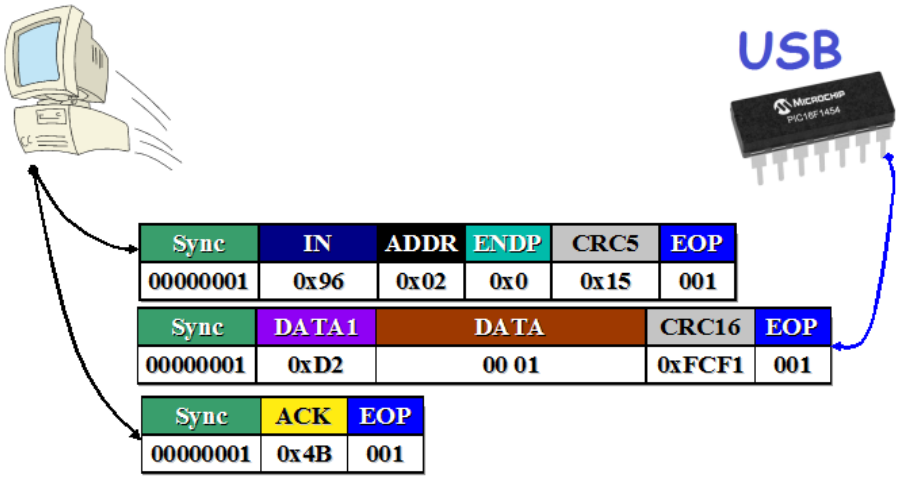
- **80 –** Požadovaný smer prenosu je od zariadenia do PC,. (Tj. počítač žiada dáta.) V tomto bajte má každý bit svoj význam. Zapnutý je však len bit 7, čo po prepočte do 16-kovej sústavy dáva číslo 80 (binárne: 1000_0000) . Ostatné bity sú nulové. Jednotka sa podobá na písmeno I (Input) Preto to tak spravili.
- **06** – Žiadosť. Hodnota 06 znamená, že sa žiadajú dáta. Presnejšie povedané, celá jedna sada čísel. (tzv. „deskriptor“).
- **00 –** Index deskriptora. (Tj. Poradové číslo ak ich je viac k dispozícii.)
- **01** – Požadovaný je device deskriptor.
- **00 00** - Tieto dva bajty sú spolu. V tomto prípade sa jedná o ID jazyka. (V prípade iných požiadaviek môžu mať iný význam.)
- **12 00** – Tieto dva bajty sú spolu. Znamenajú požadované množstvo bajtov. **Lenže pozor!** Všetky čísla v oblasti DATA, sa ukladajú zprava doľava, čiže číslo treba napísať opačne: **00 12** Keď to na kalkulačke prepočítame do desiatkovej sústavy zistíme, že je požadovaných je 18 bajtov.

Jedná sa o tú istú požiadavku čo na začiatku, akurát v tomto prípade už počítač žiada skutočné množstvo bajtov device deskriptora - tj. 18 bajtov. Prvých 8 sme už síce poslali na začiatku pred pridelením adresy, lenže aj tak ich musíme znovu zopakovať, pretože počítač od nás žiada 18 bajtov, čiže celý device deskriptor. Pošleme ich

teda všetky, rozdelené do troch paketov po 8 bajtoch. Takto to celé vyzerá:



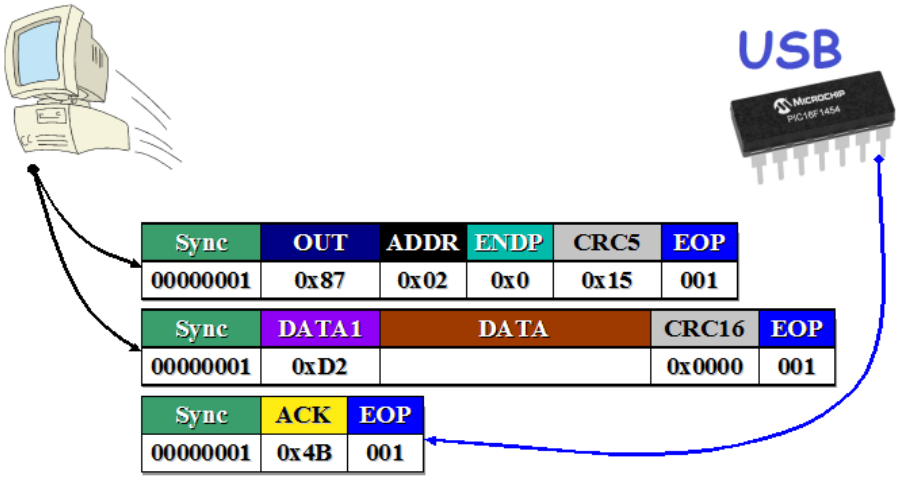

Odkedy bola zariadeniu pridelená adresa, komunikácia vyzerá už takto jednoducho. Zariadenie len počká na IN token paket z PC (s adresou 2) a hneď na to pošle svoj dátový paket. Prvý dátový paket obsahuje 8 bajtov, druhý 8 bajtov, tretí len 2 bajty, pretože 8+8+2=18 bajtov. Toľko bolo treba poslať. To je celý device deskriptor. **-Keď PC narazí na data paket ktorý nie je úplne obsadený, zistí tak, že je posledný. V našom prípade je to tretí data paket, ktorý obsahuje už len 2 bajty.** Existuje mnoho typov deskriptorov a ich dĺžky sa vzájomne líšia. To, aké deskriptory bude PC od zariadenia požadovať, záleží aj na type (triede) zariadenia. Niektoré typy deskriptorov sú spoločné všetkým triedam USB zariadení. Jeden z takých je práve tento device deskriptor, ktorý sme posielali. V ňom zariadenie mimo iné oznamuje i to, čo je to za typ (triedu). Keby sa

niekedy stalo, že nejaký deskriptor má presne toľko bajtov, že aj posledný paket je úplne obsadený, treba poslať ešte jeden prázdny dátový paket. Povedzme deskriptor, ktorý má dĺžku 16 bajtov sa celý vojde sa do dvoch dátových paketov, lebo 8+8=16. Posledný je teda tiež plný a PC sa nemá ako dozvedieť, že už je posledný. Preto vyšle ďalšiu žiadosť o DATA paket (čiže pošle ďalší IN token paket). Zariadenie mu musí odpovedať prázdnym paketom. Tým sa PC dozvie, že toto je už posledný DATA paket a ďalšiu požiadavku v podobe IN tokenu nepošle.
Teraz vypíšem význam všetkých čísel Device deskriptora, ktorý sme posielali:

*Device descriptor (celý):*

- **12** – Dĺžka device deskriptora v bajtoch. Číslo je v16-kovej sústave, čiže dĺžka je 18 bajtov.
- **01** – Identifikačné číslo deskriptora. (01 = Device deskriptor.)
- **00 02** – Verzia USB 02.00 (Dva bajty sú spolu. Ale musia sa otočiť, lebo čísla sa čítajú zprava doľava.)
- **00** – Číslo triedy zariadenia. (Existujú rôzne triedy USB zariadení a každá má svoje číslo.)
- **00** – Číslo podtriedy zariadenia.
- **00** – Protokol zariadenia
- **08** – Maximálna dĺžka dát dátového paketu pre endpoint 0 v bajtoch.
- **D8 04** – Dva bajty spolu. Treba ich otočiť: **04D8** Je to identifikátor výrobcu (Vendor ID)
- **A5 01** - Dva bajty spolu. Treba ich otočiť: **01A5** Je to identifikátor výrobku (Product ID)
- **01 00** - Dva bajty spolu. Treba ich otočiť: **00 01** Je to sériové číslo zariadenia
- **00** - Index deskriptora textového reťazca popisujúceho výrobcu
- **00** - Index deskriptora textového reťazca popisujúceho výrobok
- **00** - Index deskriptora textového reťazca popisujúceho sériové číslo zariadenia
- **01** – Počet konfigurácii.

Prvých 8 bajtov som už vysvetlil, keď sme ich posielali na začiatku. Ďalšie 2 bajty sú **D804,** čo po správnom otočení dáva číslo **04D8.** Je to identifikačné číslo výrobcu - tzv. VID. Ďalšie dva bajty sú **A501,** čo po správnom otočení dáva číslo **01A5.** Je to identifikačné číslo výrobku – tzv. PID. **Obe tieto čísla si môžeme určiť sami a sú to najdôležitejšie čísla, pomocou ktorých sa zariadenie identifikuje v operačnom systéme počítača a v konečnom dôsledku aj v počítačovom programe.** Pokiaľ by sme chceli mať dve USB zariadenia na jednom PC a s každým komunikovať samostatne, je dôležité, aby aspoň jedno z týchto čísel bolo rozdielne. (Najčastejšie sa mení VID.) -Ďalšie 3 čísla sú nulové. Jedná sa o indexy pre tzv. „textové deskriptory“. Do textových deskriptorov píšeme ASCII čísla textových znakov. Môžeme si v nich napísať logo firmy, názov výrobku, atď., čo sa zobrazí v podobe textu po zasunutí zariadenia do USB portu v operačnom systéme napr. v správcovi zariadení. Pokiaľ tieto indexy ponecháme nulové, textové deskriptory budú vypnuté a PC ich nebude od nás pýtať. V tomto projekte textové deskriptory nevyužívam, preto sú indexy nulové. -Posledné číslo Device deskriptora je počet konfigurácii. Niektoré USB zariadenia môžu obsahovať viac konfigurácii, pretože majú viac funkcii. (Napr. tlačiareň + skener.) My ale používame len jednu konfiguráciu, lebo textová USB vysielačka je jednoúčelové zariadenie. Preto má posledné číslo hodnotu 01.
Posledná paketová sekvencia je potvrdenie zo strany PC, že všetky dáta prijal. Urobí to ako tradične pomocou zaslania prázdneho dátového paketu. Ukončí sa tým zároveň celá dátová transakcia.



Tým je dátová transakcia skončená. -Znovu tu vidíme, že najprv prišiel token paket od PC. Konkrétne OUT token paket. To znamená, že PC má dáta pre zariadenie. Potom v druhom pakete prišli tie dáta. Prišiel ten potvrdzujúci prázdny paket bez dat, čo znamená koniec transakcie. Preto je typu DATA1. (Ukončovací prázdny dátový paket je vždy typu DATA1.)

*Ďalšie dátové deskriptory.*
Rovnako sú do počítača posielané aj ďalšie typy deskriptorov. Preto sa už nebudeme ďalej zdržovať podrobným rozkreslovaním paketov a paketových sekvencii, ale budeme si všímať len to hlavné: dáta a ich význam.
Ďalším dôležitým deskriptorom, spoločným pre všetky typy USB zariadení, je deskriptor konfigurácie („Configuration descriptor“). Počítač najskôr pošle v SETUP paketovej sekvencii takúto požiadavku:

80 06 00 02 00 00 FF 00

Popis významu čísel:

*Setup data paket:*

- **80 –** Požadovaný smer prenosu je od zariadenia do PC,. (Tj. počítač žiada dáta.) V tomto bajte má každý bit svoj význam. Zapnutý je však len bit 7, čo po prepočte do 16-kovej sústavy dáva číslo 80 (binárne: 1000_0000) . Ostatné bity sú nulové. Jednotka sa podobá na písmeno I (Input) Preto to tak spravili.
- **06** – Žiadosť. Hodnota 06 znamená, že sa žiadajú dáta. Presnejšie povedané, celá jedna sada čísel. (tzv. „deskriptor“).
- **00 –** Index deskriptora. (Tj. Poradové číslo ak ich je viac k dispozícii.)
- **02** – Požadovaný je deskriptor konfigurácie. (Configuration descriptor.)
- **00 00** - Tieto dva bajty sú spolu. V tomto prípade sa jedná o ID jazyka.
- **FF 00** – Tieto dva bajty sú spolu. Znamenajú požadované množstvo bajtov. Lenže pozor! Všetky čísla v oblasti DATA, sa ukladajú zprava doľava, čiže číslo treba napísať opačne: **00 FF** Keď to na kalkulačke prepočítame do desiatkovej sústavy zistíme, že je požadovaných až 256 bajtov. Deskriptor konfigurácie však toľko nemá, čiže požiadavka je len symbolická. Dôvodom je, že spolu s konfiguračným deskriptorom sa budú posielať ešte ďalšie deskriptory, ktorých počet a typ závisí na nastaveniach v samotnom konfiguračnom deskriptore. Preto počítač nevie dopredu, koľko bajtov sa bude posielať a preto napísal symbolicky len hodnotu FF (255).

Zariadenie odpovie počítaču poslaním toho *Configuration deskriptora*. V prípade našej textovej USB vysielačky obsahuje tieto čísla:

09 02 29 00 01 01 00 A0 32

Ich význam je tento:

*Configuration descriptor:*

- **09** – Veľkosť deskriptora v bajtoch. (9 bajtov)
- **02** – Typ deskriptora. (02 = konfiguračný deskriptor.)
- **29 00** – Čiže 00 29. Je to celková dĺžka všetkých deskriptorov pre túto konfiguráciu. (41 bajtov.)
- **01** – Počet rozhraní. (My máme len jedno, lebo USB vysielačka nie je multifunkčné zariadenie.)
- **01** – Číslo (index) tejto konfigurácie.
- **00** – Index textového reťazca popisujúceho túto konfiguráciu. (Nechceme žiaden, takže dáme 00.)
- **A0** – V tomto bajte majú význam jednotlivé bity. Je nimi nastavené, že zariadenie používa napájanie z USB portu a podporuje vzdialené prebudenie z režimu spánku.
- **32** – Maximálna spotreba USB zariadenia v miliampéroch. Jedna jednotka je 2mA. Najprv si teda prepočítame číslo do desiatkovej sústavy. Získame tak 50. Potom vynásobíme 50 x 2 = 100mA. Čiže maximálny odber prúdu nášho zariadenia z USB zbernice, môže byť 100mA. (Môžeme požiadať aj o viac, ale nám stačí 100mA.)

V tomto deskriptore je zadefinované, že budeme používať iba jedno rozhranie. Multifunkčné zariadenia, ako tlačiareň+skener, majú viac rozhraní. Pre každé z nich musí byť k dispozícii samostatný konfiguračný deskriptor. Zvlášť pre skener, zvlášť pre tlačiareň, atď.. Jednotlivé konfiguračné deskriptory sú potom očíslované 1, 2, 3, atď..., takže keď PC dáva požiadavku pre niektorý z nich, uvedie aj číslo indexu, ktorý

konkrétne práve požaduje. (To je ten tretí bajt v SETUP data pakete posielanom z PC, nazvaný „Index deskriptora“.) ...Naša konfigurácia je ale iba jedna a má číslo (index) 01. To stačí, viac nepotrebujeme. V konfiguračnom deskriptore je ešte uvedené, že celková dĺžka všetkých deskriptorov tejto konfigurácie bude 41 bajtov. Konfiguračný deskriptor je totiž prvý zo série ďalších deskriptorov, ktoré budú posielané hneď za ním. Ďalší ktorý nasleduje, je deskriptor rozhrania (tzv. *Interface descriptor*). Posiela sa bez prerušenia transakcie hneď za konfiguračným deskriptorom (tj. PC naň neposiela požiadavku, ale len ďalej pokračuje zasielaním IN token paketov). V prípade našej textovej USB vysielačky obsahuje tieto čísla:

09 04 00 00 02 03 00 00 00

Ich význam je tento:

*Interface descriptor:*

- **09** – Veľkosť deskriptora v bajtoch. (9 bajtov)
- **04** – Typ deskriptora. (04 = interface deskriptor.)
- **00** – Číslo rozhrania. (Máme len jedno.)
- **00** – Počet alternatívnych nastavení. (Nemáme žiadne ďalšie alternatívy.)
- **02** – Počet ďalších endpointov (okrem endpointu 0, ktorý musí vždy byť).
- **03** – Kód triedy (03 = Trieda HID)
- **00** – Kód podtriedy. (Nepoužívame žiadnu.)
- **00** – Kód protokolu. (Nepoužívame žiadny.)
- **00** - Index deskriptora textového reťazca popisujúceho rozhranie. (Nepoužívame žiadny.)

Tento deskriptor obsahuje ďalšie dôležité nastavenia. Definuje počet endpointov (tj. pamäťových priestorov v zariadení) – iných, mimo endpointu 0, ktoré sa budú používať na prenos užívateľských dát. Sú zadefinované dva. Endpoint 0 používame len na prenos konfiguračných a riadiacich dát (to sú tie, ktoré práve teraz od začiatku preberáme). Ten musí mať každé USB zariadenie, preto sa nepočíta. Po prebehnutí všetkých potrebných nastavení (pomocou všetkých deskriptorov), bude prebiehať výmena užívateľských dát medzi našou aplikáciou v počítači a zariadením. Na túto výmenu sú určené práve tieto 2 nové endpointy. Obidva budú mať číslo 1, akurát jeden bude vstupný a druhý výstupný. Počítač ukladá dátové transakcie pre zariadenia do rámcov trvajúcich 1 milisekundu (ako som už písal). V jednom rámci obslúži vždy iba jeden endpoint, jednou dátovou transakciou. Keďže však máme endpointy dva, PC umiestni do jedného rámca až dve transakcie – tj. pre každý endpoint zvlášť. (V našom prípade to vyzerá tak, že pošle IN token paket spolu s OUT a dá ich vedľa seba do jedného rámca.) Komunikácia je tým rýchlejšia. Čím viac endpointov, tým je rýchlejšia. Pre naše účely však úplne stačí aj takáto rýchlosť, preto si vystačíme s dvomi komunikačnými endpointmi.
Ďalší veľmi dôležitý údaj v *Interface deskriptore,* je bajt označujúci kód triedy zariadenia. Existujú rôzne triedy USB zariadení. Napríklad trieda *Mass Storage Class (MSC),* kde patria externé harddisky, USB flash disky, čítačky pamäťových kariet, atď.. Inou triedou je *Communications Device Class (CDC),* kde patria rôzne druhy prevodníkov – napr.: USB-Ethernet, USB-Uart, USB modemy, atď.. Ďalšia trieda je *Vendor Specific Class (VSP),* čiže trieda špecifikovaná dodávateľom. -**V našom projekte používame triedu *Human Interface Device (HID).*** Do nej patria myši, klávesnice, herné zariadenia, čiže zariadenia s rozhraním medzi človekom a počítačom. Táto trieda má obrovskú výhodu v tom, že nemusíme písať zložitý USB ovládač pre PC, lebo je už napísaný tvorcami operačného systému.

Malá odbočka: Existuje niekoľko typov prenosu dát na USB zbernici.

*Riadiacie prenosy. (Controll transfer)*
To je všetko to, čo sme doteraz preberali. Všetky tie transakcie kde PC posiela žiadosti o deskriptory a zariadenie ich posiela. (V nich oznamuje informácie o sebe a nastavuje si vlastnosti USB prenosu pre seba.) Sú dobre zabezpečené voči chybám (ako vidno z toho, čo som doteraz popísal).

*Hromadný prenos (Bulk transfer)*
Tento typ je vhodný pre prenos veľkých objemov dát, pri ktorých nezáleží na presnom časovaní. Má nižšiu prioritu ako ostatné typy prenosov, využíva voľnú prenosovú kapacitu zbernice. Typické použitie je posielanie dát tlačiarni alebo čítanie a zápis na flash disk, externý harddisk a pod..

*Prerušovací prenos (Interrupt transfer)*
**Tento typ prenosu používame my v našom zariadení.** Typicky sa totiž používa v HID zariadeniach. Jedná sa o prenos dat v pravidelných časových intervaloch. Inými slovami PC pravidelne posiela IN a OUT token pakety a zariadenie buď odpovedá dátovým paketom a/alebo prijme dáta od PC. Prípadne odpovie NAK paketom – tj. nemá žiadne dáta na odoslanie. NAK paket sme už preberali. Vyzerá takto:

| Sync | NAK | EOP |
|---|---|---|
| 00000001 | 0x5A | 001 |

Tieto pravidelné časové intervaly sa dajú samozrejme nastaviť v *Endpoint deskriptore*, ku ktorému sme sa ešte nedostali. Interval sa dá nastaviť najmenej na 1 milisekundu, pretože práve toľko trvá jeden prenosový rámec (o ktorom už bola reč). -Ďalej sa dá nastaviť aj dĺžka dátového paketu v bajtoch. V prípade prenosovej rýchlosti Full speed (12 Mbit/s) ktorú používame tomto projekte, môže byť maximálne 64 bajtov. V prípade Low speed (1.5 Mbit/s) maximálne 8 bajtov a v prípade High speed až 1024 bajtov. Náš PIC mikrokontrolér 16F1454 ktorý používame v textovej vysielačke, podporuje maximálnu rýchlosť High speed, čiže najviac čo sa dá nastaviť, je 64 bajtov dlhý paket. A na toľko som ho aj nastavil. Nastavuje sa to v deskriptoroch, ktoré sme zatiaľ ešte nepreberali, ale dostaneme sa k nim.

*Izochrónny prenos (Isochronous transfer)*
Tento typ prenosu je prúdový a prebieha v reálnom čase. Nie je zabezpečený proti chybám, pretože občasné chyby sú tolerované. Obvykle sa používa pri prenose hudby alebo videa v reálnom čase.

Koniec odbočky....

Poďme späť k deskriptorom konfigurácie. Ďalší deskriptor ktorý tesne nadväzuje (tj. PC sa naň nedopytuje ale rovno posiela IN token paket), je HID deskriptor. To preto, lebo sme si v *Interface descriptore* zadefinovali, že naše zariadenie je triedy HID (kód 03). Preto musí nasledovať HID deskriptor. Keby sme nastavili inú triedu, musel by nasledovať iný typ deskriptora, charakteristický pre tú triedu. Nastavili sme si teda triedu HID a budeme posielať HID deskriptor. Z našej textovej vysielačky preto potečú do PC teraz tieto čísla:

09 21 11 01 00 01 22 2F 00

Ich význam je tento:

*HID descriptor:*
- **09** – Veľkosť deskriptora v bajtoch. (9 bajtov)
- **21** – Typ deskriptora. (21 = HID deskriptor.)
- **11 01** – (Dva bajty spolu.) Verzia HID špecifikácie je 01.11 (Čísla treba ako obvykle otočiť.)
- **00** – Kód krajiny (Nepodporované.)
- **01** - Počet deskriptorov pre triedu HID, ktoré majú nasledovať.
- **22** – Typ report deskriptora. (Ktorý bude nasledovať.)
- **2F 00** - (Dva bajty spolu) Dĺžka report deskriptora bude 002F, čiže desiatkovo 47 bajtov

To najdôležitejšie, čo je v tomto deskriptore definované je typ a dĺžka tzv. *Report deskriptora*, ktorý bude posielaný ako úplne posledný a je zároveň najdlhší. V ňom budú podrobne definované všetky parametre, týkajúce sa prenosu užívateľských dát, formát a podobne. Existuje viac typov report deskriptorov. Iný používa myš, iný klávesnica, atď... Naše zariadenie je registrované ako **Generic HID,** čiže na všeobecné použitie. Tomu zodpovedá aj typ report deskriptora (ktorého kód je 22).
Ďalšie dva deskriptory ktoré nasledujú, sa volajú Endpoint deskriptory. Sú dva preto, lebo máme nastavené dva komunikačné endpointy. Definujú sa v nich vlastnosti a parametre endpointov. Pre každý endpoint jeden deskriptor zvlášť. Z USB vysielačky do PC teraz teda nasledujú tieto čísla:

07 05 81 03 40 00 01

Ich význam je tento:

*Endpoint descriptor:*

- **07** – Veľkosť deskriptora v bajtoch. (7 bajtov)
- **05** - Typ deskriptora. (05 = Endpoint deskriptor.)
- **81** – Číslo endpointu je 1 a smer prenosu dat endpointu je Input (...lebo siedmy bit bajtu je 1)
- **03** – Typ prenosu endpointu bude prerušovací (Interrupt transfer)
- **40 00** – Dĺžka dátového paketu tohto endpointu bude **00 40**, čiže v desiatkovej sústave 64 bajtov
- **01** – Interval prenosu bude 1 milisekunda.

Tým sú nastavené všetky potrebné parametre endpointu. Endpoint je nastavený ako vstupný (input). (Samozrejme vstupný z pohľadu počítača, čiže dáta budú z tohto endpointu do PC odchádzať.) Typ prenosu je *prerušovací prenos*, čiže PC bude v pravidelných intervaloch posielať tomuto endpointu IN token paket. Ak zariadenie má niečo na odoslanie odošle, ak nie, odpovie NAK paketom. (Nemám nič.) Tento časový interval je nastavený na jednu milisekundu, čiže najkratší možný čas. (Tým sme dosiahli najvyššiu prenosová rýchlosť.) Najkratší je preto, lebo PC (ako som už písal) posiela dáta v 1 milisekundových rámcoch. Do tohto rámca sa musia zmestiť dáta pre všetky USB zariadenia aktuálne pripojené k PC. Čím viac zariadení pripojíme, tým je rámec viac obsadený.
Ďalší deskriptor je tiež *Endpoint descriptor,* ale pre ten druhý endpoint. USB vysielačka teda pošle do PC ďalej tieto čísla:

07 05 01 03 40 00 01

Ich význam je tento:

*Endpoint descriptor:*

- **07** – Veľkosť deskriptora v bajtoch. (7 bajtov)
- **05** - Typ deskriptora. (05 = Endpoint deskriptor.)
- **01** – Číslo endpointu je 1 + smer prenosu dat endpointu je Output (...lebo siedmy bit bajtu je 0)
- **03** – Typ prenosu endpointu bude prerušovací (Interrupt transfer)
- **40 00** – Dĺžka dátového paketu tohto endpointu bude **00 40**, čiže v desiatkovej sústave 64 bajtov
- **01** – Interval prenosu bude 1 milisekunda.

Jediná zmena oproti predošlému endpoint deskriptoru je smer prenosu dat. V tomto prípade Output, čiže výstupný. Výstupný z pohľadu počítača, takže dáta budú do tohto endpointu posielané z PC. PC pošle OUT token paket a ak má pre zariadenie nejaké dáta, pošle hneď za ním dátový paket dlhý 64 bajtov. (Lebo taký dlhý sme si ho nastavili v *Endpoint deskriptore*.) -Všetko ostatné platí to isté, čo o predošlom endpoint deskriptore.
Takže toto bol posledný deskriptor tejto konfiguračnej sady. Zopakujme si ich všetky v poradí, v akom boli postupne posielané:

1. *Configuration descriptor* (9 bajtov)
2. *Interface descriptor* (9 bajtov)
3. *HID descriptor* (9 bajtov)
4. *Endpoint descriptor (IN)* (7 bajtov)
5. *Endpoint descriptor (OUT)* (7 bajtov)

Všetky sa posielali naraz spolu za sebou bez prerušenia dátovej transakcie. Jediné čo počítač celý čas posielal, boli IN token pakety. Počet bajtov všetkých deskriptorov tejto série dohromady bol zadaný už na začiatku v *Configuration deskriptore*. Je tam uvedených 41 bajtov. A skutočne ich toľko aj bolo. Môžeme si to overiť, keď si ich spočítame: 9+9+9+7+7=41.
Pri popise HID deskriptora som spomínal, že je v ňom je zadefinovaný ešte jeden deskriptor. Konkrétne *Report deskriptor.* Ten bude teraz nasledovať a je posledný. O ten si však počítač už štandardne požiada ako býva zvykom pomocou SETUP token paketovej sekvencie. Pošle teda zariadeniu túto požiadavku:

80 06 00 22 00 00 2F 00

Význam týchto čísel som už niekoľko krát popísal, takže spomeniem len tie hlavné:

*Setup data paket:*

- **22** - Požadovaný je Report deskriptor.
- **2F 00** – Očakávaná dĺžka report deskriptora je 2F čiže desiatkovo 47 bajtov

Zariadenie teda odpovie počítaču zaslaním *Report deskriptora*. Do PC pôjdu teda tieto čísla:

06 A0 FF 09 01 A1 01 09 01 15 00 26 FF 00 75 08 95 40 81 02 09 01 15 00 26 FF 00 75 08 95 40 91 02 09 02 15 00 26 FF 00 75 08 95 40 B1 02 C0

Významovo sú čísla radené po dvojiciach až po trojiciach a tu je ich popis:

- **06 A0 FF** – Užívateľom definovaná stránka.
- **09 01** – Užívateľská stránka č. 1
- **A1 01** – Začiatok kolekcie (aplikácie)

*Vstupné správy:*

- **09 01** – Identifikačné číslo správy.
- **15 00** – Logické minimum = 0
- **26 FF 00** – Logické maximum = 255
- **75 08** – Veľkosť čísel správy (8 bitov)
- **95 40** – Počet čísel správy (64 čísel)
- **81 02** - Vstup sú: údaje, premenné, absolútne hodnoty.

*Výstupné správy:*

- **09 01** – Identifikačné číslo správy.
- **15 00** – Logické minimum = 0
- **26 FF 00** – Logické maximum = 255
- **75 08** – Veľkosť čísel správy (8 bitov)
- **95 40** – Počet čísel správy (64 čísel)
- **91 02** - Vstup sú: údaje, premenné, absolútne hodnoty.

*Funkčné správy:*

- **09 02** – Identifikačné číslo správy.
- **15 00** – Logické minimum = 0
- **26 FF 00** – Logické maximum = 255
- **75 08** – Veľkosť čísel správy (8 bitov)
- **95 40** – Počet čísel správy (64 čísel)
- **B1 02** – Vstup sú: údaje, premenné, absolútne hodnoty.
- **C0** – Koniec kolekcie.

Prvé číslo dvojice / trojice definuje, čo je to za parameter. Druhé číslo je hodnota toho parametra. -V prípade trojice sú hodnota druhé aj tretie číslo, čiže dohromady16 bitová hodnota. Akurát ich treba zložiť naopak. Napr. číslo FF 00 je v skutočnosti 00 FF, čiže desiatkovo 255.

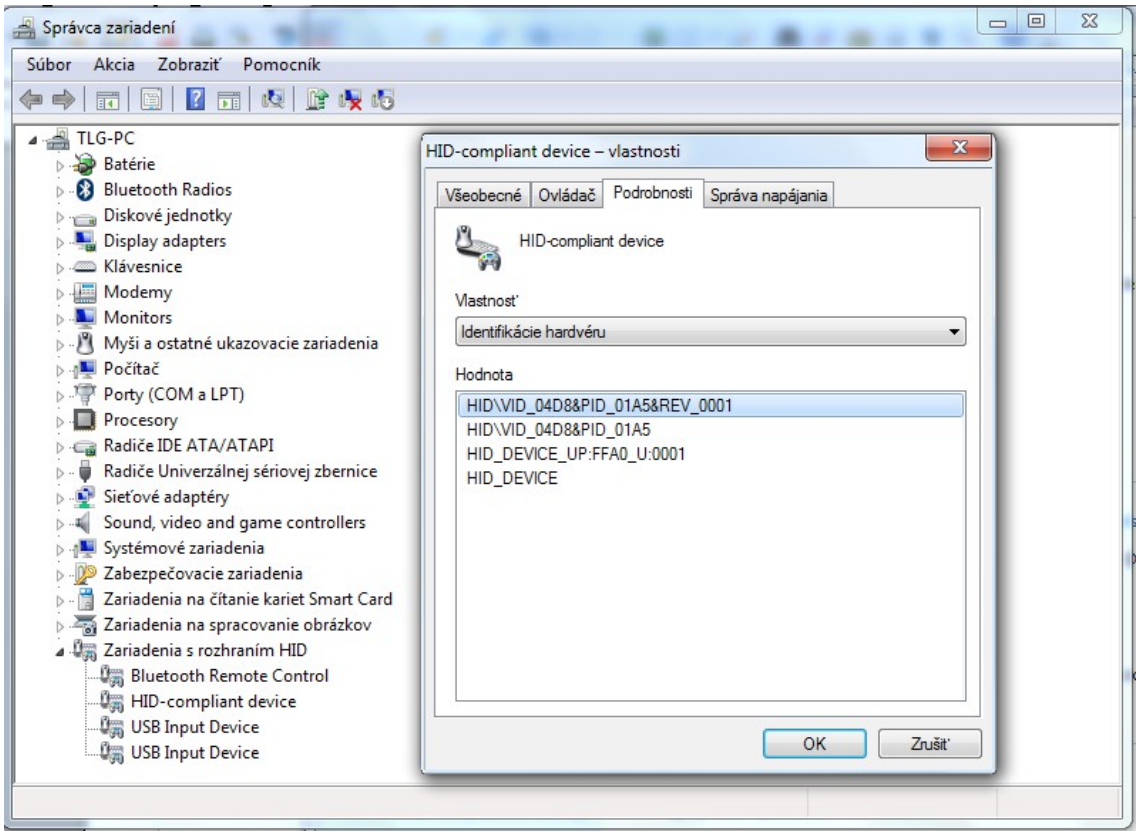
Vidíme, že report deskriptor sa skladá z akýchsi pomyselných stránok. Tento náš má len jednu. Do nej je vnorená kolekcia nastavení pre 3 rôzne typy správ, s ktorými bude zariadenie pracovať po tom, čo bude nakonfigurované: Vstupné správy, výstupné správy a nastavovacie správy. Vstupné a výstupné, to je jasné. O tom som už písal: PC bude v pravidelných intervaloch na endpoint č.1 posielať IN a OUT token pakety každú milisekundu. Ak má zariadenie niečo na odoslanie odošle 64 bajtov. Ak nie, odpovie NAK paketom. (Nemám nič.) Ak PC má nejaké dáta pre zariadenie, pošle za OUT tokenom dátový paket dlhý 64 bajtov. (Toľko sme si totiž nastavili v Endpoint deskriptore a teraz už aj v report deskriptore). Tretí typ správ, ktoré sú definované v report deskriptore, sú „Funkčné správy“ a slúžia na prenos nastavení. Umožňujú počas prebiehajúcej komunikácie meniť nastavenia niektorých vlastností prenosu. Všetky 3 typy správ majú rovnaké možnosti nastavenia. To hlavné čo sa nastavuje, je logický rozsah hodnôt, ktoré môžu jednotlivé čísla naberať. (Tu je to nastavené od 0 do 255.) Schválne píšem čísla a nie bajty, pretože ako vidíme ďalej, je možné nastaviť si aj počet bitov čísel. Môžeme si nastaviť aj viac alebo menej ako 8 bitov. Potom to už nie sú bajty. No a nakoniec sa dá nastaviť aj počet týchto čísel v jednom dátovom pakete. Hodnota sa musí zhodovať s tým, čo bolo už definované v Endpoint deskriptoroch. Tam sa nastavoval počet bajtov v jednom dátovom pakete. Máme to nastavené na 64 bajtov. Čiže 64x8=512 bitov. Tento počet sa musí presne zhodovať s tým, čo je nastavené v

report deskriptore. (Tj. Počet bitov čísla krát počet čísel.)

To je všetko. Keď deskriptory, ktoré sme od začiatku doteraz preberali počítač obdrží, bude poznať všetky potrebné informácie o našom zariadení a môže tak spustiť prenos užívateľských dát. Celému tomuto procesu sa hovorí ***Enumerácia***. V operačnom systéme počítač priradí textovej USB vysielačke ProductID a VendorID ktoré sme si definovali v Device deskriptore. Sú to čísla:

**Vendor ID = 04D8**
**Product ID = 01A5**

Tieto identifikačné čísla sa objavia aj v správcovi zariadení. USB vysielačka bude pod položkou „*Zariadenia s rozhraním HID*“. Po rozkliknutí ich tam nájdeme:




Na základe nich bude so zariadením komunikovať aj náš počítačový program. Rozlíši textovú USB vysielačku spomedzi všetkých zariadení a smerovať len ku nej / čítať od nej dáta. -Po nakonfigurovaní (enumerácii) začne okamžite prebiehať IN/OUT komunikácia. Každú milisekundu bude PC posielať IN a OUT token pakety. V prípade, že zariadenie má v input endpointe k dispozícii dáta na odoslanie, odpovie na IN token paket dátovým paketom dĺžky 64 bajtov. Ak počítač bude mať nejaké dáta pre zariadenie, umiestni za OUT token paket dátový paket s dĺžkou 64 bajtov. Zariadenie si jeho obsah uloží do output endpointu. V oboch prípadoch je dĺžka 64 bajtov, lebo takú sme si nastavili. Čísla v týchto paketoch môžu byť ľubovoľné, aké chceme. To bolo našim cieľom.
Je nutné podotknúť ešte jednu vec: **Počítač môže kedykoľvek požiadať o zopakovanie niektorého z deskriptorov, ktoré sme preberali.** Napr. niekto zasunie do iného USB portu ďalšie zariadenie a na zbernici môže vzniknúť neistota. PC potom požiada naše zariadenie o zopakovanie príslušných deskriptorov alebo len jedného deskriptora. Uvedená situácia môže nastať aj z iných dôvodov počas behu operačného systému. Preto musí byť schopné reagovať na to prerušenním. Detekovať požiadavku od PC a na základe nej odoslať príslušný deskriptor(y). To je vecou algoritmu a podmienok programu v mikrokontroléri PIC16F1454. Prebiehajúcej komunikácie medzi endpointom 1 a PC programom sa to nedotkne, pretože deskriptorové požiadavky sú adresované endpointu 0 a ukladajú sa spolu do jedného rámca vedľa dát endpointov 1in a 1out.
Je tu však jedno špecifikum: *HID descriptor,* ktorý bol súčasťou balíčka konfiguračnej sady deskriptorov (to sú tie, ktoré sa posielali sa naraz spolu), môže byť požadovaný samostatne. Preto musí byť v programe PIC k dispozícii k samostatnému odoslaniu. PC vtedy posiela požiadavku:

80 06 00 21 00 00 09 00

Význam týchto čísel som už niekoľko krát popísal, preto spomeniem len tie hlavné:

*Setup data paket:*

- **21** - Požadovaný je HID deskriptor.
- **09 00** – Očakávaná dĺžka report deskriptora je 9 bajtov

(Samozrejme v prípade, že počítač požiada znovu o *Configuration descriptor,* zariadenie musí zas poslať celú sadu vrátane HID deskriptora ktorej je súčasťou. Tú som už popisoval hore.)

*Ďalšie požiadavky z PC:*
Okrem doteraz preberaných požiadaviek o deskriptory, PC posiela ešte niektoré ďalšie požiadavky, ktoré mikrokontrolér musí spracovať. Všetky doteraz preberané mali na druhej pozícii číselnej rady zľava číslo 06. Toto číslo je kód požiadavky, ktorá sa oficiálne volá **GET_DESCRIPTOR,** čiže „*budem od teba chcieť deskriptor*“. (Štvrté číslo potom definuje, ktorý konkrétne (napr. 21 = HID deskriptor).) Iným typom požiadavky je **GET_CONFIGURATION** (kód 08). Požiadavka z PC potom vyzerá takto:

80 08 00 00 00 00 01 00

Popis významu čísel:

*Setup data paket:*

- **80 –** To už som viac krát popisoval. Číslo si prepíšeme do dvojkovej sústavy, lebo v ňom má každý bit svoj význam: 1000_0000. Bit 7 je nastavený do 1, čo znamená „Input“ (z pohľadu PC samozrejme), čiže „počítač bude žiadať dáta“. (Písmeno I sa podobá na číslo 1, preto to tak spravili, že 1 = Input).
- **08** – Počítač žiada číslo aktuálnej konfigurácie.
- **00** – Nemá v tomto prípade význam.
- **00** – Nemá v tomto prípade význam.
- **00 00** - Nemá v tomto prípade význam. (Sú to dve čísla spolu, čiže 16-bitová hodnota.)
- **01 00** – Žiada sa 1 bajt. (Čísla sa musia ako obvykle otočiť, čiže **00 01**.)

Čiže počítač žiada od zariadenia jeden jediný bajt a ním je číslo našej konfigurácie. Toto číslo bolo definované už v *Configuration deskriptore*. Bol tam uvedený počet všetkých možných konfigurácii nášho zariadenia = 1. (Keby bolo multifunkčné, ako napr. skener+tlačiareň, obsahovalo by viac konfigurácii (napr. 2) a pre každú z nich by potom musela byť zvlášť definovaná konfiguračná skupina (napr. dve).) -V konfiguračnom deskriptore bolo ďalej uvedené číslo konfigurácie, ktorej sa týka tá konfiguračná skupina. Pretože máme v zariadení iba jednu, jej číslo bolo 1. Toto číslo teda pošleme počítaču ako odpoveď na jeho požiadavku:

01

Keby sme poslali 00 znamenalo by to, že zariadenie ešte nie je nakonfigurované a nemá pridelenú adresu. PC by sa potom podľa toho zariadil a začal posielať požiadavky na deskriptory úplne od začiatku vrátane pridelenia adresy.
Posledným typom požiadavky ktorá môže prísť z PC pre naše zariadenie a ktorú je nutné spracovať je **SET_CONFIGURATION** (kód 09). Požiadavka z PC vyzerá takto:

00 09 00 01 00 00 00 00

Popis významu čísel:

*Setup data paket:*

- **00 –** To už som viac krát popisoval. Číslo si prepíšeme do dvojkovej sústavy, lebo v ňom má každý bit svoj význam: 0000_0000. Bit 7 je nastavený do 0, čo znamená „Output“ (z pohľadu PC samozrejme), čiže „počítač nám posiela dáta“. (Písmeno O sa podobá na číslo 0, preto to tak spravili, že 0 = Output).
- **08** – Počítač posiela číslo konfigurácie, ktorú si má zariadenie aktuálne nastaviť. (SET)
- **00** – Nemá v tomto prípade význam.
- **01** – Zariadenie si musí nastaviť konfiguráciu číslo 1.

- **00 00** - Nemá v tomto prípade význam. (Sú to dve čísla spolu, čiže 16-bitová hodnota.)
- **00 00** – Počet bajtov. Avšak nemá v tomto prípade význam.

Počítač od nás žiada, aby sme si v zariadení pripravili a nastavili konfiguráciu číslo č.1. Keby to bol skener+tlačiareň, PC by poslal číslo 01 keby sa skenovalo a číslo 02, keby sa tlačilo. Multifunkčné zariadenie podľa toho rozpozná, že dáta ktoré práve dostáva/odosiela sú pre tlačiareň alebo pre skener. To ale nie je náš prípad. PC nám poslal 01, čiže: „nastav si konfiguráciu číslo 1“. **Od tej chvíle začnú chodiť na endpoint 1 cyklicky každú milisekundu IN a OUT tokeny na prenos užívateľských dáta, čo je cieľom zariadenia.** -Ak by sa stalo, že PC pošle miesto 01 číslo 00 znamená to, že zariadenie sa má nastaviť do stavu adresy – tj znovu mu bude prideľovaná adresa na adrese 0, ako keby sme ho práve pripojili k PC. -V prípade, že táto hodnota bude nezmyselná (napr. 03 – tj číslo konfigurácie ktoré v zariadení nemáme) alebo počet bajtov by bol uvedený väčší ako 0, **treba to považovať za chybnú požiadavku. V takom prípade je nutné počítaču odpovedať paketom STALL (chyba), ktorý som spomínal už na začiatku.** (V podstate na akékoľvek nezmyselné požiadavky od PC treba odpovedať paketom STALL.) Len pre zopakovanie, takto vyzerá ten paket:

| Sync | STALL | EOP |
|---|---|---|
| 00000001 | 0x78 | 001 |

**Program v mikrokontroléri PIC16F1454:**

*USB:*
Procesor PIC16F1454 má v sebe implementovaný USB modul. Jeho súčasťou je transceiver, ktorý sa stará o elektrické kódovanie bitov. (To, čo som popisoval na začiatku.) Najprv si ale musíme oscilátor PIC obvodu nastaviť na 48 MHz. Ďalej USB modul obsahuje jednotku SIE (Serial Interface Engine), ktorá vypĺňa pakety a paketové sekvencie za nás. Nemusíme sa teda starať o štruktúru paketov ani o vytváranie paketových sekvencii. To všetko robí jednotka SIE. Pri programovaní sa teda môžeme venovať čisto len dátam. -USB modul sa ovláda pomocou zapínania/vypínania bitov v ovládacích registroch. Jeho aktuálny stav sa dá zisťovať čítaním bitov v stavových registroch. Začína sa samozrejme zapnutím USB modulu. Stane sa tak nastavením bitu č.3 v registri UCON (USB Control Register) do 1. Tento bit sa volá USBEN. -Register UCON má ako každý iný svoju fyzickú adresu v pamäti PIC obvodu. Tá ale nie je pre nás podstatná. Pri programovaní v prostredí MPLAB stačí písať symbolické názvy a kompilátor si adresy doplní. Podobne pri bitových operáciách nemusíme písať číslo bitu. Stačí názov. (Názvy sa pamätajú lepšie, ako čísla.)
Po zapnutí USB modulu sa do pamäti RAM automaticky namapuje od adresy 0x20 tzv. Buffer Descriptor Table (BDT). Jedná sa o 8 riadiacich registrov pre endpointy. 4 sú pre endpoint 0 out a 4 pre endpoint 0 in:

| Adresa: | Názov registra: |
|---|---|
| 0x20 | BD0STAT (out) |
| 0x21 | BD0CNT (out) |
| 0x22 | BD0ADRL (out) |
| 0x23 | BD0ADRH (out) |
| 0x24 | BD0STAT (in) |
| 0x25 | BD0CNT (in) |
| 0x26 | BD0ADRL (in) |
| 0x27 | BD0ADRH (in) |

Registrom BD0CNT sa nastavuje dĺžka pamäte endpointu v bajtoch. Samozrejme musí byť rovnaká, akú máme nastavenú v Endpoint deskriptoroch. Pre endpointy 0 in a 0 out máme nastavených 8 bajtov.
Registre BD0ADRH a BD0ADRL sú spolu a tvoria dohromady16 bitovú hodnotu. Zapisuje sa do nich adresa, kam v RAM-ke procesora PIC má byť umiestnená pamäť endpointu. Ja som zvolil pre endpoint 0 out adresu 0x0220 a pre endpoint 0 in adresu 0x0228. Čiže od adresy 0x0220 sa číta 8 bajtov a od adresy 0x0228 sa zapisuje 8 bajtov.
Register BD0STAT je riadiaci, kde majú význam jednotlivé bity. Nastavením bitu 7 (UOWN) do 1, odovzdáme

endpoint (jeho pamäť+príslušnú BDT) do vlastníctva jednotky SIE. Keď sa potom bit UOWN sám automaticky nastaví späť do 0 znamená to, že jednotka SIE odovzdala riadenie endpointu naspäť užívateľovi (programu). Vtedy si môžeme zvoliť, či a aký typ dátového paketu budeme posielať/prijímať. Či to bude typ DAT0 alebo DAT1. Ovláda sa to bitmi 6 (DTS) a 3 (DTSEN). Keď bit DTS je 1 znamená to, že aktuálny dátový paket ktorý sa bude posielať/prijímať, je typu DAT1. Keď bit DTS je 0 znamená to, že sa bude posielať/prijímať dátový paket typu DAT0. Je samozrejme vecou programátora zabezpečiť, aby bolo dodržané striedanie DAT0, DAT1, DAT0..... (O význame ktorého už bola reč.) Môžeme sa tiež rozhodnúť odpovedať STALL paketom (napr. v prípade chybnej alebo neexistujúcej požiadavky od PC. V takom prípade nastavíme bit 2 (BSTALL) do 1. -Nakoniec bitom DTSEN=1 sa ešte zapne samotné to odoslanie/príjem zvoleného dátového alebo chybového paketu a bitom UOWN=1 odovzdáme endpoint jednotke SIE, ktorá uskutoční prenos. -Keď popísanú procedúru vykonáme v registri BD0STAT(out) (čiže toho, ktorý sa týka výstupného endpointu), bude uskutočnený príjem dát z PC. Keď popísanú procedúru vykonáme v registri BD0STAT(in) (čiže toho, ktorý sa týka vstupného endpointu), bude uskutočnené vysielanie dát do PC. -Po skončení prenosu odovzdá jednotka SIE riadenie endpointu naspäť užívateľovi (programu). Spoznáme to tým, že bit UOWN sa sám opäť nastaví na hodnotu 0. -V čase, keď je endpoint v rukách jednotky SIE, je význam niektorých bitov v registri BD0STAT zmenený. Tými najdôležitejšími sú bity 2 až 5. V nich je totiž uložená hodnota PID, čiže identifikačného čísla token paketu z posledného prenosu. Tú využije programátor v programe, lebo z nej zistí, či PC poslal IN, OUT alebo SETUP token paket.

Poďme teraz naspäť do registra UCON, v ktorom je bit na zapnutie USB modulu zvaný USBEN. Ďalší bit, ktorý je v tomto registri dôležitý, je bit 4 (PKTDIS). Ten sa automaticky nastaví do 1 po tom, čo bol prijatý SETUP token paket. On zablokuje jednotku SIE, aby nespracovávala ďalšie pakety. To preto, aby mal užívateľ (program) čas obslúžiť tento aktuálny SETUP paket aj s príslušnou požiadavkou od PC. Potom nastaví bit PKTDIS naspäť do 0 a príjem ďalších SETUP paketov môže pokračovať. (Povolí jednotke SIE prijať ďalší SETUP paket.) V registri UCON je ešte jeden užitočný bit 5 (SE0). S názvom SE0 sme sa už stretli v popise USB komunikácie. Jedná sa o zmiznutie napätia na USB zbernici (koncová nula). Keď sa bit SE0 nastaví do 1 znamená to, že bola detekovaná koncová nula na zbernici. Bit slúž len na čítanie. (Nie je možné meniť jeho stav programom.)

Ďalší register, ktorým sa konfiguruje USB modul procesora, má názov UCFG (USB Configuration Register). Prvý dôležitý bit v ňom je bit 2 (FSEN). Ním si zvolíme, na ktorý z pinov USB portu sa pripojí ten odpor, o ktorom už bola reč na začiatku. Keď nastavíme FSEN do 1, odpor sa pripojí na pin D+. Tým počítač zistí, že k USB zbernici bolo pripojené nejaké zariadenie, ktoré bude komunikovať rýchlosťou Full speed (12 Mbit/s). Keď FSEN nastavíme do 0, odpor sa pripojí na pin D-. Vtedy počítač zistí, že bolo pripojené zariadenie, ktoré bude komunikovať rýchlosťou Low speed (1.5 Mbit/s). -Ďalší dôležitý bit registra UCFG, je bit 4 (UPUEN). Jeho nastavením do 1 fyzicky vykonáme to pripojenie odporu k zvolenému pinu. Od toho okamihu začne PC posielať nášmu zariadeniu kontrolné (SETUP) pakety.

Ďalší dôležitý register je statusový. Volá sa USTAT (USB Status Register). Je možné z neho len čítať. Prvý dôležitý bit je 2 (DIR). Ak je jeho hodnota 1 znamená to, že posledná transakcia bola IN token (tj dáta sa posielali do PC). Ak je jeho hodnota 0 znamená to, že posledná transakcia bola OUT token (tj dáta boli prijaté od PC). -Ďalšie dôležité bity v registri USTAT sú bity 3 až 6. Jedná sa o 4 bity, v ktorých je uložené číslo endpointu, ktorého sa týkala posledná transakcia.

Posledný register, ktorým sa nastavujú vlastnosti USB prenosu, sa volá UEPn (USB Endpoint n Control Register), pričom za „n“ treba dosadiť číslo endpointu. Napr.: UEP0. Aktivujú sa ním endpointy a nastavujú vlastnosti konkrétneho čísla endpointu, pričom do jedného čísla sú zahrnuté dva endpointy – tj. in a out (napr. 0 in a 0 out). Z toho vyplýva, že koľko máme čísel endpointov, toľko UEPn registrov treba nastaviť. -Prvý dôležitý bit tohto registra je 1 (EPINEN). Keď tento bit zapneme do 1, nastavíme si svoj endpoint ako vstupný (INPUT). Samozrejme INPUT je to z pohľadu počítača, čiže endpoint bude slúžiť len na odosielanie dát do PC. -Ďalší dôležitý bit je 2 (EPOUTEN). Keď tento bit zapneme do 1, nastavíme si svoj endpoint ako výstupný (OUTPUT). Samozrejme OUTPUT je to z pohľadu počítača, čiže tento endpoint bude slúžiť len na príjem dát z PC. Keď obidva tieto bity nastavíme do 1, číslo endpointu bude slúžiť aj na odosielanie aj na príjem dát. (Tj aktivujú sa pod týmto číslom dva endpointy: IN a OUT) -Ďalší dôležitý bit je 3 (EPCONDIS). Keď necháme tento bit na 0, povolíme pre náš endpoint SETUP dátové pakety od PC. Keď nastavíme bit do 1, zakážeme ukladanie SETUP dátových paketov do tohto endpointu. Budú fungovať len IN a/alebo OUT prenosy (podľa toho, ako sme si nastavili bity EPINEN a EPOUTEN). Posledným dôležitým bitom registra UEPn je bit 4 (EPHSHK). Ak ho nastavíme do 1, povolíme pre náš endpoint handshake pakety. (To sú tie ACK, NAK, STALL, ktoré slúžia na potvrdzovanie/odmietanie dát). Typicky sa zakazujú pri izochrónnom prenose, aby nespomaľovali komunikáciu. Taký prenos tu ale nepoužívame, takže bit EPHSHK zapneme a ACK tým

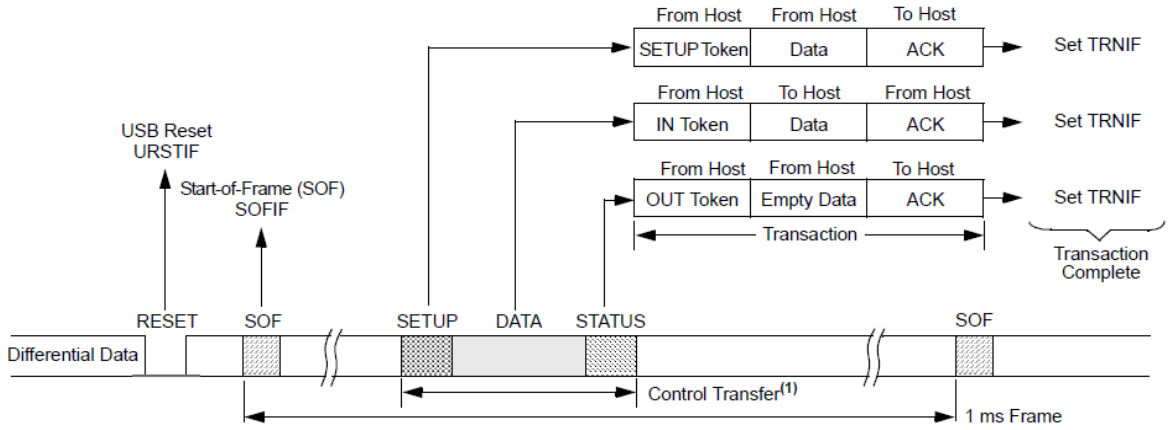
povolíme. Je tam ešte jeden bit, bit 0 (EPSTALL). Keď ho nastavíme do 1, zablokujeme tento endpoint. Naše zariadenie bude potom počítaču pri pokusoch o komunikáciu s týmto endpointom posielať automaticky STALL pakety (oznamujúce chybu prenosu).
V našom projekte USB vysielačky využívame (ako už vieme) okrem endpointu 0 ešte ďalší endpoint č. 1. Čiže dokromady 4 endpointy:

1. EP0 OUT
2. EP0 IN
3. EP1 OUT
4. EP1 IN

K aktivácii endpointov 0 použijeme práve naposledy spomínaný register UEPn. V tomto prípade konkrétne UEP0. Bity EPINEN a EPOUTEN v ňom nastavíme na 1, čím aktivujeme IN aj OUT endpointy 0. Hneď po tom sa nám namapuje od adresy 0x20 pamäti RAM už spomínaná oblasť BDT (Buffer Descriptor Table), skladajúca sa zo 4 registrov pre každý endpoint zvlášť. Pre endpointy 0 necháme bit EPCONDIS vypnutý, pretože chceme ich využívať na SETUP dátové prenosy. Samozrejme necháme tiež zapnutý aj bit EPHSHK, pretože handshake pakety (ACK, NAK, STALL) potrebujeme v našom type prenosu.
Ďalej si aktivujeme aj obidva endpointy č. 1. Tj v registri UEP1 nastavíme bity EPINEN a EPOUTEN na 1. Bit EPCONDIS však nastavíme na 1, pretože nechceme, aby nám na endpoint 1 chodili SETUP pakety. Budeme ho používať čisto len na prenos užívateľských dát. Bit EPHSHK necháme zapnutý, lebo handshake pakety sú vyžadované aj tu. Od chvíle, keď toto všetko spravíme, namapujú sa do pamäti RAM ďalšie dve štvorice registrov BDT), pre obsluhu endpointov 1. Celá pamäť bude potom vyzerať takto:

| Adresa: | Názov registra: |
|---|---|
| EP0 OUT: | |
| 0x20 | BD0STAT (out) |
| 0x21 | BD0CNT (out) |
| 0x22 | BD0ADRL (out) |
| 0x23 | BD0ADRH (out) |
| EP0 IN: | |
| 0x24 | BD0STAT (in) |
| 0x25 | BD0CNT (in) |
| 0x26 | BD0ADRL (in) |
| 0x27 | BD0ADRH (in) |
| EP1 OUT: | |
| 0x28 | BD1STAT (out) |
| 0x29 | BD1CNT (out) |
| 0x2A | BD1ADRL (out) |
| 0x2B | BD1ADRH (out) |
| EP1 IN: | |
| 0x2C | BD1STAT (in) |
| 0x2D | BD1CNT (in) |
| 0x2E | BD1ADRL (in) |
| 0x2F | BD1ADRH (in) |

Dĺžka pamätí pre endpoint 1 je nastavená na 64 bajtov. Tj do oboch registrov BD1CNT som vložil hodnotu 64. Adresa BD1ADRH(out): BD1ADRL(out) je nastavená na 0x02A0 a BD1ADRH(in): BD1ADRL(in) na 0x01A0. To znamená, že od adresy 0x02A0 sa začína oblasť 64 bajtov dát, ktoré posiela PC do nášho zariadenia a od adresy 0x01A0 začína oblasť 64 bajtov dát, ktoré odosielame počítaču. Prenos týchto dát sa ovláda registrami BD0STAT(in/out) ako som už popísal. Jedná sa o užívateľské dáta, a ich prenos sa rozbehne po skončení procesu enumerácie.
Každá úspešne dokončená paketová sekvencia sa končí tým, že PIC mikrokontrolér nastaví bit TRNIF do 1, čím sa vyvolá prerušenie hlavného programu a skočí sa na riadok 4 programu. Tam sa musí nachádzať odkaz na podprogramy, ktoré vzniknutú situáciu na USB zbernici obslúžia. O štruktúru paketov a paketových sekvencii sa programátor nemusí starať. O to sa stará jednotka SIE implementovaná do USB modulu mikrokontroléra. Na nasledujúcom obrázku je vidno, ako to vyzerá:



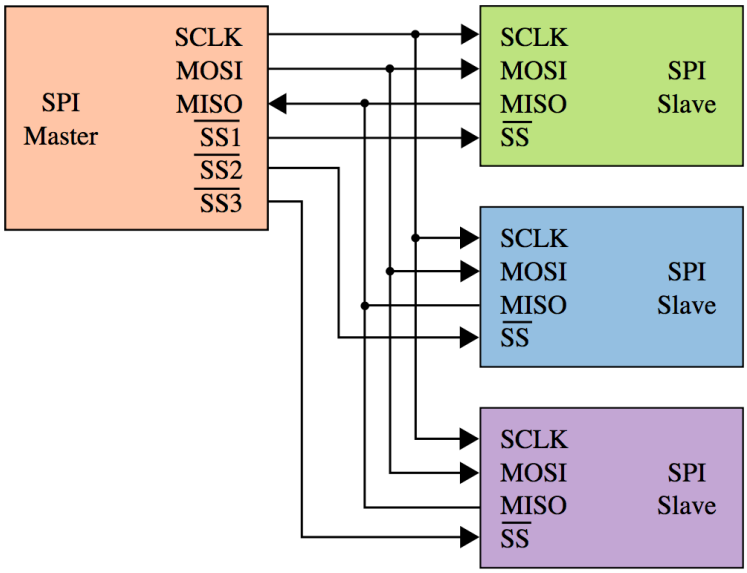

Dole je nakreslený milisekundový rámec, do ktorého sa ukladajú transakcie pre jednotlivé zariadenia pripojené k USB zbernici. Medzi nimi je aj naša transakcia. Rámec začína SOF paketom obsahujúcim poradové číslo rámca. Toto číslo sa automaticky ukladá do dvojice registrov UFRMH, UFRML. (Jedná sa o 11 bitovú hodnotu, preto sú dva.) Čísla rámcov v projekte nevyužívame. Spomínam to len pre zaujímavosť.-Ďalej v rámci nasledujú dátové transakcie, z ktorých jedna (na obrázku), je tá naša. Je tam uvedený príklad SETUP paketovej transakcie. Každá dielčia paketová sekvencia vyvolá prerušenie TRNIF a programátor ju musí správne obslúžiť podľa postupu, ktorý som už podrobne popisoval na začiatku. Prerušenia TRNIF sa samozrejme musia najprv aktivovať. Na to slúži register UIE a v ňom bit 3 (TRNIE), ktorý nastavíme do 1. To ale nie je všetko. Treba ešte zapnúť bit 2 (USBIE) v registri PIE2. Ním povoľujeme všetky prerušenia z USB portu. (Je ich viac.) Keď nastane ktorékoľvek z prerušení týkajúcich sa USB, nastaví sa automaticky bit USBIF, ktorý je v registri PIR2, do 1. Potom treba ešte zapnúť bity 7 (GIE) a 6 (PEIE) v registri INTCON. Bitom GIE (Global Interrupt Enable) sa totiž zapínajú všetky prerušenia celkovo. Bitom PEIE sa zapínajú všetky prerušenia od periférii. (USB port patrí medzi periférie. Periférii existuje viac.) Takže po nastavení bitov GIE, PEIE, USBIE a TRNIE do 1, zapneme prerušenia vyvolávané úspešne dokončenými paketovými sekvenciami. Od tej chvíle bude bit TRNIF spôsobovať prerušenie programu a skok na riadok 4 programu. Jeho vynulovaním prerušenie skončí. Vynulovať ho musí samozrejme užívateľ (program). -Bit TRNIF sa nachádza v registri UIR.

Je ešte jedno prerušenie na ktoré treba reagovať. Zapína sa pomocou bitu URSTIE. (Samozrejme bity GIE, PEIE, USBIE musia byť už zapnuté.) Jedná sa o stav RESET na zbernici. Vyvolá ho PC tým, keď po dobu dlhšiu ako 10 milisekúnd zmizne napätie z linky. Nastáva napríklad pred začatím dátovej transakcie, ktorá obsahuje adresu pre zariadenie. (Ako som už spomínal na začiatku.) Vtedy sa automaticky nastaví bit URSTIF do 1, čím sa vyvolá prerušenie a skočí na riadok 4 programu. Užívateľ (program) si potom zapíše pridelené číslo adresy do registra UADDR a vynuluje bit URSTIF, aby hlavný program mohol pokračovať. Od tej chvíle bude zariadenie reagovať už len na token pakety s touto adresou. Stav RESET sa ešte používa aj pri zobudení zariadenia zo spánku, ale režim spánku v našom projekte nevyužívame.

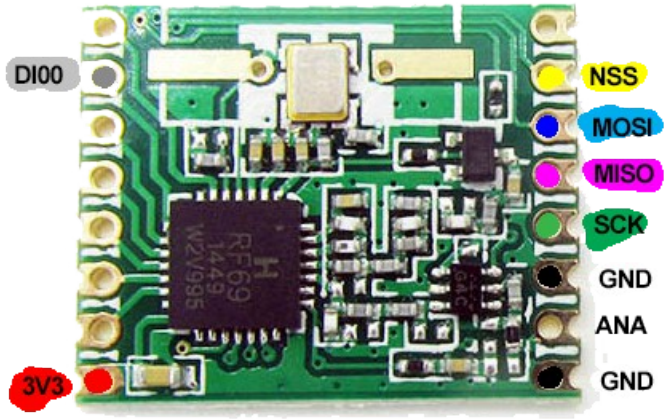
Takže: **Programátor musí mať v podprogramoch uložené k dispozícii všetky deskriptory ktoré boli použité pri enumerácii, a pomocou podmienok reagovať na prípadné požiadavky zasielané z PC vo forme SETUP data paketov. Všetky tieto požiadavky musí obslúžiť zaslaním požadovaného deskriptora(ov). Neexistuje totiž fixne dané poradie, v akom ich bude PC priebežne žiadať. Niektoré môžu byť vyžadované aj viac krát za sebou. Fixne je daný len začiatok enumerácie.**

*SPI:*

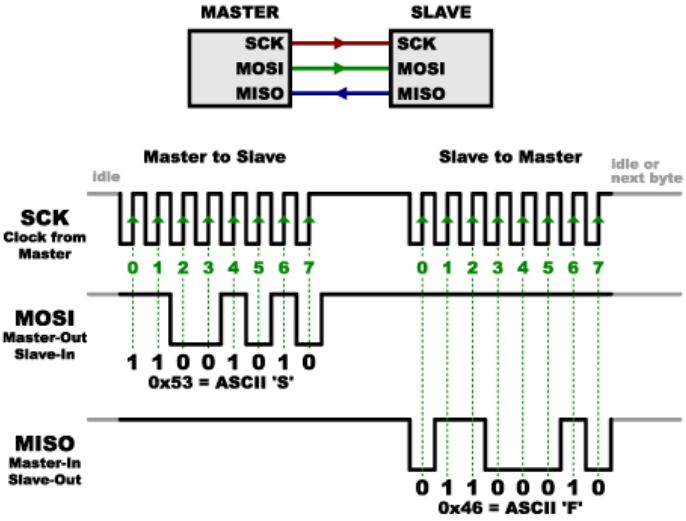
Ďalším druhom komunikácie ktorý používame v našom projekte je SPI. Pomocou nej sa komunikuje s bezdrôtovým modulom RFM69W. Využíva k prenosu 4 vodiče: zapínací, hodinový (CLK), dátový vstupný (MISO) a dátový výstupný (MOSI). Skratka MISO znamená „*Master Input Slave Output*“ (vstup dát do PIC a výstup dát z RF modulu). Skratka MOSI znamená „*Master Output Slave Output*“ (výstup dát z PIC a vstup dát do RF modulu). PIC obvod je totiž v komunikácii s RF modulom master. -Napriek tomu, že PIC16F1454 má SPI modul v sebe už implementovaný, nevyužívam ho. Spravil som to softvérovo. (Viac sa mi to tak páči.) Na linke SPI môže byť paralelne pripojených viac slave zariadení a mikrokontrolér pomocou zapínacieho vodiča si vyberá, s ktorým bude komunikovať. Pre každé zariadenie musí byť preto vyhradený jeden samostatný pin na zapínanie/vypínanie komunikácie. Takže koľko zariadení je na linke, toľko zapínacích pinov potrebuje.




Na obrázku sú tieto piny nazvané SS (s negáciou). V niektorých prípadoch sa totiž komunikácia povoľuje vypnutím napätia na zapínacom vodiči. Čiže keď je na vodiči napätie, modul je komunikačne neaktívny. Tento pin sa môže v rôznych prípadoch (pri rôznych moduloch komunikujúcich cez SPI), volať rôzne. V prípade modulu RFM69W sa volá NSS:




Písmeno „N“ znamená, že je tiež negovaný, čiže komunikácia s RF modulom sa zapína vypnutím napätia na vodiči NSS.
Samotná komunikácia prebieha tak, že na dátovom vodiči sa zapína/vypína napätie. Zapnutý stav je log.1 a vypnutý stav log.0. Samotné zapínanie a vypínanie na dátovom vodiči však nestačí. Aby príjemca považoval aktuálnu logickú úroveň za platnú, musí sa ešte na hodinovom vodiči SCK objaviť napätie. Vo chvíli, keď na hodinovom vodiči naskočí napätie, zapíše si príjemca aktuálny logický stav do registra, ako platný:




Čiže ak povedzme na vodiči MOSI drží PIC obvod nulové napätie (vypnutý stav) a počas tohoto stavu zapne na vodiči SCK napätie z 0 na 5 Volt, zapíše si RF modul do svojho interného registra tento bit ako log.0 Krátko na to však musí napätie na vodiči SCK zase vrátiť späť do 0 Volt, aby bola linka pripravená k prenosu ďalšieho bitu. Ak však na vodiči MOSI po zapnutí SCK napätie je, RF modul tento stav vyhodnotí ako log.1 a zapíše si ju do registra. Zmeny napäťových stavov na hodinovom vodiči SCK riadi výhradne Master, čiže v našom prípade PIC obvod. Ako ale prebieha čítanie dát z modulu RF? Je to rovnako jednoduché. Master posiela na hodinovom vodiči impulzy a na linke MISO číta logické stavy z RF modulu. Čiže: PIC obvod zapne na vodiči SCK napätie z 0 na 5 Volt a potom číta, aký stav je na vodiči MISO. Ak tam nič nie je, znamená to log.0 (a zapíše si ju do registra.) Ak tam však napätie je, znamená to log.1 (a zapíše si ju do registra). Potom napätie na vodiči SCK stiahne späť do 0 Voltov. Linka je tak pripravená k prenosu ďalšieho bitu. Potom znovu zapne SCK, číta stav na MISO a opäť vypne. Takto sa to opakuje 8x. Prenáša sa totiž vždy len 8 bitov. Príjemca si bity ukladá jeden vedľa druhého od najvyššieho po najnižší do 8 bitového registra, ktorého hodnotu potom spracuje. Keď je všetkých 8 bitov prenesených, master zapne na vodiči NSS napätie, čím vypne komunikáciu s RF modulom. Je to zároveň signál, že prenesený bajt je platný a môže sa jeho hodnota spracovať. Pri zahájení ďalšieho prenosu bajtu, musí master (PIC obvod) napätie na linke NSS opäť vypnúť. Tým znovu aktivuje komunikáciu. Atď... Tento proces sa stále opakuje.

Dĺžka trvania impulzov a vypnutých stavov na vodiči SCK závisí na rýchlosti prenosu, akú podporuje RF modul a akej je schopný PIC obvod. Modul RFM69W podporuje maximálnu frekvenciu hodinových impulzov 10 MHz. To znamená rýchlosť prenosu 10 Mbit/s. Napätie na vodiči SCK môže teda trvať najkratšie 50 nanosekúnd v zapnutom a 50 nanosekúnd vo vypnutom stave. Kratšie časy už budú robiť problémy. Ja som zvolil rýchlosť podstatne pomalšiu. Približne 1,2 Mbit/s. Úplne to stačí pre naše účely. A je tým zabezpečená aj vysoká miera spoľahlivosti prenosu s dostatočnou rezervou rýchlosti.

*Komunikácia s modulom RFM69W.*

Modul RFM69W obsahuje 85 registrov, ktorými sa nastavujú jeho vlastnosti a činnosť. Niektoré bajty či jednotlivé bity sú iba na čítanie. Iné na čítanie aj zápis. Každá adresa má svoje číslo od 0 do 85 a svoj špecifický význam, popísaný v data-liste modulu. V niektorých prípadoch majú svoj význam len jednotlivé bity registra, ktorými sa niečo zapína alebo vypína. Keď chceme do nejakého registra RF modulu zapisovať, najskôr sa pošle číslo adresy. Toto číslo musí mať však svoj najvyšší bit nastavený do 1. Tým modul zistí, že na adresu budeme zapisovať. Samotné číslo adresy je teda len zvyšných 7 bitov bajtu (7 bitové číslo). Vypadá to celé takto:

1. Vypneme na vodiči NSS napätie. (Tým zahájime prenos.)
2. Pošleme do modulu číslo adresy. Napr. 0x04 čiže 00000100. Pretože však do adresy chceme zapisovať, musíme ešte najvyšší bit nastaviť do 1. Takže posielané číslo bude v skutočnosti: 10000100. RF modul tým zistí, že do adresy bude zapisovať.
3. Hneď potom pošleme druhé číslo - tj tú hodnotu, ktorá má byť do registra zapísaná. Napr. 0x12, čo v dvojkovej sústave dáva číslo: 00010010. RF modul ju tam zapíše.
4. Zapneme na vodiči NSS napätie. (Prenos skončený.)

Podobne vyzerá aj čítanie z adresy:

1. Vypneme na vodiči NSS napätie. (Tým zahájime prenos.)
2. Pošleme do modulu číslo adresy. Napr. 0x14 čiže 00010100. Lenže pretože z adresy chceme čítať, ponecháme najvyšší bit 0. RF modul tým zistí, že obsah adresy bude posielať.
3. Hneď potom začne RF modul posielať obsah registra na tej adrese. V tomto kroku budeme teda čítať jeho hodnotu postupne po jednotlivých bitoch na na vodiči MISO. (Príde hodnota 0x40, čiže: 01000000)
4. Zapneme na vodiči NSS napätie. (Prenos skončený.)

Takže takýmto spôsobom sa z modulu číta a do modulu zapisuje pomocou SPI. Teraz už len treba vedieť, na čo ktorý register slúži a čo sa ním nastavuje resp. číta. Na to slúži datasheet od modulu RFM69W, kde výrobca všetko podrobne popísal.

*Činnosť modulu RFM69W*

Tento modul je široko konfigurovateľný. Dajú sa ním prenášať dáta po paketoch, ale i kontinuálne (čo je dobré napr. na digitálny prenos zvuku). Maximálna rýchlosť ktorú modul podporuje je 300 kbit/s (v režime FSK). V prípade paketového prenosu sa dáta umiestňujú do tzv pamäti FIFO (First In First Out), ktorá začína na adrese 0 a je veľká podľa toho, ako si nastavíme v jednom s nastavovacích registrov. Maximálne môže byť nastavených 255 bajtov. Dá sa nastaviť aj kedy sa má paket začať odoslať – či vtedy keď je nastavená veľkosť pamäte FIFO plná, alebo okamžite keď do nej vstúpi prvý bajt. -Ďalšia vec, ktorá sa na module nastavuje je druh rádiovej modulácie. Dajú sa nastaviť dve: FSK (Frequency Shif Keying – kľučovanie frekvenčným posunom) alebo OOK (On Off Keying – kľúčovanie zapínaním, vypínaním). V prípade modulácie FSK vysielač prepína medzi dvomi rôznymi frekvenciami. Keď vysiela na jednej frekvencii znamená to log.0, keď vysiela na druhej, log.1. To, ako sú tieto dve frekvencie od seba vzdialené je dané frekvenčným zdvihom, ktorý sa dá nastaviť. V prípade textovej USB vysielačky je jeho východzia hodnota nastavená na 1.281 kHz. Dá sa tiež nastaviť aj plynulosť prechodu medzi týmito dvomi frekvenciami pri vysielaní. Na to slúži Gaussov filter a tzv „rampa“. Štandardne mám Gaussovov filter vypnutý a rampu nastavenú na 1 milsekundu, čiže prechod medzi frekvenciou 1 a frekvenciou 2 prebehne za 1 milisekundu). Tento čas samozrejme nesmie byť dlhší, ako je charakteristické pre aktuálnu prenosovú rýchlosť. Tú mám defaultne nastavenú na najkratšiu možnú, tj 488.289 bit/s, čím sa zvýšil dosah textovej vysielačky. Prioritou nášho zariadenia je totiž čo najväčší dosah a spoľahlivosť prenosu. (Na prenos textu, riadiacich povelov, meracích dát a regulácie nie je treba veľkú prenosovú rýchlosť.) Ďalší parameter ktorý sa nastavuje je šírka pásma prijímača. Tá musí byť samozrejme dostatočne veľká na to, aby sa do nej zmestil spomínaný rozdiel medzi oboma frekvenciami (frekvenčný zdvih). Všeobecne platí, že zdvih musí byť polovicou šírka pásma. Tú používam najmenšiu akú modul ponúka, pretože čím je menšia, tým má prijímač menší šum a tým je teda citlivejší. To má automaticky vplyv na dosah, ktorý sa tým navýši. Východzia hodnota šírky pásma v zariadení je preto 2.604 kHz. Pre mód FSK sa menej nastaviť nedá. Zdvih 1.281 kHz sa do tohto okna bez problémov zmestí a je približne polovičný. -Šírka pásma je daná prenosovou rýchlosťou ktorá ju nesmie presahovať, čiže max. 2.604 kbit/s. V praxi však radšej o čosi menšia.

Teraz popíšem, ako vyzerá rádiový paket. Začína sa tzv. preambulou, ktorej dĺžka v bajtoch sa dá nastaviť. Štandardne ju mám 3 bajty. (Dá sa dať aj nulová. Vtedy je vypnutá.) Preambula je pravidelné striedanie bitov 01010,... dôsledkom čoho vysielač pravidelné prepína medzi frekvenciou 1 a frekvenciou 2. Na tomto striedajúcom sa úseku si prijímač protistrany zosynchronizuje hodiny a nameraným časom bude odmeriavať zvyšné bity paketu. Ďalšia časť paketu je voliteľná. je to tzv „*Sync word*“, čiže synchronizačné slovo. Mám ju povolenú. Jej dĺžka sa dá nastaviť. Defaultne je na 3 bajty. Jedná sa o čísla zvolené užívateľom, ktoré identifikujú paket v éteri. Je to teda akési „Identifikačné číslo“. Je vhodné ho však zvoliť tak, aby sa v ňom nevyskytovala dlhá rada núl alebo jednotiek za sebou a zároveň aby ich striedanie bolo čo najnepravidelnejšie. Identifikácia prijímacou stranou bude tak spoľahlivejšia. Ja som zvolil čísla D7 17 87 (hexadecimálne) čo dvojkovo dáva sekvenciu: 11010111 00010111 10000111. Myslím, že je to dostatočne nepravidelné a zároveň tam nie sú príliš dlhé rady rovnakých stavov. Prijímač pri vyhodnocovaní čísla *Sync word* postupuje tak, že akonáhle narazí na nezhodu, zvyšné bity už neskúma a paket zahodí. Dá sa nastaviť aj počet tolerovaných chýb v bitoch (napr. 1 bit), takže keď prijímač narazí na nezhodu v jednom bite pokračuje vo vyhodnocovaní ďalej. Maximálne je možné nastaviť 7 tolerovaných chybných bitov.. Defaultne to mám na 0. -Preambula ani synchronizačné slovo nie sú dostupné v pamäti FIFO. Sú len internou záležitosťou RF modulu. -Nasledujúca oblasť paketu je takisto voliteľná a jedná sa o adresu zariadenia (uzla), čiže ďalšie identifikačné číslo. Toto sa

už ukladá do FIFO pamäti a je preto je dostupné užívateľovi. Ja ho ale nevyužívam, preto ho mám vypnuté. -Nasledujúca oblasť paketu sú samotné dáta správy, ktorú chceme preniesť. Veľkosť FIFO pamäte sa dá nastaviť max. na 255 bajtov. Východzia hodnota v zariadení je 5 bajtov. (V hlavnom programe sa dá nastaviť max. iba do 50 bajtov, pretože dĺžka USB paketu je 64 bajtov, pričom zvyšok používam na prenos pomocných dát). -Posledná oblasť paketu je kontrolný súčet CRC, skladajúci sa z dvoch bajtov. (Čiže je to 16 bitové číslo.) Vysielač jeho hodnotu vypočíta z odosielaných dát a umiestni na koniec paketu. Prijímač si z prijatých dát vypočíta taký istý kontrolný súčet a porovná jeho hodnotu s tou prijatou v pakete. Ak sa zhodujú, dáta boli prijaté správne. **Ak nie, prijímač prijatý paket zahodí.** Kontrolný súčet je tiež voliteľnou súčasťou paketu. Ja ho mám zapnutý, pretože spoľahlivosť prenosu patrí medzi hlavné priority textovej USB vysielačky. CRC sa samozrejme neukladá do pamäti FIFO. Je to interná záležitosť modulu. **Mám to nastavené tak, že keď CRC je vyhodnotený správne, zapne sa na pine DIO0 modulu napätie. Toto napätie je vedené z RF modulu na pin RA5 PIC obvodu, kde vyvolá prerušenie hlavného programu.** Pin RA5 je nakonfigurovaný ako vstupný a sú na ňom povolené prerušenia. Piny portu „A“ sa konfigurujú v registri TRISA, pričom bit 5 nastavíme do 1. Číslo 1 sa podobá na písmeno I (Input), preto to tak spravili, že jednotkou nastavujeme pin ako vstupný. Nulou ho zas nastavujeme ako výstupný, pretože 0 sa podobá na písmeno O (Out). Pin RA5 má teda vysoký odpor lebo je nakonfigurovaný ako vstupný. Povolenie prerušenia od konkrétneho pinu portuA sa nastavuje v registri IOCAP (Interrupt On Change portA Positive edge register) - teda povolenie prerušenia od pozitívnej, tj nábežnej hrany impulzu. Inými slovami: keď sa na pine RA5 objaví napätie, táto skutočnosť vyvolá prerušenie hlavného programu v PIC obvode, čím skočí na riadok 4. Keby sme chceli vyvolávať prerušenie na spádovej hrane impulzu (čiže prerušenie nevyvolá objavenie sa napätia, ale až jeho zmiznutie), použili by sme register IOCAN (písmeno „N“ znamená Negative). Čiže v registri IOCAP nastavíme bit 5 do 1 a tým povolíme prerušenia od pinu č.5 na porte „A“.

Bezdrôtový modul RFM69W má 6 pinov, ktorých význam sa dá konfigurovať. Volajú sa DIO0, DIO1, DIO2, DIO3, DIO4, DIO5. Konfigurujú sa v registroch RF modulu. **Z nich používam len pin DIO0. Jeho význam pri príjme je ten, aby oznámil PIC obvodu, že bol prijatý paket a bol prijatý správne (CRC sa zhodujú). To je signál pre PIC obvod, aby prerušil hlavný program a prečítal prijaté dáta z FIFO pamäti RF modulu. -Pri vysielaní má však pin DIO0 iný význam. Keď PIC obvod prijme z počítača cez USB správu ktorú má odvysielať, preruší sa hlavný program a počas prerušenia začne PIC obvod posielať do FIFO pamäti RF modulu túto správu, ktorý ju okamžite začne vysielať. Keď už je RF modul s vysielaním hotový, zapne na pine DIO0 napätie. Toto napätie je pre PIC obvod signál, aby ukončil prerušenie a vrátil sa späť k hlavného programu.**

V skratke ešte spomeniem ten druhý typ modulácie, OOK (On Off Keying). I keď sa dá v programe zvoliť, ja ho nepoužívam, lebo som s ním nemal dobré skúsenosti. Dôvodom je väčší vplyv rušenia na chybovosť prenosu, nakoľko sa jedná o zmeny amplitúdy, lebo zapnutý signál = log.1, vypnutý signál = log.0. Počas vypnutého signálu (log.0), môže nejaké rušenie z vonku spôsobiť falošné nabehnutie log 1 a prijímač potom CRC vypočíta špatne. Paket tak skončí zahodený. V hlavnom okne programu je 5 nastavení súvisiacich s OOK moduláciou, ktoré sú neaktívne. Aktivujú sa jej zvolením. Ich popis však vynechávam z hore spomenutých dôvodov. **Odporúčam používať výhradne FSK.** -Paket teda reálne vyzerá takto:

- Preambula (1010...)
- Synchronizačné slovo (ID paketu)
- Dáta správy
- 2 bajty kontrolného súčtu CRC

Z toho všetkého čo som tu popísal vyplýva, že po zapnutí textovej vysielačky (tj po zasunutí do USB portu), sa modul RFM69W musí najprv inicializovať - čiže vykonať prvotné nastavenia, ktoré uvedú modul do funkčného stavu. V našom prípade je to konkrétne 35 čísel, ktoré sa zapíšu do 35 rôznych adries registrov. Vypíšem zoznam tých čísel a adries kam idú, vrátane stručného popisu významu:

| Adresa | Hodnota | Popis |
|---|---|---|
| 0x02 | 00000000 | Bitmi 5-6 je nastavený paketový mód prenosu informácii<br>Bitmi 3-4 je nastavená modulácia na FSK<br>Bitmi 0-1 je vypnutý Gaussov filter (o ktorom bola už reč v texte) |
| 0x03 | 0xFF | Bitová rýchlosť, horný bajt |
| 0x04 | 0xFF | Bitová rýchlosť, dolný bajt. Vypočíta sa: 32MHz/0xFFFF ≈ 488,289 bit/s |
| 0x05 | 0x00 | Frekvenčný zdvih, horný bajt. |
| 0x06 | 0x15 | Frekvenčný zdvih, dolný bajt. Vypočíta sa: 61Hz*0x0015 = 1.281 kHz |
| 0x07 | 0x6C | Frekvencia, horný bajt. |
| 0x08 | 0xAA | Frekvencia, stredný bajt. |
| 0x09 | 0x7F | Frekvencia, dolný bajt. Vypočíta sa: 61Hz*0x6CAA7F = 434.413635 MHz |
| 0x11 | 10000000 | Bit 7 zapína koncový stupeň.<br>Bitmi 0-4 je nastavený výkon na -18 dBm (OutPwr) |
| 0x12 | 00000010 | Bitmi 0-3 je nastavená rampa na 1 milisekundu. (PaRamp) |
| 0x18 | 10000001 | Bitom 7 je nastavená vstupná impedancia prijímača na 200 Ohm (LnaZin)<br>Z bitov 3-5 sa číta aktuálna úroveň citlivosti (keď je prepnutá na automatiku)<br>Bitmi 0-2 je nastavená citlivosť na maximum. (LnaGain) |
| 0x19 | 00010111 | Bitmi 7-5 sa nastavuje DccFreq.<br>Bitmi 0-4 je nastavená šírka pásma na minimum, tj 2.604 kHz. (RxBw) |
| 0x25 | 00000000 | Bitmi 6-7 je nastavená funkcia pinu DIO0 tak, ako som popísal v texte. |
| 0x29 | 0xFF | Spodný prah sily signálu, ktorý prijímač začne spracúvať (Rssi) |
| 0x2C | 0x00 | DĺžkA preambuly, horný bajt. |
| 0x2D | 0x03 | DĺžkA preambuly, dolný bajt. Je nastavená na 0x00003, čiže 3 bajty. |
| 0x2E | 10010000 | Bitom 7 je zapnuté synchronizačné slovo. (tj bude súčasťou paketu.)<br>Bitom 6 sa zapína/vypína FifoFillCondition<br>Bitmi 3-5 je nastavená dĺžka synchronizačného slova na 2+1=3 bajty<br>Bitmi 0-2 je nastavený počet tolerovaných chýb na 0 bajtov |
| 0x2F | 0xD7 | Hodnota prvého bajtu synchronizačného slova je 0xD7 |
| 0x30 | 0x17 | Hodnota druhého bajtu synchronizačného slova je 0x17 |
| 0x31 | 0x87 | Hodnota tretieho bajtu synchronizačného slova je 0x87 |
| 0x37 | 00010000 | Bitom 7 je nastavený formát paketu na fixnú dĺžku<br>Bitmi 5-6 je vypnuté kódovanie dát, ktoré má vylepšiť prenos.<br>Bitom 4 je zapnutý CRC v pakete (kontrolný súčet).<br>Bitom 3 je nastavené, že v prípade nesprávneho CRC sa vymaže pamäť FIFO.<br>Bitmi 1-2 je vypnutá adresa zariadenia. (Tj paket ju nebude obsahovať.) |
| 0x38 | 0x05 | Tu je nastavená dĺžka paketu na 5 bajtov. (Dĺžka jeho dátovej časti.) |
| 0x3C | 10000110 | Bitom 7 je nastavené, že vysielanie začne, keď je aspoň 1 bajt vo FIFO<br>Bity 0-6 nemajú pre toto nastavenie význam. |
| 0x58 | 0x2D | Touto hodnotou je nastavená vysoká citlivosť prijímača. |
| 0x5A | 0x55 | Touto hodnotou je nastavený normálny mód vysielača (výkon max. 13dBm) |
| 0x5C | 0x70 | Touto hodnotou je nastavený normálny mód vysielača (výkon max. 13dBm) |
| 0x6F | 0x30 | Stupeň automatického nastavovania citlivosti prijímača pri A/D prevode |
| 0x01 | 00000000 | Bitmi 2-4 je modul nastavený predbežne do režimu spánku. (hodnota 000) |

Pri zasunutí textovej USB vysielačky do USB portu, sa najprv všetky hore uvedené čísla prenesú z PIC obvodu do modulu RFM69W, čím sa vykoná jeho prvotné nastavenie. Potom sa rozbehne hlavný program. -Podobne pri otvorení PC programu sa uskutoční takýto prenos čísel, avšak už uložených v PC programe. To preto, aby sa aktualizovali posledné zmeny nastavenia vykonané užívateľom v počítači (ktoré zostali uložené v PC registroch). -Počítačový program obsahuje okno nastavení RF modulu, ako je frekvencia, zdvih, výkon, atď.... Akékoľvek zmeny nastavenia sa okamžite v reálnom čase prenášajú cez USB do PIC obvodu a odtiaľ hneď preposielané na príslušné adresy RF modulu, kde sa aplikujú. Hneď na to je daný register RF modulu (ktorého hodnota bola zmenená) prečítaný a táto hodnota preposlaná späť do PC, kde sa aplikuje späť na ovládací prvok. (Každý ovládací prvok Windows okna sa dá totiž meniť aj softvérovo.) Uvedený proces tvorí akúsi spätnú kontrolu pre užívateľa, či k zmene parametra v RF module naozaj došlo. Povedzme, že by sa RF modul pokazil.

Zmeny parametrov sa potom nebudú dať vykonávať a užívateľ to zistí tak, že ovládací prvok (track-bar alebo combo-box) sa bude vracať späť na pôvodnú hodnotu.
V sekvencii čísel, ktoré sme posielali sú všetky potrebné nastavenia, aby RF modul mohol pracovať a prenášať dáta. Sekvencia je zakončená uvedením modulu do režimu spánku. Režimy modulu sa ovládajú bitmi 2-4 na adrese 0x01. Sú tam tieto režimy (vyberám len tie, ktoré budeme používať):

- 000 → Režim spánku. (Sleep)
- 001 → Standby mode
- 011 → Režim vysielania (Tx)
- 100 → Režim príjmu (Rx)

Čiže keď chcem zostávať na prijme, musím tieto bity nastaviť na hodnotu 100 a keď chcem vysielať, na hodnotu 011. Počas komunikácie medzi oboma textovými vysielačkami sa teda prepína medzi týmito dvoma režimami. Keď užívateľ mení nastavenia, modul prejde za ten čas do režimu „Standby“, čo je kód 001. Samozrejme, že modul nie je schopný prejsť z režimu príjmu do režimu vysielania a naopak okamžite. Trvá mu to nejaký čas. Preto existujú stavové bity indikujúce pripravenosť. Keď vydám modulu príkaz prejsť do režimu Rx, musím sledovať cyklicky stav bitu č.6 na adrese 0x27. Keď modul ešte nie je pripravený na príjem, bude jeho hodnota 0. Keď už je pripravený, bude mať hodnotu 1. Až potom môže program pokračovať ďalej. Podobne je tomu s režimom vysielania. Keď vydám modulu príkaz prejsť do režimu Tx, musím sledovať cyklicky stav bitu č.5 na adrese 0x27. Keď modul ešte nie je pripravený na vysielanie, bude jeho hodnota 0. Keď už je pripravený, bude mať hodnotu 1. Až potom môžem začať posielať do pamäte FIFO svoje dáta, ktoré sa tým začnú okamžite vysielať. -Obdobne je tomu aj s prechodom do režimu „Standby“ alebo „Sleep“. Keď vydám modulu príkaz prejsť do režimu „Standby“ alebo „Sleep“, musím sledovať cyklicky stav bitu č.7 na adrese 0x27. Keď modul do jedného z týchto režimov ešte neprešiel, bude jeho hodnota 0. Keď už prešiel, bude tam hodnota 1.
Keď je modul prepnutý do režimu príjmu (Rx) a príjme nejaký paket, informáciu o jeho prijatí vyšle len vtedy, keď ho prijal celý (tj počet bajtov nastavenej dĺžky paketu). Východzia hodnota je nastavená na 5 bajtov. (To je dĺžka dátovej časti paketu.) Modul má pri tomto nastavení vyhradené v pamäti FIFO miesto pre 5 bajtov. Keď sa zaplní, objaví sa napätie na pine DIO0. Toto je vedené na pin RA5 v PIC obvode, kde sa vyvolá prerušenie hlavného programu. PIC obvod potom prečíta obsah pamäti FIFO (prenesené dáta) a odošle do PC k zobrazeniu pre užívateľa.
Pri vysielaní sa postupuje opačne. Najprv prepnem modul do režimu vysielania (Tx). Potom pošlem do pamäti FIFO dáta k odoslaniu. (To prebehne rýchlo.) Modul začne tieto dáta hneď vysielať. Keď je s tým hotový, zapne na pine DIO0 napätie. Napätie je vedené na pin RA5 PIC obvodu z ktorého sa dozvie, že sa môže prepnúť späť na príjem.

**Program na strane PC.**
Program je napísaný v jazyku C++ a používa funkcie Windows API. Nimi sa obsluhujú prvky okna ako editbox do ktorého píšeme správu alebo track-bary či combo-boxy, ktorými nastavujeme vlastnosti prenosu. Windows posiela oknám otvorených programov rôzne správy o aktuálnom dianí v PC, ako je napr. pohyb myšou, písanie na klávesnici atď.. Posiela aj správy o dianí na USB zbernici. Operačný systém Windows pristupuje k portom PC ako k súborom. Teda aj k USB portu pristupuje ako k súboru. Každý súbor treba najskôr otvoriť. Na to slúži funkcia CreateFile(). V parametroch tejto funkcie sa zadáva, že ho otvárame ako existujúci, otvárame ho na čítanie aj zápis a budeme s ním komunikovať asynchrónne. Funkcia CreateFile() nám potom vráti identifikačné číslo súboru (čiže USB portu), ktoré budeme používať pri operácii čítania vo funkcii ReadFile() a zápisu vo funkcii WriteFile(). Toto identifikačné číslo sa volá *Handle* (rukoväť). Lenže to nie je všetko, čo o nás funkcia CreateFile() vyžaduje. To najhlavnejšie je cesta k súboru. V našom prípade adresa portu, do ktorého je zasunutá USB vysielačka. Tú nepoznáme, preto ju musíme zistiť. Na to slúžia viaceré funkcie a určitý postup s nimi, ktorý teraz popíšem: Najskôr použijeme funkciu SetupDiGetClassDevs(), ktorá vráti rukoväť ku všetkým zariadeniam danej triedy pripojeným k PC. Vo svojich parametroch vyžaduje zadanie kódu tejto triedy. Naše zariadenie je triedy HID, preto do nej vložíme kód pre triedu HID. Je to však pomerne zložité číslo, preto použijeme k jeho zisteniu funkciu HidD_GetHidGuid(). Tá nám vráti a uloží do špeciálnej premennej tento kód. Túto premennú potom vložíme do už spomínaného parametra funkcie SetupDiGetClassDevs(). -Keď už teraz máme „v rukách“ rukoväť ku všetkým HID zariadeniam aktuálne pripojeným k PC, musíme vyhľadať to

naše. Na to slúži funkcia SetupDiEnumDeviceInterfaces(). Zakaždým keď túto funkciu znovu zavoláme, načíta iné zariadenie, ktoré je ďalšie v poradí. Preto ju treba volať cyklicky až dokiaľ neprejde všetky zariadenia a nenájde to naše. Funkcia vracia odkaz na aktuálne načítané zariadenie. Tento odkaz použijeme v parametroch funkcie SetupDiGetDeviceInterfaceDetail(), ktorá získa podrobné informácie o rozhraní práve načítaného zariadenia. Jedna z nich je aj „DevicePath“, čiže cesta k jeho rozhraniu. Túto cestu vyžaduje funkcia CreateFile() ktorá slúži na otvorenie zariadenia. Ona vracia rukoväť (handle) k zariadeniu. To je to hlavné, čo potrebujeme, aby sme mohli používať funkcie ReadFile() a WriteFile(). Rukoväť je potrebná aj vo funkcii HidD_GetAttributes() ktorou zistíme, či otvorené zariadenie je práve to naše. Ňou totiž získame všetky atribúty otvoreného zariadenia – mimo iné aj čísla **Product ID a Vendor ID.** (O ktorých som už podrobne písal.) V prípade textovej USB vysielačky majú hodnotu, akú sme si zvolili:

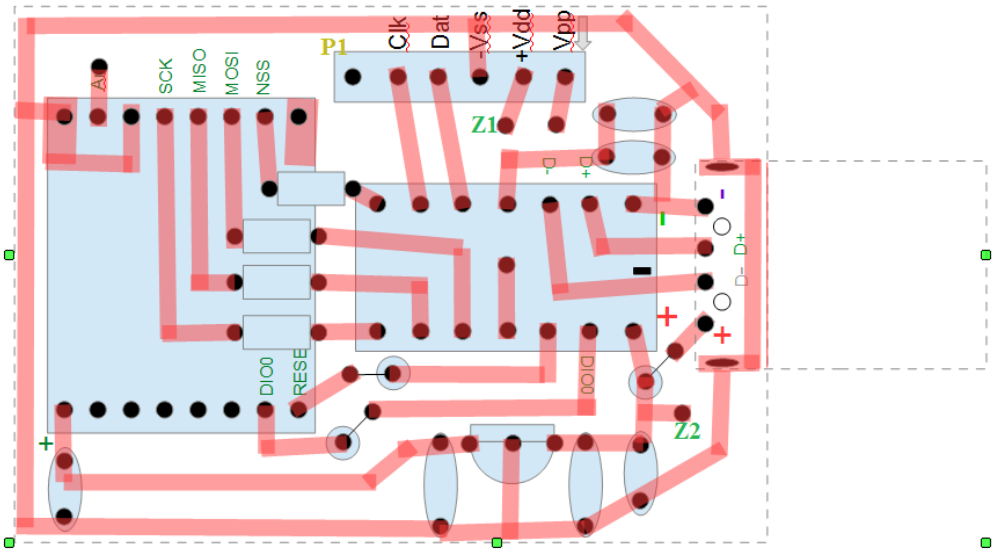
**Vendor ID = 04D8**
**Product ID = 01A5**

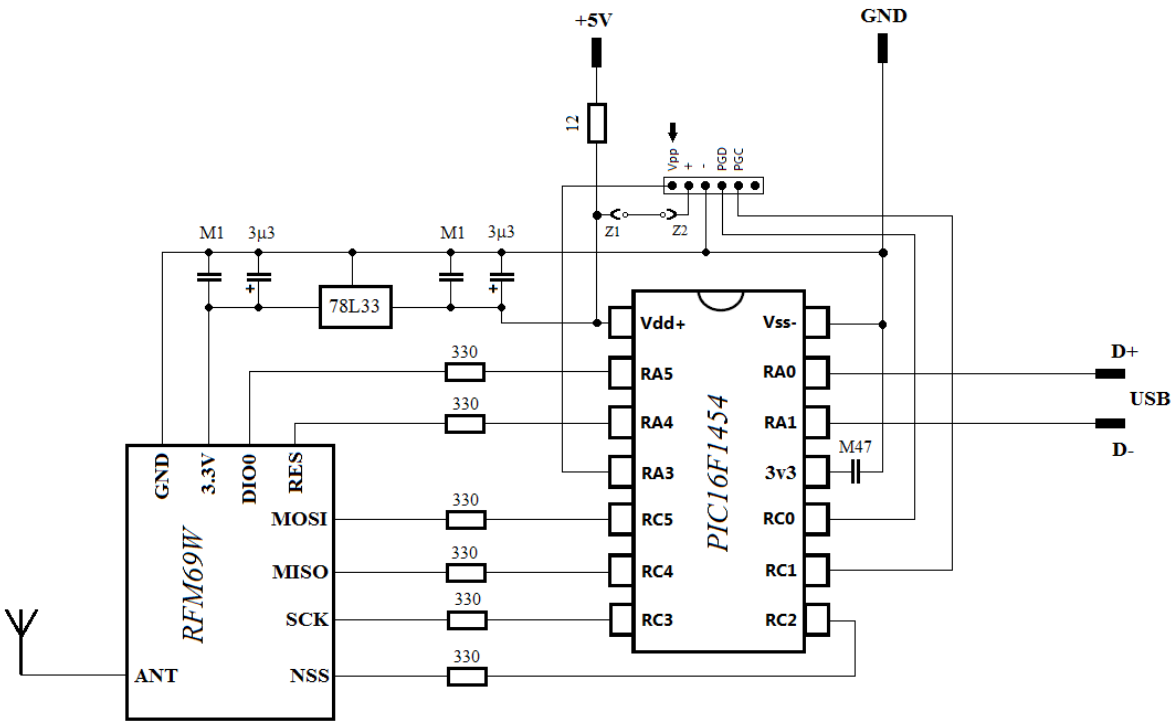
Ak sa tieto čísla nezhodujú s načítaným zariadením, nejedná sa o to naše. Treba ho potom zatvoriť a načítať ďalšie. Zatvoríme ho pomocou funkcie CloseHandle(). Keď nájdeme svoje zariadenie (textovú USB vysielačku), necháme ho otvorené a proces hľadania ukončíme. Od tejto chvíle s ním komunikujeme pomocou funkcii ReadFile() a WriteFile(). Do nich sa vkladá práve získaná rukoväť. Funkcia ReadFile() číta dáta z USB vysielačky. Lenže my nevieme dopredu, kedy z nej dáta prídu. Treba na ne čakať. Čakanie by ale zastavilo celý program. Preto je v programe vytvorené vlákno, ktoré beží súbežne s programom. Vlákno je zacyklené nekonečnou slučkou, v ktorej sa nachádza funkcia ReadFile() a funkcia WaitForSingleObject() čakajúca na príchod dát zo zariadenia. Okrem iného je schopná tiež vyhlásiť chybu, keď sa stratí spojenie so zariadením – napr. z dôvodu, že sme ho odpojili z USB portu. Preto náš program vie, kedy ho odpojíme. Keď je zariadenie odpojené, nekonečná slučka vlákna sa ukončí a program začne každých 10 milisekúnd vykonávať hore popísanú procedúru skenovania zariadení HID na USB zbernici. Keď vysielačku opäť zasunieme, program ju opäť nájde a cyklus skenovania sa skončí. Hneď potom sa vlákno znovu rozbehne a čaká na príchod dát. Funkcia ReadFile() ukladá prijaté dáta do textového bufra, ktorý má dĺžku 65 bajtov. Už som spomínal, že naše zariadenie je ale nakonfigurované tak, že prenáša len 64 bajtov. Vo Windowse je však v bufri ešte jeden bajt naviac, konkrétne bajt 0. Tento bajt je vyhradený pre iné účely a v programe ho nepoužívam. Dáta teda začínajú až od pozície 1 a pokračujú k pozícii 64. (Bajt 0 je 65-ty.) V prípade, že chceme dáta do zariadenia poslať, stačí ich vložiť do bufra (najviac 64) a zavolať funkciu WriteFile(). Vtedy sa obsah bufra okamžite odošle do PIC obvodu. Avšak aj tu však platí, že dáta treba zapisovať až od pozície 1. Na pozícii 0 musí byť vždy zapísaná hodnota 0. Inak to nebude fungovať.

**Ako pracuje komunikácia medzi textovými vysielačkami.**
Obe textové vysielačky majú v sebe naprogramovaný systém zabezpečenia prenosu proti strateným paketom napr. v dôsledku slabého signálu, rušenia, atď... Tento systém je nezávislý na PC. Keď PC pošle do PIC dáta pre prvý rádiový paket (východzia hodnota je 5 bajtov). PIC program vloží na koniec dát ešte poradové číslo paketu. Prvý paket má poradové číslo 1, preto do jeho piateho bajtu vloží číslo 1. Z toho vyplýva, že užitočné dáta, ktoré chceme z počítača preniesť sú vždy o jeden bajt kratšie. (V našom prípade len 4 bajty.) Keď protistrana paket prijme, pošle zariadeniu naspäť potvrdzujúci paket. Ten má v prvom bajte číslo 4 (toto číslo nepatrí medzi ASCII znaky využiteľné k dopisovaniu, preto sa dá použiť na takéto účely) a na poslednom piatom bajte poradové číslo prijatého paketu. Vysielacia strana tak zistí, že paket s týmto číslom bol doručený a pokračuje zaslaním ďalšieho paketu - tj ďalšiu časť správy s dĺžkou 4 bajty. Keby potvrdenie od protistrany neprišlo, vysielacia strana za jednu sekundu zopakuje ten istý paket, tj s rovnakým poradovým číslom. Opakuje ho dovtedy, dokedy neobdrží potvrdzujúci paket. Najviac ho opakuje 10x. Ak ani na desiaty raz nedostane potvrdenie, vyhlási zlyhanie prenosu v informačnom okienku. Môže nastať ešte jedna situácia: paket bol síce doručený (protistrana ho prijala), ale stratil sa potvrdzujúci paket. Čiže vysielacia strana neobdržala potvrdenie. Preto opakuje svoj aktuálny paket s rovnakým číslom. Prijímacia strana ho však už raz prijala a poslala do PC k zobrazeniu pre užívateľa. Teraz by ho zobrazoval druhý krát, keby ho poslala do PC. To ale nechceme. Nechceme, aby sa nám v prijatom texte opakovali niektoré úseky správy viac krát za sebou. Preto prijímacia strana porovnáva ešte poradové číslo aktuálne prijatého paketu s predošlým a keď sa zhodujú, neodošle dáta do PC k zobrazeniu (lebo je to ten istý), ale pošle znovu len potvrdzujúci rádiový paket vysielacej strane. Keď sa to už podarí a vysielacia strana potvrdzujúci paket konečne obdrží, dá povel PC k poslaniu ďalšieho úseku

správy (o dĺžke 4-och bajtov), ktorá má byť odoslaná.
Vysielačka komunikuje s PC cez USB pomocou 64 bajtových paketov tam aj späť. Keď je dĺžka rádiového paketu nastavená na 5 bajtov, posiela sa správa rozdelená po 4 bajtových úsekoch. Čiže každý aktuálny USB paket s dĺžkou 64 bajtov má obsadené vždy len 4 bajty zo správy, ktorú chce užívateľ odoslať protistrane. To ale nie je úplne všetko. Z PC do zariadenia sa posiela ešte jeden údaj, ktorý nemá so správou nič spoločné ale je len inštrukciou pre PIC procesor. Inštrukciou o tom, že práve poslané dáta sú správa k odoslaniu, alebo dáta k nastaveniu. Keď totiž meníme nastavenia ako je šírka pásma, výkon, prenosová rýchlosť, posielajú sa do PIC tiež dáta a procesor ich musí rozlíšiť od dát určených na rádiové vysielanie. K tomuto účelu je určený prvý bajt 64 bajtového paketu. Ak je v ňom číslo 0x01 znamená to, že sa jedná o dáta určené pre rádiové vysielanie. Ak je na tej pozícii číslo 0x02, jedná sa o nastavovacie dáta modulu. Keď napríklad meníme polohu prvého trackbaru s názvom „Ladenie Fo“, meníme tým frekvenciu. Pri každej zmene toho trackbaru sa v reálnom čase odosielajú do PIC obvodu cez USB tri čísla: v prvom bajte číslo 0x02, v druhom bajte adresa registra v RF module na ktorej sa nastavuje frekvencia - čiže 0x08 a v treťom bajte hodnota frekvencie ktorá sa mení podľa toho, ako hýbeme trackbarom. PIC procesor tieto zmeny okamžite preposiela do RF modulu na uvedenú adresu a frekvencia sa preto mení. To isté sa deje aj pri ostatných zmenách nastavení buď pomocou track-barov alebo pomocou combo-boxov. -Už som hovoril o tom, že po každej zmene nejakého parametra v RF module PIC obvod túto adresu ešte znovu prečíta a jej obsah prepošle späť do PC kvôli spätnej kontrole, či k zmene naozaj došlo. Keby nie (napr z dôvodu poruchy RF modulu), ovládací prvok v okne programu by sa vrátil späť na pôvodnú hodnotu. Čísla v USB pakete sú rovnaké s rovnakým významom ako pri odosielaní do PIC, akurát teraz v opačnom smere. Čiže v prvom bajte je číslo 0x02 (tj jedná sa o zmenu parametrov), v druhom bajte je adresa modulu a v treťom bajte hodnota na tej adrese. Táto hodnota má potom softvérový vplyv na polohu ovládacieho prvku v okne programu.
Keď PC posiela správu k rádiovému odvysielaniu, pošle do PIC obvodu v prvom bajte USB paketu číslo 0x01 (tj jedná sa o správu pre vysielanie). Potom nasleduje počet bajtov tej správy podľa toho, koľko sme si nastavili – tj aký dlhý máme nastavený rádiový paket. Ak je nastavený na 5 bajtov, posielajú sa 4 bajty správy. (Dôvod som už vysvetlil.) Keď bola správa úspešne rádiovo odvysielaná a doručená, zariadenie obdrží potvrdzujúci paket od protistrany. Vtedy pošle do PC povel k odoslaniu ďalšej časti správy. Na to slúži posledný, 64-tý bajt USB paketu. Keď sa v ňom objaví číslo 0x01 znamená to, že paket bol doručený, čo je zároveň povel pre PC k poslaniu ďalšej časti správy (v našom prípade ďalšie 4 bajty). Ak však prenos zlyhal (tj minulo sa všetkých 10 pokusov na odvysielanie), zariadenie pošle do PC na pozícii 64 číslo 0x02. Vtedy sa užívateľovi zobrazí na pracovnej ploche informačné okienko informujúce o zlyhaní prenosu. Po jeho potvrdení sa PC program resetuje a pripraví na ďalšie použitie.
Pri zaškrtnutí políčka indikátory v hlavnom okne programu sa začne zobrazovať sila signálu v dBm. Môžem si nastaviť aj riadenie citlivosti prijímača (LnaGain) na „auto“. Vtedy sa začne v hlavnom okne okrem sily signálu zobrazovať aj aktuálna hodnota automaticky nastavenej citlivosti. Vtedy sa deje pri USB komunikácii toto: pri zaškrtnutí políčka sa v prvom bajte USB paketu pošle číslo 0x0A, čo je povel pre PIC obvod, že ideme nastavovať „vypínače“. V druhom bajte sa pošle číslo 0x01, čo znamená „zapni vypínač indikátorov“. Pri odškrtnutí políčka „indikátory“ sa pošle číslo 0x00 čo znamená „vypni vypínač indikátorov“. (Ním sa vypnú všetky). Keď sú indikátory zapnuté, zobrazuje sa aktuálna sila signálu v hlavnom okne a tiež aktuálne nastavená citlivosť LnaGain. Ak však nie je nastavená na „auto“, je PC programom jej zobrazovanie vypnuté. PIC obvod posiela uvedené údaje každých 5 milisekúnd a aktualizuje ich. V prvom bajte USB paketu posiela vtedy číslo 0x0A. Z toho PC program zistí, že sa jedná o dáta z indikátorov. V druhom bajte posiela hodnotu sily signálu a v treťom bajte hodnotu aktuálne nastavenej citlivosti. (PC ju zobrazuje len vtedy, keď je nastavená na „auto“.)

**Oživenie zariadenia.**
Po osadení a zacínovaní súčiastok pripojíme programátor PICKIT 2 alebo PICKIT 3 do konektora P1. Šípka, ktorá je nakreslená na programátore je vyznačená aj v schéme a na nákrese plošného spoja aby bolo jasné, ako ho otočiť. -Potom prepojíme piny Z1 a Z2 drátovou prepojkou. (Je to dočasné napájanie pre zariadenie z programátora.) Nakoniec nahráme program do PIC a prepojku zrušíme. (Pri ďalšej potrebe programovania už bude zariadenie napájané z USB zbernice.) Po zasunutí zariadenia do USB, by malo začať fungovať. Treba však počkať na dokončenie procesu enumerácie, než mu PC pridelí adresu a zaregistruje si ho do správcu zariadení.


P1
Clk
Dat
-Vss
+Vdd
Vpp
SCK
MISO
MOSI
NSS
Z1
Z2
D+
D-
DIO0
RESE


+5V
GND
12
Vpp
PGD
PGC
Z1
Z2
M1
3µ3
78L33
330
Vdd+
Vss-
RA5
RA0
RA4
RA1
RA3
3v3
RC5
RC0
RC4
RC1
RC3
RC2
PIC16F1454
M47
D+
USB
D-
GND
3.3V
DIO0
RES
MOSI
MISO
SCK
NSS
ANT
RFM69W