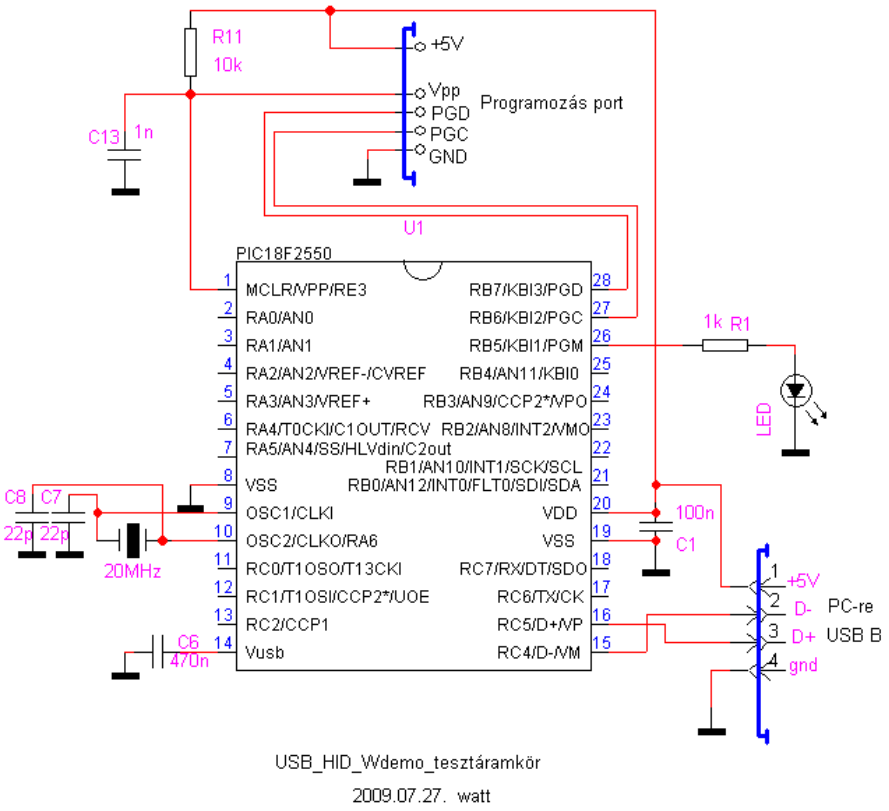
USB HID Demo @watt

1. Bevezetés

Ebben a cikkben egy egyszerű kommunikációs kapcsolatot próbálok bemutatni, elsősorban gyakorlati oldalról egy egyszerű hardveren, valamint a PIC(C18) és a PC(VB6) oldali programon keresztül. Nem célom mélyebben belemenni az USB HID szabvány rejtelmeibe, inkább a lényegre, a használatára próbálok koncentrálni.

Miért jó a HID?
A HID meghajtó programjai települnek a PC-re a rendszer (XP) telepítésékor, ezért nincs szükség külön driver telepítésére. Ez akkor jöhet jól, ha nincs lehetőségünk rendszergazdai jogokkal telepíteni egy gépre. Akkor is jó, ha elfelejtettük magunkkal cipelni azt az adathordozót, amin a driver van (Persze a felhasználói programra azért szükség van!). Itt tényleg teljesülhet a nagy reklám szöveg, hogy csatlakoztatni és használni.
A CDC firmware kezelése PIC oldalon egyforma nehézséget jelent a gyári megoldásoknak köszönhetően. A PC oldal egyszerűbb, mivel csak egy szabványos soros port megnyitását követeli meg, amit minden visual fejlesztő környezet támogat.
A HID ezzel szemben PC oldalon nagyobb gondot jelent, viszont van megoldás.
Visual C++ 2005 Express környezetre a gyári (Microchip) Demo ajánlás tartalmaz szabad kódot, ami közvetlen API beállításokkal operál, ebből ki lehet indulni.
VisualBasic 6-hoz pedig a neten lehet dll fájlokat találni, amikhez van kódgenerátor, vagy egyéb kódrészletek, melyekkel fel lehet építeni a programot. Én az Easy HID szabadon használható kódkörnyezetét választottam, amely az mcHID.dll állományt használja. Ez tartalmazza azokat az API leírásokat, amelyek a HID szabványt kiszolgálják. A VB6 programnak csak deklarálnia kell a függvényeket egy modulban, majd meghívni azokat. Az olvasó rutin eseményvezérelt, azaz akkor hívódik meg, ha USB csomag érkezik, ami a HID esetében 64Bájt méretű. A csomagok száma másodpercenként 1000, ami így maximálisan 64 KBájt/sec adatforgalmat enged meg. A PIC demo programmal 100 csomagot küldök el, ami 6,4 KBájt/sec sebességet jelent, de majd látható, hogy a csomag nagy része nem is lesz értelmezve, mert csak néhány bájt hordoz infót a PC program működéséhez.
Természetesen ebből kiindulva tovább lehet lépni, és olyan sebességet és annyi bájtot használni, amennyi szükséges az egyéni feladathoz. Végeredményben ez a fő célom ezzel a leírással.

2. A Hardver

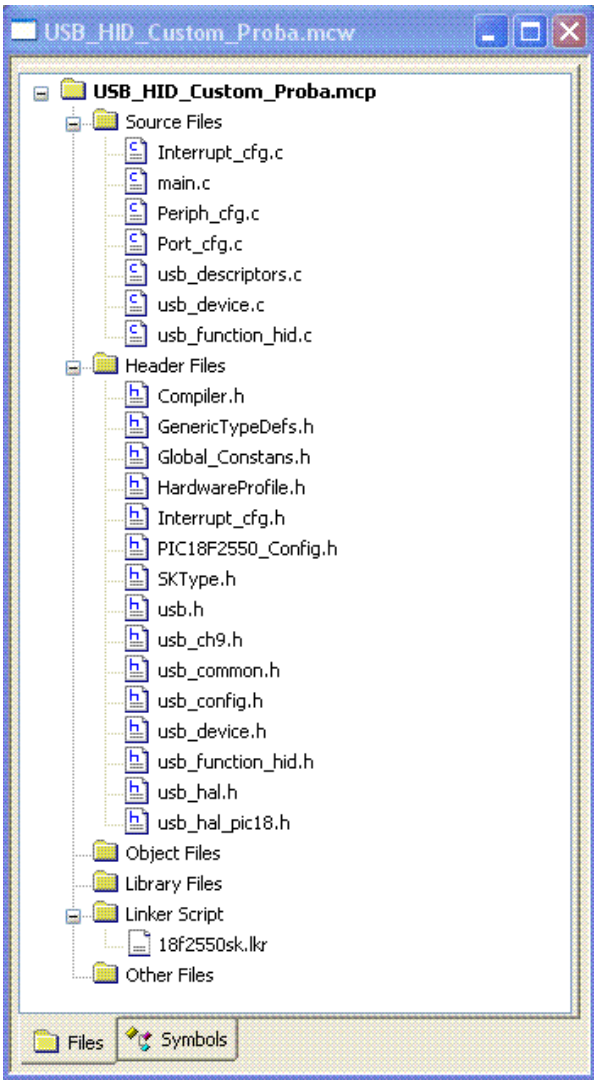


Az áramkört fel lehet építeni egy próbanyákra, vagy egy meglévő áramkör "átalakításával" is meg lehet oldani, hiszen csupán egy láb van felhasználva egy LED meghajtására. A LED-et másik lábra is át lehet kötni, ha szükséges, csak a programban kell módosítani egy sort. A C6 lehet 330nF-is. A programozási port (ICSP) kiosztása saját választás (nem egyezik pl. a PICKit2-évl (csak a sorrend!)), mindenki ügyeljen, hogy a programozójának megfelelő kiosztást alakítsa ki. Programozáskor válasszuk le az USB csatlakozót, annak ellenére, hogy túl sok bajt nem okozna, de jobb a békesség. A programozást követően, ha a fenti áramkörtől nem térünk el, a programozót nem szükséges leválasztani.

3. A PIC program

A program C18 nyelven íródott, mivel a gyári firmware ezt favorizálja (sajnos Assembler kódra nem fejlesztenek). A felépítése moduláris, ezért én is igyekeztem ezt követni. Kiválasztottam azokat a forrásokat, amik szükségesek a kód futtatásához, de még ezekből is töröltem azokat a sorokat, amikre itt nincs szükség. Így talán jobban áttekinthető, és ha tovább akarjuk fejleszteni a forrást, akkor jobban látható, mi az, amiben a gyári eltér. Megjegyzem, hogy a demo működőképes, de nem használ hibakezelést, suspend módot, tápfeszültség-figyelést és egyéb más, jelenleg számomra is ismeretlen funkciót.

A PIC C18 program MPLAB project ablaka:



Úgy gondoltam, hogy jobban áttekinthető a program, ha a fő beállításokat külön fájlokba teszem. Így külön meg lehet nyitni, és jobban lehet olvasni őket (jelenleg kevés beállítást tartalmaznak, de egy teljes kész program esetén szépen megtelnek, és hasznosak lesznek).

Ezek a következők:

Interrupt_cfg.c	A megszakítások beállításai
Periph_cfg.c	Belső perifériák beállításai
Port_cfg.c	Portok beállításai

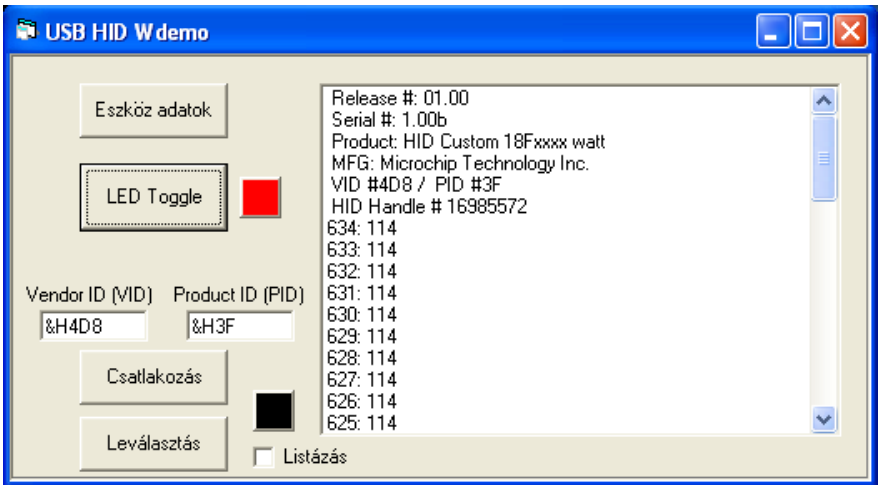
Global_Constans.h	Saját konstansok felsorolása
HardwerProfile.h	Az áramkörrel kapcsolatos azonosítók

Interrupt_cfg.h Interrupt nevek rögzítése
PIC18F2550_Config.h Konfigurációs bitek beállításai
SKType.h Saját típusok deklarálása

18f2550sk.lkr Átalakított Linker fájl.
A linker fájl bootloader nélküli megoldásra lett állítva.

A többi fájl a gyári forrás része, ahol csak a main.c-ben történt lényegi változtatás.
A többi állományban csak olyan változtatás történt, ami leegyszerűsítette a 18F2550-re a forrást (egy csomó #if eltávolítása), így talán jobban megérthető.

4. A PC program



A program a következő feladatokat mutatja be:
- VendorID (VID) és ProductID (PID) adatok bevitelét támogatja. Ezt a programban be lehet állítani alapértelmezettként is, de menet közben is lehet változtatni, ügyelve a szintaktikára, mert hibakezelés nincs beépítve. Ezek az értékek azonosítják az USB eszközt.
- Csatlakozás. Meg lehet oldani, hogy a csatlakozás a program indulásakor történjen meg, ehhez csak a Form_Load() szubrutinba kell írni a megfelelő sorokat. Itt fontosabbnak tartottam a csatlakoztatás és a leválasztás bemutatását menet közben.
- Leválasztás.
- Eszköz adatok. A képen (számozás nélküli sorokban) látható információkat olvassa ki a PIC-ből. Ezeket az értékeket és szövegeket meg lehet változtatni a PIC programban, erre még kitérek.
- LED Toggle. Egy parancsot küld el a PIC-nek. Ennek hatására a PIC átváltja a LED-et és visszaküldi a LED állapotát jelző kódot, aminek hatására a gomb melletti visszajelző a LED állapotát jelzi.
- Listázás CheckBox. Bepipálva folyamatosan listázza a PIC által 100Hz ütemben elküldött, a LED állapotára vonatkozó kódokat.
- Csatlakozás / Leválasztás gombok melletti visszajelző. Jelzi a PIC által elküldött LED állapotára vonatkozó kódok ütemét.
- Listázó ablak. Megjeleníti a fent említett adatokat, és a csatlakozással / leválasztással kapcsolatos eszköz állapotokat. A 100Hz-es adatáram 999 sor után törli a kijelzést.

5. Letöltések

PIC program: http://www.hobbielektronika.hu/kapcsolasok/files/308/usb_hid_custom_proba_pic.zip

PC program: http://www.hobbielektronika.hu/kapcsolasok/files/308/usb_hid_custom_proba_pc.zip

6. PIC program megnyitása, részletezése

Töltsük le a **PIC program** állományt, és csomagoljuk ki a C:/ meghajtóra.
Indítsuk el az USB_HID_Wdemo.mcw állományt. Ekkor be kell töltődjön az MPLAB megnyitásával a project. Fordítsuk le. Ha hibaüzenet jön, amiben az "c018i.o" fájlt keresi, akkor a Build Options ikonnal (sárga fogaskerekes) állítsuk be a C18 telepítési könyvtár lib könyvtárát (? MCC18lib) a Library Search Path kiválasztása után, ahol jelenleg az E:MPLAB_systemMCC18lib van beállítva. Ehhez természetesen fel kell

telepíteni a C18 fejlesztői környezetet, ha még nem lenne.
Ha megnyílt a project, akkor a fenti project ablak képe kell megjelenjen.
Megnézhetjük a már említett állományokat. Bízom benne, hogy a megjegyzések sokat segítenek!

USB azonosítók:
Nézzük meg az usb_descriptors.c-t. Ebben a fájlban vannak rögzítve az eszköz azonosítói, és a szöveges információs adatok. Ha összehasonlítjuk a gyári fájlokkal, látható, hogy hol és hogyan lehet változtatni. A gyári állományban nincs benne a sorozatszámra vonatkozó bejegyzés (254. sortól), ezt már én fűztem hozzá, és a PC program ezt le is kérdezi.
A 177 és 178. sorokban található a két ID, ami az eszközt azonosítja. Ezt lehet beállítani a PC program szövegmezőiben. Ezeket szabadon meg lehet választani, figyelembe véve, hogy ne ütközzünk más eszközök azonosítószámaival.

main.c:
Itt egy példát láthatunk a megszakítások deklarálására, használatára, és egy módszert az USB megszakításban való független kezelésére, valamint egy egyszerű példát, a kétirányú USB kommunikációra, amit tovább lehet fejleszteni a saját feladatunkhoz

A kezdeti linkelések és egyéb szokásos beállítások után a 74. sortól következnek a külön állományban létrehozott beállító szubrutinok meghívásai. Ezek sorrendje fontos, ha a beállítások logikai lépéseit be szeretnénk tartani. A fordító ugyanabban a sorrendben fogja befordítani ill. végrehajtani, mint ha a main.c-ben lennének létrehozva, de így talán jobban követhetőek a dolgok. Ha valakinek ez nem szimpatikus, átalakítja.

A fő ciklus a 85. sortól indul.
A 95. sortól azonosítjuk a LED átváltására vonatkozó parancskódot, majd beállítjuk a jelzőt, ha megérkezett a kód.
A 106. sorban vizsgáljuk ezt a jelet, és megvizsgáljuk a LED állapotát, és annak megfelelő kódot helyezünk el a pufferben.
A 118. sorban elküldjük a PC felé a jelet, aminek hatására a képernyőn a visszajelző kocka színt vált.
A 123. sortól vizsgáljuk a Timer0 megszakításban beállított 100Hz-es ütemjelzőt és elküldjük a LED állapotát a PC felé, ha él a jelző.

A 309. sortól kezeljük le a Timer0 megszakítását. 6000 megszakítás történik másodpercenként, ahol minden esetben meghívásra kerül az USBDeviceTasks() rutin. Ezzel a megoldással tehermentesítjük a programot a pollingozás (gyári megoldás) terhei alól. Gyakorlatilag bármit végrehajthatunk a fő programszálban, ami megszakítható, miközben az USB kérések kiszolgálása folyamatos marad. Próbáltam valós USB megszakítás figyelésekkel megoldani a lekezelést, de ez egyelőre nem sikerült. Ez viszont biztosan működik, használom, bevált.
Az ütem beállításához néhány asm utasítást (NOP), és a 324. sorban a Timer0 kezdeti értékkel való feltöltését használom. Jó szolgálatot tesz ilyenkor a StopWatch ablak és a szimulátor.
A 330. sortól a PC felé elküldött csomagok ütemét állítom be (kb. 100Hz). Itt csak egy jelzőbit kerül beállításra, a végrehajtás a fő ciklusban történik.

7. PC program megnyitása, részletezése

Töltsük le a **PC program** állományt, és csomagoljuk ki.
Az USBProject.vbp indításával a VB6 fejlesztői környezet elindul (természetesen, ha fel van telepítve). A csomagban van egy lefordított exe állomány is, ami a kipróbálásra jó, de ha magunk hasznára akarjuk fordítani, akkor szükség van a fordítóra.
Az mcHIDInterface.bas modul tartalmazza azokat a deklarációkat, amelyek az mcHID.dll szolgáltatásait hívhatják meg.
A MainForm a programunkat tartalmazza.
A minket érintő szubrutin a gyáriak közül a Public Sub OnRead(ByVal pHandle As Long). Ez hívódik meg automatikusan (eseménykezelt), ha USB csomag érkezett. Az itt látható kódokat lehet szerkeszteni a feladatunknak megfelelően. Itt a pufferekbe került adatokat válogatjuk, és azoktól függően listázunk, színeket változtatunk stb.
A Form grafikus ablakot a Project lista MainForm linkjére duplán kattintással lehet felhozni.
Az objektumokra (pl. a gombok) duplán kattintva, azok lekezelő rutinjába ugrik a szerkesztő. Így jól követhetők a hozzá tartozó folyamatok.
Ha egy kattintással kiválasztjuk az objektumot, akkor a tulajdonság ablakban láthatjuk a beállításaikat.

Belátható, hogy a fejlesztői környezet bővebb ismertetése itt nem oldható meg, de véleményem szerint ez egyszerű, és szinte magától értetődő, bátran lehet ismerkedni vele. Rengeteg segédanyag található a neten is, pl. az **MSDN**, a gyártó (Microsoft) teljes adatbázisa, ahol részletesen, példákkal segítenek. Ha ez löketet ad, én is hasonlóan kezdtem...

8. Kapcsolat

Kapcsolat:
Ha kérdésetek vagy véleményetek van, kérem a wattmep@tvn.hu címre írjatok.
A hobbielektronika fórumon is feltehetitek a kérdéseket:

PIC - USB - PC projekt: http://www.hobbielektronika.hu/forum/topic_1896_0_ASC.html

Kérem, privátot ne írjatok szakmai kérdéssel!

Jó programozást!

2009.07.28. watt

www.wattmep.tvn.hu

PDF-et összeállította: 2010.08.29. zenetom