Beágyazott és Ambiens Rendszerek

3. gyakorlat tematikája

A gyakorlat során a mikrovezérlő egyik soros perifériáját (UART0) fogjuk használni karakterek küldésére és fogadására. Az ehhez szükséges beállítások egy részét regiszter szinten fogjuk megtenni. Azonban ellentétben az előző gyakorlattal, a regiszter szintű hozzáférést ezúttal a mikrovezérlőhöz kiadott SDK részét képező, C nyelvű fejléc fájlok fogják segíteni. A szükséges beállítások többségét ennél is kényelmesebben fogjuk elvégezni a szintén az SDK részét képező *emlib* segítségével. Ez egy firmware library a mikrovezérlőben található perifériák kezelését elvégző függvényekkel.

A mikrovezérlő UART0 perifériája a debugger áramkört is magában foglaló Board Controlleren keresztül érhető el a számítógép számára, ahol egy virtuális COM portként jelenik meg. Alapértelmezetten azonban az összeköttetés az UART0 periféria és a Board Controller között meg van szakítva. A kapcsolatot engedélyezni a PF7 láb magasba állításával lehet megtenni. Ez tehát egy olyan konfigurációs lépés, amelynek ugyan semmi köze az UART0 periféria beállításaihoz, de mindenképp el kell végeznünk.

1. feladat

Engedélyezzük az összeköttetést a mikrovezérlő UART0 perifériája és a Board Controller között! Ehhez használjuk az `em_device.h` behivatkozásával rendelkezésre álló fejléc fájlokat! Mivel a PF7 láb állítását a GPIO periféria segítségével tudjuk megtenni, először gondoskodnunk kell arról, hogy kapjon órajelet!

Tipp: a feladat elvégzése során használjuk az `em_device.h` behivatkozása után elérhető `CMU` és `GPIO` nevű pointereket – amelyek a CMU ill. GPIO periféria beállításait lehetővé tevő struktúrákra mutatnak – valamint a megfelelő bit maszkot előállító `#define`-okat, amik szintén a fent említett fejléc használatával érhetőek el.

Az elvégzendő lépések tehát:

1. **A GPIO órajel engedélyezése**
    a. **Általában véve a magas frekvenciás periféria órajel (HFPERCLK) engedélyezése**, hisz ez az, amit a GPIO is megkap (reset után ez engedélyezett, így ez a lépés kihagyható). Az engedélyezéshez a CMU periféria HFPERCLKDIV regiszterének HFPERCLKEN bitjét kell egybe billenteni.
    b. **Kifejezetten a GPIO perifériához menő órajel engedélyezése**. Itt a CMU periféria HFPERCLKEN0 regiszterének GPIO bitjét kell egybe állítani.
2. **PF7 magas állapotba állítása**
    a. Ehhez először **kimenetnek kell definiálni az adott lábat**. Itt a GPIO periféria F portjához tartozó MODEL regiszter megfelelő beállítása szükséges. Tipp: mivel a GPIO periféria több portot is kezel (A-F), ezért azon regiszterek, amelyek az adott portok (és nem általában véve az egész GPIO periféria) kezeléséhez szükségesek, a GPIO nevű pointer által mutatott struktúra P nevű tömbjében találhatóak. A tömb egyes elemei szintén struktúrák, melyek az adott port beállításához szükséges regiszterek elérését teszik lehetővé. Mivel C-ben számmal, 0-val kezdve indexelünk, ezért az A-F portokhoz a 0-5 indexek tartoznak. Továbbá a két lehetséges MODEx regiszter közül (MODEL és MODEH) azért a MODEL kell, mert az tartalmazza a 0-7 lábak beállításait (míg MODEH a 8-15 lábak konfigurálásáért felelős).

b. **Ezt követően gondoskodnunk kell PF7 magas értékbe állításáról**. Ezt a DOUT és a DOUSET regiszterekkel is megtehetjük (ez utóbbi hatékonyabb kódot eredményez).

2. feladat

Az UART0 periféria konfigurálása 115200 Baud jelváltási sebességre, valamint 8N1 keret formátumra (azaz 8 adatbit, nincs paritás, 1 stop bit). A használt lábak pedig PE0 (TX, adás) és PE1 (RX, vétel), mivel az STK3700 kártyán ezen lábak vannak behuzalozva.

Ennek a feladatnak az elvégzése során az *emlib* által biztosított függvényeket fogjuk használni. Mivel a feladat megoldásához az UART0 mellett a CMU és a GPIO periféria megfelelő beállítása is szükséges, ezért az *emlib* három modulja fog kelleni nekünk: USART[1], CMU és GPIO. A három modulhoz tartozik három forrás és három fejléc fájl: `em_<modul neve kisbetűvel>.(c|h)`.

Tipp: a fejléc fájlokat a megfelelő `#include` sorok beszúrásával tudjuk használatba venni, míg a forrás fájlokat hozzá kell adni a projekthez. Ez utóbbit a legegyszerűbben úgy tudjuk megtenni, hogy a projektben már létező, `emlib/em_system.c` fájlon jobb klikk hatására megjelenő menüből a „Browse Files Here” elemet válasszuk. Feltéve, hogy a projektet nem a „Copy contents” opció kiválasztásával hoztuk létre, a felhasznált könyvtárak forrás fájljaira csak linkek mutatnak a projektből, és ebből következően a fenti menü elemre kattintva megjelenő fájl böngésző az *emlib* fájlok helyére mutat. Innen egyszerű Drag & Drop művelettel tudjuk behúzni a projektbe a szükséges fájlokat. A felugró dialógus ablakban a fájl linkelését válasszuk (a `STUDIO_SDK_LOC` környezeti változóhoz képest).

Az elvégzendő lépések:

1. **Először is órajelet kell adni az UART0 perifériának**. Ehhez a `CMU_ClockEnable()` függvényt használhatjuk (az engedélyezendő órajel megadására előre definiált enum értékek állnak rendelkezésre).
2. **Ezt követően be kell konfigurálni a kért üzemmódot**.
    a. **Ehhez létre kell hozni egy `USART_InitAsync_TypeDef` típusú struktúrát a konfigurációs paraméterek eltárolására.**
    b. **Majd fel kell tölteni a struktúra elemeit a megfelelő értékekkel:**
        i. **`baudrate`**: megadása számmal történhet
        ii. **`refFreq`**: 0 (a baudrate előállításához az UART0 alapértelmezett, HFPERCLK órajelének frekvenciáját tekintjük referenciának)
        iii. **`databits`**: megadásához a megfelelő enum értéket használjuk és ne számot, az nem lesz jó!
        iv. **`parity`**: válasszuk a megfelelő enum értéket
        v. **`stopbits`**: válasszuk a megfelelő enum értéket
        vi. **`oversampling`**[2]: válasszuk a 16-szoros túlmintavételezéshez tartozó enum értéket

[1] Alapvetően kétféle egyszerű soros kommunikációs protokoll létezik: UART (Universal Asynchronous Receiver / Transmitter) és USRT (Universal Synchronous Receiver / Transmitter). A kettő között az a különbség, hogy míg az utóbbi használ dedikált órajelet is, az előbbi nem. Mivel ezt leszámítva azonos a működésük, ezért a mikrovezérlőkben sokszor olyan perifériákat találunk, amelyek beállítástól függően a két mód bármelyikén tudnak működni, és így USART névvel illetik őket. Mivel az esetek többségében nem szükséges órajel, ezért gyakran találunk a mikrovezérlőkben csak UART módon működő perifériákat is. Az általunk használt eszközben van ilyen is és olyan is. Mivel a kezelésük nagyon hasonló, ezért az *emlib* csak egy modult biztosít a rendelkezésünkre em_usart néven, és ez hivatott kezelni mindkét fajta perifériát.

[2] Az USART vevő áramkörök egy bitből általában több mintát is vesznek (alapértelmezetten 16-ot).

vii. **`mvdis`**[3]: legyen false (nem szeretnénk a többségi szavazást kikapcsolni)
viii. **`prsRxEnable`**: false (nem szeretnénk használni a PRS[4] rendszert)
ix. **`prsRxCh`**: legyen 0 (jóllehet az értéke don't care, ha `prsRxEnable` false)
x. **`enable`**: válasszuk azt az enum értéket, ami mind a vevő, mind az adó oldali áramkört engedélyezi

c. Miután kitöltöttük a konfigurációs struktúrát, az `USART_InitAsync()` függvény segítségével **inicializáljuk az UART0 perifériát a fenti struktúra segítségével**! (Megjegyzés: most didaktikai célokból minden konfigurációs opciót kitöltöttünk. Általános esetben azonban a konfigurációs struktúrák kitöltésének megkönnyítésére léteznek alapértelmezett értékekkel feltöltött `#define`-ok – a mi esetünkben most a `USART_INITASYNC_DEFAULT` használható erre a célra. A struktúrát a létrehozásakor ezzel érdemes inicializálni, és utána csak azon paraméterek átírása szükséges, ahol máshogy kívánjuk beállítani a perifériát, mint az alapértelmezett eset.)

3. A kért üzemmód beállítása után **az UART0 periféria jeleit ki kell vezetni a használni kívánt lábakra**.
   a. Első lépésként a GPIO periféria segítségével **be kell állítani PE0 lábat kimenetnek** (1 kezdeti értékkel, mivel ez az UART kommunikáció nyugalmi állapota), **valamint PE1 lábat bemenetnek**. Erre azért van szükség, mert a lábak üzemmódjának beállításáért akkor is a GPIO periféria felel, ha amúgy nem ő használja azokat. A felhasználandó függvény a `GPIO_PinModeSet()`. A portot a megfelelő enum segítségével lehet kiválasztani, míg a láb megadása egyszerű számmal történhet. A több lehetséges ki- és bemeneti üzemmód közül kimenetnek a push-pull, míg bemenetnek az egyszerű input módot válasszuk (a megfelelő enum segítségével). (A kimeneti regiszterbe írandó érték kimenet esetén a láb kezdeti értéke lesz, így PE0-nál ezt 1-nek válasszuk! Bemenet esetén a kimeneti regiszterbe írható érték azt szabályozza, hogy legyen-e bekapcsolva egy opcionális szűrő áramkör, ami igyekszik megszabadulni az apróbb tüskéktől, vagy ne. Itt most nincs rá szükség, így a kimeneti regiszter értéke PE1 esetén legyen 0.)
   b. Miután a majdan használni kívánt lábakat beállítottuk, **meg kell kérni az UART0 perifériát, hogy legyen szíves ezeket használni** (több lehetséges konfiguráció is beállítható ugyanis az RX, TX jelek fizikai lábakhoz rendelésére). Sajnos erre jelenleg nincs *emlib* függvény, így kénytelenek vagyunk regiszter szintű hozzáféréssel megoldani. A feladat elvégzéséhez az UART0 periféria ROUTE regiszterének megfelelő beállítása szükséges. Az általunk használni kívánt lábakhoz a „Location 1" kivezetési mód tartozik. Így ezt válasszuk ki a megfelelő `#define` segítségével. Ügyeljünk arra, hogy a megfelelő „Location" kiválasztása önmagában nem elég, ugyanis azt is meg kell mondani, hogy mely jeleket szeretnénk kivezetni (természetesen mind a kettőt). Így a ROUTE regiszterben még az RX és a TX jelekhez tartozó engedélyező biteket is be kell billenteni.

3. feladat

Küldjünk ki egy karaktert az UART0 perifériára! Ehhez az `USART_Tx()` függvény használható.

[3] Az egy bitből vett minták közül általában a középső hármat szokták alapul venni, és köztük többségi szavazás dönt, ha nem lenne egyértelmű az érték.
[4] A PRS (Peripheral Reflex System) a Gecko mikrovezérlők egy speciális adatátviteli csatornája, ahol ez egyes perifériák egymás között, autonóm módon, a CPU közbeavatkozása nélkül tudnak kommunikálni.

Tipp: ahhoz, hogy az elküldött karaktert megjelenítsük, szükségünk van egy ún. terminál programra, ami a soros porton vett karaktereket kiírja a képernyőre (és a begépelt karaktereket pedig). Ehhez először is szükség van arra, hogy tudjuk, melyik számú soros port tartozik a Gecko kártyához. Ezt Windows alatt a Device Manager segítségével tudjuk megtenni (a „Ports (COM & LPT)” alatt keressük a „JLink CDC UART PORT” bejegyzést). Egy lehetséges terminál program a PuTTY. Ezt állítsuk be a megfelelő soros port figyelésére (ugyanazon kommunikációs paraméterek mellett, mint ahogy az UART0 perifériát is beállítottuk)!

4. feladat

Fogadjunk folytonosan karaktereket az UART0 perifériától, majd küldjük is vissza azokat!
Karakterek vételére az `USART_Rx()` függvény használható.