KIEGÉSZÍTŐ DRIVER AZ STK3700 LCD KIJELZŐJÉN TALÁLHATÓ KARAKTEREK SZEGMENSEINEK EGYEDI VEZÉRLÉSÉRE

LEÍRÁS

A gyári "segmentlcd" driver kiegészítése, hogy a Gecko kártya LCD kijelzőjén a szegmenseket egyesével is lehessen vezérelni. A szegmensenkénti vezérlés a kijelző alsó, hét karakterből álló (karakterenként 14 szegmenst tartalmazó), valamint a kijelző felső, négy darab hétszegmenses digit kijelzésére alkalmas részeire működik.

FÁJLOK

`segmentlcd_individual.(c|h)`

A driver kiegészítés implementációs fájljai.

`main.c`

Demó alkalmazás a driver kiegészítés használatára.

`segmentlcd_individual_test__with_linked_libraries.zip`

Demó projekt a driver kiegészítés használatára. Tartalmazza a driver kiegészítés implementációs fájljait és a demó alkalmazást. A többi forráskód (gyári könyvtárak) C fájljai linkelve vannak a projektben. A gyári könyvtárak H fájljait pedig – a projektben beállított elérési útvonalaknak megfelelően – a Simplicity Studio telepítési könyvtáraiban keresi a fordító. Az esetek többségében így hozunk létre projektet. Előnye, hogy kisebb, hátránya, hogy csak a fizikailag létező fájlokat nézegetve nem látszik, hogy más fájlokat is használ a projekt.

`segmentlcd_individual_test__with_copied_libraries.zip`

Demó projekt a driver kiegészítés használatára. Tartalmazza a driver kiegészítés implementációs fájljait és a demó alkalmazást. A többi forráskód (gyári könyvtárak) C fájljai is megtalálhatóak a projektben. A gyári könyvtárak H fájljai szintén a projektben vannak (`_inc` végű mappákban). Az esetek többségében nem így hozunk létre projektet. Előnye, hogy egyértelműen látszódik, hogy mely fájlokat használ, hátránya, hogy több helyet foglal, és plusz mappák vannak benne.

Naszály Gábor <naszaly@mit.bme.hu> 2019. 10. 03.

HASZNÁLAT

1. ADJUK HOZZÁ A DRIVER KIEGÉSZÍTÉS FÁJLJAIT A PROJEKTHEZ

Másoljátok be őket az `[src]` könyvtárba (a `main.c` mellé). Ezt egyszerű drag-and-droppal is meg lehet oldani. Javaslom a behúzott fájlokat ténylegesen a projektbe másolni (Copy files).

Megjegyzés:

A `segmentlcd_individual.c` azt feltételezi, hogy vele egy könyvtárban van a fejléc fájl. Ha máshova teszitek, akkor az is működhet, de ebben az esetben nektek kell arról gondoskodni, hogy a projekt include path-ai közé felvegyétek azt a könyvtárat, ahová a fejlécet másoltátok.

2. A GYÁRI DRIVER HOZZÁADÁSA A PROJEKTHEZ

A driver kiegészítés nem helyettesíti az eredeti "segmentlcd" drivert, csak kiegészíti azt. Így például az LCD inicializálására is az a `SegmentLCD_Init()` használatos, ami az eredeti driverben van. A gyári drivert tehát hozzá kell adni a projekthez:

- Csináljunk egy mappát a projektünk alá, pl.: `[drivers]`
- Másoljuk be az eredeti driver C forrását (`segmentlcd.c`). Ezt nem triviális megtalálni. Talán a leggyorsabb megoldás az, ha a legegyszerűbb projektekben is jelen levő "`emlib/em_system.c`" fájlon egy jobb klikket nyomtok, majd "Browse Files Here". Ez elnavigál Titeket az `emlib` telepítési könyvtárába. A driverek nem itt vannak, de innen már relatíve nincsenek messze. A mutatott könyvtár struktúra vége most az alábbi:
  `.../platform/emlib/src`
  Ezt navigáljátok át úgy, hogy ez legyen:
  `.../hardware/kit/common/drivers`
  Itt már megtaláljátok a "`segmentlcd.c`" fájlt. Ezt egyszerű drag-and-droppal is be lehet húzni a projektetek `[drivers]` mappájába. Javaslom a behúzott fájlokat hivatkozás útján hozzáadni (Link to files), és azon belül is az SDK telepítési helyéhez képest relatíve linkelni (Create link locations relative to: STUDIO_SDK_LOC).

Megjegyzés:

A header fájlt nem kell bemásolnunk, ugyanis a projekt include path-a alatt (Includes) fel van sorolva a `[.../hardware/kit/common/drivers]` könyvtár.

3. A SZÜKSÉGES EMLIB FÁJLOK HOZZÁADÁSA

A gyári segmentlcd driver csak magának a kártyán lévő LCD kijelzőnek a kezelését végzi el (pl. képes karaktereket kiírni rá, speciális szimbólumokat felvillantani rajta), de mindezt nem teljesen önállóan végzi, hanem felhasználja a mikrovezérlőben található, LCD panelek vezérlésére tervezett áramkört. A mikrovezérlőben található ilyen-olyan áramkörök (perifériák) kezelésére az `emlib` hivatott. A gyári `segmentlcd` driver helyes működéséhez tehát szükségünk van az LCD vezérlő periféria kezelését megvalósító `emlib` forrás fájlra (`em_lcd.c`). Ahhoz, hogy az LCD vezérlőt tudjuk használni, órajelet is kell adni neki, ezért a `segmentlcd` számára szükség van a CMU (Clock Management Unit) kezelésére kiadott `emlib` fájlra is (`em_cmu.c`).

Hasonlóan ahhoz, ahogy a `segmentlcd` drivert adtuk hozzá a projekthez, itt is a leggyorsabb, ha a legegyszerűbb projektekben is jelen levő "`emlib/em_system.c`" fájlon egy jobb klikket nyomtok, majd "Browse Files Here". Ez elnavigál Titeket az `emlib` telepítési könyvtárába. Innen tehát az alábbi két fájlt kellene a projekthez adni (praktikusan az `[emlib]` mappába):

- em_lcd.c
- em_cmu.c

Hasonlóan a `segmentlcd` driver hozzáadásához, itt is linkelve javaslom őket a projektbe tenni.

4. A DRIVER KIEGÉSZÍTÉS HASZNÁLATA AZ ALKALMAZÁSBÓL

Amelyik forrás fájlból használni szeretnénk a szegmensenkénti vezérlést, tegyük meg az alábbiakat

- Hivatkozzuk be mind "`segmentlcd.h`" mind "`segmentlcd_individual.h`" fejléc fájlokat!
- Hozzunk létre egy változót a használni kívánt LCD rész szegmens adatainak eltárolására! Ezek tömbök az alábbi típusokkal:
  `SegmentLCD_LowerCharSegments_TypeDef`
  `SegmentLCD_UpperCharSegments_TypeDef`
  Az alábbi méretekkel:
  `SEGMENT_LCD_NUM_OF_LOWER_CHARS` (=7)
  `SEGMENT_LCD_NUM_OF_UPPER_CHARS` (=4)
- Inicializáljuk a kijelzőt → "`SegmentLCD_Init()`"
- A fenti változó(k)ba írjuk be a kijelzésnek megfelelő (lásd alább) értéket, majd adjuk át paraméterül egy függvény hívás során "`SegmentLCD_LowerSegments()`" vagy "`SegmentLCD_UpperSegments()`" függvényeknek. A kijelző frissítését ezen függvények végzik el a kért értékeknek megfelelően.

A szegmensek adatait tartalmazó tömb elemek a felső (csak digitek megjelenítésére alkalmas) kijelzőn 8 bites egészek, az alsó (alfanumerikus karakterek megjelenítésére alkalmas) kijelzőn 16 bites egészek.

A felső kijelző a 8 bitből 7-et használ (hét szegmensből áll egy digit), míg az alsó a 16 bitből 14-et használ (mivel ott pedig 14 szegmens alkot egy karaktert).

Ezen tömb elemek "`union`" segítségével lettek definiálva. Ebben van egy "`raw`" mező, amely a teljes 8 (vagy 16) bites értéket jelöli. De vannak "`a`".."`g`" (illetve "`a`".."`q`") tartományban ún. bit field-ek, melyek egy bitesek, és egy szegmenshez tartoznak. Eset függő, hogy mikor melyik megoldást egyszerűbb használni.

A SZEGMENSEK BITEKHEZ TÖRTÉNŐ HOZZÁRENDELÉSE

A szegmensek elnevezése a Gecko kártya kapcsolási rajzában lévő LCD ábrának felel meg.

A FELSŐ KIJELZŐ SOR

```
 --- 0 (a) ---
|             |
|5 (f)        |1 (b)
|             |
 --- 6 (g) ---
|             |
|4 (e)        |2 (c)
|             |
 --- 3 (d) ---
```

AZ ALSÓ KIJELZŐ SOR

```
 --------- 0,a --------
|     \7,h  |8,j  /    |
|5,f   \    |    /9,k  |1,b
|       \   |   /      |
 --- 6,g --   -- 10,m --
|      /    |    \11,n |
|4,e  /13,q |12,p \    |2,c
|    /      |      \   |
 --------- 3,d --------
```