Mikrokontroller II. mérési jegyzőkönyv

Mérést végezte:
Csutak Balázs, Farkas Viktória

Mérés helye és ideje: ITK 320. terem, 2016.04.25.

1. A mérés célja:

Ismerkedés a mikrokontroller nyújtotta lehetőségekkel. Perifériák értékeinek vizsgálata. Joystick és gomb kipróbálása, ciklus és elágazás használata, események kezelése.

2. Felhasznált eszközök:

- MSP430F169 Texas-alapú mikrokontroller
- IAR Embedded Workbench programcsomag

3. A mérés menete:

Összekötöttük a mikrokontrollert a számítógéppel, átmásoltuk a megadott könyvtárat a Dokumentumok mappába, majd megnyitottuk a projektet. A kódban megkerestük a szerkesztésre kijelölt részt, és ide rendre beírtuk a feladatok megoldásához szükséges néhány sornyi programot. A környezet debugger funkcióját használva lépésenként végignéztük a program futását, figyelve az egyes regiszterek értékét a jobboldali ablakban. A gombot és a joystick-et használva kipróbáltuk (és javítottuk) a megírt programokat.

4. Feladatok megoldása:

4.1. Feltételes vezérlés átadás. Joystick kezelés I.

A kapott példaprogram megoldotta a feladatot. A mérés során ennek működését

```
minta:
        bic.b #STAT2 ; kikapcsoljuk a LED-et
        bic.b #STAT

tanulmányoztuk.
        bit.b #BUTTON   ; Ha nincs lenyomva,
        jc minta        ; ugrunk az elejere
        bis.b #STAT2    ; Led kigyujtasa
        jmp minta
```

Az első két parancs nullázza a P2OUT és a P2DIR 1-es bitjét, ezzel kikapcsolva a LED-et. A harmadik sorban a bit.b parancs a carry bitbe tölti a joystick (P2IN) gombjának helyzetét. Gombnyomás esetén ennek értéke 0, különben 1. A negyedik sor jc parancsa visszaugrik a program elejére, ha a carry 1 (azaz ha nem volt gombnyomás, akkor újravizsgáljuk a gombot - várunk a felhasználóra, hogy nyomja meg). Ekkor a LED nem gyúl ki. A jc parancs miatt a program csak akkor jut el az ötödik a sorba, ha az előző feltétel hamis volt (vagyis le van nyomva a gomb). Ekkor a LED kigyullad. Az utolsó parancs ismét a program elejére ugrik, várva, hogy a felhasználó elengedje/ismét megnyomha a gombot.

Megjegyzés: a gomb folyamatos nyomvatartása mellett a LED nem folyamatosan világít, hanem emberi szemmel követhetetlenül gyorsan villog - hiszen minden ciklusiteráció kikapcsolja (majd ha a gomb még mindig nyomva van visszakapcsolja) azt.

4.2. Szám növelése gombnyomásra

Ennek a pontnak a megoldásához csak minimális változtatásokra volt szükség.

```
        mov.w #1, R04
        mov.w R04, R12   ; Kiirjuk a szamot
        call #hexdraw
eleje:
        bic.b #STAT2     ; kikapcsoljuk a LED-et
        bic.b #STAT

        bit.b #BUTTON    ; ha nincs gomnyomas, kezdjuk elorol
        jc eleje

        bis.b #STAT2     ; kigyujtjuk a LED-et
        inc R04 ; Megnoveljuk a szamot
        mov.w R04, R12   ; Kiirjuk a szamot
        call #hexdraw

        jmp eleje
```

A fenti kód elvégzi a feladatot, bár a gomb nyomvatartása esetén a megjelenített szám folyamatosan (és elég gyorsan) növekszik. Ennek javításához megakasztjuk a ciklus futását, amíg a gombot el nem engedjük:

```
        mov.w #1, R04
eleje:
        ...
        call #hexdraw

        lenyomva:
        bit.b #BUTTON    ; Amig le van nyomva,
        jnc lenyomva     ; nem csinalunk semmit

        jmp eleje
```

4.3. Szám kiírása decimális formában

A szám decimális kiíratásához a BCD (Binary Coded Decimal) formátumot, és az ezzel működő DADD műveletet használtuk. A fenti program egyetlen sorát írtuk át: az INC R4-et DADD #1,R4 -re cseréltük. A BCD formátum lényege, hogy hexadecimális formában tárol decimális adatot, a DADD összeadás pedig ezt szem előtt tartja. (Tehát pl. a 9+1 művelet eredménye nem 0xA lesz, hanem 0x10 (ami már valójában 16-ot jelent, de a felhasználó számára a megjelenített érték tízes számrendszerbelinek látszik)). Kód:

```
        mov.w #0x0000, R04
        mov.w R04, R12
        call #hexdraw
eleje:
        bic.b #STAT2
        bic.b #STAT

        bit.b #BUTTON
        jc eleje

        bis.b #STAT2
        dadd.w #1, R04   ; Decimalisan megnoveljuk
        mov.w R04, R12
        call #hexdraw
        jmp eleje
```

4.4. Karakter mozgatása joystickel. Joystick kezelés II.

Megjegyzés: a LEFT és a RIGHT iránya fel van cserélve, ennek javítására az x koordináta változtatása fordítva történik: jobbra lépéskor csökken, balra lépéskor pedig nő. Mivel csak a jegyzőkönyv megírása után vettük ezt észre, az eljárások nevét már nem cseréltük ki.

```
        mov.b #2,R4      ; karakter y koordinataja
        mov.b #7,R5      ; karakter x koordinataja
        mov.b #0x4F,R6   ; karakterkod: '0'

        call #kiir ; kirajzoljuk a karaktert
eleje:
        call #torol      ; kitoroljuk az elozo karaktert

        bit.b #LEFT          ; joystick ellenorzese
        jnc move_left
        bit.b #RIGHT
        jnc move_right
        bit.b #UP
        jnc move_up
        bit.b #DOWN
        jnc move_down

        jmp eleje        ; ha nincs gombnyomas, kezdjuk elorol
```

```
move_left:
        inc R5 ;
        call #kiir

        pause1:
          bit.b #LEFT
        jnc pause1

        jmp eleje
move_right:
        dec R5
        call #kiir

        pause2:
          bit.b #RIGHT
        jnc pause2

        jmp eleje
move_up:
        dec R4
        call #kiir

        pause3:
          bit.b #UP
        jnc pause3

        jmp eleje
move_down:
        inc R4
        call #kiir

        pause4:
          bit.b #DOWN
        jnc pause4

    jmp eleje
    ; Main fuggveny vege
    ret

kiir:   ; kiiras a kepernyore - sajat fuggveny
        mov.b R6,R14
        mov.b R4,R13
        mov.b R5,R12
        call #LCDChrXY
        call #LCDUpdate
        ret
torol:  ; elozo karakter torlese - nem hiv update-t
        mov.b #0x20,R14
```

```
mov.b  R4,R13
mov.b  R5,R12
call   #LCDChrXY
ret
```