Mikrokontroller II. jegyzőkönyv

Mérést végző személyek: Hadnagy Levente, Ekart Csaba
Mérés helye: PPKE ITK 420-as mérőlabor
Méréshez felhasznált eszközök: MSP430F169 mikrokontroller, IAR Embedded Workbench program

A mérési feladatok megoldása

1. **feladat**: *A feladat során meg kell valósítani a Mikrokontroller panelen található joystick kezelését, tehát megnyomására ki kell gyújtani a STAT LED-et.*

```
asmmain:

minta:
        bic.b #STAT2 ; Kimenetre állítja a LED lábat
        bic.b #STAT ; A LED-re „0”-ás logikai szintet tesz
        bit.b #BUTTON ; A Nyomógomb értékét a Carry flag-be tölti
        jc minta ; ujrakezdi, ha nincs megnyomva a gomb
        bis.b #STAT2 ; Bemenetre állítja a LED lábat
        jmp minta ; Ismét az egész
    ret
```

A feladat leírásában szereplő példaprogramot a megfelelő helyre bemásoltuk a forráskódot, amely probléma nélkül le is futott. Ezután már csak a működését kellett megvizsgálnunk. A #define sorok alapján arra jutottunk, hogy az első két parancs nullázza a P2OUT, illetve a P2DIR 1-es bitjét, tehát kikapcsolja a LED-ünket. A következő sorokban a carry bitbe betöltjük a nyomógomb értéket. A jc parancs miatt a program visszaugrik az elejére, amennyiben a gomb nincs lenyomva. Emiatt az 5. sorba csak akkor jutunk el, ha a gombot lenyomjuk, és ekkor az LED-et felkapcsolja a 6. sorban található parancs, majd visszaugrunk a ciklus elejére.

2. **feladat**: Minden egyes gombnyomás után növeljen egy számot, és azt írja ki a grafikus kijelzőre!

```
asmmain:
     mov.b #1,R4 ;Kezdőérték beállítása

minta:  mov R4,R12 ;A szám kiíratása
        call #hexdraw

        bic.b #STAT2 ;Kimenetre allitja a LED lábat
        bic.b #STAT ;a LED-re „0”-ás logikai szintet tesz
        bit.b #BUTTON ; A Nyomógomb értékét a Carry flag-be tölti
        jc minta ;ujrakezdi, ha nincs megnyomva a gomb

        bis.b #STAT2 ;Bemenetre állítja a LED lábat
        inc.b R4 ;Az érték növelése
        mov R4,R12 ;A szám kiíratása
       call #hexdraw

break:  bit.b #BUTTON
        jnc break

        jmp minta
      ret
```

A feladat megoldása során definiáltuk az R4 regiszterben tárolt értéket (ezt fogjuk növelni), és a leírás szerint az R12-es regisztert, illetve a #hexdraw függvényt használtuk ennek a kiíratásához. Az R4 kezdeti értékét egyre állítottuk, még a minta cikluson kívül, hogy tudjuk növelni az értékét. Az előző feladat alapján a megfelelő helyre elhelyeztük az inc.b parancsot, amely eggyel növelte az R4-es regiszter értékét a gombnyomásokkor. A változó értékét betöltöttük az R12-es regiszterbe, melyet a #hexdraw függvény a vártak szerint a képernyő balfelső sarkában megjelenített.

3. **feladat**: Minden egyes gombnyomás után növeljen egy számot, és azt írja ki a grafikus kijelzőre de 10-es számrendszerbeli alakban.

```
asmmain:
     mov.b #1,R4 ;Kezdőérték beállítása

minta:  mov R4,R12 ;A szám kiíratása
        call #hexdraw

        bic.b #STAT2 ;Kimenetre allitja a LED lábat
        bic.b #STAT ;a LED-re „0”-ás logikai szintet tesz
        bit.b #BUTTON ; A Nyomógomb értékét a Carry flag-be tölti
        jc minta ;ujrakezdi, ha nincs megnyomva a gomb

        bis.b #STAT2 ;Bemenetre állítja a LED lábat
        dadd #1, R4 ;Az érték növelése decimálisan
        mov R4,R12 ;A szám kiíratása
       call #hexdraw

break:  bit.b #BUTTON
        jnc break

        jmp minta
      ret
```

A szám 10-es számrendszerbeli alakjához a BCD formátumot, illetve a DADD műveletet használtuk fel, így az előző programnak csupán egy során kellet változtatni. A dadd művelet gyakorlatilag úgy végzi el az összeadást, hogy a megjelenő eredmény számunkra 10es számrendszerbelinek tűnjön. Numerikusabban megoldva úgy lehetne elképzelni, ahogy papíron is átváltanánk a számokat, tehát a számokat mindig elosztjuk tízzel, és a maradékokból lesznek az átalakított szám számjegyei, a hozzájuk tartozó hatványértékekkel. Ha ezeket összeadjuk megkapjuk a szám decimális értéket. Ez a megoldás egyrészt szemléletesebb, másrészt a DADD függvény a 99-nél korlátba ütközik, mivel csak „látszólag” végzi el a műveleteket.

4. **feladat**: **feladat**: A joystick irányításérzékelésének megfelelően kell a kijelzőn elhelyezni egy-egy teli karaktert.

Sajnos a feladatot nem sikerült megfelelően implementálni, hogy az a vártak szerint működőképeslegyen, a félkész kódot azonban beillesztem a gondolatmenetünkkel együtt.

```
asmmain:
     mov.b #2,R4 ; y kezdő koordináta
     mov.b #7,R5 ; x kezdő koordináta
     mov.b #0x4F,R6 ; kirajzolandó karakter kódja „o”
     call #kiir ; karakter kirajzolása
minta:  mov.b R6,R14 ; értékadások
        mov.b R4,R13
        mov.b R5,R12

        call #LCDChrXY ; szükséges függvények meghívása
        call #LCDUpdate

        call #torles

        bit.b #LEFT
          jnc bal
        bit.b #RIGHT
          jnc jobb
        bit.b #UP
          jnc felfele
        bit.b #DOWN
```

```
            jnc lefele
          jmp minta

        ;   az   irányok   megadása,   a   koordináta   regiszterek
értékváltoztatásával
          bal:
            inc R5
            jmp minta
          jobb:
            dec R5
            jmp minta
          felfele:
            dec R4
            jmp minta
          lefele:
            inc R4
            jmp minta
          torles:
            mov.b #0x20,R14 ; előző karakter törlése
    ret
```

Az alkalmazott módszerünk figyeli, hogy a joystick melyik irányba mozdul, és ennek függvényében csökkent, illetve növeli az R4 és az R5 regiszterben tárolt értékeket, melyek az x, illetve az y „koordinátái” az általunk kirajzolandó karakternek. A kezdeti értéket, azért 2-re, illetve 7-re állítottuk, hogy körülbelül középről induljon a program. Az R4 és R5 regiszterek értékeivel egyenlővé tettük az R12 és R13 regiszterek értékeit, amelyeket használni fog a feladat leírásban bemutatott #LCDCharXY függvény. Az R14es regiszter értékét egyenlővé tettük az R6-tal, a függvény ezt a karaktert fogja kirajzolni a képernyőre (ASCII). Ahhoz, hogy „mozgást” imitáljunk, ahhoz az előző karaktert mindig törölnünk kell, ezt egy külön függvény létrehozásával tettük meg, amely a karaktert felülírtja egy szóközzel.