Budapesti Műszaki és Gazdaságtudományi Egyetem

Beágyazott és ambiens rendszerek

Házi feladat
MitMót programozás

Jegyzőkönyv

Készítette: Sipos- Takáts Bence László

2010. 04. 22.

Feladat

- Ismerje meg az Atmel AVR ATmega128 mikrokontrollert [1]!
- Ismerje meg a mitmót kijelzőpaneljét [2]!
- Ismerje meg az AVR Studio 4 kezelését! (A szoftvert megtalálja az MIT tanszék mitmót laboratóriumának fejlesztőgépein, vagy letöltheti az Atmel honlapjáról [3].)
- Ismerje meg a WinAVR kezelését! (A szoftvert megtalálja az MIT tanszék mitmót laboratóriumának fejlesztőgépein, vagy letöltheti a Sourcefoge honlapjáról [1].)
- Írjon egy egyszerű C programot a mitmót rendszerre, majd töltse le a hardverre és tesztelje. A program a következőket végezze:

  EEPROM: Az SW1 és 3 nyomógombokkal növeljen ill. csökkentsen egy T1 SW számlálót 0 és 99 között, és jelezze ki az aktuális értéket a 7 szegmenses kijelzőn. Az SW2 nyomógomb megnyomására mentse el ezt az értéket az EEPROM-ba. A nyomógomb újabb megnyomására egy következő memória címre mentse az EEPROM-ban az adatot. Az EEPROM tartalmát lehessen visszaolvasni a következőképpen:
  DIP1 kapcsoló on állása: EEPROM visszaolvasás
  DIP2 on: EEPROM mem. relatív cím kijelzése
  DIP2 off: EEPROM tartalom kijelzése
  SW1 ill. SW3-mal a kijelzendő memória cím növelése ill. csökkentése
  DIP1 kapcsoló off állása: T1 SW számláló kijelzése.
  Ellenőrizze a működést úgy is, hogy lekapcsolja a tápfeszültséget, majd visszaolvastatja az utoljára elmentett értéket.

- A működő szoftver-t személyesen mutassa be konzulensének, demonstrálja a működését, és magyarázza el a szoftver felépítését.

[1] Atmel AVR ATmega128 processzor adatlapja:
http://www.atmel.com/dyn/resources/prod_documents/2467S.pdf,
http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf
[2] mitmót kijelző panelje: http://bri.mit.bme.hu/?l=mitmot

Mitmót dokumentációk:

- AVR kontroller kártya felhasználói útmutatója
- DPY kijelző kártya felhasználói útmutatója
- AVR kontroller kártya C API útmutató
- DPY kijelző kártya C API útmutató

Feladatok beadása

Formai követelmények:

- A kinyomtatva beadandó dokumentáció első oldala az innen letölthető, névre szóló, kinyomtatott és aláírt feladatkiírás.
- Elektronikusan beadandó dokumentáció: a feladatkiírásnak megfelelő; önállóan megírt szoftver.

Konzultáció:

- A konzultáció nem kötelező, de legkésőbb a feladat beadásakor a konzulenssel találkozni és beszélgetni kell. Konzultáció előre egyeztetett időpontban.

Feladat beadás:

- A feladatot személyesen kell beadni a konzulensnek, legkésőbb a beadási határidő napján 12.00 óráig.

Eredmények:

- A feladatok eredménye alapvetően két állapotú lehet: ***elfogadott*** és ***nem elfogadott***.
- Az eredményeket a weben láthatjátok majd.

A feladatokat a tárgyfelelős által kijelölt társammal, meg nem engedett segítség igénybevétele nélkül oldottam meg:

.....
aláírás

Email cím (olvashatóan): .....

A feladat megoldásához a következő globális változókra volt szükségünk:

```
volatile unsigned int         num=0.0, numcount=0;
int                           elementnum=0;
unsigned char                 data;
volatile unsigned char        dip1, dip2;
unsigned char                 but1, but2, but3;
unsigned char                 but_fw1=1, but_fw2=1, but_fw3=1, but_now1=1,
                              but_now2=1, but_now3=1;
```

A num változó a kijelző aktuális értékét tárolja, míg a numcount a num változó értékének helyét az EEPROM-ban. A volatile módosítóra azért van szükség, hogy a fordító a program futása közben semmilyen optimalizálást ne hajtson végre ezeken a változókon. Az elementnum az elmentett elemek számát tárolja, erre majd a kiolvasásnál lesz szükségünk. A dip1 és dip2 a kapcsolók értékét, míg a but1-3 változók a nyomógombok aktuális állapotát tárolják. A but_fw1-3 és a but_now1-3 –ra a lefutó él vizsgálatánál lesz majd szükség. Ezekkel vizsgálhatjuk meg, hogy történt-e változás a nyomógombok állapotában. Azért állítottuk őket 1-re, mert a nyomógombok alap állapotban logikai 1-be vannak húzva.

A main függvény előtt nyolc függvényt valósítottunk meg, melyek a main-ben hívódnak meg. Ezek rendre:

```
int up(unsigned int num, unsigned int x)
 {
      if (num<x)  num++;
      else num=0;
      return num;
}
```

Ennek segítségével egyesével számolhatunk felfelé és az új értékkel térünk vissza. A függvény megvizsgálja, hogy egy x határértéken belül vagyunk-e, ha nem, akkor 0-ra ugrik és onnan folytatja a felfelé számlálást. Ez a feladatunkban annyit jelent, hogy ha elszámoltunk 99-ig akkor a következő szám ismét a 0 lesz.

```
int down(unsigned int num, unsigned int x)
 {
      if (num>0) num--;
      else num=x;
      return num;
}
```

Hasonló módon működik, mint az előző, de ezzel lefelé számolunk. A függvény azt vizsgálja meg, hogy elértünk-e már a 0-hoz, ha igen akkor a következő szám a 99 lesz, és onnan folytatja a leszámolást.

```
void store(unsigned int num)
      {
            while(EECR & (1<<EEWE));            // Előző írás befejezéséig vár
```

```
      EEAR = elementnum;                  // Írási hely beállítása
      EEDR = num;                         // Adat beállítása
      EECR |= (1<<EEMWE);                 // Logikai 1-esek beírása EEMWE-be
      EECR |= (1<<EEWE);                  // Elkezdi írni az EEPROM-ot az
EEWE beállításával
      elementnum++;
}
```

Ez a függvény elmenti a kijelző aktuális értékét az EEPROM-ba és növeli az elementnum értékét.

```
unsigned int read(unsigned int numcount,unsigned int elementnum)
{
      if (elementnum) {
            unsigned int data;

            while(EECR & (1<<EERE));            // Előző olvasási folyamat
befejezéséig vár
            EEAR = numcount;              // Kiolvasási hely beállítása
            EECR |= (1<<EERE);                  // Olvasás elkezdése az EERE
beállításával
            data=EEDR;
            return data;                        // Visszatér az adattal
            }
      else return -1;
}
```

Ez a függvény a megadott helyen lévő értéket olvassa ki az EEPROM-ból. Mivel a memóriában a számozás 0-tól kezdődik, a main függvényben a „kívánt-1” helyen lévő értéket kell megadni a függvénynek. A függvény persze először megvizsgálja, hogy van-e elmentett érték, és ha nincs, -1-el tér vissza.

```
void display(void)
{
      if (num==-1) dpy_trm_s01__7seq_write_3digit(0xFF,0xFF,0xDF);
      else if (dip1 && dip2) dpy_trm_s01__7seq_write_number(numcount,0);
                  //relatív cím kiírása
            else dpy_trm_s01__7seq_write_number(num,0);    //tartalom
kiírása
}
```

A display függvény a hétszegmenses kijelzőt állítja be annak megfelelően, hogy a kapcsolók milyen állásban vannak és ezután kiírat.

```
int elementnum_count(void){               // Megszámlálja, hogy hány 100-nál
kisebb elem található az
         EEPROM-ban
      for(unsigned int i=0;i<100;i++){
            unsigned int read_i=read(i,1);
            if (read_i<100) elementnum++;
      }
      return elementnum;                  // Visszatér az elemszámmal
}
```

Az elementnum_count függvény megszámolja, hogy hány 100-nál kisebb szám van a memóriában. Erre azért van szükség, mivel bekapcsolás után nem tudjuk hány elem van a memóriában. A tapasztalatok azt mutatták, hogy a memória fennmaradó helyei 225 értékkel automatikusan feltöltődnek, így valósítható meg a vizsgálat. A függvényt a inicializálásnál használjuk.

A main fügvény:

A main függvényt inicializálással kezdtük. Töröltük a kijelzőt és az elementnum értékét.

```
dpy_trm_s01__Init();                  // DPY kijelző kártya inicializálása
Timer0_Init();                        // Timer0 inicializálása
SYS_LED_DIR_OUTPUT();                 // Beállítja a rendszer LED vezérléséhez
                                      a kimeneti lábat
SYS_LED_ON();                         // Bekapcsolja a rendszer LED-et
sei();                                // Engedélyezi a megszakításokat
dpy_trm_s01__7seq_clear_dpy();        // Kijelző törlése
elementnum=elementnum_count();        // Elementnum megszámláló fgv meghívása
```

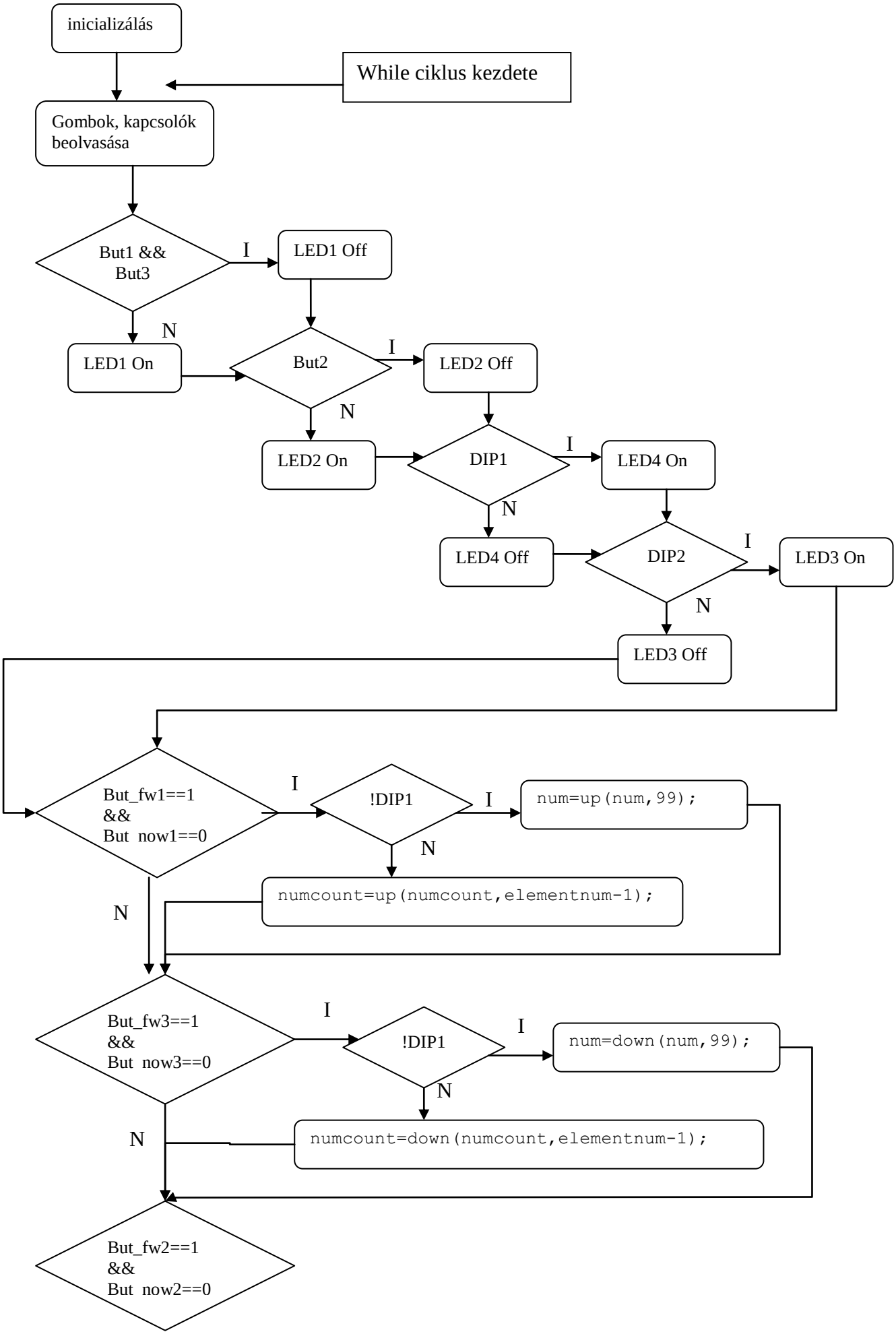
Ezek után egy while ciklusba helyeztük a fő törzsét a programunknak. Miután beolvastuk a kapcsolók és nyomógombok állapotát, beállítottuk, hogyan működjenek a LED-ek. Ez nem volt benne a feladat-specifikációban, de visszajelzést ad programozás és felhasználás közben a helyes illetve helytelen működésről.

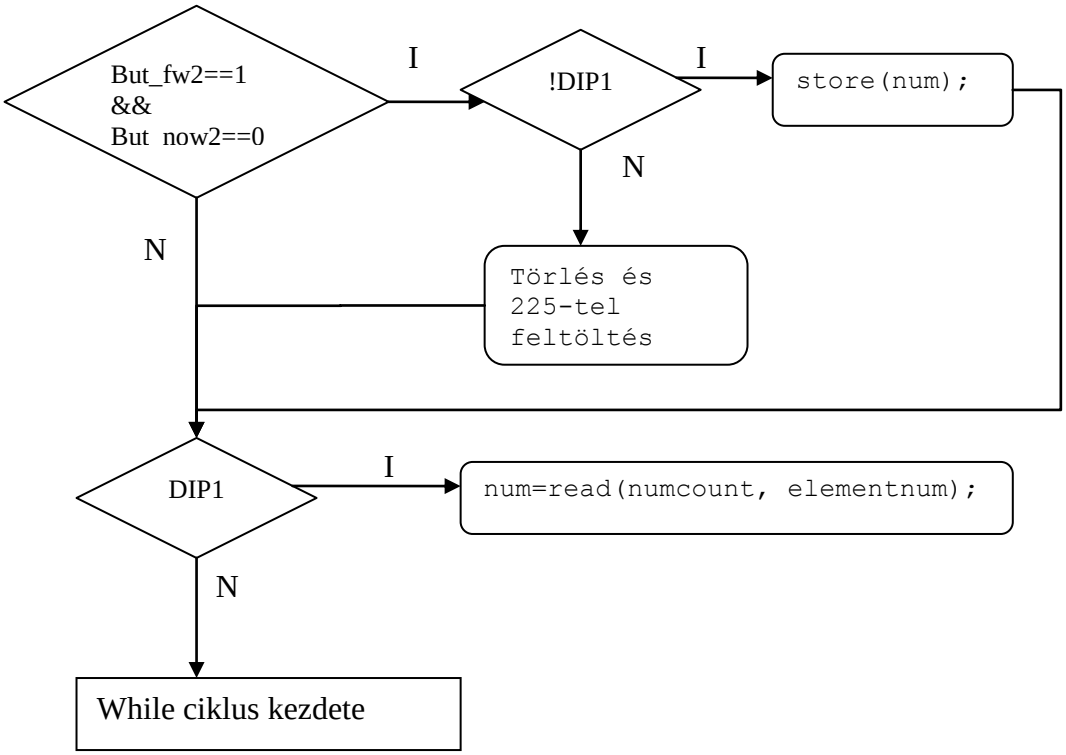
Következő lépésben a nyomógomb előző állapota a but_fw-val, az aktuális állapota pedig a but_now-val lesz egyenlő. Megvizsgáljuk, hogy volt-e változás a nyomógomb állapotában. Ha igen, akkor a dip1 kacsoló állásának megfelelően, vagy a kijelzőt vezéreljük, vagy a memóriát.

```
while(1)
   {
      _delay_ms(50);                              // Pergésmentesítés
      but1=DPY_TRM_S01__BUTTON_1_GET_STATE();     // Gombok, kapcsolók
                                                  beolvasása
      but2=DPY_TRM_S01__BUTTON_2_GET_STATE();
      but3=DPY_TRM_S01__BUTTON_3_GET_STATE();

      dip1=DPY_TRM_S01__SWITCH_1_GET_STATE();
      dip2=DPY_TRM_S01__SWITCH_2_GET_STATE();

      if (but1 && but3) DPY_TRM_S01__LED_1_OFF();
      else DPY_TRM_S01__LED_1_ON();
            // Fel és leszámlálánál LED1 világít
      if (but2) DPY_TRM_S01__LED_2_OFF(); else DPY_TRM_S01__LED_2_ON();
           // Tárolásnál LED2
      if (dip1) DPY_TRM_S01__LED_4_ON(); else DPY_TRM_S01__LED_4_OFF();
           // Memória tartalom kijelzésénél LED4
      if (dip2) DPY_TRM_S01__LED_3_ON(); else DPY_TRM_S01__LED_3_OFF();
           // Relatív cím kijelzésnél LED3

      but_fw1=but_now1;                           // Lefutó él vizsgálat
      but_now1=but1;
      if(but_fw1==1 && but_now1==0) {             // SW1 megnyomásánál
                                                  felszámlálás
            if (!dip1) num=up(num,99);    // Ha dip1 off kijelző vezérlése

            else numcount=up(numcount,elementnum-1);  // Ha dip1 on memória
                                                        vezérlés
            }

      but_fw3=but_now3;
      but_now3=but3;
```

```
      if(but_fw3==1 && but_now3==0) {          // SW3 megnyomásánál
                                               leszámlálás
            if (!dip1) num=down(num,99);       // Ha dip1 off kijelző
                                               vezérlése
            else numcount=down(numcount,elementnum-1);     // Ha dip1 on
                                                           memória vezérlés
            }

      but_fw2=but_now2;                  // SW2 megnyomásánál...
      but_now2=but2;
      if(but_fw2==1 && but_now2==0) {
            if (!dip1) store(num);             // tárolás
            else {                             // vagy törlés
                  elementnum=0;
                  for (int i=0;i<100;i++) store(255);
                  elementnum=0;
            }
      }

      if(dip1) num=read(numcount, elementnum);       // Kiolvasás

   }

}
```

A következő oldalon a program folyamatábrája látható.


inicializálás
While ciklus kezdete
Gombok, kapcsolók beolvasása
But1 && But3
I
LED1 Off
N
LED1 On
But2
I
LED2 Off
N
LED2 On
DIP1
I
LED4 On
N
LED4 Off
DIP2
I
LED3 On
N
LED3 Off
But_fw1==1 && But now1==0
I
!DIP1
I
num=up(num,99);
N
numcount=up(numcount,elementnum-1);
N
But_fw3==1 && But now3==0
I
!DIP1
I
num=down(num,99);
N
numcount=down(numcount,elementnum-1);
N
But_fw2==1 && But now2==0


But_fw2==1
&&
But now2==0
I
!DIP1
I
store(num);
N
N
Törlés és
225-tel
feltöltés
DIP1
I
num=read(numcount, elementnum);
N
While ciklus kezdete