Törekedjen az áttekinthető munkára! Olvashatatlan munkát nem javítunk ki! Tömör érthető válaszokat várunk. Beágyazott rendszerek ZH Rendelkezésre álló idő: 50 perc Felhasználható segédeszköz: -------- Maximális pontszám: 18 pont Elégséges szint: 9 pont

1. Milyen feldolgozóegység (mikrokontroller, FPGA, DSP) segítségével valósítanád meg a következő funkciókat: a) lift kezelői felülete, b) 32-csatornás, 10 MHz-es mintavételi frekvenciájú, szinkron mintavételező adatgyűjtő egység. Egy-egy mondatban magyarázd a választást. 2 pont

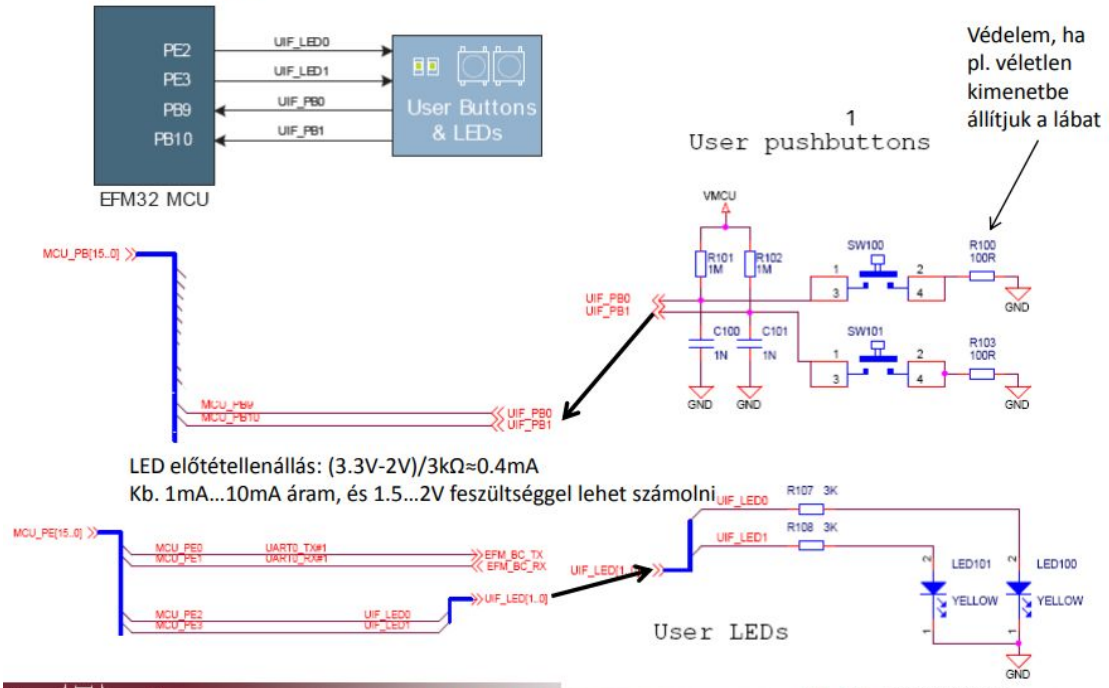
a)
Egy mezei mikrokontroller megfelelő, nem bonyolult a feladat, nem is időkritikus.

b)
A DSP konkrétan ilyesmire lett kitalálva, így “könnyen” elvégezhető a feladat vele, de FPGA-val is megoldható csak, hosszabb a fejlesztési idő, egyébként az is megoldja. Mikrokontroller nem javasolt, túl drága mikrokontroller lenne, meg bonyolult.

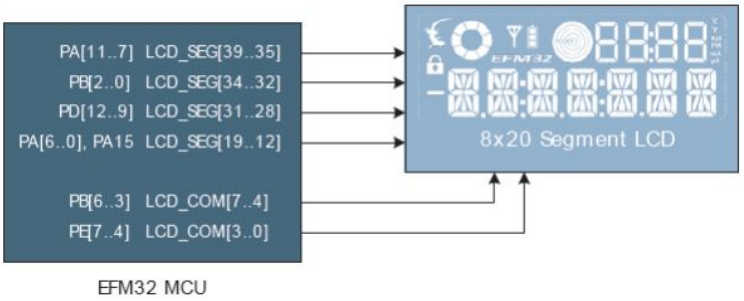
2. Ismertesd az EFM32 Giant Gecko STK kártyán található xxx periféria működési elvét! 4 pont

Dialife:

- Nyomógombok és LED-ek



- LCD kijelző
  - A kontroller rendelkezik beépített vezérlőegységgel, dedikált perifériaként
  - Alafanumerikus és speciális alakzatok



- Fototranzisztorral
  - Fény: „bázisáram”
- Irodai megvilágítás: kb. 400 lux →

  0.1mA →

U_{ki} =22 kΩ * 0.1 mA = 2.2 V

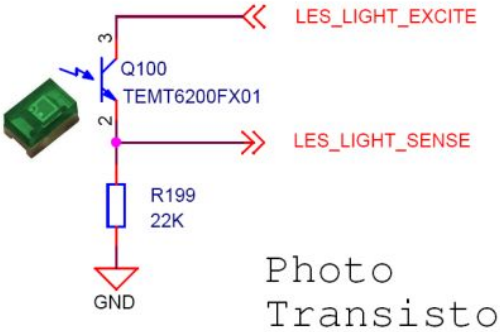
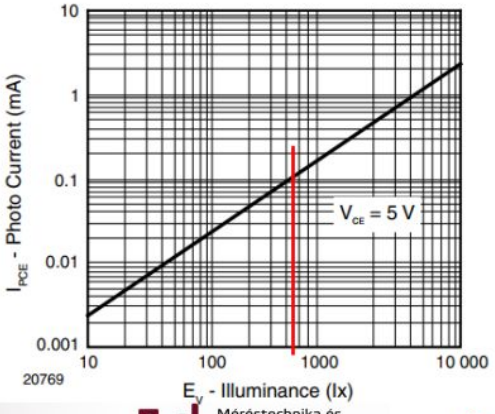
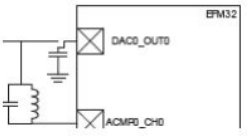


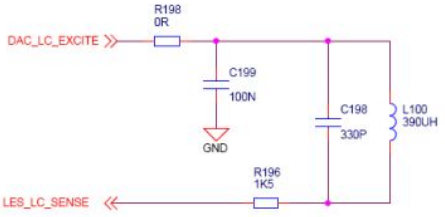
Photo Transistor

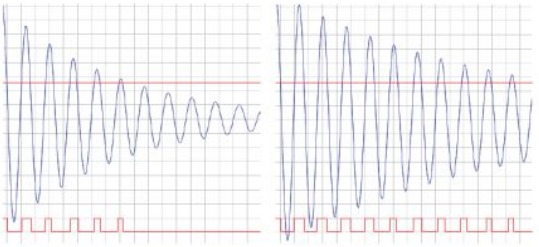


LC fémérzékelő

- Működési alapelv: gerjesztünk egy külső LC kört.
- Mérjük az oszcilláció idejét.
- Minél lassabb a lecsengés, annál nagyobb az LC kör jósági tényezője, tehát annál kisebb az induktivitás vesztesége.
- Ha fémtárgy kerül az induktivitás közelébe, akkor nő az induktivitás vesztesége tehát gyorsabban lecseng az oszcilláció.

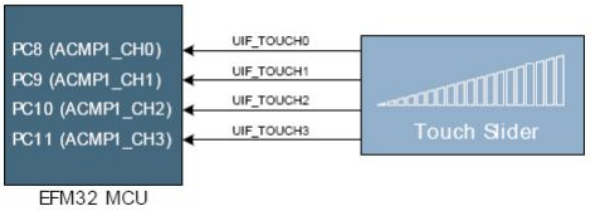


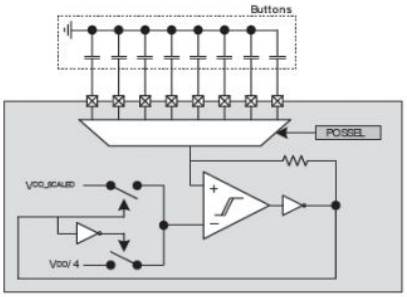




Érintésérzékelő

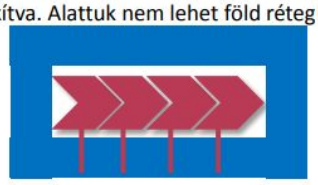
- Touch slider





NYÁK lemezen kialakított RC oszcillátor. A rézfóliához közelítve az ujjat megnő annak kapacitása, így elhangolja a frekvenciát. Minél nagyobb felületen érintkezik az ujj, annál nagyobb a kapacitás → csökken a frekvencia.

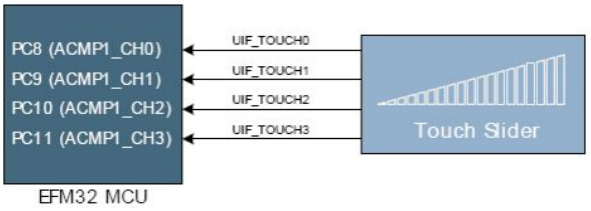
Az érzékelő pontok a NYÁK lemezen vannak kialakítva. Alattuk nem lehet föld réteg!!!



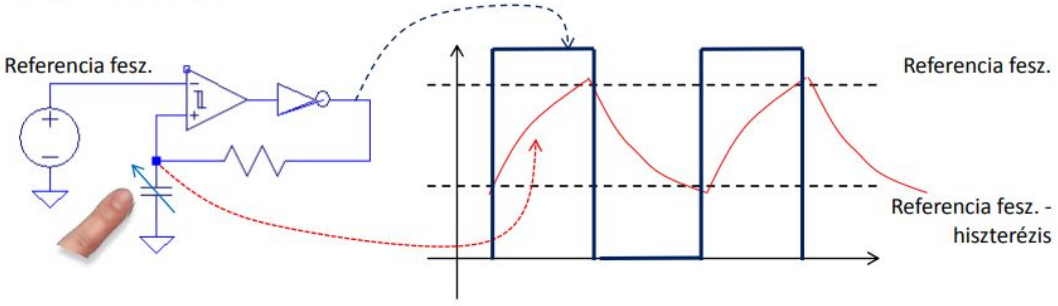
:Ha föld réteg van alul akkor az ujj és vezető közötti erővonalak hatása kicsi

Érintésérzékelő

- Touch slider



Referencia feszültség alatt a komparátor kimenete alacsony→ inverter kimenete magas, tehát a kondenzátor töltődik.
A referenciafeszültséget elérve a komparátor kimenete magas→az invertet kimenete alacsony lesz, így a kondenzátor kisül, amíg az alsó hiszterézis szintet el nem érjük.

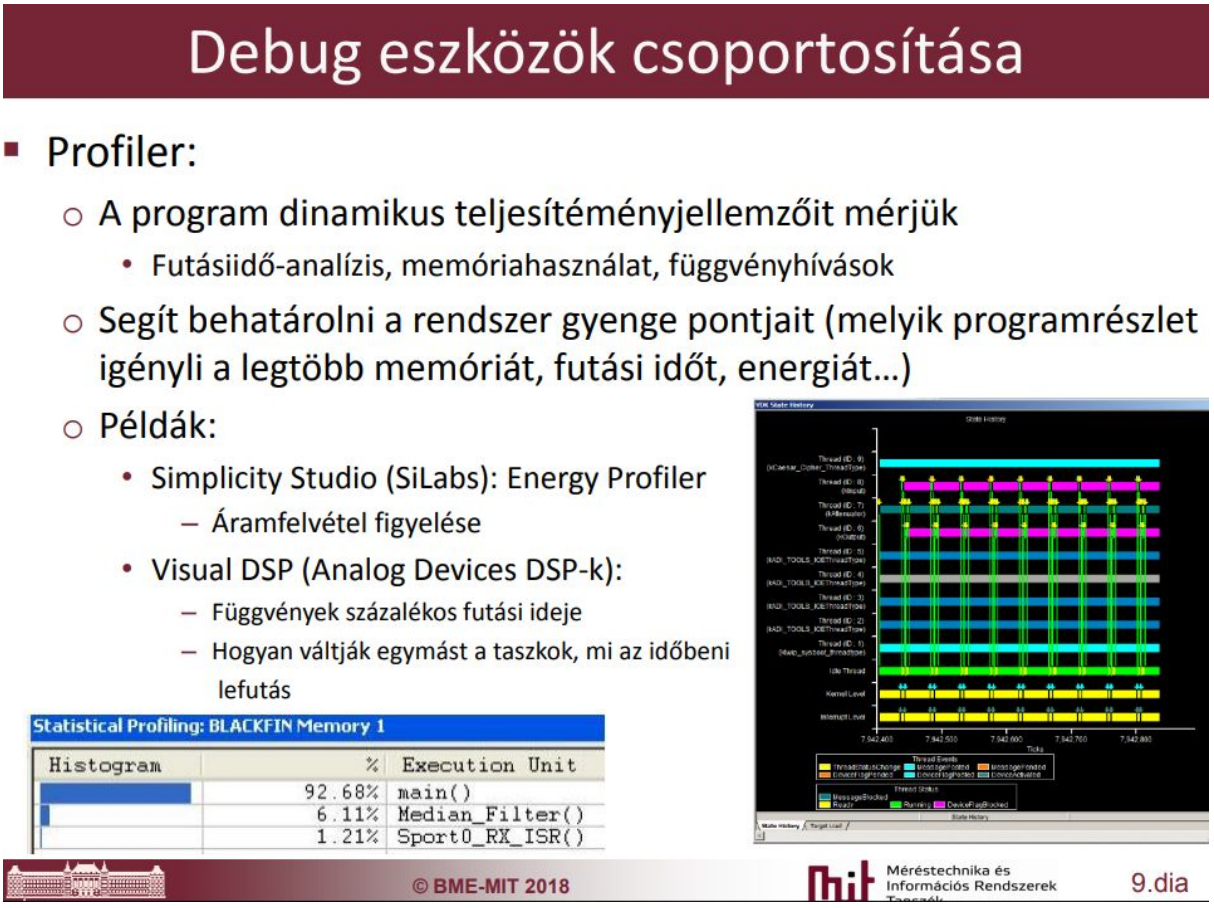


Többi már bonyolult 4 pontért...

3. Egy alkalmazási szintű programban melyik utasítást használnád egy LED bekapcsolására strukturált programszervezés irányelveit szem előtt tartva? Választásodat indokold.
a) SetLED();
b) GPIO->P[5].DOUT |= 1 << LED_BIT; 2 pont

Az a) verziót ugyanis, itt nem kell ismerni magát a hardware-t a program hordozhatóbb lesz, hasonló kártyákon is tud futni.

4. Mi a profiler? 2 pont



5. Ismertesd a makefile-ok szerkezetét! Milyen esetben futnak le az utastások? 2 pont

makefile szerkezete

- A makefile szabályokat tartalmaz.
- Szabályok felépítése (függőségek / dependencies)

cél: előfeltétel(ek)

utasítás(ok) [Tabulátorral kell kezdeni a sort]

Példa

```
main.o: main.c
    gcc -o main.o  main.c
```

A main.o fájl függ a main.c fájltól (abból állítjuk elő). A main.o fájlt a gcc parancs segítségével állítjuk elő.

- Az utasítás(ok) akkor fut(nak) le, ha:
  - Ha a cél fájl még nem létezik.
  - A program ellenőrzi a függőségekben a cél és előfeltétel fájlok dátumát. Csak akkor futnak le az utasítások, ha az előfeltételben adott fájlok frissebbek, mint a cél fájlok.
    - Az időbeliség ellenőrzésével nem fordít le feleslegesen fájlokat → gyorsabb.

6. Miért kell egy megszakítás engedélyezése előtt törölni a hozzá tartozó megszakítás flaget? 2 pont

Lehetséges, hogy reset-után vagy bármikor, bent ragad egy flag, így ha előbb engedélyezzük akkor egyből egy hamis megszakítás jut érvényre, ezért először töröljük a flaget, majd engedélyezünk.

7. Ismertesd rövid pszeudokóddal a súlyozott körforgó programszervezést! 2 pont

Súlyozott körforgó programszervezés

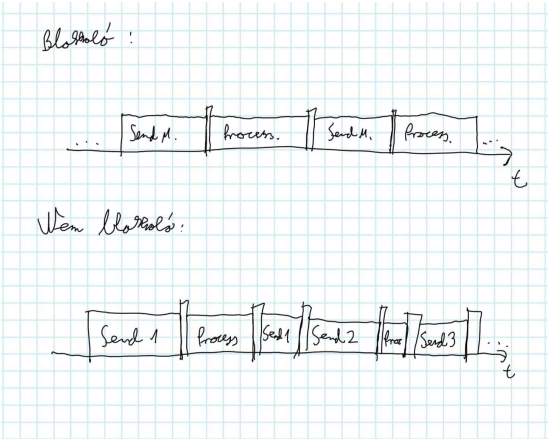
- A feladatokat egymás után ciklikus jelleggel hajtjuk végre, de bizonyos taskokat sűrűbben ellenőrzünk.

```
void main() {
while (TRUE){
        if (erzekelo_allapot==aktiv) {sebesseg_szamol() ;}
        if (erzekelo_allapot==aktiv) {sebesseg_szamol() ;}
        if (gomb1_lenyomva==true) {menuallapot_valtas() ;}
        if (erzekelo_allapot==aktiv) {sebesseg_szamol() ;}
        if (erzekelo_allapot==aktiv) {sebesseg_szamol() ;}
        if (gomb1_lenyomva==true) {menuallapot_valtas() ;}
         if (erzekelo_allapot==aktiv) sebesseg_kijelez();
```

8. Adott a következő kódrészlet:

SendMessage(&dataPacket);
ProcessNewData();

Rajzold le, hogy az adatküldés és ProcessNewData() függvény futása időben hogyan viszonyul egymáshoz, ha a SendMessage függvény blokkoló, illetve nem blokkoló jellegű. 2 pont



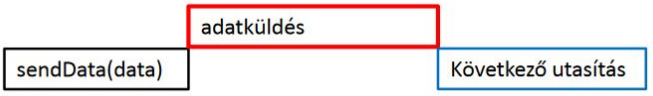
Jobban belegondolva, lehet, hogy a nem blokkoló azt jelenti, hogy a két folyamat futhat párhuzamosan, ha a uP tud ilyet...

Ide ilyesmire gondol???

Könyvtári függvények használata

- Tisztában kell lenni, hogy egy adott függvény:
  - Milyen szinten használja ki a perifériákat,
  - Milyen erőforrást igényel,
  - Igényel-e inicializálást (pl. adatküldés előtt inicializálni kell-e az adott perifériát).
- Blokkoló/nem blokkoló függvény
  - A futás befejezte előtt visszatér-e a függvény.
  - Például üzenetküldés soros porton keresztül:
    - a teljes adatcsomag elküldésekor tér vissza a függvény,
    - vagy átveszi a teljes tömböt, és a háttérben elküldi, a futás joga pedig visszakerül a főprogramhoz.

Blokkoló adatküldés: addig nem tér vissza a függvény, amíg minden adatot el nem küldött:



Nemblokkoló adatküldés: a küldés előkészítése után visszatér a függvény és háttérben megy az adatküldés: