

Mikrokontroller labor, 1. mérés ellenőrző kérdések

1. Milyen típusú objektumokat tartalmazhat egy integrált alkatrészkönyvtár az Altium Designer programban?

Kapcsolási rajz ábra, NYÁK rajzolat (footprint), 3D modell, szimulációs modell.

2. Mi a különbség az component (alkatrész) és a part között? Hogyan kapcsolódnak egymáshoz?

A part a component önálló része, melyet a kapcsolási rajzban külön lehet kezelni, azonban a NYÁK rajzon az egy alkatrészhez tartozó part-ok már együtt jelennek meg.

3. Mire való a port?

A jobb átláthatóság érdekében a hosszú vezetékek helyett az összekötendő alkatrészekhez portokat rakhatunk, valamint a különböző oldalak közötti kapcsolatteremtésre használjuk. Az azonos nevű portok összekötődnek.

4. Mi a busz, és mikor érdemes használni?

Több jelvezeték összefogva, sok egymás mellett futó vezetéket érdemes összefogni, hogy átláthatóbb legyen az ábra.

5. Milyen szimulációs lehetőségeket nyújt az Altium Designer program? Soroljon fel legalább hármat!

Munkaponti analízis (DC), Frekvencia-átvitel (Bode), Időtartománybeli analízis (Tranziens), Hőmérséklet, Zaj.

6. Milyen módon ellenőrizhetjük a kapcsolási rajzot az Altium Designer programban? Hogyan lehet ezt beállítani?

Fordítás közben a beállításoktól függően automatikusan hibaüzeneteket kapunk. Beálíthatjuk például, hogy 2 output pin összeköttetése errort generáljon, míg egy input és egy IO pin összeköttetése legyen elfogadott.

7. Ipari környezetben használt digitális berendezéseknél miért van szükség a bemeneteken speciális fogadóáramkörökre?

A zavarok miatt a szenzorok magasabb feszültséget használnak, ezért szükség van egy illesztő áramkörre. Valamint túlfeszültség-védelem.

8. A mérés során vizsgált 24V-os bemeneti fogadóáramkörben mi a szerepe a Zener diódának?

A bemeneten található Zener dióda a bemenetre kapcsolt fordított polaritású feszültség és túlfeszültség elleni védelmet lát el.

9. Miért használunk a bemeneti fogadóáramkörben hiszterézises komparátort?

A zajos bemenet miatt házardos lenne a sima komparátor kimenete.

10. Mi okozza a bemeneti fokozat aluláteresztő jellegű viselkedését?

A fokozat kapacitása.

Nem volt beugró

Mikrokontroller labor, 2. mérés ellenőrző kérdések

1. Milyen rétegeket használunk a route-oláshoz?

Top Layer és Bottom Layer, esetleg további belső rétegek.

2. Mire használjuk a viákat?

Ez egy kis átmérőjű fémezett furat, amellyel a rétegek közötti elektromos kontaktust hozzuk létre.

3. Mire való a Solder Mask réteg?

A forrasztásgátló maszkot tartalmazza. Ahol van rajta alakzat, oda nem kerül lakkréteg.

4. Mi a polygon pour?

Ez a felület rezezését jelenti, amely egy bizonyos területet borít. GND bekötéséhez használjuk.

5. Mi a footprint?

Az alkatrészek a huzalozáson megjelenő rajzolata, lenyomata.

6. Mik a tervezési szabályok?

A gyártó által elérhető technológia szabja meg, például mekkora a minimális huzalvastagság.

7. Mi található az Overlay (Silkscreen) rétegeken?

Ezen a rétegen a szitázott feliratok találhatók.

8. Hogy zajlik egy SMD ellenállás beültetése?

Forraszpaszta felvitele, alkatrész beültetése, újraömlesztés.

9. Mi a Clearance?

Egy kiöntés és egy eltérő vezeték közötti minimális távolság.

10. Mi indokolja a méretbeli különbséget a pad-eknél a Solder Mask és a Top/Bottom rétegek között?

Hogy véletlenül se kerüljön forrasztásgátló anyag a forrasztási felületre.

Nem volt beugró

Mikrokontroller labor, 3. mérés ellenőrző kérdések

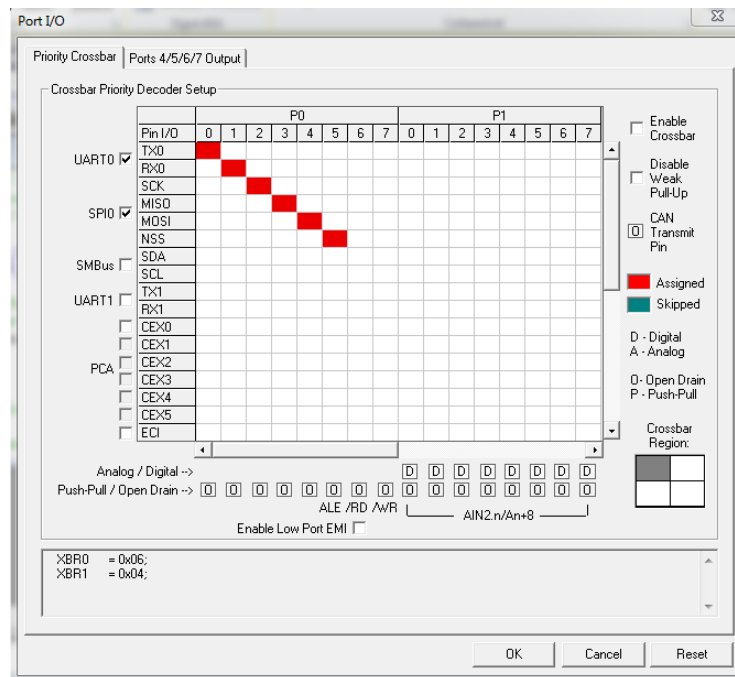
Mire jó a WDT?

A WDT egy olyan biztonsági felügyeleti eszköz, amely tipikusan a programrendszerének egy nem kívánatos végtelen hurokban ragadása ellen védi az alkalmazást. Aktiválása esetén biztosítja, hogy amennyiben a WDT számlálóját nem inicializáljuk adott időnként újra, akkor az elérve a nulla számlálóértéket hardver úton alaphelyzetbe (reset) állítja a processzort.

Mire jó a priority crossbar decoder? Hogy alakul a lábkiosztás a P0-P3 portokon, ha a következő perifériák vannak engedélyezve a crossbarban: UART0, SPI, INT0?

Ennek lényege, hogy egy kapcsolótábla határozza meg, hogy melyik periféria (UART, SPI, I2C, Timer, ADC, stb....), melyik port lábához legyen hozzárendelve.

UART0	TX0	P0.0
	RX0	P0.1
SPI0	SCK	P0.2
	MISO	P0.3
	MOSI	P0.4
	NSS	P0.5
INT0		P0.6



Milyen órajel forrásai lehetnek a C8051F040-es mikrokontrollernek?

Belső programozható oszcillátor 25 MHz-ig vagy külső oszcillátor (Crystal, RC, C, Clock).

Milyen IT prioritási szintjei vannak a vezérlőnek?

2 prioritási szint: high vagy low. Default: low

Mi a különbség a MOV, a MOVC és a MOVX utasítás között?

MOV: normál mozgó utasítás

MOVC: code byte másolása

MOVX: external data

Beugró:

- Watch dog
- MOV, MOVC, MOVX

Mikrokontroller labor, 4. mérés ellenőrző kérdések

- Mi az Analog-to-Digital Converter (ADC)? Mire fogjuk használni a mérésen?

Az ADC egy olyan átalakító egység, amely egy analóg jelből (jelen esetben a potenciométer kimenő feszültségéből) a digitális megfelelőjét állítja elő.

- Honnan tudhatja meg a programunk, hogy az ADC befejezett egy konverziót, az új eredmény rendelkezésünkre áll?

Például beállíthatjuk, hogy az ADC minden konverzió után megszakítást váltson ki.

- Ha szoftveresen készítünk PWM kimenetet egy 8 bites számláló segítségével, akkor mekkora kitöltési tényezőt eredményeznek a következő küszöbértékek: 0, 64, 128, 192, 255?

A LED-hez tartozó kimenet pontosan akkor 1, ha counter>limit.

0: 100% 64: 75% 128: 50% 192: 25% 255: 0%

- Mi a feladata az LCD kijelző Input Enable jelének?

Ez egyfajta óra- (vagy strobe) jel. A kívánt bemenet beállítása előtt 0-ra lehúzzuk, beállítjuk a bemenetet (LCD_DB), majd felemeljük 1-re, amitől a kijelző végrehajtja.

- Mi jelenik meg egy PWM kimeneten, ha 50%-os kitöltési tényezőt használva megállítjuk a mikrokontrollert a debugger segítségével?

Amikor a mikrokontroller megáll, a PWM kimenete beragad 0 vagy 1 értékre.

- Miért kell a 7 szegmenses kijelzőre folyamatosan kiírni a megjeleníteni kívánt értékeket? Az LCD kijelző esetében erre miért nincs szükség?

A kiírt karaktereket az LCD kijelző tárolja, így nem kell folyamatosan frissíteni őket, a 7 szegmens kijelző nem tárol.

- Mi a feladata a priority crossbar decodernek?

Azért, hogy a fel nem használt perifériák ne foglaljanak le feleslegesen lábakat, a priority crossbar decoder a bekapcsolt perifériáknak sorban osztja ki az első 4 port (0-3) lábait, a maradék pedig megmarad általános célú I/O lábnak.

- Egy kész áramkörben miért kell nagyon figyelni, ha az ember bekapcsol egy új perifériát a mikrokontrollerben?

Az új perifériát nem szabad egy használatban lévő lábra kötni.

- Hogyan vizsgálhatjuk meg egy változó értékét akkor, amikor a program futása egy adott sorhoz ér?

Töréspontot helyezünk a sorhoz. Amikor a program megáll a töréspontnál, az i változót (a forráskódban szöveggént) kijelölve és jobb gombbal kattintva felveheti az i változót a megfigyelő ablakba (Add „i” to watch window > #1).

Beugró:

- Miért kell a 7 szegmenses kijelzőre folyamatosan kiírni a megjeleníteni kívánt értékeket? Az LCD kijelző esetében erre miért nincs szükség?

- Mi a hiba?

```
char* str;
```

```
sprintf(str, "valami")
```

Válasz: nincs helyfoglalás az str-nek

Mikrokontroller labor, 5. mérés ellenőrző kérdések

- **Reset után milyen órajelforrást használ a vezérlő?**

Reset után a vezérlő a belső 16MHz-es RC oszcillátorról (HSI) működik.

- **Mekkora a vezérlő maximális rendszerórajele?**

168MHz (a külső kvarc oszcillátor és a PLL-ek megfelelő beállítása segítségével)

- **Milyen üzemmódjai vannak egy GPIO portnak?**

- GPIO_MODE_INPUT // Input Floating Mode
- GPIO_MODE_OUTPUT_PP // Output Push Pull Mode
- GPIO_MODE_OUTPUT_OD // Output Open Drain Mode
- GPIO_MODE_AF_PP // Alternate Function Push Pull Mode
- GPIO_MODE_AF_OD // Alternate Function Open Drain Mode
- GPIO_MODE_ANALOG // Analog Mode
- GPIO_MODE_IT_RISING // External Interrupt Mode with Rising edge trigger detection
- GPIO_MODE_IT_FALLING // External Interrupt Mode with Falling edge trigger detection
- GPIO_MODE_IT_RISING_FALLING // External Interrupt Mode with Rising/Falling edge trigger detection
- GPIO_MODE_EVT_RISING // External Event Mode with Rising edge trigger detection
- GPIO_MODE_EVT_FALLING // External Event Mode with Falling edge trigger detection
- GPIO_MODE_EVT_RISING_FALLING // External Event Mode with Rising/Falling edge trigger detection

- **Mik egy általános periféria inicializálásának lépései? Milyen könyvtári függvények állnak ehhez rendelkezésre?**

A használni kívánt GPIO portnak minden esetben engedélyezni kell az órajelét a `__GPIOx_CLK_ENABLE()` makró segítségével. Reset után minden láb input floating módban van, ezt a `HAL_GPIO_Init()` függvény megfelelő paraméterezésével tudjuk megváltoztatni. Az egyik bemenő paramétere az adott port (GPIOA, GPIOB,...), a másik a beállításokat leíró struktúra (`GPIO_InitTypeDef`).

Minden perifériához külön source és header fájl tartozik (pl.: *stm32f4xx_hal_gpio.c*, *stm32f4xx_hal_gpio.h*), használatához az *stm32f4xx_hal.h* és az *stm32f4xx_hal_cortex.h* header fájlt kell include-olni.

- **Sorolja fel azokat a könyvtári függvényeket, amikkel szinte mindegyik periféria rendelkezik!**

A legtöbb periféria rendelkezik az alábbi függvényekkel:

- `PPP_DeInit(...)` – a PPP regiszterek reset utáni állapotra állítása
- `PPP_Init(...)` – a periféria paramétereinek beállítása
- `PPP_Cmd(ENABLE/DISABLE)` – PPP periféria ki-/bekapcsolása (a periféria órajelét nem állítja)
- `PPP_ITConfig(...)` – a PPP periféria megszakítási forrásainak beállítása
- `PPP_GetFlagStatus(...)` – a periféria flagek olvasása (polling)
- `PPP_ClearFlag(...)` – periféria flagek törlése
- `PPP_ClearITPendingBit(...)` – IRQ flagek törlése

- **Milyen prioritási csoportok vannak és mi a lényegi különbség közöttük?**

Összesen ötféle prioritásmodell közül lehet választani, melyek azt határozzák meg, hogy a rendelkezésre álló négy bitből hány bitet használunk csoport prioritáshoz, és hányat csoporton belüli al-prioritáshoz. A megszakítás vezérlő mindig a magasabb csoport prioritású megszakításokat engedi érvényre jutni, ezek az alacsonyabb csoport prioritású megszakításokat is megszakíthatják. Az al-prioritás azt szabja meg, hogy az egyszerre bekövetkező azonos csoportú megszakításokat milyen sorrendben fogja a vezérlő érvényre juttatni. A megszakítás vezérlőben a kisebb számok jelentik a magasabb prioritásokat.

- **Az STM32 HAL függvénykönyvtárban milyen függvényeket kell megírunk egy megszakítás kezeléséhez?
Milyen főbb feladatokat végeznek ezek a függvények?**

A megszakítás vezérlő konfigurálásához használható függvények:

- HAL_NVIC_SetPriorityGrouping – prioritásmodell beállítása
- HAL_NVIC_SetPriority – egy adott megszakítás prioritásának beállítása
- HAL_NVIC_EnableIRQ – egy adott megszakítás engedélyezése
- HAL_NVIC_DisableIRQ – egy adott megszakítás letiltása

A függvénykönyvtár elvégéz helyettünk minden ellenőrzést, és törli is a megszakításjelző flag-et, nekünk csak a megszakítás alkalmazás specifikus kezelését kell megírunk a callback függvényben.

Beugró: első és utsó kettő kérdés volt

Mikrokontroller labor, 6. mérés ellenőrző kérdések

- **Mire való a Call stack ablak? Hogyan lehet vele kideríteni, hogy oda, ahol a program éppen áll, melyik függvényhívási sorozattal jutott el?**
Ez azt mutatja meg, hogy mely függvényeken keresztül jutottunk oda, ahol most éppen állunk. Így ki lehet deríteni például, hogy pontosan honnan hívódott meg az a függvény, amiben most vagyunk. A nevére duplát kattintva oda is ugorhatunk.
- **Mit jelent az extern kulcsszó egy globális változó esetében?**
Egy külső hivatkozás, egy másik fájlban van mint globális változó és azt szeretnénk használni.
- **C nyelven mi a különbség a & és && operátorok között? Mutasson példát eltérő működésre!**
& az bitművelet. Bitenként alkalmazza az AND műveletet.
&& pedig logikai művelet. 2 logikai változó („boolean”) közötti AND kapcsolat
- **Írjon egy „void mindenKarakterre(const char *szoveg)” függvényt, mely a kapott string minden karakterére meghívja a „void elkuld(const char c)” függvényt, és eközben csakegyszer megy végig a stringen.**

```
void mindenKarakterre(const char * szoveg){
    char * temp=szoveg;
    for(;*temp!='\0';temp++){
        elkuld(*temp);
    }
}
```
- **Mi a baj az alábbi kódrészlettel, mely egy stringbe próbálja beírni azt, hogy „Hello”?**

```
char *szoveg;
strcpy(szoveg, „Hello”);
```

A memóriába nincs hely foglalva neki.
- **Ha végtelen ciklusba kerül a program és a debuggerrel lepause-olja a futását, az pedig egy while(1) végtelen ciklusban áll egy assert hibára utaló nevű függvényben, hogyan tudja megállapítani, hogy a forráskódban pontosan honnan jutott oda a végrehajtás?**
Call stack ablak segítségével

Miért érdemes előre definiált konstansokat használni a konkrét értékek forráskódba bedrótozása helyett?

Később könnyebb módosítani, átláthatóbb.

I2C kommunikáció

- **Jellemezze az I2C protokollt: egyirányú, félduplex, vagy duplex?**

Az I2C kétvezetékes szinkron adatátviteli rendszer, melyet a Philips cég dolgozott ki, integrált áramkörök összekapcsolására, a két vezeték: SCL (órajel) és SDA (adat).

Soros, 8-bit-es, kétirányú adatforgalom, melynek maximális sebessége normál üzemmódban (standard mode) 100 kbit/s, gyors üzemmódban (fast mode) pedig 400 kbit/s.

Kétirányú, de csak egy vezetéken megy az adat: félduplex.

- **Mire való az SCL vezeték?**

SCL (órajel)

- **Mire való az SDA vezeték?**

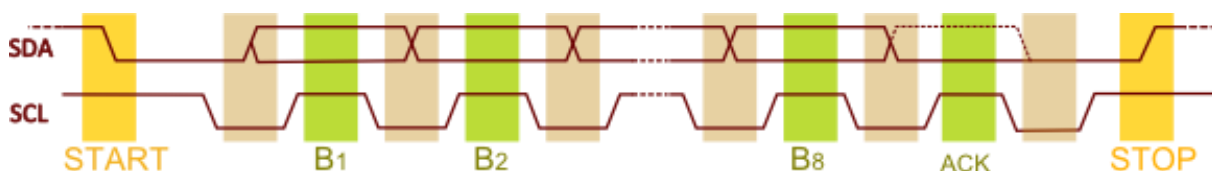
SDA (adat)

- **Mi a master szerepe?**

A kommunikáló eszközök között egyszerű master/slave kapcsolat áll fenn. Mindig a master kezdeményezi és vezérli a kommunikációt (a master szolgáltatja az órajelet) és a master képes adóként és vevőként is üzemelni.

- **Milyen speciális kommunikációs feltételek (más szóhasználat: fázisok) léteznek az I2C protokollban, és ezek mikor teljesülnek? Például hogyan jelezzük egy kommunikáció elejét?**

Az I2C buszon folyó kommunikáció jellemző tulajdonságainak megismeréséhez nézzük meg egy bájtkiküldésének idődiagramja az alábbi ábrán! Adatküldésnél az adatvonal állapota csak az órajel alacsony állapotában változhat. Az órajel magas állapotában az adatvonalnak stabilnak kell maradnia. Ha magas órajelszint mellett változik az adatvonal állapota, az mindig speciális feltételt (**START**, **RESTART** vagy **STOP**) jelent. Az adatküldés kezdetét a **START** feltétel jelzi (magas órajelszint mellett az adatvonal magas állapotból alacsonyra vált). Az adatküldés végét (nem feltétlenül az első bájtnál!) egy **STOP** feltétel jelzi, amelynél magas órajelszint mellett alacsonyról magas szintre vált az adatvonal. A **RESTART** feltétel pedig csak annyiból áll, hogy egy újabb **START** feltételt generálunk, anélkül, hogy a korábbi küldést egy **STOP** jellel lezártuk volna.



- **Minek a rövidítése az I2C?**

Inter-Integrated Circuit = integrált áramkörök közötti

- **Milyen lépéseket kell végrehajtani, hogy az I2C-s hőmérőből kiolvassuk a hőmérsékletet? (Részletesen, mikor mire kell utasítani az I2C perifériát?)**

Ha az inicializálás megvan akkor a regiszterből kell kiolvasni az adatot.

```
HAL_I2C_Mem_Read(&hi2c, deviceAddress, registerAddress, 1, pData, dataSize, i2c_timeout);
```

Beolvasunk a "deviceAddress" eszköz "registerAddress" regiszteréből "dataSize" bájtot a "pData" által mutatott bufferbe.

SPI kommunikáció

- **Hány vezeték szükséges az SPI interface használatához, melyek ezek?**

3+x vezeték, ahol x az eszközök száma. Szinkron soros rendszer, három jelcsatlakozás:

DI (adatbemenet), DO (adatkimenet) és CLK (órjel); valamint engedélyező vezeték minden eszköznek.

- **Jellemezze az SPI protokollt: egyirányú, félduplex, vagy duplex?**

Az SPI (soros periféria illesztő = Serial Peripheral Interface) busz kétirányú (duplex) szinkron soros kommunikációt valósít meg két eszköz között.

Egyszerre kétirányú kommunikáció a két adatvezetéken, full duplex.

- **Mit jelent a MISO és a MOSI?**

MOSI (Master kimenet, Slave bemenet)

MISO (Master bemenet, Slave kimenet)

- **Mi engedélyezi az egyes szolga egységeket?**

A Chip Select engedélyezőjel.

- **Egy üzenet átvitele során hány bitet továbbítunk?**

Az SPI 8 bites

- **Mi az SPI protokoll előnye az I2C-hez képest, mi a hátránya?**

I2C egy több mester és több szolga egységet is kezelni tudó protokoll, míg az SPI-ban van egy kitüntetett eszköz a kommunikációban (master).

Az I2C busz legfőbb előnye a cím alapú kommunikáció. Minden egyes buszra kötött eszköz rendelkezik egy 7 (vagy 10) bites címmel, amellyel azonosítani lehet a hálózaton, tehát az SPI-jal ellentétben nem kell külön engedélyező vezetéket kötni az egyes eszközökhöz.

SPI: full duplex, gyorsabb, de több vezeték (mindenkinek CS)

I2C: félduplex, lassabb, de kevés vezeték kell

- **Miért valósíthatja meg könnyen ugyanaz a függvény az SPI kommunikáció során az írást és az olvasást?**

?

UART kommunikáció

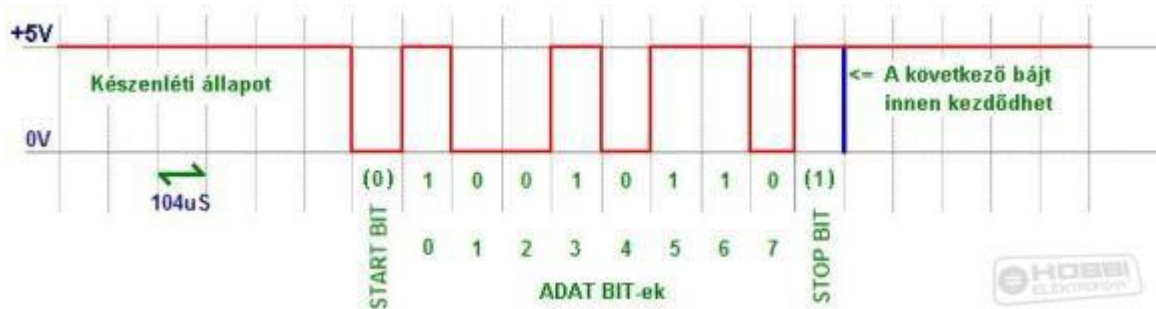
- Milyen részekből épül fel egy karakterkeret?**

Start bit

Adat bitek

Paritásbit is állhat (itt nincs)

Stop bit



- Mi az RX és mi a TX vezeték szerepe?**

RxD : adatfogadás

TxD : adatküldés

- Jellemezze az UART interface-t: egyirányú, félduplex, vagy duplex?**

Duplex.

- Mi a legfontosabb különbség az UART és az előző két (az I2C és az SPI) protokoll között?**

UART nem szinkron, nem küldünk órajelet.

- Hogyan valósul meg a kommunikáció idejére az adó és vevő közötti szinkronizáció?**

Előre meghatározott frekvenciával kell mind az adatokat küldő készüléknek az adatot küldeni, mind a vevő készüléknek a bejövő adatból meghatározott időközönként mintát venni. Mielőtt 2 eszköz kommunikálni tudna egymással, a közöttük történő információcsere alapvető szabályait tisztázni kell. (Baud rate)

- Milyen feltételnek kell teljesülnie a kommunikáció során, hogy ne legyen a kerethiba valószínűsége túl nagy?**

Az adó és vevő közötti szinkronizációhoz nagyon pontos időzítés kell, ezért a busz mindkét oldalán ugyanazt a kommunikációs sebességet kell használni

- Mi jelenik meg az oszcilloszkópon, ha akkor nyomunk „Auto Scale”-t, amikor az UARTon nincsen kommunikáció, és utána egy UART keretet akarunk megnézni? Miért?**

Logikai magas szinten zajt fogunk csak látni. A scope ilyenkor megvizsgálja az éppen érkező adatokat és úgy állítja be a skálázást, hogy a jel bizonyos szempontok szerint legyen „minél jobban látható”

CAN kommunikáció

- **Milyen üzenet típusok vannak?**

- Adathordozó üzenet (Data frame)
- Adatkérő üzenet (Remote frame)
- Hiba üzenet (Error frame)
- Túlsordulás üzenet (Overload frame)

- **Mire való az adathordozó üzenet?**

Egy küldőtől egy vagy több fogadóhoz továbbít adatokat a küldő fél kezdeményezésére

- **Mire jó az adatkérő üzenet?**

Minden csomópont kezdeményezheti a számára szükséges információ elküldését az adatot szolgáltató csomóponttól. Ehhez az igényelt adathordozó üzenettel egyező azonosítójú adatkérő üzenetet kell küldenie.

- **Mi az üzenetek prioritási sorrendje?**

11 bites azonosító mező jelenti a prioritást, minél kisebb a szám, annál magasabb a prioritás.

- **Milyen mezőtípusok vannak egy üzenetben?**

Üzenet kezdete bit

Arbitrációs mező

IDE bit(Az azonosító kiterjesztés bit)

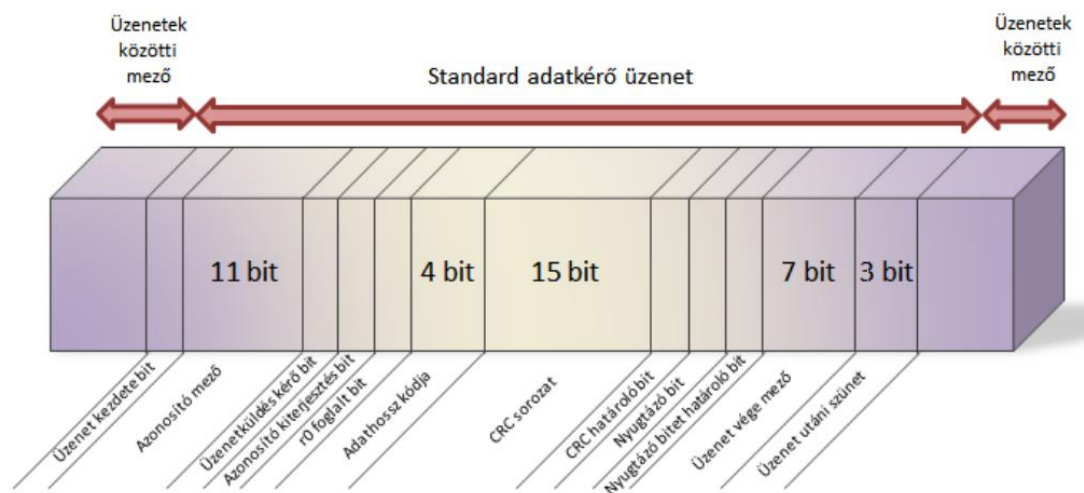
Vezérlési mező

Adatmező

CRC mező

Nyugtázás mező

Üzenet vége mező



Beugró: Nekem a void mindenKarakterre(const char * szoveg) függvény volt

és mi jelenik meg oszcilloszkópon, ha nincs kommunikáció és Auto Scale-t nyomsz

Mikrokontroller labor, 7. mérés ellenőrző kérdések

- Mi az a QEP és mi a feladata?**

Az inkrementális fordulatszámadó jeleiből a relatív pozíciót és forgásirányt meghatározó egységet QEP-nek (Quadratic Encoder Periphery) nevezzük. A hajtásirányítási feladatokhoz tervezett mikrokontrollerek, DSP-k tartalmaznak ilyen perifériát.

- Milyen lehetőségek vannak egy állandómágneses egyenáramú motor fordulatszámának és forgásirányának megváltoztatására?**

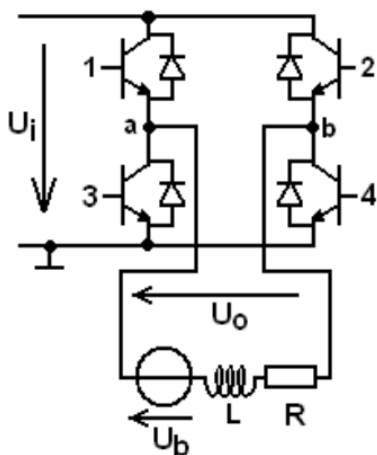
Fordulatszám:
$$n = \frac{1}{2\pi} \frac{U_k - IR}{K_E}$$

A egyenletből láthatjuk, hogy a gép fordulatszámát többféle módon is megváltoztathatjuk.

Egyrészt adott U_k kapocsfeszültség mellett ellenállások beiktatásával (vesztéses megoldás), másrészt magának a kapocsfeszültségnek a változtatásával. A gyakorlatban az utóbbit használják.

A gép forgásirányának megváltoztatása (reverzálás) a kapocsfeszültség polaritásának megváltoztatásával lehetséges.

- Rajzolja fel a hídkapcsolású DC/DC főáramköri kapcsolását! Hogyan határozható meg az átalakító kimenőfeszültsége?**



$$U_o = (2d-1)U_i$$

4. ábra Hídkapcsolású egyenáramú szaggató

- Milyen vezérléseket alkalmazhatunk a hídkapcsolású DC/DC átalakítónál, milyen előnyei vannak ezeknek?**

A mérés során alapvetően kétféle vezérléssel találkozhatunk, az ellenütemű és eltolt vezérléssel.

Az ellenütemű vezérléssel a $-U_i \div U_i$ tartományban a kimenőfeszültség fokozatmentesen változtatható, míg az eltolt vezérlésnél csak 0 és U_i között.

Az eltolt vezérlésnél a maximális áramhullámosság negyedére csökkent az ellenütemű vezérléshez képest.

Az eltolt vezérlés további előnye, hogy a kimenőfeszültség lüktetési frekvenciája kétszerese a kapcsolási frekvenciának, így fele akkora kapcsolási frekvenciával érhető el ugyanaz az áramhullámosság, mint ellenütemű vezérlés esetén.

- Miért kell vezérlési holtidőt biztosítani az egy hídágban lévő kapcsolóelemek között?**

Egy valószínűs kapcsolóelemnek van bekapcsolási és kikapcsolási késleltetése, ezért az egy hídágban lévő félvezetők között vezérlési holtidőt kell biztosítani, különben zárlatot idéznénk elő.

Mikrokontroller labor, 8. mérés ellenőrző kérdések

9.1. A 113, 203, 301, 303, 402, 501, stb. köröknek mi a feladata, hogyan működnek?
??

9.2. Mi a HMI?

Human Machine Interface, ember-gép közötti kapcsolatot valósítja meg, kezelőfelület.

9.3. Mi a Simatic Manager és a WinCC?

Simatic Manager: PLC programok megírása, szimulációja, eszköz felprogramozása

WinCC: HMI-t megvalósító grafikus felület

9.4. Mi a különbség a passzív és az aktív slave között?

Az aktív slave tud beavatkozni is.

A mérésen 3 PLC van, 1 master, 1 aktív slave, 1 passzív slave.

9.5. Milyen üzemmódjai vannak a PLC-ben megvalósított szabályozóknak?

A szoftverrel megvalósított szabályozóknak az alábbi három üzemmódja van:

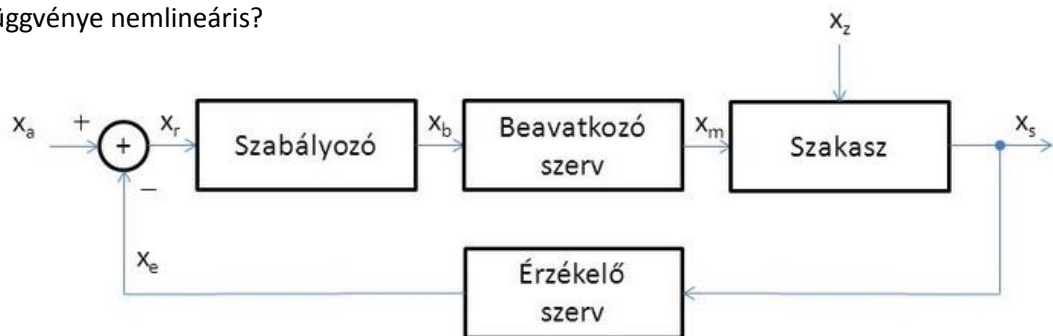
M (manual): kézi üzemmód, a végrehajtó jelet a kezelő állítja be.

A (automatic): automatikus üzemmód, a végrehajtó jelet a szabályozó algoritmus adja.

R (remote): a szabályozó hálózaton keresztül távvezérelhető.

9.6. Mi a helyzetbeállító? Rajzolja fel a hatásvázlatát! Milyen lehet az átmeneti függvénye?

Olyan szervomotor, melynek elmozdulását érzékelő méri, és egy belső visszacsatoláson keresztül szabályozza. Átmeneti függvénye nemlineáris?

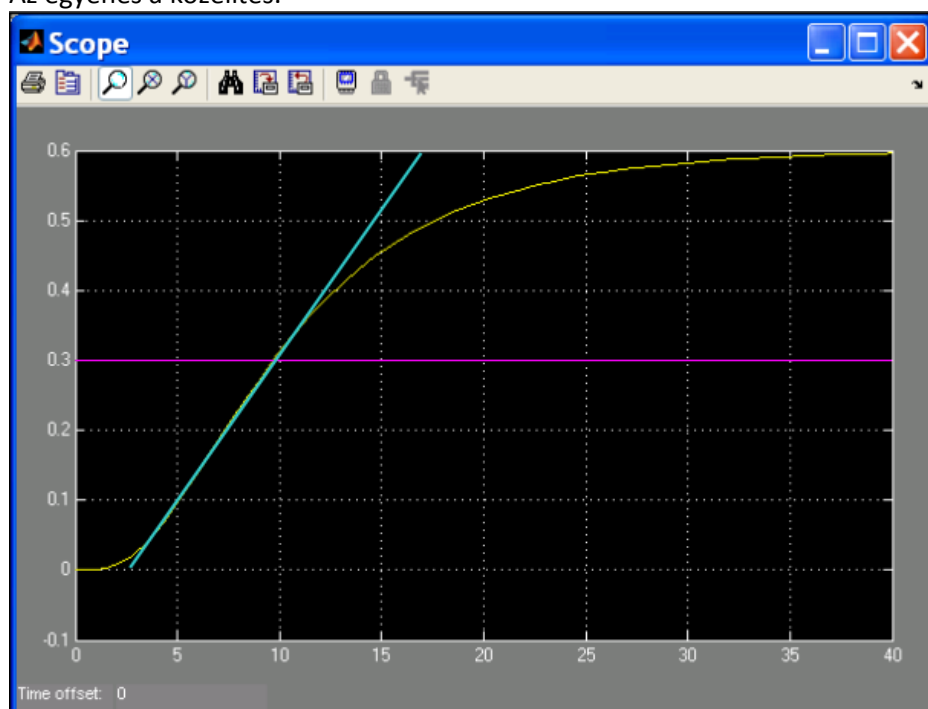


9.7. Miért nemlineáris a legtöbb szabályozási kör?

A beavatkozók (itt szelepek) nemlineáris működésűek.

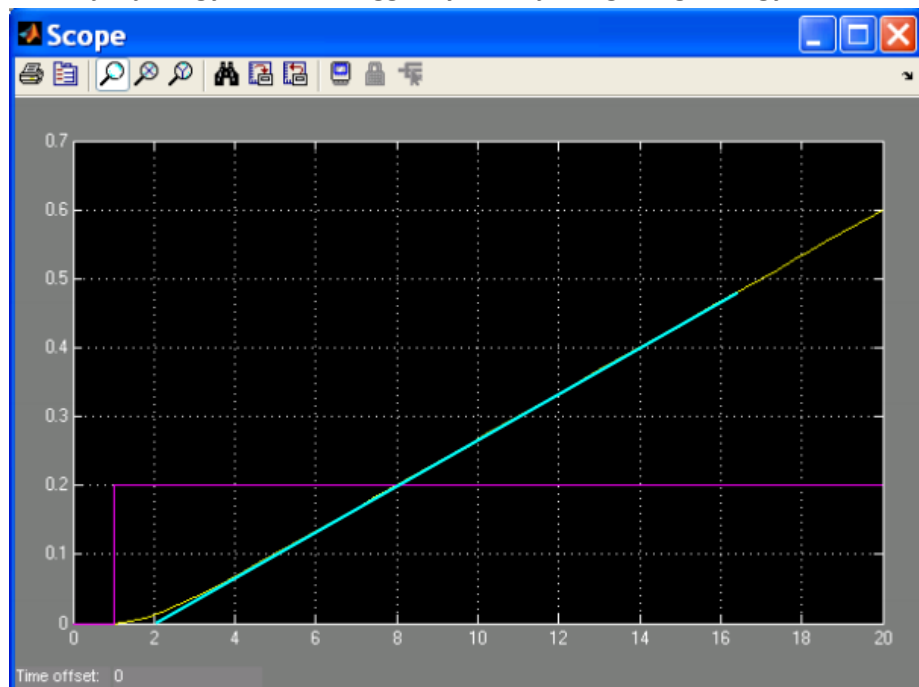
9.8. Rajzoljon egy átmeneti függvényt és adja meg holtidős egytárolós közelítését!

Az egyenes a közelítés.



Holtidő: $T_H \cong 2sec$ Erősítés: $A_p = \frac{0.6}{0.3} = 2$ Időállandó: $\tau \cong 17 - 3 = 14sec$

9.9. Rajzoljon egy átmeneti függvényt és adja meg integráló egytárolós közelítését!



Integrálási idő (0.2-t integráljuk, tehát az az idő, amíg a kimenet 0.2-t nő): $T_I \cong 6sec$

Az időállandó (a gerjesztés megjelenése és a kék érintő közti szakasz): $T \cong 1sec$

9.10 Mit jelölünk PB-vel?

Proportional Band – arányossági tartomány

$PB = 1 / K_c \cdot 100[\%]$, ahol K_c a szabályozó erősítése.

9.11. Egy körre az OPPELT módszer nem alkalmazható. Melyik az és miért? Hogyan méretezne hozzá PI szabályozót?

Integráló folyamathoz az OPPELT módszer nem alkalmazható (szintmérés), mivel a módszer az egységugrás vizsgálatára épül. Vegyük fel a folyamat átmeneti függvényét és közelítsük integráló egytárolós taggal.

9.12. A $\xi=0,25$ –ös csillapítási tényező eléggé lengő megoldást jelent. Hogyan változtatná a szabályozó paramétereit, hogy nyugodtabb viselkedést kapjon?

Csökkenteném a szabályozó erősítését, növelném az integrálási időt.

9.13. Az egyenszálalékos szelepet tartalmazó körben melyik a stabilitás szempontjából legkedvezőtlenebb munkapont?

Ahol a legmeredekebb a teljesítmény-térfogatóram karakterisztika.

9.14 Az átmeneti függvényből holtidőt és időállandót határozunk meg. Mi lehet ezeknek a fizikai oka elméletileg és a valóságban?

Holtidő: a szelepeket mozgató végrehajtók fogaskerék áttétele okozhatja

Időállandó: a szelepek mozgásának véges sebessége miatt

Beugró: mindenkinek más, az összes kérdés szerepelt

Mikrokontroller labor, 9. mérés ellenőrző kérdések

- Milyen alap, ill. kiegészítő szolgáltatásokat nyújt egy beágyazott operációs rendszer általában?

A beágyazott operációs rendszerek alapvető feladata, hogy az egyes elvégzendő feladatok közötti ütemezéshez támogatást nyújtson. További kiegészítő szolgáltatások például a megosztott erőforrások biztonságos kezelése, valamint a taszkok közötti kommunikáció.

- Mit nevezünk taszknak?

Az operációs rendszer által ütemezhető végrehajtási egység.

- Milyen állapotokban lehet egy taszk?

A taszk öt lehetséges állapotban lehet: - alszik - futásra kész - fut - várakozik – megszakítva

Az alvó állapotban lévő taszkok a memóriában vannak, viszont létezésükről az operációs rendszer nem tud, illetve törölve lettek. A futásra kész állapotban lévő taszkok készen állnak a futásra, viszont az ütemező még nem adta meg a processzorhasználat jogát nekik. A taszk futó állapotban van, amennyiben birtokolja a processzort. A taszk várakozhat valamilyen esemény bekövetkezésére (közös erőforrás felszabadulása, I/O művelet befejezése, stb.), ilyenkor várakozó állapotban van. Végül a taszk megszakítva állapotban van akkor, ha futása közben egy megszakítás következett be.

- Hogyan működik egy preemptív ütemező?

Mindig a legmagasabb prioritású futásra kész taszk használhatja a processzort. Az operációs rendszer minden ütemezéskor megvizsgálja, hogy van-e a jelenleg futó taszknál magasabb prioritású futásra kész taszk, amennyiben van, az éppen futó taszktól megvonja a processzor használati jogát, és odaadja azt a legmagasabb prioritású futásra kész taszknak.

- Szemléltesse a közös erőforráshoz történő konkurens hozzáférés problémáját!

Képzeli el, hogy van egy globális változónk (közös erőforrás), mely értékének lekérdezése, ill. beállítása nem valószínűsíthető meg egyetlen assembly utasítással (8 bites processzoron ilyenek pl. az int típusú változók). Tételezzük fel, hogy az egyik taszkunk be szeretné az értéket állítani, a másik pedig ki szeretné olvasni azt. Elképzelhető olyan szituáció, hogy a változó értékének beállítását végző taszktól pontosan akkor vonja meg a futási jogot az ütemező, amikor az int típusú változó egyik bájtját már beállította, de a másikat még nem. Ha ilyenkor a lekérdezést végző taszk kap futási jogot, akkor az int változóból nem azt az értéket fogja kiolvasni, amit a másik taszk be szeretett volna állítani, hanem egy más, érvénytelen értéket (mivel csak az egyik bájtot tudta átírni).

- Milyen módszerekkel lehet a közös erőforrásokat konkurens hozzáférésekkel szemben védeni?

A védelem többféleképpen megvalósítható:

- megszakítások tiltásával
- ütemező tiltásával
- elemi művelettel lekérdezhető/beállítható jelzőváltozó használatával (test-and-set)
- szemafor jellegű objektumok segítségével

- Szemléltesse a szemafor fogalmát!

Tehát a szemafor olyan objektum, amivel jelezhetjük egy erőforrás foglaltságát. Amennyiben az erőforrás foglalt, és más valaki is hozzá szeretne férni, akkor várnia kell a lefoglalással, amíg felszabadul.

- Mi a különbség a bináris és számláló jellegű szemaforok között?

A bináris szemafor a számláló jellegű szemafor speciális esete. A számláló jellegű szemaforok a hozzáférések számát számolják, amikor valaki hozzáfér a szemaforhoz, akkor a számláló csökken (várakozás szemaforra), amikor elengedi (jelzett állapot), akkor nő. Amennyiben a számláló értéke nullára csökken, a további hozzáférések addig várnak, amíg valaki el nem engedi a szemafort.

- Ismertesse a taszkok közötti szinkronizáció lehetséges módjait!

Szemaforok segítségével (A taszk addig vár, amíg a másik el nem engedi a szemafort).

- Ismertessen legalább két, taszkok kezelésére szolgáló µC/OS-II függvényt!

Taszk létrehozása: Taszkot létrehozni az ütemező indulása előtt, ill. egy már futó taszkból is lehetséges az OSTaskCreate() függvény segítségével.

```
INT8U OSTaskCreate (void (*task)(void *pd), void *p_data, OS_STK *ptos, INT8U prio);
```

A függvény első paramétere a taszkot megvalósító függvényre mutató pointer, a második a taszkfüggvénynek átadásra kerülő paraméter, a harmadik a taszk stackjére mutató pointer, az utolsó pedig a taszkhoz rendelt prioritás (kisebb szám jelenti a nagyobb prioritást), ami egyben a taszk azonosítására is szolgál.

Taszk megszüntetése: Taszkot megszüntetni az OSTaskDel() függvény segítségével lehet. Vigyázat, az OSTaskDel() függvényhívás után az adott taszk többet nem indítható el. INT8U OSTaskDel (INT8U prio);

A függvény paramétere a törölni kívánt taszk prioritása, ami egyben az azonosítója is.

- Ismertesse a legfontosabb, szemaforok kezelésére szolgáló µC/OS-II függvényeket!

Szemafor létrehozása: OS_EVENT *OSSemCreate (INT16U cnt);

A paraméterben a számláló kezdeti értékét adhatjuk meg.

Szemafor jezett állapotba hozása: INT8U OSSemPost (OS_EVENT *pevent);

A paraméterben megadott szemafort jezett állapotba hozza (a számlálót növeli).

Várakozás szemaforra: void OSSemPend (OS_EVENT *pevent, INT16U timeout, INT8U *err)

Az első paraméterben megadott szemafort lefoglalja (a számlálót csökkenti), amennyiben a szemafor nem szabad, akkor a második paraméterben megadott számú órajelig vár, a hívó taszk blokkolásával (isr-ben sosem hívunk blokkoló utasítást). A harmadik paraméter pedig azt mondja meg, hogy sikerült-e lefoglalni a szemafort, ill. timeout vagy egyéb hiba következett be.

Várakozás szemaforra blokkolás nélkül: INT16U OSSemAccept (OS_EVENT *pevent)

Abban az esetben, ha nem feltétlenül van szükségünk a szemafor által biztosított erőforrásra használhatjuk az OSSemAccept függvényt a szemafor értékének vizsgálatára, amennyiben a szemafor szabad, akkor a függvény lefoglalja, amennyiben nem, akkor hibával tér vissza.

- Ismertesse a legfontosabb, üzenet várakozási sorok kezelésére szolgáló µC/OS-II függvényeket!

Várakozási sor létrehozása: OS_EVENT *OSQCreate (void **start, INT16U size)

A függvény első paramétere a sor számára lefoglalt memóriaterület kezdőcímére mutató pointer, a második paraméter pedig a lefoglalt memóriaterület bájtokban kifejezett mérete.

Elem elhelyezése a sorban: INT8U OSQPost (OS_EVENT *pevent, void *msg)

A függvény első paraméterében megadott sorba helyezi a második paraméterben megadott elemet.

Várakozás új elemre: void *OSQPend (OS_EVENT *pevent, INT16U timeout, INT8U *err)

Az első paraméterben megadott sorból kiveszi az első elemet, amennyiben a sor üres, akkor a második paraméterben megadott ideig vár új elem érkezésére. A harmadik paraméter megadja, hogy az elemet sikerült e kivenni a sorból, ill. timeout vagy egyéb hiba következett be.

Sorban lévő elem lekérdezése: void *OSQAccept (OS_EVENT *pevent, INT8U *perr)

Megvizsgálja, hogy van-e elem a paraméterben megadott sorban, ha van, akkor visszaadja, egyébként hibával tér vissza.

Beugró: preemptív ütemező, közös erőforrásokat konkurens hozzáférésekkel szemben védeni