Beágyazott és Ambiens Rendszerek (vimiac06)

Témakörök

1. Beágyazott rendszerek általános felépítése: tipikus érzékelők (nagy komplexitású eszköz választása vs. egyedi fejlesztés előnye/hátránya), jelkondicionálási feladatok, ADC és DAC típusok és felhasználási területeik. Tipikus feldolgozó egységek (µP, µC, DSP, FPGA) egymáshoz képesti viszonya az eszközök teljesítménye, tervezési idő és megoldható feladat komplexitása szempontjából (milyen feladatot melyik eszközzel oldanánk meg).

2. Giant Gecko (EFM32-STK3700) fejlesztői kártya: tipikus szenzorok (fényerősségmérő, LC fémérzékelő, érintésérzékelő) ismertetése (működési elv, milyen tervezési irányelvek vannak, milyen µC perifériát használnak). Órajel-menedzsment alapelve a Giant gecko processzoron, hogyan szolgálja a fogyasztás csökkentését.

3. Fejlesztői környezetek és fordítók: fordítás folyamata (.c → obj → link). Néhány tipikus fordítási csatoló ismerése (-D, -O0…-O3, -o, -mcpu, -I, -Wall, -std). make program és makefile formátum ismerése: makefile szabályok (cél, előfeltétel, utasítás szintaktikája, egyszerűbb minták értelmezése)

4. Szoftverarchitektúrák: szoftverarchitektúra tervezésének szempontjai. A tipikus szoftverarchitektúrák jellegzetes felépítése (mintakód, ütemezési diagram, válaszidő…) és tulajdonságaik: ciklikus programszervezés (és alesetei), megszakítással kiegészített, ütemezett függvények.

5. Megszakításkezelés: Megszakítás inicializálásának és kezelésének általános elvei. A megszakítás-engedélyezés általános hierarchiája, vektoros megszakításkezelés elve, Cortex-M3, ATmega128 és ADSP-BF357 megszakítás-kezelő felépítése (elvi szinten, bonyolult ábrák nem kellenek). C nyelvű megszakítás-kezelő függvények megadásának néhány módja (Cortex-M3, ATmega128, ADSP-BF537, ADSP21364). Hogyan kerülnek a függvények a vektortáblába, és hogyan jelezzük a fordítónak, hogy ők megszakításfüggvények?

6. Megosztott változók: alapprobléma megfogalmazása, milyen változók esetén kritikus, mintapélda, megoldási lehetőségek. Kettős bufferelés. Dinamikus memóriahasználat beágyazott rendszerekben (malloc függvény). Stack túlcsordulás. Robusztus programozás (timeout, biztonságos kódolás, strukturált programszervezés, típushasználat, redundancia).

7. Speciális C nyelvi elemek: inline függvények, bitmező struktúrák, union adattípus, regisztertömbök struktúrált kezelése (memóriaterületek közvetlen elérése), __attribute__ (p.: interrupt, always_inline, weak) és #pragma (pl.: once, interrupt, align) kulcsszavak, idiom recognition. #define specialitások Példák.

9. Hordozható kód: mit jelent, miért fontos, integer adattípusok (stdint.h), könyvtári függvények (mire kel figyelni, blokkoló/nem blokkoló). Virtualizáció: 1-es és 2-es működési modell, hypervisor feladata, vele szemben támasztott követelmények.

8. Hibakeresés. Beágyazott rendszer debuggolási sajátosságai. Hibakeresésre használt eszközök: debugger, tracer, profiler, watchpoint. JTAG port segítségével felépített debug rendszer blokkdiagramja, JTAG segítségével megvalósított tipikus feladatok. Tipikus

alapszintű debug lehetőségek (GPIO, UART: printf átirányítása). Futási idő mérésének módszerei. Időzítők alapfelépítése, konfigurálása (gyakorlat).

10. Adatfeldolgozó rendszerek tervezésének lépései (mérés, algoritmus tervezése/tesztelése, hardver kiválasztása, alg. implementálása, tesztelés). Adatfeldolgozó rendszerek szoftverarchitektúrái. Mintánkénti és blokkos adatfeldolgozó szoftvermodell ismertetése.
Mintánkénti feldolgozás: késleltetés, bonyolultság, kihasználtság és determinisztikusság alakulása az időzítő, AD és DA különböző sorrendű kezelése esetén. Időzítési diagrammok különböző felépítés esetén, késleltetés és adatfeldolgozásra használható idő számítása. Egyszerű programkód adatgyűjtő szoftverre (lásd: gyakorlaton is szerepelt). Adatfeldolgozó algoritmus elhelyezkedése az adatfolyamban. Adatfeldolgozási programok értelmezése.
Blokkos adatfeldolgozás: tipikus feladatok és prioritásuk (adatgyűjtés és feldolgozás). Bufferek kezelésének módja, kettős bufferelés jelentősége. Mintapélda elemzése jelfeldolgozási szempontból: pitch shift algoritmus (adatmozgatás, decimálás, bufferméret megválasztása).
Blokkos és adatonkénti feldolgozás összehasonlítása. Áttérés az egyes módok között.

11. Mozgóablak-átlagolás: Átlagolás képlete. Átlagolás átviteli függvénye (legalább jellegre helyesen tudni kell lerajzolni, hogy *N* minta átlagolása esetén milyen az átvitel, hány zérus helye van, és azok hova esnek). Hogyan viszonyul a diszkrét idejű átlagolás a folytonos idejű átlagoláshoz. Mintapélda megértése: PWM-el kialakított háromszög alakú áramjel mérése, zajszűrése: spektrum alapján hasznos jel és zaj elkülönítése. Fokszám megválasztása spektrum alapján. Milyen hatása van időtartományban/frekvenciatartományban a túl rövid vagy túl hosszú átlagolásnak. Implementáció gyorsítása rekurzív számítással ($sum_{n+1}=sum_n - x_{n-N} + x_n$). Módszer előnyei hátrányai.
Exponenciális átlagolás: Átlagolás átviteli függvénye (legalább jellegre helyesen tudni kell lerajzolni, hogy α értéke hogyan állítja a törésponti frekvenciát). α értékének számítása a mintavételi frekvencia, illetve az időállandó vagy a törésponti frekvencia ismeretében. Kapcsolat az exponenciális átlagolás és első fokú analóg RC szűrő között. Mintapélda megértése: PWM-el kialakított háromszög alakú áramjel mérése, zajszűrése: spektrum alapján hasznos jel és zaj elkülönítése. Időállandó megválasztása spektrum alapján (hova tegyük a törésponti frekvenciát). Milyen hatása van időtartományban a túl nagy vagy túl kicsi időállandónak. Exponenciális átlagolás implementációja. Módszer előnyei hátrányai.

12. Szűrési műveletek implementációja.
FIR szűrők előnyei/hátrányai. LS és Remez tervezési módszerek. Szűrés képlete ($y_n = \sum_{i=0}^{N-1} w_i x_{n-i}$) Megvalósítás bemeneti adatok shiftelésével. Cirkuláris buffer: milyen paraméterekkel írjuk le, hogyan tároljuk az adatokat. DSP-n milyen, a konvolúciót támogató HW egységek találhatók, ezek hogyan működnek (MAC, HW-es ciklusszervezés, cirkuláris buffer, párhuzamos memória-hozzáférés, párhuzamos utasítás-végrehajtás). Egy ASM program elemzése (nem kell fejből tudni). SIMD üzemmód: mit jelent, hogyan működik pl. konvolúció esetén.
Cirkuláris buffer kezelése szoftveresen, HW támogatás nélkül mikrokontrolleren: moduló osztás, ciklus megosztása, adatok redundáns tárolása (ismétlése), teljes és részleges loop unroll,
IIR szűrők előnyei/hátrányai. Butterworth, Cheby 1 / 2, elliptikus szűrők: jellegzetes átvitelek jellemzése és felismerése. Szűrés képlete időtartományban ($y_n = \sum_{i=0}^{N-1} b_i x_{n-i} - \sum_{i=1}^{N-1} a_i y_{n-i}$). Egyszerű implementáció bemeneti adatok shiftelésével. Biquad implementáció: miért szükséges másodfokú tagokra (biquadokra) bontani, hogyan néz ki egy biquad átvitele, hogyan programozható le. C-ben hogyan tudok egész aritmetika felhasználásával egész és tört

számokat szorozni (példa programkód 120*2.625-re szerepelt). Exponenciális átlagolás fixpontos törtszámábrázolással.

13. FPGA-k: (a bonyolult blokkdiagramokat memorizálni nem kell, a megértés elősegítését szolgálják) Konfiguráció tárolásának módja. IO blokkok főbb funkciói. Órajel-menedzsment: DCM főbb funkciói, órajelelosztó hálózat topológiája. FPGA hierarchikus felépítése: CLB→Slice → (LUT, FF + kiegészítő logikák). Gyors átvitelterjesztés felhasználásának módjai: összeadás, komparátor, számláló, szorzó. Multiplexer logika. LUT alapú shift regiszter. Szorzás támogatása: LUT alapú és HW-es. RAM típusok. Hardvergenerálás menete. Tipikus felhasználási területek. Párhuzamosítás és pipeline.