Rendszerarchitektúrák laboratórium – DSP fejlesztési technológiák

Bevezető mérési feladatok a „Beágyazott operációs rendszer alkalmazása jelfeldolgozó processzoron” című altémához

BME-MIT

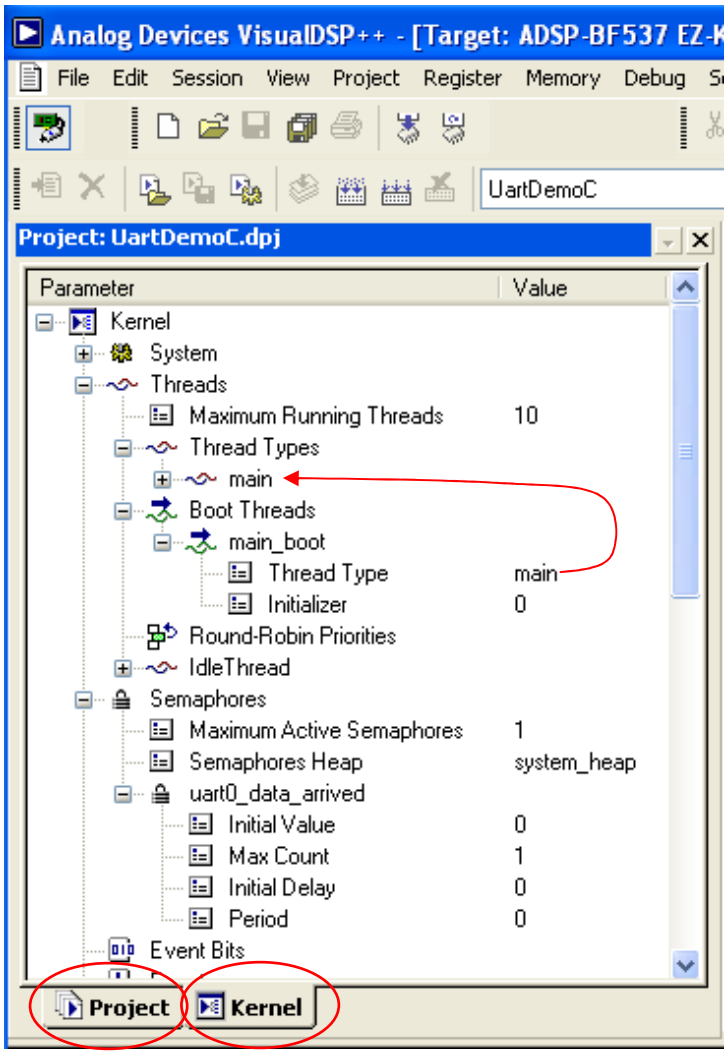
2014

Fejlesztőkártya és fejlesztőkörnyezet előkészítése

- Csatlakoztasd a DSP-kártyát a PC-hez!
- Csatlakoztasd az analóg ki- és bemeneteket az oszcilloszkóphoz és a jelgenerátorhoz!
- Csatlakoztasd a soros portot a kártyához!
- Töltsd le a honlapról a második feladathoz tartozó mintaprojektet, és tömörítsd ki!
- Indítsd el a fejlesztői környezetet!
- Indítsd el a Hyper Terminalt (Start menü → Programs → Accesories → Communications → Hyperterminal)!
    - A soros port beállításai: 115200kbps, 8N1, **Flow Control: None**

UART kezelés

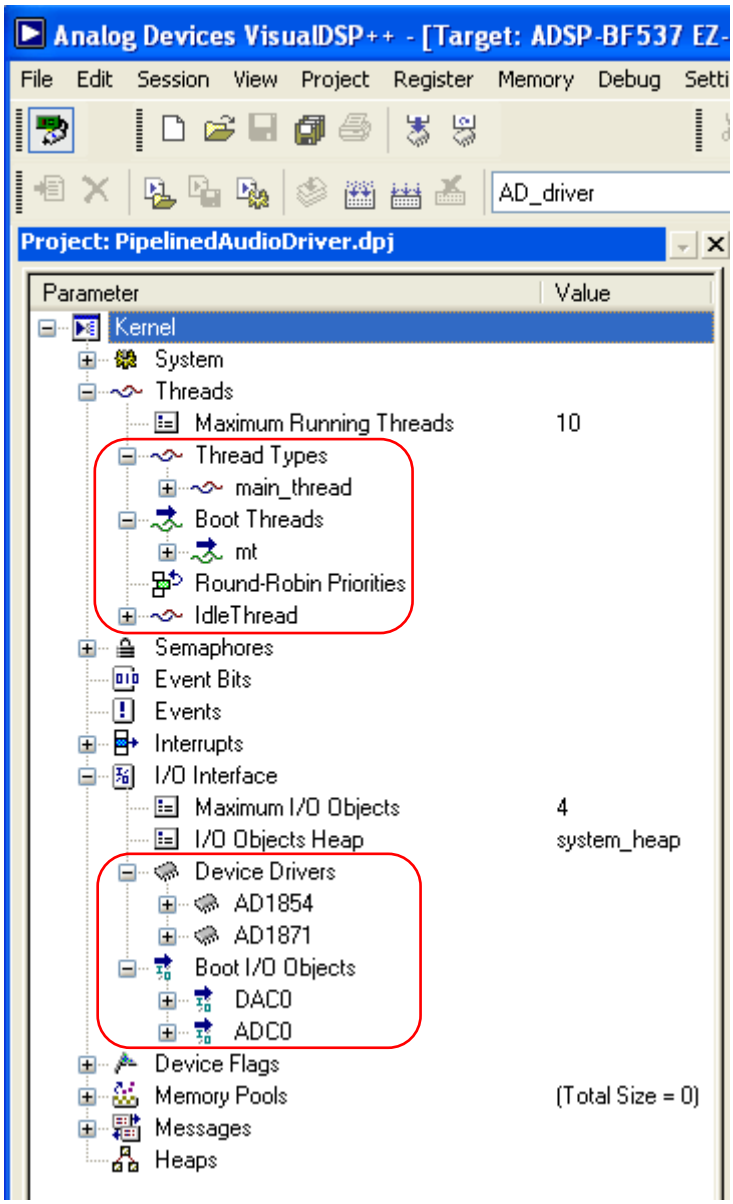
- Nyisd meg az UartDemoC projektet!
- Fordítsd és futtasd a programot! A HyperTerminalban üss le egy billentyűt, és *várj türelmesen* addig, amíg nem látod a hatását a fejlesztőrendszer Output Window ablakában!
- Elemezd a kódot különös tekintettel main.c és EVT_IVG10.c fájlokra! Hogyan történik az UART kezelése operációs szinten (hogyan kommunikálnak a szálak)?
- Egészítsd ki az EVT_IVG10.c fájlt az UART adat beolvasását követő sorban a következő paranccsal: `*pUART0_THR = uart0_value;` Mi a hatása?
- A VDK keretrendszer alatt a Projektfájlokon kívül az operációs rendszer konfigurálását lehetővé tévő Kernel fül is megtalálható a Project ablakban. Tanulmányozd az operációs rendszer konfigurációs lehetőségeit:



- Hogyan vannak tárolva a szál típusok és a bootoláskor elinduló szálak, mi teremt kapcsolatot a kettő között? Milyen tulajdonságai vannak egy szálnak?
- Milyen szemafor található a projektben? Milyen tulajdonságai vannak? Hogyan hivatkozunk rá a kódban?
- Milyen interrupt található a projektben?

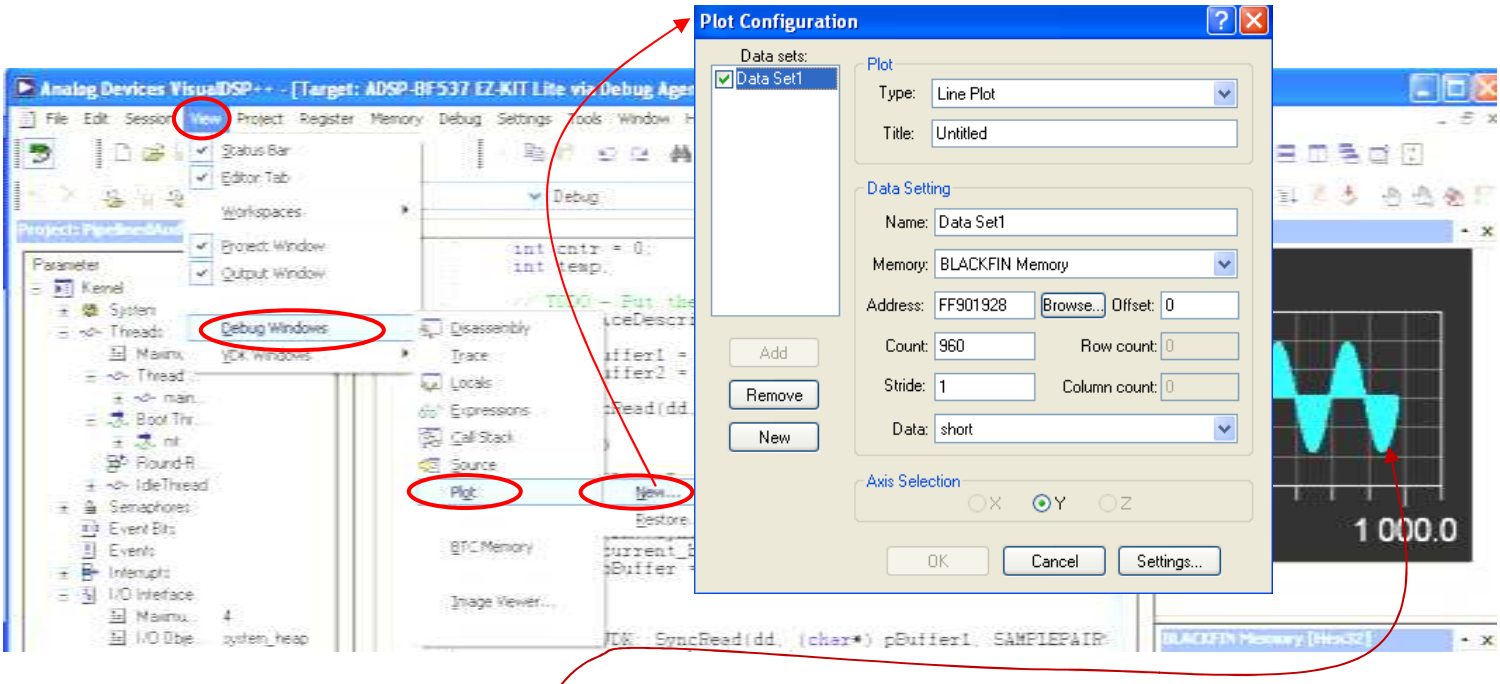
AD átalakító kezelése

- Nyisd meg az AD_driver projektet (PipelinedAudioDriver.dpj)!
- Vizsgáld meg, hogy milyen szálak találhatóak a projektben, és hogy hogyan jelenik meg operációs rendszer szintjén az AD és DA kezelése!



- A main_thread.c fájlban vizsgáld meg, hogy hogyan történik az AD kezelése és a kettős bufferelés!
- AD/DA adatformátumok vizsgálata:
  - Fordítsd le a projektet! Még NE futtasd!
  - Állíts brakepointot a `cntr = 0;` sorra!
  - Futtasd a kódot! Várd meg, amíg a brakepointra ráfut a kód! Ne futtasd tovább! (Ha mégis véletlenül továbbfut, akkor újra le kell fordítani!)

- A pBuffer pointer mutat az aktuális bufferre, az általa mutatott területet kell megjeleníteni a következő módon:
  - A pBuffer változó tartalmának megkeresése (vagy egyszerűen rávisszük az egeret, és megvárjuk, míg megjelenik az értéke; vagy Memory menü → BLACKFIN Memory, és a megjelenő ablak szövegmezejébe beírni: „pBuffer”, formátumként pedig Hex32-t kell beállítani). A pBuffer pointer értéke az a memóriacím, ahol a bejövő adatok találhatóak.
  - A pBuffer által mutatott területet jelenítsük meg (View menü → Debug Windows → Plot → New… menüpont) a pointer által mutatott területet megadva kezdőcímként, hosszként pedig 960-at adjunk meg (2 csatorna, 480 adat/csatorna), formátum: short (lásd alább).



- Zoomolj rá az adatokra! Mit lehet megállapítani az adatok tárolására vonatkozóan?

AD/DA és UART együttes kezelése

- Nyisd meg az ADDA_UART projektet (driver_try.dpj)!
- Vizsgáld meg, hogy milyen szálak találhatóak a projektben!
- Fordítsd le a programot és futtasd! Az oszcilloszkópon vizsgáld meg, hogy HyperTerminal-ban „+” és „-” gombokat nyomva, hogyan módosítja a DSP a bemenetére érkező jelet!
- Vizsgáld meg, hogy milyen üzenetküldési mechanizmusokon keresztül adják át egymásnak az adatokat a fő szál, valamint a be- és kimeneti adatokat kezelő szálak!
- Vizsgáld meg, hogy az uart_control.cpp-ben hogyan történik az UART kezelése!
- Módosítsd a programot úgy (main_thread.cpp, az AttenuateBuffer16to32(…) függvény helyére kell írni a kódot), hogy a bemenetre érkező adatokat egy `for` ciklusban másold át a kimenetre (egyszerű echo program)!
  Megjegyzések:
  - Érdemes a buffereket tömbként, és nem pointerként kezelni, pl.: p_output[ii].
  - A bemeneti adatok 16 bitesek, a kimeneti adatok 24 bitesek, ezért 8 bites shiftelést kell alkalmazni.
  - A jobb és bal csatorna adatai a ki és bemeneti bufferekben párosával felváltva szerepelnek: `[b0 j0 b1 j1 b2 j2 … b479 j479];` `j`: jobb csatorna adatai, `b`: bal csatorna adatai.

Saját feladat elkészítése

- Az ADDA_UART (driver_try.dpj) projektből kiindulva valósítsd meg a mérésvezetővel egyeztetett feladatot!