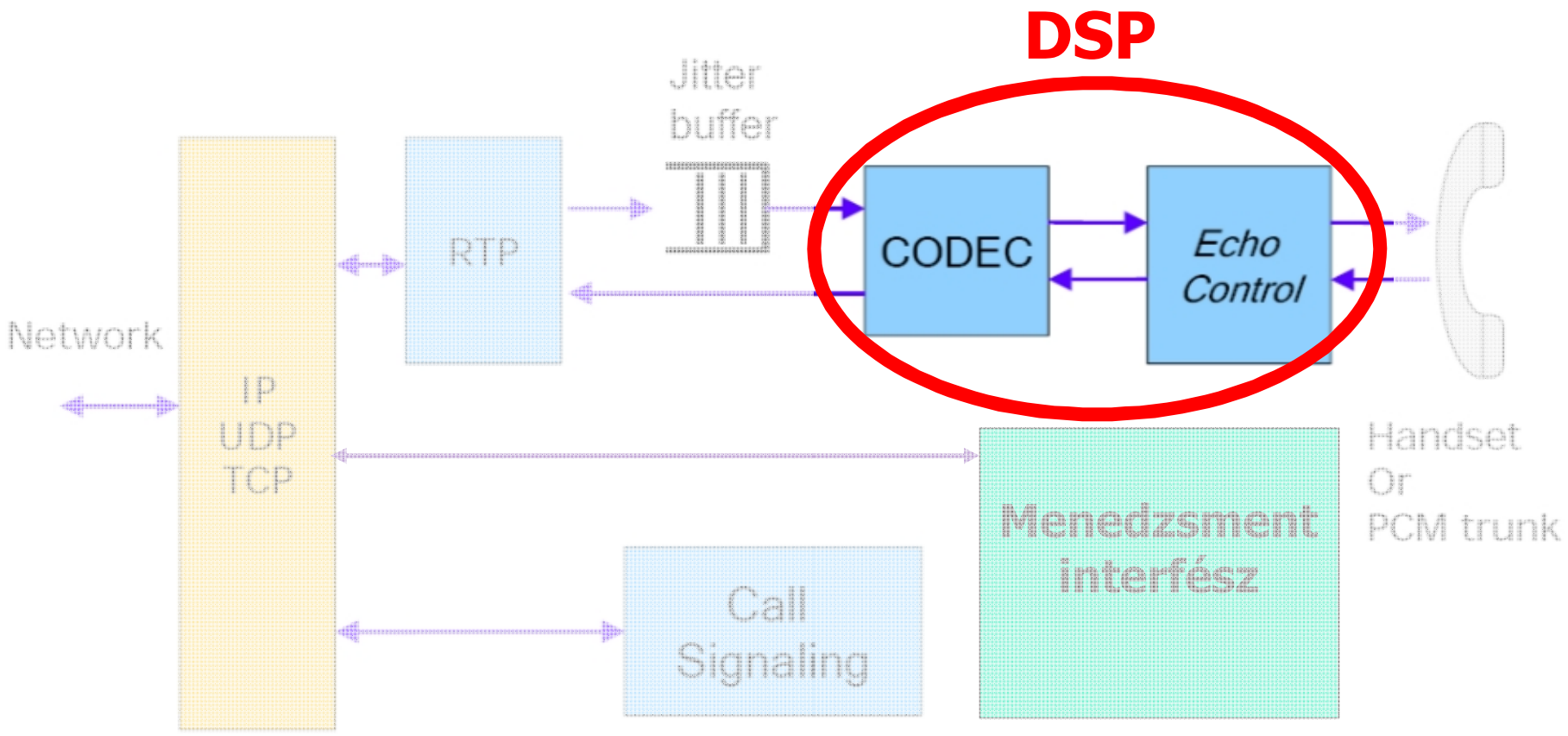
Építsünk IP telefont!

Moldován István

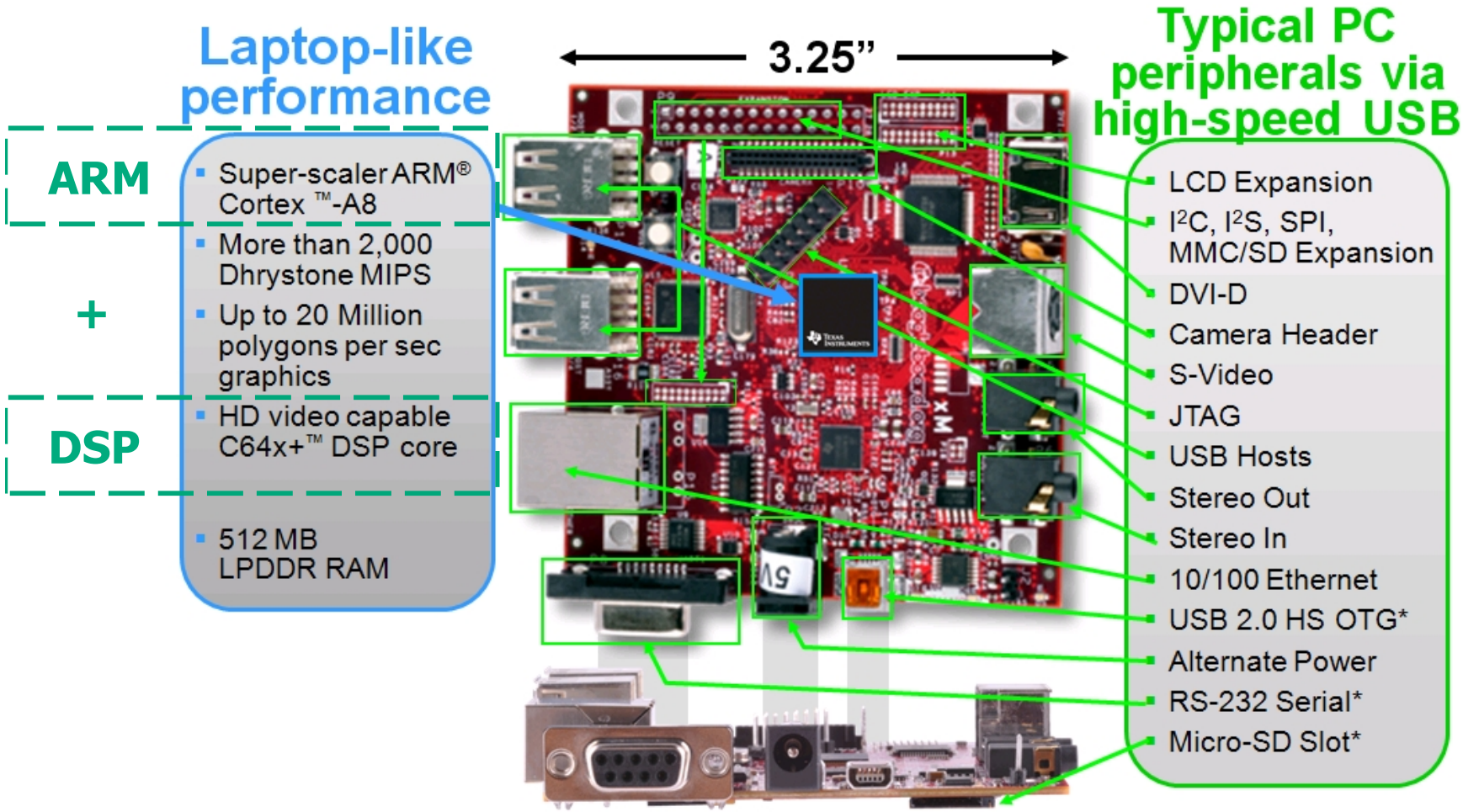
Sonkoly Balázs

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
TÁVKÖZLÉSI ÉS MÉDIAINFORMATIKAI TANSZÉK

Egy IP telefon felépítése



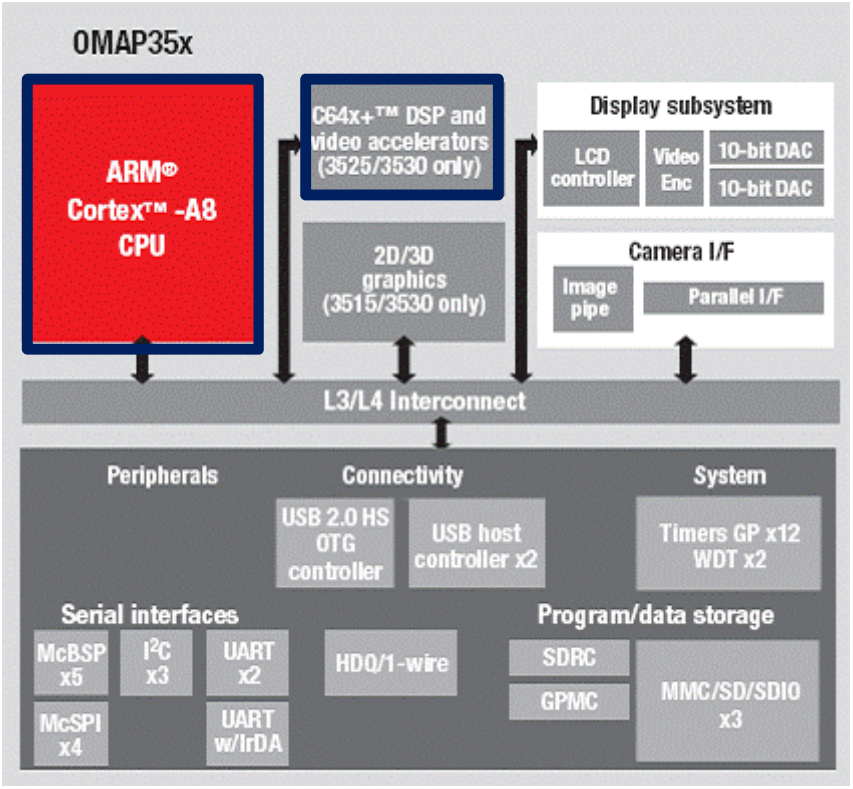
OMAP3530 Platform



*** Supports booting from this peripheral**

OMAP3530 ARM-DSP

- **ARM-DSP SoC**
  - **System-on-Chip megoldás**
- **A DSP-k és SoC-ok speciális jelfeldolgozási feladatokra fejlesztett egycsipes mikroszámítógépek**
- **A szenzorok és egyéb jelek feldolgozására optimalizáltak**
- **Integrált perifériák és egyéb gyorsítók**



OMAP3530 - DSP

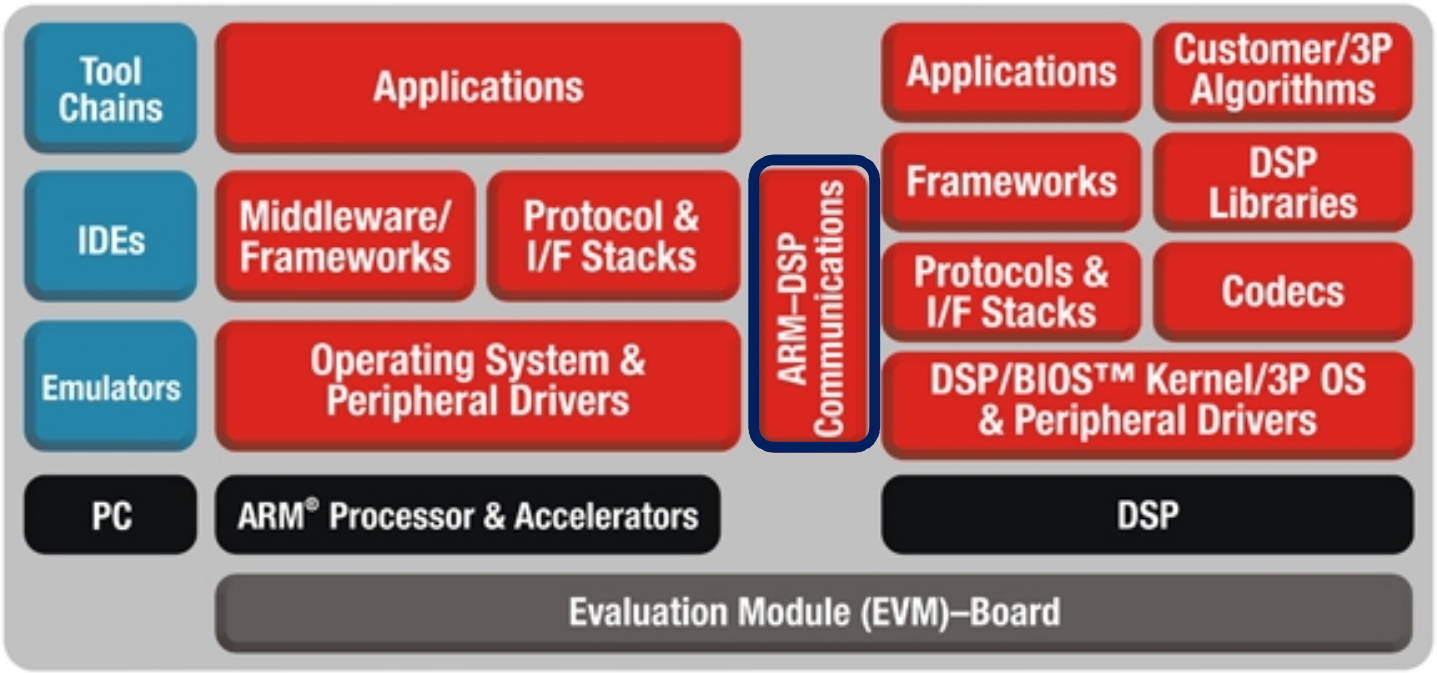
- ***A DSP*** **(*Digital Signal Processor*) egy mikroprocesszor típus, amely jelfeldolgozási feladatokat rendkívül gyorsan képes megoldani.**
- **A DSP a digitális jelet dolgozza fel, és alkalmazási köre a hang- és videofeldolgozáson kívül rendkívül sok helyen alkalmazható (pl. telekommunikáció: szoftver rádió megvalósítás, vagy akár DSL jelek feldolgozása)**
- **Az OMAP3530-ben a DSP jól kiegészíti a nagyteljesítményű superscalar ARM Cortex-A8 processzort, és egy általánosan használható gyorsítót kapunk amely akár grafika és video feldolgozására is kitűnően alkalmas, mindezt egyetlen chipben**

DSP programozása

- DSP programozása – meglepetés!
  - C-ben lehetséges
- A TI biztosít hozzá fejlesztői környezetet: az eXpressDSP-t ami tartalmaz
  - Code Composer Studio – fordító
  - DSP/BIOS realtime kernel
  - Szabványok az együttműködés és újrafelhasználásra

OMAP3530 DSP Core

- **ARM-DSP**
- **A kapcsolat az ARM mag és a DSP között speciáls követelményeket támaszt**

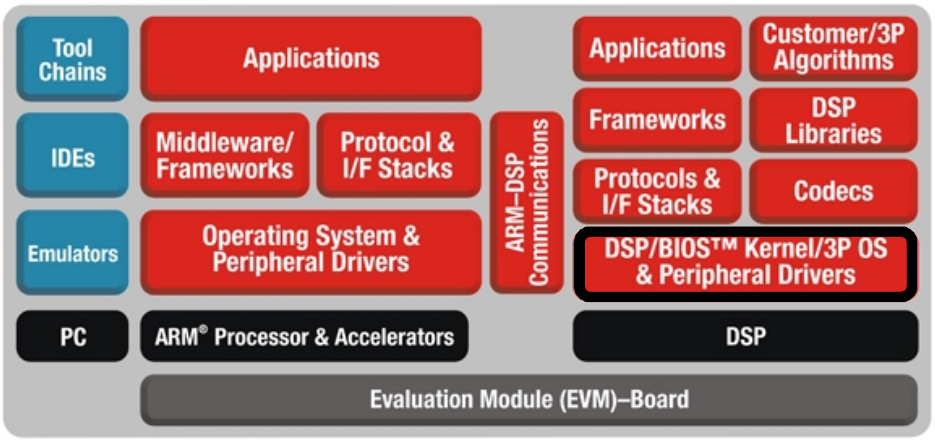


Kapcsolat a DSP-vel

- Az ARM processzoron Linux fut
  - A Linux indítja az alkalmazásokat amelyek DSP-t igényelnek
- A DSP-n speciális operációs rendszer fut
  - DSP/BIOS - Texas Instruments által fejlesztett nyílt oprendszer

DSP/BIOS

- DSP/BIOS egy real time operációs rendszer többféle DSP-re és mikrokontrollerre
- A DSP/BIOS széles körű rendszer szolgáltatásokat nyújt a beágyazott alkalmazás felé
  - Preemptive multitasking
  - Memory management
  - Real-time analysis



DSP/BIOS szolgáltatások

- ***Preemptive multitasking***
  - **provides support for several different types of *threads* *in an embedded system.***
  - ***Hardware Interrupt, Software interrupt, Task, Idle***
- ***Memory management***
  - **Provide to set up an embedded system's *memory map* *and also to allow memory buffers to be allocated and deallocated***
- ***Real-time analysis***
  - **provide modules to provide information about how the system is executing.**

OMAP3530 DSP Core - ARM-DSP kapcsolat

- **DSP/BIOS**
- **SYS/BIOS 6.x Letölthető a TI honlapjáról**
- **http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/bios/sysbios/**

SYS/BIOS 6.x Product Releases	
SYS/BIOS 6.32.03.43 Recommended for use with: IPC 1.23.03.31 XDCtools 3.22.02.27	SYS/BIOS 6.x patch release compatible with CCS 4.2, CCSv5 or newer. This release is recommended for use with Concerto devices. It also includes a number of bug fixes. See release notes for details.
SYS/BIOS 6.32.02.39 Recommended for use with: IPC 1.23.02.27 XDCtools 3.22.01.21	SYS/BIOS 6.x patch release compatible with CCS 4.2, CCSv5 or newer. Single Bug fix for missing C6000 and ARM project wizard examples in CCS. See release notes for details.
SYS/BIOS 6.32.01.38 Recommended for use with: IPC 1.23.01.26 XDCtools 3.22.01.21	SYS/BIOS 6.x patch release compatible with CCS 4.2, CCSv5 or newer. Bug fixes and custom build improvements. Improved FLASH boot support for 28x. See release notes for details.
SYS/BIOS 6.32.00.28 Recommended for use with: IPC 1.23.00.16 XDCtools 3.22.00.09	SYS/BIOS 6.x point release compatible with CCSv4.2 or CCSv5. Adds support for pre-built libraries and a new custom library flow to improve the build times. Improved configuration user interface and bug fixes. Support for additional MSP430 and Stellaris devices (see release notes for full list).
SYS/BIOS 6.31.05.31 Recommended for use with: IPC 1.22.05.27 XDCtools 3.20.08.88	SYS/BIOS 6.x patch release compatible with CCS 4.2, CCSv5 or newer. Update to installer to add logging. Added jar key signing for CCSv4 installation. No need to update if you are standalone. See release notes for details.
SYS/BIOS 6.31.04.27 Recommended for use with: IPC 1.22.05.27 XDCtools 3.20.08.88	SYS/BIOS 6.x patch release compatible with CCS 4.2, CCSv5 or newer.
SYS/BIOS 6.31.03.25	SYS/BIOS 6.x patch release compatible with CCS 4.2, CCSv5 or newer
SYS/BIOS 6.31.02.23	SYS/BIOS 6.x patch release compatible with CCS 4.2, CCSv5 or newer
SYS/BIOS 6.31.01.19	SYS/BIOS 6.x patch release compatible with CCS 4.2, CCSv5 or newer
SYS/BIOS 6.31.00.18	SYS/BIOS 6.x point release compatible with CCS 4.2, CCSv5 or newer
SYS/BIOS 6.30.03.46	SYS/BIOS 6.x patch release compatible with CCS 4.2 or newer
SYS/BIOS 6.30.02.42	SYS/BIOS 6.x patch release compatible with CCS 4.2 or newer
SYS/BIOS 6.21.03.21	SYS/BIOS 6.x patch release compatible with CCS 4.0 or newer
SYS/BIOS 6.21.02.19	SYS/BIOS 6.x patch release compatible with CCS 4.0 or newer
SYS/BIOS 6.21.01.16	SYS/BIOS 6.x patch release compatible with CCS 4.0 or newer
SYS/BIOS 6.21.00.13	SYS/BIOS 6.x release compatible with CCS 4.0 or newer
SYS/BIOS 6.20.02.43	SYS/BIOS 6.x release compatible with CCS 4.0

Kommunikáció egy tipikus alkalmazás esetén

- Tipikus alkalmazás - codec. A következő dolgokra van szükség:
  - Codec API
  - Buffer/memória lefoglalása
  - Buffer/memória menedzsment
  - A ki/bemenetek vezérlése a driverek között
  - Kommunikáció az ARM processzor és DSP között
    - IPC – inter processz kommunikáció

Kommunikációs lehetőségek

- A TI két megoldást adott
  - DSP/BIOS Bridge
    - developed by Texas Instruments, but It is currently maintained by OMAPpedia/Openomap
    - Enables parallel computing on the GPP and the IVA2 of the OMAP
  - DSP /BIOSLink
    - developed and still maintained by Texas Instruments
    - A slimmer version of the dsp-bridge, also developed by TI.
    - Focus on multimedia encoding and decoding

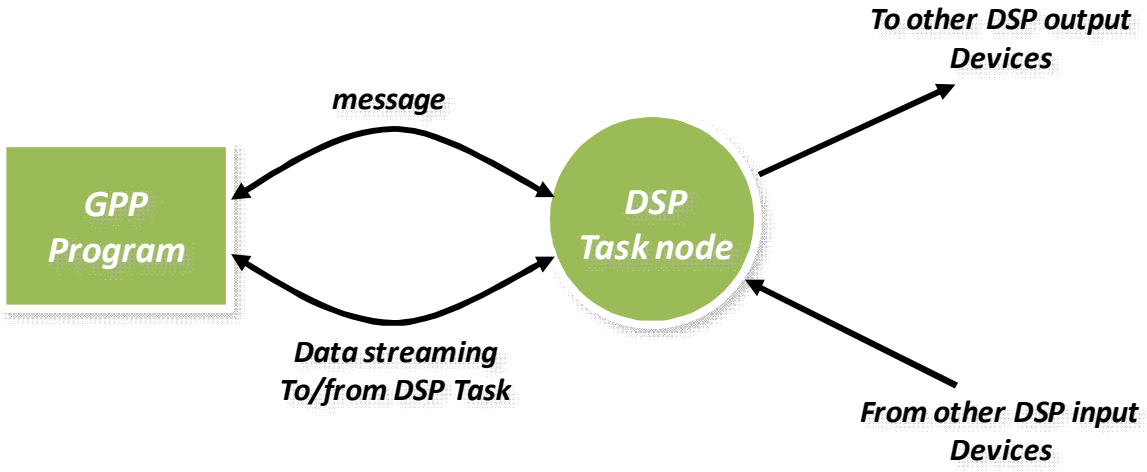
DSP/BIOS BRIDGE

DSP/BIOS Bridge

- ***Az OMAPPEDIA/OMAPZOOM tartja karban***
  - ***De a TI fejlesztette ki***
- ***DSP/BIOS Bridge***
  - A DSP-n futó taszkokkal kétféle módon kommunikál
    - ***Messaging*** (short, fixed-length packets).
    - ***Data streaming*** (multiple, large buffers).
  - Megengedi a GPP (General Purpose Processor) applikációnak hogy megmondja milyen ki/bemeneteket (streaming channels) használhat egy DSP task
  - Használja : Gstreamer (an open source media framework)

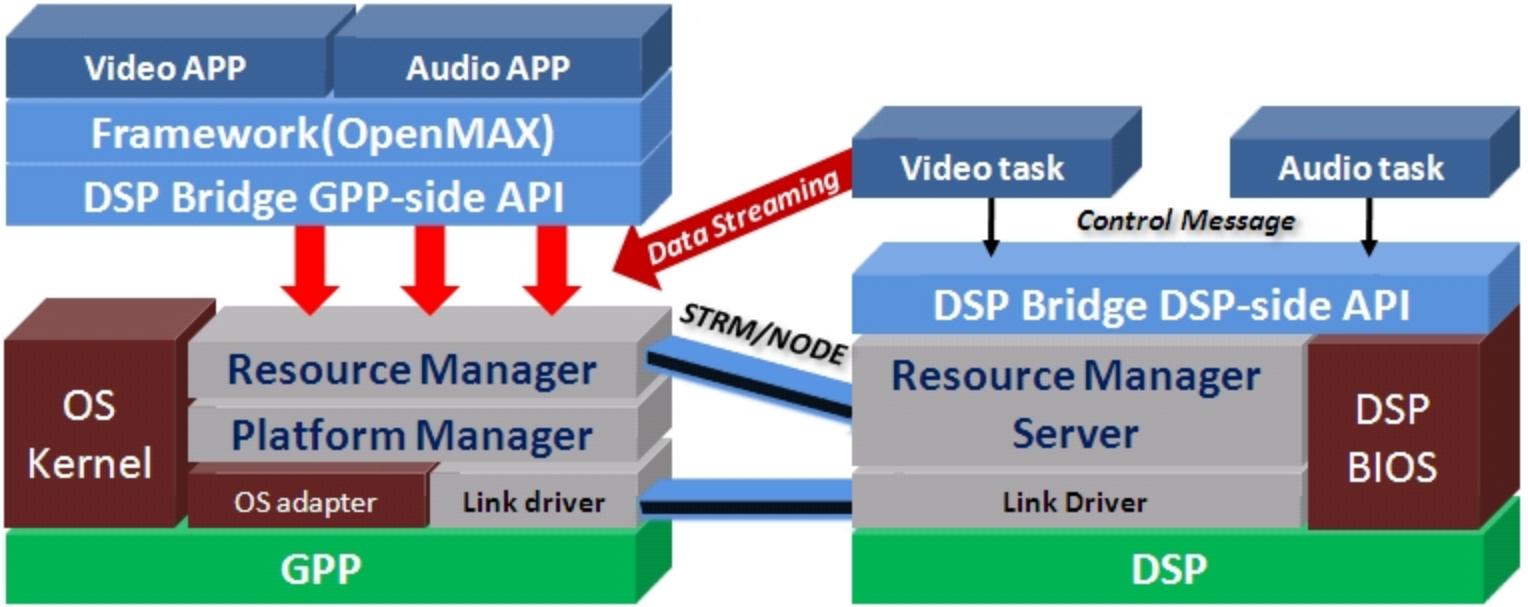
DSP/BIOS Bridge

- ***DSP/BIOS Bridge***
  - DSP/BIOS Bridge eredetileg olyan környezetekre volt kidolgozva, ahol egy GPP mellett több DSP is lehet a rendszerben
  - A GPP a ***master vagy "host"*** processzor, és a kapcsolódó DSP-k párhuzamos adatfeldolgozással támogatják a GPP-n futó alkalmazást



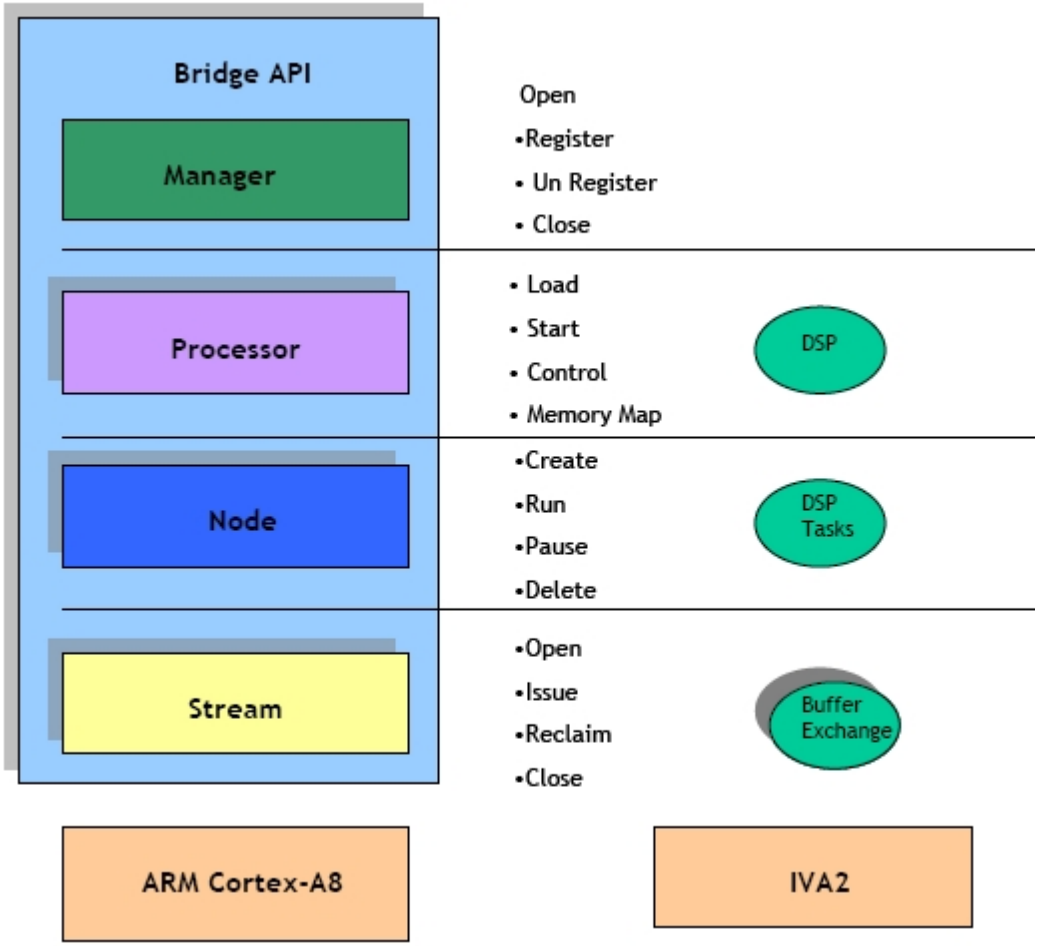
DSP/BIOS Bridge

- ***DSP/BIOS Bridge***
  - GPP Szoftver Architektúra
    - A GPP alkalmazás a hozzárendelt DSP taszkkal amely a DSP alrendszeren fut a DSP/BIOS Bridge API segítségével kommunikál.
  - DSP Szoftver Architektúra
    - A DSP/BIOS Bridge hozzárendel egy eszközüggetlen streaming I/O (***STRM***) interfészt, egy jelzési üzenet interfészt (***NODE***), és egy Resource Manager (***RM***) Servert amely az erőforrásokat menedzseli



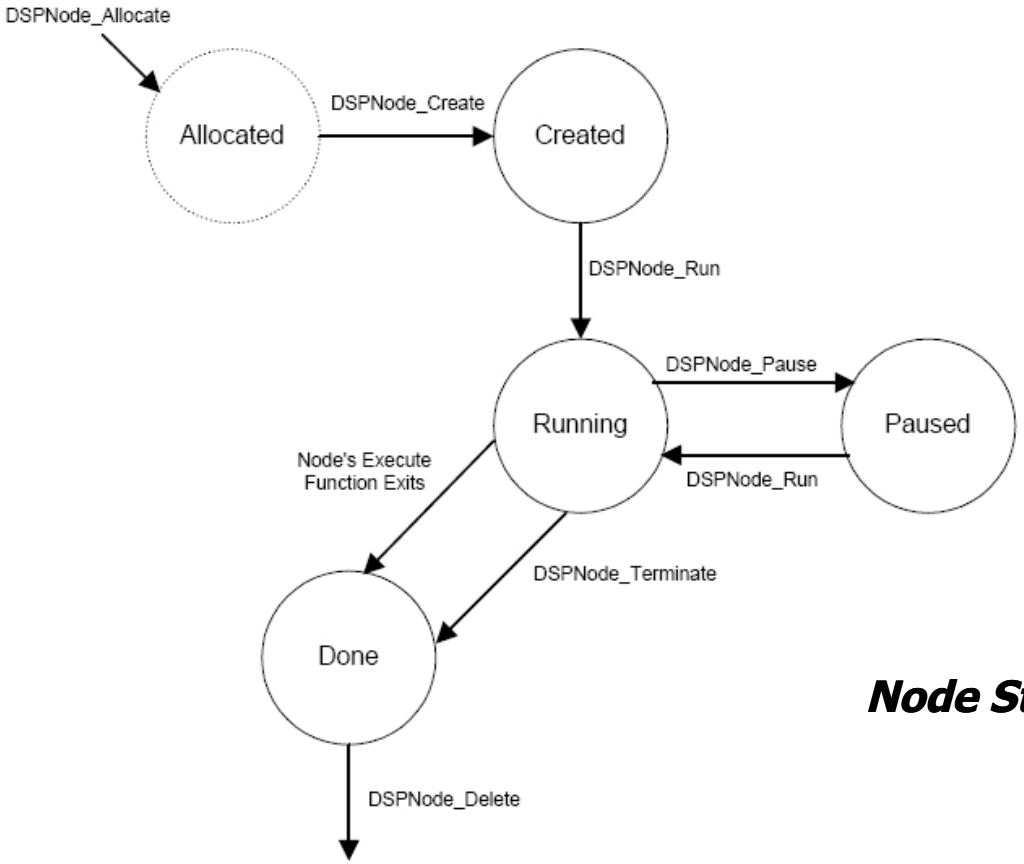
DSP/BIOS Bridge

- ***DSP/BIOS Bridge***
  - Host Sotware
    - DSP Manager Module
    - DSP Processor Module
    - DSP Node Module
    - DSP Stream Module
  - DSP Software
    - STRM Module
    - NODE Module
    - Resource Manager Server
    - DSP Task Model



DSP/BIOS Bridge működése

- ***DSP Node Modul***
  - A vezérlési és jelzési elemeket jelenti amelyek egy adott DSP-n futnak



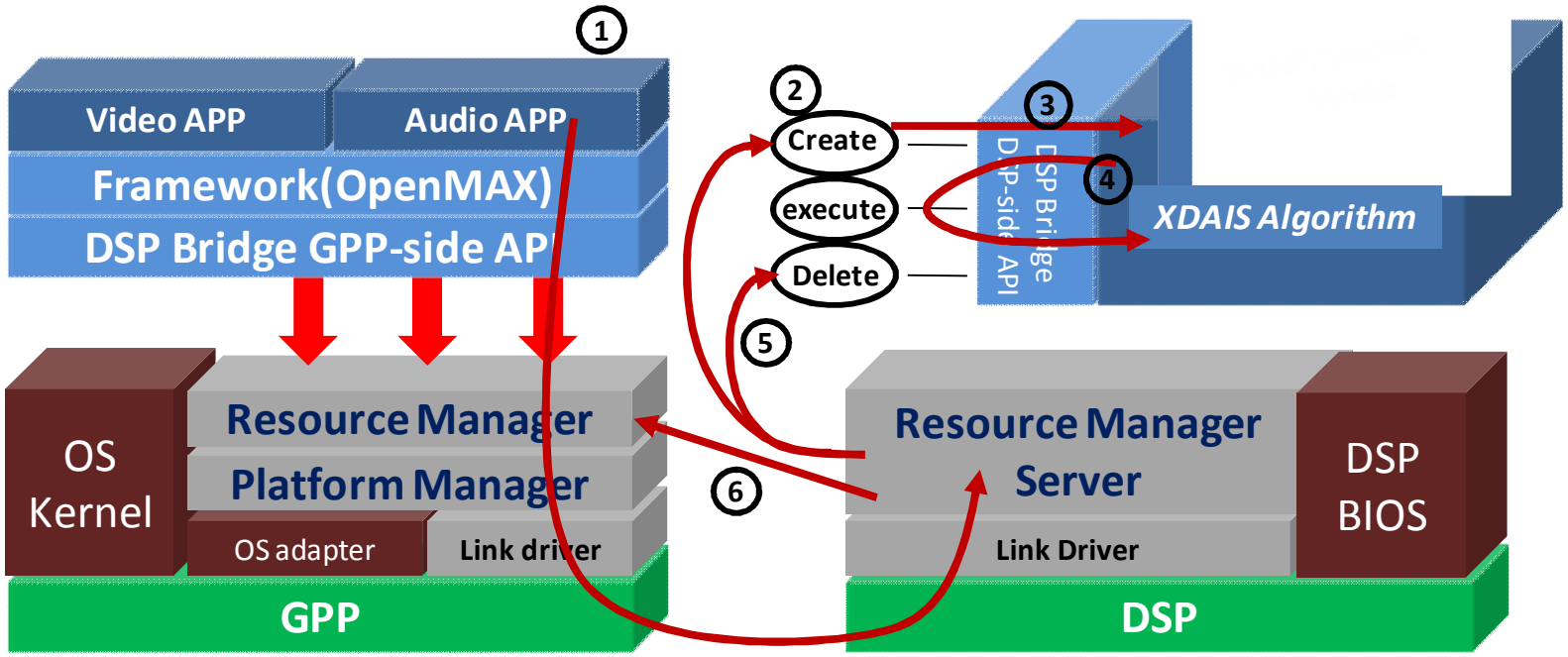
Node State Diagram

DSP/BIOS Bridge - lépések

- ***Control Flow***
  1. Select and Attach to a DSP
     - DSPManager_Open, DSPManager_EnumProcessorInfo, DSPProcessor_Attach
  2. Allocate and Connect DSP Nodes
     - DSPNode_Allocate, DSPNode_Connect
  3. Create Nodes on the DSP
     - DSPNode_Create
  4. Launch DSP Tasks
     - DSPNode_Run
  5. Stream Data to/from DSP Tasks
     - DSPStream_Open, DSPStream_AllocateBuffers, DSPStream_Issue, DSPStream_Reclaim
  6. Exchange Messages with DSP Nodes
     - DSPNode_GetMessage, DSPNode_PutMessage
  7. Terminate DSP Nodes
     - DSPNode_Terminate
  8. Delete DSP Nodes
     - DSPNode_Delete
  9. Detach from DSP
     - DSPProcessor_Detach

DSP/BIOS Bridge – Node-ok

- ***DSP/BIOS Bridge Nodes***
  - DSP Task Model
    - A DSP-n a Resource Manager Server támogatja a node-ok létrehozását, futtatását és törlését valamint a GPP és node-ok közötti üzenetváltást



DSP/BIOS Bridge – fejlesztői környezet

- ***DSP/BIOS Bridge fejlesztő környezet***
  1. Bridge driver telepítése a GPP-re
     - Az újabb verziók már a kernelben vannak a ***2.6.37 verziótól***

```
HOST$ git clone git://dev.omapzoom.org/pub/scm/tidspbridge/kernel-dspbridge.git
HOST$ cd kernel-dspbridge
HOST$ git checkout -b my_dspbridge --track origin/dspbridge
HOST$ make menuconfig arch=ARM
```

```
Symbol: TIDSPBRIDGE [=y]
Type  :
Prompt: DSP Bridge
  Defined at drivers/staging/tidspbridge/Kconfig:5
  Depends on: STAGING [=y] && !STAGING_EXCLUDE_BUILD [=n] && ARCH_OMAP3 [=y]
  Location:
    -> Device
      -> Staging drivers (STAGING [=y])
        -> Exclude Staging drivers from being built (STAGING_EXCLUDE_BUILD [=n])
```

DSP/BIOS Bridge - fejlesztés

- ***DSP/BIOS Bridge fejlesztés***

```
HOST$ make uImage
HOST$ make modules
```

  - driver : bridgedriver.ko, mailbox.ko
  - ***/dev/DspBridge létre kell jöjjön***

2. Alap image betöltése
  - baseimage fájl tartalmazza az összes DSP/BIOS primitivet, min pl. a DSP/BIOS kernel, DSP/BIOS Bridge, driverek, alkalmazás indító kód
  - Toolchain és DSP OS : ***CGT(Code Generation Tools)***, ***DSP/BIOS Package***.
  - ***cexec*** alkalmazást használjuk a DSP baseimage indítására
  - static baseimage :

```
HOST$ ./cexec.out ddspbase_tiomap3430.dof64P
```

  - dynamic baseimage :

```
HOST$ ./cexec.out dynbase_tiomap3430.dof64P
```

DSPLINK ÉS DVSDK

DSPLink

- ***DSP Link***
  - DSP Task Model –t használ
  - Sok TI eszközt és platformot támogat(e.g. DaVinci, OMAP2, OMAP3, és diszkrét GPP+DSP eszközöket).
  - A fő különbség a DSP-bridge –hez képest
    - Hiányoznak az advanced power management támogatások (mint a DVFS és Smart Reflex)
  - DE: van Codec Engine és DMAI támogatás
  - DSPLink –et használ a DVSDK alatt

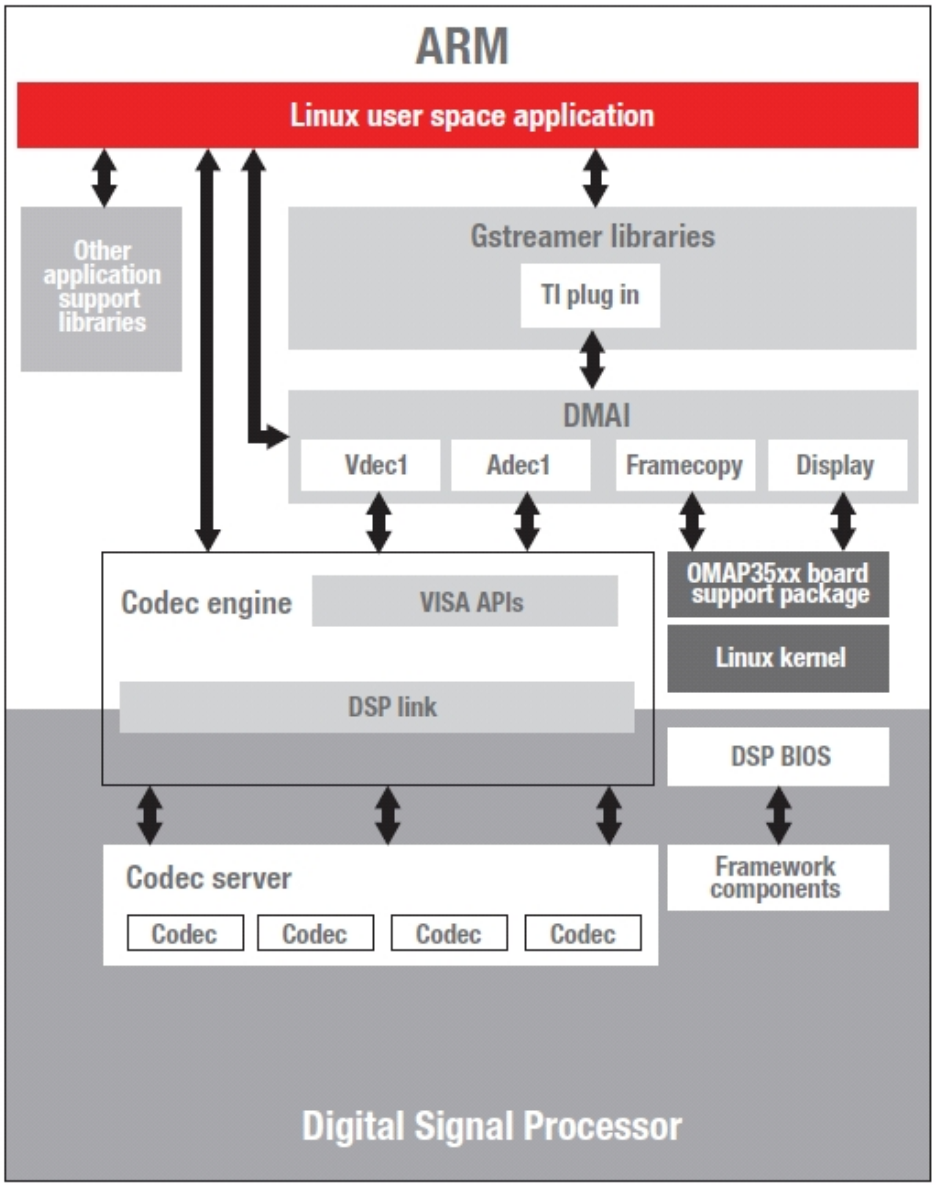
DVSDK

- Mi is a DVSDK ?
  - ***Linux Digital Video Software Development Kits***
  - DVSDK egy tool kit komplex multimédiás feladatok hatékony és egyszerű megoldására
  - DSP-Link-et használ
- Miért használjuk?
  - ARM és DSP együttműködő feladatok kényelmes megoldása
  - Szabványos codec-ek használata és kezelése
  - Közvetlen TI támogatás

DSPLink és DVSDK

- Szoftver Architektúra áttekintés
  - A hardver feladatok absztrakciója miatt , a TI OMAP technológiájú szoftver fejlesztés 3 területre lett bontva
    - Application Layer
    - I/O Layer
    - Signal Processing Layer

DVSDK felépítés



DSPLink és DVSDK

- Codec Engine(CE)
  - Codec Engine egy API szet ami lehetővé teszi ***xDAIS*** szabványnak megfelelő algoritmusok betöltését és indítását
  - A VISA API ugyanaz az alábbi esetek számára
    - A rendszer lehet GPP+DSP, csak DSP, vagy csak GPP
    - Az összes támogatott GPP és DSP ugyanazt az API-t hsználja
    - Az algoritmus futhat lokálisan (a GPP-n) vagy távoli módon (a DSP-n).
    - Minden támogatott oprendszeren ugyanaz az API. (pl.Linux, PrOS, VxWorks, DSP/BIOS, vagy WinCE.)
  - Kezeli az Inter Processors Communication (IPC) réteget az ARM és DSP között
  - Átlátszó módon használja a ***"DSP Server"*** és DSP Link transzportot nagysebességű algoritmusok számára

DSPLink és DVSDK

- Codec Server(CS)
  - A Codec Server egy bináris alkalmazás ami integrálja a codecet, framework komponenseket ésrendszer kódot
  - A Codec Server hasonló egy web szerverhez. Akár egy "***web server***"
  - Ha ***xDAIS-compliant algoritmust*** szeretnél használni, előbb implementálni kell a saját stub-okat és skeletonokat

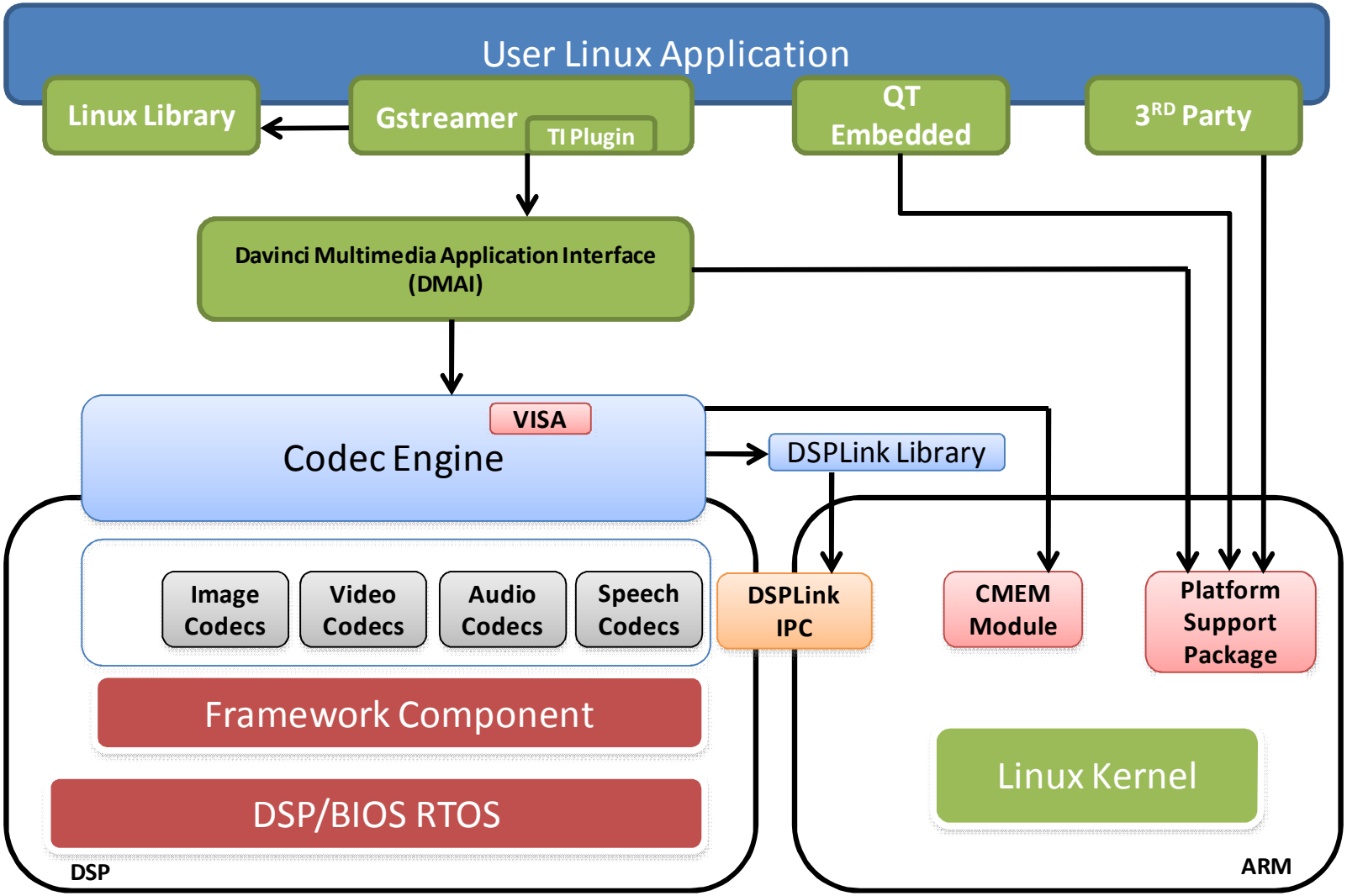
VISA

- VISA (SP réteg) Interfész
  - Video, Imaging, Speech, Audio (VISA)
  - The Signal Processing Layer (SP Layer) 4 fő funkciót támogat
    - Create()
    - Control()
    - Process()
    - Delete()

DSPLink és DVSDK

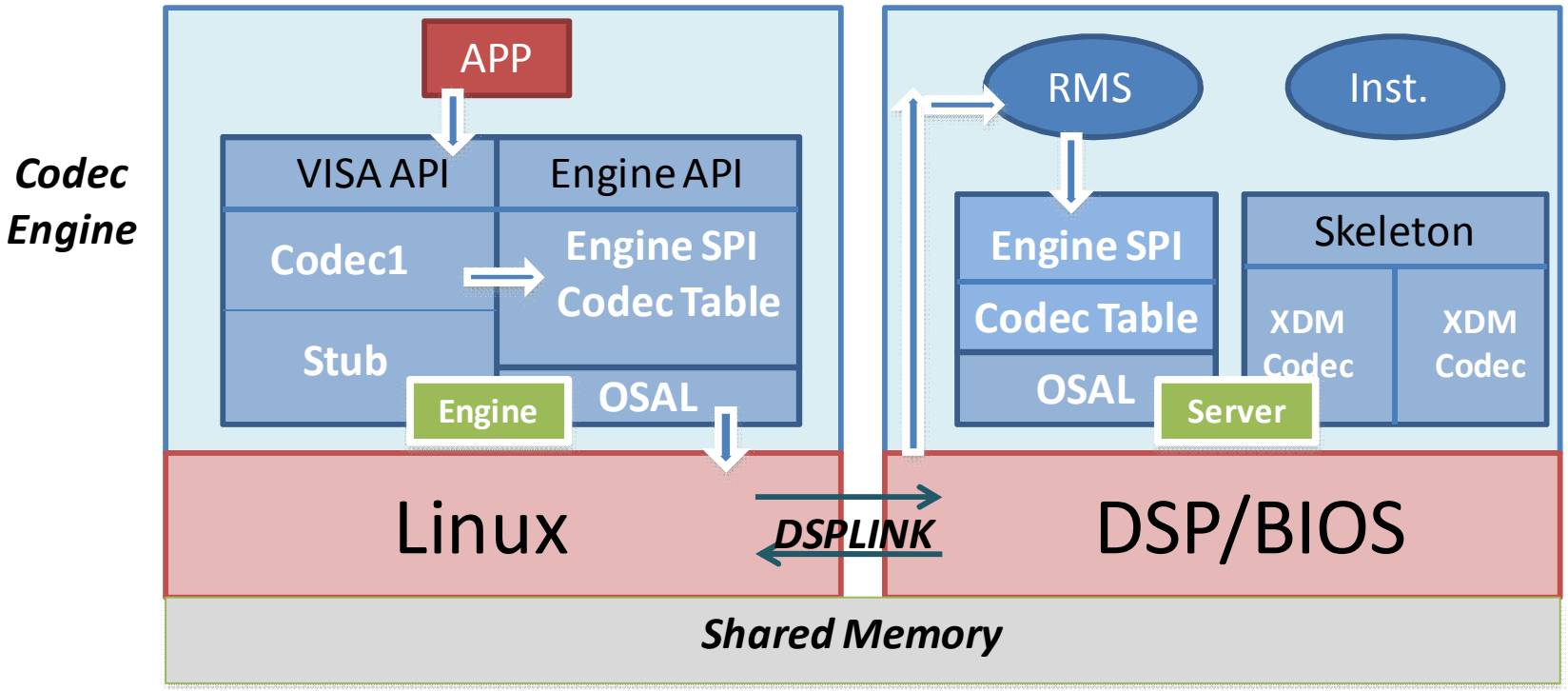
- The algorithms (codec) follow the xDAIS (**eXpressDSP Algorithm Interface Standard**) Standard in order to define their memory resource requirements and enable efficient use of on-chip data memories by client applications

DSPLink és DVSDK működés



Codec Engine

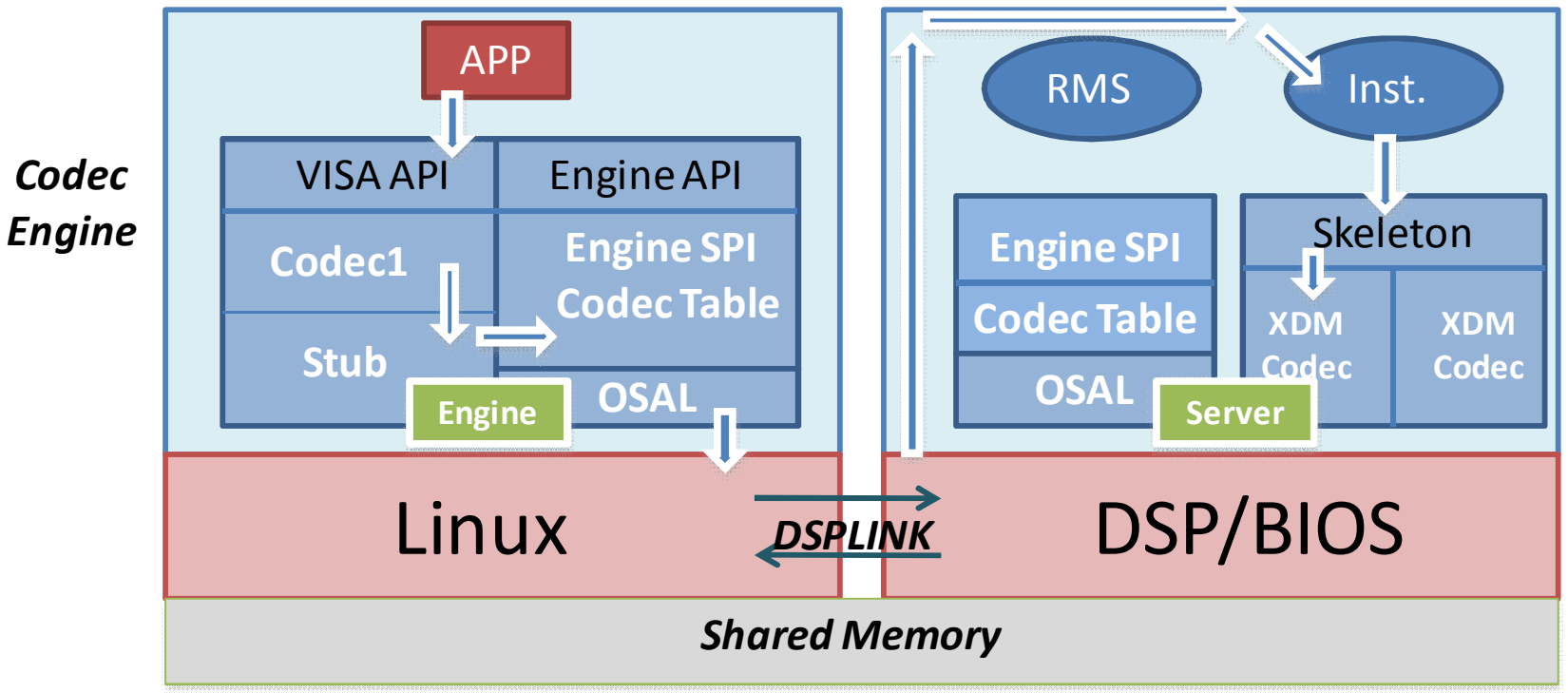
- **Codec Engine Create/Delete Overview**



- **RMS-Remote Management Service**
- **DSP Server: also called Codec server**

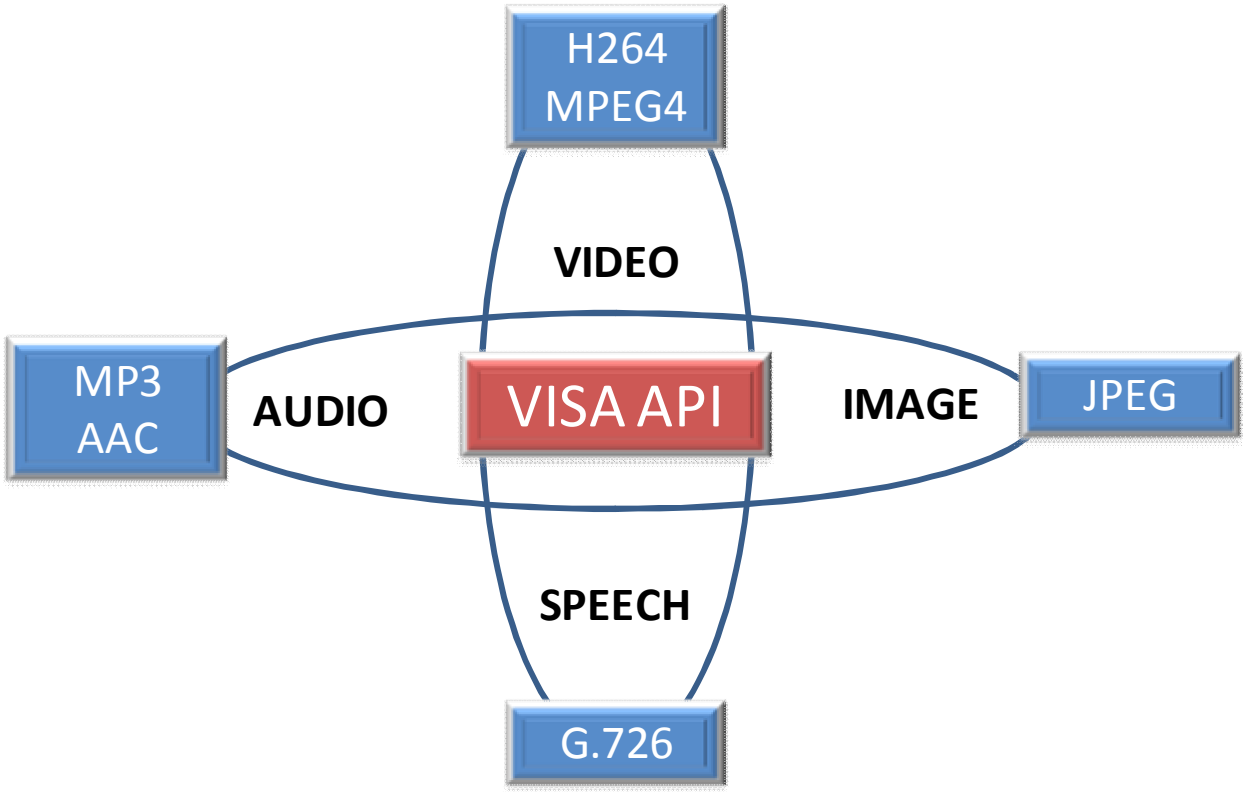
Codec Engine

- **Codec Engine Peocess/Control Overview**



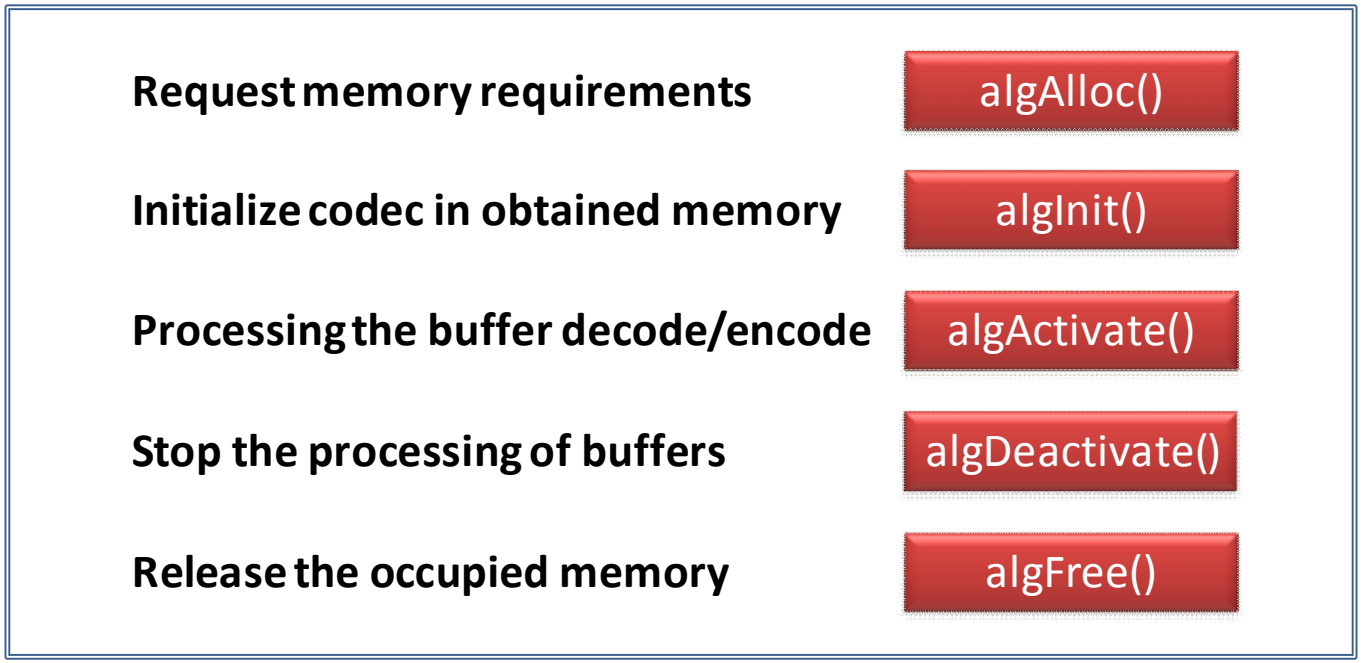
VISA API

- A kódolókat 4 osztályba osztották és a ***VISA API*** támogatja a használatukat



DSPLink és DVSDK

- The functions defined by XDAIS for algorithm
- IALG interface defines a framework independent interface



IALG Interface

DSPLink és DVSDK

- Since the still some complex for developing a application using xDAIS, so xDAIS is extending to xDM(xDAIS for Digital Media)
- xDM is an extension to the IALG interface standard. It defines and standardizes interfaces for multimedia codecs.
- Uniform ***lightweight and high level*** APIs
- xDAIS is wrapped by xDM

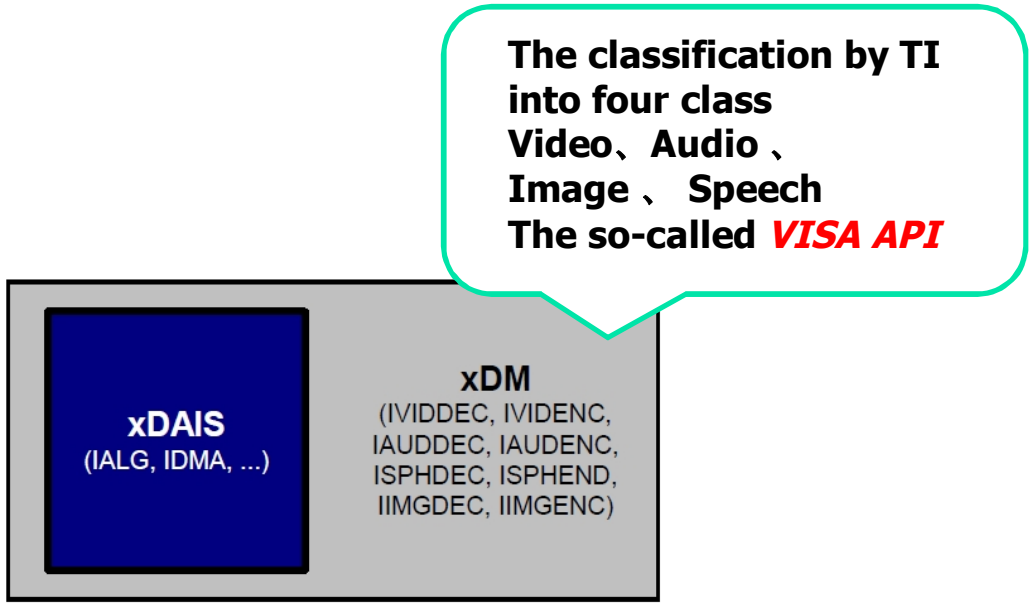


Figure 1-1. Relationship between xDM and xDAIS

DVSDK API

DVSDK API

- A Codec Engine két API csoportot tartalmaz
  - Core Engine API
    - CERuntime_init()
    - Engine_open()
    - Engine_getCpuLoad()
    - Engine_getNumAlgs()
    - Engine_close()
  - VISA API
    - xxx_Create()
    - xxx_Control()
    - xxx_Process()
    - xxx_Delete()

xxx lehet viddec、videnc、auddec、audenc、imgdec、imgenc、sphdec、sphenc

DVSDK API

- Core Engine API
  - Engine_open()
    - After opening an Engine, you can create algorithm instances using the VISA APIs
    - example

```
static String engineName = "auddec";
Engine_Handle ce;
Engine_Error errorcode;
ce = Engine_open(engineName, NULL,
NULL);
```

DVSDK API

- Core Engine API
  - Engine_getNumAlgs()
  - Engine_getAlgInfo()
    - Get the algorithm information on engine
    - example

```
err = Engine_getNumAlgs("audio_copy", &numAlgs);
for (i = 0; i < numAlgs; i++) {
err = Engine_getAlgInfo(name, &algInfo, i);
printf("alg[%d]: name = %s typeTab = %s local = %d\n", i, algInfo.name,
*(algInfo.typeTab), algInfo.isLocal);}

The output may look like the following:
alg[0]: name = auddec_copy typeTab = ti.sdo.ce.audio.IAUDDEC local = 0
alg[1]: name = audenc_copy typeTab = ti.sdo.ce.audio.IAUDENC local = 0
```

DVSDK API

- Core Engine API
  - Engine_close()
    - To close an Engine instance and free memory it uses
    - example

```
Engine_close(cehandle);
```

DVSDK API

- VISA API
  - **Create()**
    - Allocate required resources for the specific codec to run
    - Example:

```
Engine_Handle ce;
AUDDEC_Handle dec;
static String decoderName = "auddec_copy";
/* allocate and initialize audio decoder on the
Engine */
dec = AUDDEC_create(ce, decoderName,
NULL);
```

DVSDK API

- VISA API
  - **Control()**
    - Every codec exports some parameters to be controled
    - Dynamically change Bit-rate、Buffer Size、Volume …etc
    - Example:

```
/* Query the decoder */
status = AUDDEC_control(dec, XDM_GETSTATUS,
&decDynParams,
&decStatus);
if (status != AUDDEC_EOK) {
/* failure, report error and exit */
printf("decode control status = %ld\n", status);
return;
}
```

DVSDK API

- VISA API
  - **Process()**
    - Decode/Encode from a input buffer and generate a raw data/encodeded output buffer
    - Example

```
for (n = 0; fread(inBuf, sizeof (inBuf), 1, in) == 1; n++) {
/* decode the frame */
status = AUDDEC_process(dec, &inBufDesc, &outBufDesc,
&decInArgs, &decOutArgs);
/* write to file */
fwrite(dst, sizeof (outBuf), 1, out);
}
printf("%d frames decoded\n", n);
```

DVSDK API

- VISA API
  - **delete()**
    - Basically the complement of create() API. Once the APL decides to stop the xDM algorithm, it calls this API, which reclaims the resources.
    - example

```
/* tear down the codec and Engine */
AUDDEC_delete(dec);
```

DVSDK API

- Tipikus program szekvencia

```
Idevfd = open(…);
ofilefd= open(…);
Ioctl(…);
CERuntime_init();
myCE = Engine_open(…);
myVE = VIDENC_create(…);

While(…){
Read(idevfd,…);
VIDENC_control(myVE,…);
VIDENC_process(myVE,..);
Write(ofilefd,…);}

close(idevfd);
close(ofilefd);
VIDENC_delete(myVE);
Engine_close(myCE);
```

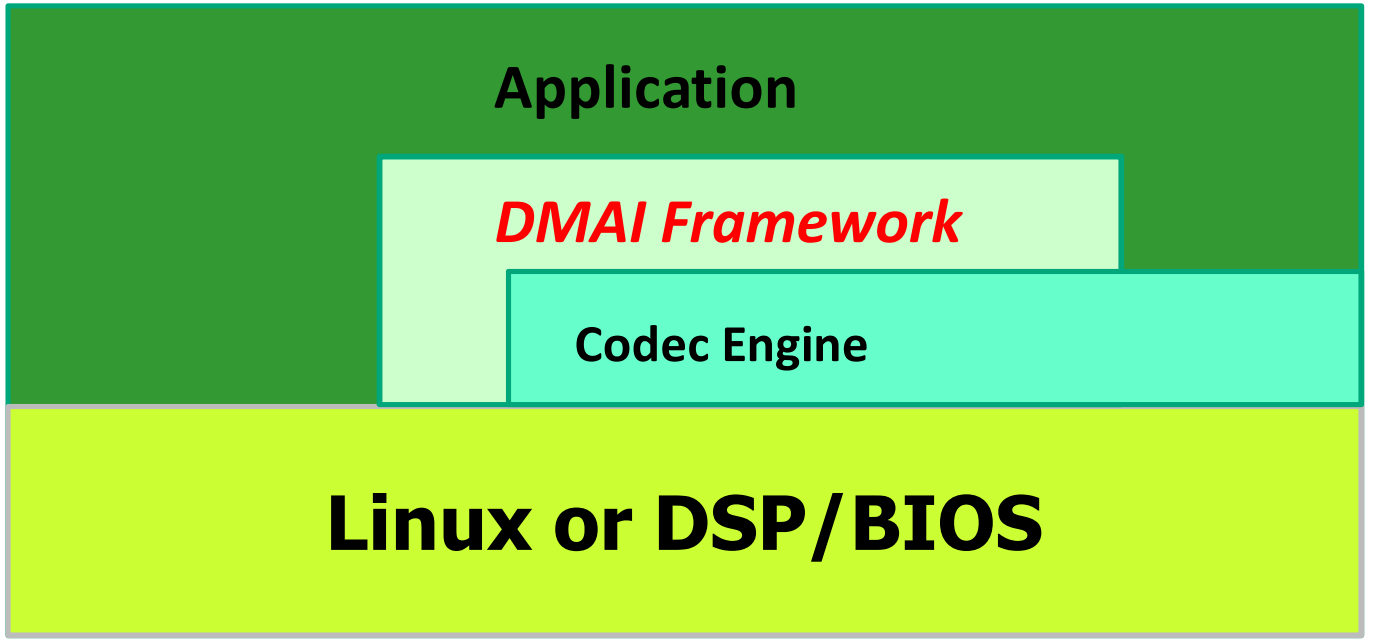
```
// Create Phase
// get input device
// get output device
// initialize IO devices…
// prepare VISA environment
// prepare to use video encoder

// Execute phase
// read/swap buffer with Input
device
// option: perform VISA std algo ctrl
// run algo with new buffer
// pass results to Output device
// Delete phase
// return IO devices back to OS
// algo RAM back to heap
// close VISA framework
```

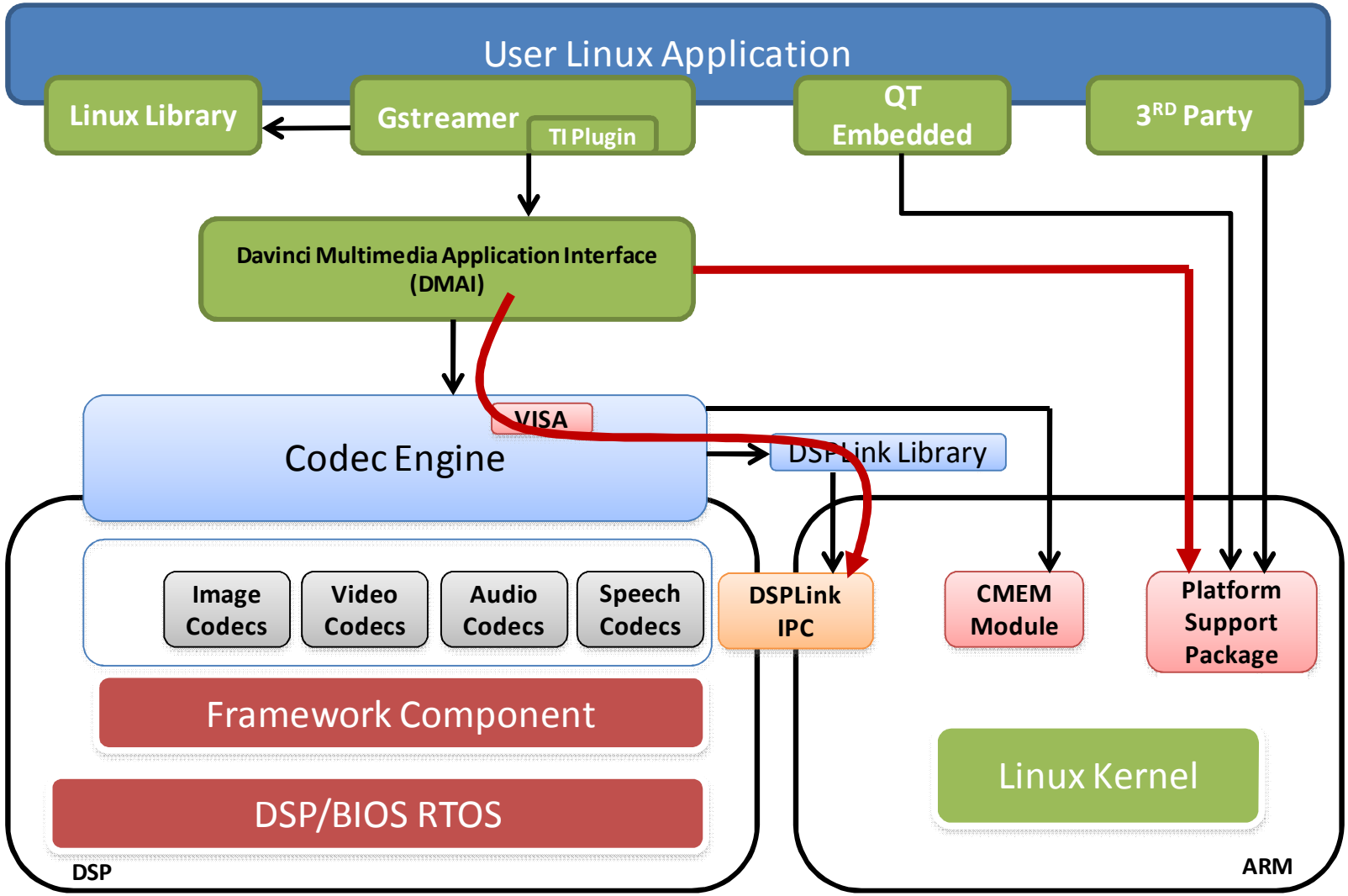
DMAI

DMAI

- A Codec Engine mellett elérhető egy másik API, a ***DMAI (DaVinci Multimedia Application Interface)***
- DMAI egy vékony réteg az OS felett (Linux vagy DSP/BIOS) és lehetővé teszi a Codec Engine használatát hogy gyorsan lehessen PORTOLHATÓ alkalmazásokat írni



DMAI


User Linux Application
Linux Library
Gstreamer
TI Plugin
QT Embedded
3RD Party
Davinci Multimedia Application Interface (DMAI)
VISA
Codec Engine
DSPLink Library
Image Codecs
Video Codecs
Audio Codecs
Speech Codecs
DSPLink IPC
CMEM Module
Platform Support Package
Framework Component
Linux Kernel
DSP/BIOS RTOS
DSP
ARM

DMAI

- Konszistens szoftver interfész
  - The DMAI library provides a simple software interface, but implements complex device driver and codec engine handshaking under the hood
  - Enables all ***VISA codecs***, reducing the need to understand details and differences of xDM version
  - Low-level details are managed by PSP(Platform Support Package) driver
- Modul gyűjtemény

DMAI

- DMAI simplifies the functions of Codec Engine APIs
  - Create() example
    - VIDDEC_ create() →Vdec_create()
    - Vdec_create (Engine_Handle hEngine, Char *codecName, VIDDEC_Params *params, VIDDEC_DynamicParams *dynParams)
      - Allocate the memory space of buffer and handle
      - VIDDEC_create()
      - VIDDEC_control():set the codec initialization parameters
  - Process() example
    - VIDDEC_ process() →Vdec_process()
    - Vdec_process (Vdec_Handle hVd, Buffer_Handle hInBuf, Buffer_Handle hDstBuf)
      - Recode the buffer information:number of in/out buffer
      - VIDDEC_process()
      - Control the buffer overflow

Kérdések?

KÖSZÖNÖM A FIGYELMET!

Források

- http://omappedia.org/wiki/DSPBridge_Project
- http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/bios/sysbios/6_34_01_14/index_FDS.html
- http://www.ti.com/tool/linuxdvsdk-dv
- http://processors.wiki.ti.com/index.php/CMEM_Overview
- http://tidsp.es.ncku.edu.tw/cinfon/resource/slides/10132012-01-final.pdf