

# DaVinci Software Architecture

Cui Jing  
Catalog DSP FAE  
Texas Instruments

**Minds in Motion**



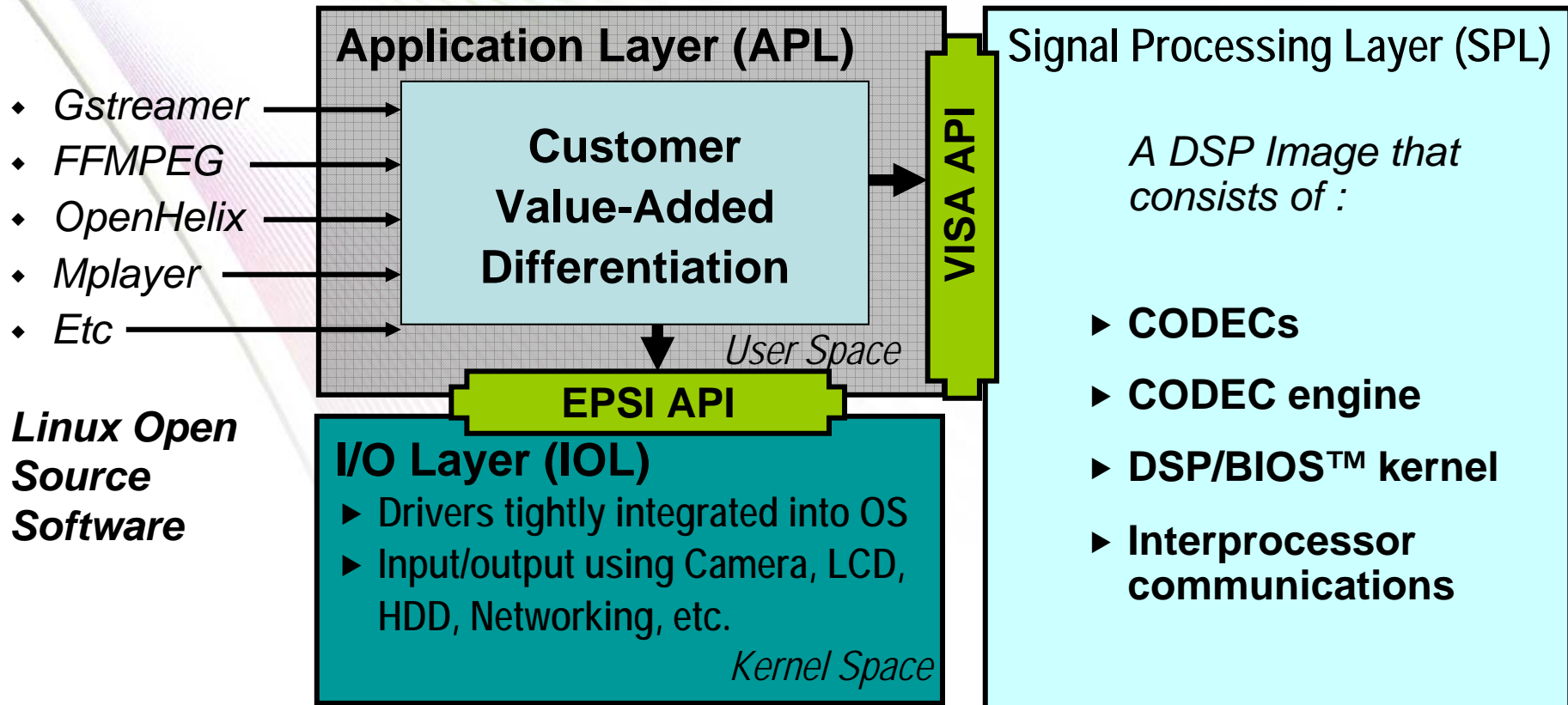
# DaVinci Technology S/W Architecture

- ◆ *DaVinci S/W Overview*
- ◆ **Application layer**
  - ◆ VISA (SP Layer) interface
- ◆ **Signal processing layer**
  - ◆ xDAIS (eXpressDSP™ Algorithm Standard)
  - ◆ xDM
- ◆ **CODEC engine details**
  - ◆ Local Call
  - ◆ RPC – Remote Procedure Call
  - ◆ Code review
- ◆ **DM643x S/W Architecture**

Minds in Motion



# TMS320DM644x Processor Software Overview



◆ **DaVinci technology software development is divided into three areas:**

1. Application layer
2. I/O layer (drivers)
3. Signal processing layer

◆ **Goal: abstract details of SPL and IOL so application programmer can quickly author a differentiated system**

Minds in Motion



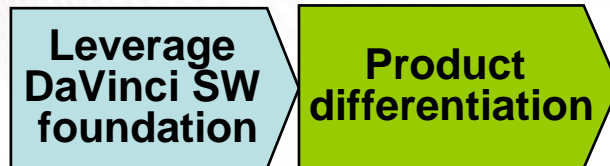
# Goal: Accelerated Time to Market



## Standard multimedia product development



The DaVinci™  
Effect



## Shorter development cycle and/or



Minds in Motion

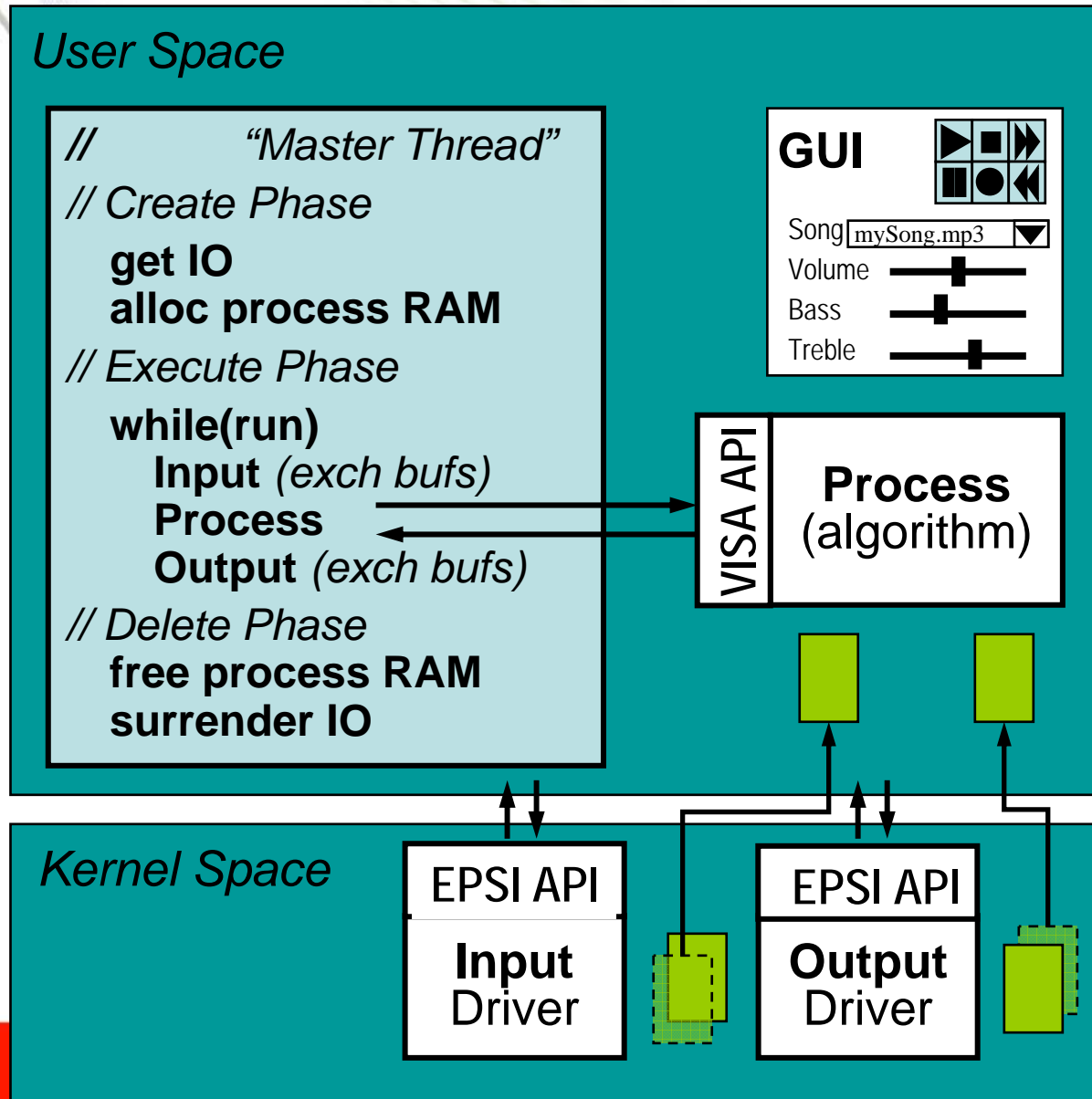
# DaVinci<sup>TM</sup> Technology S/W Architecture

- ◆ **DaVinci S/W Overview**
- ◆ *Application layer*
  - ◆ VISA (SP Layer) interface
- ◆ **Signal processing layer**
  - ◆ xDAIS (eXpressDSP<sup>TM</sup> Algorithm Standard)
  - ◆ xDM
- ◆ **CODEC engine details**
  - ◆ Local
  - ◆ RPC – Remote Procedure Call
  - ◆ Code review
- ◆ **DM643x S/W Architecture**

Minds in Motion



# DaVinci APL API: VISA and EPSI



## VISA API

- ♦ create
- ♦ **process**
- ♦ **control**
- ♦ delete

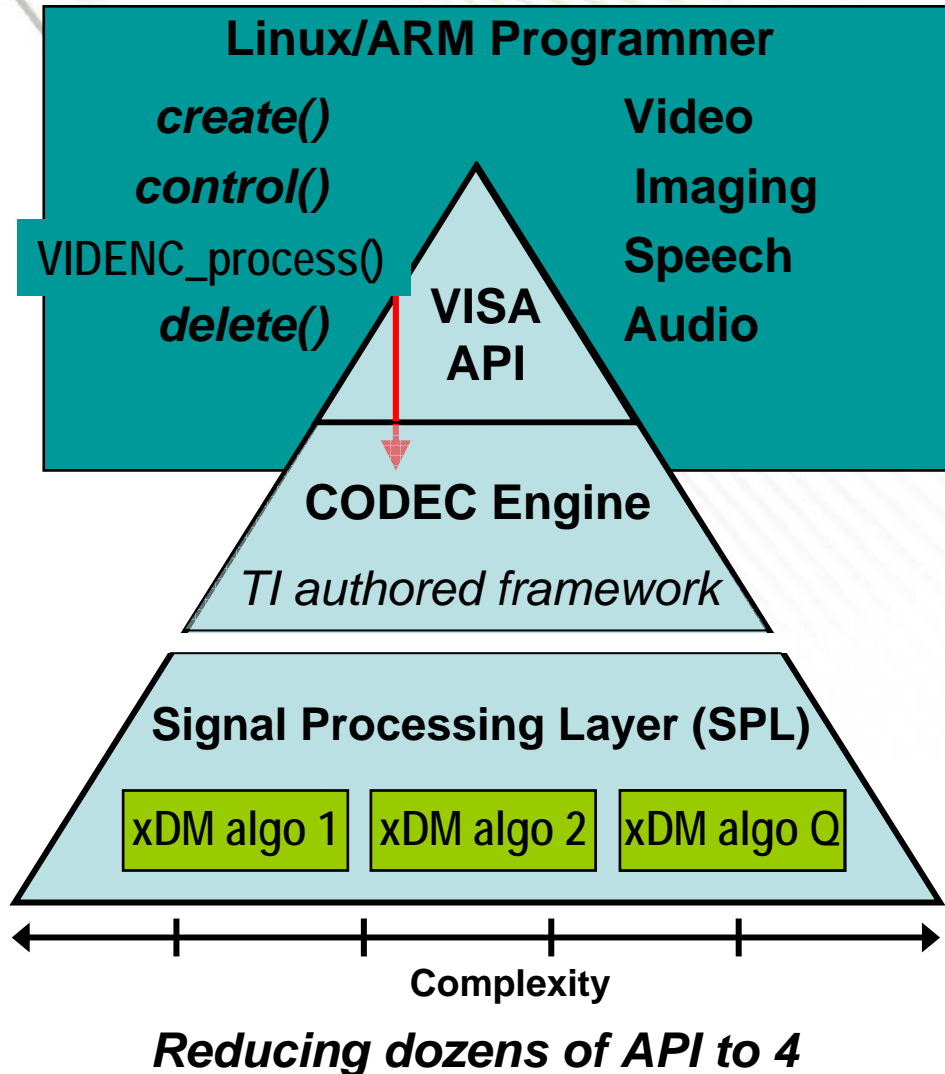
## EPSI API

- ♦ open
- ♦ **read**
- ♦ **write**
- ♦ ioctl
- ♦ close

VISA = Video, Imaging, Speech, Audio

EPSI = Easy Peripheral Software Interface

# VISA – SPL Interface



- ◆ Complexities of Signal Processing Layer (SPL) are abstracted into four functions:
 

<code>_create</code>	<code>_delete</code>
<code>_process</code>	<code>_control</code>
- ◆ VISA = 4 processing domains :  
Video Imaging Speech Audio
- ◆ Separate API set for encode and decode thus, a total of 8 API classes:
 

<b>VIDENC</b>	<b>IMGENC</b>	<b>SPHENC</b>	<b>AUDENC</b>
<b>VIDDEC</b>	<b>IMGDEC</b>	<b>SPHDEC</b>	<b>AUDDEC</b>
- ◆ TI's CODEC engine (CE) provides abstraction between VISA and algorithms
- ◆ Application programmers can purchase xDM algorithms from TI third party vendors  
... or, hire them to create complete SPL soln's
- ◆ Alternatively, experienced DSP programmers can create xDM compliant algos (discussed next)
- ◆ *Author your own algos or purchase depending on your DSP needs and skills*

Minds in Motion



# Master Thread Key Activities

```
idevfd = open("/dev/xxx", O_RDONLY);
ofilefd = open("./fname", O_WRONLY);
ioctl(idevfd, CMD, &args);
myCE = Engine_open("vcr", myCEAttrs);
myVE = VIDENC_create(myCE, "videnc", params);

while( doRecordVideo == 1 ) {
    read(idevfd, &rd, sizeof(rd));
    VIDENC_control(myVE, ...);
    VIDENC_process(myVE, ...);
    write(ofilefd, &wd, sizeof(wd));
}
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
```

*Note: the above pseudo-code does not show double buffering, often essential in R/T systems!*

**Minds in Motion**



# DaVinci<sup>TM</sup> Technology S/W Architecture

- ◆ **DaVinci S/W Overview**
- ◆ **Application layer**
  - ◆ VISA (SP Layer) interface
- ◆ ***Signal processing layer***
  - ◆ xDAIS (eXpressDSP<sup>TM</sup> Algorithm Standard)
  - ◆ xDM
- ◆ **CODEC engine details**
  - ◆ Local Call
  - ◆ RPC – Remote Procedure Call
  - ◆ Code review
- ◆ **DM643x S/W Architecture**

Minds in Motion



# eXpressDSP™ Algorithm Standard (xDAIS) API

- ◆ xDAIS provides the ability to manage memory resources for all compliant algos with a consistent API
- ◆ xDAIS does not attempt to define the API for the processing function, nor the algo creation parameters or control API
- ◆ Each algo author extends IALG with their own chosen “IMOD”

## ***Required Algo Functions (IALG) – Common across all algos***

```

Int    (*algNumAlloc)    (Void)
Int    (*algAlloc)      (const IALG_Params *, struct IALG_Fxns **, IALG_MemRec *);
Int    (*algInit)       (IALG_Handle, const IALG_MemRec *, IALG_Handle, const IALG_Params *);
Void   (*algActivate)   (IALG_Handle);
Void   (*algDeactivate) (IALG_Handle)
Void   (*algMoved)      (IALG_Handle, const IALG_MemRec *, IALG_Handle, const IALG_Params *)
Int    (*algFree)       (IALG_Handle, IALG_MemRec *)
    
```

## ***Processing Functions (IMOD) – Unique to a given algo from a given author***

```

#include <ialg.h>
typedef struct ITC_Fxns { // "ITC" extends IALG with 'runTC' fxn
    IALG_Fxns ialg;
    Void (*runTC)(ITC_Handle handle, XDAS_Int16 in[],XDAS_Int16 out[]);
} ITC_Fxns;
    
```

Minds in Motion



# xDAIS for Digital Media “xDM”

- ◆ A simple extension of xDAIS (i.e. the iAlg interface.)
- ◆ Ensures algorithm will integrate into the CODEC engine

```
typedef struct IVIDENC_Fxns {
```

```
    IALG_Fxns  ialg;
```

```
    XDAS_Int32 (*process)( IVIDDEC_Handle, XDM_BufDesc *, XDM_BufDesc *,  
                           IVIDDEC_InArgs *, IVIDDEC_OutArgs *);
```

```
    XDAS_Int32 (*control)( IVIDDEC_Handle, IVIDDEC_Cmd,  
                           IVIDDEC_DynamicParams *, IVIDDEC_Status *);
```

```
} IVIDENC_Fxns;
```

Minds in Motion



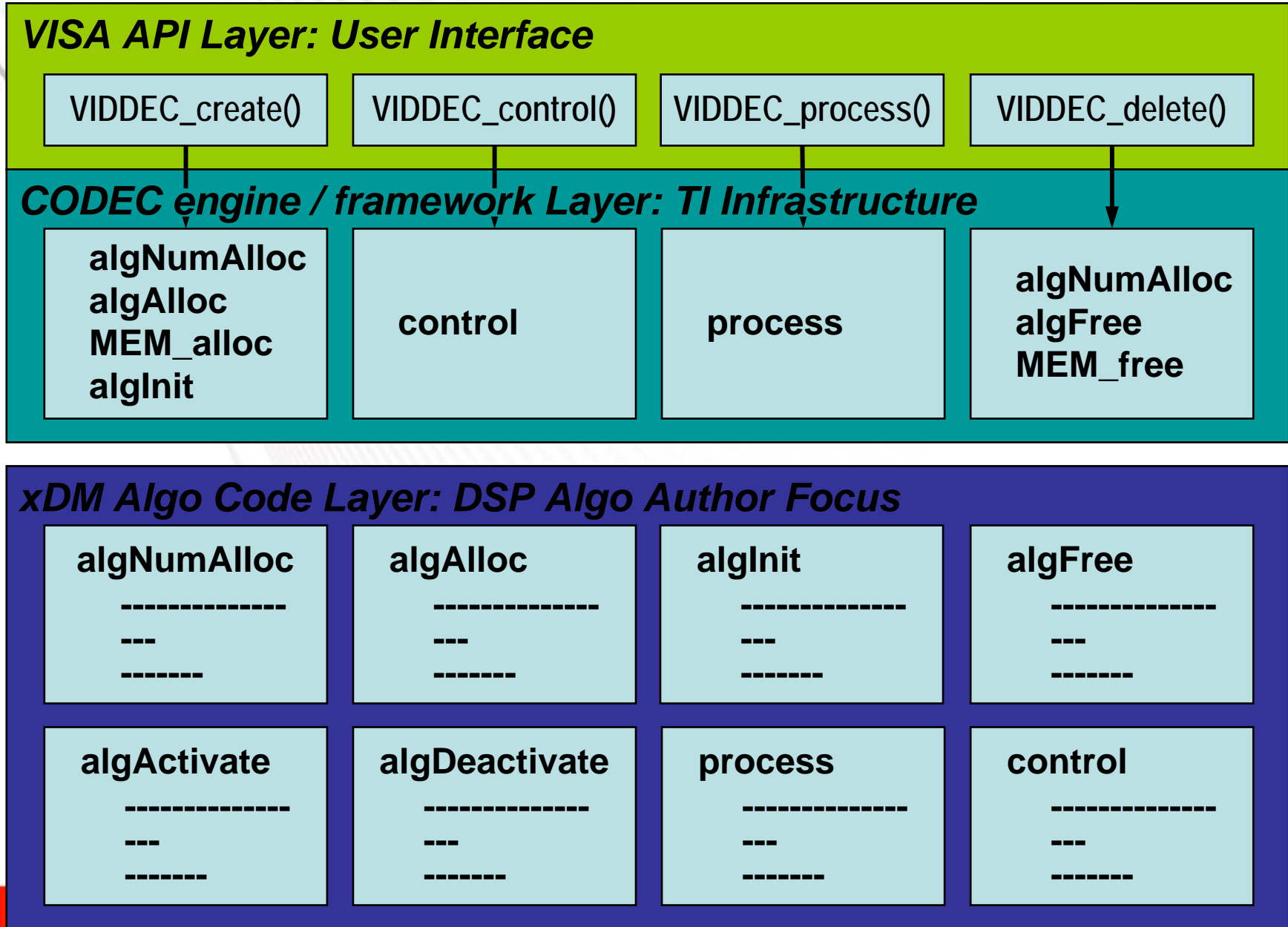
# Example xDM (Video Decoder)

## ***Required Algo Functions (IALG) – Common across all algos***

```
Void (*algActivate)    (IALG_Handle);
Int  (*algAlloc)      (const IALG_Params *, struct IALG_Fxns **, IALG_MemRec *);
Void (*algDeactivate) (IALG_Handle)
Int  (*algFree)       (IALG_Handle, IALG_MemRec *)
Int  (*algInit)       (IALG_Handle, const IALG_MemRec *, IALG_Handle, const IALG_Params *);
Void (*algMoved)      (IALG_Handle, const IALG_MemRec *, IALG_Handle, const IALG_Params *)
Int  (*algNumAlloc)   (Void)
```

## ***Processing Functions (IMOD) – Unique to a given algo from a given author***

```
Int  (*process)       (IVIDDEC_Handle handle, XDM_BufDesc *inBufs, XDM_BufDesc *outBufs,
                       IVIDDEC_InArgs *inargs, IVIDDEC_OutArgs *outargs);
Int  (*control)       (IVIDDEC_Handle handle, IVIDDEC_Cmd id,
                       IVIDDEC_DynamicParams *params, IVIDDEC_Status *status)
```



### VISA API Layer: User Interface

VIDDEC\_create()

VIDDEC\_control()

VIDDEC\_process()

VIDDEC\_delete()

CODEC engine is provided by TI  
You need only be concerned with  
VISA or xDM

### xDM Algo Code Layer: DSP Algo Author Focus

algNumAlloc

-----  
---  
-----

algAlloc

-----  
---  
-----

algInit

-----  
---  
-----

algFree

-----  
---  
-----

algActivate

-----  
---  
-----

algDeactivate

-----  
---  
-----

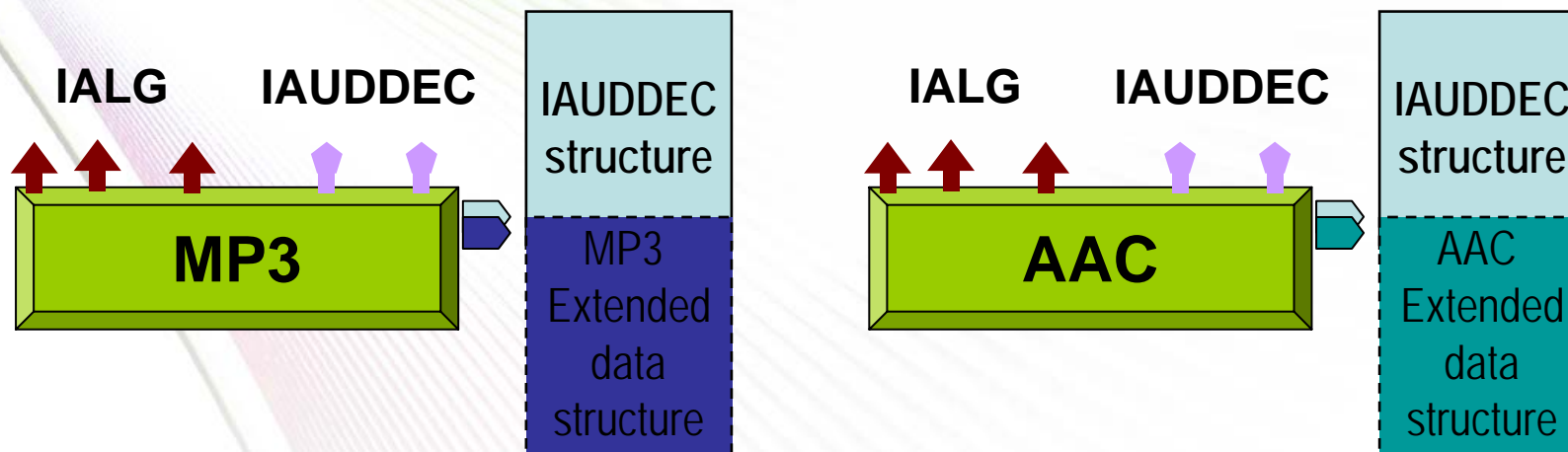
process

-----  
---  
-----

control

-----  
---  
-----

# Easily Switch xDM Components



- ◆ All audio class decoders (eg: MP3 & AAC ) provide the identical API
- ◆ Plug and Play: App using the IAUDDEC\_Structures can call all audio decoders
- ◆ Any algorithm specific arguments must be set to default values internally by the vendor (insulating the application from need to specify these parameters)
- ◆ Specific functionality can be invoked by the app using extended data structures
- ◆ To summarize:
  - Most authors can use the default settings of the extended features provided by vendors
  - “Power users” can (optionally) obtain further tuning via an algorithm extended structures

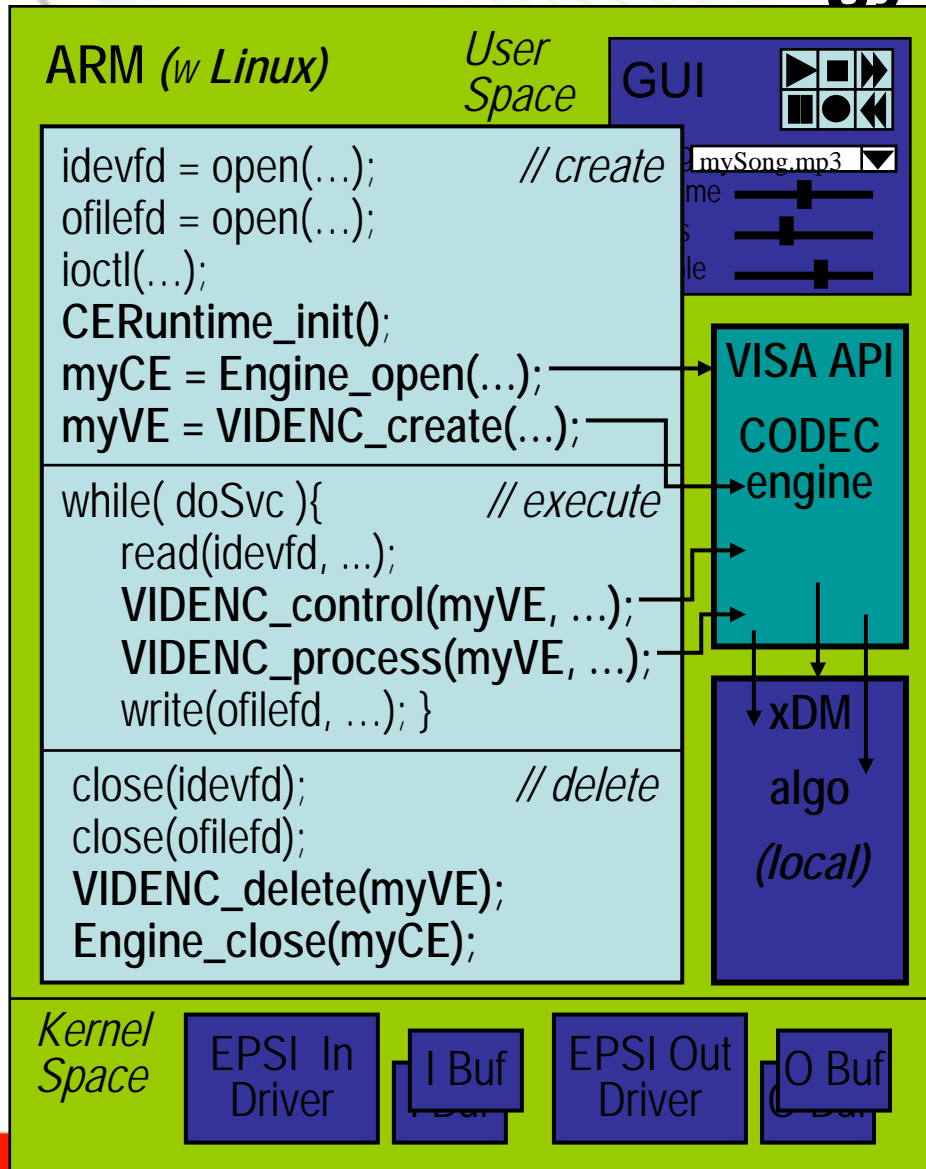
# DaVinci<sup>TM</sup> Technology S/W Architecture

- ◆ **DaVinci S/W Overview**
- ◆ **Application layer**
  - ◆ VISA (SP Layer) interface
- ◆ **Signal processing layer**
  - ◆ xDAIS (eXpressDSP<sup>TM</sup> Algorithm Standard)
  - ◆ xDM
- ◆ ***CODEC engine details***
  - ◆ Local Call
  - ◆ RPC – Remote Procedure Call
  - ◆ Code review
- ◆ **DM643x S/W Architecture**

Minds in Motion



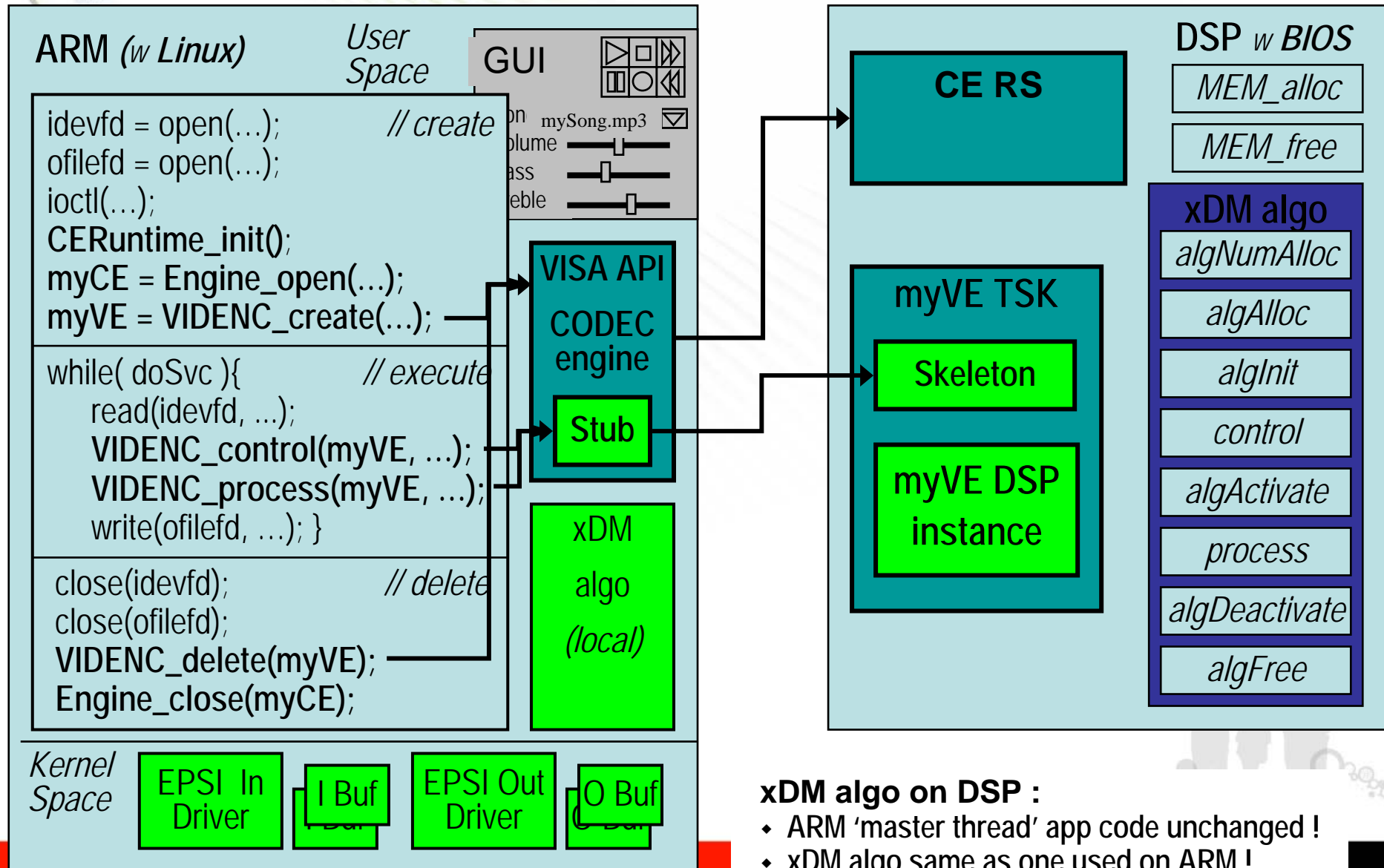
# DaVinci Technology Framework: ARM Only



- ◆ **xDM algo on ARM :**
  - ◆ Familiar layout to 1000's of Linux programmers
  - ◆ Optimal for low to medium demand algos
- ◆ **Will all algos run successfully on the ARM?**
  - ◆ MIPS
  - ◆ Power efficiency
  - ◆ Separate I/O interruptions from DSP processing
  - ◆ Non-determinism of GP OS's
- ◆ **So, in many cases, hard real-time high demand DSP work needs a DSP to implement the processing phase of the system**
- ◆ **How much extra work will be imposed on the application author to locate the xDM algo on DaVinci technology-based DSPs?**

Minds in Motion

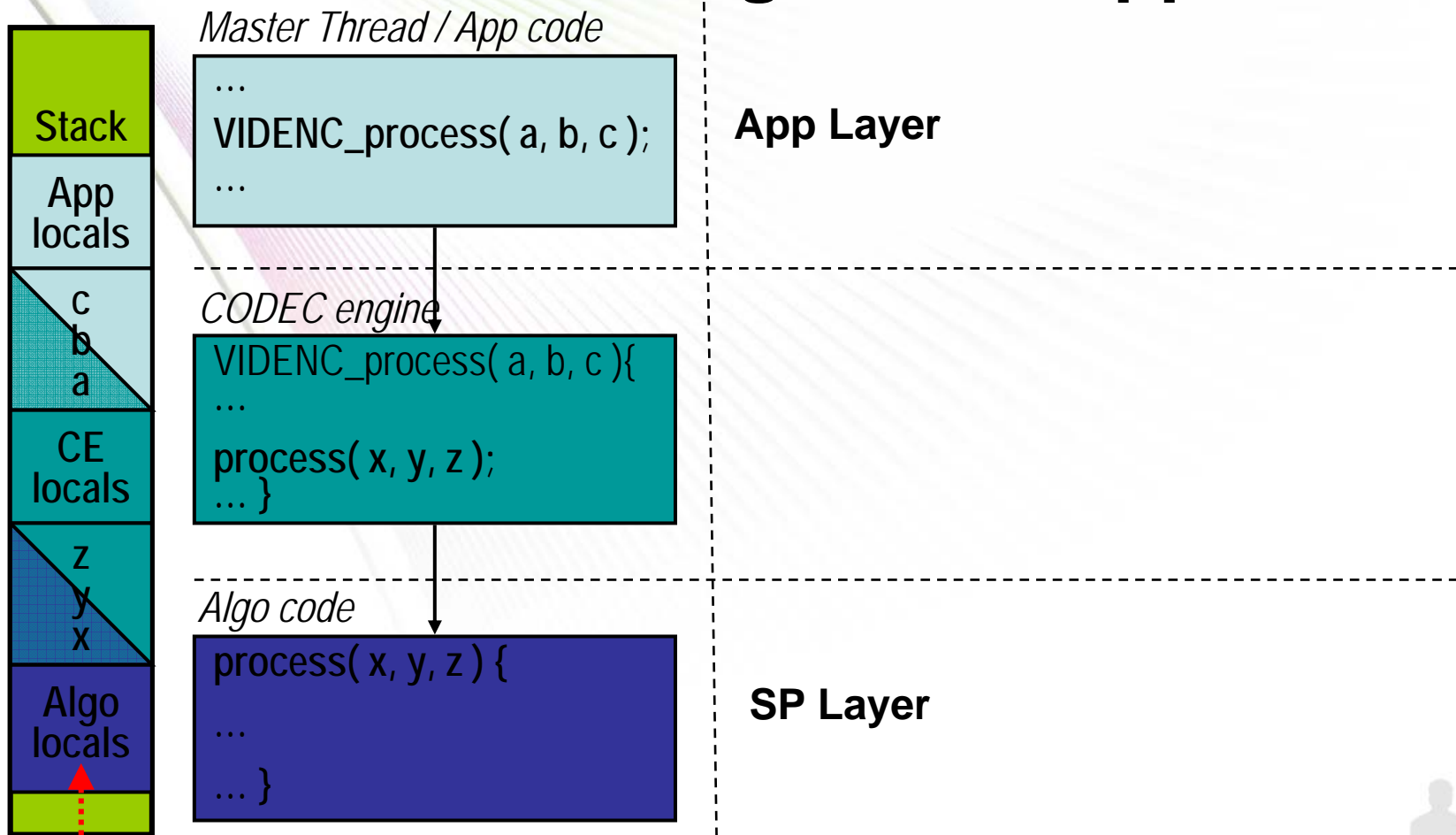
# DaVinci Technology Framework: ARM + DSP



## xDM algo on DSP :

- ARM 'master thread' app code unchanged !
- xDM algo same as one used on ARM !
- CODEC engine abstracts *all* detail from user !

# Local Call of Algo From Application

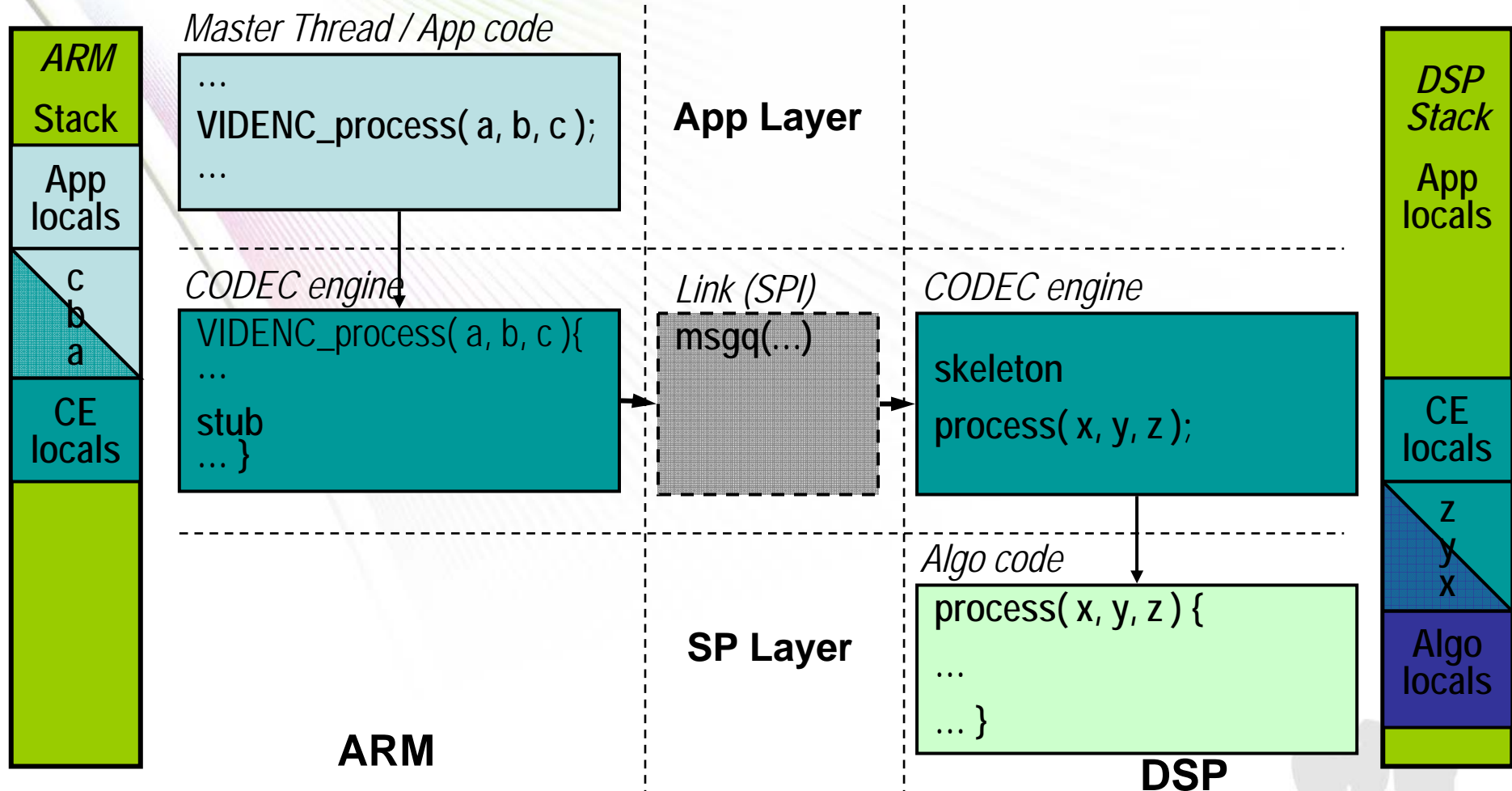


**ARM**  
*A local call is simply a function call – easy since all the data resides on the same CPU stack*

Minds in Motion



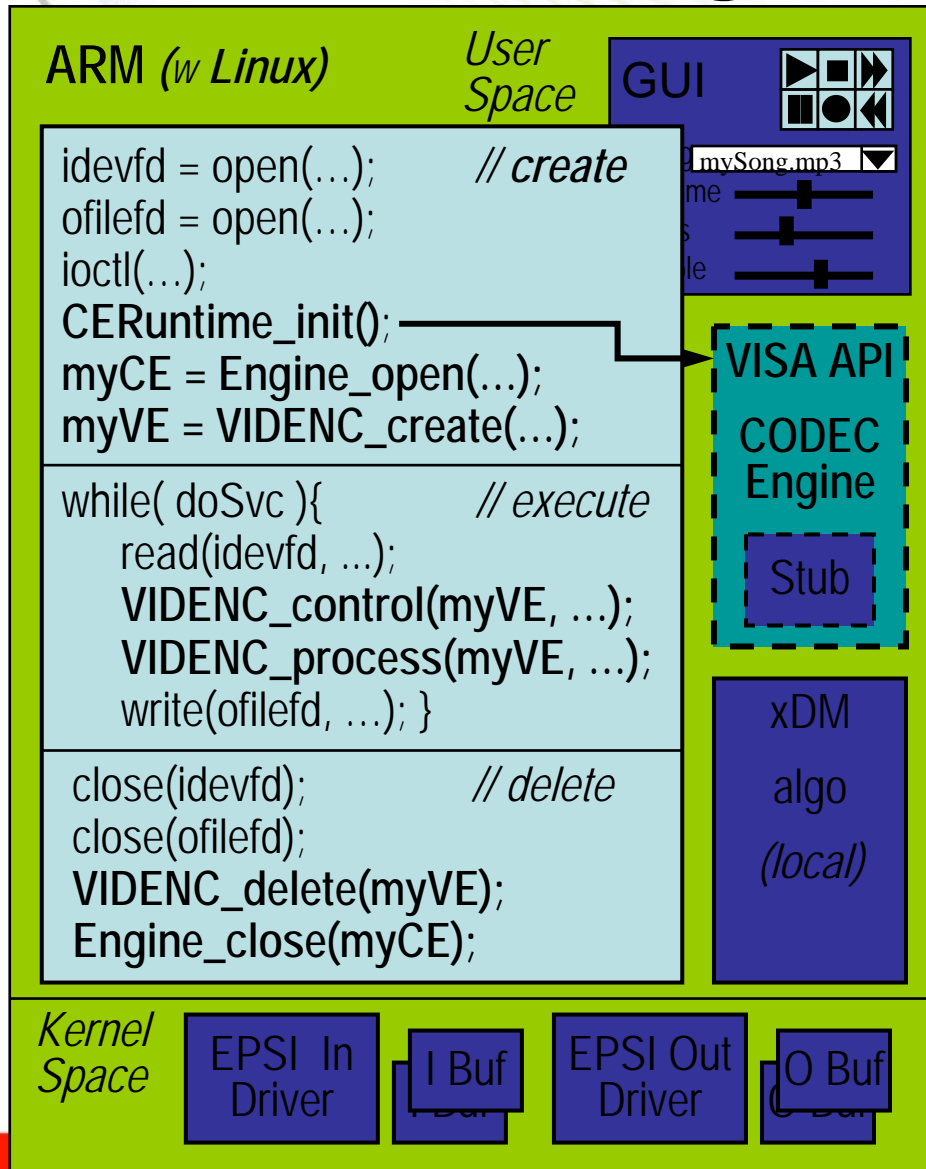
# Remote Procedure Call "RPC"



- ♦ *The CODEC engine abstracts remote calls*
- ♦ *Stub functions marshall (i.e. gather together) the required arguments*
- ♦ *Skeletons unpack args, call the algo on the remote processor*



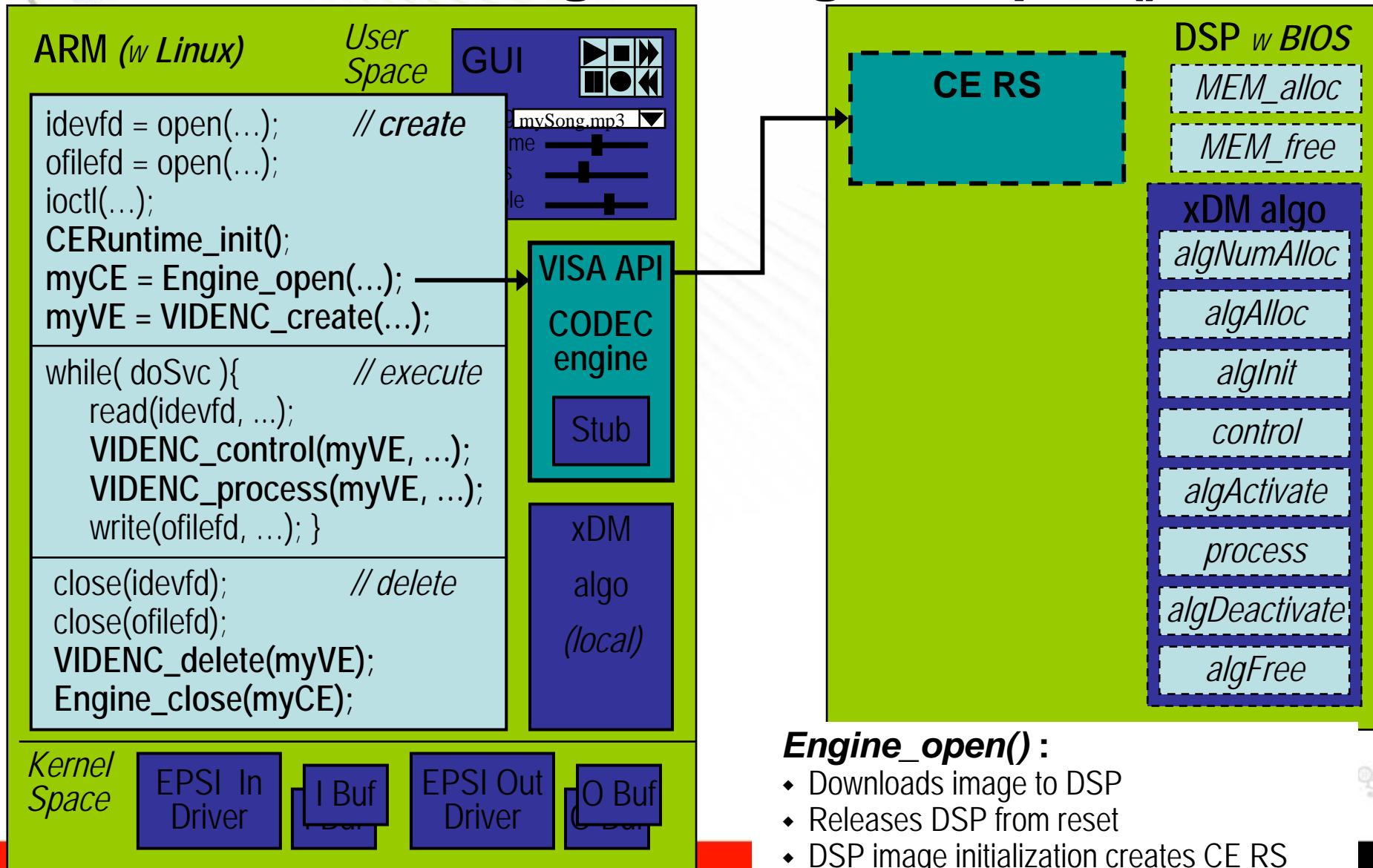
# CODEC Engine: CERuntime\_init()



## **CERuntime\_init() :**

- ◆ Create-phase activity
- ◆ Creates the CODEC engine thread
- ◆ Only needs to be done once in a system

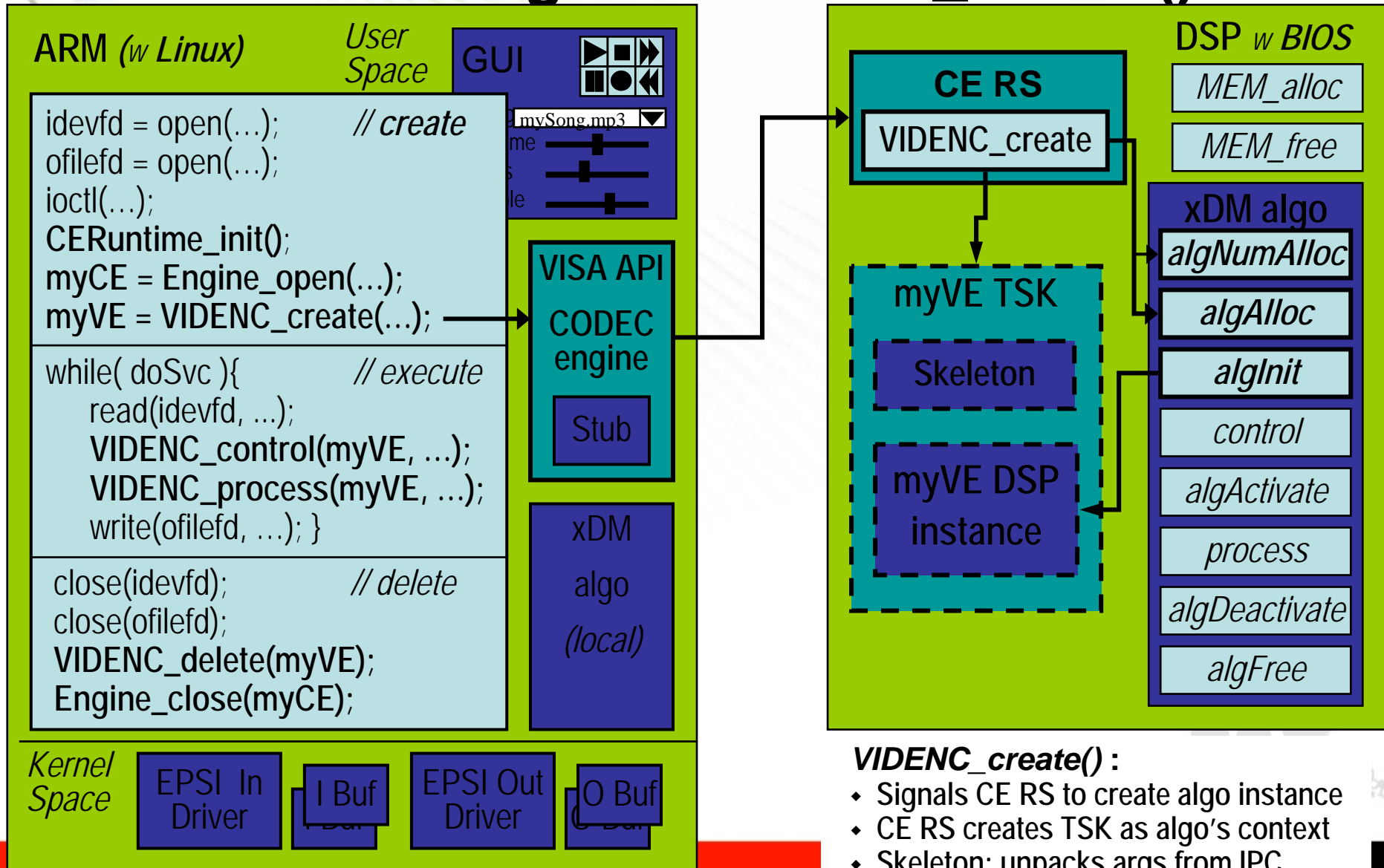
# CODEC Engine: Engine\_open()



## Engine\_open() :

- Downloads image to DSP
- Releases DSP from reset
- DSP image initialization creates CE RS

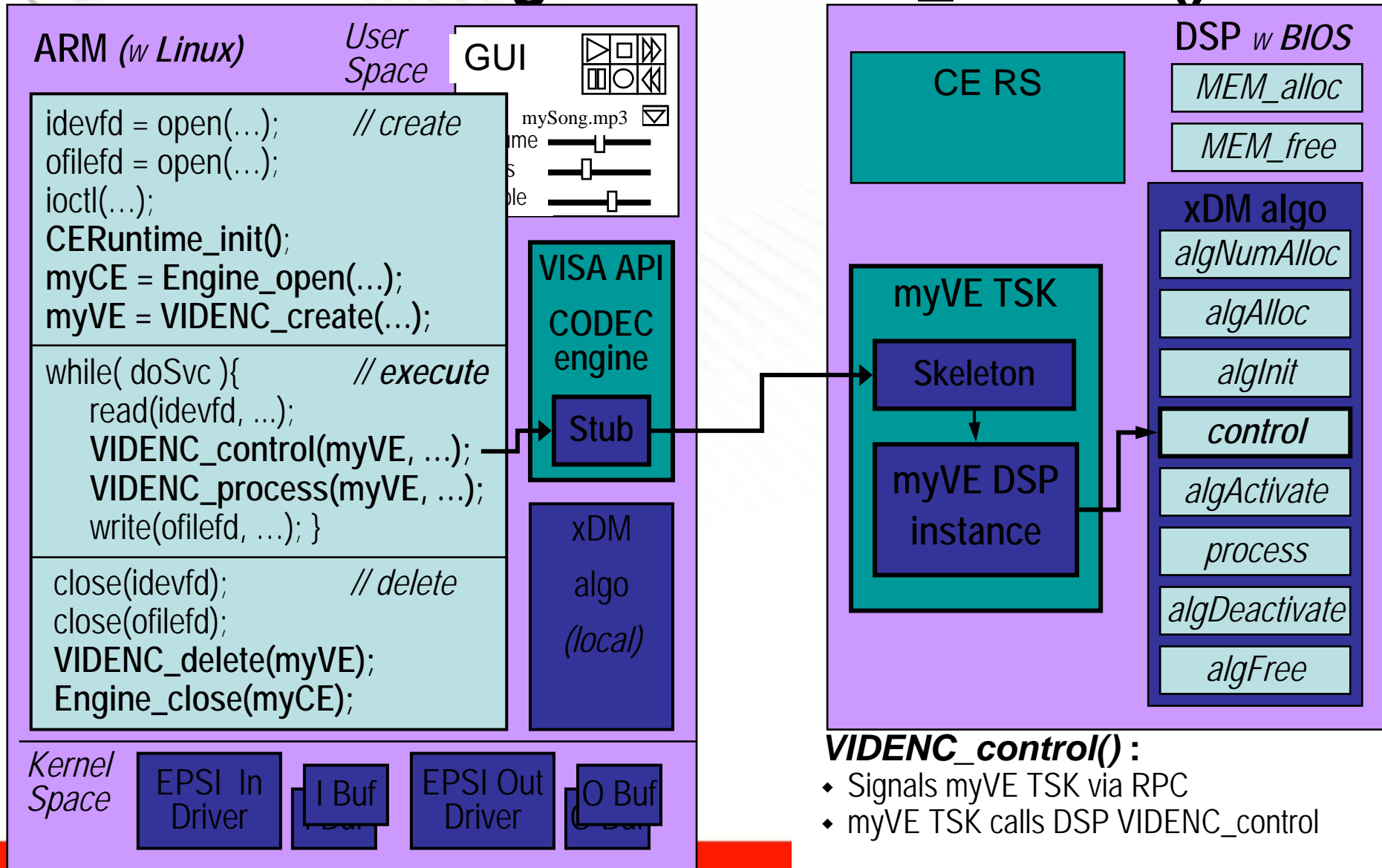
# CODEC Engine: VIDENC\_create()



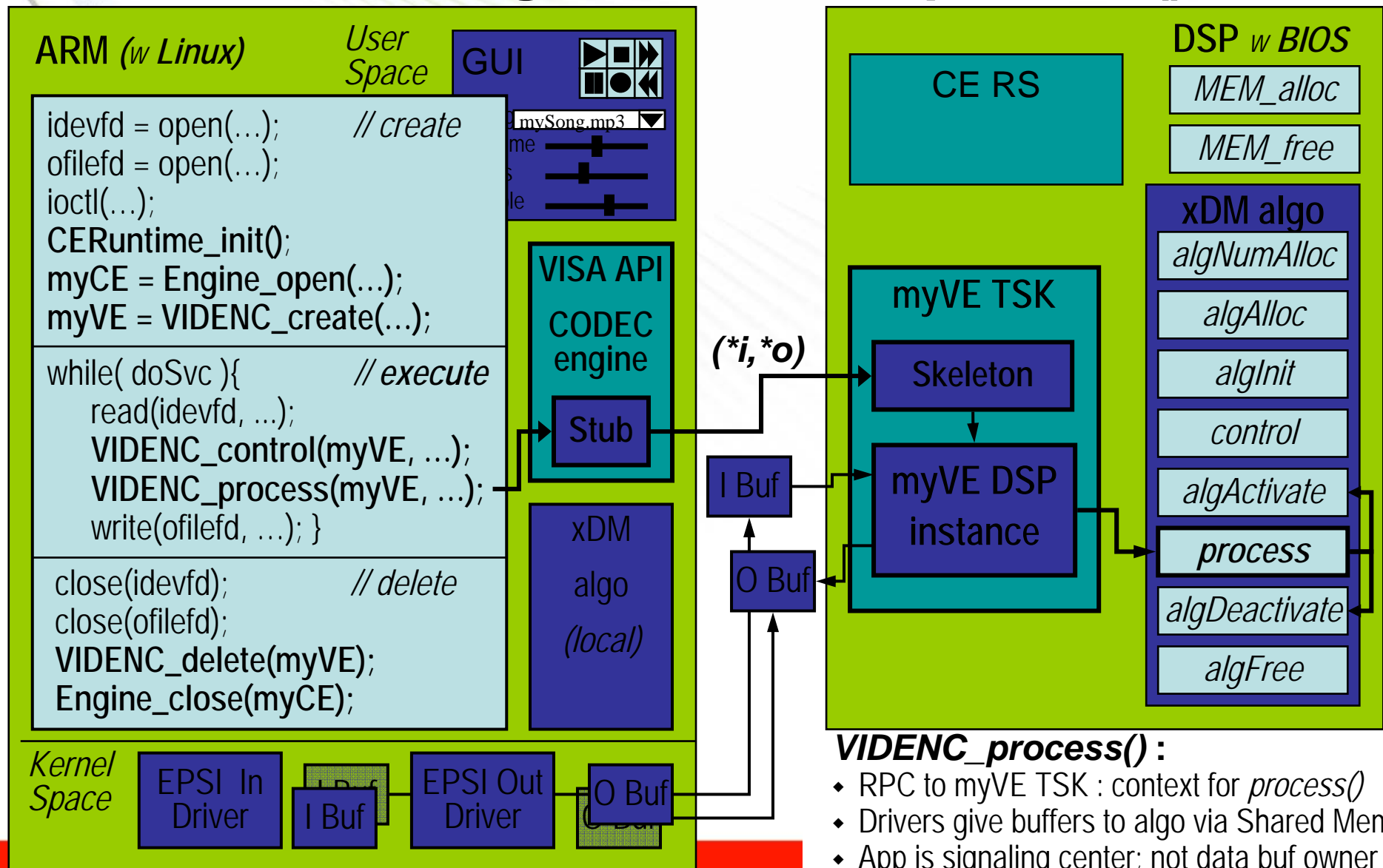
## VIDENC\_create() :

- Signals CE RS to create algo instance
- CE RS creates TSK as algo's context
- Skeleton: unpacks args from IPC

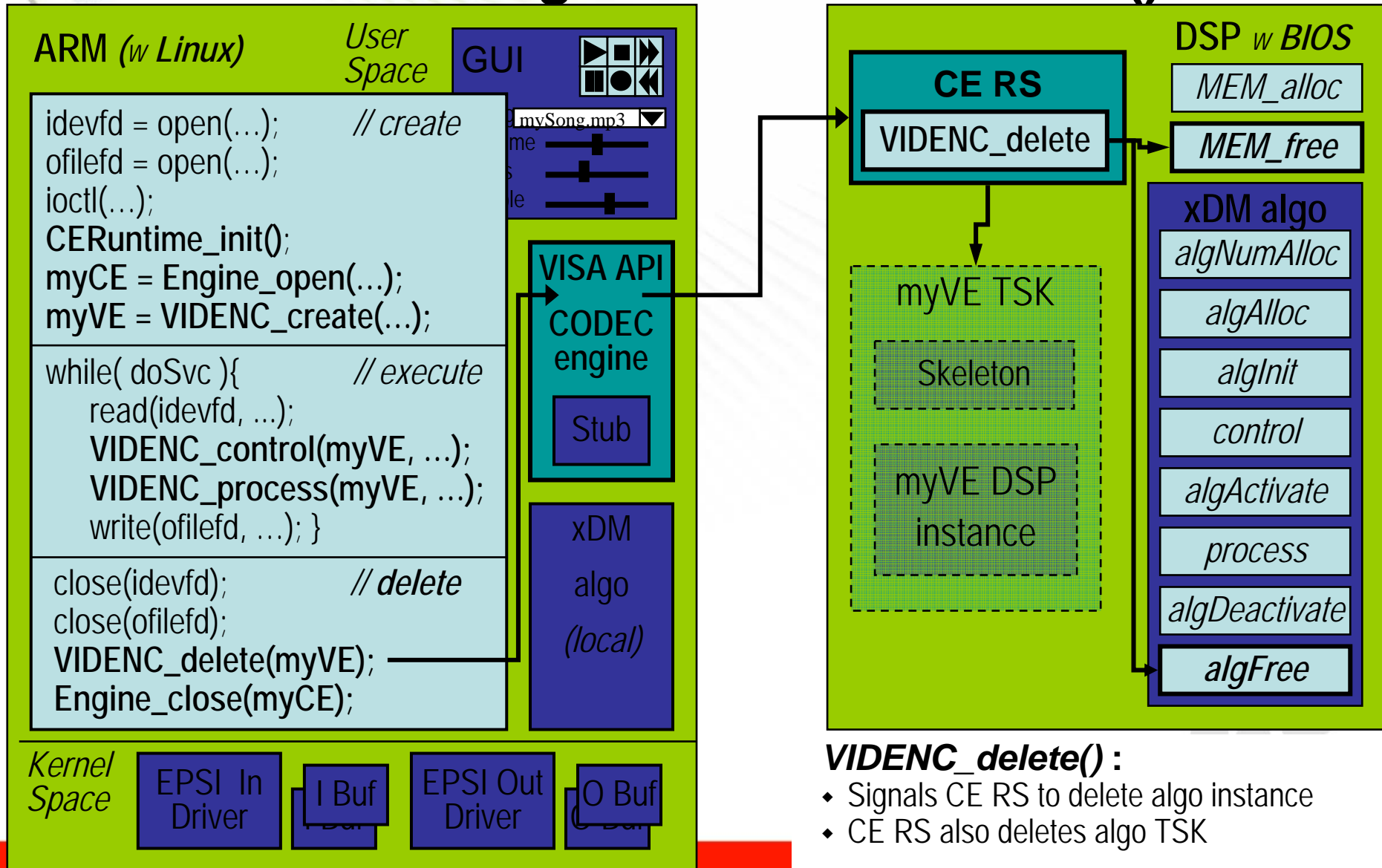
# CODEC Engine: VIDENC\_control()



# CODEC Engine: VIDENC\_process()



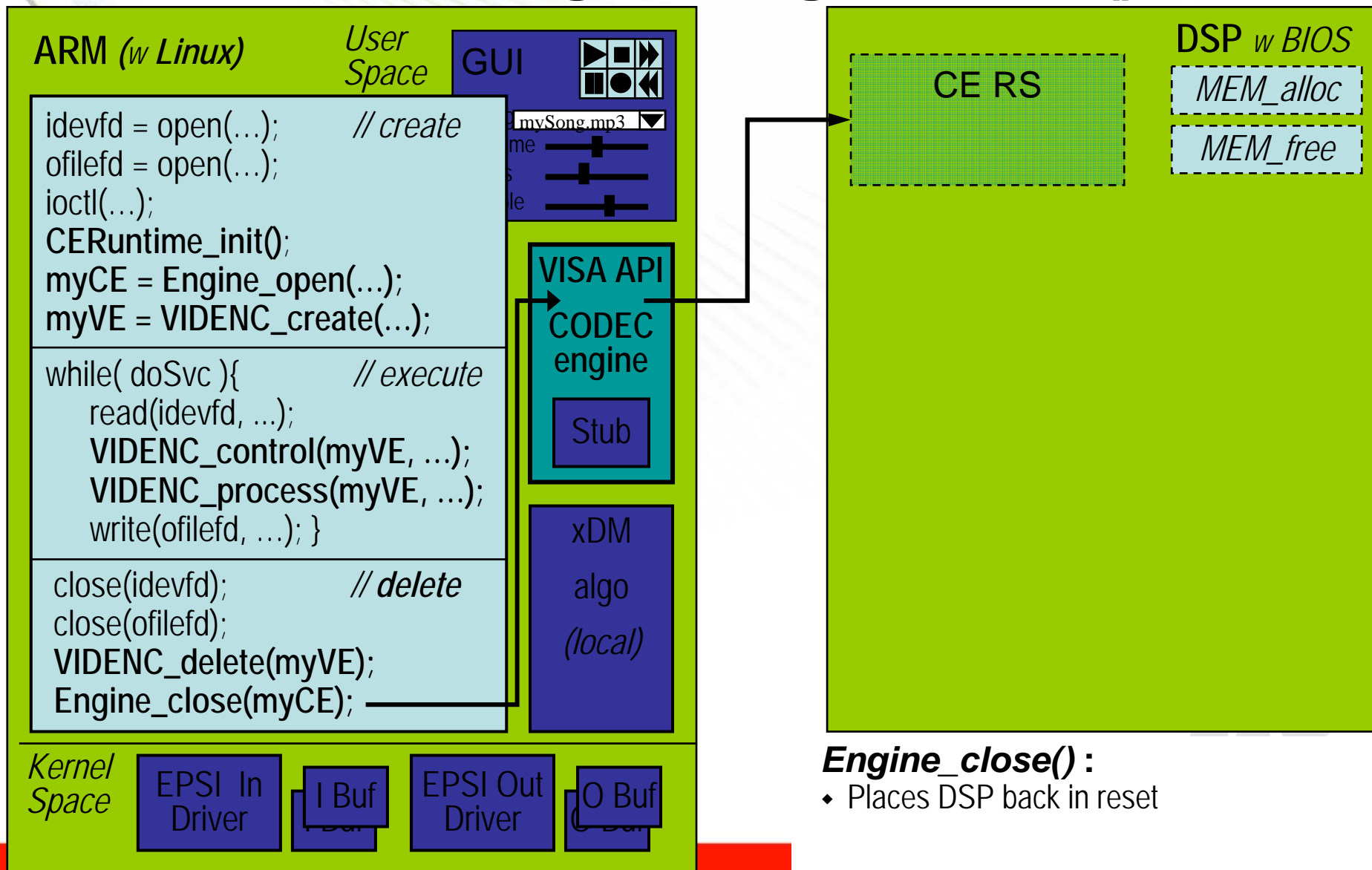
# CODEC Engine: VIDENC\_delete()



## VIDENC\_delete() :

- Signals CE RS to delete algo instance
- CE RS also deletes algo TSK

# CODEC Engine: Engine\_close()



# DaVinci<sup>TM</sup> Technology S/W Architecture

- ◆ **DaVinci S/W Overview**
- ◆ **Application layer**
  - ◆ VISA (SP Layer) interface
- ◆ **Signal processing layer**
  - ◆ xDAIS (eXpressDSP<sup>TM</sup> Algorithm Standard)
  - ◆ xDM
- ◆ **CODEC engine details**
  - ◆ Local Call
  - ◆ RPC – Remote Procedure Call
  - ◆ Code review
- ◆ *DM643x S/W Architecture*

Minds in Motion

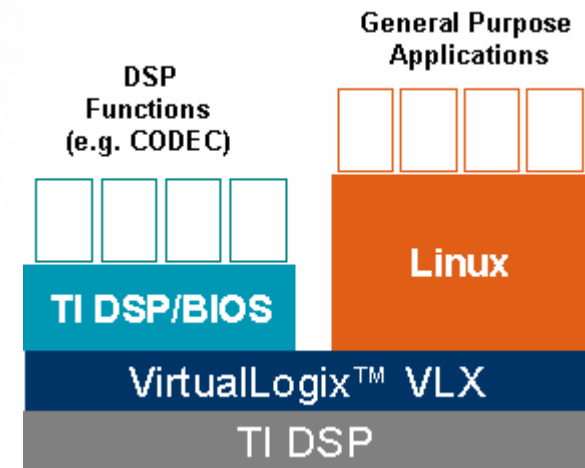


# DaVinci-based Solution

- **TI Dual-Core ARM+DSP (e.g. DM644x):**
  - Addresses both general purpose processing (Linux) and signal processing (DSP/BIOS) that is mentioned above

- **DM643x + VLX**

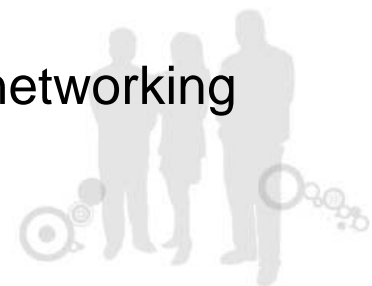
- VLX provides a virtual Linux-host processor on a single-core DSP allowing the silicon to support both Linux functions and DSP/BIOS functions (similar to DM644x)



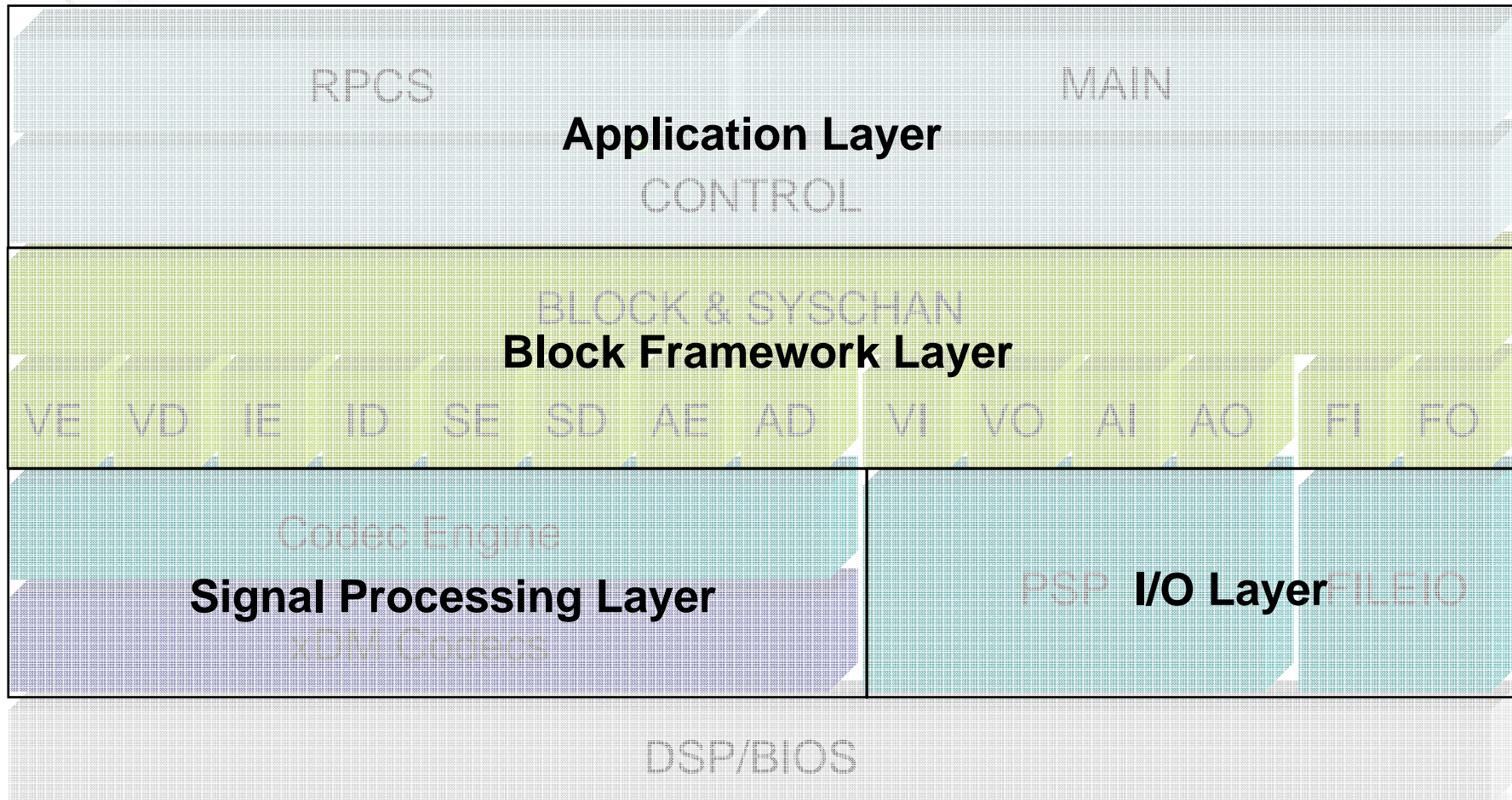
- **TI Single-Core DSP (e.g. DM643x)**

- Addresses signal processing (DSP/BIOS) with light networking (NDK) requirements

Minds in Motion



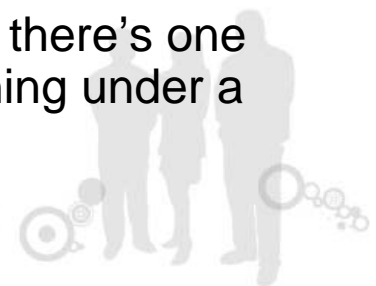
# DM643x Ref App SW Stack



# Differences vs Multi-processor

- DM643x VISA calls are similar to local codec calls on the DM644x SOC
- Local codecs run in the same thread (and thread context) from where VISA call is made
  - When using remote codecs, server manages the codec's thread context (declared in .cfg)
  - But, with local codecs, the application programmer must manage thread context, specifically:
    - Thread priority
    - Resource sharing (scratch memory and DMA channels)
    - Stack size (Remote codecs run in their own TSK, so there's one stack per algo; now, there might be many algos running under a single TSK)

Minds in Motion



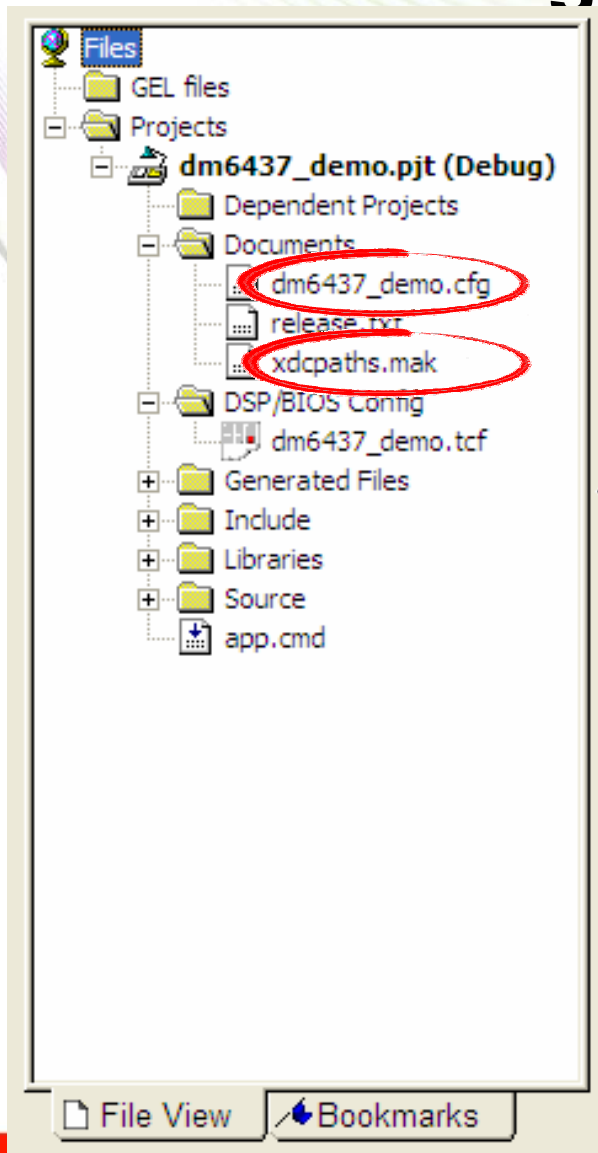
# Running Everything on the DSP

- While algo is running, there are likely to be multiple application-layer interrupts, therefore preemptions may occur during codec processing
- Cache Management:
  - Cache management is automatically handled for remote VISA calls by the Codec Engine (within RPC's stubs and skeletons on DM644x).
  - Since local function calls do not use stubs and skeletons, there is no "automatic" cache management for local VISA calls made on DM643x
  - Though, the DM643x Block Architecture handles cache management for you (i.e. cache is managed if you use the example applications blocks, but not handled for you if you don't use these blocks)

Minds in Motion



# Codec Engine Configuration



- ◆ app.cfg configures packaged content for the executable, similarly to DM6446 builds.
- ◆ xdcpaths.dat provides repository search paths for all packages used in app.cfg
- ◆ app.tcf is a standard BIOS tconf file. It's name and path must match the .cfg file used in the project
- ◆ RTSC build mechanism is natively supported within CCS for BIOS 5.31 (and later) installations. Users do not need to specify package.xdc, package.bld or config.bld
- ◆ Users may also develop their own RTSC-configurable components (such as codecs) for use in app.cfg

Minds in Motion



```
/*
 * ===== video_encdec.cfg =====
 */

/* set up OSAL */
var osalGlobal = xdc.useModule('ti.sdo.ce.osal.Global');
osalGlobal.runtimeEnv = osalGlobal.DSPBIOS;

osalGlobal.defaultMemSegId = "DDR2";
Program.main = Program.system = null;

/* Import Log Server support: */
xdc.useModule('ti.bios.utils.Load');
var LogTrack = xdc.useModule('ti.bios.log.support.LogTrack');
xdc.useModule('ti.bios.log.ndk.LogServerCgi');
LogTrack.createLogSize = 4096;
LogTrack.trackLogSize = 65536;

/* get various codec modules; i.e., implementation of codecs */
var H264ENC = xdc.useModule('codecs.h264enc.H264ENC');
var H264DEC = xdc.useModule('codecs.h264dec.H264DEC');

/*
 * ===== Engine Configuration =====
 */

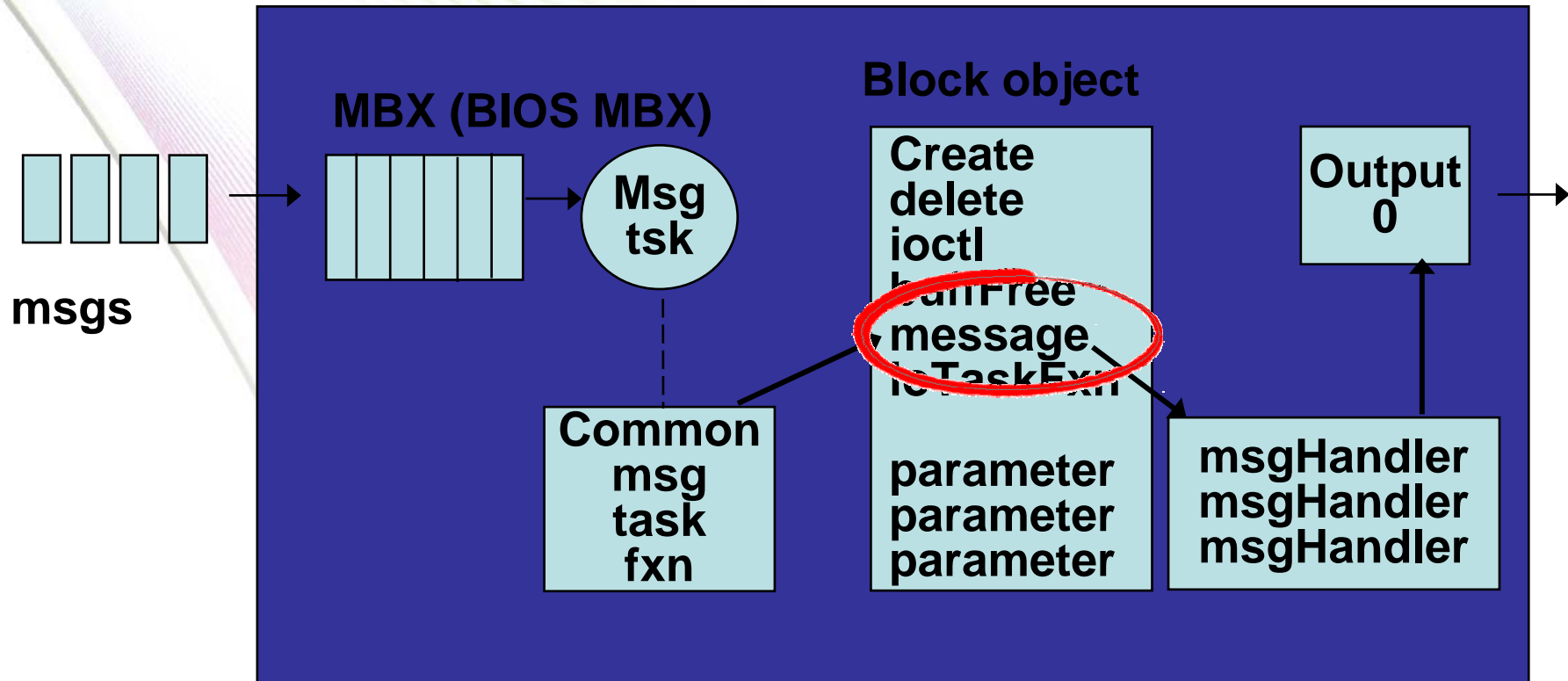
var Engine = xdc.useModule('ti.sdo.ce.Engine');
var vcr = Engine.create("encdec", [
    {name: "h264enc", mod: H264ENC, groupId:0, local: true},
    {name: "h264dec", mod: H264DEC, groupId:0, local: true},
]);
```

# Case Study - Video Encoder Block

Minds in Motion



# Block Structure



When a MBX message is received by the block's Msg task, it is passed to the block's message function for routing to the proper message handler.

Minds in Motion

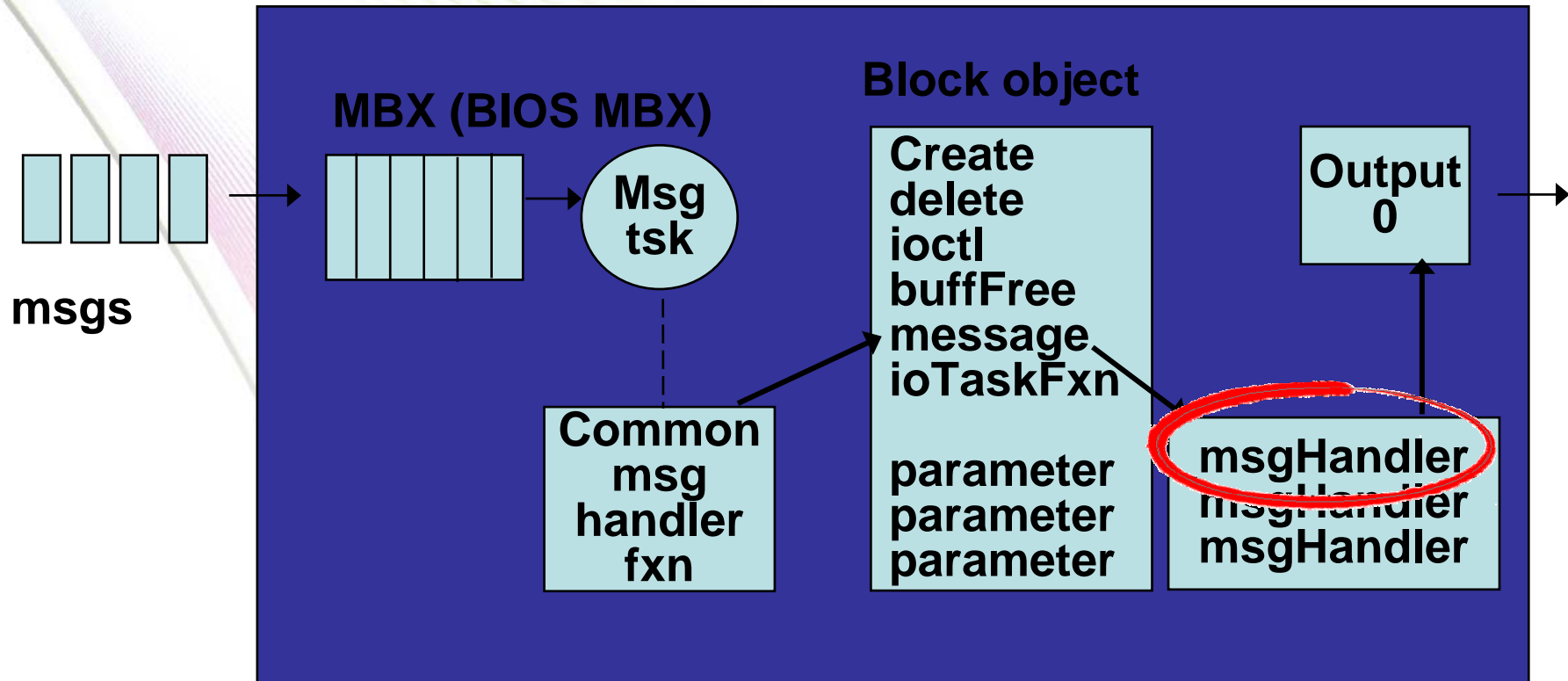


# Block Example : LOCAL\_msgHandler

```
static int LOCAL_msgHandler(APP_BLOCK_Handle hBlock, APP_BLOCK_MsgObj *msgPtr) {  
  
    /* dispatch the message to the corresponding message handler, return 0 if not handled */  
    switch (msgPtr->msgId) {  
        case APP_BLOCK_MSGID_SET_PARAMS: return LOCAL_msgSetParams(hBlock,msgPtr);  
        case APP_BLOCK_MSGID_STARTUP:    return LOCAL_msgStartup(hBlock,msgPtr);  
        case APP_BLOCK_MSGID_OPEN:       return LOCAL_msgOpen(hBlock,msgPtr);  
        case APP_BLOCK_MSGID_PLAY:       return LOCAL_msgPlay(hBlock,msgPtr);  
        case APP_BLOCK_MSGID_PAUSE:      return LOCAL_msgPause(hBlock,msgPtr);  
        case APP_BLOCK_MSGID_RESUME:     return LOCAL_msgResume(hBlock,msgPtr);  
        case APP_BLOCK_MSGID_STOP:       return LOCAL_msgStop(hBlock,msgPtr);  
        case APP_BLOCK_MSGID_CLOSE:      return LOCAL_msgClose(hBlock,msgPtr);  
        case APP_BLOCK_MSGID_RAW_VIDEO:  return LOCAL_msgRawVideo(hBlock,msgPtr);  
        default: return 0;  
    }  
}
```

Minds in Motion

# Block Structure



The message handler calls the block's underlying codec's VISA function to process the input buffer into an output buffer.

The output buffer is placed into an outgoing MBX message and sent to the next block in the data path.



```
static int LOCAL_msgRawVideo(APP_BLOCK_Handle hBlock, APP_BLOCK_MsgObj *msgPtr) {  
    APP_BLOCK_VIDEO_ENCODE_Stats *statsPtr = hBlock->stats;  
    LOCAL_InstData *self = (LOCAL_InstData *)hBlock->instData;  
    APP_BLOCK_BuffDescHandle hBuffDescOut, hBuffDescIn;  
    int postCnt = 0, status = 0;  
  
    /* create the output buffer */  
    hBuffDescOut = APP_BLOCK_buffAlloc(hBlock, 256, self->bytesPerFrame);  
    BCACHE_inv(hBuffDescOut->buffPtr, hBuffDescOut->buffSize, TRUE);  
  
    /* get the input buffer from the input message */  
    hBuffDescIn = msgPtr->hBuffDesc;  
    BCACHE_wbInv(hBuffDescIn->dataPtr, hBuffDescIn->dataSize, TRUE);  
  
    /* encode the frame */  
    status = LOCAL_encodeFrame(hBlock, hBuffDescIn->dataPtr, hBuffDescIn->dataSize,  
        hBuffDescOut->dataPtr, hBuffDescOut->buffSize);  
  
    /* setup the output buffer */  
    hBuffDescOut->dataSize = self->outArgs.bytesGenerated;  
    hBuffDescOut->bitsPerPixel = self->bitsPerPixel;  
    hBuffDescOut->pixelsPerLine = self->pixelsPerLine;  
    hBuffDescOut->linesPerFrame = self->linesPerFrame;  
    BCACHE_wbInv(hBuffDescOut->dataPtr, hBuffDescOut->dataSize, TRUE);  
  
    ...  
}
```

# Block Example : LOCAL\_encodeFrame

```
static int LOCAL_encodeFrame(APP_BLOCK_Handle hBlock, void *inBuff, int inSize, void *outBuff,
    int outSize) {
    LOCAL_InstData *self = hBlock->instData;
    int status = 0, result = 0;

    /* setup the codec input arguments */
    self->inBufPtrs[0] = inBuff;
    self->inBufSizes[0] = inSize;
    self->outBufPtrs[0] = outBuff;
    self->outBufSizes[0] = outSize;

    /* call the codec */
    result = VIDENC_process(self->hCodec, &self->inBufs, &self->outBufs, &self->inArgs, &self->outArgs);

    /* get the codec status */
    result = VIDENC_control(self->hCodec, XDM_GETSTATUS, &self->videncDynamicParams,
        &self->videncStatus);

    return status;
}
```

Minds in Motion

# TI|Developer Conference DVDP Directory Contents

File Edit View Favorites Tools Help

Back Search Folders

Address C:\dvsdk\_1\_00\_00\_12 Go

Name	Size	Type	Date Modified
biosutils_1_00_00_18		File Folder	2/15/2007 2:35 PM
ccs		File Folder	2/15/2007 2:35 PM
codec_combos_1_00		File Folder	2/15/2007 2:35 PM
codec_engine_1_20		File Folder	2/15/2007 2:41 PM
codecs		File Folder	2/15/2007 2:41 PM
dm643x_ais_utility		File Folder	2/15/2007 2:41 PM
dm6437_demo_0_90_00		File Folder	2/15/2007 2:42 PM
docs		File Folder	2/15/2007 2:42 PM
dvtb_1_00		File Folder	2/15/2007 2:42 PM
dvtb_host		File Folder	2/15/2007 2:42 PM
examples		File Folder	2/15/2007 2:43 PM
framework_components_1_20		File Folder	2/15/2007 2:46 PM
ndk_1_92_00_22_eval		File Folder	2/15/2007 2:46 PM
PSP_0_04_03		File Folder	2/15/2007 2:47 PM
xdais_5_20		File Folder	2/15/2007 2:50 PM
xdctools		File Folder	2/15/2007 2:55 PM
.installinfo	1,637 KB	INSTALLINFO File	2/15/2007 3:10 PM
README.txt	7 KB	Text Document	2/12/2007 6:49 PM
release_notes_DM6437.html	8 KB	HTML File	2/12/2007 6:49 PM
uninstall.exe	1,021 KB	Application	2/15/2007 3:08 PM
xdcpaths.mak	2 KB	Makefile	2/15/2007 3:09 PM

**File and Folder Tasks**

- Make a new folder
- Publish this folder to the Web
- Share this folder

**Other Places**

- Local Disk (C:)
- My Documents
- My Computer
- My Network Places

**Details**

**dvsdk\_1\_00\_00\_12**  
File Folder  
Date Modified: Today, February 15, 2007, 3:09 PM

# TI Developer Conference

19 - 26 June 2007 • Beijing • Shanghai • Shenzhen • Taipei

# Thanks!

**Minds in Motion**



Technology for Innovators™

 TEXAS INSTRUMENTS