TEXAS INSTRUMENTS

Application Report
Draft v0.3 – Oct 2017

Memory Map of Vision SDK

Shiju Sivasankaran *ADAS Software, Processor BU*

ABSTRACT

TDA2x and TDA3x series of automotive processor are designed to be used in automotive safety systems. TI Vision SDK is a multi-processor multi-channel software development platform, which enables the easy integration of new vision applications using different heterogeneous CPUs of TI ADAS SoCs. VISION SDK allows different usecases for different platforms and hence a generic memory map might not be sufficient for all users. This document gives insight on different sections of the memory map that a user can change for their usecases. The document discusses about the Vision SDK (VSDK) memory map implementation on TDA2xx, TDA2Ex, TDA2Px and TDA3xx

Contents

Figures

No table of figures entries found.

Tables

No table of figures entries found.

1. Introduction

It is expected that the user has gone through the below documentations to understand the hardware and software architecture/partitioning.

- TDAxx SoC architecture (TRM)
- Vision SDK (Links & chain) Framework UserGuides
  - VisionSDK_SW_Architecture_Details.pdf
  - VisionSDK_SW_Architecture_Overview.pdf
- VSDK Developer guides
  - VisionSDK_DevelopmentGuide.pdf
  - VisionSDK_Linux_DevelopmentGuide.pdf

2. Memory Sections of VISION SDK Memory Map

VISION SDK has multiple memory map configurations supported based on the usage scenarios and available total memory on various platforms. The total system memory is divided into various sub-sections/processors. The broad classification of the sections is listed below:

Shared regions: SRs are different memory partitions that are shared across processors

- SR0: Shared Region 0. This shared region is used to allocate memory for data structures needed for inter processor communication. This shared region is not cached on any of the processor cores.
- SR1_FRAME_BUFFER_MEM: This memory region is used for allocating data buffers for capturing video data, scaling, Alg processing & displaying video frames. Accessible & cached from all cores.
- SR2_MEM: Used only with VSDK Linux reserving memory for CMEM allocations (contiguous memory for Linux)
- SYSTEM_IPC_SHM_MEM: Non-Cached Memory section for keeping IPC link data structures

Log Mem:

- REMOTE_LOG_MEM: Non-Cached Memory section reserved and accessible from all processor cores to dump the debug/profile print messages. Each processor core uses VPS_printf() to dump the status/debug messages on this memory region. Remote Debug Client running on master cores (IPU1-0 for Bios and A15 in case of Linux) reads this memory region and prints the content on the (UART) console.
- LINK_STATS_MEM: Non Cached Memory section reserved and accessible from all processor cores to dump the Link statistics
- TRACE_BUF: Remote proc logs, non-Cached, used only with VSDK Linux

DDR Code/data:

- CODE_MEM: Partition for code section of each core's executable binary
- DATA_MEM: Partition for data section of each core's executable binary

Internal memory:

- OCMC_RAM: Internal Memory section accessible from all processor cores
- L2_SRAM: DSP L2 Internal Memory section, only accessible from DSP core

Linux mem:

- Memory partitions given to Linux kernel memory manager

Others:

- HDVPSS_DESC_MEM: Memory section used by BSP/STW. It is used by these drivers for its internal descriptor data structures.

3. VSDK Memory Map Table

Refer the .xs file under /vision_sdk/<apps>/build/<SoC>/ for the complete memory map configuration,
For example, /vision_sdk/apps/build/tda2xx/mem_segment_definition_bios.xs defines the memory map for TDA2xx BIOS only with 512MB DDR.
Also refer the generated map files under
/vision_sdk/binaries/<apps>/<tdaxx_evm_bios_all>/vision_sdk/bin/<tdaxx-evm>/
For example, vision_sdk_ipu1_0_release.xem4.map,
Memory sections are arranged as below

```
Name                        Origin    Length    Used      Unused    Attr
----------------------      --------  --------- --------  --------  ----
L2_ROM                      00000000  00004000  000005ec  00003a14  RWIX
L2_RAM                      20000000  00010000  00000000  00010000  RWIX
OCMC_RAM1                   40300000  00080000  00080000  00000000  RWIX
OCMC_RAM2                   40400000  00100000  00000000  00100000  RWIX
OCMC_RAM3                   40500000  00100000  00000000  00100000  RWIX
DSP1_L2_SRAM                40800000  00048000  00000000  00048000  RWIX
DSP2_L2_SRAM                41000000  00048000  00000000  00048000  RWIX
NDK_MEM                     85800000  00200000  00000000  00200000  RWIX
SR1_FRAME_BUFFER_MEM        85a00000  0fa00000  0fa00000  00000000  RWIX
DSP1_CODE_MEM               99000000  00200000  00000000  00200000  RWIX
DSP1_DATA_MEM               99200000  01800000  00000000  01800000  RWIX
IPU1_1_CODE_MEM             9d000000  00200000  00000000  00200000  RWIX
IPU1_1_DATA_MEM             9d200000  00200000  00000000  00200000  RWIX
IPU1_1_BSS_MEM              9d400000  00200000  00000000  00200000  RWIX
IPU1_0_CODE_MEM             9d600000  00600000  004fa45d  00105ba3  RWIX
IPU1_0_DATA_MEM             9dc00000  00400000  00000000  00400000  RWIX
IPU1_0_BSS_MEM              9e000000  00d00000  00bf5cdc  0010a324  RWIX
DSP2_CODE_MEM               9f000000  00200000  00000000  00200000  RWIX
DSP2_DATA_MEM               9f200000  00a00000  00000000  00a00000  RWIX
SR0                         a0100000  00100000  00100000  00000000  RWIX
REMOTE_LOG_MEM              a0200000  00040000  00027890  00018770  RWIX
LINK_STATS_MEM              a0240000  00080000  00020d9c  0005f264  RWIX
SYSTEM_IPC_SHM_MEM          a02c0000  00040000  00035a60  0000a5a0  RWIX
HDVPSS_DESC_MEM             a0300000  00100000  000b07c0  0004f840  RWIX
TRACE_BUF                   a0400000  00060000  00008000  00058000  RWIX
EXC_DATA                    a0460000  00010000  00000000  00010000  RWIX
PM_DATA                     a0470000  00080000  00000000  00080000  RWIX
EVE1_VECS_MEM               a2000000  00080000  00000000  00080000  RWIX
EVE1_CODE_MEM               a2080000  00200000  00000000  00200000  RWIX
EVE1_DATA_MEM               a2280000  00d80000  00000000  00d80000  RWIX
EVE2_VECS_MEM               a3000000  00080000  00000000  00080000  RWIX
EVE2_CODE_MEM               a3080000  00200000  00000000  00200000  RWIX
EVE2_DATA_MEM               a3280000  00d80000  00000000  00d80000  RWIX
EVE3_VECS_MEM               a4000000  00080000  00000000  00080000  RWIX
EVE3_CODE_MEM               a4080000  00200000  00000000  00200000  RWIX
EVE3_DATA_MEM               a4280000  00d80000  00000000  00d80000  RWIX
EVE4_VECS_MEM               a5000000  00080000  00000000  00080000  RWIX
EVE4_CODE_MEM               a5080000  00200000  00000000  00200000  RWIX
EVE4_DATA_MEM               a5280000  00d80000  00000000  00d80000  RWIX
```

Refer /vision_sdk/links_fw/include/link_api/systemLink_common_if.h for the available Memory Heaps, Here is a brief list; detailed description is in systemLink_common_if.h file

SYSTEM_HEAPID_DDR_NON_CACHED_SR0: Heap ID of heap in DDR, This is non-cached memory
SYSTEM_HEAPID_DDR_CACHED_SR1: Heap ID of heap in DDR, This is cached memory
SYSTEM_HEAPID_OCMC_SR2: Heap ID of heap in OCMC
SYSTEM_HEAPID_RESERVED1: Heap ID of heap in DDR, This is cached memory
SYSTEM_HEAPID_RESERVED2: Heap ID of heap in L2 Memory, Internal memory

4. Software and Hardware Constraints to Consider For Deciding Memory Map

4.1 Hardware Constraints

- AMMU in IPU1/2 handles large memory segments of size 512MB/32MB only and there can be 4 such segments.
- For EVE we have only 32 TLB entries to map the memory, one TLB can map a max of 16MB

4.2 Software Constraints

- For VSDK Linux, the first 64MB from 0x8000 0000 is reserved for Linux Kernel
- Frame Buffer Shared Region (SR1) is mapped on A15 Linux. But if user wants Linux side app/links to allocate memory/buffers from any other heap, then user need to mmap the physical address in the application code to map those memory/buffers on A15.

Important things to pay attention to, while porting the map file are:

- DDR, OCMC & DSP/EVE SRAM sizes
- Core specific code/data/vecs sizes
- Size of the shared frame buffer pools

DDR is divided into 2 sections: cached and non-cached.

- The cached part is used mainly for frame buffer (SR1) and core specific code/data sections.
- The non-cached part is used mainly for Vision SDK log buffers, IPC shared data structure (SR0), HDVPSS descriptors etc.

5. Memory Allocation

This section describes the different methods by which memory is allocated in the Vision SDK framework. The Vision SDK framework also supports static memory allocation.

Memory in Vision SDK framework is allocated for the following purposes

Purpose	Region in memory used for allocation	Type of allocation (Dynamic, Static)
External Buffer memory for storing video, algorithms results and/or HW engine results	SR1_FRAME_BUFFER_MEM	Dynamic (heap based) and/or Static
Internal Buffer memory for storing algorithms results and/or HW engine results	OCMC_RAM	Dynamic (heap based) and/or Static
Notify Shared region – ONLY used	SR0	Dynamic (heap based)

during Notify setup (IPC_Start()), not used later		
Temporary scratch memory in internal memory for algorithms results	DMEM in EVE L2SRAM in DSP	Dynamic (non-heap, linear allocation)
Shared memory for remote core print logs	REMOTE_LOG_MEM	Static
Shared memory for link statistics	LINK_STATS_MEM	Static
Shared memory for inter processor communication	SYSTEM_IPC_SHM_MEM	Static
VPDMA descriptor memory for VIP, VPE HW engines	HDVPSS_DESC_MEM	Static
CPU specific memory for BIOS objects like semaphores, tasks, interrupts, clocks	CPU specific data section	Static

The subsequent sections provide more details on each type of memory allocation
In the below description,
<soc> = tda2xx, tda2ex, tda3xx, tda2x-entry & TDA2Px
<ddr_size> = 128MB, 256MB, 512MB, 1024MB
<os_type> = Bios, Linux

5.1 External Buffer Memory Allocation

Location where memory map is specified

- The memory region used for external buffer memory allocation is specified via the below file
    - File: /vision_sdk/<apps>/build/<soc>/mem_segment_definition_<os_type>.xs
    - Variable SR1_FRAME_BUFFER_SIZE
- The heap from which memory is allocated is defined in file
    - FILE: /vision_sdk/links_fw/src/rtos/utils_common/src/utils_mem_ipu1_0.c
        - #pragma DATA_SECTION(gUtils_memHeapDDR, ".bss:heapMemDDR")
    - This heap is placed in “SR1_FRAME_BUFFER” section via the IPU1-0 cfg file
        - FILE: /vision_sdk/links_fw/src/rtos/bios_app_common/<soc>/ipu1_0/Ipu1_0.cfg
            - Program.sectMap[".bss:heapMemDDR"]   = "SR1_FRAME_BUFFER_MEM";
- The heap is defined only on IPU1-0 CPU; all other CPUs sends message to IPU1-0 to allocate memory. This is done internally inside the Utils_memAlloc APIs.

API to allocate and free memory

- Below APIs are used to allocate and free memory
    - FILE: /vision_sdk/links_fw/src/rtos/utils_common/include/utils_mem.h
    - API:
        - Utils_memAlloc() with heapId as SYSTEM_HEAPID_DDR_CACHED_SR1
        - Utils_memFree() with heapId as SYSTEM_HEAPID_DDR_CACHED_SR1
        - Utils_memGetHeapStats() with heapId as SYSTEM_HEAPID_DDR_CACHED_SR1
- Other APIs from this file are not recommended to be used by users and are used internally by the framework

Using static memory allocation

- When a system wants to use static memory allocation and avoid the heap, it should set the size of this heap segment as 0 by modifying the #define in utils_mem_cfg.h file
- Define static memory objects (arrays, data structures) in IPU1-0 use-case file. Make sure the objects are placed in data section ".bss:heapMemDDR" via #pragma
- The links which support static memory allocation allow passing of memory region pointers from use-case file via System_LinkMemAllocInfo data structure
- When creating a link from a use-case, user should now pass memory pointer allocated statically from use-case file. This prevents the link for allocating memory internally. Thus dynamic memory allocation is avoided
    - See capture link "captureLink.h" for example
    - See use-case "/vision_sdk/apps/src/rtos/usecases/vip_single_cam_view" for sample usage of passing user memory pointer to a link
    - NOTE: In the use-case the memory allocation is still done using Utils_memAlloc APIs. In a fully static memory system, this API won't be used by the user.
- The links assert if the memory segment size passed to it is smaller than what is needed. In this case, it also reports the size required by the link.
- When creating user specific AlgPlugins same mechanism should be used, i.e algorithm plugin should take memory pointer passed from use-case file rather than allocating memory internally. See "Capture" link for example

5.2 Internal Buffer Memory Allocation

Location where memory is specified

- The memory region used for buffer memory allocation is specified via the below file
    - File: /vision_sdk/<apps>/build/<soc>/mem_segment_definition_<os_type>.xs
    - Variable OCMC1_SIZE
- The heap from which memory is allocated is defined in file
    - FILE: /vision_sdk/links_fw/src/rtos /utils_common/src/utils_mem_ipu1_0.c
        - #pragma DATA_SECTION(gUtils_memHeapOCMC, ".bss:heapMemOCMC")
    - FILE: /vision_sdk/links_fw/src/rtos /utils_common/include/utils_mem_cfg.h
        - #define UTILS_MEM_HEAP_OCMC_SIZE
    - This heap is placed in "OCMC" section via the IPU1-0 cfg file
        - FILE: /vision_sdk/links_fw/src/rtos/bios_app_common/<soc>/ipu1_0/Ipu1_0.cfg
            - Program.sectMap[".bss:heapMemOCMC"]   = "OCMC_RAM";
- The heap is defined only on IPU1-0 CPU; all other CPUs send a command to IPU1-0 to allocate memory. This is done internally inside the Utils_memAlloc APIs.

API to allocate and free memory

- Below APIs are used to allocate and free memory
    - FILE: /vision_sdk/links_fw/src/rtos/utils_common/include/utils_mem.h
    - API:
        - Utils_memAlloc() with heapId as SYSTEM_HEAPID_OCMC_SR2

    - Utils_memFree() with heapId as SYSTEM_HEAPID_OCMC_SR2
    - Utils_memGetHeapStats() with heapId as SYSTEM_HEAPID_OCMC_SR2
- Other APIs from this file are not recommended to be used by users and are used internally by the framework

5.3 Static Memory Sections Allocation

- The memory region used for these sections are specified via the below file
    - File: /vision_sdk/<apps>/build/<soc>/mem_segment_definition_<os_type>.xs
    - Variable "REMOTE_LOG_SIZE" for Remote Log memory
    - Variable "SYSTEM_IPC_SHM_SIZE" for inter-processor communication
    - Variable "LINK_STATS_SIZE" for Link Statistics
    - Variable "HDVPSS_DESC_SIZE" for VPDMA descriptors

6. Memory Map of the Application

Memory map of the entire usecase is governed by following artifacts.

1. DDR_MEM variable in /vision_sdk/apps/configs/tdaxxx_evm_<OS>_all/cfg.mk

List of VSDK files need to be reviewed & modified

1. */vision_sdk/apps/build/tdaxxx/mem_segment_definition_<OS>.xs*
2. */vision_sdk/links_fw/rtos/src/utils_common/include/utils_mem_cfg.h*
3. */vision_sdk/links_fw/src/rtos/bios_app_common/tdaxx/ipu1_0/ Ammu1.cfg or Ammu1_linux.cfg (if you modify the IPU1 memory map)*
4. */vision_sdk/links_fw/src/rtos/ bios_app_common/tdaxx/ipu2/ Ammu2.cfg or Ammu2_linux.cfg (if you modify the IPU2 memory map)*
5. */vision_sdk/links_fw/src/rtos/links_ipu/system/system_bsp_init.c (vpsInitPrms.virtBaseAddr & vpsInitPrms.physBaseAddr translation)*
6. */vision_sdk/links_fw/include/link_api/system_vring_config.h (only for Linux/HLOS build)*
7. */vision_sdk/links_fw/src/hlos/osa/include/osa_mem_map.h (Generated file, only for Linux/HLOS build)*
8. */vision_sdk/links_fw/src/rtos /links_common/system/system_rsc_table_ipu.h (only for Linux/HLOS build, if resource table modification required)*
9. */vision_sdk/links_fw/src/rtos /links_common/system/system_rsc_table_dsp.h (only for Linux/HLOS build, if resource table modification required)*
10. */vision_sdk/links_fw/src/rtos/bios_app_common/tdaxxx/<ipu2 or ipu1>/gen_system_mem_map.xs (only for Linux/HLOS build, This file auto generates a .h file using XDC varaiables defined in mem_segment_definition_<OS>.xs*

List of Linux Kernel files need to be modified

11. */ti_components/os_tools/kernel/omap/arch/arm/boot/dts/dra7-evm-infoadas.dts (For TDA2x)*
12. */ti_components/os_tools/kernel/omap/arch/arm/boot/dts/dra72-evm-infoadas.dts (For TDA2Ex)*
13. */ti_components/os_tools/kernel/omap/arch/arm/boot/dts/dra76-evm-infoadas.dts (For TDA2Px)*

#1 – DDR_MEM is an environment variable that tells build system which .xs is to be picked up for the final executable.

#2 – The .xs file overrides default implementation for the platform defined by xdc.runtime. This file can be modified to increase/decrease size of a section or add/remove sections from the memory map. For Linux/HLOS, Linux enables L2MMU for each core, so all the addresses mentioned in the .xs file are slave virtual addresses.

#3 – The .dts file is used to reserve memory from Linux, this is a platform specific file. This ensures Linux and bios side don't overwrite into each other. Typically the bios side needs some memory sections and rest all can be given to Linux. Essentially this creates a few holes in Linux memory that is later mapped to user space at the application startup time.

6.1 Adding a new Section to Memory map

While adding a new section in the memory map of ipu/dsp/eve/A15, following things needs to be taken care of:

1. Add a new section in appropriate .xs file by defining NEW_SECTION_SIZE, NEW_SECTION_ADDR & NEW_SECTION_MEM (just follow the convention used in .xs file)
2. Its advised to remove or reduce some unwanted sections to free-up the memory required for the new section
3. Make sure the total memory should not exceeds the total available physical memory
4. Make sure the new section doesn't overlap with any other sections.
5. If you add any new memory section and, the data/code corresponding to that can be placed into the section by adding Program.sectMap in the appropriate */vision_sdk/links_fw/src/rtos/bios_app_common/*tdaxxx/<cpu>/**<cpu>.cfg** file.
6. **In case of Linux**, it should lie within the hole of memory declared in .dts file in kernel using /memreserve
7. If needed, /memreserve can be used to increase the size of the hole accommodate new section's memory requirement.
8. If this newly added section has to be mapped into L2MMU of ipu/dsp by Linux and hence it needs to be added in the resource table i.e. in system_rsc_table_ipu.h or system_rsc_table_dsp.h accordingly.
9. If this section is going to be accessed from Linux user space or kernel space, this mapping needs to be taken care by the application or through OSA_mem module in vision_sdk
10. If you are changing base addresses and sizes for IPU's, DSP's carve-out sections (code/data) and if you plan to change CMA address in linux kernel (.dts) please ensure you also make this changes to */vision_sdk/links_fw/include/link_api/system_vring_config.h*.
11. Refer section "6.4: How To – Modify Linux/Bios VSDK Memory Map" for more details

6.2 Changing size of a Section in the Memory map

While changing the size of the section in the memory map from ipu/dsp/eve/A15, following things needs to be taken care of:

1. Do changes in respective .xs file for the section sizes
2. Its advised to reduce some unwanted sections to free-up the memory required for new size (increase)
3. Make sure the total memory should not exceeds the total available physical memory
4. Make sure the new section doesn't overlap with any other sections.
5. **In case of Linux**, it should lie within the hole of memory declared in .dts file in kernel using /memreserve
6. If needed, /memreserve can be used to increase the size of the hole accommodate new section's memory requirement.
7. As you are modifying existing section, no need to change resource table mappings, the updated value will be picked up in resource table in the build process.
8. If you are changing base addresses and sizes for IPU's, DSP's carve-out sections (code/data) and if you plan to change CMA address in linux kernel (.dts) please ensure you also make this changes to */vision_sdk/links_fw/include/link_api/system_vring_config.h*.
9. Refer section "6.4: How To – Modify Linux/Bios VSDK Memory Map" for more details

6.3 How To - Add a new Memory map

In general, if you are planning to have your own memory map for the application, you can follow these steps

1. Evaluate memory requirements of the sections e.g. (Is 256 MB SR1 sufficient or you need more or less?)
2. Add appropriate .xs file under $INSTALL_DIR/*vision_sdk/apps/build/tdaxxx/,* for example mem_segment_definition_bios.xs
2. Modify DDR_MEM_XXXX, for example DDR_MEM_512M variable in /vision_sdk/apps/configs/tdaxxx_evm_<OS>_all/cfg.mk
3. Modify appropriate platform ISI build files to pick the correct memory map (.xs) file, for example, refer below files for TDA2x,
    a. /vision_sdk/build/rtos/tda2xx/config_arp32.bld
    b. /vision_sdk/build/rtos/tda2xx/config_c66.bld
    c. /vision_sdk/build/rtos/tda2xx/config_m4.bld
    d. /vision_sdk/build/rtos/tda2xx/config_a15.bld
4. Now follow the section "6.4: How To – Modify Linux/Bios VSDK Memory Map" for more details and how to modify all necessary VSDK and Linux kernel files

6.4 How To – Modify Linux/Bios VSDK Memory Map

The memory map of complete VISION SDK is controlled in
/vision_sdk/build/rtos/tdaxxx/config_<ISI>.bld

The mem_segment_definition (for example - mem_segment_definition_linux.xs) file is included in this build configuration file, size of each sections are reconfigured in .xs file. For example, DSP code size and DSP data size section can be changed by modifying the following entries.
DSP1_CODE_SIZE = 3*MB;
DSP1_DATA_SIZE = 13*MB;

The base addresses of each section are incremented based on the base address of previous section and the size of the previous section. For example, if sections are created in the numerical order, base address of Section 2 is calculated as below:
<Start Addr of Sect 2> = <Start Addr of Sec 1> + <Size of Sect 1>

VSDK need one cached and one non-cached memory segments in the memory map (.xs) file.
Cached segment: You can place any memory sections in this segment except the ones listed under Non-cached segments

Non-cached segments: Below sections must be placed under non-cached segments
SR0_ADDR
REMOTE_LOG_ADDR
LINK_STATS_ADDR
SYSTEM_IPC_SHM_ADDR
HDVPSS_DESC_ADDR
OPENVX_SHM_ADDR

Below Sections are used only for A15 Linux memory map
TRACE_BUF_BASE

EXC_DATA_BASE
PM_DATA_BASE

To modify the memory map, user needs to consider the following:

- Refer to the hardware limitations and software limitations in section 4:
    - We assume a one-to-one mapping of AMMU virtual address to physical address.
- The other sections like “Remote Debug”, “HDVPSS Shared Memory” etc. are read directly from the build configuration file
- Changes in the Linux memory size in build configuration file has to be reflected in the boot arguments of the Linux kernel using “**mem=<SIZE>M**” entry.
- Consider the overall buffer requirement for the specific usecase before modifying the Frame Buffer or Meta data buffer or BitsBuffer section sizes.

The current default memory maps of VSDK (as per 3.0 releases) as below

- TDA2xx Bios – 512 MB
- TDA2xx Linux – 1024 MB
- TDA2Ex Bios – 512 MB
- TDA2Ex Linux – 1024 MB
- TDA3xx Bios – 512 MB
- TDA3xx Bios – 128 MB
- TDA2Px Bios – 512 MB
- TDA2Px Linux – 1024 MB

6.4.1 Cache and MMU configurations

List of VSDK files sets the Cache and MMU configurations

<1> DSP

DSP L1 & L2 Cache configuration is in
*/vision_sdk/links_fw/src/rtos/bios_app_common/*tdaxxx/cfg/DSP_common.cfg,
Below the default cache size setting for L1P, L1D and L2 as 32K

```
var Cache = xdc.useModule('ti.sysbios.family.c66.Cache');
Cache.initSize.l1pSize = Cache.L1Size_32K;
Cache.initSize.l1dSize = Cache.L1Size_32K;
Cache.initSize.l2Size  = Cache.L2Size_32K;
```

Cache ON/OFF setting of DSP also in
*/vision_sdk/links_fw/src/rtos/bios_app_common/*tdaxxx/cfg/DSP_common.cfg

```
/* Set cache sections */
/* configure MARs, by default cache is enabled for the entire memory region */
for (var i = 0; i < Program.cpu.memoryMap.length; i++)
{
memSegment = Program.cpu.memoryMap[i];
Cache.setMarMeta(memSegment.base, memSegment.len, Cache.Mar_ENABLE);
}

/* set non-cached sections */
for (var i = 0; i < Program.cpu.memoryMap.length; i++)
{
memSegment = Program.cpu.memoryMap[i];
if ((memSegment.name == "SR0") ||
(memSegment.name == "REMOTE_LOG_MEM") ||
(memSegment.name == "LINK_STATS_MEM") ||
(memSegment.name == "SYSTEM_IPC_SHM_MEM") ||
(memSegment.name == "OPENVX_SHM_MEM"))
{
Cache.setMarMeta(memSegment.base, memSegment.len, Cache.Mar_DISABLE);
}
}
```

If you plan to add any new non-cached DSP section, then add the same section in above code snippet in DSP_common.cfg.

<2> IPU
Refer below files which set the AMMU of IPU to configure MMU and Cache settings
/vision_sdk/links_fw/src/rtos/bios_app_common/tdaXxx/ipu1_0/Ammu1_bios.cfg or Ammu1_linux.cfg
/vision_sdk/links_fw/src/rtos/bios_app_common /tdaXxx/ipu2/Ammu2_bios.cfg or Ammu2_linux.cfg
Map program code/data & other cached memory into ammu (cacheable) by AMMU.largePages[1]
Map SR_0 & other non-cached data memory into ammu (non-cacheable) by AMMU.largePages[2]

Note: Only for TDA3x, IPU1 AMMU setting is done in SBL.
Refer */vision_sdk/links_fw/src/rtos/bios_app_common*/tda3xx/ipu1_0/Ammu1_bios.cfg,
AMMU.configureAmmu = false;
If configureAmmu is set to false, then AMMU setting is done in SBL.

<3> EVE
Refer */vision_sdk/links_fw/src/rtos/bios_app_common*/tda2xx/eve_common/tlb_config_eve_common.c for EVE memory mapping by programming the TLB registers, each TLB can map a max of 16MB contiguous region and 16MB aligned.
No changes required in this file as it has been taken care automatically.

<4> A15 (Bios)
If A15 running Bios, then */vision_sdk/links_fw/src/rtos/bios_app_common*/tda2xx/a15_0/ a15_0.cfg does the MMU & cache configurations,
Mmu.setSecondLevelDescMeta();

6.4.2 *Modify default Memory maps of VSDK*

Please revisit & modify the list of below files (as required)

List of VSDK files
<1>
/vision_sdk/apps/build/tdaxxx/**mem_segment_definition_<OS>.xs**.
A single file that configures the entire memory map of all cores in the SoC, and the major file to be modified if you want to make changes in VSDK memory map. This file defines all the memory sections for IPU, DSP, EVE, A15 and other heaps such as SR1, SR0 etc. Below a sample section,

```
DSP1_START_ADDR         = 0x99000000;
DSP1_CODE_SIZE          = 2*MB;
DSP1_DATA_SIZE          = 24*MB;
…………………………………………………………
```

```
…………………………………………………………

DSP1_CODE_ADDR           = DSP1_START_ADDR;
DSP1_DATA_ADDR           = DSP1_CODE_ADDR      + DSP1_CODE_SIZE;
…………………………………………………………
…………………………………………………………

function getMemSegmentDefinition_external(core)
{
memory[index++] = ["DSP1_CODE_MEM", {
comment : "DSP1_CODE_MEM",
name    : "DSP1_CODE_MEM",
base    : DSP1_CODE_ADDR,
len     : DSP1_CODE_SIZE
}];

…………………………………………………………
…………………………………………………………
}
```

If you add any new memory section and, the data/code corresponding to that can be placed into the section by adding **Program.sectMap[]** in the appropriate */vision_sdk/links_fw/src/rtos/bios_app_common*/tdaxxx/<cpu>/**<cpu>.cfg** file.
For example, */vision_sdk/links_fw/src/rtos/bios_app_common*/tda2xx/dsp1/Dsp1.cfg is the file for DSP1 of TDA2x, and below code place “bss:extMemNonCache:ipcShm” into the memory section “SYSTEM_IPC_SHM_MEM”

```
Program.sectMap[".bss:extMemNonCache:ipcShm"] = "SYSTEM_IPC_SHM_MEM";
Program.sectMap[".bss:extMemNonCache:linkStats"] = "LINK_STATS_MEM";
```

Same apply for all cores like IPU1-0, IPU1-1, IPU2, A15, DSP1-2 and EVE1-4 of TDA2x, TDA2Px, TDA2Ex and TDA3x.

The .xs file to look at depends on SoC, A15 OS and DDR memory config selected; here is a list of default memory map of VSDK 3.1.0.0

SoC	**A15 OS**	**DDR config**	**.xs file**
TDA2XX_BUILD	Bios	TDA2XX_512MB_DDR	vision_sdk/apps/build/tda2xx/mem_segment_definition_bios.xs
TDA2XX_BUILD	Linux	TDA2XX_1024MB_DDR	vision_sdk/ apps/build/tda2xx/mem_segment_definition_linux.xs
TDA3XX_BUILD	NA	TDA3XX_128MB_DDR	vision_sdk/ apps/build/tda3xx/mem_segment_definition_128mb.xs
TDA3XX_BUILD	NA	TDA3XX_512MB_DDR	vision_sdk/ apps/build/tda3xx/mem_segment_definition_512mb.xs
TDA2EX_BUILD	Bios	TDA2EX_512MB_DDR	vision_sdk/ apps/build/tda2ex/mem_segment_definition_bios.xs
TDA2EX_BUILD	Linux	TDA2EX_1024MB_DDR	vision_sdk/ apps/build/tda2ex/mem_segment_definition_linux.xs
TDA2PX_BUILD	Bios	TDA2PX_512MB_DDR	vision_sdk/apps/build/tda2px/mem_segment_definition_bios.xs
TDA2PX_BUILD	Linux	TDA2PX_1024MB_DDR	vision_sdk/ apps/build/tda2px/mem_segment_definition_linux.xs

Modify “start address”, “size” or even add/remove a memory section to change the memory map as per your requirement.

<2>
/vision_sdk/links_fw/src/rtos/utils_common/include/utils_mem_cfg.h

This file defines the size of major internal memory heaps, and these sizes need to be in sync with above .xs file. The memory allocation utility/API refers this file for the heap size. Any modification of these heap size to be updated in both **utils_mem_cfg.h & mem_segment_definition_<OS>.xs** file.

```
#define UTILS_MEM_HEAP_L2_SIZE  (224*1024) – DSP internal memory (SRAM)
#define UTILS_MEM_HEAP_L2_SIZE  (24*1024) - EVE internal memory (SRAM)
#define UTILS_MEM_HEAP_OCMC_SIZE       (512*1024) – Shared OCMC internal memory
#define UTILS_MEM_HEAP_DDR_CACHED_SIZE      (256*1024*1024) – Shared cached DDR heap memory.
```

<3>
/vision_sdk/links_fw/src/rtos/links_ipu/system/system_bsp_init.c (vpsInitPrms.virtBaseAddr & vpsInitPrms.physBaseAddr translation)

```
    /* This one to one mapping is required for the 1GB builds */
     vpsInitPrms.virtBaseAddr = 0x80000000U;
     vpsInitPrms.physBaseAddr = 0x80000000U;
     /* if Virtual address != Physical address then enable translation */
     vpsInitPrms.isAddrTransReq = FALSE;

    /* This one to one mapping is required for the 512MB builds */
     vpsInitPrms.virtBaseAddr = 0xA0000000U;
     vpsInitPrms.physBaseAddr = 0x80000000U;
     /* if Virtual address != Physical address then enable translation */
     vpsInitPrms.isAddrTransReq = TRUE;
```

<4>
/vision_sdk/links_fw/src/rtos/bios_app_common/tda2xx/ipu1_0/**Ammu1_bios.cfg** or **Ammu1_linux.cfg** (if you modify the IPU1 memory map)
/vision_sdk/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/**Ammu2_bios.cfg** or **Ammu2_linux.cfg** (if you modify the IPU2 memory map)
This file set IPU subsystem (core 0 and core 1) AMMU and Cache configurations. IPU can access only the memory sections mapped via AMMU. A sinle AMMU sets the memory map of both cores (core 0 & core 1) of an IPU subsystem.

```
function init()
{
…………………………………………………………
…………………………………………………………

     Map program code/data & other cached memory into ammu (cacheable) by AMMU.largePages[1]
```

Map SR_0 & other non-cached data memory into ammu (non-cacheable) by AMMU.largePages[2]
…………………………………………………………
…………………………………………………………
}

<5>
/vision_sdk/links_fw/src/rtos/bios_app_common/tda2xx/eve_common/**tlb_config_eve_common.c** (if you modify any EVE1-4 memory map)
This file implements common MMU configuration for all EVE as per Vision SDK requirements
There are only 32 TLB entries in EVE, Each TBL entry can maps a max of 16MB, and need 16MB alignment,
No changes required in this file as it has been taken care automatically.

<6>
/vision_sdk/links_fw/include/link_api/**system_vring_config.h** (used only in A15 Linux Build)

Vring virtual addresses (For Start address) of IPU & DSP are used by IPC, if it is changed in .XS file, same need to be updated in this system_vring_config.h also.

```
#ifdef BUILD_M4_0
#define IPU_PHYS_MEM_IPC_VRING      0x9e000000
#endif

#ifdef BUILD_DSP_1
#define DSP_PHYS_MEM_IPC_VRING      0xa1000000
#endif

#ifdef BUILD_DSP_2
#define DSP_PHYS_MEM_IPC_VRING      0xa3000000
#endif

#ifdef BUILD_M4_2
#define IPU_PHYS_MEM_IPC_VRING      0x99000000
#endif
```

<7>
/vision_sdk/links_fw/src/hlos/osa/include/osa_mem_map.h (Build time generated file, used only in A15 Linux Build)

This is an auto generated file from [gen_system_mem_map.xs], please check and confirm the entries in osa_mem_map.h is matching with .XS file or the MAP file

```
#define SR0_ADDR    0xa0100000
#define SR0_SIZE    0x100000

#define SYSTEM_IPC_SHM_MEM_ADDR    0xa02c0000
#define SYSTEM_IPC_SHM_MEM_SIZE    0x80000

#define REMOTE_LOG_MEM_ADDR      0xa0200000
#define REMOTE_LOG_MEM_SIZE      0x40000

#define SR1_FRAME_BUFFER_MEM_ADDR 0x84203000
```

#define SR1_FRAME_BUFFER_MEM_SIZE 0xfa00000

#define SR2_FRAME_BUFFER_MEM_ADDR 0xa9000000
#define SR2_FRAME_BUFFER_MEM_SIZE 0x4000000

#define OPENVX_OBJ_DESC_MEM_ADDR 0xa0440000
#define OPENVX_OBJ_DESC_MEM_SIZE 0x200000

<8>
If a newly added section has to be mapped into L2MMU of ipu/dsp by Linux and hence it needs to be added in the resource table i.e. in system_rsc_table_ipu.h or system_rsc_table_dsp.h accordingly.

*/vision_sdk/links_fw/src/rtos/*links_common/system/**system_rsc_table_ipu.h** (modify if required, used only in A15 Linux Build)
struct my_resource_table {…}
struct my_resource_table ti_ipc_remoteproc_ResourceTable = {…}

system_rsc_table_ipu.h define the resource table entries for all IPU cores. This will be incorporated into corresponding base images, and used by the remoteproc on the host-side to allocated/reserve resources.

*/vision_sdk/links_fw/src/rtos/*links_common/system/**system_rsc_table_dsp.h** (modify if required, used only in A15 Linux Build)
struct my_resource_table {…}
struct my_resource_table ti_ipc_remoteproc_ResourceTable = {…}

system_rsc_table_dsp.h define the resource table entries for all DSP cores. This will be incorporated into corresponding base images, and used by the remoteproc on the host-side to allocated/reserve resources.

<9>
/vision_sdk/links_fw/src/rtos/bios_app_common/tdaxxx/<ipu2 or ipu1>/gen_system_mem_map.xs (only for Linux/HLOS build, This file auto generates a .h file using XDC varaiables defined in mem_segment_definition_<OS>.xs

Change this file only when you want to create a 512MB or <512MB Linux memory map

List of Linux Kernel files

<1>
/ti_components/os_tools/kernel/omap/arch/arm/boot/dts/**dra7-evm-infoadas.dts** (for TDA2x only)
Modify the start address of IPU1, IPU2, DSP1, DSP2 or CMEM, if any of this is changed in the memory map,

```
/* Update the CMA regions for Vision SDK binaries */
&ipu2_cma_pool {
      reg = <0x0 0x99000000 0x0 0x5000000>;
};

&dsp1_cma_pool {
      reg = <0x0 0xa1000000 0x0 0x2000000>;
};
```

```
&ipu1_cma_pool {
      reg = <0x0 0x9e000000 0x0 0x2000000>;
};

&dsp2_cma_pool {
      reg = <0x0 0xa3000000 0x0 0x2000000>;
};
```

This file reserves the memory for SR1, SR0, CMEM and EVE data/code memory sections and the size. Additional memory regions required for Vision SDK Keep this in sync with VSDK apps/build/tda2xx/mem_segment_definition_linux.xs

```
&reserved_mem {
      cmem_ocmc: cmem@40300000 {
                        compatible = "shared-dma-pool";
                        reg = <0x0 0x40300000 0x0 0x300000>;
                        sram = <&ocmcram1>;
                        no-map;
                        status = "okay";
      };

      cmem_pool: cmem@A9000000 {
                        compatible = "shared-dma-pool";
                        reg = <0x0 0xA9000000 0x0 0x4000000>;
                        no-map;
                        status = "okay";
      };

      vsdk_sr1_mem: vsdk_sr1_mem@84000000 {
                        compatible = "shared-dma-pool";
                        reg = <0x0 0x84000000 0x0 0x10000000>;
                        status = "okay";
      };

      vsdk_sr0_mem: vsdk_sr0_mem@A0000000 {
                        compatible = "shared-dma-pool";
                        reg = <0x0 0xA0000000 0x0 0x1000000>;
                        status = "okay";
      };

      vsdk_eve_mem: vsdk_eve_mem@A5000000 {
                        compatible = "shared-dma-pool";
                        reg = <0x0 0xA5000000 0x0 0x4000000>;
                        status = "okay";
      };
};
```

<2>
/ti_components/os_tools/kernel/omap/arch/arm/boot/dts/**dra72-evm-infoadas.dts** (for TDA2Ex only)
Modify the start address of IPU1, IPU2, DSP1 or CMEM, if any of this is changed in the memory map,

```
/* Update the CMA regions for Vision SDK binaries */
&ipu2_cma_pool {
	reg = <0x0 0x99000000 0x0 0x5000000>;
};

&dsp1_cma_pool {
	reg = <0x0 0xa1000000 0x0 0x2000000>;
};

&ipu1_cma_pool {
	reg = <0x0 0x9e000000 0x0 0x2000000>;
};
```

This file reserves the memory for SR1, SR0 and CMEM sections and the size. Additional memory regions required for Vision SDK Keep this in sync with VSDK apps/build/tda2Ex/mem_segment_definition_linux.xs

```
&reserved_mem {
	cmem_ocmc: cmem@40300000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0x40300000 0x0 0x300000>;
				sram = <&ocmcram1>;
				no-map;
				status = "okay";
	};

	cmem_pool: cmem@A9000000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0xA9000000 0x0 0x4000000>;
				no-map;
				status = "okay";
	};

	vsdk_sr1_mem: vsdk_sr1_mem@84000000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0x84000000 0x0 0x10000000>;
				status = "okay";
	};

	vsdk_sr0_mem: vsdk_sr0_mem@A0000000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0xA0000000 0x0 0x1000000>;
				status = "okay";
	};
};
```

<3>
/ti_components/os_tools/kernel/omap/arch/arm/boot/dts/**dra76-evm-infoadas.dts** (for TDA2Px only)
Modify the start address of IPU1, IPU2, DSP1, DSP2 or CMEM, if any of this is changed in the memory map,

```
/* Update the CMA regions for Vision SDK binaries */
```

```
&ipu2_cma_pool {
	reg = <0x0 0x99000000 0x0 0x5000000>;
};

&dsp1_cma_pool {
	reg = <0x0 0xa1000000 0x0 0x2000000>;
};

&ipu1_cma_pool {
	reg = <0x0 0x9e000000 0x0 0x2000000>;
};

&dsp2_cma_pool {
	reg = <0x0 0xa3000000 0x0 0x2000000>;
};
```

This file reserves the memory for SR1, SR0, CMEM and EVE data/code memory sections and the size. Additional memory regions required for Vision SDK Keep this in sync with VSDK apps/build/tda2Px/mem_segment_definition_linux.xs

```
&reserved_mem {
	cmem_ocmc: cmem@40300000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0x40300000 0x0 0x300000>;
				sram = <&ocmcram1>;
				no-map;
				status = "okay";
	};

	cmem_pool: cmem@A9000000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0xA9000000 0x0 0x4000000>;
				no-map;
				status = "okay";
	};

	vsdk_sr1_mem: vsdk_sr1_mem@84000000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0x84000000 0x0 0x10000000>;
				status = "okay";
	};

	vsdk_sr0_mem: vsdk_sr0_mem@A0000000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0xA0000000 0x0 0x1000000>;
				status = "okay";
	};

	vsdk_eve_mem: vsdk_eve_mem@A5000000 {
				compatible = "shared-dma-pool";
				reg = <0x0 0xA5000000 0x0 0x4000000>;
```

```
                        status = "okay";
        };
};
```

Clean and Rebuild Kernel & VSDK.

Note1: If SR1 or IPU1-2 memory needs to be increased to very high value, then Move DSP1-2 or EVE1-4 out of 0xA000 0000 address space. This will avoid the need of any IPU AMMU reconfiguration.

Note2: To build SBL, Build appropriate secondary boot loader as per your memory configuration. Refer file /vision_sdk/build/rtos/makerules/build_sbl.mk for all valid configurations. For examples, EMIFMODE = DUAL_EMIF_1GB_512MB (default) or DUAL_EMIF_2X512MB or SINGLE_EMIF_256MB

Note3: In case of A15 Linux, You also need to change DMM configuration in Uboot to set the DDR configuration, i.e., the EMIF and LISA map configuration as per custom board or memory map.

6.5 How To – Modify 512MB Bios Memory map to 1GB Bios Memory map

Assume TDA2x build

```
Subject: [PATCH] 1GB Bios memory map for TDA2x
        - all default cores enabled
        - DSP and EVE moved to the second 512MB
---
 apps/build/tda2xx/mem_segment_definition_bios.xs   | 53 ++++++++++++----------
 build/rtos/makerules/build_sbl.mk            |  2 +-
 .../rtos/bios_app_common/tda2xx/a15_0/a15_0.cfg   |  2 +-
 .../bios_app_common/tda2xx/ipu1_0/Ammu1_bios.cfg  |  6 +--
 .../bios_app_common/tda2xx/ipu2/Ammu2_bios.cfg    |  6 +--
 .../src/rtos/links_ipu/system/system_bsp_init.c    |  4 +-
 6 files changed, 38 insertions(+), 35 deletions(-)
 mode change 100644 => 100755 links_fw/src/rtos/bios_app_common/tda2xx/a15_0/a15_0.cfg
 mode change 100644 => 100755 links_fw/src/rtos/bios_app_common/tda2xx/ipu1_0/Ammu1_bios.cfg
 mode change 100644 => 100755 links_fw/src/rtos/bios_app_common/tda2xx/ipu2/Ammu2_bios.cfg

diff --git a/apps/build/tda2xx/mem_segment_definition_bios.xs
b/apps/build/tda2xx/mem_segment_definition_bios.xs
index ebdae5d..bbf0e17 100755
--- a/apps/build/tda2xx/mem_segment_definition_bios.xs
+++ b/apps/build/tda2xx/mem_segment_definition_bios.xs
@@ -18,7 +18,7 @@ function getMemSegmentDefinition_external(core)
     MB=KB*KB;

     DDR3_ADDR                   = 0x80000000;
-    DDR3_SIZE                  = 512*MB;
+    DDR3_SIZE                  = 1024*MB;

     /*
      * In case of ECC_FFI_INCLUDE, DDR3_BASE_ADDR_1 and DDR3_BASE_SIZE_1
@@ -27,7 +27,7 @@ function getMemSegmentDefinition_external(core)
      * If this DDR3_BASE_SIZE_0 is changed, update  Ipu1_0.cfg
     */
    DDR3_BASE_ADDR_0          = DDR3_ADDR;
-   DDR3_BASE_SIZE_0          = 508*MB;
+   DDR3_BASE_SIZE_0          = 512*MB;

    /* The start address of the second mem section should be 16MB aligned.
     * This alignment is a must as a single 16MB mapping is used for EVE
@@ -42,7 +42,7 @@ function getMemSegmentDefinition_external(core)
        *  in non-cached virtual address of
        *  DDR3_BASE_ADDR_1 + 512*MB
        */
-       DDR3_BASE_ADDR_1       = DDR3_BASE_ADDR_1+512*MB;
+       /* DDR3_BASE_ADDR_1       = DDR3_BASE_ADDR_1+512*MB;  */
    }

    DSP1_L2_SRAM_ADDR         = 0x40800000;
@@ -188,35 +188,17 @@ function getMemSegmentDefinition_external(core)
     * be kept constant across all platforms and should match the increment
     * to heapStats.heapSize in utils_xmc_mpu.c
     */
-   EVE1_VECS_ADDR          = DDR3_BASE_ADDR_0;
-   EVE1_CODE_ADDR          = EVE1_VECS_ADDR          + EVE1_VECS_SIZE;
-   EVE1_DATA_ADDR          = EVE1_CODE_ADDR          + EVE1_CODE_SIZE;
-   EVE2_VECS_ADDR          = EVE1_DATA_ADDR          + EVE1_DATA_SIZE;
-   EVE2_CODE_ADDR          = EVE2_VECS_ADDR          + EVE2_VECS_SIZE;
-   EVE2_DATA_ADDR          = EVE2_CODE_ADDR          + EVE2_CODE_SIZE;
-   EVE3_VECS_ADDR          = EVE2_DATA_ADDR          + EVE2_DATA_SIZE;
-   EVE3_CODE_ADDR          = EVE3_VECS_ADDR          + EVE3_VECS_SIZE;
-   EVE3_DATA_ADDR          = EVE3_CODE_ADDR          + EVE3_CODE_SIZE;
-   EVE4_VECS_ADDR          = EVE3_DATA_ADDR          + EVE3_DATA_SIZE;
-   EVE4_CODE_ADDR          = EVE4_VECS_ADDR          + EVE4_VECS_SIZE;
-   EVE4_DATA_ADDR          = EVE4_CODE_ADDR          + EVE4_CODE_SIZE;
-   NDK_DATA_ADDR           = EVE4_DATA_ADDR          + EVE4_DATA_SIZE;
+   NDK_DATA_ADDR           = DDR3_BASE_ADDR_0        + 16*MB;
    IPU1_1_CODE_ADDR        = NDK_DATA_ADDR           + NDK_DATA_SIZE;
    IPU1_0_CODE_ADDR        = IPU1_1_CODE_ADDR        + IPU1_1_CODE_SIZE;
    IPU2_CODE_ADDR          = IPU1_0_CODE_ADDR        + IPU1_0_CODE_SIZE;
-   DSP1_CODE_ADDR          = IPU2_CODE_ADDR          + IPU2_CODE_SIZE;
-   DSP2_CODE_ADDR          = DSP1_CODE_ADDR          + DSP1_CODE_SIZE;
-   IPU1_1_DATA_ADDR        = DSP2_CODE_ADDR          + DSP2_CODE_SIZE;
+   IPU1_1_DATA_ADDR        = IPU2_CODE_ADDR          + IPU2_CODE_SIZE;
    IPU1_0_DATA_ADDR        = IPU1_1_DATA_ADDR        + IPU1_1_DATA_SIZE;
    IPU2_DATA_ADDR          = IPU1_0_DATA_ADDR        + IPU1_0_DATA_SIZE;
-   DSP1_DATA_ADDR          = IPU2_DATA_ADDR          + IPU2_DATA_SIZE;
-   DSP2_DATA_ADDR          = DSP1_DATA_ADDR          + DSP1_DATA_SIZE;
-   A15_0_DATA_ADDR         = DSP2_DATA_ADDR          + DSP2_DATA_SIZE;
+   A15_0_DATA_ADDR         = IPU2_DATA_ADDR          + IPU2_DATA_SIZE;
    SR1_BUFF_ECC_ASIL_ADDR  = A15_0_DATA_ADDR         + A15_0_DATA_SIZE;
    SR1_BUFF_ECC_QM_ADDR    = SR1_BUFF_ECC_ASIL_ADDR  + SR1_BUFF_ECC_ASIL_SIZE;
-   DSP1_DATA_ADDR_2        = SR1_BUFF_ECC_QM_ADDR    + SR1_BUFF_ECC_QM_SIZE;
-   DSP2_DATA_ADDR_2        = DSP1_DATA_ADDR_2        + DSP1_DATA_SIZE_2;
-   SR1_BUFF_NON_ECC_ASIL_ADDR = DSP2_DATA_ADDR_2          + DSP2_DATA_SIZE_2;
+    SR1_BUFF_NON_ECC_ASIL_ADDR = SR1_BUFF_ECC_QM_ADDR       + SR1_BUFF_ECC_QM_SIZE;
    SR1_FRAME_BUFFER_ADDR      = SR1_BUFF_NON_ECC_ASIL_ADDR + SR1_BUFF_NON_ECC_ASIL_SIZE;

    /* Non Cached Section */
@@ -232,6 +214,27 @@ function getMemSegmentDefinition_external(core)
    HDVPSS_DESC_ADDR          = SYSTEM_IPC_SHM_ADDR   + SYSTEM_IPC_SHM_SIZE;
    OPENVX_SHM_ADDR           = HDVPSS_DESC_ADDR      + HDVPSS_DESC_SIZE;

+    EVE1_VECS_ADDR           = DDR3_BASE_ADDR_1          + 16*MB;
+    EVE1_CODE_ADDR           = EVE1_VECS_ADDR           + EVE1_VECS_SIZE;
+    EVE1_DATA_ADDR           = EVE1_CODE_ADDR           + EVE1_CODE_SIZE;
+    EVE2_VECS_ADDR           = EVE1_DATA_ADDR           + EVE1_DATA_SIZE;
+    EVE2_CODE_ADDR           = EVE2_VECS_ADDR           + EVE2_VECS_SIZE;
+    EVE2_DATA_ADDR           = EVE2_CODE_ADDR           + EVE2_CODE_SIZE;
+    EVE3_VECS_ADDR           = EVE2_DATA_ADDR           + EVE2_DATA_SIZE;
+    EVE3_CODE_ADDR           = EVE3_VECS_ADDR           + EVE3_VECS_SIZE;
+    EVE3_DATA_ADDR           = EVE3_CODE_ADDR           + EVE3_CODE_SIZE;
+    EVE4_VECS_ADDR           = EVE3_DATA_ADDR           + EVE3_DATA_SIZE;
+    EVE4_CODE_ADDR           = EVE4_VECS_ADDR           + EVE4_VECS_SIZE;
+    EVE4_DATA_ADDR           = EVE4_CODE_ADDR           + EVE4_CODE_SIZE;
+
+    DSP1_CODE_ADDR           = EVE4_DATA_ADDR           + EVE4_DATA_SIZE;
+    DSP2_CODE_ADDR           = DSP1_CODE_ADDR           + DSP1_CODE_SIZE;
+    DSP1_DATA_ADDR           = DSP2_CODE_ADDR           + DSP2_CODE_SIZE;
+    DSP2_DATA_ADDR           = DSP1_DATA_ADDR           + DSP1_DATA_SIZE;
+    DSP1_DATA_ADDR_2         = DSP2_DATA_ADDR           + DSP2_DATA_SIZE;
+    DSP2_DATA_ADDR_2         = DSP1_DATA_ADDR_2         + DSP1_DATA_SIZE_2;
+
+
    if ((SR1_FRAME_BUFFER_ADDR + SR1_FRAME_BUFFER_SIZE) > (DDR3_BASE_ADDR_0 +
DDR3_BASE_SIZE_0))
    {
      throw xdc.$$XDCException("MEMORY_MAP OVERFLOW ERROR ",
diff --git a/build/rtos/makerules/build_sbl.mk b/build/rtos/makerules/build_sbl.mk
index 7f2d52e..ea40268 100755
--- a/build/rtos/makerules/build_sbl.mk
+++ b/build/rtos/makerules/build_sbl.mk
@@ -28,7 +28,7 @@ PDK_BUILD_OPTIONS += TOOLCHAIN_PATH_EVE=$(CODEGEN_PATH_EVE)
BIOS_INSTALL_PATH=$(
 PDK_BUILD_OPTIONS += EDMA3LLD_BIOS6_INSTALLDIR=$(edma_PATH) radarLink_PATH=$(radarLink_PATH)
 PDK_BUILD_OPTIONS += utils_PATH=$(UTILSPATH)

-PDK_SBL_BUILD_OPTIONS = $(PDK_BUILD_OPTIONS) EMIF_MODE=SINGLE_EMIF_512MB
+PDK_SBL_BUILD_OPTIONS = $(PDK_BUILD_OPTIONS) EMIF_MODE=DUAL_EMIF_2X512MB
 ifneq ($(ECC_FFI_INCLUDE),yes)
 PDK_SBL_BUILD_OPTIONS+= SBL_CONFIG=disable_safety
 endif
diff --git a/links_fw/src/rtos/bios_app_common/tda2xx/a15_0/a15_0.cfg
b/links_fw/src/rtos/bios_app_common/tda2xx/a15_0/a15_0.cfg
old mode 100644
new mode 100755
index 25a03f9..8a96266
--- a/links_fw/src/rtos/bios_app_common/tda2xx/a15_0/a15_0.cfg
+++ b/links_fw/src/rtos/bios_app_common/tda2xx/a15_0/a15_0.cfg
@@ -162,7 +162,7 @@ attrs2.attrIndx = 0;                    // Non-cache, normal memory
 // Set the descriptor for each entry in the address range
 for (var i=0xA0000000; i < 0xC0000000; i = i + 0x00200000) {
    // Each 'BLOCK' descriptor entry spans a 2MB address range
-    Mmu.setSecondLevelDescMeta(i, i-0x20000000, attrs2);
+    Mmu.setSecondLevelDescMeta(i, i, attrs2);
 }

 // Region for NDK packet data buffers.
diff --git a/links_fw/src/rtos/bios_app_common/tda2xx/ipu1_0/Ammu1_bios.cfg
b/links_fw/src/rtos/bios_app_common/tda2xx/ipu1_0/Ammu1_bios.cfg
old mode 100644
new mode 100755
index 7809abc..685a934
--- a/links_fw/src/rtos/bios_app_common/tda2xx/ipu1_0/Ammu1_bios.cfg
+++ b/links_fw/src/rtos/bios_app_common/tda2xx/ipu1_0/Ammu1_bios.cfg
@@ -204,13 +204,13 @@ function init()
    entry.pageEnabled = AMMU.Enable_YES;
    entry.translationEnabled = AMMU.Enable_YES;
    entry.logicalAddress = 0xA0000000;
-    entry.translatedAddress = 0x80000000;
+    entry.translatedAddress = 0xA0000000;
    entry.size = AMMU.Large_512M;
    entry.L1_cacheable = AMMU.CachePolicy_NON_CACHEABLE;
    entry.L1_posted = AMMU.PostedPolicy_NON_POSTED;
    entry.L2_cacheable = AMMU.CachePolicy_NON_CACHEABLE;
    entry.L2_posted = AMMU.PostedPolicy_NON_POSTED;
-/*
+
    var entry = AMMU.largePages[3];
    entry.pageEnabled = AMMU.Enable_YES;
    entry.translationEnabled = AMMU.Enable_YES;
@@ -221,6 +221,6 @@ function init()
    entry.L1_posted = AMMU.PostedPolicy_NON_POSTED;
    entry.L2_cacheable = AMMU.CachePolicy_NON_CACHEABLE;
    entry.L2_posted = AMMU.PostedPolicy_NON_POSTED;
-*/
+
 }

diff --git a/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/Ammu2_bios.cfg
b/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/Ammu2_bios.cfg
old mode 100644
new mode 100755
index 21d55cc..f44a488
--- a/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/Ammu2_bios.cfg
+++ b/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/Ammu2_bios.cfg
```

```
@@ -204,13 +204,13 @@ function init()
    entry.pageEnabled = AMMU.Enable_YES;
    entry.translationEnabled = AMMU.Enable_YES;
    entry.logicalAddress = 0xA0000000;
-    entry.translatedAddress = 0x80000000;
+    entry.translatedAddress = 0xA0000000;
    entry.size = AMMU.Large_512M;
    entry.L1_cacheable = AMMU.CachePolicy_NON_CACHEABLE;
    entry.L1_posted = AMMU.PostedPolicy_NON_POSTED;
    entry.L2_cacheable = AMMU.CachePolicy_NON_CACHEABLE;
    entry.L2_posted = AMMU.PostedPolicy_NON_POSTED;
-/*
+
    var entry = AMMU.largePages[3];
    entry.pageEnabled = AMMU.Enable_YES;
    entry.translationEnabled = AMMU.Enable_YES;
@@ -221,5 +221,5 @@ function init()
    entry.L1_posted = AMMU.PostedPolicy_NON_POSTED;
    entry.L2_cacheable = AMMU.CachePolicy_NON_CACHEABLE;
    entry.L2_posted = AMMU.PostedPolicy_NON_POSTED;
-*/
+
 }
\ No newline at end of file
diff --git a/links_fw/src/rtos/links_ipu/system/system_bsp_init.c
b/links_fw/src/rtos/links_ipu/system/system_bsp_init.c
index f4a4b80..63bad9d 100755
--- a/links_fw/src/rtos/links_ipu/system/system_bsp_init.c
+++ b/links_fw/src/rtos/links_ipu/system/system_bsp_init.c
@@ -265,10 +265,10 @@ Int32 System_bspInit(void)
        vpsInitPrms.virtBaseAddr = 0x80000000U;
        vpsInitPrms.physBaseAddr = 0x80000000U;
 #else
-       vpsInitPrms.virtBaseAddr = 0xA0000000U;
+       vpsInitPrms.virtBaseAddr = 0x80000000U;
        vpsInitPrms.physBaseAddr = 0x80000000U;
        /* if Virtual address != Physical address then enable translation */
-       vpsInitPrms.isAddrTransReq = TRUE;
+       vpsInitPrms.isAddrTransReq = FALSE;
 #endif
        Vps_printf(" SYSTEM: VPDMA Descriptor Memory Address translation"
                  " ENABLED [0x%08x -> 0x%08x]\n",
--
```

6.6 How To - Modify 1GB Linux Memory map to 512MB Linux Memory map

Assume TDA2xx build

6.6.1 Sample patch for TDA2xx VSDK files modification

Subject: [PATCH] [vsdk.30] - TDA2x 512MB linux build

- VSDK changes to make TDA2x 512MB linux build

```
        - All EVE cores and IPU1 are disabled
---
 apps/build/tda2xx/mem_segment_definition_linux.xs  | 70 ++++++++++++----------
 apps/configs/tda2xx_evm_linux_all/cfg.mk          | 10 ++--
 build/Rules.make                                  |  2 +-
 links_fw/include/link_api/system_vring_config.h    | 16 ++---
 links_fw/src/hlos/osa/include/osa_mem_map.h        | 15 ++---
 .../tda2xx/ipu2/gen_system_mem_map.xs             |  6 +-
 .../links_common/system/system_rsc_table_ipu.h     | 24 ++++++--
 .../src/rtos/links_ipu/system/system_bsp_init.c    |  4 +-
 8 files changed, 85 insertions(+), 62 deletions(-)
 mode change 100644 => 100755 links_fw/include/link_api/system_vring_config.h
 mode change 100644 => 100755 links_fw/src/rtos/links_common/system/system_rsc_table_ipu.h

diff --git a/apps/build/tda2xx/mem_segment_definition_linux.xs
b/apps/build/tda2xx/mem_segment_definition_linux.xs
index 405d123..e66832c 100755
--- a/apps/build/tda2xx/mem_segment_definition_linux.xs
+++ b/apps/build/tda2xx/mem_segment_definition_linux.xs
@@ -12,6 +12,9 @@
  *  ======== Single file for the memory map configuration of all cores =========
  */

+function getMemSegmentDefinition_external(core)
+{
+
 KB=1024;
 MB=KB*KB;

@@ -49,12 +52,12 @@ SR2_BASE_ADDR             0xA9000000
 A15-Linux                0xC0000000(End of Interleaving)
 *************************/
 DDR3_ADDR                = 0x80000000;
-DDR3_SIZE                = 1024*MB;
+DDR3_SIZE                = 512*MB;
 DDR3_LINUX_MEM_OFFSET      = 64*MB;

 /* First 512 MB - cached */
 DDR3_BASE_ADDR_0          = 0x80000000;
-DDR3_BASE_SIZE_0          = 448*MB + DDR3_LINUX_MEM_OFFSET;
+DDR3_BASE_SIZE_0          = 507*MB;


 /* The start address of the second mem section should be 16MB aligned.
@@ -62,13 +65,22 @@ DDR3_BASE_SIZE_0          = 448*MB + DDR3_LINUX_MEM_OFFSET;
  * to map SR0, REMOTE_LOG_MEM sections.
  * tlb_config_eveX.c need to be modified otherwise
  */
-DDR3_BASE_ADDR_1          = 0xA0000000;
-DDR3_BASE_SIZE_1          = 512*MB;
+DDR3_BASE_ADDR_1          = DDR3_BASE_ADDR_0 + DDR3_BASE_SIZE_0;
```

```
+DDR3_BASE_SIZE_1          = DDR3_SIZE - DDR3_BASE_SIZE_0;
+
+if(core=="ipu1_1" || core=="ipu1_0" || core=="ipu2")
+{
+  /*  for ipu1_0, ipu1_1, ipu2 DDR3_BASE_ADDR_1 should be
+   *  in non-cached virtual address of
+   *  DDR3_BASE_ADDR_1 + 512*MB
+   */
+  DDR3_BASE_ADDR_1       = DDR3_BASE_ADDR_1+512*MB;
+}

 /* Address and Size definitions of different components running on different cores */
 NDK_START_ADDR            = DDR3_BASE_ADDR_0 + DDR3_LINUX_MEM_OFFSET;
 NDK_MEM_SIZE              =   2*MB
-SR1_FRAME_BUFFER_SIZE      = 250*MB;
+SR1_FRAME_BUFFER_SIZE      = 205*MB;
 SR1_BUFF_ECC_ASIL_SIZE     =   4*KB;
 SR1_BUFF_ECC_QM_SIZE       =   4*KB;
 SR1_BUFF_NON_ECC_ASIL_SIZE =   4*KB;
@@ -83,20 +95,20 @@ if (ipummSupport == "yes") {
   IPU2_DATA_SIZE          =  60*MB;
 }
 else {
-  IPU2_DATA_SIZE          =  20*MB;
+  IPU2_DATA_SIZE          =  16*MB;
 }

-IPU1_START_ADDR           = 0x9e000000;
-IPU1_0_CODE_SIZE          =   8*MB;
-IPU1_0_DATA_SIZE          =  21*MB;
+IPU1_START_ADDR           = 0x9D000000;
+IPU1_0_CODE_SIZE          =   1*KB;
+IPU1_0_DATA_SIZE          =   1*KB;

-DSP1_START_ADDR           = 0xA1000000;
+DSP1_START_ADDR           = 0x9B000000;
 DSP1_CODE_SIZE           =   2*MB;
-DSP1_DATA_SIZE           =  24*MB;
+DSP1_DATA_SIZE           =  12*MB;

-DSP2_START_ADDR           = 0xA3000000;
+DSP2_START_ADDR           = 0x9C000000;
 DSP2_CODE_SIZE           =   2*MB;
-DSP2_DATA_SIZE           =  14*MB;
+DSP2_DATA_SIZE           =  12*MB;

 /* The start address of the second mem section should be 16MB aligned.
  * This alignment is a must as a single 16MB mapping is used for EVE
@@ -118,25 +130,25 @@ PM_DATA_LEN              =  512*KB;
 OPENVX_SHM_SIZE           =    2*MB;
 /* The start address of EVE memory must be 16MB aligned. */
-EVE_START_ADDR            = 0xA5000000;
+EVE_START_ADDR            = 0x9D100000;
 /* EVE vecs space should be align with 16MB boundary, and if possible try to fit
  * the entire vecs+code+data in 16MB section. In this case a single TLB map would
  * be enough to map vecs+code+data of an EVE.
  * tlb_config_eveX.c need to be modified if any of these EVE memory sections or
  * SR1_FRAME_BUFFER_MEM section is modified.
  */
-EVE1_VECS_SIZE           = 0.5*MB;
-EVE1_CODE_SIZE           =   2*MB;
-EVE1_DATA_SIZE           =13.5*MB;
-EVE2_VECS_SIZE           = 0.5*MB;
-EVE2_CODE_SIZE           =   2*MB;
-EVE2_DATA_SIZE           =13.5*MB;
-EVE3_VECS_SIZE           = 0.5*MB;
-EVE3_CODE_SIZE           =   2*MB;
-EVE3_DATA_SIZE           =13.5*MB;
-EVE4_VECS_SIZE           = 0.5*MB;
-EVE4_CODE_SIZE           =   2*MB;
-EVE4_DATA_SIZE           =13.5*MB;
+EVE1_VECS_SIZE           =1*KB;
+EVE1_CODE_SIZE           =1*KB;
+EVE1_DATA_SIZE           =1*KB;
+EVE2_VECS_SIZE           =1*KB;
+EVE2_CODE_SIZE           =1*KB;
+EVE2_DATA_SIZE           =1*KB;
+EVE3_VECS_SIZE           =1*KB;
+EVE3_CODE_SIZE           =1*KB;
+EVE3_DATA_SIZE           =1*KB;
+EVE4_VECS_SIZE           =1*KB;
+EVE4_CODE_SIZE           =1*KB;
+EVE4_DATA_SIZE           =1*KB;

 TOTAL_MEM_SIZE            = (DDR3_SIZE);

@@ -209,12 +221,10 @@ if (A15TargetOS == "Qnx") {
 }
 else {
   /* Shared Region handled by A15 HLOS Linux*/
-  SR2_BASE_ADDR        = 0xA9000000;
-  SR2_SIZE             = 0x4000000;
+  SR2_BASE_ADDR        = 0x9D200000;
+  SR2_SIZE             = 0x2000000;
 }

-function getMemSegmentDefinition_external(core)
-{
    var memory = new Array();
    var index = 0;
```

```
diff --git a/apps/configs/tda2xx_evm_linux_all/cfg.mk b/apps/configs/tda2xx_evm_linux_all/cfg.mk
index 1e32d6b..bc445b9 100755
--- a/apps/configs/tda2xx_evm_linux_all/cfg.mk
+++ b/apps/configs/tda2xx_evm_linux_all/cfg.mk
@@ -23,10 +23,10 @@ PROC_IPU2_INCLUDE=yes
 PROC_A15_0_INCLUDE=yes
 PROC_DSP1_INCLUDE=yes
 PROC_DSP2_INCLUDE=yes
-PROC_EVE1_INCLUDE=yes
-PROC_EVE2_INCLUDE=yes
-PROC_EVE3_INCLUDE=yes
-PROC_EVE4_INCLUDE=yes
+PROC_EVE1_INCLUDE=no
+PROC_EVE2_INCLUDE=no
+PROC_EVE3_INCLUDE=no
+PROC_EVE4_INCLUDE=no


 VSDK_BOARD_TYPE=TDA2XX_EVM
@@ -112,7 +112,7 @@ DATA_VIS_INCLUDE=no
 #
 # Enable below macro to enable OPENVX into Vision SDK
 #
-OPENVX_INCLUDE=yes
+OPENVX_INCLUDE=no

 CIO_REDIRECT=yes

diff --git a/build/Rules.make b/build/Rules.make
index 000ddd4..e061397 100755
--- a/build/Rules.make
+++ b/build/Rules.make
@@ -55,7 +55,7 @@ MAKEAPPNAME?=apps
 #    tda2px_evm_bios_iss
 #
 #
-MAKECONFIG?=tda2xx_evm_bios_all
+MAKECONFIG?=tda2xx_evm_linux_all

 # Default build environment
 # Options: Windows_NT or Linux
diff --git a/links_fw/include/link_api/system_vring_config.h
b/links_fw/include/link_api/system_vring_config.h
old mode 100644
new mode 100755
index 3a6b0e9..b062abb
--- a/links_fw/include/link_api/system_vring_config.h
+++ b/links_fw/include/link_api/system_vring_config.h
@@ -107,11 +107,11 @@ extern "C" {
 #define IPU_MEM_VRING_BUFS0     0x60040000
 #define IPU_MEM_VRING_BUFS1     0x60080000
```

```
-#define DSP_MEM_IPC_VRING       0xA0000000
-#define DSP_MEM_RPMSG_VRING0    0xA0000000
-#define DSP_MEM_RPMSG_VRING1    0xA0004000
-#define DSP_MEM_VRING_BUFS0     0xA0040000
-#define DSP_MEM_VRING_BUFS1     0xA0080000
+#define DSP_MEM_IPC_VRING       0x9FB00000
+#define DSP_MEM_RPMSG_VRING0    0x9FB00000
+#define DSP_MEM_RPMSG_VRING1    0x9FB04000
+#define DSP_MEM_VRING_BUFS0     0x9FB40000
+#define DSP_MEM_VRING_BUFS1     0x9FB80000

 /*
  * Vring physical addresses
@@ -129,15 +129,15 @@ extern "C" {
  */

 #ifdef BUILD_M4_0
-#define IPU_PHYS_MEM_IPC_VRING      0x9e000000
+#define IPU_PHYS_MEM_IPC_VRING      0x9D000000
 #endif

 #ifdef BUILD_DSP_1
-#define DSP_PHYS_MEM_IPC_VRING      0xa1000000
+#define DSP_PHYS_MEM_IPC_VRING      0x9B000000
 #endif

 #ifdef BUILD_DSP_2
-#define DSP_PHYS_MEM_IPC_VRING      0xa3000000
+#define DSP_PHYS_MEM_IPC_VRING      0x9C000000
 #endif

 #ifdef BUILD_M4_2
diff --git a/links_fw/src/hlos/osa/include/osa_mem_map.h b/links_fw/src/hlos/osa/include/osa_mem_map.h
index a83dca5..9eab2b8 100644
--- a/links_fw/src/hlos/osa/include/osa_mem_map.h
+++ b/links_fw/src/hlos/osa/include/osa_mem_map.h
@@ -11,23 +11,20 @@
 #define _SYSTEM_MEM_MAP_H_


-#define SR0_ADDR    0xa0100000
+#define SR0_ADDR    0x9fc00000
 #define SR0_SIZE    0x100000

-#define SYSTEM_IPC_SHM_MEM_ADDR    0xa02c0000
+#define SYSTEM_IPC_SHM_MEM_ADDR    0x9fdc0000
 #define SYSTEM_IPC_SHM_MEM_SIZE    0x80000

-#define REMOTE_LOG_MEM_ADDR      0xa0200000
+#define REMOTE_LOG_MEM_ADDR      0x9fd00000
```

```
 #define REMOTE_LOG_MEM_SIZE      0x40000

 #define SR1_FRAME_BUFFER_MEM_ADDR 0x84203000
-#define SR1_FRAME_BUFFER_MEM_SIZE 0xfa00000
+#define SR1_FRAME_BUFFER_MEM_SIZE 0xcd00000

-#define SR2_FRAME_BUFFER_MEM_ADDR 0xa9000000
-#define SR2_FRAME_BUFFER_MEM_SIZE 0x4000000
-
-#define OPENVX_OBJ_DESC_MEM_ADDR 0xa0530000
-#define OPENVX_OBJ_DESC_MEM_SIZE 0x200000
+#define SR2_FRAME_BUFFER_MEM_ADDR 0x9d200000
+#define SR2_FRAME_BUFFER_MEM_SIZE 0x2000000

 #endif /* _SYSTEM_MEM_MAP_H_ */

diff --git a/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/gen_system_mem_map.xs
b/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/gen_system_mem_map.xs
index 93c07eb..929d0ad 100755
--- a/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/gen_system_mem_map.xs
+++ b/links_fw/src/rtos/bios_app_common/tda2xx/ipu2/gen_system_mem_map.xs
@@ -48,13 +48,13 @@ function GenSystemMemMap()
    fd.writeLine("#define _SYSTEM_MEM_MAP_H_");
    fd.writeLine("");
    fd.writeLine("");
-   fd.writeLine("#define SR0_ADDR    " + Value2HexString(SR0.base));
+   fd.writeLine("#define SR0_ADDR    " + Value2HexString((SR0.base)-0x20000000));
    fd.writeLine("#define SR0_SIZE    " + Value2HexString(SR0.len) );
    fd.writeLine("");
-   fd.writeLine("#define SYSTEM_IPC_SHM_MEM_ADDR    " +
Value2HexString(SYSTEM_IPC_SHM_MEM.base));
+   fd.writeLine("#define SYSTEM_IPC_SHM_MEM_ADDR    " +
Value2HexString((SYSTEM_IPC_SHM_MEM.base)-0x20000000));
    fd.writeLine("#define SYSTEM_IPC_SHM_MEM_SIZE    " + Value2HexString(SYSTEM_IPC_SHM_MEM.len) );
    fd.writeLine("");
-   fd.writeLine("#define REMOTE_LOG_MEM_ADDR      " + Value2HexString(REMOTE_LOG_MEM.base));
+   fd.writeLine("#define REMOTE_LOG_MEM_ADDR      " + Value2HexString((REMOTE_LOG_MEM.base)-
0x20000000));
    fd.writeLine("#define REMOTE_LOG_MEM_SIZE      " + Value2HexString(REMOTE_LOG_MEM.len) );
    fd.writeLine("");
    fd.writeLine("#define SR1_FRAME_BUFFER_MEM_ADDR " + Value2HexString(SR1.base));
diff --git a/links_fw/src/rtos/links_common/system/system_rsc_table_ipu.h
b/links_fw/src/rtos/links_common/system/system_rsc_table_ipu.h
old mode 100644
new mode 100755
index f97437a..4789fc5
--- a/links_fw/src/rtos/links_common/system/system_rsc_table_ipu.h
+++ b/links_fw/src/rtos/links_common/system/system_rsc_table_ipu.h
@@ -79,9 +79,9 @@ Limited License.

 /* Number of entries in resource table */
 #ifdef IPU1_LOAD_EVES
-#define RSC_NUM_ENTRIES        19
+#define RSC_NUM_ENTRIES        20
 #else
-#define RSC_NUM_ENTRIES        18
+#define RSC_NUM_ENTRIES        19
 #endif

 /* IPU Memory Map */
@@ -107,10 +107,13 @@ Limited License.
 #define IPU_TILER_MODE_0_1     0xA0000000

 #define L3_TILER_MODE_2        0x70000000
-#define IPU_TILER_MODE_2       0xB0000000
+#define IPU_TILER_MODE_2       0xA1000000

 #define L3_TILER_MODE_3        0x78000000
-#define IPU_TILER_MODE_3       0xB8000000
+#define IPU_TILER_MODE_3       0xB0000000
+
+#define L3_AMMU_NONCACHED      0x9FB00000
+#define IPU_AMMU_NONCACHED     0xBFB00000

 #define IPU_MEM_TEXT           0x0

@@ -126,7 +129,7 @@ Limited License.
 #define IPU_MEM_IPC_DATA       XDC_CFG_IPC_DATA
 #define IPU_NDK_MEM            XDC_CFG_NDK_MEM
 #define SYSTEM_COMMON_SHM_VIRT  XDC_CFG_SYSTEM_COMMON_SHM_VIRT
-#define SYSTEM_COMMON_SHM      XDC_CFG_SYSTEM_COMMON_SHM_VIRT
+#define SYSTEM_COMMON_SHM      (XDC_CFG_SYSTEM_COMMON_SHM_VIRT-0x20000000)
 #define EVE_MEM_VIRT           XDC_CFG_EVE_MEM
 #define EVE_MEM                XDC_CFG_EVE_MEM

@@ -227,6 +230,9 @@ struct my_resource_table {
     /* devmem entry */
     struct fw_rsc_devmem devmem13;

+    /* devmem entry */
+    struct fw_rsc_devmem devmem14;
+
 };

 extern char ti_trace_SysMin_Module_State_0_outbuf__A;
@@ -262,6 +268,7 @@ struct my_resource_table ti_ipc_remoteproc_ResourceTable = {
         offsetof(struct my_resource_table, devmem12),
 #endif
         offsetof(struct my_resource_table, devmem13),
+        offsetof(struct my_resource_table, devmem14),
     },
    /* rpmsg vdev entry */
@@ -380,6 +387,13 @@ struct my_resource_table ti_ipc_remoteproc_ResourceTable = {
        SR2_VIRT, SR2_PHYS,
        SR2_SIZE, 0, 0, "SR2_MEM",
    },
+
+   {
+       TYPE_DEVMEM,
+       IPU_AMMU_NONCACHED, L3_AMMU_NONCACHED,
+       0x500000, 0, 0, "IPU_AMMU_NONCACHED",
+   },
+
 };

 #endif /* _RSC_TABLE_IPU_H_ */
diff --git a/links_fw/src/rtos/links_ipu/system/system_bsp_init.c
b/links_fw/src/rtos/links_ipu/system/system_bsp_init.c
index 97287ff..5a18372 100755
--- a/links_fw/src/rtos/links_ipu/system/system_bsp_init.c
+++ b/links_fw/src/rtos/links_ipu/system/system_bsp_init.c
@@ -251,8 +251,10 @@ Int32 System_bspInit(void)

 #ifdef A15_TARGET_OS_LINUX
       /* This one to one mapping is required for the 1GB Linux builds */
-       vpsInitPrms.virtBaseAddr = 0x80000000U;
+       vpsInitPrms.virtBaseAddr = 0xA0000000U;
        vpsInitPrms.physBaseAddr = 0x80000000U;
+       /* if Virtual address != Physical address then enable translation */
+       vpsInitPrms.isAddrTransReq = TRUE;
 #else
        vpsInitPrms.virtBaseAddr = 0xA0000000U;
        vpsInitPrms.physBaseAddr = 0x80000000U;
--
```

6.6.2 Sample patch for TDA2xx Linux Kernel modification

```
Subject: [PATCH] Linux kernel- TDA2x 512MB VSDK build
     - Changes in DRA7x infoAdas dts for 512MB VSDK linux build
---
 arch/arm/boot/dts/dra7-evm-infoadas.dts | 22 +++++++++++-----------
 1 file changed, 11 insertions(+), 11 deletions(-)

diff --git a/arch/arm/boot/dts/dra7-evm-infoadas.dts b/arch/arm/boot/dts/dra7-evm-infoadas.dts
index c8fd926..98cf975 100755
--- a/arch/arm/boot/dts/dra7-evm-infoadas.dts
+++ b/arch/arm/boot/dts/dra7-evm-infoadas.dts
@@ -61,19 +61,19 @@

 /* Update the CMA regions for Vision SDK binaries */
 &ipu2_cma_pool {
-      reg = <0x0 0x99000000 0x0 0x5000000>;
```

```
+     reg = <0x0 0x99000000 0x0 0x2000000>;
 };

 &dsp1_cma_pool {
-     reg = <0x0 0xa1000000 0x0 0x2000000>;
+     reg = <0x0 0x9B000000 0x0 0x1000000>;
 };

 &ipu1_cma_pool {
-     reg = <0x0 0x9e000000 0x0 0x2000000>;
+     reg = <0x0 0x9D000000 0x0 0x100000>;
 };

 &dsp2_cma_pool {
-     reg = <0x0 0xa3000000 0x0 0x2000000>;
+     reg = <0x0 0x9C000000 0x0 0x1000000>;
 };

 &reserved_mem {
@@ -85,28 +85,28 @@
                              status = "okay";
      };

-     cmem_pool: cmem@A9000000 {
+     cmem_pool: cmem@9D200000 {
                              compatible = "shared-dma-pool";
-                             reg = <0x0 0xA9000000 0x0 0x4000000>;
+                             reg = <0x0 0x9D200000 0x0 0x2000000>;
                              no-map;
                              status = "okay";
      };

      vsdk_sr1_mem: vsdk_sr1_mem@84000000 {
                              compatible = "shared-dma-pool";
-                             reg = <0x0 0x84000000 0x0 0x15000000>;
+                             reg = <0x0 0x84000000 0x0 0xD000000>;
                              status = "okay";
      };

-     vsdk_sr0_mem: vsdk_sr0_mem@A0000000 {
+     vsdk_sr0_mem: vsdk_sr0_mem@9FB00000 {
                              compatible = "shared-dma-pool";
-                             reg = <0x0 0xA0000000 0x0 0x1000000>;
+                             reg = <0x0 0x9FB00000 0x0 0x500000>;
                              status = "okay";
      };

-     vsdk_eve_mem: vsdk_eve_mem@A5000000 {
+     vsdk_eve_mem: vsdk_eve_mem@9D100000 {
                              compatible = "shared-dma-pool";
-                             reg = <0x0 0xA5000000 0x0 0x4000000>;
+                          reg = <0x0 0x9D100000 0x0 0x100000>;
                           status = "okay";
       };
 };
--
```

6.7 Moving reserved memory entries to 'high-mem' in Linux builds

For Linux configurations (such as tda2xx_evm_linux_all, tda2ex_linux_infoadas etc.) 500-600MB of data is carved-out as a 'reserved memory' entry to prevent the kernel from overwriting the contents within this address space. A summary of the memory-maps for various configurations is listed below

Section	tda2xx-linux(MB)	tda2ex-linux(MB)	tda2ex_17x17-linux(MB)	tda2px-linux(MB)	tda2xx-infoadas(MB)	tda2ex-infoadas(MB)	tda2px-infoadas(MB)
IPU-1 CMA	32	32	32	32	32	32	32
IPU-2 CMA	80	80	80	80	80	80	80
DSP-1 CMA	32	32	32	32	32	32	32
DSP-2 CMA	32	NA	NA	32	32	NA	32
EVEs CMA	64	NA	NA	64	64	NA	64
SR-0	16	16	16	16	16	16	16
SR-1	256	256	256	336	256	256	256
SR-2	64	64	64	64	64	64	64
Total	**576**	**480**	**480**	**656**	**576**	**480**	**576**

For all the configurations the 'reserved mem' entries lie within the address space 0x80000000 – 0xB0000000. This region also is the 'low-mem' region in the kernel. The 'low-mem' region in the kernel is where all kernel data-structures reside. By default the kernel is configured to work on a 1:3 split (***CONFIG_VMSPLIT_3G***), i.e in a system with 4GB RAM, the low-mem, vmalloc and high-mem regions are as below:

1. 0x80000000 – 0xB0000000 – Low-mem region (768MB)
2. 0xB0000000 – 0xC0000000 – Vmalloc region (256MB)
3. 0xC0000000 – 0xFFFFFFFFF – High-mem region (3GB)

With roughly 500-600MB reserved for shared region and CMA entries, often the kernel runs out of low-mem. To overcome the constraint, few sections can be moved. These are:

1. **DSP-1**
2. **DSP-2**
3. **EVEs**
4. **SR-1**

DSP-1, DSP-2 and EVEs CMA regions can be moved to high-mem. This can be achieved by referring to section-6.4 of this document. The address can be in the range 0xC0000000 – 0xFFFFFFFF depending on the size of the RAM available.

While moving SR-1 from low-mem the following constraints must be considered:

1. **The SR-1 region must be contained within 0xBFFFFFF:** The cached shared-region can be moved from low-mem address space to the vmalloc address space, i.e the SR-1 region can reside within 0xB0000000 – 0xBFFFFFFF. The shared-region can't reside outside the 0xBFFFFFFF boundary, since the AMMU configuration for IPU lists entries only upto 0xBFFFFFFF.

2. **All entries constituting SR-1 must be atleast 1MB in size:** SR-1 comprises of the following entities:
    a. SR1_FRAME_BUFFER_SIZE – typically greater than 200 MB
    b. SR1_BUFF_ECC_ASIL_SIZE – 4KB
    c. SR1_BUFF_ECC_QM_SIZE – 4KB
    d. SR1_BUFF_NON_ECC_ASIL_SIZE – 4KB

    When moving SR-1 outside low-mem, the entries '**SR1_BUFF_ECC_ASIL_SIZE**', '**SR1_BUFF_ECC_QM_SIZE**' and '**SR1_BUFF_NON_ECC_ASIL_SIZE**' must be set to atleast 1MB. This change is essential as the remote-proc driver on the kernel performs a 'dma_map_single' on page-table entries less than 64KB. This function expects an address within low-mem address space, and crashes if address is outside low-mem.

3. **Performance impact for direct CPU-access :** The region 0x80000000 – 0xA0000000 is mapped as cached region in the IPU-AMMU configuration, while the region 0xA000000 – 0xC0000000 is mapped as non-cached region. By moving SR-1 to non-cached region of IPU, use-cases which perform CPU-access from IPU will have a performance impact. While most use-cases perform DMA access, certain links such as the grpxSrcLink, algFrameDraw perform CPU access to update contents of buffers. These use-cases may notice a performance drop.

It's strongly advised not to move the below reserved-memory regions:

1. **IPU-2 CMA region**
2. **SR-0**
3. **NDK base-address**

An example indicating moving of SR-1 to high-mem is demonstrated below:

```
--- a/apps/build/tda2xx/mem_segment_definition_linux.xs
+++ b/apps/build/tda2xx/mem_segment_definition_linux.xs
@@ -62,10 +62,10 @@ DDR3_BASE_SIZE_1          = 512*MB;
 /* Address and Size definitions of different components running on different cores */
 NDK_START_ADDR            = DDR3_BASE_ADDR_0 + DDR3_LINUX_MEM_OFFSET;
 NDK_MEM_SIZE              =  2*MB
 SR1_FRAME_BUFFER_SIZE      = 250*MB;
-SR1_BUFF_ECC_ASIL_SIZE     =   4*KB;
-SR1_BUFF_ECC_QM_SIZE       =   4*KB;
-SR1_BUFF_NON_ECC_ASIL_SIZE  =   4*KB;
+SR1_BUFF_ECC_ASIL_SIZE     =   1*MB;
+SR1_BUFF_ECC_QM_SIZE       =   1*MB;
+SR1_BUFF_NON_ECC_ASIL_SIZE  =   1*MB;

 var ipummSupport = java.lang.System.getenv("IPUMM_INCLUDE");

@@ -133,7 +133,7 @@ TOTAL_MEM_SIZE            = (DDR3_SIZE);
  * So both of these sections should be one after another without any gap
  */
 NDK_MEM_ADDR              = NDK_START_ADDR
-SR1_BUFF_ECC_ASIL_ADDR     = NDK_MEM_ADDR            + NDK_MEM_SIZE;
+SR1_BUFF_ECC_ASIL_ADDR     = 0xb0000000;
 SR1_BUFF_ECC_QM_ADDR       = SR1_BUFF_ECC_ASIL_ADDR    + SR1_BUFF_ECC_ASIL_SIZE;
 SR1_BUFF_NON_ECC_ASIL_ADDR  = SR1_BUFF_ECC_QM_ADDR      + SR1_BUFF_ECC_QM_SIZE;
```

```
SR1_FRAME_BUFFER_ADDR       = SR1_BUFF_NON_ECC_ASIL_ADDR + SR1_BUFF_NON_ECC_ASIL_SIZE;

_ _
```

6.8 VSDK Linux builds for boards with greater than 2GB RAM

From Vision-SDK 3.05 support for Linux configurations on boards having greater than 2GB RAM is available. By default the RAM available for Linux is still set to 1GB. To use all the RAM available on the board, remove the argument “mem=1024M” from the uenv.txt file present in the boot media. If early-boot configuration with single-stage boot is used, no changes are required.

6.9 VSDK Memory Map FAQ

Some useful tips when you deal with bigger memory maps

- Always keep Linux Kernel from 0x80000000 to 64MB (this is mandatory)
- Also keep IPU data/code in first 512MB as this area is cached from IPU by default
- Try to keep DSP and EVE in second 512MB or even beyond as the entire memory is cached from DSP/EVE (except the 16M section where SR0, SYSTEM_ IPC_SH, HDVPSS_DESC etc. are present)
- SR1 can be kept anywhere, but if possible try to keep in the first 512MB (This mandates if any IPU touches any of these memory buffers)

How can I use remote cores binaries built from VSDK along with GLSDK or processor SDK LINUX?

- It’s not advisable to mix different SDKs as you may find tons of issues with resource sharing violations, memory map overlaps etc.
- Better use VSDK Linux for building both A15 (Linux) executables and Remote cores (Bios) binaries

Revision History

Version #	Date	Author Name	Revision History
0.1	26/12/2016	Shiju S	First draft
0.2	28/6/2017	Shiju S	Updated for VSDK3.0
0.3	16/10/2017	Shiju S	Updated for VSDK3.1
0.4	05/04/2018	Shravan Karthik	Updated for VSDK3.3