EZSDK Memory Map

Contents

Memory Map in the current EZSDK (version 5.02 onwards)

This document explains the memory map (memory layout) used in EZSDK.

The total system memory is divided across various processors/subsystems. Following are the broad categories of usage:

1. **Linux Memory** - One or more memory partitions are given to the Linux kernel memory manager.
2. **CMEM** - Contiguous physical memory used by C6accel to create buffers. These buffers are passed to the DSP.
3. **Shared Region (Host Configured at Load Time)** - Memory used for creating shared regions that are dynamically configured from host ARM at the firmware load time. However, the default shared region (Id 0) is currently not configurable from host processor.
4. **Slave Local Heap (Host Configured at Load Time)** - Memory used for creating local heaps on slave processors that are dynamically configured from host at the firmware load time. BIOS system heap is currently not configurable from host.
5. **Slave Executable** - One or more memory segments used to place code and data sections in an executable. These segements are defined in XDC configuration files as explained later.
6. **Memory Configuration** - This memory is reserved for passing memory and other system information between host and slave processors.

The following table shows the default memory map for a system that has 1 GB of system memory.

MEMORY MAP TABLE (DEFAULT 1GB SYSTEM)

Category	Memory Segment Name	Length	Start Address	Usage
Linux Memory	LINUX_MEM_1	364 MB	0x80000000	First memory segment given to Linux Memory Manager in Kernel
CMEM (C6Accel)	CMEM	20 MB	0x96C00000	Contiguous memory segment used by C6accel
Slave Local Heap (DSP)	DSP_ALG_HEAP	20 MB	0x98000000	Used for allocating memory for DSP algorithms
Shared Region	IPC_SR_HOST_DSP	1 MB	0x99400000	Shared Region between Host and DSP
Slave Executable (DSP)	DSP_DATA	12 MB	0x99500000	DSP Executable data (bss, const etc.)

	DSP_CODE	0 MB	0x9A100000	DSP Executable text (may not be used by all DSP applications.
Shared Region	IPC_SR_MC_HDVICP2_HDVPSS (IPC_SR_VIDEO_M3_VPSS_M3)	1 MB	0x9A100000	Internal Shared Region between Media Controllers MC-HDVICP2 and MC-HDVPSS
Slave Local Heap (MC_HDVPSS)	MC_HDVPSS_INT_HEAP_CACHED (VPSS_M3_INT_HEAP_CACHED)	27 MB	0x9A200000	MC-HDVPSS local heap in the cached region, used for allocating internal buffers
Slave Local Heap (MC_HDVICP2)	MC_HDVICP2_INT_HEAP_CACHED (VIDEO_M3_INT_HEAP_CACHED)	32 MB	0x9BD00000	MC-HDVICP2 local heap in the cached region, used for codec memory allocation
Reserved - MC-HDVICP2 Firmware	---	7 MB	0x9DD00000	Media Controller HDVICP2 Executable Code and data
UIA/Logger SM buffer	LOGGER_SM	2 MB	0x9E400000	UIA/Logger buffer
Reserved - MC-HDVPSS Firmware	---	17 MB	0x9E600000	Media Controller HDVPSS Executable Code and data
Shared Region	IPC_SR_COMMON	2 MB	0x9F700000	Default Shared Region between Host, DSP, MC-HDVICP2 and MC-HDVPSS
Unused Hole	Unused	3 MB	0x9F900000	Unused memory, this can not be assigned to Linux, as linux needs to start at 4MB boundary
Linux Memory	LINUX_MEM_2	320 MB	0x9FC00000	Second memory segment given to Linux Memory Manager in Kernel
Shared Region	IPC_SR_FRAME_BUFFERS	188 MB	0xB3D00000	Shared Region between Host and MC-HDVICP2 and MC-HDVPSS, used to allocate all video buffers
Reserved - MC-HDVPSS Firmware	MC_HDVPSS_NOTIFY_MEM (HDVPSS_NOTIFY_MEM)	2 MB	0xBF900000	Used for IPC between Host and MC-HDVPSS. This is in non-cached memory region.
	MC_HDVPSS_V4L2_FBDEF_MEM (HDVPSS_V4L2_FBDEF_MEM)	2 MB	0xBFB00000	Used by V4L2 and Fbdev drivers. This is in non-cached memory region.
	MC_HDVPSS_DESC (HDVPSS_DESC)	2 MB	0xBFD00000	Used by HDVPSS driver data structures. This is in non-cached memory region.
Reserved for Memory Configuration	MEMCFG_SPACE	1 MB	0xBFF00000	Reserved. Used by firmware loader.

Color Code
Linux Memory
Memory used for regions that can be adjusted without rebuilding Media Controller executable.
Memory Reserved for Media Controller Executables

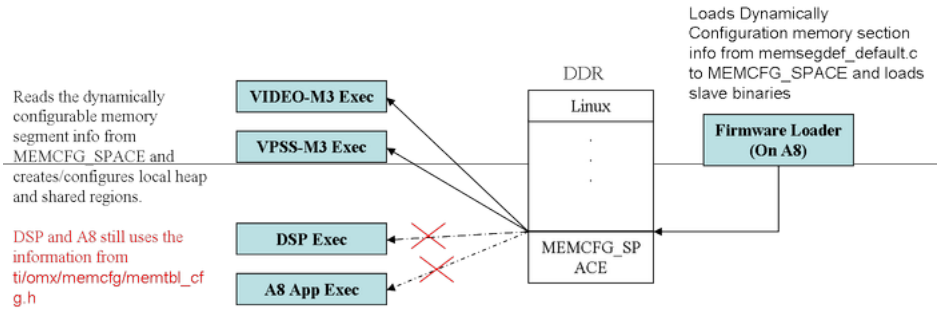
NOTE:

1. Shared Region IPC_SR_COMMON is statically configured within all slave executables.
2. SysBios System Heap is statically configured within each slave executable and the memory is allocated in the bss section.

H/W and S/W Limitations To Consider For Deciding Memory Map

1. Hardware Limitations
    1. Media Controller cannot access any program memory above 0xA0000000.
    2. Media Controller cannot access any data memory above 0xE0000000.
    3. AMMU in Media Controller handles large memory segments of size 512MB/32MB only and there can be 4 such segments. As one 512MB address range is used to access memory mapped registers, only 3 regions can be used to access DDR.
2. Software Limitations
    1. Default build of Linux handles memory of size 768MB. Configure different memory split for kernel/user memory split, or use HIGHMEM support to allocate more memory to Linux.
    2. Many memory sections are still statically defined within Media Controller firmwares and DSP executables.
        1. Default Shared Region (IPC_SR_COMMON) is statically defined in Media Controller firmware and DSP executables. Changing address or size of this requires changes in multiple places.
        2. Following are statically defined in Media Controller firmwares. These have to be placed in non-cached region.
            1. MC-HDVPSS hardware descriptors (MC_HDVPSS_DESC)
            2. Shared Memory for Fbdev (MC_HDVPSS_V4L2_FBDEF_MEM)
            3. Shared Memory used for IPC between HOST and MC-HDVPSS firmware (MC_HDVPSS_NOTIFY_MEM)
            4. SysBIOS System Heaps are statically defined within Media Controller firmwares and DSP executables.
        3. UIA/Log buffers are statically defined within Media Controller firmwares and DSP executables.
        4. Though Frame Buffers are accessed only by Hardware accelerators, still this memory section needs to be configured in the AMMU as Media Controller core still access frame buffer regions for the below reasons.
            1. OMX embeddeds Meta information about Frame at the end of frame buffer
            2. Shared Region modules embeds meta information in Frame Buffer regions.

Memory Configuration Flow



Firmware loader load the Dynamically configurable Memory Section info from firmware_loader/src/memsegdef_default.c to MEMCFG_SPACE before loading slave binaries.

During bootup of Video-M3 and VPSS-M3, they read the dynamically memory section info from MEMCFG_SPACE and creates the Local Heaps and Shared regions. This method is not yet integrated to DSP and A8 side of applications, DSP and A8 reads the dynamically configurable memory section info from ti/omx/memcfg/memtbl_cfg.h

Finding Memory Usage Statistics For Local Heaps and Shared Regions

You can use '**sys_top'** Linux utility to find out memory usage statistics

% sys_top

```
sys_top Ver : 0.2.0.0
Number Of Running Cores: 3

MC.HDVICP2:
 0 Heap:Size 2097152    Used 161720     MaxU 161720     Free 1935432    LarF 1709712
Num SR :4
SRIn 0 :PhyA 0x9f700000 Virt 0x0        Size 0x200000
 SRHeap:Size 2095488    Used 19072      MaxU 19072      Free 2076416    LarF 2075648
SRIn 1 :PhyA 0x9a100000 Virt 0x0        Size 0x100000
 SRHeap:Size 1048448    Used 0          MaxU 0          Free 1048448    LarF 1048448
SRIn 2 :PhyA 0xb3d00000 Virt 0x0        Size 0xbc00000
 SRHeap:Size 197132160  Used 0          MaxU 0          Free 197132160  LarF 197132160

MC.HDVPSS:
 0 Heap:Size 2097152    Used 107600     MaxU 107600     Free 1989552    LarF 1989552
 1 Heap:Size 28311552   Used 0          MaxU 0          Free 28311552   LarF 28311552
Num SR :4
SRIn 0 :PhyA 0x9f700000 Virt 0x0        Size 0x200000
 SRHeap:Size 2095488    Used 19072      MaxU 19072      Free 2076416    LarF 2075648
SRIn 1 :PhyA 0x9a100000 Virt 0x0        Size 0x100000
 SRHeap:Size 1048448    Used 0          MaxU 0          Free 1048448    LarF 1048448
SRIn 2 :PhyA 0xb3d00000 Virt 0x0        Size 0xbc00000
 SRHeap:Size 197132160  Used 0          MaxU 0          Free 197132160  LarF 197132160

Legend:
LarF - Largest Free size   SRIn - Shared Region Index
PhyA - Physical Address    Virt - Virtual Address
MaxU - Maximum Used Size   SR   - Shared Region
MC   - Media Controller
```

Configuring Slave Processor Heaps and Shared Regions From Host ARM

Local heaps on the slave processors and heaps in the shared regions can be made configurable from Linux at the time of loading of slave executable. Information about the memory segments for these heaps are provided to the firmware loader. Currently, this information is provided in the file ***memsegdef.c*** as a table. Each entry of the table is defined by the type ***LDR_Memseg*** and it can be found in the header file ***ldr_memseg.h***.

Following fields of the structure LDR_Memseg are used:

- **name** : Name of the memory segment
- **size** : Size of the segment in bytes
- **seg_type** : Type of the memory segment. LDR_SEGMENT_TYPE_DYMANMIC_LOACL_HEAP and LDR_SEGMENT_TYPE_DYNAMIC_SHARED_HEAP are used.
- **system_addr** : System address of the segment.
- **slave_virtual_addr** : Virtual address used by slave. Currently, it is same as system_addr.
- **master_core_id** : Core id for the processor that owns it.
- **core_id_mask** : Specifies set of core id masks for which the segment is valid
- **shared_region_id** : Shared Region Id, when used across multiple cores

In this release, following segments are configurable by the firmware loader.

- MC_HDVPSS_INT_HEAP_CACHED
- MC_HDVICP2_INT_HEAP_CACHED
- IPC_SR_FRAME_BUFFERS

- IPC_SR_MC_HDVICP2_HDVPSS

Modifying Memory Map

Linux Kernel Memory

Linux size is passed though the ***mem*** parameter in the bootargs environment variable, base address is always <LINUX_MEM_1> -

% setenv bootargs 'console=ttyO2,115200n8 root=/dev/nfs nfsroot=<nfs-server-ip>:<path-to-nfs-share>,nolock rw mem=<size of LINUX_MEM_1>@<addr of LINUX_MEM_1> mem=<size of LINUX_MEM_2>@<addr of LINUX_MEM_2>'

For 1GB default memory map,

% setenv bootargs 'console=ttyO2,115200n8 root=/dev/nfs nfsroot=<nfs-server-ip>:<path-to-nfs-share>,nolock rw mem=364M'

The second partition can be added to linux mem=320M@0x9FC00000 Ex: mem=364M mem=320M@0x9FC00000

When the 2 partition is set it throws below error while running EZSDK. "vmap allocation for size 197136384 failed: use vmalloc=500M to increase size."

To solve this rebuild the kernel with HIGHMEM support enabled. This can be enabled by menuconfig->Kernel Features->HIGH Memory Support. Please note Linux partiton memory needs to start and end at 4MB boundary.

For quick check you can pass vmalloc=500M to bootargs, but note that the actual physical memory which can be managed by Linux would less that what is passed via mem= bootargs.

Linux Notify Module Memory

The memory for kernel Notify module is specified through Linux bootargs. This address also used in the MC-HDVPSS firmware.

- **notifyk.vpssm3_sva=<addr of MC_HDVPSS_NOTIFY_MEM>**

FbDev Memory

Fbdev memory address is specified through the ***sbufaddr*** parameter.

- **insmod vpss.ko sbufaddr=<addr of MC_HDVPSS_V4L2_FBDEF_MEM>**

CMEM memory

CMEM memory address and size (used by DSP) are passed as module install arguments as shown below

- **insmod cmemk.ko phys_start=<addr of CMEM> phys_end=<addr of next segment> pools=20x4096**

Heaps and Shared Regions Configured at Load Time

To change memory for heaps and shared regions at load time,

1. Modify appropriate file with new memory map in ***$(EZSDK_INSTALL_DIR)/board-support/media-controller-utils_2_02_00_08/src/mm_host_util***
    1. Ex: memsegdef_dm81xxbm.c
2. Rebuild the mm_host_util, this is the host side utility
3. Run the utility to generate .bin file which needs to be fed to firmware loader as input file
    1. Ex: mm_dm81xxbm.bin file gets generated.
4. While loading the firmware pass the above generated .bin file in the command line argument
    1. Ex: ./firmware_loader 1 dm816x_hdvicp.xem3 start mm_dm81xxbm.bin
    2. Ex: ./firmware_loader 2 dm816x_hdvpss.xem3 start mm_dm81xxbm.bin

Note: if the .bin file not passed in the firmware command line, then the default memory map is used as per the memory map defined in file ***$(EZSDK_INSTALL_DIR)/board-support/media-controller-utils_2_02_00_08/src/firmware_loader/memsegdef_default.c***

MC-HDVICP2 & MC-HDVPSS firmwares

These are provided as binaries.

DSP for OMX

1. It uses the same memory segment definition file $(EZSDK_INSTALL_DIR)/component-sources/omx_05_02_00_09/packages/ti/omx/build/MemSegmentDefinition.xs.
2. DSP build also uses Shared Region information from file $(EZSDK_INSTALL_DIR)/component-sources/omx_05_02_00_09/packages/ti/omx/memcfg/memtbl_cfg.h through below defines, Ensure that these information matches the mem cfg defined in mm_host_util. This dependency will be removed in the patch release .
    1. MEMCFG_SRBASE1
    2. MEMCFG_SRSIZE1
    3. MEMCFG_SRBASE2
    4. MEMCFG_SRSIZE2

3. Rebuild DSP side OMX library & application.

Linux applications for OMX

1. Linux side app uses Shared Region information address & size from file as mentioned in the "DSP for OMX" sections. The same file is used in Linux side. This dependency will be removed in the patch release .
2. Rebuild Linux side OMX libraries & application

DSP C6accel Application

1. Modify the memory segments in file ***$(C6ACCEL_INSTALL_DIR)/soc/packages/ti/c6accel_unitservers/TI816X/serverplatforms.xs***
2. Rebuild DSP executable binary.

Excel File To Calculate Memory Map

```
Use the Excel file media:EZSDK_MemCfg.zip to define memory map
```

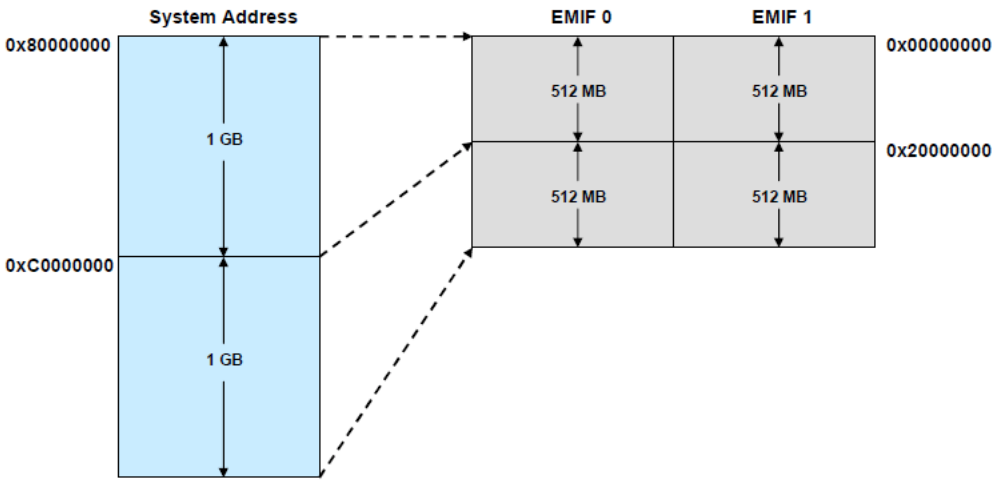
Changing Memory Map For 512MB DM816x Board

Following are the list of changes required for 512 MB memory map in DM816x board

Step 1: Update LISA Registers

In the default uboot program (in file $(EZSDK_ROOT)/board-support/u-boot-<REL-TAG>/board/<ti8168 or ti8148>/ti/evm.c), 2GB physical memory is mapped using register 3 and 4. Register 1 and 2 are unused.

- System Address 0x80000000 is mapped to physical address 0x00000000
- System Address 0xC0000000 is mapped to physical address 0x20000000

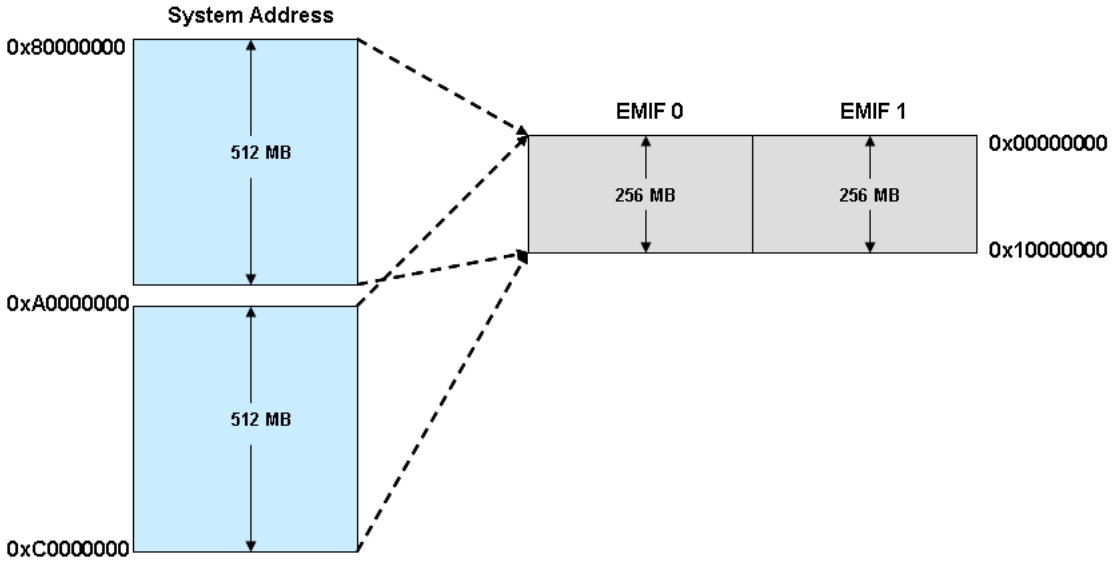


Following code programs the LISA registers for 2GB physical memory (default configuration).

```
/* Program the DMM to for 2 GB (interleaved) configuration */
__raw_writel(0x0, DMM_LISA_MAP__0); /* Register 0 is unused */
__raw_writel(0x0, DMM_LISA_MAP__1); /* Register 1 is unused */
__raw_writel(0x80640300, DMM_LISA_MAP__2); /* Register 2 maps 0x80000000 to 0x00000000, length 1GB */
__raw_writel(0xC0640320, DMM_LISA_MAP__3); /* Register 3 maps 0xC0000000 to 0x20000000, length 1GB */
```

For a 512MB system, we create 2 system address spaces of length 512MB and map to the same physical memory. On Media Controllers, the first address range is used to access DDR with cache enabled and the second address is used to access the same DDR with cache disabled.

- System Address 0x80000000 is mapped to physical address 0x00000000
- System Address 0xA0000000 is mapped to physical address 0x00000000



Following code programs the LISA registers for **512MB** physical memory.

```
/* Program the DMM to for 512MB (interleaved) configuration */
__raw_writel(0x0, DMM_LISA_MAP__0); /* Register 0 is unused */
__raw_writel(0x0, DMM_LISA_MAP__1); /* Register 1 is unused */
__raw_writel(0x80540300, DMM_LISA_MAP__2); /* Register 2 maps 0x80000000 to 0x00000000, length 512MB */
__raw_writel(0xA0540300, DMM_LISA_MAP__3); /* Register 3 maps 0xA0000000 to 0x00000000, length 512MB */
```

Note: For DM814X device ensure to update the while instructions immediately after the above lines in the file to reflect the same settings.

Step 2: Update Memory Map

In a 512MB system, Linux memory manager is given only one memory segment of reduced size and the Shared Region for video buffers is shifted to a different place to fit within 512MB. Changes can be found in the first two entries in the following table.

Step 2.1: Change Linux Memory size

Change Linux memory sizeto **176 MB** starting at 0x80000000

Step 2.2: Change Location of Video Frame Buffers

Change location of **IPC_SR_FRAME_BUFFERS** to **0xAB000000**. This change needs to be done in the firmware loader input file as explained earlier.

Step 2.3: Update memory map for DSP

Follow the steps mentioned in section EZSDK_Memory_Map#DSP_for_OMX.

Step 2.4: Update memory map for Linux side app

Follow the steps mentioned in section EZSDK_Memory_Map#Linux_applications_for_OMX.

Changing Memory Map For 512MB DM814x Board

Following are the list of changes required for 512 MB memory map in DM814x version 2.1 board

Step 1: Update LISA Registers

Update the macros **PG2_1_DMM_LISA_MAP__2** and **PG2_1_DMM_LISA_MAP__3** in **ddr_defs_ti814x.h** ($(EZSDK_ROOT)/board-support/u-boot-<REL-TAG>/arch/arm/include/asm/arch-ti81xx)

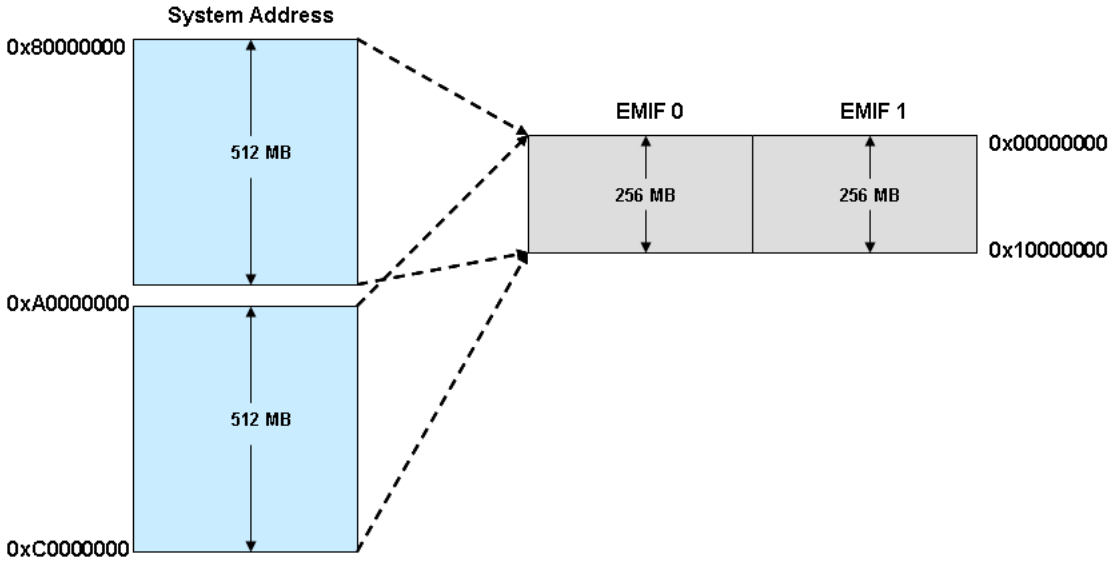
Default values

```
#define PG2_1_DMM_LISA_MAP__2 0x0
#define PG2_1_DMM_LISA_MAP__3 0x80640300
```

Modified values for 512 MB

```
#define PG2_1_DMM_LISA_MAP__2 0x80540300
#define PG2_1_DMM_LISA_MAP__3 0xA0540300
```

Re-build the u-boot with the new changes. Copy the generated u-boot.bin and MLO and use them to boot the board



Step 2: Update Memory Map

In a 512MB system, Linux memory manager is given only one memory segment of reduced size and the Shared Region for video buffers is shifted to a different place to fit within 512MB. Changes can be found in the first two entries in the following table.

Step 2.1: Change Linux Memory size

Change Linux memory size to **176 MB** . This can be done by setting the **MEM** to **176M** in bootargs

Step 2.2: Change Location of Video Frame Buffers

Change location of **IPC_SR_FRAME_BUFFERS** to **0xAB000000**. This change needs to be done in the firmware loader input file. Update the **system_addr** and **slave_virtual_addr** values for **IPC_SR_FRAME_BUFFERS** from **0xB3D00000** to **0xAB000000** in the source memsegdef_default.c ($(EZSDK_ROOT)/board-support/media-controller-utils_<REL-TAG>/src/firmware_loader). Rebuild the firmware loader and copy the generated firmware loader in the target filesystem ($(EZSDK_ROOT)/filesys/usr/bin)

Step 2.3: Update memory map for DSP

Follow the steps mentioned in section EZSDK_Memory_Map#DSP_for_OMX.

Step 2.4: Update memory map for Linux side app

Modify the macro **MEMCFG_SRBASE2** in memtbl_cfg.h ($(EZSDK_ROOT)/component-sources/omx_<REL-TAG>/src/ti/omx/memcfg) from **0xB3D00000** to **0xAB000000**. Rebuilt the OMX samples.

MEMORY MAP TABLE FOR DEFAULT 512MB SYSTEM

Memory segment Name	Length	System Address in Region Starting at 0x80000000	System Address in Region Starting at 0xA0000000
LINUX_MEM_1	***176 MB***	***0x80000000***	0xA0000000 (not to be accessed)
IPC_SR_FRAME_BUFFERS	***188 MB***	0x8B000000 (not to be accessed)	***0xAB000000***
CMEM	20 MB	0x96C00000	0xB6C00000 (not to be accessed)
DSP_ALG_HEAP	20 MB	0x98000000	
IPC_SR_HOST_DSP	1 MB	0x99400000	
DSP_DATA	12 MB	0x99500000	
DSP_CODE	0 MB	0x9A100000	
IPC_SR_MC_HDVICP2_HDVPSS (IPC_SR_VIDEO_M3_VPSS_M3)	1 MB	0x9A100000	
MC_HDVPSS_INT_HEAP_CACHED (VPSS_M3_INT_HEAP_CACHED)	27 MB	0x9A200000	
MC_HDVICP2_INT_HEAP_CACHED (VIDEO_M3_INT_HEAP_CACHED)	24 MB	0x9BD00000	
Reserved - HDVICP2 Media Controller Firmware	15 MB	0x9D500000	
LOGGER_SM	2 MB	0x9E400000	
Reserved - HDVPSS Media Controller Firmware	17 MB	0x9E600000	
IPC_SR_COMMON	2 MB	0x9F700000	

MC_HDVPSS_NOTIFY_MEM (HDVPSS_NOTIFY_MEM)	2 MB	0x9F900000 (not to be accessed)	0xBF900000
MC_HDVPSS_V4L2_FBDEF_MEM (HDVPSS_V4L2_FBDEF_MEM)	2 MB		0xBFB00000
MC_HDVPSS_DESC (HDVPSS_DESC)	2 MB		0xBFD00000
MEMCFG_SPACE	1 MB		0xBFF00000

Changing Memory map for 256MB board

Follow the steps mentioned in section EZSDK_Memory_Map#Want_to_change_memory_map_of_Firmware to change memory map

Below is the example memory map for 256MB

Sample MEMORY MAP TABLE FOR 256MB SYSTEM

Memory segment Name	Length	System Address in Region Starting at 0x80000000	System Address in Region Starting at 0xA0000000
LINUX_MEM_1	***145 MB***	***0x80000000***	0xA0000000 (not to be accessed)
IPC_SR_FRAME_BUFFERS	***16 MB***	0x89100000 (not to be accessed)	***0xA9100000***
CMEM	0 MB	0x8A100000	0xAA100000 (not to be accessed)
DSP_ALG_HEAP	0 MB	0x8A100000	
IPC_SR_HOST_DSP	0 MB	0x8A100000	
DSP_DATA	0 MB	0x8A100000	
DSP_CODE	0 MB	0x8A100000	
IPC_SR_MC_HDVICP2_HDVPSS (IPC_SR_VIDEO_M3_VPSS_M3)	1 MB	0x8A100000	
MC_HDVPSS_INT_HEAP_CACHED (VPSS_M3_INT_HEAP_CACHED)	27 MB	0x8A200000	
MC_HDVICP2_INT_HEAP_CACHED (VIDEO_M3_INT_HEAP_CACHED)	24 MB	0x8BD00000	
Reserved - HDVICP2 Media Controller Firmware	15 MB	0x8D500000	
LOGGER_SM	2 MB	0x8E400000	
Reserved - HDVPSS Media Controller Firmware	17 MB	0x8E600000	
IPC_SR_COMMON	2 MB	0x8F700000	
MC_HDVPSS_NOTIFY_MEM (HDVPSS_NOTIFY_MEM)	2 MB	0x8F900000 (not to be accessed)	0xAF900000
MC_HDVPSS_V4L2_FBDEF_MEM (HDVPSS_V4L2_FBDEF_MEM)	2 MB		0xAFB00000
MC_HDVPSS_DESC (HDVPSS_DESC)	2 MB		0xAFD00000
MEMCFG_SPACE	1 MB		0xAFF00000

Want to change memory map of Firmware

With NDA signed it is possible to get complete source to the Media Controller firmwares

In this case it is also possible to change memory map for reserved memroy sections also

To do this ensure follow below steps

Step1:

Update the MemSegmentDefinition.xs file as per your new memory map

Step2:

Update the below macro in file ldrmemcfg\ldr_memseg.h, this reflects the memory section MEMCFG_SPACE

```
#define LDR_CONFIG_ADDR_BASE          (0xBFF00000)
```

Step3:

Rebuild Media controller firmwares

Step4:

Follow steps mentioned in earlier sections to rebuild firmware loader, DSP & Linux images for the new memory map, Changes here needs to match the memory map defined in MemSegmentDefinition.xs file mentioned in step1

Step5:

Now app can be run with the new memory map.

Memory Section Legacy names & New name

LEGACY & NEW NAMEs

Sr Id	Legacy Name	New Name	Comment
1.	LINUX_PHYSICAL_ADDRESS_GAP, LINUX_MEM	LINUX_MEM_1 or 2	Two partitions used
2.	SHARED_CTRL, SR0, DDR3_SR0	IPC_SR_COMMON	
3.	EXTMEM_CORE0	VIDEO_M3_CODE	
4.	PRIVATE_CORE0_DATA	VIDEO_M3_DATA	The new section VIDEO_M3_INT_HEAP_CACHED also forked from this section
5.	EXTMEM_CORE1	VPSS_M3_CODE	
6.	PRIVATE_CORE1_DATA	VPSS_M3_DATA	The new section VIDEO_M3_INT_HEAP_CACHED also forked from this section
7.	SHARED_CTRL_DUCATI_ONLY_ACCESS	IPC_SR_VIDEO_M3_VPSS_M3	Dynamically configurable from firmware_loader
8.	SHARED_DATA	removed	
9.	EVENT_LIST_CORE0	VIDEO_M3_EVENT_BUFFER	
10.	EVENT_LIST_CORE1	VPSS_M3_EVENT_BUFFER	
11.	HDVPSS_DESCRIPTOR_NON_CACHED	HDVPSS_DESC	
12.	not exists	HDVPSS_NOTIFY_MEM	With the usage of Notify in kernel this needs to be defined.
13.	LOGGERSM	Remain Same	
14.	FQMEM_BUFFERS_NON_CACHED	IPC_SR_FRAME_BUFFERS	Dynamically configurable from firmware_loader
15.	Not exists	MEMCFG_SPACE	Region to pass dynamically configurable memory section info to slaves from firmware_loader.
16.	Not Exists	VIDEO_M3_INT_HEAP_CACHED	Refer memory section PRIVATE_CORE0_DATA
15.	Not Exists	VIDEO_M3_INT_HEAP_CACHED	Refer memory section PRIVATE_CORE1_DATA
17.	FBDEV_V4L2_Mem	HDVPSS_V4L2_FBDEF_MEM	

Additional References

1. Understanding the Linux Virtual Memory Manager (http://ptgmedia.pearsoncmg.com/images/0131453483/downloads/gorman_book.pdf)

Download the Latest EZSDK

The latest EZSDK is available for download from http://software-dl.ti.com/dsps/dsps_public_sw/ezsdk/latest/index_FDS.html.

The current version is **5.05.02.00**. The supported platforms are DM816x (http://www.ti.com/tool/tmdxevm8168) and DM814x (http://www.ti.com/tool/tmdxevm8148).

EZSDK Support

DM816x E2E Support Forum	http://e2e.ti.com/support/dsp/davinci_digital_media_processors/f/717.aspx
DM814x E2E Support Forum	http://e2e.ti.com/support/dsp/davinci_digital_media_processors/f/716.aspx
Linux E2E Support Forum	http://e2e.ti.com/support/embedded/linux/default.aspx
BIOS E2E Support Forum	http://e2e.ti.com/support/embedded/bios/default.aspx

{{

1. switchcategory:MultiCore=
   - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
   - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

   Please post only comments related to the article **EZSDK Memory Map** here.

Keystone=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **EZSDK Memory Map** here.

C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article **EZSDK Memory Map** here.*

DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article **EZSDK Memory Map** here.*

MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article **EZSDK Memory Map** here.*

OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **EZSDK Memory Map** here.*

OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **EZSDK Memory Map** here.*

MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article **EZSDK Memory Map** here.*

*For technical su please post you questions at http://e2e.ti.com Please post only comments abou article **EZSDK Memory Map** h*

}}

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity