User's Manual

April 20, 2017. Renesas Electronics Corporation.

INTEGRITY® Virtualization Environment Performance Evaluation Report rev1.3

1. Revision History

Following changes have been made on top of official End of Feb 28 release.

rev	Date	Description	Note
1.0	Feb 28, 2017	Created official release	
1.1	March 17, 2017	Update following section 2.1.1, 3.1, 5.1, 5.2, 5.4, 5.5.1, 5.5.2,5.6, 5.7, 5.8.1,5.9.3, 5.10.1, 5.11, 5.17.1, 5.19, 5.20, 5.23	
1.2	March 21, 2017	Update following section 5.2.4	
1.3	April 20,217	Update following section 5.4	

2. Project Requirements

Virtualization PoC project scope and requirements are based on the following documents (provided by Renesas):

- 1Soc_Evaluation_Parameter_v0.4.11.xlsx

2.1. System requirements

2.1.1. OEM Display / NAVI / HUD Architecture

The following architecture needs to be evaluated.

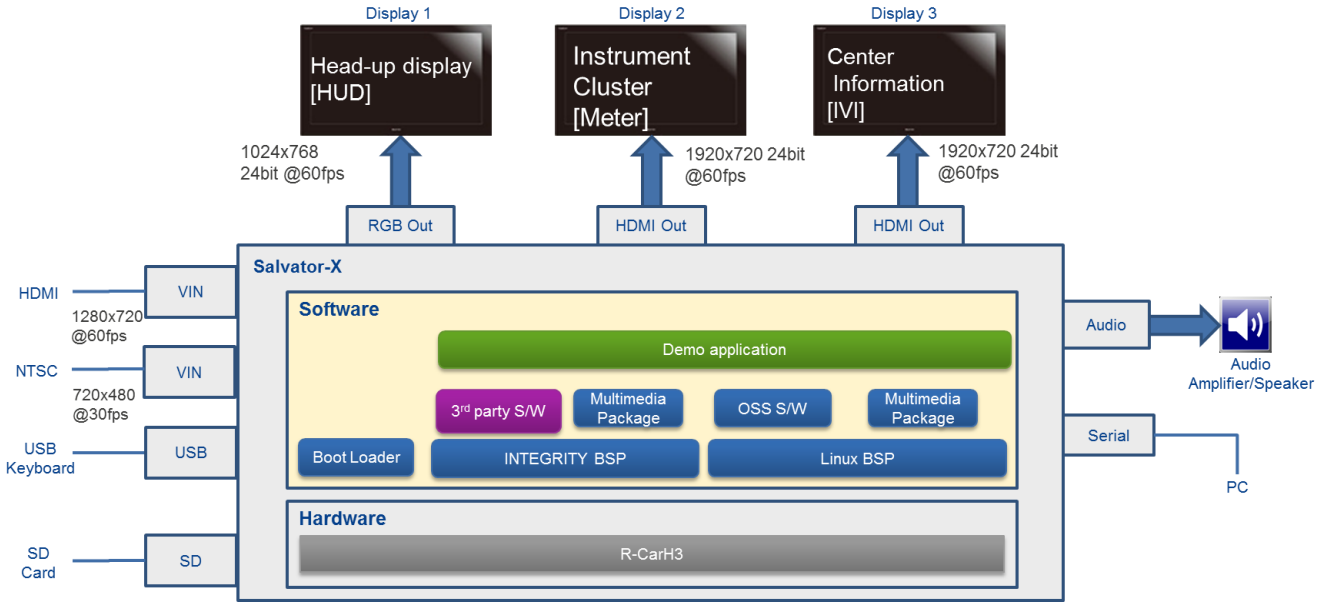


Figure 2-1: Image of System architecture

2.2. Metrics and parameters for evaluation

Virtualization PoC evaluates the following viewpoints.

- CPU Load
- Bus Load/Bandwidth
- Bus Latency
- Boot Time
- Interrupt Time
- Drawing Performance
- Video & Audio Performance
- Camera Performance
- Display Performance
- RAM I/O Performance
- Memory Performance
- Network Performance(Linux)
- Power Consumption Performance
- RTOS performance
- Application Switching performance
- Malicious App
- Robustness
- Rebooting of Linux
- Memory usage
- Stress Tolerance
- Security
- End-to-End Latency
- Memory Utilization of Each Module
- Network Performance(RTOS, Multivisor)

Currently we are using R-Car H3 WS1.1 with DDR2400. When moving to WS2.0, it will be set to 3200, which is expected to improve benchmark results.

3. Virtualization PoC Implementation. Setup and Hardware Configuration

3.1. Introduction

This chapter describes hardware required for virtualization PoC setup.

3.2. Virtualization PoC Setup

The following figure shows the virtualization PoC setup.

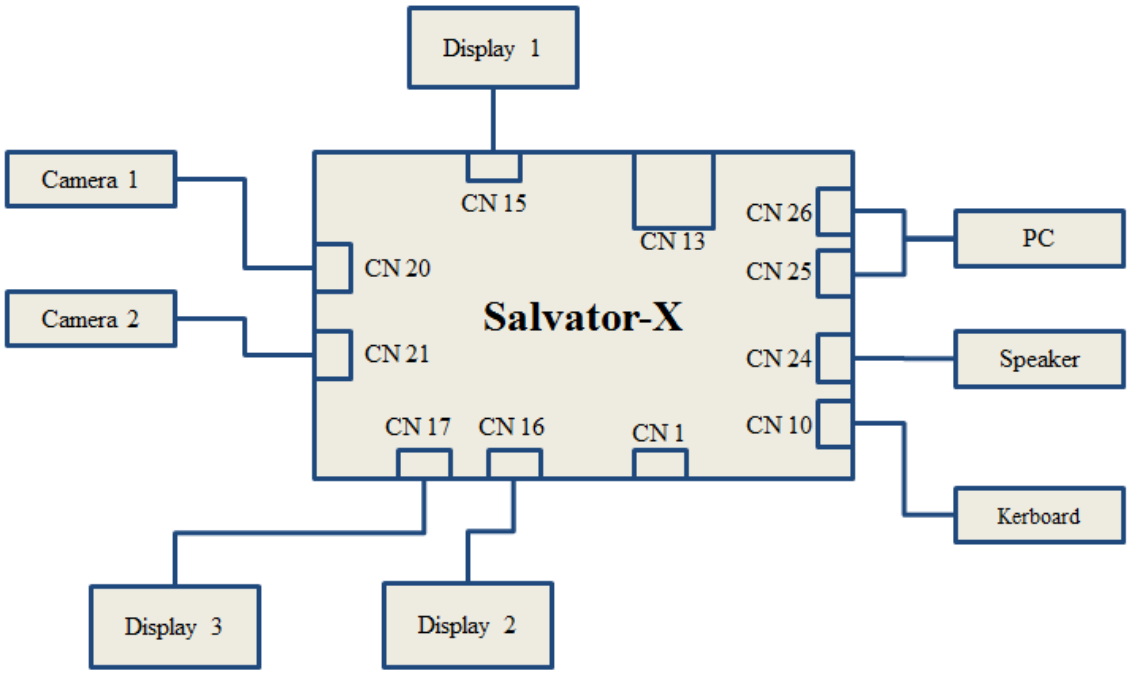


Figure 3-1: Virtualization PoC setup

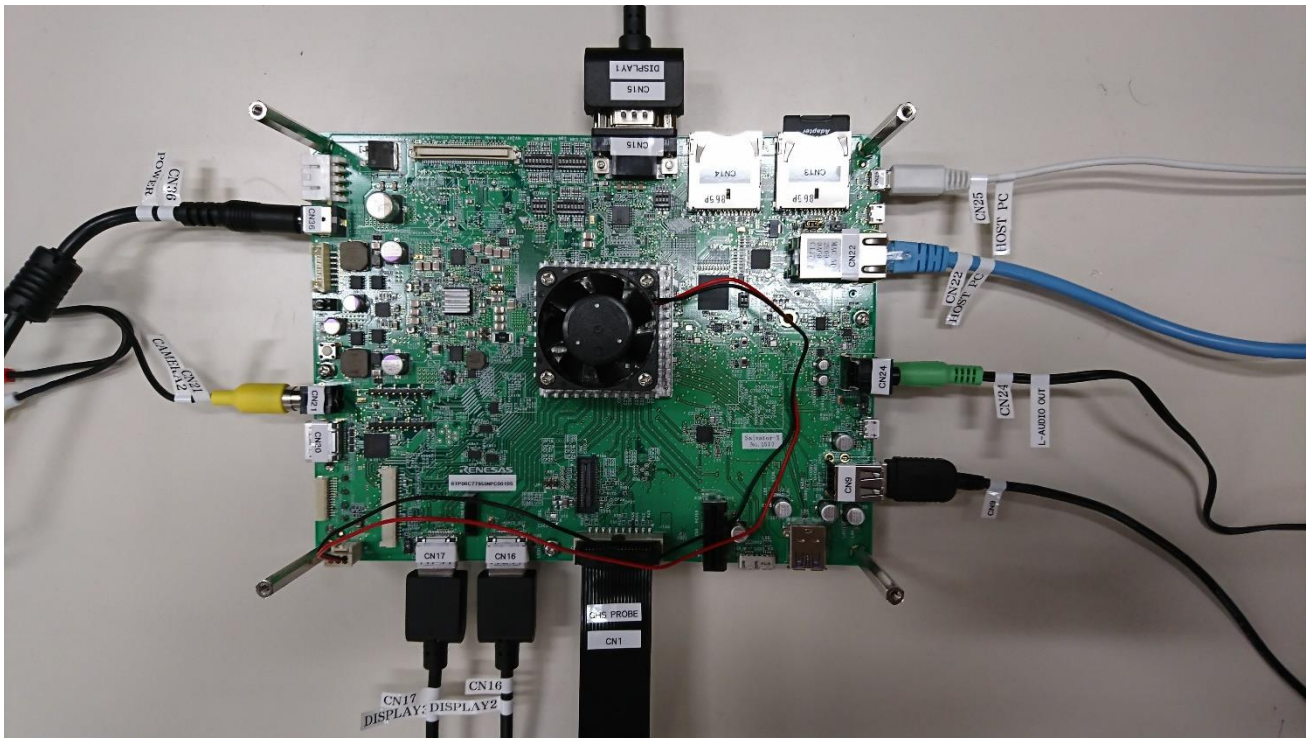


Figure 3-2: Virtualization PoC top view

Table 3-1: Connector List

Connector	Item	Description
CN 10	Keyboard	Insert a USB-Keyboard
CN 13	SD Card	Insert a SD card
CN 15	Display 1	Connect an Analog RGB Display (use for display Head-up display)
CN 16	Display 2	Connect a HDMI Display (use for display Center Information)
CN 17	Display 3	Connect a HDMI Display (use for display Instrument Cluster)
CN 21	Camera 2	Connect a NTSC Camera
CN 24	Speaker	Connect a stereo speaker to AudioOut
CN 25/CN26	PC	Connect a PC (use for terminal software on PC)

3.3. Hardware Components List

The following Table shows the virtualization Hardware components.

Table 3-2: Hardware Components List

Item	Quantity	Description
Salvator-X board	1	R-Car H3-SiP System Evaluation Board
Display	3	On-Lap 1303I, 13.3 inch, microHDMI support
Speaker	1	SANWA MM-SPL2N2
USB charger	1	SANWA 700-AC011-W
Power Supply TAP	1	SANWA TAP-TSH61N
SD card (*)	7	IO-DATA: SDMCH-W8G/A x5, IO-DATA: BMS-8G10RW x2
Camera	1	Gopro HERO3+ miniHDMI,miniUSB support
mini HDMI to HDMI Converter cable	1	Sony DLC-HEU10A
miniUSB to NTSC Converter cable	1	Gopro ACMPS-301

(*) This product includes components of the PowerVR SDK from Imagination Technologies Limited.

3.4. The Salvator-X board features

The following table shows the Salvator-X board features.

Table 3-3: Salvator-X board features

Item	Description
SoC	● R-CarH3-SiPand its also includes LPDDR4-3200 memories and RPC flash memory
Display Interfaces	● 2 channels HDMI output connector for HDMI0 and HDMI1 ● LVDS output connector for LVDS ● Analog RGB output connector for DU
Video Input Interfaces	● HDMI input connector for CSI0 ● CVBS input connector for CS1
Audio Interfaces	● Audio output connector for SSI0 ● Microphone input connector for SSI1
Storage Interfaces	● USB2.0 type micro AB receptacle for USB0 ● 2 channels USB2.0 type A receptacle for USB1 and USB2 ● USB3.0 type A receptacle for USB0 ● USB3.0 type micro B receptacle for USB1 ● Serial ATA connector for SATA ● 2 channels SD Card Slot for SDHI0 and SDHI3 ● eMMC memory for MMCIF0
Network Interfaces	● PCIE x1 connector for PCIEC0 ● BT/Wi-Fi connector for PCIEC1 ● GbE connector for EtherAVB
Peripheral Interfaces	● 2 channels ‘Debug Serial’ connector for SCIF2_A and SCIF1/HSCIF1_A
Debugger Interfaces	● 20-pin JTAG connector
Peripheral connectors	● External memory connector for LBSC ● Four EXIO connectors for LBSC, SSI, and various modules
Power Supply	● DC12.0V input
Operating temperature	● +25 degrees C at ambient temperature

4. Virtualization PoC Implementation. Software

This chapter describes software required for virtualization PoC setup.
The following figure shows the software configurations.

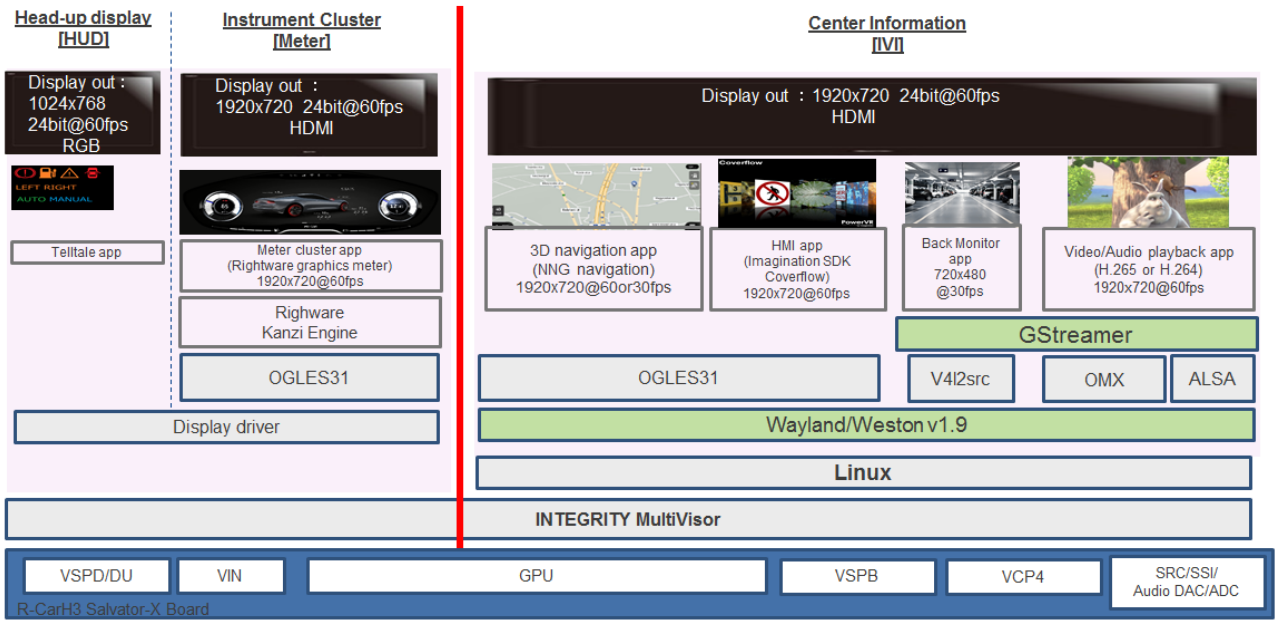


Figure 4-1: Image of Software configurations

Virtualization PoC doesn't have Back Monitor on Head-up display. Evaluation is performed according to the specification on each OS.

The following Table shows the software lists.

Table 4-1: Software lists

Category	Item	Model name	Version
OS	Yocto	-	2.7.0
	INTEGRITY	-	T9.0
Driver / Middleware	R-eBSP RENESAS RTP0RC7795SIPB0010S Basic Driver Package for INTEGRITY(R) Multivisor(TM) source package	RTP0RC7795SIPB0010S-ITGBDPM	0.2.10
	R-eBSP RENESAS RTP0RC7795SIPB0010S Basic Driver Package for INTEGRITY(R) Multivisor(TM) source package	Linux patch for RENESAS RTP0RC7795SIPB0010S Basic Driver Package for INTEGRITY(R) Multivisor(TM)	0.2.10
	Graphics R8A7795 GX6650 OpenGL ES 3.1 Library for INTEGRITY(R) Multivisor	RTM0RC7795GLTG0011SGH2C	1.1.4a
	Graphics R8A7795 GX6650 OpenGL ES 3.1 Library for Linux on INTEGRITY(R) Multivisor	RTM0RC7795GLTG0011SL40C	1.1.4a
	OMX Media Component Common Library for Linux	RTM0AC0000XCMCTL30SL40C	3.0.2
	OMX Media Component Video Decoder Common Library for Linux	RTM0AC0000XVCMND30SL40C	3.0.2
	OMX Media Component H.264 Decoder Library for Linux	RTM0AC0000XV264D30SL40C	3.0.2
	UVCS Driver for Linux	RCG3VUDRL4001ZDO	3.0.2
	OMX Media Component Audio Common Library for Linux	RTM0AC0000XACMND30SL40C	3.0.2
	OMX Media Component for AAC-LC Decoder Library for Linux	RTM0AC0000XAAACD30SL40C	3.0.2
	AAC-LC 2ch Decoder Middleware Library for Linux	RTM0AC0000ADAACMZ1SL40C	3.0.2
Application	3D navigation	-	9.35.0.0 Aug 31 2016 Demo (20160915)
	HMI	-	3.5
	Video/Audio playback with media player	-	1.4.5
	Meter cluster	-	20161222
	Telltale	-	0.2.10a
	Back monitor for HUD	-	0.2.10a

4.1. Center Information

This section describes application software for Center Information.

4.1.1. 3D navigation

The following figure shows the image of 3D navigation.
Note: NNG navigation shall be installed only when we evaluate it. The deliverables shall not include it.

Figure 4-2: Image of 3D navigation

The following table shows the 3D navigation features.

Table 4-2: 3D navigation features

Item	Description
Application	● NNG navigation ● OpenGL ES 3.1 application
Resolution	● 1920x720

4.1.2. HMI

The following figure shows the image of HMI.

Figure 4-3: Image of HMI

The following table shows the HMI features.

Table 4-3: HMI features

Item	Description
Application	● Imagination SDK Coverflow ● OpenGL ES 3.1 application
Resolution	● 1920x720

4.1.3. Back monitor

The following figure shows the image of Back monitor.

Figure 4-4: Image of Back monitor

The following table shows the Back monitor features.

Table 4-4: Back monitor features

Item	Description
Application	● NTSC Camera application on Gstreamer
Frame per second	● 30
Resolution	● 720x480

4.1.4. Video/Audio playback with media player

The following figure shows the image of Video/Audio playback with media player.

Figure 4-5: Image of Video/Audio playback with media player

The following table shows the Video/Audio playback with media player features.

Table 4-5: Video/Audio playback with media player features

Item	Description
Application	● GStreamer
Video Codec	● H.264
Frame per second	● Maximum 60 fps
Resolution	● 1920 x720
Media file location	● SD storage

4.2. Instrument Cluster

This section describes application software for Instrument Cluster.

4.2.1. Meter Cluster

The following figure shows the image of Meter Cluster.

Figure 4-6: Image of Meter Cluster

The following table shows the Meter Cluster features.

Table 4-6: Meter Cluster features

Item	Description
Application	● Sakura ● Rightware graphics meter ● OpenGL ES 3.1 application running on Rightware Kanzi
Frame per second	● 60
Resolution	● 1920x720

4.3. Head-up display

This section describes application software for Head-up display.

4.3.1. Telltale

The following figure shows the image of Telltale.

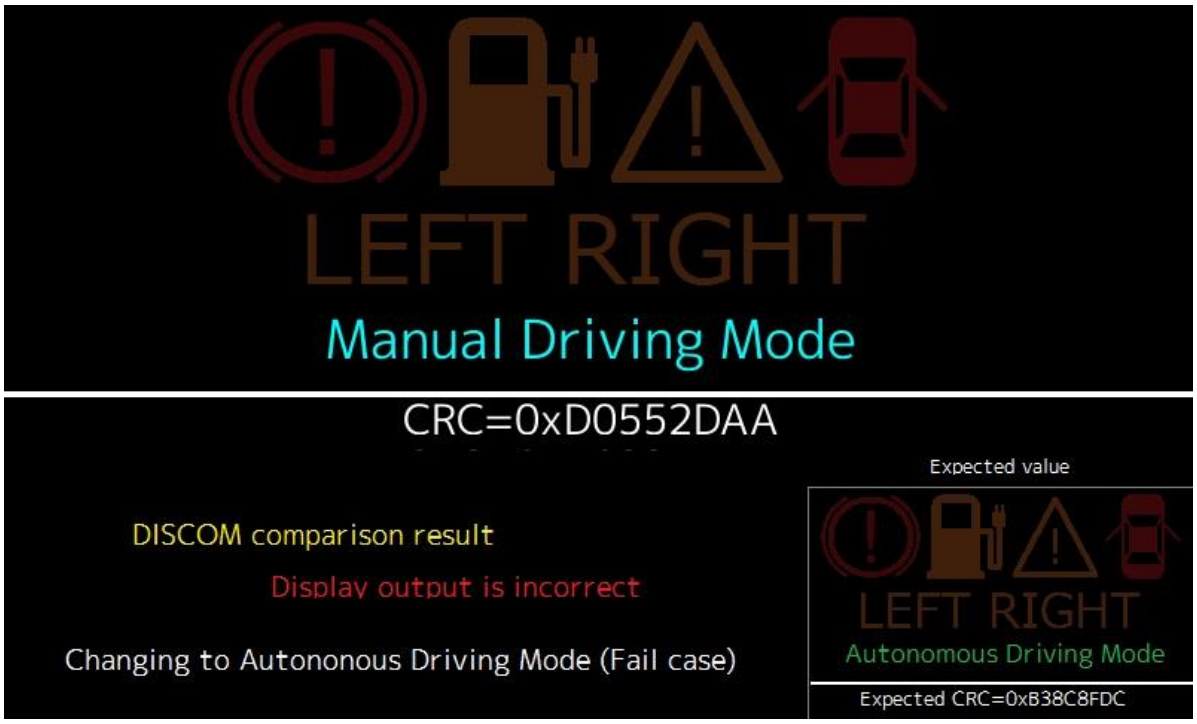


Figure 4-7: Image of Telltale

The following table shows the Telltale features.

Table 4-7: Telltale features

Item	Description
Application	● DISCOM ● Telltale, the simulated car status indicator on HUD or meter cluster. ● This demonstrates the operation of DISCOM module, to check whether the display contents are correct or not. ● All text and image drawings are performed by software.
Frame per second	● 60fps
Resolution	● 800x480

4.3.2. Back monitor

The following figure shows the image of Back monitor.

Figure 4-8: Image of Back monitor

The following table shows the Back monitor features.

Table 4-8: Back monitor features

Item	Description
Application	● HDMI Camera Application made by Renesas
Frame per second	● 60
Resolution	● 1280x720

5. Measurement

Executing the performance test, there are four test environments below.

If no special mentions in performance test procedure, set up a configuration that matches each type of measurement cases. Then start performance test.

INTEGRITY kernel is included debug library

Note :

Type1,3,4 are using the debug library in INTEGRITY kernel in order to measure performance from INTEGRITY side. It may produce some overhead came from debug functionality.

The underlined text in each consideration includes our guess or assumption.

This evaluation uses 4 types of Test environment. It is use for the following reasons.

Type 1 use for measuring PoC performance.

When comparing the performance of Linux, compare virtualized Linux of Type 4 and native Linux of Type 2.

Type 3 use for measuring only INTEGRITY.

- Virtualization PoC (Type1)

It is to build up the different character OS/application environments into the some domains, typically safety (INTEGRITY) and open (Linux).

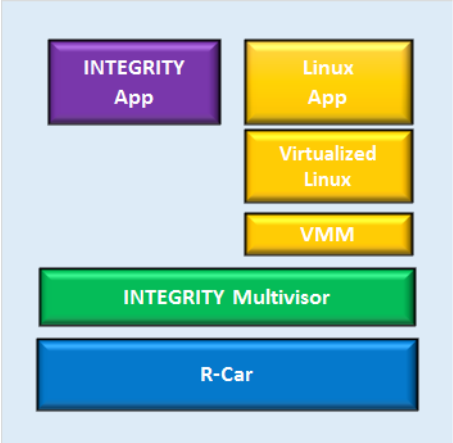


Figure 5-1: Image of Virtualization PoC(Type1)

- Native Linux (Type2)

It is to build up the only Linux OS on R-car board without INTEGRITY Multivisor.

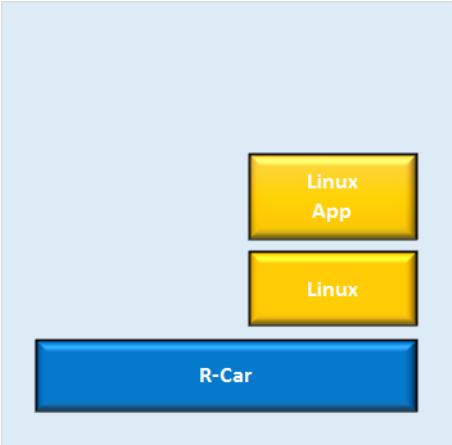


Figure 5-2: Image of Native Linux(Type2)

Native INTEGRITY (Type3)
It is based on build up the virtualization PoC, in addition stop the function of virtualized Linux part.

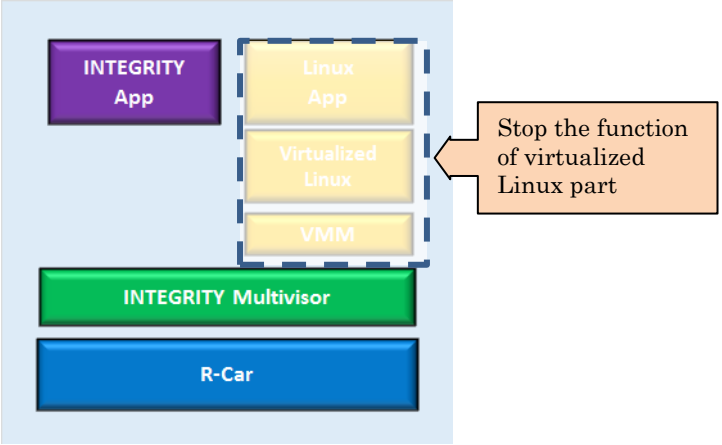


Figure 5-3: Image of Native INTEGRITY(Type3)

● Virtualized Linux(Type4)

It is based on build up the virtualization PoC, in addition stop the function of INTEGRITY App part.

INTEGRITY including drivers and INTEGRITY Multivisor are running, because INTEGRITY and INTEGRITY Multivisor are actually same modules.

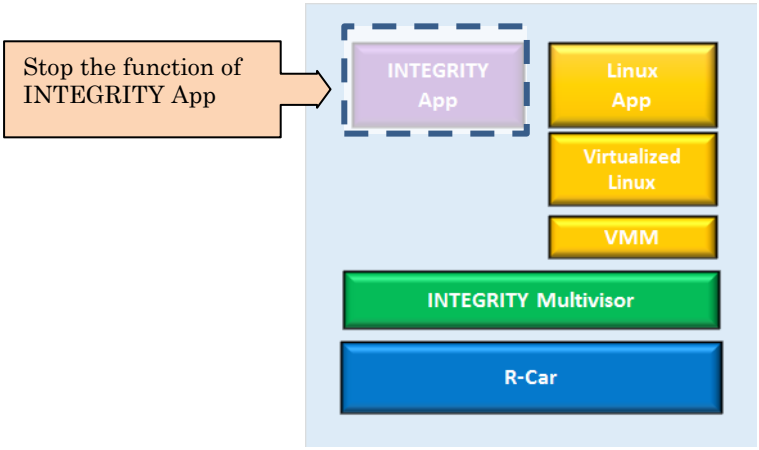


Figure 5-4: Image of virtualized Linux(Type4)

Attention:
When you test following cases except section 5.4,
please write U-Boot to HyperFlash in Appendix A and download monolith binary in Appendix B.
Appendix A and Appendix B require MULTI and GreenHills Probe.

The following list shows the version of the measurement tool used for measurement.

Table 5-1: measurement tool lists

Measurement Tool	Version
UnixBench	5.1.3
lmbench	3.0 alpha-9
Busmoni(INTEGRITY)	RENESAS original
Busmoni(Linux)	1.2
Cyclictest	0.9.2
gsize	T9.0 (Including INTEGRITY)
IPMMUDemo	T9.0 (Including INTEGRITY)
random_write	RENESAS original
alloc_dealloc	RENESAS original

The following list shows the build configurations used for measurement.

Table 5-2: Build configurations

Configurations	Setting	Note
INTEGRITY kernel configurations	-kernel	Use the normal kernel
	-ldebug	Use the debug library to use some debugging functions

5.1. CPU Load

5.1.1. Total CPU usage on Linux

(1) Description

Measure the CPU usage of Center Information application on native Linux using Unixbench.

(2) Precondition

- Measure on native Linux(Type2)
- Use Unixbench on terminal software.
- Use “cat /proc/stat” command in order to measure the average of CPU load.

(3) How to measure

- Only “cat /proc/stat” command

1. Login to Linux.

```
salvator-x login: root
```

2. Run “cat /proc/stat” command as the “measure start” of CPU status.

```
root@salvator-x:~# cat /proc/stat
```

After finishing a command, you will see the log like below.

```
cpu  101077 0 25728 270652 4690 0 1982 0 0 0
cpu0 19369 0 7014 67547 386 0 1981 0 0 0
cpu1 22733 0 6785 71907 1136 0 0 0 0 0
cpu2 27188 0 6164 67437 1714 0 0 0 0 0
cpu3 31786 0 5763 63760 1453 0 0 0 0 0
```

These items mean “usr”, ”nice”, “sys”, “idle” in turn.

3. Waiting for 10 minutes.
4. Run “cat /proc/stat” command as the “measure end” of CPU status.
5. Calculate the CPU load with following formula.

(1) CPU usage = 100 – “idle ratio”

“idle ratio” = “idle” of “measure end” - “idle” of “measure start”/ Total CPU count

Total CPU count = Sum of (“measure end”-“measure start”)

- Unixbench and “cat /proc/stat” command

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/UnixBench
```

3. Run the following command to burden the software.

```
root@salvator-x:~/tools/UnixBench# ./Run -c 4 -i 100 syscall&
```

4. Run “cat /proc/stat” command as the “measure start” of CPU status.

```
root@salvator-x:~# cat /proc/stat
```

5. Waiting for 10 minutes.
6. Run “cat /proc/stat” command as the “measure end” of CPU status.
7. Calculate the CPU load with the above formula.

(4) Result

Table 5-3: Only “cat /proc/stat” command Result(%)

Test environment	CPU Usage.
Native Linux (Type2)	31.6

Table 5-4: Unixbench and “cat /proc/stat” command Result(%)

Test environment	CPU Usage.
Native Linux (Type2)	84.6

- Only “cat /proc/stat” command

Result of “measure start”

```
root@salvator-x:~# cat /proc/stat
cpu  101077 0 25728 270652 4690 0 1982 0 0 0
cpu0 19369 0 7014 67547 386 0 1981 0 0 0
cpu1 22733 0 6785 71907 1136 0 0 0 0 0
cpu2 27188 0 6164 67437 1714 0 0 0 0 0
cpu3 31786 0 5763 63760 1453 0 0 0 0 0
```

Result of “measure end”

```
root@salvator-x:~# cat /proc/stat
cpu  163044 0 40878 437813 4782 0 1984 0 0 0
cpu0 31650 0 11245 110965 406 0 1983 0 0 0
cpu1 36322 0 10726 115796 1152 0 0 0 0 0
cpu2 43863 0 9780 108582 1758 0 0 0 0 0
cpu3 51208 0 9126 102468 1465 0 0 0 0 0
```

Calculation:

CPU load = 100 – (167161/244372) = 31.6 %

	usr	nice	sys	idle	other	
measure end	163044	0	40878	437813	6766	
measure start	101077	0	25728	270652	6672	
difference	61967	0	15150	167161	92	244372(Sum)

- Unixbench and “cat /proc/stat” command

Result of “measure start”

```
root@salvator-x:~/tools/UnixBench# cat /proc/stat
cpu  32059 0 41454 43550 3951 0 1239 0 0 0
cpu0 7268 0 10083 9880 398 0 1239 0 0 0
cpu1 7751 0 10485 11634 1243 0 0 0 0 0
cpu2 8369 0 10490 11281 994 0 0 0 0 0
cpu3 8670 0 10394 10754 1315 0 0 0 0 0
```

Result of “measure end”

```
root@salvator-x:~/tools/UnixBench# cat /proc/stat
cpu  103536 0 169657 80465 4820 0 2852 0 0 0
cpu0 24481 0 41070 19083 596 0 2852 0 0 0
cpu1 25201 0 42952 21432 1460 0 0 0 0 0
cpu2 26168 0 43118 20586 1222 0 0 0 0 0
cpu3 27686 0 42516 19363 1541 0 0 0 0 0
```

Calculation:

CPU load = 100 – (36915/239077) = 84.6 %

	usr	nice	sys	idle	other	
measure end	103536	0	169657	80465	7672	
measure start	32059	0	41454	43550	5190	
difference	71477	0	128203	36915	2482	239077(Sum)

(5) Consideration

This result(%) is the percentage of all 4 CPU resources, so if only the 1 core is fully occupied and other cores are idle, the result becomes 25%.

From the table 5-3, this PoC demo program consumes about the power of one whole CPU.

The table 5-4 increases approximately 53% CPU load in addition to table 5-3.

5.1.2. Total CPU usage of Hypervisor(Multivisor)

(1) Description

Measure the total CPU utilization of Multivisor Task (VM task of guest OS) when using the application of Center Information application on virtualized Linux using Multi Debugger.

(2) Precondition

- Measure on virtualized Linux (Type4)
- Use a tool including in Multi Debugger.

(3) How to measure

1. Select [Target] – [Connect] from Menu bar
2. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
3. Select “Run mode target”
4. Run the following command on “Trg” tab.

```
INDRT2>ct
```

5. Waiting for 10 minutes.
6. Run the following command on “Trg” tab.

```
INDRT2>lt
```

After finishing a command, you will see the log like below.

MultivisorTask0-3 means CPU usage of Linux on virtualization.

```
multivisor_vmm                          0x000000000022a000/0x0000000000c3b000
   0xffffffa6c0073000 pending          127 0x0000000000000790/0x0000000000002000   0.00% Initial
   0xffffffa00a876000 pending          254 0x0000000000000280/0x00000000000000dd0   0.00% OSAAgent
   0xffffffa6feedd000 pending          127 0x00000000000004b0/0x00000000000000dd0   0.00% AsyncPollTask
   0xffffffa6feea0000 pending          127 0x00000000000011f0/0x00000000000002dd0   0.00% GipcStdio_StdinTask
   0xffffffa6fee9a000 pending          127 0x0000000000001210/0x00000000000002dd0   0.00% GipcStdio_StdoutTask
   0xffffffa6fee96000 pending          200 0x0000000000000440/0x00000000000001dd0  36.54% MultivisorTask0
   0xffffffa6fee92000 pending          200 0x00000000000003d0/0x00000000000001dd0  31.61% MultivisorTask1
  *0xffffffa6fee8d000 running          200 0x00000000000003d0/0x00000000000001dd0  36.74% MultivisorTask2
   0xffffffa6fee89000 pending          200 0x0000000000001410/0x00000000000001dd0  40.84% MultivisorTask3
```

(4) Result

This result is average of MultivisorTask0-3. MultivisorTask0-3 means CPU usage of Linux on virtualization.

Table 5-5: Result of Type4(%)

Test environment	CPU Usage.
Virtualized Linux (Type4)	36.43

```
                                    Memory:  Used/Size or
 Task Id             Status          Pri Stack:HiWater/Size    Time    Task Name
 ------------------- --------------- --- ------------------------------------ ------- ---------
 Type4_kernel                            0x000000000012c000/0x00000000170aa000
   0xffffffa010010000 exited          127 0x0000000000000630/0x0000000000008000   0.00% Initial
  *0xffffffa010012000 running           0 0x0000000000000028/0x0000000000000400  62.00% Idle0
   0xffffffa010014000 running           0 0x0000000000000028/0x0000000000000400  66.39% Idle1
   0xffffffa010016000 running           0 0x0000000000000028/0x0000000000000400  61.48% Idle2
   0xffffffa010018000 running           0 0x0000000000000028/0x0000000000000400  57.57% Idle3
   0xffffffa6c00c0000 pending         254 0x0000000000000060/0x0000000000001000   0.00% RunModePartnerTask
   0xffffffa6c00c2000 pending         254 0x0000000000001240/0x0000000000004800   0.00% ResourceManager
   0xffffffa6c00c6000 pending         200 0x00000000000000f8/0x0000000000001000   0.00% rcar-avb-0
   0xffffffa6c00c8000 pending         253 0x00000000000006e8/0x0000000000005000   0.00% DriverDebugControl
   0xffffffa6c00d3000 pending         255 0x0000000000000360/0x0000000000002000   0.00% GipcTarget_Dispatch
  *0xffffffa6c00dc000 running         254 0x00000000000001e8/0x0000000000002000   0.00% IDB_Receiver
   0xffffffa6c00e0000 pending         254 0x00000000000001b0/0x0000000000002000   0.00% IDB_Sender
   0xffffffa6c00e2000 pending         127 0x0000000000000358/0x0000000000008000   0.00%
FrameBufferManagerTask
   0xffffffa6c00e6000 pending         127 0x00000000000002e0/0x0000000000008000   0.12% VINManagerTask
   0xffffffa6c00e8000 pending         150 0x0000000000000488/0x0000000000002000   0.00% SDIOCardIOTask
   0xffffffa6c00ea000 pending         150 0x00000000000004e0/0x0000000000002000   0.00% SDIOCardIOTask1
   0xffffffa6c00ed000 pending         254 0x00000000000001d0/0x0000000000000dd0   0.00% OSAAgent
   0xffffffa6c00f1000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c00f5000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c00f8000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c00fc000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c00ff000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0102000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0106000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0109000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c010c000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c010f000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0112000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0116000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0119000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c011c000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c011f000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0122000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0126000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0129000 pending          11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c012b000 pending         128 0x0000000000000170/0x0000000000003cb8   0.00% PosixServer
                                          0x0000000000006000/0x0000000000006000
```

```
FBServer                                0x0000000000000000/0x000000000103d000
   0xfffffffa6c0077000 pending          127 0x0000000000001560/0x0000000000006000   0.00% Initial
   0xfffffffa00c936000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 pvrserver_as0                              0x0000000001fb9000/0x0000000009dfd000
   0xfffffffa6c0076000 pending          200 0x0000000000000f00/0x0000000000006000   0.00% Initial
   0xfffffffa00c8f6000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa6ee648000 pending          200 0x0000000000000310/0x000000000000fdd0   0.00% pvr_defer_free
   0xfffffffa6ee636000 pending          200 0x00000000000005d0/0x000000000000fdd0   0.00% pvr_device_wdg
   0xfffffffa6ec75e000 pending          200 0x00000000000002a0/0x0000000000000dd0   1.25% 3DGIntrTask
   0xfffffffa6ec750000 pending          200 0x0000000000000060/0x0000000000000dd0   0.00% GRAPHICS_MISR
   0xfffffffa6ec74d000 pending          200 0x0000000000000230/0x0000000000000dd0   1.86% GRAPHICS_MISR
   0xfffffffa6ec746000 pending          128 0x0000000000000050/0x0000000000003dd0   0.00% CheckSSHRebootThread
 multivisor_net_server                      0x0000000000003000/0x0000000000040000
   0xfffffffa6c0075000 pending          127 0x0000000000000790/0x0000000000002000   0.00% Initial
   0xfffffffa00a8f6000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 multivisor_loader                          0x0000000000000000/0x000000000c823000
   0xfffffffa6c0074000 exited           127 0x00000000000008f0/0x0000000000002000   0.00% Initial
   0xfffffffa00a8b3000 pending          254 0x0000000000000310/0x0000000000001dd0   0.00% LoaderTask
   0xfffffffa00a8a9000 pending          254 0x0000000000000220/0x0000000000001dd0   0.00% MULTILoadAgent
   0xfffffffa00a8a6000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00a8a3000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00a8a0000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00a89c000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 multivisor_vmm                             0x0000000000022a000/0x0000000000c3b000
   0xfffffffa6c0073000 pending          127 0x0000000000000790/0x0000000000002000   0.00% Initial
   0xfffffffa00a876000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa6feedd000 pending          127 0x00000000000004b0/0x0000000000000dd0   0.00% AsyncPollTask
   0xfffffffa6feea0000 pending          127 0x00000000000011f0/0x0000000000002dd0   0.00% GipcStdio_StdinTask
   0xfffffffa6fee9a000 pending          127 0x0000000000001210/0x0000000000002dd0   0.00% GipcStdio_StdoutTask
   0xfffffffa6fee96000 pending          200 0x0000000000000440/0x0000000000001dd0  36.54% MultivisorTask0
   0xfffffffa6fee92000 pending          200 0x00000000000003d0/0x0000000000001dd0  31.61% MultivisorTask1
  *0xfffffffa6fee8d000 running          200 0x00000000000003d0/0x0000000000001dd0  36.74% MultivisorTask2
   0xfffffffa6fee89000 pending          200 0x0000000000001410/0x0000000000001dd0  40.84% MultivisorTask3
 ip46router_devtree_module                  0x000000000003b000/0x0000000004af000
   0xfffffffa6c0072000 exited           127 0x0000000000000af0/0x0000000000003000   0.00% Initial
   0xfffffffa00a836000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa00a831000 pending          200 0x00000000000000c0/0x0000000000000dd0   0.00% rcar-avb-0
  *0xfffffffa00a828000 running          253 0x00000000000007a0/0x0000000000004dd0   0.06% DriverDebugService
   0xfffffffa6ff353000 pending          200 0x00000000000005a0/0x0000000000005dd0   0.24% InetServer
   0xfffffffa6ff341000 pending          200 0x00000000000001a0/0x0000000000003dd0   0.00% FibArpRefreshTask0
   0xfffffffa6ff332000 pending          127 0x0000000000000080/0x0000000000000dd0   0.00%
PingWatchdog_ResetTask
 ivfsserver_devtree_module                  0x00000000000ca000/0x000000000073d000
   0xfffffffa6c0071000 exited           127 0x0000000000000880/0x0000000000002000   0.00% Initial
   0xfffffffa00a736000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa6ff954000 pending          152 0x00000000000000c0/0x0000000000003dd0   0.00% HealthMonitor
   0xfffffffa6ff949000 pending          150 0x0000000000000a40/0x0000000000008dd0   0.00% FileServer
   0xfffffffa6ff942000 pending          149 0x00000000000005d0/0x0000000000003dd0   0.03% NFSTimerTask
   0xfffffffa6ff8c1000 pending          152 0x0000000000000180/0x0000000000004dd0   0.00% IOTask
   0xfffffffa6ff8b4000 pending          150 0x0000000000000910/0x0000000000008dd0   0.00% IOAssistant0
   0xfffffffa6ff8ab000 pending          150 0x0000000000000690/0x0000000000006dd0   0.00% Syncer
   0xfffffffa6ff8a4000 halted           151 0x0000000000000090/0x0000000000003dd0   0.00% Unmounter
   0xfffffffa6ff88d000 pending          150 0x00000000000004c0/0x0000000000008dd0   0.00% IOAssistant1
 devtree_generic_server_module              0x00000000002b6000/0x0000000001501000
   0xfffffffa6c0070000 pending          127 0x0000000000000860/0x0000000000002000   0.00% Initial
   0xfffffffa6fbf56000 pending          220 0x00000000000003f0/0x0000000000001dd0   0.00% fb-map-server
```

(5) Consideration

The measured result is 36.4 % on this virtualized Linux environment. As the corresponding result on the native Linux is 31.6%, the CPU load on the virtualized Linux is 4.8% higher. From the result of 5.1.4, the virtualization overhead of the Linux itself is relatively small, so the possible factor of the CPU load difference is the GPU driver which require a large number of interrupt and event processing.

5.1.3. Total CPU usage on INTEGRITY

(1) Description

Measure the CPU usage of Instrument Cluster / Head-up display application on native INTEGRITY.

(2) Precondition

- Measure on native INTEGRITY (Type3)
- Use a tool including in Multi Debugger.

(3) How to measure

1. Refer to 5.1.2.

   After finishing a command in 6 procedures, see the log like below.

   [Instrument Cluster]

   See at the left value of "Initial", “PosixServer” and “name_too_long” in section of “sakura”.

```
Sakura                                  0x0000000000df000/0x0000000004000000
   0xfffffffa6c00d9000 pending          127 0x00000000000187b0/0x0000000000200000  14.37% Initial
   0xfffffffa00fab3000 pending          128 0x0000000000000168/0x0000000000003cb8   0.00% PosixServer
   0xfffffffa00faae000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa00fa4b000 pending          127 0x0000000000000b38/0x0000000000003cb8   0.00% name_too_long
```

   [Head-up display application]

   See at the left value of "Initial" in section of “DISCOM_sample_virt”.

```
DISCOM_sample_virt                      0x0000000000005000/0x0000000000080000
   0xfffffffa6c00e0000 pending          127 0x0000000000000af0/0x0000000000008000   0.64% Initial
   0xfffffffa00fe76000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
```

   This log (%) is based on one CPU base. If the all CPUs are fully loaded, the value will be 400%.

(4) Result

Result is based on full CPU base. If the all CPUs are fully loaded, the value will be 100%.

Table 5-6: Result(%)

Test environment	Total CPU Usage.
Native INTEGRITY (Type3)	7.39%

Test environment	Application	CPU Usage.
Native INTEGRITY (Type3)	**DISCOM_sample_virt**	0.16
	Sakura	3.59
	Other Task	3.63

```
                                        Memory:  Used/Size or
 Task Id              Status           Pri Stack:HiWater/Size    Time    Task Name
 -------------------- ---------------- --- ------------------------------------ ------- ---------
 Type3_kernel                              0x0000000000017f000/0x000000000ccaa000
   0xffffffa6c0010000 exited           127 0x0000000000000630/0x0000000000008000   0.00% Initial
   0xffffffa6c0012000 running            0 0x0000000000000028/0x0000000000000400  87.58% Idle0
   0xffffffa6c0014000 running            0 0x0000000000000028/0x0000000000000400  93.19% Idle1
   0xffffffa6c0016000 running            0 0x0000000000000028/0x0000000000000400  94.48% Idle2
  *0xffffffa6c0018000 running            0 0x0000000000000028/0x0000000000000400  92.65% Idle3
   0xffffffa6c012f000 pending          254 0x0000000000000060/0x0000000000001000   0.00% RunModePartnerTask
   0xffffffa6c0131000 pending          254 0x0000000000001140/0x0000000000004800   1.19% ResourceManager
   0xffffffa6c0135000 pending          200 0x00000000000000f8/0x0000000000001000   0.00% rcar-avb-0
   0xffffffa6c0137000 pending          253 0x00000000000006e8/0x0000000000005000   0.00% DriverDebugControl
   0xffffffa6c0142000 pending          255 0x0000000000000360/0x0000000000002000   0.00% GipcTarget_Dispatch
  *0xffffffa6c014d000 running          254 0x00000000000001e8/0x0000000000002000   0.01% IDB_Receiver
   0xffffffa6c014f000 pending          254 0x00000000000001b0/0x0000000000002000   0.00% IDB_Sender
   0xffffffa6c0151000 pending          127 0x0000000000000358/0x0000000000008000   0.00%
FrameBufferManagerTask
   0xffffffa6c0155000 pending          127 0x00000000000002e8/0x0000000000008000   0.02% VINManagerTask
   0xffffffa6c0157000 pending          150 0x0000000000000480/0x0000000000002000   0.00% SDIOCardIOTask
   0xffffffa6c0159000 pending          150 0x00000000000004e8/0x0000000000002000   0.00% SDIOCardIOTask1
   0xffffffa6c015c000 pending          254 0x00000000000001d0/0x0000000000000dd0   0.00% OSAAgent
   0xffffffa6c0160000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0163000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0166000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c016a000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c016d000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0170000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0173000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0176000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0179000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c017c000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c017f000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0182000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0185000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0188000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c018c000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c018f000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0192000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0195000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xffffffa6c0198000 pending          128 0x0000000000000178/0x0000000000003cb8   0.00% PosixServer
 INT_Logo_sample_virt                      0x0000000000003000/0x0000000000040000
   0xffffffa6c00e3000 pending          127 0x0000000000000880/0x0000000000008000   0.00% Initial
   0xffffffa011ef6000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 FBServer                                  0x0000000000000000/0x000000000103d000
   0xffffffa6c00e2000 pending          127 0x00000000000017d0/0x0000000000006000   0.39% Initial
   0xffffffa011eb6000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 pvrserver_as0                             0x00000000025b5000/0x0000000009dfd000
   0xffffffa6c00e1000 pending          200 0x0000000000000b80/0x0000000000006000   7.98% Initial
   0xffffffa011e76000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xffffffa6ee648000 pending          200 0x0000000000000310/0x000000000000fdd0   0.00% pvr_defer_free
   0xffffffa6ee636000 pending          200 0x0000000000000350/0x000000000000fdd0   0.00% pvr_device_wdg
   0xffffffa6ec75a000 pending          200 0x00000000000002a0/0x0000000000000dd0   1.04% 3DGIntrTask
   0xffffffa6ec74c000 pending          200 0x00000000000000e0/0x0000000000000dd0   0.12% GRAPHICS_MISR
   0xffffffa6ec749000 pending          200 0x0000000000000530/0x0000000000000dd0   1.99% GRAPHICS_MISR
   0xffffffa6ec742000 pending          128 0x0000000000000050/0x0000000000003dd0   0.00% CheckSSHRebootThread
   0xffffffa6ec68c000 pending          200 0x0000000000000330/0x0000000000000dd0   0.26% DC_OS_Task
   0xffffffa6ec689000 pending          200 0x0000000000000120/0x0000000000000dd0   0.83% GRAPHICS_MISR
   0xffffffa6ec675000 pending          200 0x0000000000000570/0x0000000000000dd0   0.45% FBIntrTask
 DISCOM_sample_virt                        0x0000000000005000/0x0000000000080000
   0xffffffa6c00e0000 pending          127 0x00000000000000af0/0x0000000000008000   0.64% Initial
   0xffffffa00fe76000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
```

```
 multivisor_net_server                    0x0000000000003000/0x0000000000040000
   0xfffffffa6c00df000 pending          127 0x0000000000000790/0x0000000000002000   0.00% Initial
   0xfffffffa00fdf6000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 multivisor_loader                        0x0000000000000000/0x000000000c823000
   0xfffffffa6c00de000 exited           127 0x00000000000008f0/0x0000000000002000   0.00% Initial
   0xfffffffa00fdb3000 pending          254 0x0000000000000310/0x0000000000001dd0   0.00% LoaderTask
   0xfffffffa00fda9000 pending          254 0x0000000000000220/0x0000000000001dd0   0.00% MULTILoadAgent
   0xfffffffa00fda6000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00fda3000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00fda0000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00fd9c000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 multivisor_vmm                           0x0000000000003000/0x0000000000040000
   0xfffffffa6c00dd000 halted           127 0x0000000000000670/0x0000000000002000   0.00% Initial
   0xfffffffa00fd76000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 ip46router_devtree_module                0x0000000000041000/0x000000004af000
   0xfffffffa6c00dc000 exited           127 0x00000000000000af0/0x0000000000003000   0.00% Initial
   0xfffffffa00fd36000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa00fd31000 pending          200 0x00000000000000c0/0x0000000000000dd0   0.00% rcar-avb-0
  *0xfffffffa00fd28000 running          253 0x00000000000007a0/0x0000000000004dd0   0.15% DriverDebugService
  *0xfffffffa6ff353000 pending          200 0x00000000000005a0/0x0000000000005dd0   0.10% InetServer
   0xfffffffa6ff341000 pending          200 0x00000000000000f0/0x0000000000003dd0   0.00% FibArpRefreshTask0
   0xfffffffa6ff327000 pending          127 0x0000000000000080/0x0000000000000dd0   0.00%
PingWatchdog_ResetTask
 ivfsserver_devtree_module                0x00000000000ca000/0x000000000073d000
   0xfffffffa6c00db000 exited           127 0x0000000000000880/0x0000000000002000   0.00% Initial
   0xfffffffa00fc36000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa6ff954000 pending          152 0x00000000000000c0/0x0000000000003dd0   0.00% HealthMonitor
   0xfffffffa6ff949000 pending          150 0x0000000000000a40/0x0000000000008dd0   0.00% FileServer
   0xfffffffa6ff942000 pending          149 0x00000000000005d0/0x0000000000003dd0   0.00% NFSTimerTask
   0xfffffffa6ff8c1000 pending          152 0x0000000000000180/0x0000000000004dd0   0.00% IOTask
   0xfffffffa6ff8b4000 pending          150 0x0000000000000910/0x0000000000008dd0   0.00% IOAssistant0
   0xfffffffa6ff8ab000 pending          150 0x0000000000000690/0x0000000000006dd0   0.00% Syncer
   0xfffffffa6ff8a4000 halted           151 0x0000000000000090/0x0000000000003dd0   0.00% Unmounter
   0xfffffffa6ff899000 pending          150 0x0000000000000450/0x0000000000008dd0   0.00% IOAssistant1
 devtree_generic_server_module            0x00000000002b6000/0x0000000001501000
   0xfffffffa6c00da000 pending          127 0x0000000000000860/0x0000000000002000   0.00% Initial
   0xfffffffa6fbf56000 pending          220 0x00000000000003f0/0x0000000000001dd0   0.00% fb-map-server
 Sakura                                   0x00000000000df000/0x0000000004000000
   0xfffffffa6c00d9000 pending          127 0x00000000000187b0/0x0000000000200000  14.37% Initial
   0xfffffffa00fab3000 pending          128 0x0000000000000168/0x0000000000003cb8   0.00% PosixServer
   0xfffffffa00faae000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa00fa4b000 pending          127 0x0000000000000b38/0x0000000000003cb8   0.00% name_too_long
```

(5) Consideration

Telltale application (DISCOM_sample_virt) draws periodically (3 seconds/1 frame draw), so it may result 0 to 0.5 percent depends on the measurement timing.

Meter cluster application (Sakura) uses OpenGLES library and application, and it is designed to consume under 12.5% of CPU.

5.1.4. The overhead (CPU usage) compared virtualized Linux with native Linux

(1) Description

Compare the overhead (CPU usage) of using the application of Center Information on virtualized Linux and native Linux.

(2) Precondition

- Measure on virtualized Linux and native Linux (Type4 and Type2)
- Use Unixbench on terminal software.
- Compare the performance between virtualized Linux and native Linux.

(3) How to measure

- Type2
  1. measurement method of Unixbench refer to 5.1.1.
- Type4
  2. Login to Linux.

```
salvator-x login: root
```

  3. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/UnixBench
```

  4. Run the following command to burden the software.

```
root@salvator-x:~/tools/UnixBench# ./Run -c 4 -i 100 syscall
```

  5. Run “cat /proc/stat” command as the “measure start” of CPU status.

```
root@salvator-x:~# cat /proc/stat
```

- Compare
  1. Check the overhead as follows.

     (Virtualized Linux) - (Native Linux)

(4) Result

Table 5-7: Virtualized Linux (Type2) Result(%)

Test environment	CPU Usage.
Native Linux (Type2)	74.57

Table 5-8: Native Linux (Type4) Result(%)

Test environment	CPU Usage.
Virtualized Linux (Type4)	76.09

Table 5-9: Compare Result

Test environment	Value(%)	Overhead: (B –A)
(A) Native Linux (Type2)	74.57	1.52 %
(B) Virtualized Linux (Type4)	76.09	

- Type2 Result.

Result of "measure start"

```
root@salvator-x:~# cat /proc/stat
cpu  72 0 262 7836 481 0 5 0 0 0
cpu0 22 0 61 1867 199 0 5 0 0 0
cpu1 12 0 55 2015 90 0 0 0 0 0
cpu2 18 0 108 1915 104 0 0 0 0 0
cpu3 20 0 36 2037 87 0 0 0 0 0
```

Result of "measure end"

```
root@salvator-x:~/tools/UnixBench# cat /proc/stat
cpu  116184 0 285179 145004 1706 0 9 0 0 0
cpu0 28644 0 71690 36094 498 0 9 0 0 0
cpu1 29248 0 71079 36560 167 0 0 0 0 0
cpu2 29182 0 71203 36153 486 0 0 0 0 0
cpu3 29110 0 71205 36195 553 0 0 0 0 0
```

Calculation:

CPU load = 100 – (137168/532496) = 74.57 %

	usr	nice	sys	idle	other	
measure end	116184	0	285179	145004	1715	
measure start	72	0	262	7836	486	
Difference	116112	0	284917	137168	1229	539426(Sum)

- Type4 Result.

```
                                      Memory:  Used/Size or
 Task Id              Status           Pri Stack:HiWater/Size    Time    Task Name
 -------------------- ---------------- --- ------------------------------------ ------- ---------
 Type4_kernel                              0x000000000012c000/0x00000000170aa000
   0xfffffffa010010000 exited           127 0x0000000000000630/0x0000000000008000   0.00% Initial
   0xfffffffa010012000 running            0 0x0000000000000028/0x0000000000000400  23.61% Idle0
   0xfffffffa010014000 running            0 0x0000000000000028/0x0000000000000400  23.91% Idle1
  *0xfffffffa010016000 running            0 0x0000000000000028/0x0000000000000400  23.92% Idle2
  *0xfffffffa010018000 running            0 0x0000000000000028/0x0000000000000400  23.94% Idle3
   0xfffffffa6c00c0000 pending          254 0x0000000000000060/0x0000000000001000   0.00% RunModePartnerTask
   0xfffffffa6c00c2000 pending          254 0x0000000000001130/0x0000000000004800   0.00% ResourceManager
   0xfffffffa6c00c6000 pending          200 0x00000000000000f8/0x0000000000001000   0.00% rcar-avb-0
   0xfffffffa6c00c8000 pending          253 0x000000000000006e8/0x0000000000005000   0.00% DriverDebugControl
   0xfffffffa6c00d2000 pending          255 0x0000000000000360/0x0000000000002000   0.00% GipcTarget_Dispatch
  *0xfffffffa6c00de000 running          254 0x00000000000001e8/0x0000000000002000   0.00% IDB_Receiver
   0xfffffffa6c00e0000 pending          254 0x00000000000001b0/0x0000000000002000   0.00% IDB_Sender
   0xfffffffa6c00e2000 pending          127 0x0000000000000358/0x0000000000008000   0.00%
FrameBufferManagerTask
   0xfffffffa6c00e6000 pending          127 0x000000000000002e0/0x0000000000008000   0.00% VINManagerTask
   0xfffffffa6c00e8000 pending          150 0x0000000000000488/0x0000000000002000   0.00% SDIOCardIOTask
   0xfffffffa6c00ea000 pending          150 0x000000000000004e0/0x0000000000002000   0.00% SDIOCardIOTask1
   0xfffffffa6c00ed000 pending          254 0x00000000000001d0/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa6c00f1000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c00f5000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c00f8000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c00fc000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c00ff000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0102000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0106000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0109000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c010c000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c010f000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0112000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0116000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0119000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c011c000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c011f000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0122000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0126000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c0129000 pending           11 0x0000000000000090/0x0000000000001000   0.00%
VirtualDriverMediator
   0xfffffffa6c012b000 pending          128 0x0000000000000170/0x0000000000003cb8   0.00% PosixServer
                                            0x0000000000006000/0x0000000000006000
 FBServer                                   0x0000000000000000/0x000000000103d000
   0xfffffffa6c0077000 pending          127 0x0000000000001560/0x0000000000006000   0.00% Initial
   0xfffffffa00c936000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 pvrserver_as0                              0x0000000001fb9000/0x0000000009dfd000
   0xfffffffa6c0076000 pending          200 0x0000000000000f00/0x0000000000006000   0.00% Initial
   0xfffffffa00c8f6000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa6ee648000 pending          200 0x0000000000000310/0x000000000000fdd0   0.00% pvr_defer_free
   0xfffffffa6ee636000 pending          200 0x00000000000005d0/0x000000000000fdd0   0.00% pvr_device_wdg
   0xfffffffa6ec75e000 pending          200 0x00000000000002a0/0x0000000000000dd0   0.00% 3DGIntrTask
   0xfffffffa6ec750000 pending          200 0x0000000000000060/0x0000000000000dd0   0.00% GRAPHICS_MISR
   0xfffffffa6ec74d000 pending          200 0x0000000000000200/0x0000000000000dd0   0.00% GRAPHICS_MISR
   0xfffffffa6ec746000 pending          128 0x0000000000000050/0x0000000000003dd0   0.00% CheckSSHRebootThread
```

```
multivisor_net_server                    0x0000000000003000/0x0000000000040000
   0xfffffffa6c0075000 pending          127 0x0000000000000790/0x0000000000002000   0.00% Initial
   0xfffffffa00a8f6000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 multivisor_loader                          0x0000000000000000/0x000000000c823000
   0xfffffffa6c0074000 exited           127 0x0000000000000530/0x0000000000002000   0.00% Initial
   0xfffffffa00a8b3000 pending          254 0x0000000000000680/0x0000000000001dd0   0.00% LoaderTask
   0xfffffffa00a8a9000 pending          254 0x0000000000000220/0x0000000000001dd0   0.00% MULTILoadAgent
   0xfffffffa00a8a6000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00a8a3000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00a8a0000 halted           254 0x0000000000000000/0x0000000000000dd0   0.00% LoaderHelperTask
   0xfffffffa00a89c000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
 multivisor_vmm                             0x000000000022a000/0x0000000000c3b000
   0xfffffffa6c0073000 pending          127 0x0000000000000790/0x0000000000002000   0.00% Initial
   0xfffffffa00a876000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa6feedd000 pending          127 0x00000000000004b0/0x0000000000000dd0   0.00% AsyncPollTask
   0xfffffffa6feea0000 pending          127 0x00000000000011f0/0x0000000000002dd0   0.00% GipcStdio_StdinTask
   0xfffffffa6fee9a000 pending          127 0x0000000000001210/0x0000000000002dd0   0.00% GipcStdio_StdoutTask
   0xfffffffa6fee96000 pending          200 0x00000000000003d0/0x0000000000001dd0  76.32% MultivisorTask0
   0xfffffffa6fee92000 pending          200 0x0000000000001410/0x0000000000001dd0  76.02% MultivisorTask1
   0xfffffffa6fee8d000 pending          200 0x00000000000003d0/0x0000000000001dd0  76.03% MultivisorTask2
   0xfffffffa6fee89000 pending          200 0x00000000000003d0/0x0000000000001dd0  76.02% MultivisorTask3
 ip46router_devtree_module                  0x000000000003b000/0x000000004af000
   0xfffffffa6c0072000 exited           127 0x0000000000000690/0x0000000000003000   0.00% Initial
   0xfffffffa00a836000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa00a831000 pending          200 0x00000000000000c0/0x0000000000000dd0   0.00% rcar-avb-0
  *0xfffffffa00a828000 running          253 0x00000000000007a0/0x0000000000004dd0   0.02% DriverDebugService
   0xfffffffa6ff353000 pending          200 0x00000000000005a0/0x0000000000005dd0   0.00% InetServer
   0xfffffffa6ff341000 pending          200 0x00000000000000f0/0x0000000000003dd0   0.00% FibArpRefreshTask0
   0xfffffffa6ff32b000 pending          127 0x0000000000000080/0x0000000000000dd0   0.00%
PingWatchdog_ResetTask
 ivfsserver_devtree_module                  0x00000000000ca000/0x000000000073d000
   0xfffffffa6c0071000 exited           127 0x0000000000000880/0x0000000000002000   0.00% Initial
   0xfffffffa00a736000 pending          254 0x0000000000000280/0x0000000000000dd0   0.00% OSAAgent
   0xfffffffa6ff954000 pending          152 0x00000000000000c0/0x0000000000003dd0   0.00% HealthMonitor
   0xfffffffa6ff949000 pending          150 0x0000000000000a40/0x0000000000008dd0   0.00% FileServer
   0xfffffffa6ff942000 pending          149 0x00000000000005d0/0x0000000000003dd0   0.00% NFSTimerTask
   0xfffffffa6ff8c1000 pending          152 0x0000000000000180/0x0000000000004dd0   0.00% IOTask
   0xfffffffa6ff8b4000 pending          150 0x0000000000000910/0x0000000000008dd0   0.00% IOAssistant0
   0xfffffffa6ff8ab000 pending          150 0x0000000000000690/0x0000000000006dd0   0.00% Syncer
   0xfffffffa6ff8a4000 halted           151 0x0000000000000090/0x0000000000003dd0   0.00% Unmounter
   0xfffffffa6ff88d000 pending          150 0x0000000000000490/0x0000000000008dd0   0.00% IOAssistant1
 devtree_generic_server_module              0x00000000002b6000/0x0000000001501000
   0xfffffffa6c0070000 pending          127 0x0000000000000860/0x0000000000002000   0.00% Initial
   0xfffffffa6fbf56000 pending          220 0x00000000000003f0/0x0000000000001dd0   0.00% fb-map-server
```

(5) Consideration

The CPU load difference of the virtualized Linux and the native Linux is 1.52%, and the possible factors of the CPU load difference are interrupt virtualization overhead and the debug service overhead of the INTEGRITY kernel.

5.1.5. Overhead API Forwarding performance (for Hypervisor)

(1) Description

Measure the GIPC communication of API Forwarding when using the Center Information application on virtualization Linux only GIPC test application in both OSes.

The following figure shows the measurement part of the API transfer.

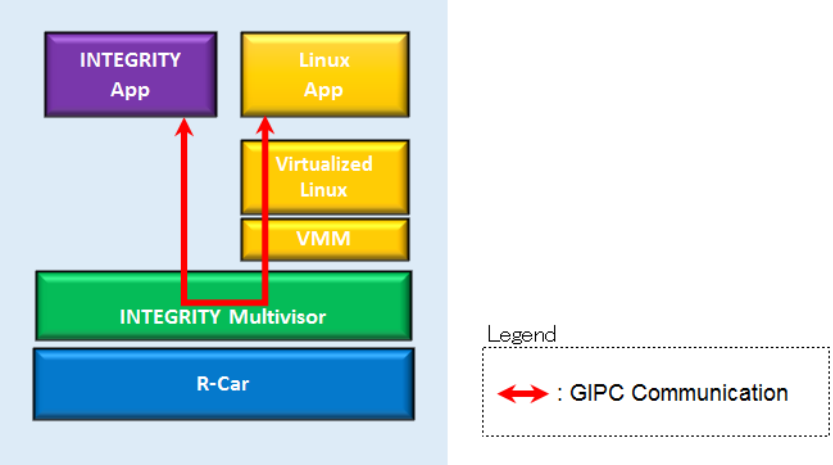


Figure 5-5: Images of API transfer

GIPC is an API provided for INTEGRITY OS to communicate INTEGRITY and Linux.

(2) Precondition

- Measure on virtualization Linux (Type4)
- Use GIPC communication performance tool only. (RENESAS original)
- Transfer data

  The communication size (amount) to be measured is as follows.

  16byte, 128byte, 4Kbyte, 64Kbyte, 128 byte, 256 byte, 384 byte, 512 byte, 640 byte, 768 byte, 896 byte, 1024 byte, 1152 byte, 1280 byte
- Verified 100,1000,10000 times and use the average as the result value.

(3) How to measure

1. Select [Target] - [Connect] from Menu bar of MULTI.
2. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
3. Select “Run mode target”
4. Select [Target] - [Load Module] - [Load Module...] from Menu bar.
5. Load the " GIPCMeasure_dyn.ael" file included in the deliverables.
6. Press F5 for start.

7. Login to Linux from Terminal software.

```
salvator-x login: root
```

8. Run the following command to change the directory.

```
root@salvator-x:~# cd tools
```

9. Run the following command to measure the GIPC communication.

```
root@salvator-x:~/tools# ./gipc_main fbcon_gipc_alias
```

After finishing a command, you will see the log like below.

Check the following results at the number of times, size, and communication time.

```
number of loops = 100
buffer size = 16
processing time:615
number of loops = 100
buffer size = 128
processing time:637
number of loops = 100
buffer size = 256
processing time:665
```

(4) Result

Table 5-10: Result

	Virtualized Linux (Type4) [us]
size	10000 times
16byte	5.67
128byte	5.63
256byte	5.60
384byte	5.60
512byte	5.57
640byte	5.56
768byte	5.55
896byte	5.57
1024byte	5.54
1152byte	5.57
1280byte	5.54
4Kbyte	5.53
64Kbyte	5.54

(5) Consideration

There is some interference from other Tasks running that causes the average to differ so much from the minimum, which requires further Renesas investigation to eliminate.

5.1.6. Math operation (for Hypervisor)

(1) Description

Measure the floating point performance when using the Center Information application on virtualization PoC using whetstone of Unixbench.

(2) Precondition

- Measure on virtualization PoC and native Linux (Type1 and Type2) Native Linux value is reference.
- Use Unixbench on terminal software.

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/UnixBench
```

3. Run the following command to measure the floating point performance 10 times.

```
root@salvator-x:~/tools/UnixBench# i=1; while [ $i -le 10 ]; do ./Run -c 4
whetstone-double; i=$(expr $i + 1);done
```

After finishing a command, you will see the log like below.

Red square is results.

```
Benchmark Run: Fri Jan 02 1970 00:01:29 - 00:05:28
4 CPUs in system; running 4 parallel copies of tests

Double-Precision Whetstone                      5701.0 MWIPS (9.9 s, 7 samples)
```

(4) Result

Table 5-11: Result(MWIPS)

Test environment	Ave.	1	2	3	4	5	6	7	8	9	10
Native Linux (Type2)	5714.46	5712.10	5715.20	5715.00	5714.10	5716.40	5711.90	5716.20	5717.20	5711.20	5715.30
Virtualization PoC (Type1)	5728.36	5701.00	5689.10	5734.50	5736.50	5736.50	5737.20	5738.50	5736.20	5737.50	5736.60

Table 5-12: Overhead(%)

Test environment	Value(MWIPS)	Overhead: ((B – A) /A)*100
(A) Native Linux (Type2)	5714.46	0.24%
(B) Virtualization PoC(Type1)	5728.36	

(5) Consideration

The result is as expected, the virtualization overhead is less than the variation of each test result.

5.2. Bus Load/Bandwidth

5.2.1. Total bus bandwidth on virtualization environment

(1) Description

Measure the DDR memory bandwidth (MB /s) on virtualization PoC.

Measurement tool is bandwidth monitoring tool that is made by Renesas.

(2) Precondition

- Measure on virtualization PoC (Type1)
- Use bandwidth monitoring tool for INTEGRITY.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Select [Target] - [Connect] from Menu bar.
2. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
3. Select “Run mode target”
4. Select [Target] - [Load Module] - [Load Module...] from Menu bar.
5. Load the "Busmoni_dyn.ael" file in the following path.
   <...int1144\modules\renesas\app\Busmoni_sample\Busmoni>
6. Press F5 for start.
7. Run the following command on "I/O" tab.

```
Please input command: -p UC11 -c 1
```

After finishing a command, you will see the log like below.

Red square is results.

```
Please input command: -p UC11 -c 1
Device "R-Car H3 ES1.1"
-----------------------------------------------------------------
UC11:  DDR   R =  3000MiB/s        W =  1945MiB/s
-----------------------------------------------------------------
```

8. Select [Target] - [Unload Module] - [Unload Module...] from Menu bar.
9. Unload the "Busmoni_dyn.ael" file in the following path.
   <...int1144\modules\renesas\app\Busmoni_sample\Busmoni>
10. Run from step 3 to step8 process 10 seconds after the result is displayed.
    Repeat this 9 times.

(4) Result

Table 5-13: Result

	Virtualization PoC (Type1)	
	R(MB/s)	W(MB/s)
Ave.	3070.23	2004.67
1	3145.73	2039.48
2	2889.88	1977.61
3	3011.51	2037.38
4	3058.70	2018.51
5	3178.23	1991.25
6	2984.25	2101.35
7	3159.36	1974.47
8	3197.11	2048.92
9	2904.56	1872.76
10	3172.99	1984.95

(5) Consideration

Approximately 5.1GB/s (read+write) is observed in this virtualization environment.

The calculated DDR bus bandwidth is 38.4GB/s (DDR2400 x 32 bit x 4 channels), this use case is considered that it has enough performance margin.

5.2.2. Total bus bandwidth on native Linux environment

(1) Description

Measure the DDR memory bandwidth (MB / s) on native Linux.

Measurement tool is bandwidth monitoring tool.

(2) Precondition

- Measure on native Linux (Type2)
- Use bandwidth monitoring tool for Linux.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/busmoni
```

3. Run the following command for measurement.

```
root@salvator-x:~/tools/busmoni# ./busmoni_app -p UC11 -c -1
```

After finishing a command, you will see the log like below.

Red square is results.

```
root@salvator-x:~/tools/busmoni# ./busmoni_app -p UC11 -c -1
Device "R-Car H3 WS1.1"
-----------------------------------------------------------
UC11:  DDR   R =  2076MB/s        W =  1523MB/s
------------------------------------------------------------
```

(4) Result

Table 5-14: Result

	Native Linux (Type2)	
	R(MB/s)	W(MB/s)
Ave.	2259.80	1541.70
1	2269.00	1602.00
2	2264.00	1563.00
3	2212.00	1347.00
4	2173.00	1337.00
5	2261.00	1538.00
6	2245.00	1544.00
7	2272.00	1589.00
8	2303.00	1634.00
9	2297.00	1626.00
10	2302.00	1637.00

(5) Consideration

Approximately 3.8GB/s (read+write) is observed in this environment.
This is the case of Linux side only. This result is as same as expected.

5.2.3. Total bus bandwidth on native INTEGRITY environment

(1) Description

Measure the DDR memory bandwidth (MB / s) on native INTEGRITY.

Measurement tool is bandwidth monitoring tool.

(2) Precondition

- Measure on native INTEGRITY (Type3)
- Use bandwidth monitoring tool.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Refer to 5.2.1

(4) Result

Table 5-15: Result

	Native INTEGRITY (Type3)	
	R(MB/s)	W(MB/s)
Ave.	716.91	555.75
1	809.50	578.81
2	748.68	579.86
3	782.24	631.24
4	757.07	560.99
5	683.67	571.47
6	736.10	568.33
7	700.45	546.31
8	754.97	551.55
9	706.74	565.18
10	489.68	403.70

(5) Consideration

Approximately 1.3GB/s (read+write) is observed in this virtualization environment.

This is the case of INTEGRITY side only. This result is as same as expected.

5.2.4. The overhead (DDR memory bandwidth) compared virtualized Linux with native Linux

(1) Description

Compare the overhead (DDR memory bandwidth) on virtualized Linux and native Linux.

(2) Precondition

- Measure on virtualized Linux and native Linux (Type4 and Type2)

(3) How to measure

1. Measure only type 4. Measurement method refer to 5.2.1.
2. Check the overhead as follows. The comparison result is 5.2.2.
   ((Virtualized Linux) - (Native Linux)) / (Native Linux) * 100 [%]

(4) Result

Table 5-16: Type4 Result

	Virtualized Linux (Type4)	
	R(MB/s)	W(MB/s)
Ave.	2263.88	1489.29
1	2246.05	1502.61
2	2307.92	1538.26
3	2294.28	1556.09
4	2403.34	1560.28
5	2311.06	1429.21
6	2225.08	1363.15
7	2259.68	1531.97
8	2306.87	1569.72
9	2211.45	1376.78
10	2073.03	1464.86

Table 5-17: Read Result

Test environment	R (MB/s)	Overhead: ((B – A) /A)*100
(A) Native Linux (Type2)	2259.80	0.18%
(B) Virtualized Linux (Type4)	2263.88	

Table 5-18: Write Result

Test environment	W (MB/s)	Overhead: ((B – A) /A)*100
(A) Native Linux (Type2)	1541.70	-3.4%
(B) Virtualized Linux (Type4)	1489.29	

(5) Consideration

The bus utilization difference between the virtualized Linux and the native Linux is observed almost same in read access, but the write access bandwidth of the virtualized Linux is 3.4% lower than the native Linux. One possibility is that the virtualized Linux has the interrupt processing overhead and may affect to the GPU workload and reduce the write access bandwidth. Further investigation is necessary for the detailed explanation.

5.3. Bus Latency

5.3.1. Bus Latency on virtualization environment

Out of Scope.

5.3.2. Bus Latency on native LINUX environment

Out of Scope.

5.3.3. Bus Occupancy on virtualization environment

(1) Description

Calculate maximum theoretical performance from H/W specification.

(2) Precondition

- Calculate from memory controller specification.
- Memory controller for LPDDR4-3200 with 32 bits x 4 channels.
  However, this time Salvator is 2400 MHz due to the limitation of SoC.

(3) How to measure

2400(MHz)x 32bit x 4ch = 307200 Mbit/s = 38400MB/s = 38.4GB/s

(4) Result

Table 5-19: Result

Test environment	Score
maximum theoretical bandwidth	38.4GB/s

(5) Consideration

This result is calculated by hardware specification. R-Car H3 maximum bandwidth is 38.4GB/s.
But this score will be improved to 51.2GB/s on R-Car H3 next silicon.

5.3.4. Insufficient Bus utilization for native Linux environment

Out of Scope.

5.3.5. Bus Data Occupancy

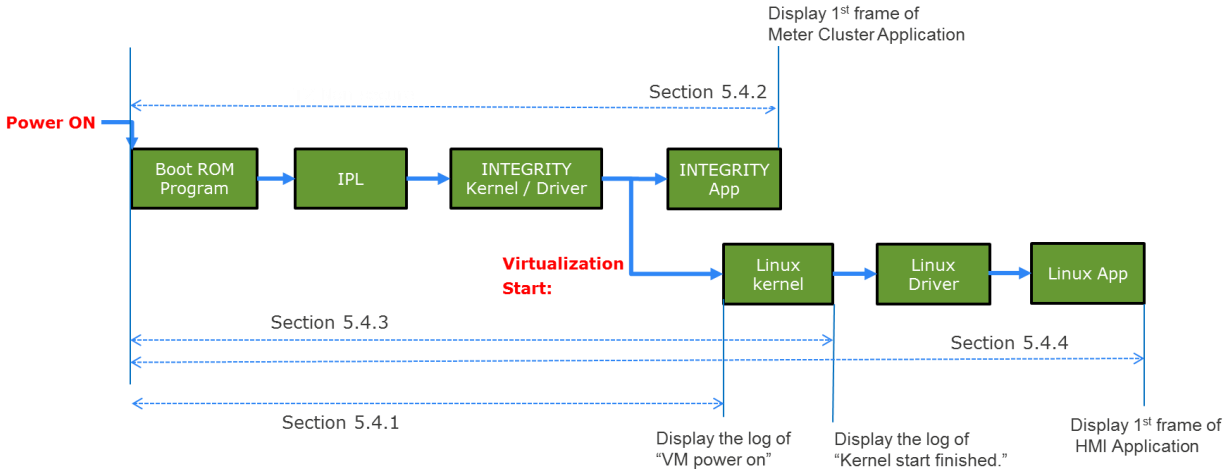
Out of Scope.

5.3.6. Ethernet bus utilization

Out of Scope.

5.4. Boot Time

This section measure the time from power on to each OS / application starts up.
The following figure describes the Booting sequence of virtualization PoC and measurement point.



Legend

Boot ROM Program:	Built-in mask ROM program to load the IPL from external serial flash memory.
IPL:	Initial Program Loader to load OS and/or U-boot from external serial flash memory.

Figure 5-6: Simplified BOOT Sequence

In this section, we are using following optimization to create the INTEGRITY monolith image.

- Decreasing the size of a monolith's binary in rcar_kernel_wrapper.readme.txt

- Multivisor Fast Boot in release_notes.txt
  This system is setting unused ram size to 112MB
  ghs,size-megabytes = <112>;

Above files are provided by GHS.
Note that the virtualized Linux boot time can be reduced by using the updated INTEGRITY environment. This result is currently measured in T9.0 environment. It is known that the T10.1 release reduces boot time by approximately 1 second, and further releases are expected to reduce boot time even further.

Attention:
If you have modified the HyperFlash contents of supplied Salvator-X board, please write Monolith(Type1_mono.5.4) to HyperFlash in Appendix A.

5.4.1. From power on to booting of INTEGRITY OS

(1) Description

Measure the time from power on until ready of INTEGRITY Kernel to start virtual machine on virtualization PoC.

Measurement tool is boot time measurement tool. (RENESAS original)

The following figure describes measurement target.

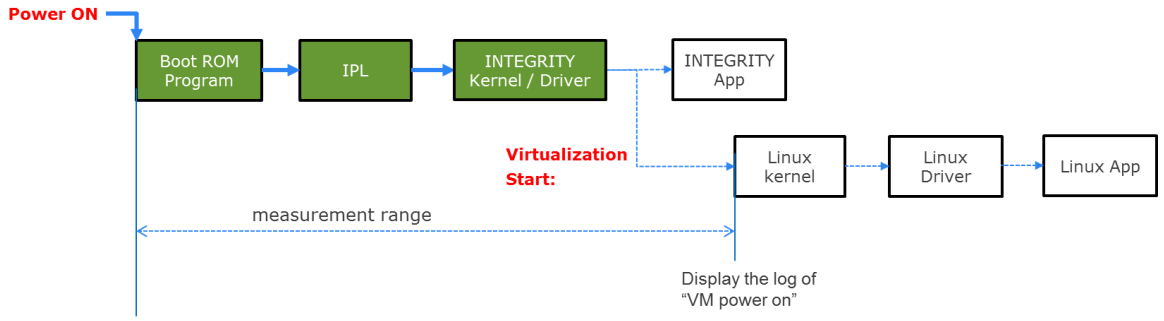


Figure 5-7: Images of power on to booting of INTEGRITY OS

(2) Precondition

- Measure on virtualization PoC (Type1)
- Verified 10 times and use the average as the result value.
- Measure time until the log of "VM power on" in INTEGRITY side.
- Take the boot situation on video.

(3) How to measure

1. Take the boot situation on video.
2. Search the frame which Salvator-X LED14 changes to ON from the video.
3. Search the frame which contains the INTEGRITY Kernel log "VM powered on" from the video.

```
[    2.04387] VM powered or
```

4. Calculate the interval time between the frames specified at the above step 2 and step 3.

(4) Result

Table 5-20: Result

	Virtualization PoC (Type1) [sec]
Ave.	3.38
1	3.37
2	3.40
3	3.37
4	3.37
5	3.34
6	3.40
7	3.40
8	3.40
9	3.37
10	3.37

(5) Consideration

In the boot time analysis, we use IPL to load the INTEGRITY OS image from HyperFlash directly, instead of loading the u-boot and use it for loading INTEGRITY OS image from either network or flash storage (SD Card, USB flash, or eMMC) to simulate the actual target hardware.

At the result of 3.38sec and the message “VM powered on” is 2.04sec so it is calculated that 1.34sec is the time before stating INTEGRITY Kernel including IPL initialization and image copy.

This time we are using the serial output on IPL for development, but actually it is possible to reduce the boot time by disabling IPL's serial message output.
The estimated boot time from power on to launch the INTEGRITY kernel is below:

INTEGRITY kernel launch time =
17ms (power-on reset time of Salvator-X board)
+ 77ms (IPL common processing time)
+ 386ms (IPL loading and validation time for 32MB INTEGRITY image from HyperFlash)
= 480ms

5.4.2. From power on to starting up of Meter cluster application on INTEGRITY

(1) Description

Measure the time from power on until the completion of the 1st frame.

Measurement tool is boot time measurement tool. (RENESAS original)

The following figure describes measurement target.

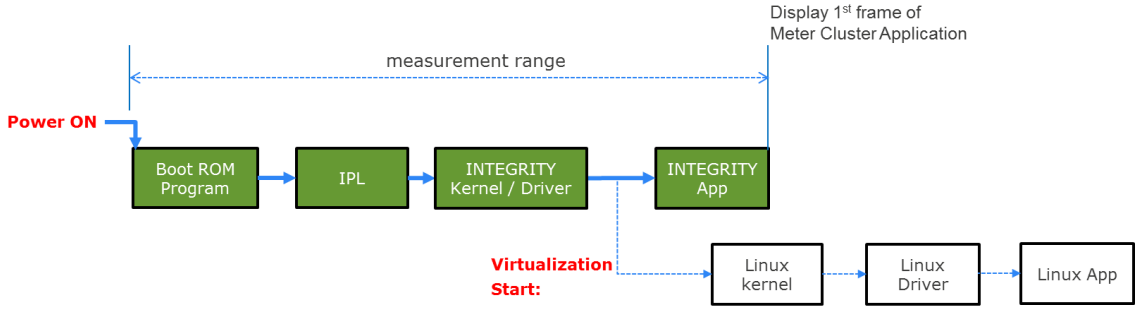


Figure 5-8: Images of power on to starting up of Meter cluster application on INTEGRITY

(2) Precondition

- Measure on virtualization PoC (Type1)
- Verified 10 times and use the average as the result value.
- Measure time until the Meter Cluster Application is launched in HDMI1 Display.
- Use the video taken at section 5.4.1.

(3) How to measure

1. Take the boot situation on video.
2. Search the frame which Salvator-X LED14 changes to ON from the video.
3. Search the frame which contains a title ”R-Car” in orange color, which is output in the HDMI1 display right after the Meter Cluster Application is launched.
4. Calculate the interval time between the frames specified at the above step 1 and step 2.

(4) Result

Table 5-21: Result

	Virtualization PoC (Type1) [sec]
Ave.	4.04
1	4.04
2	4.07
3	4.00
4	4.04
5	4.00
6	4.07
7	4.04
8	4.07
9	4.04
10	4.04

(5) Consideration

The time difference from section 5.4.1 is observed only 660ms. The GFX Library usually take longer time for initialization before drawing the 1st frame so we are estimating that the GFX driver initialization start earlier than the time of “VM powered on”.

5.4.3. From power on to booting of Linux OS

(1) Description

Verify the time from power on until ready of Linux Kernel on virtualization PoC.

The following figure describes measurement target.

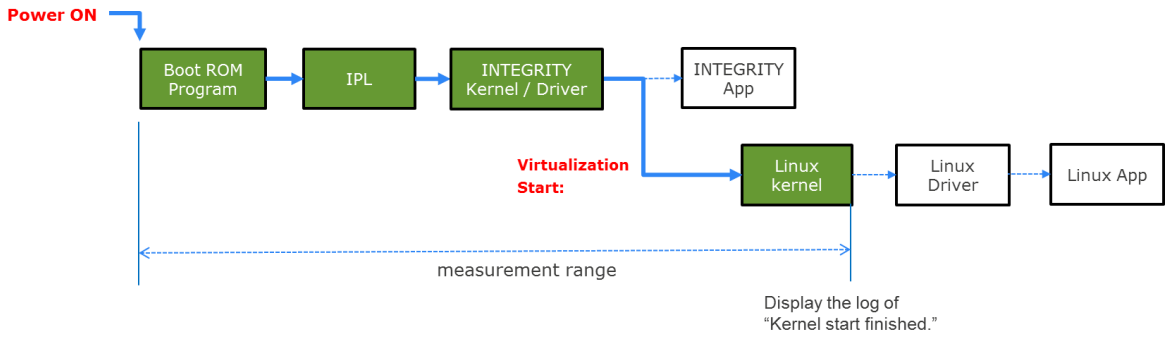


Figure 5-9: Images of power on to booting of Linux OS

(2) Precondition

- Measure on virtualization PoC (Type1)
- Verified 10 times and use the average as the result value.
- Measure time until the log of "Kernel start finished" is output on Linux side.
- Use the video taken at section 5.4.1.

(3) How to measure

1. Take the boot situation on video.
2. Search the frame which Salvator-X LED14 changes to ON from the video..
3. Search the frame which contains the Linux Kernel log "Kernel start finished." from the video.

```
[    2.376525] Kernel start finished.
```

4. Calculate the interval time between the frames specified at the above step 1 and step2.

(4) Result

Table 5-22: Result

	Virtualization PoC (Type1) [sec]
Ave.	5.94
1	6.17
2	6.50
3	6.07
4	5.24
5	5.67
6	6.07
7	5.70
8	5.80
9	6.10
10	6.07

(5) Consideration

Refer 5.4.1.

5.4.4. From power on to starting up of Video app and MAP/HMI of graphics on Linux OS

(1) Description

Verify the time from power on program until the completion of the 1st frame of Video and Map/HMI on virtualization PoC.

The following figure describes measurement target.

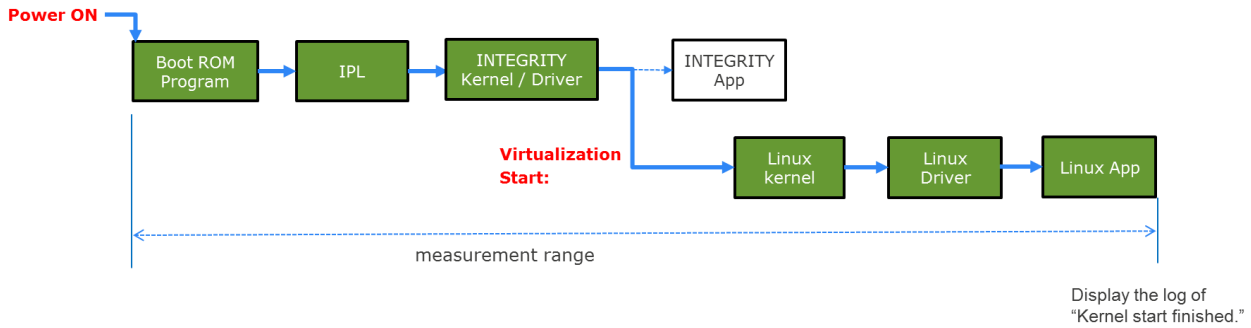


Figure 5-10: power on to starting up of Video app and MAP/HMI of graphics on Linux OS

(2) Precondition

- Measure on virtualization PoC (Type1)
- Verified 10 times and use the average as the result value.
- Measure time until the HMI Application is launched in HDMI0 Display.
- Use the video taken at section 5.4.1.

(3) How to measure

1. Take the boot situation on video.
2. Search the frame which Salvator-X LED14 changes to ON from the video.
3. Search the frame which contains the HMI image, which is output in the HDMI0 display right after the HMI Application is launched.
4. Calculate the interval time between the frames specified at the above step 1 and step2.

(4) Result

Table 5-23: Result

	Virtualization PoC (Type1) [sec]
Ave.	20.88
1	20.64
2	20.74
3	20.97
4	21.47
5	20.07
6	20.97
7	20.97
8	21.37
9	20.97
10	20.60

(5) Consideration

Refer 5.4.1.

5.5. Interrupt Time

5.5.1. Delay time for interrupt

(1) Description

Measure the interrupt performance on virtualized Linux using Cyclictest.

(2) Precondition

- Measure on virtualized Linux (Type4) without other applications
- Use Cyclictest on terminal software.
- Verified 10 times and use the average as the result value.
- This environment is set kernel optimization (kernel_opt).

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools
```

3. Run the following Cyclictest's command.

```
root@salvator-x:~/tools# /cyclictest -n -p 70 -i 4 -l 100000 -q -a
```

After finishing a command, you will see the log like below.

Red square is results.

```
# /dev/cpu_dma_latency set to 0us
T: 0 ( 1733) P:70 I:4 C: 100000 Min:      3 Act:    4 Avg:    5 Max:      29
```

4. Run the step 2 process 10 seconds after the result is displayed.

Repeat this 9 times.

(4) Result

Table 5-24: Result

	Virtualized Linux (Type4)		
	Ave.(us)	**Min.(us)**	**Max.(us)**
	Ave 5	Min 3	Max 35
1	5.00	3.00	29.00
2	5.00	4.00	31.00
3	5.00	3.00	27.00
4	5.00	3.00	29.00
5	5.00	4.00	31.00
6	5.00	3.00	35.00
7	5.00	3.00	25.00
8	5.00	4.00	30.00
9	6.00	3.00	26.00
10	4.00	3.00	26.00

(5) Consideration

This result is expected.

5.5.2. Delay time variation

(1) Description

Measure the maximum and minimum interrupt performance value on virtualized Linux using Cyclictest.

(2) Precondition

- Extract maximum value and minimum value at 5.5.1 results.

(3) How to measure

1. Type 4: Refer to 5.5.1.

(4) Result

Table 5-25: Result

	Min(us)	Max(us)
Virtualized Linux (Type4)	3	35

(5) Consideration

This result is expected.

5.5.3. Lock Synchronization latency

Out of Scope.

5.6. Drawing Performance

5.6.1. FPS on Linux graphics

(1) Description

Measure the FPS of the Linux graphics application (glmark2 2014). The target OS is native Linux, and the window size is 1920x720.

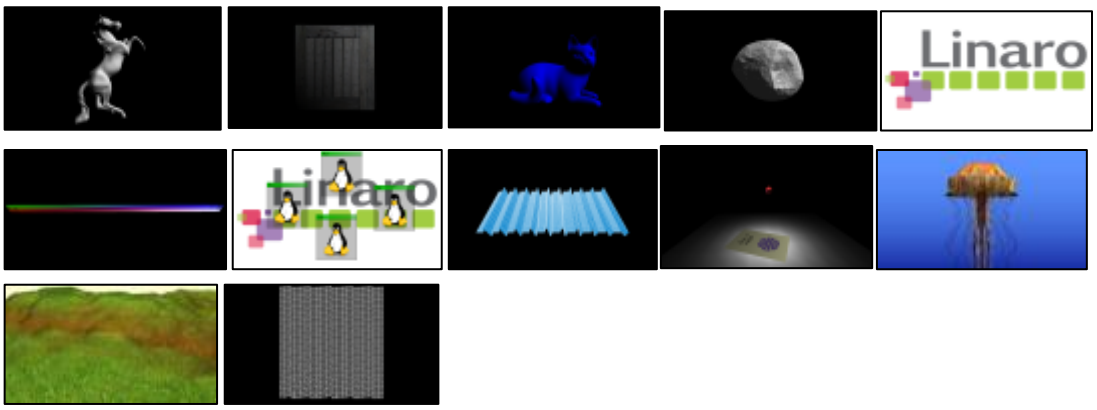


Figure 5-11: Images of glmark2 2014 (extracted)

(2) Precondition

- Measure on Native Linux. (Type2)
- Application: glmark2 2014 2.0.1 (it shall be stored in the root file system)
- Window size for Linux: 1920x720
- Window System: DRM

Note: glmark2 runs without VSYNC synchronization. Therefore, the FPS will be more than 60 FPS.

(3) How to measure

1. Login to Linux

```
salvator-x login: root
```

2. Stop Weston using the following command.

```
root@salvator-x:~# systemctl stop weston
```

3. Edit /etc/powervr.ini as follows to use the DRM Window System and HDMI output.

   Before the modification

```
WindowSystem=libpvrWAYLAND_WSEGL.so
```

   After the modification

```
;WindowSystem=libpvrWAYLAND_WSEGL.so
WindowSystem=libpvrDRM_WSEGL.so

;Use HDMI output
SetDefaultDisplay=1
```

4. Disable Weston using the following command.

```
root@salvator-x@:~# systemctl disable weston
```

5. Reboot Salvator-X
6. Push any key on the terminal before counting down of the U-boot to setup the U-boot.
7. Enter editenv command as follows.

```
=> editenv bootargs
```

8. You will see the following line.

```
edit: bootargs = "consoleblank=0 console=ttySC0,115200 rw
rootwait root=/dev/mmcblk0p2 cma=512M"
```

Add “rootdelay=10 video=HDMI-A-1:1920x720@60m” as follows and press Enter.

```
edit: bootargs = "consoleblank=0 console=ttySC0,115200 rw
rootwait root=/dev/mmcblk0p2 cma=512M rootdelay=10 video=HDMI-A-
1:1920x720@60m"
```

9. Save the see the following line.

```
=> saveenv
```

10. Reboot Salvator-X and start Linux.
11. Login to Linux

```
salvator-x login: root
```

12. Run glmark2 as follows

```
root@salvator-x:~# cd glmark2
root@salvator-x:~/glmark2# ./glmark2-es2-nullws --size 1920x720
```

After finishing glmark2, you will see the log like below. Get the “glmark2 Score: XXXX” as a FPS result.

```
=======================================================
    glmark2 2014.03
=======================================================
    OpenGL Information
    GL_VENDOR:     Imagination Technologies
    GL_RENDERER:   PowerVR Rogue GX6650
    GL_VERSION:    OpenGL ES 3.2 build 1.7@4276001
=======================================================
(omitted)
=======================================================
                                  glmark2 Score: 1246
=======================================================
```

(4) Result

Table 5-26: Result

Test environment	Application	Score [FPS]
Native Linux (Type2)	glmark2 2014	1246

(5) Consideration

For comparisons between native INTEGRITY or virtualized Linux, refer to 5.6.2 or 5.6.3.

5.6.2. FPS on INTEGRITY graphics

(1) Description

Measure the FPS of the INTEGRITY graphics applications (glmark2 2014 and GFXBench3.0). The target OS is INTEGRITY Multivisor and the window size is 1920x720.

Manhattan T-Rex

Figure 5-12: Images of GFXBench 3.0

(2) Precondition

- Measure on native INTEGRITY (Type3)
- Application1: glmark2 2014 2.0.1
- Application2: GFXBench 3.0.22, Manhattan and T-Rex
- Window size for INTEGRITY: 1920x720

Note: GFXBench3.0 shall be installed only when we evaluate it. The deliverables shall not include it. glmark2 2014 for INTEGRITY shall be installed only when we evaluate it because it is GPLv3. It need to be ported from glmark2 2014 for Linux.

Note: glmark2 runs without VSYNC synchronization. Therefore, the FPS will be more than 60 FPS.

(3) How to measure

1. Rebuild the device tree blob file (.dtb) specifying the NFS server address. Store it to the SD card.
2. Store the data for glmark2 or GFXBench to the NFS.
3. Start Type3_glmark2_mono (glmark2) or Type3_GFXBench_mono (GFXBench3.0) using MULTI Debugger.
4. Start the application.
5. In case of glmark2: After finishing glmark2, you will see the log like below. Get the “glmark2 Score: XXXX” as an FPS result.

```
I/O: ======================================================
I/O:     glmark2 2014
I/O: ======================================================
I/O:     OpenGL Information
I/O:     GL_VENDOR:     Imagination Technologies
I/O:     GL_RENDERER:   PowerVR Rogue GX6650
I/O:     GL_VERSION:    OpenGL ES 3.1 build 1.7@4276001 (MAIN)
I/O: ======================================================
(omitted)
I/O: ======================================================
I/O:                                  glmark2 Score: 1201
I/O: ======================================================
```

6. In case of GFXBench3.0: After finishing GFXBench3.0, you will see the log like below. Get '"fps": xxx' an FPS results.

```
Manhattan

I/O: [INFO ]: gl_manhattan_off test running...
I/O: Log: DeleteGLobal: 376d4d0
I/O: [INFO ]: {
I/O:  "results":
I/O:  [
I/O:   {
I/O:    "elapsed_time": 62015,
I/O:    "error_string": "",
I/O:    "gfx_result":
I/O:    {
I/O:     "egl_config_id": -1,
I/O:     "fps": 20.7,
I/O:     "frame_count": 1265,
I/O:     "frametimes": [ 822, 95, 52, 52, 43, 57, 735, 93, 54, 48, 45,
(omitted)
I/O:                     51, 55, 52, 51, 52 ],
I/O:     "graphics_version": "OpenGL ES 3.1 build 1.7@4276001 (MAIN)",
I/O:     "renderer": "PowerVR Rogue GX6650",
I/O:     "surface_height": 720,
I/O:     "surface_width": 1920,
I/O:     "vendor": ""
I/O:    },
I/O:    "load_time": 143594,
I/O:    "measured_time": 62015,
I/O:    "result_id": 0,
I/O:    "score": 1264.89794921875,
I/O:    "status": "OK",
I/O:    "test_id": "gl_manhattan_off",
I/O:    "unit": "frames",
I/O:    "version": 1
I/O:   }
```

```
T-Rex

I/O: [INFO ]: gl_trex_off test running...
I/O: Log: DeleteGLobal: 22e0f88
I/O: [INFO ]: {
I/O:  "results":
I/O:  [
I/O:   {
I/O:    "elapsed_time": 56005,
I/O:    "error_string": "",
I/O:    "gfx_result":
I/O:    {
I/O:     "egl_config_id": -1,
I/O:     "fps": 59.1,
I/O:     "frame_count": 3247,
I/O:     "frametimes": [ 1154, 42, 31, 26, 23, 23, 14, 27, 28, 28, 28,
(omitted)
I/O:                     18, 17, 19, 17, 19 ],
I/O:     "graphics_version": "OpenGL ES 3.1 build 1.7@4276001 (MAIN)",
I/O:     "renderer": "PowerVR Rogue GX6650",
I/O:     "surface_height": 720,
I/O:     "surface_width": 1920,
I/O:     "vendor": ""
I/O:    },
I/O:    "load_time": 76027,
I/O:    "measured_time": 56005,
I/O:    "result_id": 0,
I/O:    "score": 3246.884033203125,
I/O:    "status": "OK",
I/O:    "test_id": "gl_trex_off",
I/O:    "unit": "frames",
I/O:    "version": 1
I/O:   }
I/O:  ]
I/O: }
```

(4) Result

Table 5-27: Result

Test environment	Application	Score [FPS]
Native INTEGRITY (Type3)	glmark2 2014	1201
	GFXBench3.0 Manhattan	20.7
	GFXBench3.0 T-Rex	59.1

(5) Consideration

The glmark2 FPS for native Linux was 1246. Therefore, the difference of the performance between native Linux and native INTEGRITY was ((1246-1201)/1246)*100 = 4%. The difference is currently under investigation and will be updated in a future revision.

5.6.3. The overhead (FPS) compared virtualized Linux with native Linux

(1) Description

Measure the FPS of the Linux graphics application (glmark2 2014). The target OS is both native Linux and virtualized Linux. The window size is 1920x720.

(2) Precondition

Type 2: refer to 5.6.1.

Type 4: see below.

- Measure on virtualized Linux. (Type4)
- Application: glmark2 2014 2.0.1 (it shall be stored in the root file system)
- Window size for Linux: 1920x720
- Window System: DRM

Note: glmark2 runs without VSYNC synchronization. Therefore, the FPS will be more than 60 FPS.

(3) How to measure

Type 2: refer to 5.6.1.

Type 4: see below.

1. Update the device tree blob (.dtb) file to change the window size for Linux. Edit int1144/devtree-arm64/rcar/device-tree-source/r8a7795-salvator-x/r8a7795-salvator-x-multivisor.dtsi as follows.

   Before the modification

   ```
   bootargs = "consoleblank=0 console=ttySC0,115200 rw rootwait
   root=/dev/mmcblk0p2 cma=512M rootdelay=10";
   ```

   After the modification

   ```
   bootargs = "consoleblank=0 console=ttySC0,115200 rw rootwait
   root=/dev/mmcblk0p2 cma=512M rootdelay=10 video=HDMI-A-
   1:1920x720@60m";
   ```

2. Build the .dtb file and store it to the SD card.
3. Start Type4_mono using MULTI Debugger.
4. Login to Linux using Terminal.

   ```
   salvator-x login: root
   ```

5. Stop Weston using the following command.

   ```
   salvator-x@root# systemctl stop weston
   ```

6. Edit /etc/powervr.ini as follows to use the DRM Window System.

   Before the modification

   ```
   WindowSystem=libpvrWAYLAND_WSEGL.so
   ```

   After the modification

   ```
   ;WindowSystem=libpvrWAYLAND_WSEGL.so
   WindowSystem=libpvrDRM_WSEGL.so
   ```

7. Disable Weston using the following command.

```
salvator-x@root# systemctl disable weston
```

8. Restart Type4_mono.
9. Login to Linux.

```
root@salvator-x:~# systemctl stop weston
```

10. Run glmark2 as follows

```
root@salvator-x:~# cd glmark2
root@salvator-x:~/glmark2# ./glmark2-es2-nullws --size 1920x720
```

After finishing glmark2, you will see the log like below. Get the “glmark2 Score: XXXX” as a FPS result.

```
=======================================================
    glmark2 2014.03
=======================================================
    OpenGL Information
    GL_VENDOR:      Imagination Technologies
    GL_RENDERER:    PowerVR Rogue GX6650
    GL_VERSION:     OpenGL ES 3.2 build 1.7@4276001
=======================================================
(omitted)
=======================================================
                                  glmark2 Score: 1121
=======================================================
```

11. Check the overhead as follows.

((Native Linux FPS) - (Virtualized Linux FPS)) / (Native Linux FPS) * 100 [%]

(4) Result

Table 5-28: Result

Test environment	Application	Score [FPS]	Overhead: ((A – B) /A)*100
(A) Native Linux (Type2)	glmark2 2014	1246	10.0%
(B) Virtualized Linux (Type 4)	glmark2 2014	1121	

(5) Consideration

The overhead was 10%. As this is unexpected, we are investigating this. Currently we found that especially glBindFramebuffer(), glDrawArrays(), and eglSwapBuffers() on virtualized Linux took much time than those on native Linux. We will investigate it to reduce the overhead.

5.7. Video & Audio Performance

5.7.1. FPS on Linux Video decode

(1) Description

Measure the video display performance (Frame per second) of Center Information when using media player to playback Video on virtualization PoC.

(2) Precondition

- Measure on virtualization PoC (Type1)
- Incorporate a mechanism to measure the fps (Frame per second) at media player using the following files.

Table 5-29: Lists of Video attributes

File name	size	FPS	Bitrate (Kbps)	Codec
big_buck_bunny_720p_h264.mp4	1280x720	24.000	5283	H.264
big_buck_bunny_720p_h264_60fps.mp4 (*1)	1280x720	60.000	4889	H.264

*1 : re-encoding big_buck_bunny_720p_h264.mp4 to FPS 60

- Disable automatic playback of Video/Audio playback app.
  1. Login to Linux.
     ```
     salvator-x login: root
     ```
  2. Run the following command to change the directory.
     ```
     root@salvator-x:~# cd /etc/init.d/
     ```
  3. Edit “rundemo” file.
     ```
     root@salvator-x:/etc/init.d# vi rundemo
     ```
  4. Enter "i" for insert mode
  5. Add a "#" to the beginning of "/home/root/movie/playloop.sh"
  6. Press "Esc" button for command mode
  7. Press ":wq" button for write and finish
  8. Reboot Salvator-X

Note) Delete "#" after the test

```
#!/bin/sh
sleep 5
export XDG_RUNTIME_DIR=/run/user/root
export LD_LIBRARY_PATH=/home/root/Futuremark


sleep 5
cd /home/root/IMG_SDK35

export WSEGL_ENABLE_TRIPLE_BUFFERING=2
./OGLES3Coverflow -aasamples=4 -width=1920 -height=720 -posx=0 -posy=0
&


#/home/root/movie/playloop.sh &

sleep 2

/home/root/camera.sh &

sleep 180
reboot
```

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command.

```
root@salvator-x:~# amixer set "DVC Out" 1%
```

3. Run the following command.

```
root@salvator-x:~# modprobe -a mmngr mmngrbuf vspm vspm_if vsp2 uvcs_drv
```

4. Run the following command.

```
root@salvator-x:~# export XDG_RUNTIME_DIR=/run/user/root
```

5. Run the following command.

```
root@salvator-x:~# gst-launch-1.0 --padprobe v:sink --timer filesrc
location=movie/big_buck_bunny_720p_h264.mp4 ! qtdemux name=demux
demux.audio_0 ! queue ! omxaaclcdec ! alsasink device=hw:0,0 demux.video_0 !
queue ! h264parse ! omxh264dec ! vspfilter ! video/x-raw, format=BGRA,
width=1920, height=720 ! waylandsink name=v
```

After finishing a command, you will see the log like below.

Red square is a result.

```
Got EOS from element "pipeline0".
Execution ended after 0:09:56.509504679
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Total time: 596.509521 seconds
Frames: 14155 processed
Avg. FPS: 23.73
```

6. Run the following command.

```
root@salvator-x:~# gst-launch-1.0 --padprobe v:sink --timer filesrc
location=movie/big_buck_bunny_720p_h264_60fps.mp4 ! qtdemux name=demux
demux.audio_0 ! queue ! omxaaclcdec ! alsasink device=hw:0,0 demux.video_0 !
queue ! h264parse ! omxh264dec ! vspfilter ! video/x-raw, format=BGRA,
width=1920, height=720 ! waylandsink name=v
```

After finishing a command, you will see the log like below.

Red square is a result.

```
Got EOS from element "pipeline0".
Execution ended after 0:09:56.541268921
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Total time: 596.541260 seconds
Frames: 30287 processed
Avg. FPS: 50.77
```

(4) Result

Table 5-30: Result

virtualization PoC (Type1)	Value [FPS]
big_buck_bunny_720p_h264.mp4	23.96
big_buck_bunny_720p_h264_60fps.mp4	51.23

(5) Consideration

We are expecting the full frame rate in playback, and the result for the 60fps content (big_buck_bunny_720p_h264_60fps.mp4) is not expected. Further investigation will be necessary to identify the problem. Refer 5.7.2.

5.7.2. The overhead (FPS) compared virtualized Linux with native Linux

(1) Description

Compare the video display performance (Frame per second) of Center Information when using media player to playback Video/Audio on virtualized Linux and native Linux.

(2) Precondition

- Measure on virtualized Linux and native Linux (Type4 and Type2)
- Video/Audio application run only.
- Incorporate a mechanism to measure the fps (Frame per second) at media player using the following files.

Table 5-31: Lists of Video attributes

File name	size	FPS	Bitrate (Kbps)	Codec
big_buck_bunny_720p_h264.mp4	1280x720	24.000	5283	H.264
big_buck_bunny_720p_h264_60fps.mp4 (*1)	1280x720	60.000	4889	H.264

*1 : re-encoding big_buck_bunny_720p_h264.mp4 to FPS 60

- Disable automatic playback of Media player. Refers to 5.7.1.
- Compare the performance between virtualized Linux and native Linux.

(3) How to measure

1. Measurement method refers to 5.7.1.

(4) Result

Table 5-32: Result (FPS)

	(A) Native Linux (Type2) [FPS]	(B) Virtualized Linux (Type4) [FPS]	Performance difference: (A-B / A)*100 [%]
big_buck_bunny_720p_h264.mp4	24.00	23.85	0.63%
big_buck_bunny_720p_h264_60fps.mp4	54.51	51.05	6.35%

(5) Consideration

We are expecting the full frame rate in playback on both virtualized Linux and native Linux, the result for the 60fps content (big_buck_bunny_720p_h264_60fps.mp4) is not expected. Further investigation, especially for the Linux's software structure related to the movie decoding and screen composition, will be required to identify the problem.

About the performance difference between virtualized Linux and native Linux, one possibility is the performance difference of SD Card driver. The SD card read performance is measured 5.0MB/s on virtualized Linux, and 17.9MB/s on native Linux, due to the clock difference of SD host controller.

5.7.3. Audio playback performance

(1) Description

Measure the audio playback performance of Center Information when using media player on virtualization PoC.

(2) Precondition

- Measure on virtualization PoC (Type1)
- Check that there is no noise in playing the specified audio file(s).

Table 5-33: Lists of Audio attributes

File name	Sampling rate(KHz)	Bitrate (Kbps)	Codec
big_buck_bunny_720p_h264.mp4	44.1	132	AAC

- Check by listening.

(3) How to measure

1. Measurement method refers to 5.7.1.

(4) Result

Sound skipping doesn't occurs on virtualization PoC.

(5) Consideration

This result is expected.

5.7.4. H.264 decoder/encoder latency

(1) Description

Measures the performance (Frame per second) of H.264 Decoder outputs vide of Center Information on virtualization PoC. The requirement of the decoder is to prepare the frame within the specified time (24fps, 60fps). Therefore, measure the frame rate of the output.

We have evaluated decoder only. The encoder is not supported on this virtualization environment (Yocto 2.7.0).

(2) Precondition

- Measure on virtualization PoC (Type1)
- Incorporate a mechanism to measure the decode performance media player using the following files.

Table 5-34: Lists of Video attributes

File name	size	FPS	Bitrate (Kbps)	Codec
big_buck_bunny_720p_h264.mp4	1280x720	24.000	5283	H.264
big_buck_bunny_720p_h264_60fps.mp4 (*1)	1280x720	60.000	4889	H.264

*1 : re-encoding big_buck_bunny_720p_h264.mp4 to FPS 60

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command.

```
root@salvator-x:~# amixer set "DVC Out" 1%
```

3. Run the following command.

```
root@salvator-x:~# modprobe -a mmngr mmngrbuf vspm vspm_if vsp2 uvcs_drv
```

4. Run the following command.

```
root@salvator-x:~# export XDG_RUNTIME_DIR=/run/user/root
```

5. Run the following command.

```
root@salvator-x:~# #gst-launch-1.0 filesrc
location=movie/big_buck_bunny_720p_h264.mp4 ! qtdemux ! h264parse !
omxh264dec name=v ! fakesink sync=false -rp v:src
```

After finishing a command, you will see the log like below.

Red square is a result.

```
Got EOS from element "pipeline0".
Execution ended after 0:01:33.271932153
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Total time: 93.271935 seconds
Frames: 14315 processed
Avg. FPS: 153.48
Freeing pipeline ...
```

6. Run the following command.

```
root@salvator-x:~# gst-launch-1.0 filesrc
location=movie/big_buck_bunny_720p_h264_60fps.mp4 ! qtdemux ! h264parse !
omxh264dec name=v ! fakesink sync=false -rp v:src
```

After finishing a command, you will see the log like below.

Red square is a result.

```
Got EOS from element "pipeline0".
Execution ended after 0:02:40.763217348
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Total time: 160.763229 seconds
Frames: 35790 processed
Avg. FPS: 222.63
Freeing pipeline ...
```

(4) Result

Table 5-35: Result

virtualization PoC (Type1)	Value [FPS]
big_buck_bunny_720p_h264.mp4	153.48
big_buck_bunny_720p_h264_60fps.mp4	222.63

(5) Consideration

This result is expected.

5.8. Camera Performance

5.8.1. FPS on Linux of camera

(1) Description

Measure the video display performance (Frame per second) of Center Information when using Back Monitor on virtualization PoC.

(2) Precondition

- Measure on virtualization PoC (Type1)
- Incorporate a mechanism to measure the FPS (Frame per second) at Back Monitor.
- Connect a Camera to composite (CVBS_IN: CN21)
- Verified 10 minutes and use the average of FPS as the result value.

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command.

```
root@salvator-x:~# modprobe -a mmngr mmngrbuf vspm vspm_if vsp2 uvcs_drv
```

3. Run the following command.

```
root@salvator-x:~# media-ctl -d /dev/media4 -e "vspi0_vsp2@0 rpf.0 input"
```

4. Run the following command.

```
root@salvator-x:~# media-ctl -d /dev/media4 -e "vspi0_vsp2@0 wpf.0 output"
```

5. Run the following command.

```
root@salvator-x:~# export XDG_RUNTIME_DIR=/run/user/root
```

6. Run the following command.

```
root@salvator-x:~# gst-launch-1.0 --padprobe v:sink --timer v4l2src
device=/dev/video0 io-mode=dmabuf ! video/x-raw, format=RGB16, interlace-
mode=interleaved ! queue ! vspfilter ! video/x-raw, format=BGRA ! waylandsink
name=v
```

Take the log for 10 minutes, you will see the log like below.

Red square is a result.

```
FPS:  30  TIME 00:00:45
FPS:  30  TIME 00:00:46
FPS:  30  TIME 00:00:47
FPS:  30  TIME 00:00:48
 :
 :
```

(4) Result

Table 5-36: Result

Test environment	FPS
Virtualization PoC (Type1)	29.94

(5) Consideration

This result is expected.

5.8.2. FPS on INTEGRITY of camera

(1) Description

Measure the video display performance (Frame per second) of Head-up display when using Back Monitor on virtualization PoC.

(2) Precondition

- Measure on virtualization PoC (Type1)
- Incorporate a mechanism to measure the fps at Back Monitor.
- Verified 10 minutes and use the average of FPS as the result value.

(3) How to measure

1. Launch Type1 and take a log for 10 minutes.

You will see the log like below. Red square is a result.

```
[VIN]  30 FPS [70568280]
[VIN]  30 FPS [71567723]
LOG: fps: 59.980335
```

(4) Result

Table 5-37: Result

Test environment	FPS
Virtualization PoC (Type1)	29.97

(5) Consideration

This result is expected.

5.8.3. The overhead (FPS) compared virtualized Linux with native Linux

(1) Description

Compare the video display performance (Frame per second) of Center Information when using Back monitor on virtualized Linux and native Linux.

(2) Precondition

- Measure on virtualized Linux and native Linux (Type4 and Type2)
  *Both types stop the function of Video/Audio playback app. Refer to 5.7.1 Precondition.
- Measurement method is same as 5.8.1.

(3) How to measure

- Type4: Measurement method refers to 5.8.1.
- Type2

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command.

```
root@salvator-x:~# modprobe -a mmngr mmngrbuf vspm vspm_if vsp2 uvcs_drv
```

3. Run the following command.

```
root@salvator-x:~# media-ctl -d /dev/media4 -e "vspi0_vsp2@0 rpf.0 input"
```

4. Run the following command.

```
root@salvator-x:~# media-ctl -d /dev/media4 -e "vspi0_vsp2@0 wpf.0 output"
```

5. Run the following command.

```
root@salvator-x:~# export XDG_RUNTIME_DIR=/run/user/root
```

6. Run the following command.

```
root@salvator-x:~# gst-launch-1.0 --padprobe v:sink --timer v4l2src
device=/dev/video1 io-mode=dmabuf ! video/x-raw, format=RGB16, interlace-
mode=interleaved ! queue ! vspfilter ! video/x-raw, format=BGRA ! waylandsink
name=v
```

Take the log for 10 minutes, you will see the log like below. Red square is a result.

```
FPS:  30  TIME 00:00:45
FPS:  30  TIME 00:00:46
FPS:  30  TIME 00:00:47
FPS:  30  TIME 00:00:48
 :
 :
```

(4) Result

Table 5-38: Result

Test environment	FPS	Performance difference: ((B – A) /A)*100
(A) Native Linux (Type2)	29.95	0.00%
(B) Virtualized Linux (Type4)	29.95	

(5) Consideration

This result is expected.

5.9. Display Performance

5.9.1. OpenGL Performance

This and Section 5.6.1 and 5.6.2 are duplicated. Refer to Section 5.6.1 and 5.6.2.

5.9.2. Total performance of display system

(1) Description

Measure Composite performance of display on virtualization PoC.

(2) Precondition

- Measure on virtualization PoC (Type1)

(3) How to measure

1. Check that there are no noise in display for 10 minutes.

(4) Result

No noise in 3 displays (Center Information, Instrument Cluster and Head-up display) for 10 minutes.

(5) Consideration

This result is expected.

5.9.3. The overhead (processing time) of display virtualization

(1) Description

Measure the paint processing time of display on virtualized Linux and native Linux .

(2) Precondition

- Measure on virtualized Linux and native Linux(Type4 and Type2)
- Use Video / Audio playback app only.
- Use JSON-timeline on Weston
- Verified 10 minutes and use the average, minimum and maximum value as the result value.
- Use a Keyboard connected to Salvator-X

(3) How to measure

1. Enter the following key from the keyboard connected to Salvator-X for starting record a log.
   Press [Super(windows key) + Shift + Space + T]
2. Run the Salvator-X
3. Enter the following key from the keyboard connected to Salvator-X
   Press [Super(windows key) + Shift + Space + T] for stopping record a log.
4. Output a Timeline log (weston-timeline - *. Log) on root folder (/)
   After finishing a command, you will see the log like below.
   Red square is results.

```
{ "id":1, "type":"weston_output", "name":"HDMI-A-1" }
{ "T":[99, 620759003], "N":"core_repaint_finished", "wo":1, "vblank":[99, 620699000] }
{ "T":[99, 629974763], "N":"core_repaint_begin", "wo":1 }
{ "id":2, "type":"weston_surface", "desc":"top-level window 'nng_navi_win_0'" }
{ "T":[99, 630139403], "N":"core_flush_damage", "ws":2, "wo":1 }
{ "id":3, "type":"weston_surface", "desc":"top-level window 'IMG MDK test'" }
{ "T":[99, 630174903], "N":"core_flush_damage", "ws":3, "wo":1 }
{ "T":[99, 633777563], "N":"core_repaint_posted", "wo":1 }
```

5. Check the processing time of display as follows. Calculate the average and find minimum and maximum value.
   Processing time (ms) = (core_repaint_posted value) – (core_repaint_begin value) / 1000000
   Overhead = (Virtualized Linux) - (Native Linux)

(4) Result

Table 5-39: Result

	Processing time (ms)	
	Ave.	**Max.**
Virtualized Linux (Type4)	3.56	32.154 *Greater than 3.251 times: 12/14225 times
Native Linux(Type2)	1.824	3.251

Table 5-40: Overhead

	Virtualized Linux - Native Linux	
	Ave.	Max.
Processing time (ms)	1.74	28.9

(5) Consideration

We expected the same time. As this performance is not expected. Further investigation will be required to identify the problem.

5.9.4. Image composition performance

(1) Description

Measure underflow in the display driver on virtualization PoC. As a result, confirm that display processing is within 1V.

(2) Precondition

- Measure on virtualization PoC (Type1)
- Use image composition performance measurement tool. (RENESAS original)

(3) How to measure

1. Launch Type1 45000 times

You will see the log like below with Linux. Red square is a result.

```
[   19.087542] @ vspd1 underflow count :600
[   22.874916] @ vspd1 underflow count :600
[   26.746795] @ vspd1 underflow count :600
[   30.570880] @ vspd1 underflow count :600
[   34.454215] @ vspd1 underflow count :600
        :
        :
```

Log of INTEGRITY OS like below. The result is the red square which is outputted as vspd2_underflow_count and vspd3_underflow_count.

```
[VIN] 30 FPS [74514595]
@ vspd2 underflow count :600
@ vspd3 underflow count :600
[VIN] 30 FPS [75514178]
LOG: fps: 60.000000
        :
        :
```

* This log outputs the number of times that processing image composition as usual whenever inspects 600 times

(4) Result

Table 5-41: Result

	Virtualization PoC (Type1)		
	Inspect Counts	Normally Counts	Under-flow Counts
vspd1 : Display of Center Information (HDMI0)	45000	45000	0
vspd2 : Display of Instrument Cluster (HDMI1)	45000	45000	0
vspd3 : Display of Head-up display (Analog-RGB)	45000	44998	2

(5) Consideration

This result means that the analog RGB output caused underflow, so the proper display operation is not guaranteed. In this kind of case, R-Car H3 SoC can improve the available display performance by modifying the internal bus arbitration algorism settings (Quality-of-Service; QoS).

5.10. RAM I/O Performance

5.10.1. RAM I/O Performance

(1) Description

Measure the RAM's Read/Write performance when use virtualized Linux, native Linux, and native INTEGRITY.

Measurement tool is lmbench.

(2) Precondition

- Measure on special native INTEGRITY for this measurement.
- Use porting lmbench for native INTEGRITY.
- Measure on virtualized Linux, native Linux, and native INTEGRITY (Type2, Type4 and Type3).
- Use lmbench for Linux.
- Verified 10 times and use the average as the result value.

(3) How to measure

- Type2, Type4 using lmbench for Linux.
  1. Login to Linux.

```
salvator-x login: root
```

  2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/lmbench
```

  3. Run the following command for measurement.

```
root@salvator-x:~/tools/lmbench# ./bw_file_rd 64M io_only /home/root/test.txt
```

  After finishing a command, you will see the log like below.

  Red square is results.

```
root@salvator-x:~/tools/lmbench# ./bw_file_rd 64M io_only /home/root/test.txt
64.00 1265.77
```

- Type3 using porting lmbench for native INTEGRITY
  1. Lunch Type3 for this measure.
  2. Select "Dynamic Download/INDRT Connection (rtserv2) for Device Tree" and press "Connect" button.
  3. Select "Run mode target"
  4. Right click on "create file"
  5. Select [Target] - [Load Module] - [Load Module...] from Menu bar.

6. Load the " bw_file_rd.ael " file included in the deliverables.

You will see the log like below on cmd-tab.

Red square is results.

```
Target: Loader: Application started
I/O: 1073.7418 MB in 0.0004 secs, 3012743.6139 MB/sec
```

(4) Result

Table 5-42: Type3 Result

	Native INTEGRITY [MB/sec]
Ave.	3011624.721
1	3012743.614
2	3007153.717
3	3005435.223
4	2990403.687
5	3041223.081
6	3008884.215
7	3008241.943
8	2988545.832
9	3036922.253
10	3016693.643

Table 5-43: Type2 Result

	Native Linux (Type2) [MB/sec]
Ave.	1432.17
1	1442.81
2	1431.45
3	1431.06
4	1435.49
5	1426.98
6	1422.98
7	1438.40
8	1423.77
9	1432.47
10	1436.30

Table 5-44: Type4 Result

	Virtualized Linux (Type4) [MB/sec]
Ave.	1270.79
1	1279.82
2	1265.77
3	1281.56
4	1268.56
5	1276.43
6	1272.09
7	1265.42
8	1270.12
9	1264.27
10	1263.82

(5) Consideration

The INTEGRITY's result is unnaturally large. As this benchmark program (bw_file_rd in lmbench) just issue file read command and not actually use the file data, possibly the actual file read operation can be omitted.

The result of the native Linux is 1432.17MB/sec, and the virtualized Linux is 1270.79MB/sec. As the row memory read bandwidth that are measured in 5.11.1 are 2345.02MB/sec and 2338.82MB/sec, the results are reasonable. The performance of virtualized Linux is 11.27% slower than native one. As the filesystem operation usually require many synchronization in the OS domain, it is assumed that relatively larger performance overhead will appear.

5.11. Memory Performance

5.11.1. Sequential reading performance

(1) Description

Measure the performance to read sequential blocks of memory on virtualized Linux and native Linux.
Measurement tool is lmbench.

(2) Precondition

- Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)
  *Both types stop the function of Linux App
- Use lmbench's lat_mem_rd command on terminal software.
- Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
- Verified 10 times and use the average as the result value.

(3) How to measure

- Memory read latency

2. Login to Linux.

```
salvator-x login: root
```

4. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/lmbench
```

5. Run the following command for measurement.

```
root@salvator-x:~/tools/lmbench# ./lat_mem_rd 64m 64
```

After finishing a command, you will see the log like below.
Red square is results.

```
root@salvator-x:~/tools/lmbench# ./lat_mem_rd 64m 64
"stride=64
0.00049 2.671
0.00098 2.671
0.00195 2.671
0.00293 2.671
0.00391 2.671
0.00586 2.671
0.00781 2.671
0.01172 2.671
0.01562 5.235
0.02344 4.483
0.03125 5.310
0.04688 6.111
0.06250 5.699
0.09375 6.090
0.12500 6.074
0.18750 6.317
0.25000 6.314
0.37500 6.312
0.50000 6.311
0.75000 6.310
1.00000 6.255
1.50000 6.458
2.00000 35.175
3.00000 39.063
4.00000 31.825
6.00000 31.588
8.00000 31.409
12.00000 31.255
16.00000 31.164
24.00000 31.096
32.00000 31.037
48.00000 30.996
64.00000 31.199
```

6. Run the step 3 process 10 seconds after the result is displayed.
   Repeat this 9 times.

- Measure Mbyte/s

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/lmbench
```

3. Run the following command for measurement.

```
root@salvator-x:~/tools/lmbench# ./bw_mem 64m rd
```

After finishing a command, you will see the log like below.

Red square is results.

```
root@salvator-x:~/tools/lmbench# ./bw_mem 64m rd
67.11 2346.46
```

4. Run the step 3 process 10 seconds after the result is displayed.

Repeat this 9 times.

(4) Result

Table 5-45: Memory read latency Result (Type4)

Array size (MB)	1	2	3	4	5	6	7	8	9	10
0.00049	2.67	2.67	2.67	2.68	2.67	2.67	2.67	2.67	2.67	2.67
0.00098	2.67	2.67	2.67	2.68	2.67	2.67	2.67	2.67	2.67	2.67
0.00195	2.67	2.67	2.67	2.68	2.67	2.67	2.67	2.67	2.67	2.67
0.00293	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67
0.00391	2.67	2.67	2.67	2.68	2.67	2.67	2.67	2.67	2.67	2.67
0.00586	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67
0.00781	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67
0.01172	2.67	2.67	2.67	2.68	2.67	2.67	2.67	2.67	2.67	2.67
0.01562	5.24	5.34	2.67	2.68	2.67	2.67	2.67	2.67	2.67	2.67
0.02344	4.48	2.67	4.42	4.38	2.67	4.52	4.50	2.67	2.67	2.67
0.03125	5.31	3.98	4.47	3.98	4.06	3.98	4.90	5.69	4.40	4.41
0.04688	6.11	5.80	5.02	5.01	5.22	5.84	5.80	5.26	5.80	4.43
0.06250	5.70	6.11	5.90	6.10	6.09	6.09	5.90	6.02	5.89	6.09
0.09375	6.09	6.07	6.09	6.07	6.07	6.07	5.50	6.07	6.07	6.08
0.12500	6.07	6.08	6.07	6.08	6.08	6.07	5.67	6.06	6.07	6.07
0.18750	6.32	6.33	6.32	6.33	6.32	6.32	6.32	6.31	6.33	6.32
0.25000	6.31	6.31	6.31	6.32	6.32	6.31	6.31	6.31	6.32	6.31
0.37500	6.31	6.31	6.31	6.31	6.31	6.31	6.31	6.31	6.31	6.31
0.50000	6.31	6.32	6.32	6.30	6.31	6.31	6.31	6.31	6.32	6.31
0.75000	6.31	6.31	6.31	6.11	6.31	6.31	6.31	6.31	6.31	6.31
1.00000	6.26	6.32	6.31	6.13	6.31	6.31	6.30	6.31	6.27	6.43
1.50000	6.46	12.61	11.93	7.09	8.19	13.29	9.64	10.34	9.71	8.16
2.00000	35.18	34.06	35.93	31.88	33.05	36.66	35.88	38.78	34.59	38.16
3.00000	39.06	37.51	42.61	37.27	43.93	43.61	38.87	40.44	39.81	41.78
4.00000	31.83	32.23	32.76	31.58	32.47	33.19	31.91	32.65	31.83	32.75
6.00000	31.59	31.98	32.18	31.36	32.03	32.45	31.65	32.12	31.73	32.14
8.00000	31.41	31.60	31.83	31.25	31.76	32.04	31.45	31.78	31.42	31.82
12.00000	31.26	31.49	31.54	31.26	31.52	31.66	31.27	31.49	31.26	31.50
16.00000	31.16	31.26	31.39	31.08	31.41	31.48	31.18	31.35	31.18	31.36
24.00000	31.10	31.15	31.21	31.03	31.28	31.30	31.10	31.20	31.08	31.21
32.00000	31.04	31.11	31.27	31.01	31.22	31.20	31.06	31.15	31.17	31.26
48.00000	31.00	31.03	31.09	30.98	31.16	31.11	31.01	31.07	31.00	31.07
64.00000	31.20	31.06	31.05	31.06	31.17	31.06	31.05	31.05	31.06	31.05

Table 5-46: Memory read latency Result (Type2)

Array size (MB)	1	2	3	4	5	6	7	8	9	10
0.00049	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.68	2.67	2.67
0.00098	2.67	2.67	2.68	2.67	2.67	2.67	2.67	2.68	2.67	2.67
0.00195	2.67	2.67	2.67	2.67	2.67	2.67	2.68	2.68	2.67	2.67
0.00293	2.67	2.67	2.68	2.68	2.67	2.67	2.67	2.67	2.67	2.67
0.00391	2.67	2.68	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67
0.00586	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.68	2.67	2.67
0.00781	2.67	2.67	2.68	2.67	2.67	2.68	2.68	2.68	2.67	2.67
0.01172	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67
0.01562	2.67	2.67	2.68	2.67	2.67	5.37	5.51	5.37	2.67	2.67
0.02344	2.67	2.67	4.97	4.45	2.67	2.68	4.45	2.68	4.51	4.41
0.03125	4.10	3.98	4.00	5.24	2.67	4.45	4.46	4.48	2.67	3.98
0.04688	6.11	5.57	5.58	5.81	5.54	5.83	6.13	6.13	4.95	5.58
0.06250	5.25	6.09	6.09	5.70	5.70	5.29	5.70	5.27	5.25	5.86
0.09375	6.09	6.07	6.09	6.07	6.09	6.07	6.09	6.11	5.80	6.06
0.12500	6.07	6.07	6.07	6.08	6.07	6.10	6.07	6.10	5.88	6.07
0.18750	6.31	6.31	6.31	6.32	6.31	6.31	6.32	6.31	6.31	6.31
0.25000	6.31	6.31	6.34	6.31	6.31	6.31	6.34	6.34	6.31	6.31
0.37500	6.31	6.33	6.31	6.31	6.31	6.33	6.31	6.31	6.31	6.31
0.50000	6.30	6.31	6.33	6.33	6.31	6.30	6.33	6.33	6.31	6.30
0.75000	6.23	6.31	6.15	6.31	6.31	6.30	6.31	6.33	6.30	6.30
1.00000	6.18	6.30	6.14	6.51	6.31	6.30	6.32	6.33	6.30	6.30
1.50000	6.04	6.91	7.47	15.30	12.11	22.10	7.40	7.28	9.00	10.78
2.00000	27.18	32.63	27.14	35.95	30.11	31.23	33.97	33.97	39.46	34.08
3.00000	41.69	43.86	38.90	45.89	41.06	46.38	40.83	47.37	46.87	44.68
4.00000	31.56	33.43	31.49	32.83	32.41	32.44	32.43	33.53	33.22	33.18
6.00000	31.37	32.41	31.61	32.15	31.95	31.93	31.78	32.30	32.33	32.33
8.00000	31.21	32.00	31.25	31.86	31.59	31.95	31.74	32.23	31.90	31.89
12.00000	31.04	31.63	31.11	31.52	31.29	31.43	31.21	31.59	31.50	31.48
16.00000	30.97	31.30	31.06	31.58	31.16	31.30	31.42	31.30	31.28	31.31
24.00000	30.89	31.12	31.01	31.21	31.02	31.37	31.10	31.23	31.11	31.12
32.00000	30.88	31.02	31.17	31.31	30.95	31.10	31.00	31.15	31.01	31.03
48.00000	30.81	30.95	30.95	30.93	30.88	31.21	30.98	31.04	30.92	30.94
64.00000	30.88	31.22	31.21	31.21	30.87	31.00	31.22	30.99	30.87	31.00

Table 5-47: Memory read latency Result

	Virtualized Linux (Type4) [ns]			Native Linux(Type2) [ns]		
Array size (MB)	**Ave.**	**Min.**	**Max.**	**Ave.**	**Min.**	**Max.**
0.00049	2.67	2.67	2.68	2.67	2.67	2.68
0.00098	2.67	2.67	2.68	2.67	2.67	2.68
0.00195	2.67	2.67	2.68	2.67	2.67	2.68
0.00293	2.67	2.67	2.67	2.67	2.67	2.68
0.00391	2.67	2.67	2.68	2.67	2.67	2.68
0.00586	2.67	2.67	2.67	2.67	2.67	2.68
0.00781	2.67	2.67	2.67	2.67	2.67	2.68
0.01172	2.67	2.67	2.68	2.67	2.67	2.67
0.01562	3.20	2.67	5.34	3.49	2.67	5.51
0.02344	3.57	2.67	4.52	3.62	2.67	4.97
0.03125	4.52	3.98	5.69	4.00	2.67	5.24
0.04688	5.43	4.43	6.11	5.72	4.95	6.13
0.06250	5.99	5.70	6.11	5.62	5.25	6.09
0.09375	6.02	5.50	6.09	6.05	5.80	6.11
0.12500	6.03	5.67	6.08	6.06	5.88	6.10
0.18750	6.32	6.31	6.33	6.31	6.31	6.32
0.25000	6.31	6.31	6.32	6.32	6.31	6.34
0.37500	6.31	6.31	6.31	6.31	6.31	6.33
0.50000	6.31	6.30	6.32	6.32	6.30	6.33
0.75000	6.29	6.11	6.31	6.28	6.15	6.33
1.00000	6.29	6.13	6.43	6.30	6.14	6.51
1.50000	9.74	6.46	13.29	10.44	6.04	22.10
2.00000	35.42	31.88	38.78	32.57	27.14	39.46
3.00000	40.49	37.27	43.93	43.75	38.90	47.37
4.00000	32.32	31.58	33.19	32.65	31.49	33.53
6.00000	31.92	31.36	32.45	32.02	31.37	32.41
8.00000	31.64	31.25	32.04	31.76	31.21	32.23
12.00000	31.42	31.26	31.66	31.38	31.04	31.63
16.00000	31.29	31.08	31.48	31.27	30.97	31.58
24.00000	31.17	31.03	31.30	31.12	30.89	31.37
32.00000	31.15	31.01	31.27	31.06	30.88	31.31
48.00000	31.05	30.98	31.16	30.96	30.81	31.21
64.00000	31.08	31.05	31.20	31.05	30.87	31.22

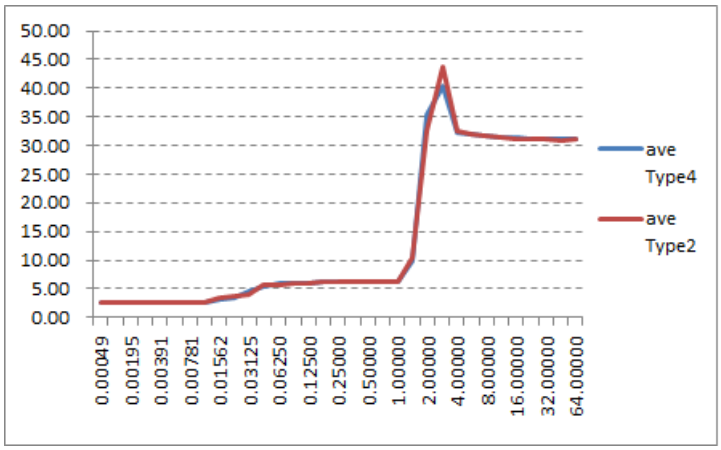


Figure 5-13: Average Result Linux and Native Linux (Type4 and Type2)

Table 5-48: Data Transmission rate (Mbyte/s)

Test environment	Ave.	1	2	3	4	5	6	7	8	9	10
Virtualized Linux (Type4)	2338.82	2346.46	2338.45	2333.09	2334.79	2346.46	2336.09	2339.27	2338.21	2345.56	2329.85
Native Linux (Type2)	2345.02	2332.60	2347.61	2357.18	2332.68	2356.93	2332.44	2348.19	2348.35	2347.53	2346.71

(5) Consideration

The result of "lat_mem_rd" reflects well about the memory system hierarchy of the target hardware.

Both Native Linux and Virtualized Linux has the similar form which has the 3 specific score groups.

The first one is around 2.7ns and mainly observed under 32kB test size.

In these cases, the read access of this test is cached in the main CPU's L1 data cache (ARM Cortex-A57: 32kB L1 data cache per core), so this reflects the L1 cache read performance.

The next one is about 6.0ns to 6.3ns and mainly observed between 32kB to 1.5MB.

In these cases, the read access size exceeds the L1 data cache size and causes L1 data cache miss, but can be cached in the 2MB L2 unified cache, so this reflects the L2 cache read performance.

And the last one is around 31ns and mainly observed in 2MB or more.

In this size, the read access of this test exceeds the L2 cache size and causes L2 cache miss every time once the L2 cache is fully filled by the test read.

As L2 cache miss takes cache line fetch from the main memory (LPDDR4 memory on R-Car H3 SiP), this reflects the LPDDR4 read performance.

It is observed that the latency time of virtualized Linux increases a little earlier than native Linux when the test size increases. For this behavior, we are estimating that this is effected by the program working set size difference of the operating system portion. The virtualized Linux requires OS kernel feature not only for Linux but also for INTEGRITY kernel, and consumes more cache lines for operation. When the test size is nearly equal to either L1 or L2 cachesize, it increases the virtualized Linux's latency time.

5.11.2. Sequential writing performance

(1) Description

Measure the performance to write sequential blocks of memory on virtualized Linux and native Linux.

Measurement tool is lmbench.

(2) Precondition

- Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)
  *Both types stop the function of Linux App
- Use lmbench's lat_mem_wr command on terminal software.
- Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/lmbench
```

3. Run the following command for measurement.

```
root@salvator-x:~# cd tools/lmbench
root@salvator-x:~/tools/lmbench# ./bw_mem 1m wr
root@salvator-x:~/tools/lmbench# ./bw_mem 2m wr
root@salvator-x:~/tools/lmbench# ./bw_mem 4m wr
root@salvator-x:~/tools/lmbench# ./bw_mem 8m wr
root@salvator-x:~/tools/lmbench# ./bw_mem 16m wr
root@salvator-x:~/tools/lmbench# ./bw_mem 32m wr
root@salvator-x:~/tools/lmbench# ./bw_mem 64m wr
```

After finishing a command, you will see the log like below.

Red square is results.

```
root@salvator-x:~/tools/lmbench# ./bw_mem 1m wr
1.05 13676.19
```

4. Run the step 3 process 10 seconds after the result is displayed.

Repeat this 9 times.

(4) Result

Table 5-49: Result (Type4)

Test size	1	2	3	4	5	6	7	8	9	10
1M	13609.71	13669.76	13666.27	13660.88	13666.27	13672.12	13570.52	13627.70	13664.70	13664.70
2M	3913.92	4018.93	4286.26	4491.49	4316.61	4146.06	4414.28	4215.77	4917.56	3799.19
4M	2211.02	2202.89	2221.95	2209.47	2209.08	2212.19	2215.30	2216.47	2195.59	2203.66
8M	1952.20	1959.50	1962.25	1952.88	1952.20	1950.16	1958.58	1958.35	1950.16	1953.79
16M	1938.89	1943.61	1946.54	1939.56	1942.26	1942.93	1944.73	1937.55	1940.01	1938.44
32M	1940.80	1928.75	1940.68	1939.45	1947.44	1945.64	1946.76	1940.35	1948.91	1947.10
64M	1944.56	1940.80	1948.06	1946.93	1947.10	1941.70	1938.67	1940.80	1940.96	1940.85

Table 5-50: Result (Type2)

Test size	1	2	3	4	5	6	7	8	9	10
1M	13676.19	13633.49	13676.19	13631.28	13680.58	13674.38	13671.91	13670.09	13609.71	13699.20
2M	4759.94	5047.76	5493.01	5387.18	4085.12	5110.21	4948.81	4822.74	4979.54	4768.96
4M	2237.36	2236.56	2221.17	2248.55	2220.38	2235.37	2218.03	2234.98	2231.01	2231.80
8M	1971.01	1966.39	1951.97	1967.77	1969.85	1964.09	1975.42	1972.40	1947.44	1968.46
16M	1949.71	1952.66	1952.20	1951.75	1953.34	1952.43	1957.44	1952.20	1951.97	1953.11
32M	1955.04	1935.65	1950.05	1955.27	1951.52	1954.13	1953.00	1954.82	1935.87	1936.09
64M	1937.66	1951.92	1950.67	1952.31	1952.37	1952.03	1952.20	1955.04	1936.20	1953.96

Table 5-51: Result

Test size	Virtualized Linux (Type4) [ns]			Native Linux(Type2) [ns]		
	Ave.	**Min.**	**Max.**	**Ave.**	**Min.**	**Max.**
1M	13647.26	13570.52	13672.12	13662.30	13609.71	13699.20
2M	4252.01	3799.19	4917.56	4940.33	4085.12	5493.01
4M	2209.76	2195.59	2221.95	2231.52	2218.03	2248.55
8M	1955.01	1950.16	1962.25	1965.48	1947.44	1975.42
16M	1941.45	1937.55	1946.54	1952.68	1949.71	1957.44
32M	1942.59	1928.75	1948.91	1948.14	1935.65	1955.27
64M	1943.04	1938.67	1948.06	1949.44	1936.20	1955.04

(5) Consideration

As lmbench doesn’t has the benchmark for the memory write latency, this time we used memory bandwidth benchmark instead. As the cache line length is 64 bytes, the bandwidth of 1938.67MB/sec (64MB on Type4) and 1936.20MB/sec (64MB on Type2) are corresponding to 33.012ns and 33.054ns of the latency time. They are a little larger than read latency, and this is reasonable because the write cache miss initially issues memory read to fill the cache line, and replace the specific position of the cache line by the store data.

5.11.3. Random reading performance

(1) Description

Measure the performance to read random blocks of memory on virtualized Linux and native Linux.

Measurement tool is lmbench.

(2) Precondition

- Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)
  *Both types stop the function of Linux App
- Use lmbench's lat_mem_rd command on terminal software.
- Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/lmbench
```

3. Run the following command for measurement.

```
root@salvator-x:~/tools/lmbench# ./lat_mem_rd 64m 131072
```

After finishing a command, you will see the log like below.

Red square is results.

```
root@salvator-x:~/tools/lmbench# ./lat_mem_rd 64m 131072
"stride=131072
0.12500 2.669
0.18750 2.669
0.25000 2.669
0.37500 2.669
0.50000 7.675
0.75000 11.827
1.00000 13.012
1.50000 14.905
2.00000 15.180
3.00000 85.090
4.00000 124.127
6.00000 150.806
8.00000 161.898
12.00000 174.101
16.00000 183.324
24.00000 203.268
32.00000 219.112
48.00000 316.886
64.00000 324.883
```

4. Run the step 3 process 10 seconds after the result is displayed.
   Repeat this 9 times.

(4) Result

Table 5-52: Result (Type4)

Array size (MB)	1	2	3	4	5.	6	7	8	9	10
0.1250	2.678	2.677	2.678	2.671	2.678	2.678	2.678	2.674	2.674	2.677
0.1875	2.677	2.675	2.671	2.671	2.674	2.675	2.675	2.675	2.675	2.674
0.2500	2.675	2.671	2.674	2.671	2.671	2.671	2.671	2.671	2.675	2.674
0.3750	2.671	7.589	2.671	2.671	2.675	2.675	2.675	2.671	2.671	2.675
0.5000	2.675	7.681	2.675	2.67	14.126	2.675	2.671	2.671	2.674	2.671
0.7500	2.671	7.635	2.675	2.671	14.154	2.671	2.675	2.674	2.671	14.045
1.0000	2.675	7.261	9.692	7.154	13.04	6.062	9.925	7.154	10.872	9.148
1.5000	11.053	10.047	10.351	14.08	13.86	12.149	11.982	12.132	11.033	12.133
2.0000	13.015	12.4	11.603	13.443	15.215	13.109	12.065	12.746	13.71	13.249
3.0000	14.612	13.876	25.972	22.967	100.178	13.767	51.394	13.765	14.156	53.043
4.0000	14.942	33.825	87.631	86.079	126.024	50.151	103.167	42.234	17.947	103.829
6.0000	95.295	104.702	127.579	129.772	150.442	103.938	134.868	103.974	95.932	138.147
8.0000	124.612	127.143	143.325	144.049	159.85	130.508	148.273	129.189	122.214	151.129
12.0000	145.637	145.644	157.804	157.374	169.459	147.583	163.278	149.143	145.502	165.212
16.0000	158.973	157.879	167.848	167.535	176.399	160.876	171.76	160.594	156.41	172.546
24.0000	174.125	173.915	180.59	179.913	186.548	175.466	181.717	174.025	173.975	183.141
32.0000	189.616	190.056	193.498	194.073	199.172	191.493	195.995	190.194	189.508	196.876
48.0000	201.914	203.956	211.815	210.206	218.474	204.41	212.537	203.416	202.865	213.208
64.0000	259.192	260.523	260.651	258.855	260.39	260.431	258.424	260.634	256.988	259.821

Table 5-53: Result (Type2)

Array size (MB)	1	2	3	4	5.	6	7	8	9	10
0.1250	2.685	2.669	2.685	2.669	2.669	2.685	2.685	2.669	2.685	2.684
0.1875	2.685	2.685	2.685	2.669	2.669	2.685	2.669	2.669	2.669	2.669
0.2500	2.685	2.669	2.669	2.669	2.669	2.685	2.685	2.669	2.685	2.685
0.3750	2.685	2.685	2.685	2.669	2.669	7.717	2.669	2.669	2.685	2.685
0.5000	2.685	7.673	2.669	2.669	2.669	14.89	2.685	2.669	7.673	2.685
0.7500	7.618	14.457	10.907	2.669	7.618	16.108	2.685	7.618	12.976	5.07
1.0000	6.084	14.934	9.925	11.321	10.426	16.108	10.571	14.513	12.836	2.685
1.5000	12.585	15.513	13.424	11.566	11.176	15.771	11.232	11.232	13.759	9.731
2.0000	13.047	15.395	12.124	14.013	12.972	15.86	13.089	12.805	14.346	12.459
3.0000	14.846	94.259	40.594	14.599	14.596	112.466	14.934	14.68	73.577	14.431
4.0000	50.366	128.609	95.775	24.694	24.349	135.323	50.15	49.882	116.946	72.706
6.0000	112.184	149.779	132.711	98.118	98.052	153.704	110.25	102.457	142.839	117.665
8.0000	133.017	156.419	144.157	123.454	121.296	159.388	131.707	127.372	151.527	134.726
12.0000	146.128	162.947	154.907	139.549	139.507	166.064	145.149	143.174	160.646	148.956
16.0000	154.18	167.999	161.478	150.347	149.064	169.785	154.012	154.19	165.46	156.009
24.0000	163.744	172.576	169.516	160.806	160.214	173.623	163.827	163.545	171.947	166.264
32.0000	171.928	179.99	175.803	169.935	171.108	180.781	171.61	170.506	179.332	174.417
48.0000	182.319	187.053	185.623	180.483	181.039	188.694	182.848	181.476	187.615	184.219
64.0000	219.294	221.76	219.582	221.968	219.059	227.985	219.693	220.078	221.206	221.885

Table 5-54: Result

	Virtualized Linux (Type4) [ns]			Native Linux(Type2) [ns]		
Array size (MB)	**Ave.**	**Min.**	**Max.**	**Ave.**	**Min.**	**Max.**
0.1250	2.68	2.67	2.68	2.68	2.67	2.69
0.1875	2.67	2.67	2.68	2.68	2.67	2.69
0.2500	2.67	2.67	2.68	2.68	2.67	2.69
0.3750	3.16	2.67	7.59	3.18	2.67	7.72
0.5000	4.32	2.67	14.13	4.90	2.67	14.89
0.7500	5.45	2.67	14.15	8.77	2.67	16.11
1.0000	8.30	2.68	13.04	10.94	2.69	16.11
1.5000	11.88	10.05	14.08	12.60	9.73	15.77
2.0000	13.06	11.60	15.22	13.61	12.12	15.86
3.0000	32.37	13.77	100.18	40.90	14.43	112.47
4.0000	66.58	14.94	126.02	74.88	24.35	135.32
6.0000	118.46	95.30	150.44	121.78	98.05	153.70
8.0000	138.03	122.21	159.85	138.31	121.30	159.39
12.0000	154.66	145.50	169.46	150.70	139.51	166.06
16.0000	165.08	156.41	176.40	158.25	149.06	169.79
24.0000	178.34	173.92	186.55	166.61	160.21	173.62
32.0000	193.05	189.51	199.17	174.54	169.94	180.78
48.0000	208.28	201.91	218.47	184.14	180.48	188.69
64.0000	259.59	256.99	260.65	221.25	219.06	227.99

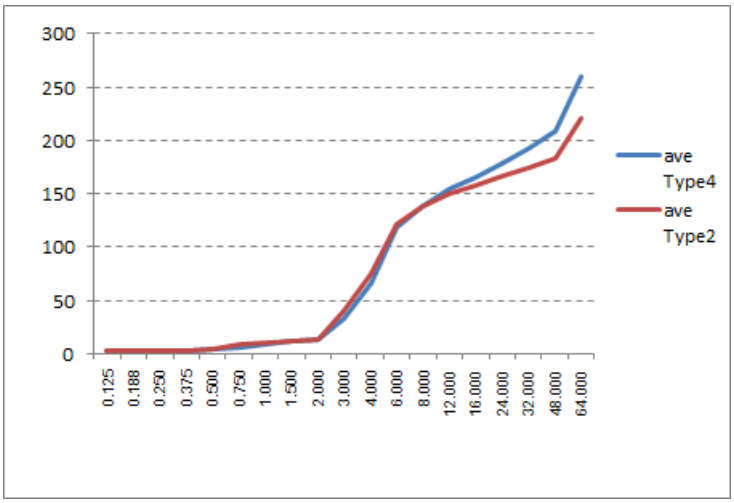


Figure 5-14: Average Result Linux and Native Linux (Type4 and Type2)

(5) Consideration

As it is not easy to perform 'random' access performance to the memory, this test simulates the random access condition by mostly-cache-miss address condition. In R-Car H3, the main CPU (ARM Cortex-A57) has 2MB unified L2C with 16-way and 64-byte line length. This means when the cacheable memory access is issued in every top boundary of 128kB, the top line of all 16 ways of cache arrays are consumed after 16th memory access, and all of the following memory access causes cache miss.

It is observed that when the test size is smaller than 8MB, the result is similar, or virtualized Linux is faster than native Linux. The faster virtualized Linux is currently unexpected and require further investigation.

In contrast, when the test size is larger than 8MB, the virtualized Linux is meaningfully slower than native Linux. As the TLB miss (both L1 data TLB and L2 unified TLB miss) occurs when the test size is larger than 4MB, the performance difference is expected to be produced by the difference of TLB miss processing. When in the native Linux environment, TLB miss causes the translation table walk once, but in virtualized Linux it causes twice due to the 2-stage address translation. See 5.11.7 and 5.11.8 for detail.

5.11.4. Random writing performance

(1) Description

Measure the performance to write random blocks of memory on virtualized Linux and native Linux.

Measurement tool is Renesas original test program.

(2) Precondition

- Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)
  *Both types stop the function of Linux App
- Use Renesas original test program. (random_write)
- Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools
```

3. Run the following command to measure the Random writing performance 10 times.

```
root@salvator-x:~/tools# ./random_write
```

After finishing a command, you will see the log like below.

Red square is results.

```
root@salvator-x:~/tools# ./random_write
alloc_size[byte], count, count_max, start[sec], start[nsec], end[sec], end[nsec]
2097152, 0, 10, 184, 162289764, 184, 162338604, 48840
2097152, 1, 10, 184, 162434604, 184, 162472524, 37920
2097152, 2, 10, 184, 162481884, 184, 162482364, 480
2097152, 3, 10, 184, 162488124, 184, 162488604, 480
2097152, 4, 10, 184, 162494124, 184, 162494484, 360
2097152, 5, 10, 184, 162499884, 184, 162500364, 480
2097152, 6, 10, 184, 162505884, 184, 162506244, 360
2097152, 7, 10, 184, 162511524, 184, 162511884, 360
2097152, 8, 10, 184, 162517164, 184, 162517644, 480
2097152, 9, 10, 184, 162522804, 184, 162523164, 360
4194304, 0, 10, 184, 162541884, 184, 163453644, 911760
4194304, 1, 10, 184, 163559004, 184, 163601364, 42360
4194304, 2, 10, 184, 163608804, 184, 163612524, 3720
4194304, 3, 10, 184, 163618404, 184, 163621884, 3480
4194304, 4, 10, 184, 163627284, 184, 163628364, 1080
4194304, 5, 10, 184, 163633524, 184, 163634364, 840
4194304, 6, 10, 184, 163639644, 184, 163640364, 720
4194304, 7, 10, 184, 163645764, 184, 163646484, 720
4194304, 8, 10, 184, 163651644, 184, 163652364, 720
4194304, 9, 10, 184, 163657644, 184, 163658364, 720
8388608, 0, 10, 184, 163677084, 184, 165532045, 1854961
8388608, 1, 10, 184, 165683485, 184, 166216525, 533040
8388608, 2, 10, 184, 166231525, 184, 166237165, 5640
8388608, 3, 10, 184, 166243045, 184, 166246165, 3120
8388608, 4, 10, 184, 166251685, 184, 166254325, 2640
8388608, 5, 10, 184, 166259605, 184, 166262245, 2640
8388608, 6, 10, 184, 166267765, 184, 166270405, 2640
8388608, 7, 10, 184, 166275805, 184, 166278325, 2520
8388608, 8, 10, 184, 166283965, 184, 166286245, 2280
8388608, 9, 10, 184, 166292125, 184, 166294765, 2640
16777216, 0, 10, 184, 166320925, 184, 172131925, 5811000
16777216, 1, 10, 184, 172387765, 184, 174471565, 2083800
16777216, 2, 10, 184, 174489325, 184, 174509965, 20640
16777216, 3, 10, 184, 174516085, 184, 174525325, 9240
16777216, 4, 10, 184, 174531085, 184, 174539605, 8520
16777216, 5, 10, 184, 174545245, 184, 174553525, 8280
16777216, 6, 10, 184, 174559165, 184, 174566965, 7800
16777216, 7, 10, 184, 174572605, 184, 174581485, 8880
16777216, 8, 10, 184, 174587005, 184, 174595645, 8640
16777216, 9, 10, 184, 174601045, 184, 174608845, 7800
33554432, 0, 10, 184, 174636325, 184, 185291005, 10654680
33554432, 1, 10, 184, 185696725, 184, 197594606, 11897881
33554432, 2, 10, 184, 198046166, 184, 207793166, 9747000
33554432, 3, 10, 184, 208227446, 184, 221096247, 12868801
33554432, 4, 10, 184, 221500527, 184, 233233767, 11733240
33554432, 5, 10, 184, 233638287, 184, 244573288, 10935001
33554432, 6, 10, 184, 245018488, 184, 256515928, 11497440
33554432, 7, 10, 184, 256938688, 184, 268807409, 11868721
33554432, 8, 10, 184, 269193449, 184, 279993569, 10800120
33554432, 9, 10, 184, 280422569, 184, 290886090, 10463521
50331648, 0, 10, 184, 291260970, 184, 308172330, 16911360
50331648, 1, 10, 184, 308732370, 184, 325567291, 16834921
50331648, 2, 10, 184, 326156611, 184, 342395252, 16238641
50331648, 3, 10, 184, 343343972, 184, 359209892, 15865920
50331648, 4, 10, 184, 360752372, 184, 378412053, 17659681
50331648, 5, 10, 184, 378997413, 184, 395120494, 16123081
50331648, 6, 10, 184, 395674654, 184, 411530734, 15856080
50331648, 7, 10, 184, 412315774, 184, 429724175, 17408401
50331648, 8, 10, 184, 430367855, 184, 448244856, 17877001
50331648, 9, 10, 184, 448877616, 184, 465900817, 17023201
67108864, 0, 10, 184, 466462537, 184, 489351938, 22889401
67108864, 1, 10, 184, 491128778, 184, 514392819, 23264041
67108864, 2, 10, 184, 515150979, 184, 538065940, 22914961
67108864, 3, 10, 184, 538811260, 184, 561737380, 22926120
67108864, 4, 10, 184, 562498421, 184, 584818901, 22320480
67108864, 5, 10, 184, 585567341, 184, 607276542, 21709201
67108864, 6, 10, 184, 607982022, 184, 632766823, 24784801
67108864, 7, 10, 184, 633606943, 184, 657430784, 23823841
67108864, 8, 10, 184, 658176824, 184, 682881945, 24705121
67108864, 9, 10, 184, 683602905, 184, 706038106, 22435201
-- finished.
```

(4) Result

Table 5-55: Result (Type4)

Array size (MB)	1	2	3	4	5	6	7	8	9	10
2M	0.04236	0.02928	0.00072	0.00036	0.00036	0.00036	0.00036	0.00036	0.00036	0.00048
4M	0.65052	0.03360	0.00276	0.00120	0.00072	0.00072	0.00072	0.00072	0.00072	0.00072
8M	1.92516	0.60564	0.00420	0.00396	0.00216	0.00264	0.00204	0.00216	0.00228	0.00204
16M	4.38828	1.90884	0.01512	0.00708	0.00660	0.00756	0.00804	0.00780	0.00684	0.00684
32M	9.27384	8.73480	8.98452	9.02604	9.01296	9.14568	8.94012	9.05112	8.95584	8.90136
48M	13.67736	13.82160	13.53372	13.63812	13.56264	13.97652	14.03928	13.80252	13.91436	13.74948
64M	18.93768	17.70492	18.25560	17.74824	18.28896	17.91444	18.00984	17.75580	18.18672	18.36936

Table 5-56: Result (Type2)

Array size (MB)	1	2	3	4	5	6	7	8	9	10
2M	0.03756	0.00192	0.00060	0.00072	0.00036	0.00048	0.00036	0.00072	0.00060	0.00036
4M	0.65796	0.00444	0.00228	0.00228	0.00216	0.00228	0.00180	0.00204	0.00216	0.00228
8M	1.86924	0.47436	0.00768	0.00504	0.00528	0.00576	0.00480	0.00504	0.00552	0.00540
16M	4.37940	1.61028	0.02004	0.01044	0.01044	0.01044	0.01044	0.01044	0.01032	0.01044
32M	9.33072	8.56212	8.75352	8.93064	9.02640	8.97372	9.09624	9.02280	9.07680	8.67060
48M	13.89948	12.71736	13.64856	13.78860	13.88268	13.87536	13.67964	13.75200	13.82088	14.01768
64M	17.94108	17.47488	17.56656	18.25212	18.69636	18.13944	18.29844	18.31908	17.66844	18.49116

Table 5-57: Result

Array size (MB)	Virtualized Linux (Type4) [ns]			Native Linux (Type2) [ns]		
	Ave.	Min.	Max.	Ave.	Min.	Max.
2M	0.04236	0.00036	0.00750	0.03756	0.00036	0.004368
4M	0.65052	0.00072	0.06924	0.65796	0.0018	0.067968
8M	1.92516	0.00204	0.25523	1.86924	0.0048	0.238812
16M	4.38828	0.00660	0.63630	4.3794	0.01032	0.608268
32M	9.27384	8.73480	9.00263	9.33072	8.562121	8.9443564
48M	14.03928	13.53372	13.77156	14.01768	12.717361	13.7082245
64M	18.93768	17.70492	18.11716	18.696361	17.474881	18.0847568

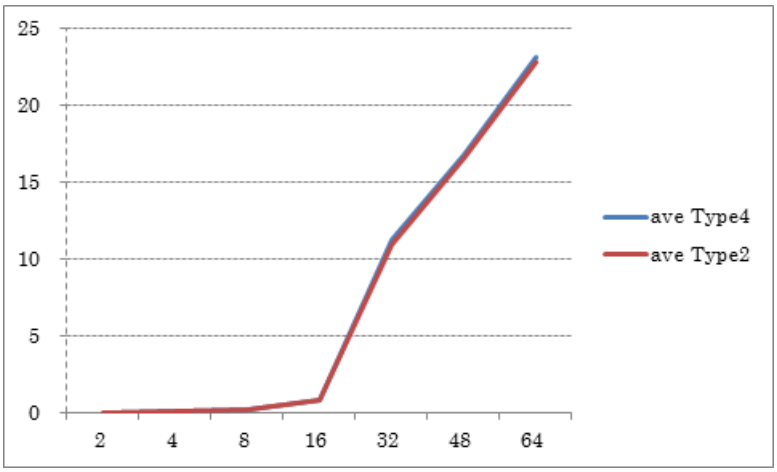


Figure 5-15: Average Result Virtualized Linux and Native Linux (Type4 and Type2)

(5) Consideration

As same as 5.11.3, this test also simulates the random access by using specific address condition to force the cache line conflict.
Generally, the test result (ms) is in proportional when the test size is larger than 8MB. The result for less or equal 8MB are vary. We are estimating that if the test size is smaller or equal to 8MB (= only 64 addresses for test), the store request will possibly merged on the store buffer and reduces the write requests for the LPDDR4 main memory. Unlike the random read case, the performance of the native Linux and virtualized Linux are observed almost same in the all test size. The possible reason is estimated that the store request can be buffered, and the TLB miss latency can be hidden in the preceding cache miss processing time.

5.11.5. Memory Allocate/Deallocate performance

(1) Description

Measure the performance to allocate blocks of memory and deallocate the same blocks on virtualized Linux and native Linux.

Measurement tool is Renesas original test program.

(2) Precondition

- Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)
  *Both types stop the function of Linux App
- Use Renesas original test program. (alloc_dealloc)
- Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools
```

3. Run the following command to measure the Memory Allocate/Deallocate performance 10 times.

```
root@salvator-x:~/tools# ./alloc_dealloc
```

After finishing a command, you will see the log like below.

Red square is results.

```
root@salvator-x:~/tools# ./alloc_dealloc
alloc_size[byte], count, count_max, start[sec], start[nsec], end[sec], end[nsec], diff[nsec]
2097152, 0, 10, 119, 675161486, 119, 676956686, 1795200
2097152, 1, 10, 119, 676999046, 119, 678334646, 1335600
2097152, 2, 10, 119, 678348446, 119, 678367286, 18840
2097152, 3, 10, 119, 678373286, 119, 678384326, 11040
2097152, 4, 10, 119, 678389606, 119, 678399326, 9720
2097152, 5, 10, 119, 678404606, 119, 678414686, 10080
2097152, 6, 10, 119, 678420686, 119, 678430526, 9840
2097152, 7, 10, 119, 678435806, 119, 678445166, 9360
2097152, 8, 10, 119, 678450446, 119, 678459806, 9360
2097152, 9, 10, 119, 678465206, 119, 678474806, 9600
4194304, 0, 10, 119, 678480566, 119, 680846246, 2365680
4194304, 1, 10, 119, 680868446, 119, 683235326, 2366880
4194304, 2, 10, 119, 683251166, 119, 683314886, 63720
4194304, 3, 10, 119, 683321006, 119, 683380886, 59880
4194304, 4, 10, 119, 683388326, 119, 683445086, 56760
4194304, 5, 10, 119, 683452046, 119, 683507726, 55680
4194304, 6, 10, 119, 683513966, 119, 683566406, 52440
4194304, 7, 10, 119, 683573006, 119, 683624366, 51360
4194304, 8, 10, 119, 683630486, 119, 683684486, 54000
4194304, 9, 10, 119, 683691326, 119, 683745086, 53760
8388608, 0, 10, 119, 683751566, 119, 688888647, 5137081
8388608, 1, 10, 119, 688919367, 119, 691322367, 2403000
8388608, 2, 10, 119, 691341807, 119, 691468287, 126480
8388608, 3, 10, 119, 691476447, 119, 691601847, 125400
8388608, 4, 10, 119, 691610727, 119, 691734567, 123840
8388608, 5, 10, 119, 691741647, 119, 691862967, 121320
8388608, 6, 10, 119, 691870767, 119, 691991727, 120960
8388608, 7, 10, 119, 691998927, 119, 692149887, 150960
8388608, 8, 10, 119, 692162247, 119, 692284047, 121800
8388608, 9, 10, 119, 692291247, 119, 692412927, 121680
16777216, 0, 10, 119, 692420607, 119, 698691087, 6270480
16777216, 1, 10, 119, 698722647, 119, 701908287, 3185640
16777216, 2, 10, 119, 701931567, 119, 702532047, 600480
16777216, 3, 10, 119, 702540447, 119, 703520247, 979800
16777216, 4, 10, 119, 703528647, 119, 703806927, 278280
16777216, 5, 10, 119, 703814127, 119, 704105007, 290880
16777216, 6, 10, 119, 704115087, 119, 704399367, 284280
16777216, 7, 10, 119, 704406807, 119, 704664927, 258120
16777216, 8, 10, 119, 704672247, 119, 704933967, 261720
16777216, 9, 10, 119, 704942007, 119, 705197367, 255360
33554432, 0, 10, 119, 705206007, 119, 718937368, 13731361
33554432, 1, 10, 119, 719255848, 119, 732041368, 12785520
33554432, 2, 10, 119, 732076408, 119, 745467089, 13390681
33554432, 3, 10, 119, 745498289, 119, 758549009, 13050720
33554432, 4, 10, 119, 758579489, 119, 772443090, 13863601
33554432, 5, 10, 119, 772473930, 119, 784075530, 11601600
33554432, 6, 10, 119, 784103850, 119, 799029931, 14926081
33554432, 7, 10, 119, 799061731, 119, 812426132, 13364401
33554432, 8, 10, 119, 812463812, 119, 825670772, 13206960
33554432, 9, 10, 119, 825697652, 119, 838968933, 13271281
50331648, 0, 10, 119, 838995813, 119, 858921573, 19925760
50331648, 1, 10, 119, 858953973, 119, 879499534, 20545561
50331648, 2, 10, 119, 879529774, 119, 899781215, 20251441
50331648, 3, 10, 119, 899813015, 119, 917663136, 17850121
50331648, 4, 10, 119, 917694096, 119, 939956617, 22262521
50331648, 5, 10, 119, 939993217, 119, 958662937, 18669720
50331648, 6, 10, 119, 958694017, 119, 978534818, 19840801
50331648, 7, 10, 119, 978571298, 119, 998353539, 19782241
50331648, 8, 10, 119, 998388579, 120, 17454540, 19065961
50331648, 9, 10, 120, 17488380, 120, 38087221, 20598841
67108864, 0, 10, 120, 38126341, 120, 63385262, 25258921
67108864, 1, 10, 120, 63419462, 120, 90176103, 26756641
67108864, 2, 10, 120, 90215103, 120, 114481744, 24266641
67108864, 3, 10, 120, 114515224, 120, 141748265, 27233041
67108864, 4, 10, 120, 141783305, 120, 166453986, 24670681
67108864, 5, 10, 120, 166494426, 120, 193258027, 26763601
67108864, 6, 10, 120, 193285987, 120, 217231868, 23945881
67108864, 7, 10, 120, 217264868, 120, 243662349, 26397481
67108864, 8, 10, 120, 243694629, 120, 267378070, 23683441
67108864, 9, 10, 120, 267408310, 120, 293071151, 25662841
-- finished.
```

(4) Result

Table 5-58: Result (Type4)

Array size (MB)	1	2	3	4	5	6	7	8	9	10
2M	1.63740	1.34880	0.01728	0.00996	0.00948	0.00960	0.00972	0.00948	0.00936	0.00948
4M	2.22336	1.29840	0.05184	0.04308	0.04248	0.04440	0.04500	0.04380	0.04440	0.04380
8M	3.50856	1.97088	0.13080	0.12852	0.12360	0.12120	0.12156	0.12120	0.12180	0.12216
16M	6.16272	3.23760	0.25104	0.24816	0.24696	0.24960	0.24876	0.24660	0.24828	0.24912
32M	11.08860	10.80300	10.97712	10.92012	10.93188	10.86540	11.00568	10.93836	10.89228	10.86912
48M	16.34928	15.21552	15.54264	15.69276	16.03968	15.81336	15.92712	16.00848	15.96396	15.99540
64M	21.41364	19.84644	20.74308	20.96724	20.87376	20.91876	20.95980	20.31012	20.85504	20.67840

Table 5-59: Result (Type2)

Array size (MB)	1	2	3	4	5	6	7	8	9	10
2M	1.72224	1.18176	0.01440	0.00984	0.00960	0.00984	0.00972	0.00960	0.00984	0.00960
4M	2.28300	1.27488	0.05088	0.04308	0.04260	0.05064	0.04416	0.04428	0.04248	0.04272
8M	3.53136	2.00724	0.12324	0.12084	0.13680	0.12480	0.11880	0.12000	0.11976	0.11964
16M	6.00696	3.31860	0.24864	0.26520	0.24528	0.24456	0.25056	0.25344	0.25260	0.26844
32M	11.28720	10.94304	10.96320	10.90296	10.72728	10.85376	11.14068	11.02704	10.81728	11.05776
48M	15.91368	15.74028	15.86424	15.99348	15.88404	15.86100	15.63744	16.13556	15.80496	15.93732
64M	21.41364	19.68180	20.44608	20.51640	20.91144	20.47980	20.67780	21.11076	20.79684	20.89596

Table 5-60: Result

Array size (MB)	Virtualized Linux (Type4) [ns]			Native Linux (Type2) [ns]		
	Ave.	Min.	Max.	Ave.	Min.	Max.
2M	1.63740	0.00936	0.30706	1.72224	0.0096	0.298644
4M	2.22336	0.04248	0.38806	2.283	0.04248	0.391872
8M	3.50856	0.12120	0.64703	3.53136	0.1188	0.652248
16M	6.16272	0.24660	1.13888	6.006961	0.24456	1.1354281
32M	11.08860	10.80300	10.92916	11.2872	10.72728	10.9720204
48M	16.34928	15.21552	15.85482	16.13556	15.637441	15.8772007
64M	21.41364	19.84644	20.75663	21.413641	19.6818	20.6930528

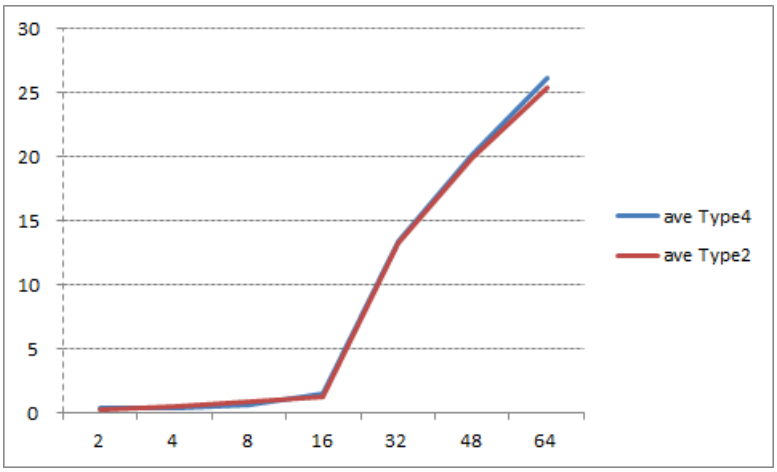


Figure 5-16: Average Result Virtualized Linux and Native Linux (Type4 and Type2)

(5) Consideration

This test result is observed quite similar to the result of 5.11.4, random write performance. As this memory allocation benchmark program performs 32-bit data write per every 4kB allocation size to force the actual allocation of the memory page, this data write is estimated to determine this memory allocation/deallocation performance benchmark.

5.11.6. Read Cached/Uncached memory performance

(1) Description

Measure the performance to read the cached/uncached memory on virtualized Linux and native Linux.

Measurement tool is lmbench.

(2) Precondition

- Measure on virtualized Linux and native Linux (Type4 and Type2)

  *Both types stop the function of Linux App
- Use the result of Sequential Read for the smallest test size as cached performance, and the result of Random Read for the largest size as uncached performance.
- Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Cached performance is taken from the minimum size (0.00049MB) result of 5.11.1
2. Uncached performance is taken from the maximum size (64.000MB) result of 5.11.3

As the uncached memory access is similar to the cache miss operation, the uncached memory access performance can be substituted by the cache miss performance. The typical cached performance is the smallest size of the sequential read test (5.1.1) which the test size is enough smaller than L1 cache size (32kB), and the typical uncached performance is the largest size of the random read test (5.1.3) wich is the nearest one of the true uncache performance.

(4) Result

Table 5-61: Cached Result

Virtualized Linux (Type4) [ns]	Native Linux (Type2) [ns]
2.67	2.67

Table 5-62: UnCached Result

Virtualized Linux (Type4) [ns]	Native Linux (Type2) [ns]
260.65	227.99

(5) Consideration

This result is expected. As the uncached memory access performance is taken from the result of random access test, it includes the effect of TLB miss. See 5.11.3 for detail.

5.11.7. TLB(Translation look aside buffer) miss performance

(1) Description

Measure the performance of TLB miss penalty on virtualized Linux and native Linux.

Measurement tool is lmbench.

Following Figure 5-13 describes the address translation hardware. When the TLB miss occur, the Native Linux performs the page table walk once for Linux only, and the Virtualized Linux does twice, one for the Virtualized Linux itself, and one more for Hypervisor. On the other hand, when TLB hits in address translation, the performance will not differ because page table walk doesn't happen.

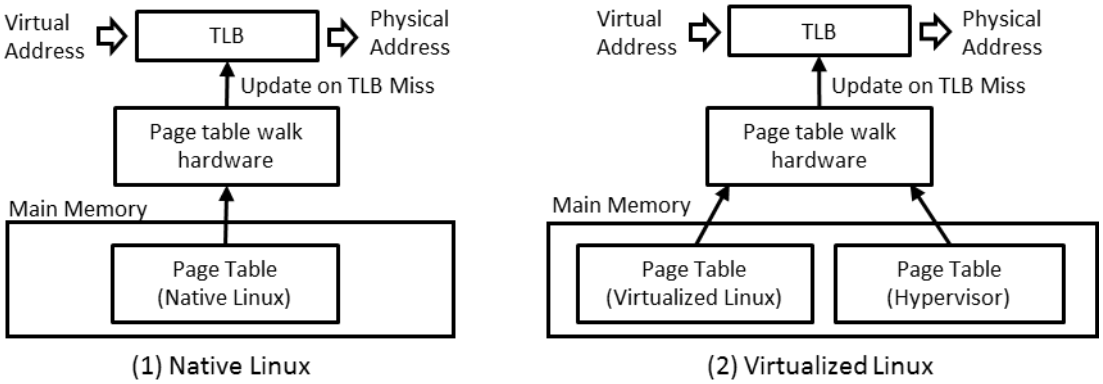


Figure 5-17: Address translation model

(2) Precondition

- Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)
  *Both types stop the function of Linux App
- Use lmbench's lat_mem_rd command on terminal software.
- Compare the performance between virtualized Linux and native Linux.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Login to Linux.

```
salvator-x login: root
```

2. Run the following command to change the directory.

```
root@salvator-x:~# cd tools/lmbench
```

3. Run the following command for measurement.

```
root@salvator-x:~/tools/lmbench# ./lat_mem_rd 16M 1048576
```

After finishing a command, you will see the log like below.

Red square is results.

```
root@salvator-x:~/tools/lmbench# ./lat_mem_rd 16M 1048576
"stride=1048576
1.00000 2.674
1.50000 2.671
2.00000 2.671
3.00000 2.670
4.00000 2.670
6.00000 13.135
8.00000 11.771
12.00000 12.521
16.00000 13.693
```

(4) Result(Reference value)

Table 5-63: Result (Type4)

Array size (MB)	1	2	3	4	5	6	7	8	9	10
1	2.671	2.671	2.670	2.674	2.674	2.674	2.671	2.671	2.671	2.67
1.5	2.671	2.671	2.671	2.670	2.671	2.671	2.671	2.671	2.671	2.67
2	2.671	2.671	2.671	2.671	2.671	2.671	2.671	2.671	2.671	2.68
3	2.671	2.671	2.671	2.671	2.671	2.671	2.671	2.671	2.671	2.68
4	2.671	7.679	7.679	7.679	2.671	2.671	13.545	13.942	13.619	13.79
6	13.134	13.134	10.940	13.134	11.036	11.036	14.025	14.026	13.150	13.13
8	14.026	13.357	11.770	14.025	11.789	11.789	13.357	13.357	14.025	14.05
12	12.523	13.580	13.579	13.579	12.523	12.523	13.579	13.581	14.025	13.58
16	12.899	13.691	12.902	12.898	12.899	12.899	13.692	12.919	13.698	12.92

Table 5-64: Result (Type2)

Array size (MB)	1	2	3	4	5	6	7	8	9	10
1	2.685	2.669	2.684	2.669	2.669	2.669	2.669	2.685	2.669	2.684
1.5	2.669	2.669	2.669	2.669	2.669	2.669	2.669	2.685	2.669	2.685
2	2.669	2.669	2.669	2.669	2.669	2.669	2.669	2.685	2.669	2.685
3	2.669	2.669	7.125	7.671	2.669	2.669	2.669	2.685	2.669	7.693
4	7.719	13.806	7.720	2.669	2.669	7.673	7.673	2.685	7.676	7.720
6	14.013	10.911	13.132	13.123	11.010	13.123	11.010	10.842	13.201	13.201
8	13.354	11.761	14.022	11.760	13.345	13.345	11.761	11.830	13.354	13.354
12	13.576	14.013	14.022	12.511	12.511	13.568	12.511	13.650	13.577	13.577
16	13.762	13.679	12.966	13.680	12.887	12.888	13.679	12.964	12.896	12.965

Table 5-65: Result

	Virtualized Linux(Type4) [ns]			Native Linux(Type2) [ns]		
Array size (MB)	Latency Ave.	Latency min.	Latency max.	Latency Ave.	Latency min.	Latency max.
1	2.672	2.670	2.674	2.675	2.669	2.685
1.5	2.671	2.670	2.671	2.672	2.669	2.685
2	2.671	2.671	2.671	2.672	2.669	2.685
3	2.671	2.671	2.671	4.119	2.669	7.693
4	8.017	2.671	13.942	6.801	2.669	13.806
6	12.624	10.940	14.026	12.357	10.842	14.013
8	13.055	11.770	14.026	12.789	11.760	14.022
12	13.277	12.523	14.025	13.352	12.511	14.022
16	13.166	12.898	13.698	13.237	12.887	13.762

(5) Consideration

The TLB miss latency is calculated by subtracting the average result of typical TLB and L2C hit latency (1MB results of 5.11.1) from the typical TLB miss and L2C hit latency (16MB results of 5.11.7).
It is about 7ns (virtualized Linux: 6.88, native Linux: 6.94) per one access, and it is smaller than the L2C miss latency (about 31ns) measured in 5.1.1. This result also mean the TLB page table walk in this test scenario does not produce external memory access, and the 2-stage translation overhead is not observed in the result of virtualized Linux.

5.11.8. VA - IPA -PA conversion performance

(1) Description

Measure the performance to translate the virtual address into intermediate and physical address by the MMU on virtualized Linux.

(2) Precondition

- This VA-IPA-PA conversion performance is calculated by the TLB miss penalty of both the virtualized Linux and the native Linux. As the test result of 5.11.7 does not have the specific difference between the native Linux and the virtualized Linux, the result of 5.11.3 is used to calculate the performance difference of the VA-IPA-PA translation on virtualized Linux, and VA-PA translation on native Linux. The result of 5.11.7 is used as the common TLB miss overhead for both VA-IPA-PA and VA-PA translations.

(3) How to measure

1. Use result of 5.11.3 and 5.11.7.

(4) Result

Table 5-66: Result

	VA-IPA-PA translation latency (ns)
Type 4 (Virtualized Linux)	83.56

The calculation of VA-IPA-PA translation time (Ttr) is listed below.

Ttr =

((Virtualized Linux's random read latency: A)
- (Native Linux's random read latency: B)) * 2
+ (Virtualized Linux's one TLB-miss overhead: C)
= 83.56 (ns)

The A value includes the L2 cache miss read time and TLB miss overhead with 2-stage translation. (VA-IPA-PA), and the B value includes the L2 cache miss read time and TLB miss overhead with 1-stage translation. (VA-PA). So the Ttr can be calculated by ((A) - (B)) * 2 + (C).

The actual value is:

A: 64MB average result of Virtualized Linux in 5.11.3 (259.59ns)

B: 64MB average result of Native Linux in 5.11.3 (221.25ns)

C: Calculated TLB miss overhead of Virtualized Linux in 5.11.7 (6.88ns)

(5) Consideration

From the result of 5.11.1 and 5.11.3, this VA-IPA-PA overhead include 2 times of external memory read cycles. For the result of 5.11.7, the observed of TLB miss doesn’t contain any external memory read access, and this is why 5.11.7 doesn’t observe the meaningful difference between native Linux and virtualized Linux.

5.12. Network Performance(Linux)

5.12.1. Send / Receive data to cloud

Out of Scope.

5.12.2. Packet Loss

Out of Scope.

5.12.3. End-to-end Input events delivery latency

Out of Scope.

5.12.4. Delay variation(Jitter)

Out of Scope.

5.12.5. Send / Receive data to cloud

Out of Scope.

5.12.6. Throughput(Bandwidth)

Out of Scope.

5.12.7. Ethernet Bit error rate (BER)

Out of Scope.

5.13. Power Consumption Performance

5.13.1. standby current

Out of Scope.

5.13.2. Power consumption when sleep mode

Out of Scope.

5.13.3. Average power usage performance

Out of Scope.

5.14. RTOS performance

5.14.1. INTEGRITY OS Performance

All measurements below using api_measurements.gpj in release T14.0 and are in nanoseconds.

5.14.1.1. Deadlock break time

(1) Description
Out of scope.

(2) Precondition

(3) How to measure

(4) Result

(5) Consideration

5.14.1.2. Semaphore processing

(1) Description

(2) Precondition
GHS special environment is needed. Our PoC doesn't include it.

(3) How to measure
GHS special environment is needed. Our PoC doesn't include it.

(4) Result

Table 5-67: Result (nano second)

TryToObtainSemaphore on available Semaphore	321.332
TryToObtainSemaphore on unavailable Semaphore	324.627
TimedWaitForSemaphore on available Semaphore	1136.588
ReleaseSemaphore no waiter	310.069
TryToClearSemaphore	310.685
GetSemaphoreValue	312.120
WaitForLocalMutex on available LocalMutex	7.336
ReleaseLocalMutex	6.002
TryToObtainLocalMutex on available LocalMutex	6.002
TryToObtainLocalMutex on unavailable LocalMutex	6.002
TryToObtainSemaphore on available Semaphore	321.332

(5) Consideration

This result is expected.

5.14.1.3. Message Passing

(1) Description

(2) Precondition
GHS special environment is needed. Our PoC doesn't include it.

(3) How to measure
GHS special environment is needed. Our PoC doesn't include it.

(4) Result

Table 5-68: Result (nano second)

0-byte message SynchronousSend	440.850
SynchronousReceive on Object with pending message	322.660
Simple AllocateMessageQueueBuffer	67.586
Simple SendOnMessageQueue	64.801
Simple GetMessageCountForMessageQueue	60.468
Simple TimedReceiveOnMessageQueue	65.464
Simple ReceiveOnMessageQueue	68.113
Simple FreeMessageQueueBuffer	21.527

(5) Consideration
This result is expected.

5.14.1.4. Context Switching

(1) Description

(2) Precondition
GHS special environment is needed. Our PoC doesn't include it.

(3) How to measure
GHS special environment is needed. Our PoC doesn't include it.

(4) Result

Table 5-69: Result (nano second)

context-switch overhead	680.429

(5) Consideration

This result is expected.

5.14.1.5. Interrupt Latency

(1) Description

(2) Precondition

GHS special environment is needed. Our PoC doesn't include it.

(3) How to measure

GHS special environment is needed. Our PoC doesn't include it.

(4) Result

Table 5-70: Result (nano second)

Timer interrupt latency	360.121

(5) Consideration

This result is expected.

5.15. Application Switching performance

5.15.1. Application Switching performance

Out of Scope.

5.16. Malicious App

5.16.1. INTEGRITY meter cluster application keeps 60fps even if Linux malicious app runs

(1) Description

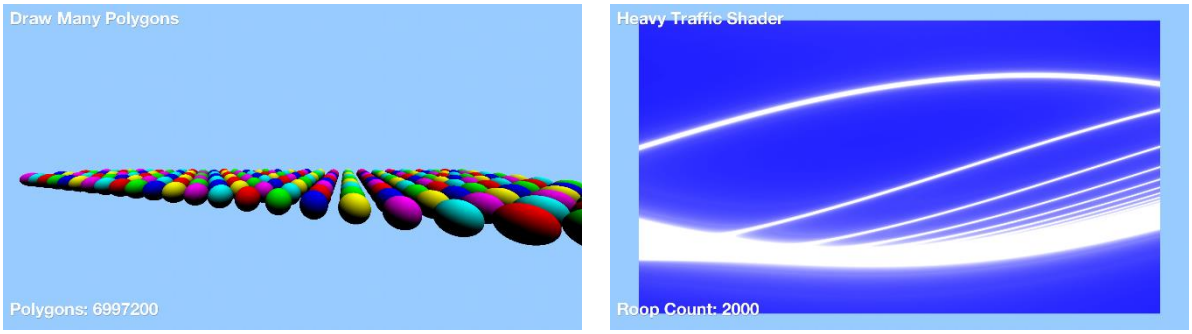
Measure the FPS of the INTEGRITY graphics meter application (Rightware graphics meter) with a malicious application running on virtualized Linux.

The GPU on R-Car H3 can process multiple graphics applications at the same time using the context switch. There is a granularity in the context switch. If a graphics application (e.g. malicious app on Linux) did either of the followings, the GPU could not do the context switch for a long time and the performance of other graphics applications (e.g. meter cluster on INTEGRITY Disable script to run the demo applications automatically.) would drop.

- Draw many polygons in one glDrawXXX()
- Do many pixel processing in one sub-surface (“tile”) of the drawing surface

The GPU on R-Car H3 has a GPU robustness feature to avoid the performance drop caused by malicious applications on Linux. On Linux, a malicious application would issue a high load GPU workload to the GPU via the graphics driver. GPU has a timeout value for the GPU workloads. If the workload from Linux run too long and exceeds the timeout, GPU judges that the workload is “malicious” and kill it.

This section tests it.



(a) Many polygons (b) Heavy shaders

Figure 5-18: Images of malicious applications

Table 5-71: Malicious applications features

Item	Description
Application	● Renesas original malicious applications (a) Many polygons: draw 1,800 balls (about 10,000 polygons/ball) (b) Heavy shaders: draw 5000 lines in one fragment shader ● OpenGL ES 3.1 application
Resolution	● 1920x720

(2) Precondition

- Measure on virtualization PoC but Rightware meter cluster only runs (Type1)
- Enable the GPU robustness feature
- Application on INTEGRITY: Rightware meter cluster

- Application on Linux: Renesas original malicious application (Many polygons or Heavy shaders. They shall be stored in the root file system)
- Window size on INTEGRITY/Linux: 1920x720
- Window system on Linux: DRM

(3) How to measure (preliminary)

1. Start Type1_mono.5.16.1 using MULTI debugger.
2. Login to Linux using Terminal.

```
salvator-x login: root
```

3. Stop Weston using the following command.

```
root@salvator-x:~# systemctl stop weston
```

4. Edit /etc/powervr.ini as follows to use the DRM Window System.

   Before the modification

```
WindowSystem=libpvrWAYLAND_WSEGL.so
```

   After the modification

```
;WindowSystem=libpvrWAYLAND_WSEGL.so
WindowSystem=libpvrDRM_WSEGL.so
```

5. Disable Weston using the following command.

```
root@salvator-x:~# systemctl disable weston
```

6. Restart Type1_mono.5.16.1.
7. Login to Linux.

```
salvator-x login: root
```

8. Run the malicious application as

```
root@salvator-x:~# cd malicious_app
```

   In case of Many polygons

```
root@salvator-x:~/malicious_app# ./ManyPolygons -count=1800 -fps
```

   In case of Heavy shaders

```
root@salvator-x:~/malicious app# ./HeavyShaders -count=5000 -fps
```

9. Check the performance of Rightware meter cluster by the FPS outputs shown to MULTI Debugger.

```
(omitted)
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
(omitted)
```

(4) Result (preliminary figures)

Table 5-72: Result

App on Linux	FPS of the Rightware meter cluster on INTEGRITY
(a) Many polygons	60
(b) Heavy traffic shaders	60

(5) Consideration

Rightware graphics meter kept 60FPS even if there is a malicious application on virtualized Linux. On the other hand, if we test with the GPU robustness feature disabled, the FPS of the Rightware graphics meter dropped and it ran awkwardly. The FPS was 20 in case of (a) and 16 in case of (b) respectively. This means that the effect of the GPU robustness feature.

Note: As the GPU robustness feature kills the GPU workload of the malicious application, nothing is drawn and the malicious application seems stopped.

5.16.2. INTEGRITY meter cluster application keeps 60fps even if memory leak or memory corruption is occurred on Linux side

(1) Description

Measure the FPS of the INTEGRITY graphics meter application (Rightware graphics meter) with an application which cause memory leaking or a memory corruption on virtualized Linux.

Table 5-73: Applications which cause memory leaking / memory corruption features

Item	Description
Application	● Renesas original applications (a) Memory leaking: allocate 50MB memory every 500ms (b) Memory corruption: write data to the Linux kernel memory
Resolution	● 1920x720

(2) Precondition

- Measure on virtualization PoC but Rightware meter cluster only runs (Type1)
- Application on INTEGRITY: Rightware meter cluster
- Application on Linux: Renesas original application (Memory leaking or Memory corruption. They shall be stored in the root file system)
- Window size on INTEGRITY/Linux: 1920x720
- Window system on Linux: DRM

(3) How to measure

1. Start Type1_mono.5.16.2 using MULTI debugger.
2. Login to Linux using Terminal.

```
salvator-x login: root
```

3. Restart Type1_mono.5.16.2.
4. In case of memory leaking application, run it as follows.

```
root@salvator-x:~# cd memory_app
root@salvator-x:~/memory_app# ./MemoryLeak
```

You will see the following message like follows. The application will be killed by the Linux kernel.

```
MemoryLeak: Memory allocation succeeded.
MemoryLeak: cnt=1 memory size=50000000byte total size=50000000byte
MemoryLeak: Memory allocation succeeded.
MemoryLeak: cnt=2 memory size=50000000byte total size=100000000byte
MemoryLeak: Memory allocation succeeded.
(omitted)
MemoryLeak: cnt=19 memory size=50000000byte total size=950000000byte
MemoryLeak: Memory allocation succeeded.
MemoryLeak: cnt=20 memory size=50000000byte total size=1000000000byte
[   93.484050] gst-launch-1.0 invoked oom-killer: gfp_mask=0x24201ca,
order=0, oom_score_adj=0
[   93.492438] CPU: 1 PID: 1926 Comm: gst-launch-1.0 Tainted: G
W  O    4.4.0-yocto-standard #3
[   93.501526] Hardware name: Renesas SalvatorX board based on r8a7795
(DT)
[   93.508233] Call trace:
(omitted)
[   94.063130] Out of memory: Kill process 1994 (MemoryLeak) score 536
or sacrifice child
[   94.071063] Killed process 1994 (MemoryLeak) total-vm:1037088kB,
anon-rss:1024464kB, file-rss:4kB, shmem-rss:0kB
Killed
```

5. In case of memory corrupt application, run it as follows.

```
root@salvator-x:~# cd memory_app
root@salvator-x:~/memory_app# ./MemoryCorrpution_mmap 1208483840 8800
```

Then sometimes the Terminal returns nothing because Linux hanged up depending on the timing.

6. During memory leaking or memory corruption application is running, check the performance of Rightware meter cluster by the FPS outputs shown to MULTI Debugger.

```
(omitted)
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
(omitted)
```

(4) Result

Table 5-74: Result

App on Linux	FPS of the Rightware meter cluster on INTEGRITY
(a) Memory leaking	60
(b) Memory corruption	60

(5) Consideration

Rightware graphics meter kept 60FPS even if there is a memory leaking or memory corruption application on virtualized Linux. This shows the safety of the virtualization system using INTEGRITY Multivisor. However we sometimes got the following results. We will solve them in the future.

- The Rightware cluster runs jumpy one time
- The Rightware cluster stops

5.17. Robustness

5.17.1. Unexpected memory access blocking system by using IPMMU,LifeCycle

(1) Description

Figure 5-19 describes the operation model of IPMMU. It can provide the address translation and address space protection for built-in bus master modules in R-Car SoCs.

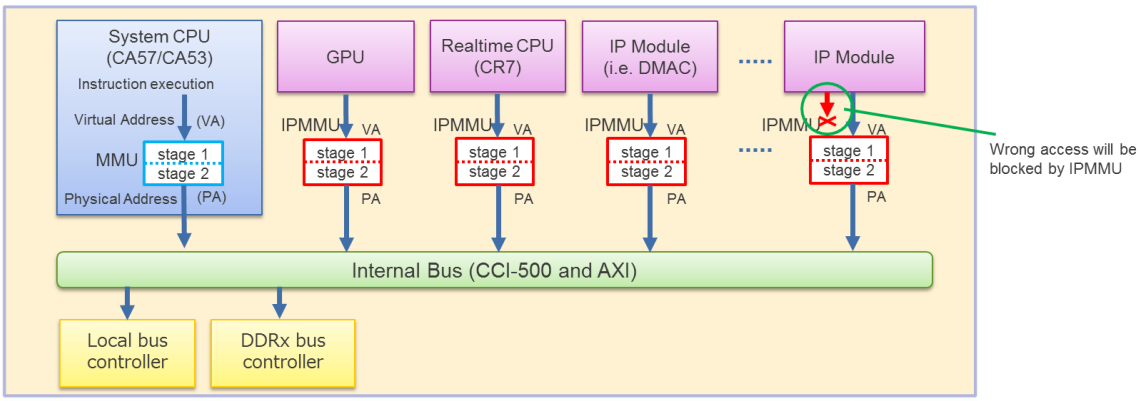


Figure 5-19: IPMMU operation model

Figure 5-20 describes the operation model of LifeCycle. LifeCycle will put the specified security level to each access request issued by R-Car Gen3 built-in modules, and guard its register space by the request's security level. This generally protect the hardware modules from being accessed by the unauthorized bus master, either CPU or other built-in module.

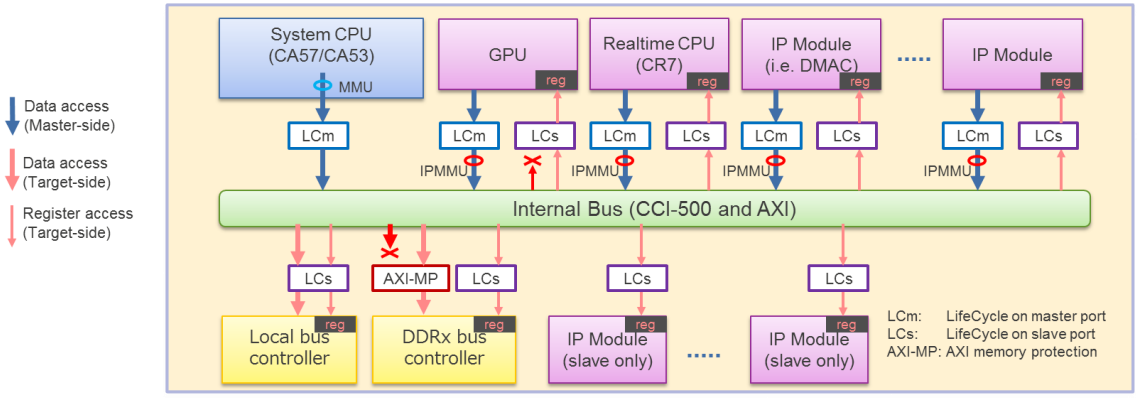


Figure 5-20: LifeCycle operation model

(2) Precondition

- Measure on virtualization PoC (Type1)
- Connect a Camera to composite (CVBS_IN: CN21)

(3) How to measure

- IPMMU Confirmation procedure

Use GHS test program and Renesas original test program.
GHS test program uses IPMMU, to check whether the meter cluster application is not disturbed even if the display memory is to be corrupted by other hardware module intentionally.

Normal camera image -> broken camera image, in this way image are displayed alternately for the first 10 seconds. In the next 10 seconds, IPMMU protects illegal memory access and normal camera image continues to be displayed.

Renesas original test program uses LifeCycle, to check whether the part of the display unit hardware (DU0/DU1) that Linux are using can be protected by LifeC, and when the specific LifeC setting is issued, Linux can no longer access the display unit hardware (stopped and put error due to blocked DU0/1 access).

- LifeCycle Confirmation procedure

1. Launch Type1
2. Select [Target] – [Connect] from Menu bar of MULTI
3. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
4. Select “Run mode target”
5. Select [Target] - [Load Module] - [Load Module...] from Menu bar.
6. Load the " LifeCTest.ael" file included in the deliverables.
7. Press F5 for start.

(4) Result

- IPMMU

Result is expected.

- LifeCycle

When LifeCycle application start, Kernel panic in Linux is occurred.

```
[   40.219695] [<ffffffc0000855e0>] el1_irq+0xa0/0x10c
[   40.224568] [<ffffffc0000867c0>] arch_cpu_idle+0x10/0x18
[   40.229883] [<ffffffc0000ebf1c>] cpu_startup_entry+0x13c/0x230
[   40.235718] [<ffffffc00070bd2c>] rest_init+0x84/0x90
[   40.240682] [<ffffffc000a3e940>] start_kernel+0x384/0x398
[   40.246077] [<ffffffc0000811b4>] 0xffffffc0000811b4
[   40.250950] handlers:
[   40.253225] [<ffffffc000431ea0>] rcar_du_crtc_irq
[   40.257930] Disabling IRQ #51
```

(5) Consideration

These results are expected.

5.18. Rebooting of Linux

5.18.1. INTEGRITY meter cluster application keeps 60fps even if Linux rebooting is executed

(1) Description

Measure the display performance (frame per sec) of Instrument Cluster keeps 60fps even if Linux rebooting is executed on virtualization PoC.

(2) Precondition

- Measure on virtualization PoC (Type1)
- Measure FPS by incorporating the FPS measurement method on Meter Cluster.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Make a following script (file name: rundemo) to reboot Linux shortly.

```
#!/bin/sh
sleep 5
export XDG_RUNTIME_DIR=/run/user/root
export LD_LIBRARY_PATH=/home/root/Futuremark


sleep 5
cd /home/root/nng_env/center
export WSEGL_ENABLE_TRIPLE_BUFFERING=12
./salvatorx-linux_salvatorx_64-release&
sleep 5
cd /home/root/IMG_SDK35

export WSEGL_ENABLE_TRIPLE_BUFFERING=2
./OGLES3Coverflow -aasamples=4 -width=1920 -height=720 -posx=0 -posy=0 &

export XDG_RUNTIME_DIR=/run/user/root
modprobe -a mmngr mmngrbuf vspm vspm_if vsp2 uvcs_drv

gst-launch-1.0 \
filesrc location=/home/root/movie/big_buck_bunny_720p_h264.mp4 \
! qtdemux name=demux \
demux.audio_0 ! queue \
! omxaaclcdec \
! alsasink device=hw:0,0 \
demux.video_0 ! queue \
! h264parse \
! omxh264dec \
! vspfilter \
! video/x-raw, format=BGRA, width=1920, height=720 \
! waylandsink &

sleep 20
reboot -f
```

2. Replace the above rundemo with /etc/init.d/rundemo in the rootfs.
3. Run the monolith for the PoC (Type1_mono).
4. Login to Linux.

```
salvator-x login: root
```

5. Enable autorun setting for rundemo.

```
root@salvator-x:~# cd /etc/init.d
root@salvator-x:/etc/init.d# update-rc.d rundemo defaults
```

6. Run the monolith for the PoC (Type1_mono). Linux applications will start automatically.
7. After the 3D navigation start to show the map, clear the log in the MULTI Debugger.
8. Start to check the performance of Rightware meter cluster by the FPS outputs shown to MULTI Debugger until the Linux rebooting finish.

```
(omitted)
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
I/O:
I/O: LOG: fps: 59.980335
(omitted)
```

9. Repeat “step 3 - step 5” nine times.

(4) Result

Table 5-75: FPS of the Rightware meter cluster

	Ave.	1	2	3	4	5	6	7	8	9	10
Rightware meter cluster FPS	59	59	60	60	59	59	59	59	59	60	59

(5) Consideration

Rightware graphics meter kept almost 60FPS even if Linux rebooted. This shows the safety of the virtualization system using INTEGRITY Multivisor. If any issues occur on Linux (e.g. hang-up, run unstably, etc.) we can reboot Linux without any side effects on INTEGRITY, especially for the meter cluster.

5.19. Memory usage

5.19.1. Check the memory usage of Multivisor

(1) Description

Measure the memory usage of Multivisor of Center Information and Instrument Cluster / Head-up display on virtualization PoC.

Measurement tool is top command for Linux, Multi Debugger for INTEGRITY.

(2) Precondition

- Measure on virtualization PoC (Type1)
- Use a tool including in Multi Debugger.

(3) How to measure

Calculate a use memory size by subtracting unused memory from a total memory size (3,968 Mbyte).

Estimated memory usage = Total memory - free memory [Virtualized Linux, Memory Pool, heap]

Note that this value is not exactly because we have omitted the small pieces of memory.

- Measurement of the unused memory size in Center Information
  1. Login to Linux.

```
salvator-x login: root
```

  2. Run the following top command.

```
root@salvator-x:~# top -d10 -n61
```

  After finishing a command, you will see the log like below.

  Red square is a result.

```
root@salvator-x:~/bin/x86_64-linux-gnu# top -d10
Mem: 578956K used, 1273804K free, 20824K shrd, 9940K buff, 119992K cached
CPU:  27% usr   5% sys   0% nic  67% idle   0% io   0% irq   0% sirq
Load average: 1.08 1.21 1.21 1/104 1794
```

- Measurement of the unused memory size in MemoryPool
  1. Select [Target] – [Connect] from Menu bar on MULTI.
  2. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
  3. Select “Run mode target”
  4. Run the following command on “Trg” tab.

```
INDRT2>ct
```

  5. Run the following command on “Trg” tab.

```
INDRT2>lt
```

After finishing a command, you will see the log like below. Check the result of the memory size (the right side) of the line of "Type1_kernel".

```
Type1_kernel                              0x0000000000001000/0x000000000cb24000
   0xffffffa6c571e000 exited          127 0x0000000000000630/0x0000000000008000   0.17% Initial
   0xffffffa6c5720000 running           0 0x0000000000000020/0x0000000000000400  41.14% Idle0
   0xffffffa6c5722000 running           0 0x0000000000000020/0x0000000000000400  61.42% Idle1
   0xffffffa6c5724000 running           0 0x0000000000000020/0x0000000000000400  63.93% Idle2
```

Similarly, we choice the large value of result of "multivisor_loader", "pvrserver_as0", "devtree_generic_server_module" and "Sakura".

6. Run the step 5 process 10 seconds after the result is displayed.

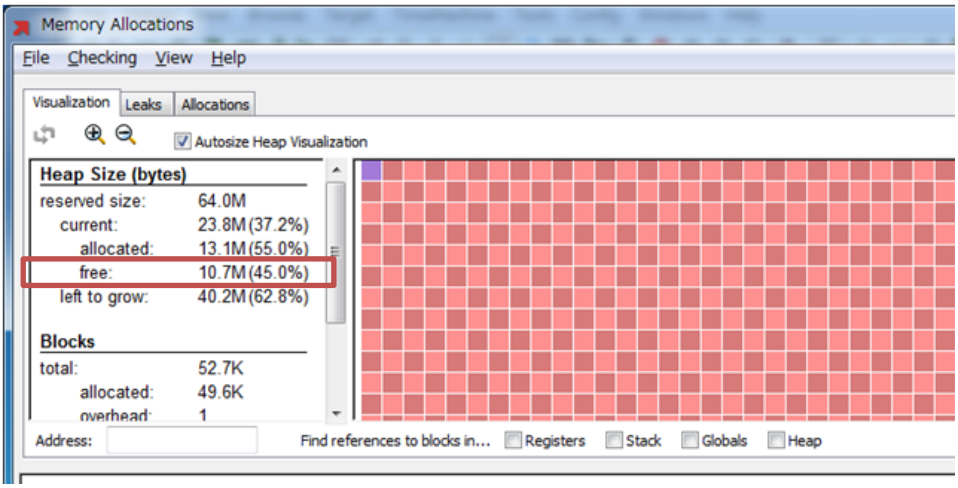
   Repeat this for 10 minutes.

- Measurement of the unused memory size in Heap Memory

1. Select [Target] – [Connect] from Menu bar of MULTI
2. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
3. Expand a tree of "INTEGRITY SMP Application" in "Target View", and choose "Core 1, Cortex-A57/Type1_mono".
4. Select [Debug] – [Halt on Selected Items] from Menu bar of MULTI
5. Run the following command on “Cmd” tab.

```
MULTI> heapview
```

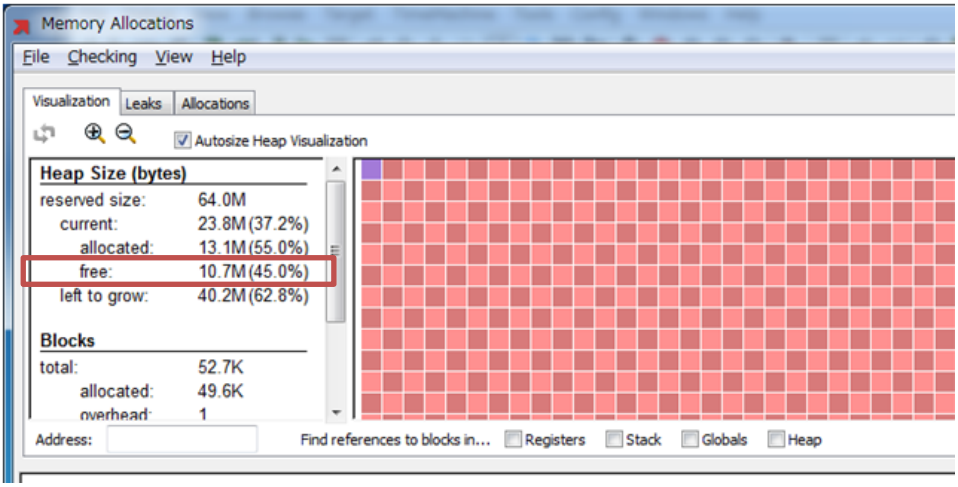
After finishing a command, you will see the log like below. Check the result of the “free” value.



6. Expand a tree of "Run mode target" in "Target View", and find “devtree_generic_server_module”
7. Expand a tree of “devtree_generic_server_module”, and choose “Initial”
8. Select [Debug] – [Halt on Selected Items] from Menu bar of MULTI
9. Run the following command on “Cmd” tab.

```
MULTI> heapview
```

After finishing a command, you will see the log like below. Check the result of the “free” value.



10. Continue, and run "step 6 - step 9" for the following targets.
    “DISCOM_sample_virt”
    “FBServer”
    “INT_Logo_sample_virt”
    “ip46router_devtree_module”
    “ivfsserver_devtree_module”
    “multivisor_loader”
    “multivisor_net_server”
    “multivisor_vmm”
    “pvrserver_as0”
    “Sakura”

(4) Result

Table 5-76: Unused memory in Center Information

	Center Information [Byte]
Ave.	862529536
1	885567488
2	877494272
3	871903232
4	870629376
5	867295232
6	861986816
7	857751552
8	854102016
9	843231232
10	835334144

Table 5-77: Unused memory in Memory Pool

AddressSpace	MemoryPool [Byte]
Type1_kernel	213008384
multivisor_loader	209858560
pvrserver_as0	165662720
devtree_generic_server_module	22028288
Sakura	67108864
Total	677666816

Table 5-78: Unused memory in Heap Memory

	Type1_kernel [Byte]	devtree_generic_server_module [Byte]	DISCOM_sample_virt [Byte]	FBServer [Byte]	INT_Logo_sample_virt [Byte]	ip46router_devtree_module [Byte]
Ave.	35574	12083	10824	11674	10711	4813
1	34918	12083	10752	11674	10752	4812
2	35737	12083	10854	11674	10650	4812
3	35737	12083	10854	11674	10650	4812
4	35737	12083	10854	11674	10650	4812
5	35430	12083	10854	11674	10650	4812
6	34918	12083	10854	11674	10752	4812
7	35942	12083	10854	11674	10752	4812
8	35942	12083	10752	11674	10752	4812
9	35635	12083	10752	11674	10752	4812
10	35737	12083	10854	11674	10752	4812
	ivfsserver_devtree_module [Byte]	**multivisor_loader [Byte]**	**multivisor_net_server [Byte]**	**multivisor_vmm [Byte]**	**pvrserver_as0 [Byte]**	**Sakura [Byte]**
Ave.	13537	10547	7373	65536	540201	13956547
1	16486	10547	7373	65536	600781	11324621
2	13209	10547	7373	65536	657510	13316915
3	13209	10547	7373	65536	384410	14365491
4	13209	10547	7373	65536	441242	14365491
5	13209	10547	7373	65536	441242	14365491
6	13209	10547	7373	65536	711270	14365491
7	13209	10547	7373	65536	512000	14365491
8	13209	10547	7373	65536	541082	14365491
9	13209	10547	7373	65536	556339	14365491
10	13209	10547	7373	65536	556134	14365491

Table 5-79: Result

Estimated memory usage	4160749568 – (862529536 + 677666816 + 14679419) = 2605873797 (Byte) =2485.16(MByte)

(5) Consideration

This result is expected

5.20. Stress Tolerance

5.20.1. Perform a continuous test for 48 hours with stress of various tools and video/audio playback

(1) Description

Confirm to perform virtualization PoC keeping for 48 hours with stress of various tools and video/audio playback.

(2) Precondition

- Measure on virtualization PoC (Type1)

(3) How to measure

1. Start Type1_mono.
2. Leave it for 48 hours.
3. Check that it keeps running without any issues or not.

(4) Result

We have checked that the system ran 5 hours. We have seen several issues when running longer time.

(5) Consideration

We think there are still some stability issues remaining in this environment and will need further investigation.

5.21. Security

5.21.1. Domain, Application Isolation

(1) Description

Measure that INTEGRITY keeps running without any influence even if the Linux application accesses the memory on INTEGRITY.

The following figure shows the image of memory protection. Each application should run in its own space.

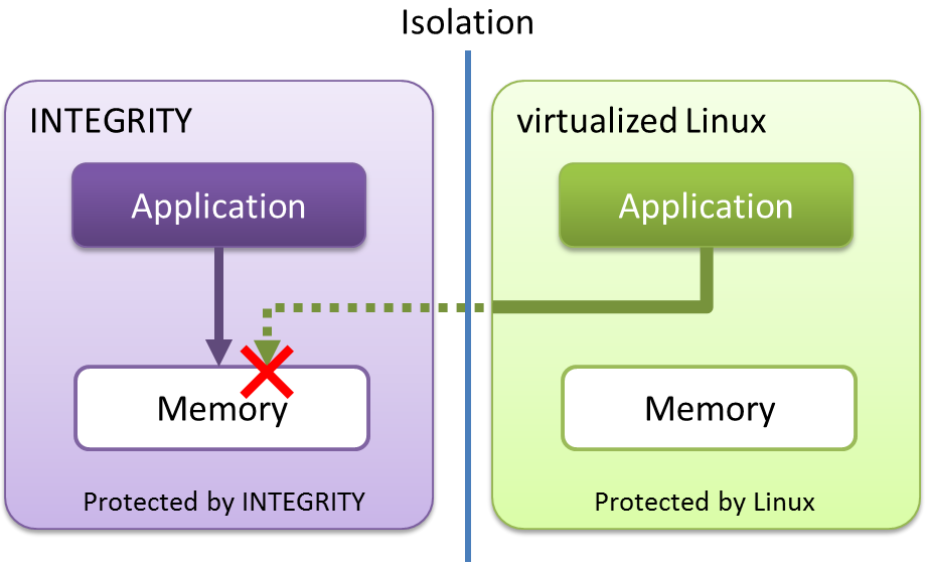


Figure 5-21: Image of memory protection

(2) Precondition

- Measure on virtualization PoC (Type1)
- Use Renesas original test program. The test program generates access request to unauthorized address space intentionally. And an application in INTEGRITY keep running without influence.

(3) How to measure

1. Refer to 5.16.

(4) Result

Refer to 5.16

(5) Consideration

Refer to 5.16

5.21.2. Illegal access of Resources / Memory

This and Section 5.21.1 is duplicated. Refer to Section 5.21.1.

5.21.3. Encryption/Decryption Performance

Out of Scope.

5.21.4. Secure boot process for each domain

Out of Scope.

5.22. End-to-End Latency

5.22.1. End-to-End UI Latency between RTOS and Linux (Image, binary, text)

Out of Scope.

5.23. Memory Utilization of Each Module

5.23.1. Memory utilization in IVI (Center Information)

(1) Description

Measure RAM utilization in Center Information application on virtualization PoC using top command on virtualization PoC.

(2) Precondition

- Measure on virtualized Linux on virtualization PoC (Type1)
- Use top command.
- Verified 10 times and use the average as the result value.

(3) How to measure

1. Refers to 5.19.1.

(4) Result

Table 5-80: Result

Virtualization PoC (Type1)	Value [Kbyte]
Center Information	1340790.00

(5) Consideration

This result come from the average of Table 5-73. This result is expected.

5.23.2. Memory utilization in meter (Instrument Cluster)

(1) Description

Measure RAM utilization in Instrument Cluster application on virtualization PoC using Command prompt.

The following applications are measurement target. Refer to Chapter 4 for details.

- Meter Cluster app
- OGLES31
- Display driver

(2) Precondition

- Measure on INTEGRITY on virtualization PoC (Type1)
- Use a command on Command prompt.

(3) How to measure

Calculate a used memory size by measurement for several kinds of memory sizes that each application uses, and adding each.

Memory utilization = MemoryPool + heap memory + ROM/RAM size + Memory Region.

- Measurement of the used memory size in MemoryPool

  Measurement method refers to “Measurement of the unused memory size in MemoryPool” of 5.19.1 .

  Excerpt from results of 5.19.1. Check the following results of the memory size (the left side) of the line of "Sakura".

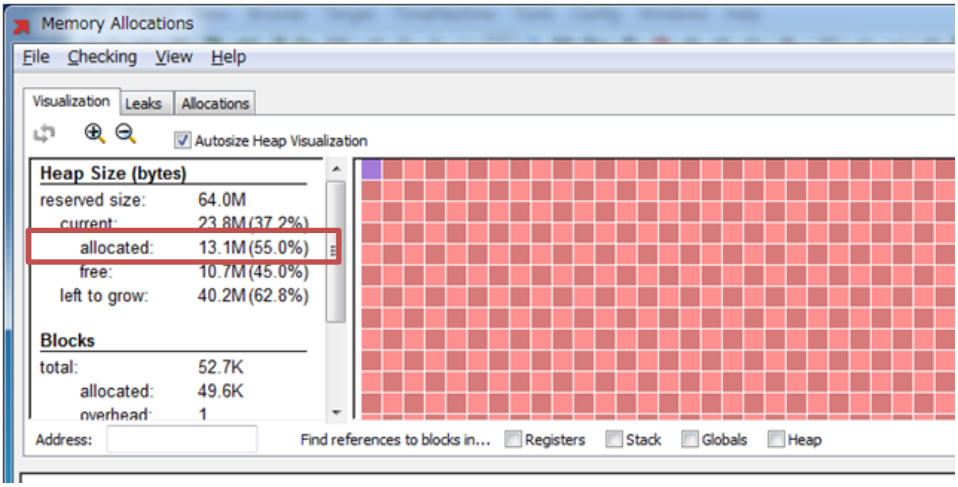
```
Sakura                                0x000000000000cb000/0x0000000004000000
   0xffffffa6c57f0000 running         127 0x0000000000001870/0x0000000000200000  31.39% Initial
   0xffffffa00fae3000 pending         128 0x0000000000000168/0x0000000000003cb8   0.00%
PosixServer
```

  Similarly, check the result of "pvrserver_as0" and "FBServer".

- Measurement of the used memory size in Heap Memory

  Measurement method refers to “Measurement of the unused memory size in MemoryPool” of 5.19.1 .

  Excerpt from results of 5.19.1. Check the following results of the memory size (the left side) of the line of "Sakura".



- Measurement of the used memory size in ROM/RAM size

1. Run the following command.(measurement ROM/RAM size of Sakura)

```
C:\>C:\ghs\comp_201516\gsize.exe -ram -rom C:\Sakura-h3
```

After finishing a command, you will see the log like below.

Red square is results.

```
C:\ghs\comp_201516\gsize.exe -ram -rom C:\Sakura-h3
        RAM_Size:         1828492
        ROM_Size:        19707803
```

2. Run the following command. (measurement ROM/RAM size of pvrserver_as0)

```
C:\>C:\ghs\comp_201516\gsize.exe -ram -rom C:\ghs\int1144\bin\devtree-arm64\pvrserver_a
s0
```

After finishing a command, you will see the log like below.

Red square is results.

```
C:\>C:\ghs\comp_201516\gsize.exe -ram -rom C:\pvrserver_as0
        RAM_Size:          109684
        ROM_Size:          784716
```

3. Run the following command. (measurement ROM/RAM size of FBServer)

```
C:\>C:\ghs\comp_201516\gsize.exe -ram -rom C:\FBServer
```

After finishing a command, you will see the log like below.

Red square is results.

```
C:\>C:\ghs\comp_201516\gsize.exe -ram -rom C:\FBServer
        RAM_Size:           23148
        ROM_Size:          122484
```

- Measurement of the Memory Region size.

The memory region of sakura use maximum following size.

Memory Region Size : 147540214 (Byte)

(4) Result

- Instrument Cluster
  - Sakura (Meter Cluster app)
  - pvrserver_as0 (OGLES31)
  - FBServer (Display driver)

Table 5-81: Used memory Pool

	Sakura (Byte)	pvrserver_as0 (Byte)	FBServer (Byte)
Ave.	895385.6	40686387.2	0
1	872448	40693760	0
2	913408	40701952	0
3	913408	40701952	0
4	913408	40685568	0
5	880640	40677376	0
6	880640	40677376	0
7	880640	40677376	0
8	921600	40685568	0
9	921600	40685568	0
10	856064	40677376	0

Table 5-82: Used heap memory

	Sakura (Byte)	pvrserver_as0 (Byte)	FBServer (Byte)
Ave.	13715374	12457083	568
1	13631488	12373197	568
2	13631488	12478054	568
3	13736346	12478054	568
4	13736346	12478054	568
5	13736346	12478054	568
6	13736346	12373197	568
7	13736346	12478054	568
8	13736346	12478054	568
9	13736346	12478054	568
10	13736346	12478054	568

Table 5-83: Result

AddressSpace	MemoryPool (Byte)	Heap Memory (Byte)	ROM size (Byte)	RAM size (Byte)	Memory Region (Byte)
Sakura	895385.6	13715374	19707803	1828492	147073269.8
pvrserver_as0	40686387.2	12457083	784716	109684	466944
FBServer	0	568	122484	23148	0
Total	41581772.8	26173025	20615003	1961324	147540213.8

Used memory Size	237628037(Byte) = 226.62 (MByte)

(5) Consideration

This results is expected.

5.23.3. Memory utilization in HUD (Head-up display)

(1) Description

Measure RAM utilization in Head-up display application on virtualization PoC using Command prompt.

The following applications are measurement target. Refer to Chapter 4 for details.

- Telltail app
- Display driver

(4) Precondition

- Measure on INTEGRITY on virtualization PoC (Type1)
- Use a command on Command prompt.

(5) How to measure

Calculate a used memory size by measurement for several kinds of memory sizes that each application uses, and adding each.

Memory utilization = MemoryPool + heap memory + ROM/RAM size + Memory Region.

- Measurement of the used memory size in MemoryPool

  Measurement method refers to “Measurement of the unused memory size in MemoryPool” of 5.19.1 .

  Excerpt from results of 5.19.1. Check the following results of the memory size (the left side) of the line of "Telltale".

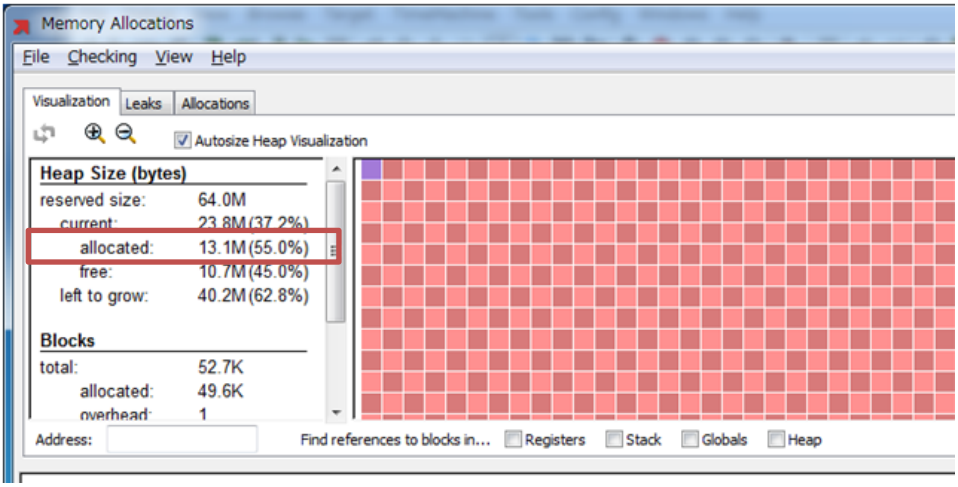
```
DISCOM_sample_virt                        0x0000000000005000/0x0000000000080000
   0xffffffa6c57f8000 pending          127 0x00000000000009d0/0x0000000000008000   1.61% Initial
   0xffffffa00ff26000 pending          254 0x0000000000000280/0x00000000000000dd0   0.00% OSAAgent
```

  Similarly, check the result of "pvrserver_as0" and "FBServer".

- Measurement of the used memory size in Heap Memory

  Measurement method refers to “Measurement of the unused memory size in MemoryPool” of 5.19.1 .

  Excerpt from results of 5.19.1. Check the following results of the memory size (the left side) of the line of "DISCOM_sample_virt".



- Measurement of the used memory size in ROM/RAM size

4. Run the following command.(measurement ROM/RAM size of Sakura)

```
C:\ghs\comp_201516\gsize.exe -ram -rom
```

After finishing a command, you will see the log like below.

Red square is results.

```
C:\ghs\int1144\bin\devtree-arm64\DISCOM sample_virt
        RAM_Size: 285296
        ROM_Size: 1150472
```

5. Run the following command. (measurement ROM/RAM size of FBServer)

```
C:\>C:\ghs\comp_201516\gsize.exe -ram -rom C:\FBServer
```

After finishing a command, you will see the log like below.

Red square is results.

```
C:\>C:\ghs\comp_201516\gsize.exe -ram -rom C:\FBServer
        RAM_Size:          23148
        ROM_Size:         122484
```

- Measurement of the used memory size in ROM/RAM size

The memory region of Telltale use maximum following size.

Memory Region Size : 2334720 (Byte)

(5) Result

- Instrument Cluster
  - DISCOM_sample_virt (Telltale)
  - FBServer (Display driver)

Table 5-84: Used memory Pool

	Telltale (Byte)	FBServer (Byte)
Ave.	20480	0
1	20480	0
2	20480	0
3	20480	0
4	20480	0
5	20480	0
6	20480	0
7	20480	0
8	20480	0
9	20480	0
10	20480	0

Table 5-85: Used heap memory

	Sakura (Byte)	FBServer (Byte)
Ave.	1464	568
1	1536	568
2	1434	568
3	1434	568
4	1434	568
5	1434	568
6	1434	568
7	1434	568
8	1536	568
9	1536	568
10	1434	568

Table 5-86: Result

AddressSpace	MemoryPool (Byte)	Heap Memory (Byte)	ROM size (Byte)	RAM size (Byte)	Memory Region (Byte)
DISCOM_sample_vir	20480	1464	1150472	285296	2334720
FBServer	0	568	122484	23148	0
Total	20480	2036	1272956	308444	2334720

Used memory Size	3938632(Byte) = 3.76 (MByte)

(6) Consideration

This results is expected.

5.24. Network Performance(RTOS, Multivisor)

5.24.1. Send / Receive data to cloud

Out of Scope.

5.24.2. Packet Loss

Out of Scope.

5.24.3. End-to-end Input events delivery latency

Out of Scope.

5.24.4. Delay variation(Jitter)

Out of Scope.

5.24.5. Throughput(Bandwidth)

Out of Scope.

5.24.6. Data Queuing

Out of Scope.

5.24.7. Ethernet Bit error rate (BER)

Out of Scope.

Appendix-A. How to write U-boot or Monolith to Hyper Flash

1. Activate the Terminal Software.
2. Power switch turn on Salvator-X board.
3. Run MULTI.
4. Select [Components] – [Open Debugger…] from menu bar

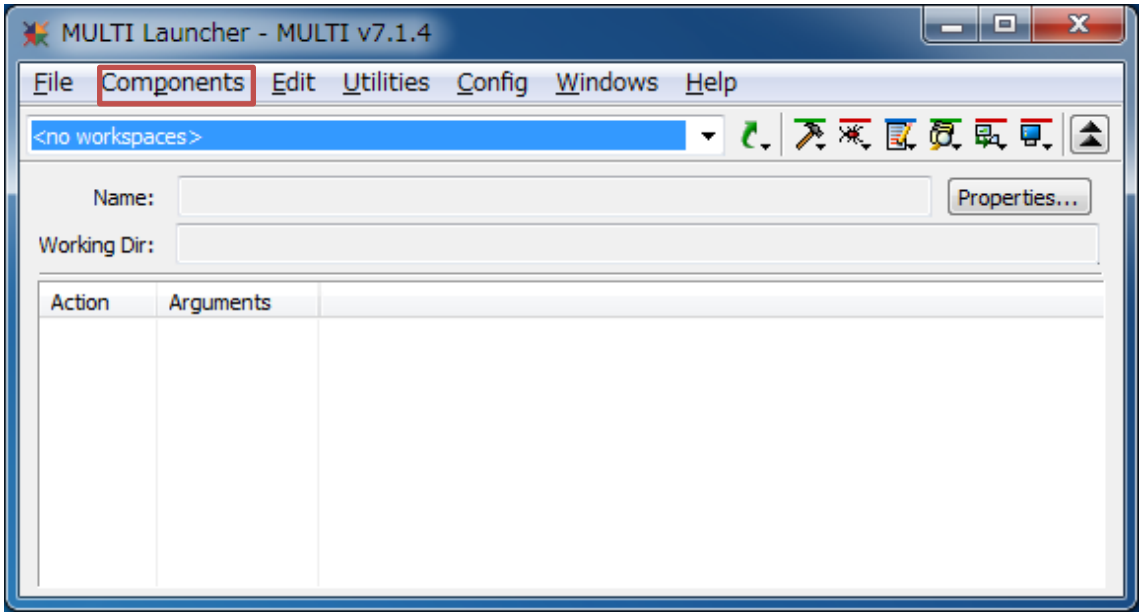


Figure A-1: Selects [Components] – [Open Debugger…]

The following “Choose a program to debug” window appears.

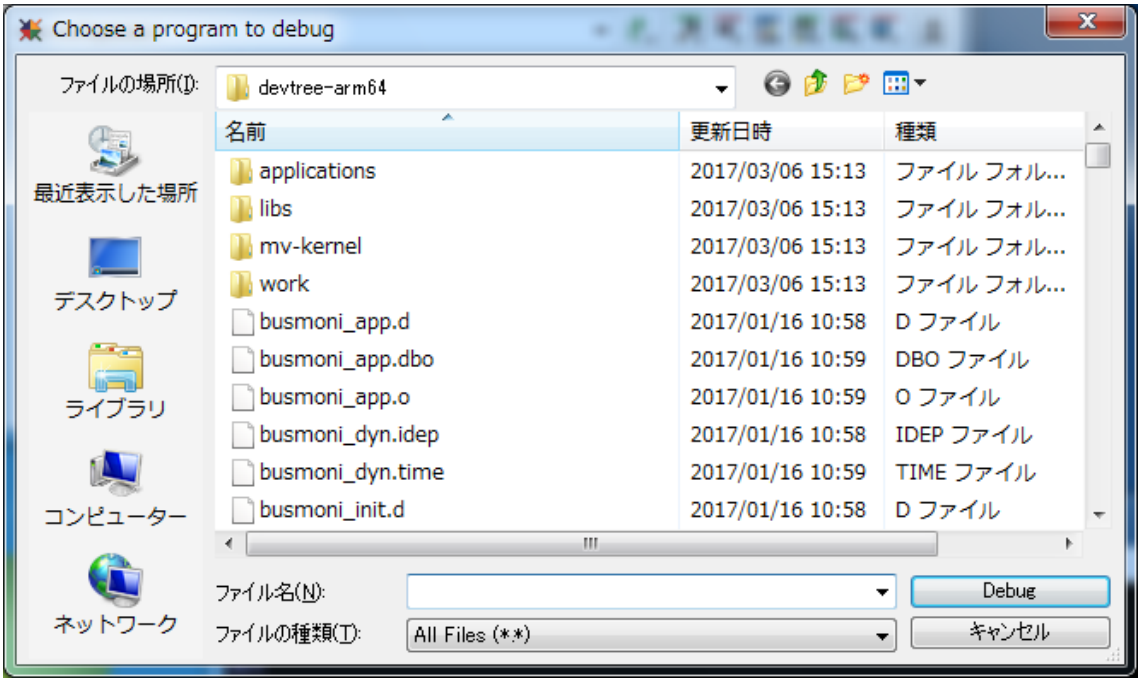


Figure A-2: “Choose a program to debug” window

5. Select the appropriate monolith file and press [Debug] button

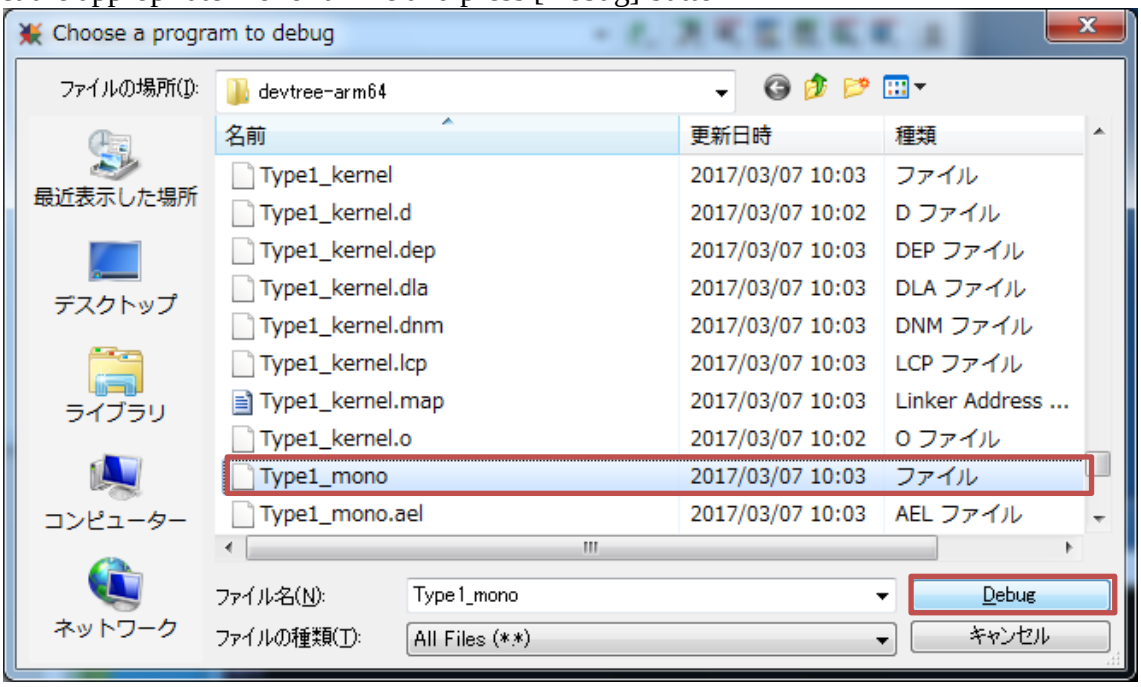


Figure A-3: Select the appropriate monolith file

Please select Type1_mono, Type3_mono or Type4_mono.
The following debug window appears.

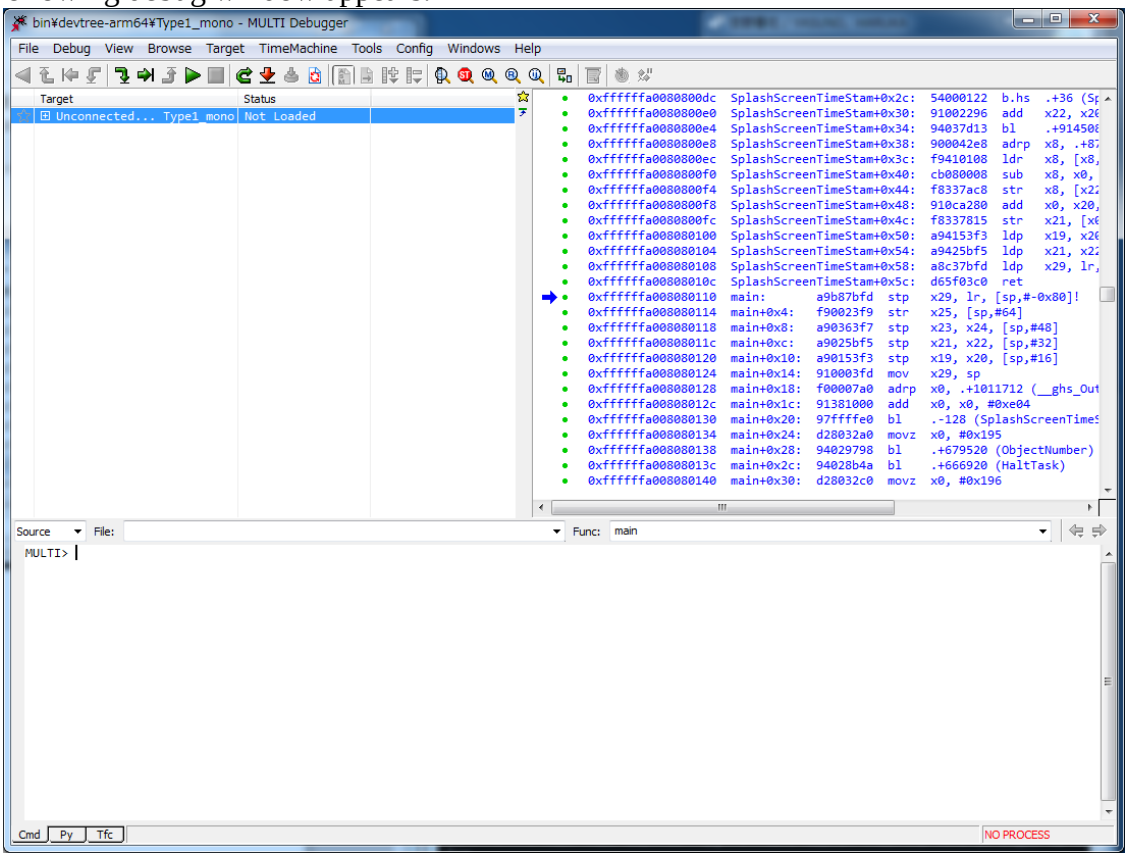


Figure A-4: Debug window

6. Run to following command in MULTI

- In case of writing U-Boot binary to Hyper Flash

Please following command in MULTI window.

```
MULTI > py -b -f int1144/devtree-arm64/rcar/flash-bootable-files.py rcar-uboot ${BOOTLOADER}
```

- In case of writing Monolith binary(Type1_mono.5.4) to Hyper Flash

Please locate integrity-flash.dtb which Renesas provide to int1144/devtree-arm folder
Please following command in MULTI window.

```
MULTI > py -b -f .int1144/devtree-arm64/rcar/flash-bootable-files.py rcar Type1_mono.5.4. ${BOOTLOADER}
```

${BOOTLOADER} means uboot folder path

Appendix-B. How to launch Type1, 3 and 4.

1. Activate the Terminal Software.
2. Power switch turn on Salvator-X board.
3. Run MULTI.
4. Select [Components] – [Open Debugger…] from menu bar

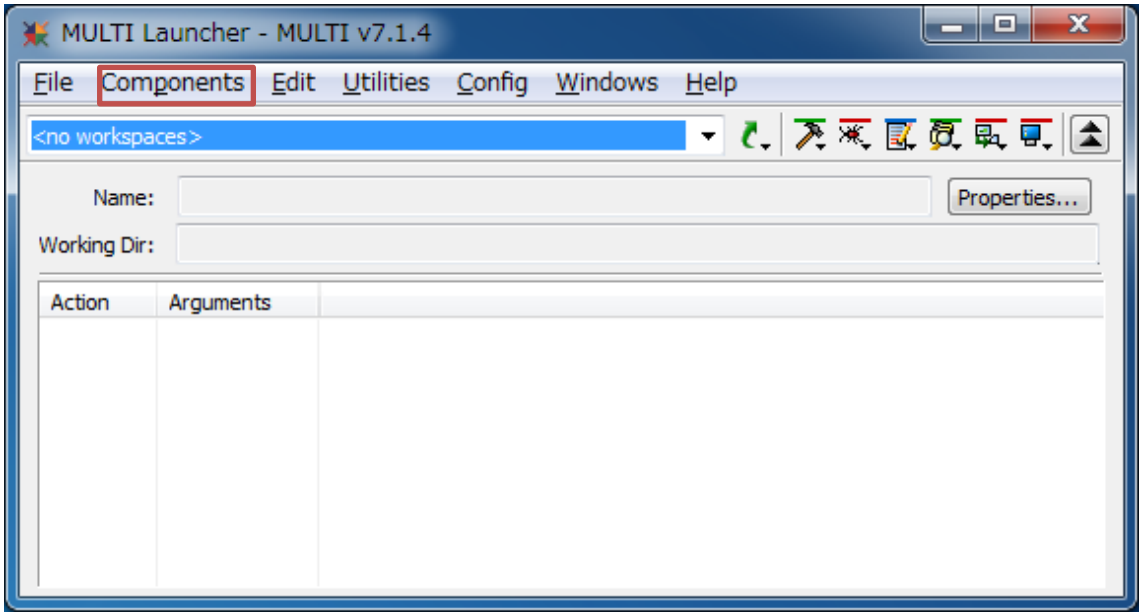


Figure B-1: Selects [Components] – [Open Debugger…]

The following “Choose a program to debug” window appears.

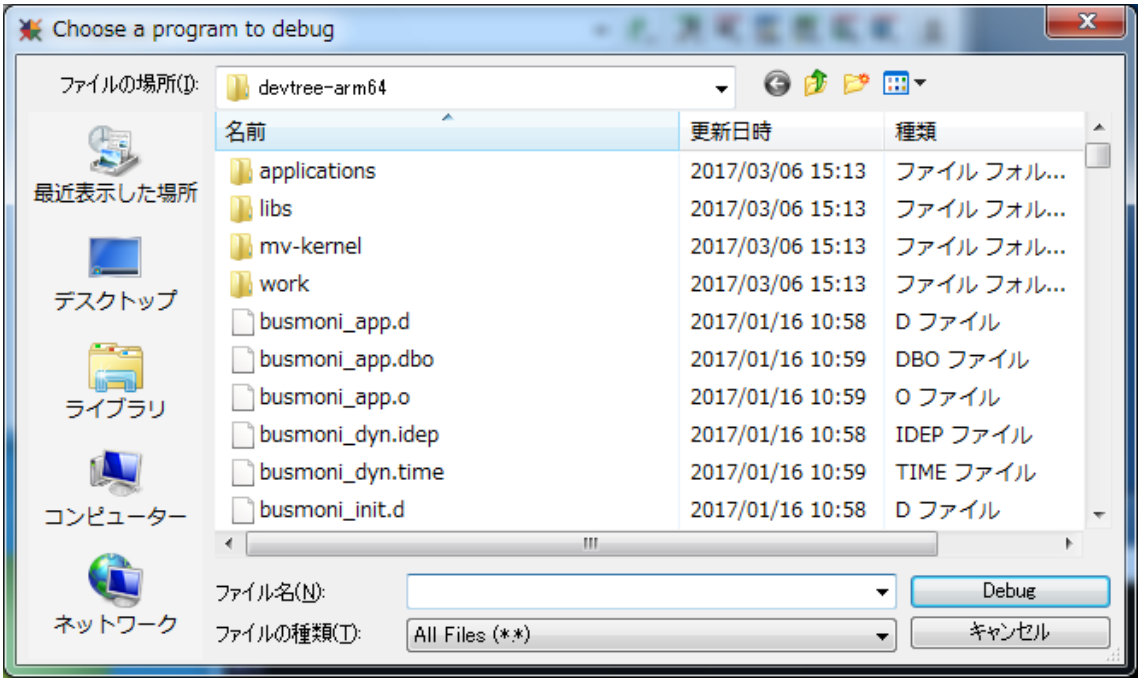


Figure B-2: “Choose a program to debug” window

5. Select the appropriate monolith file and press [Debug] button

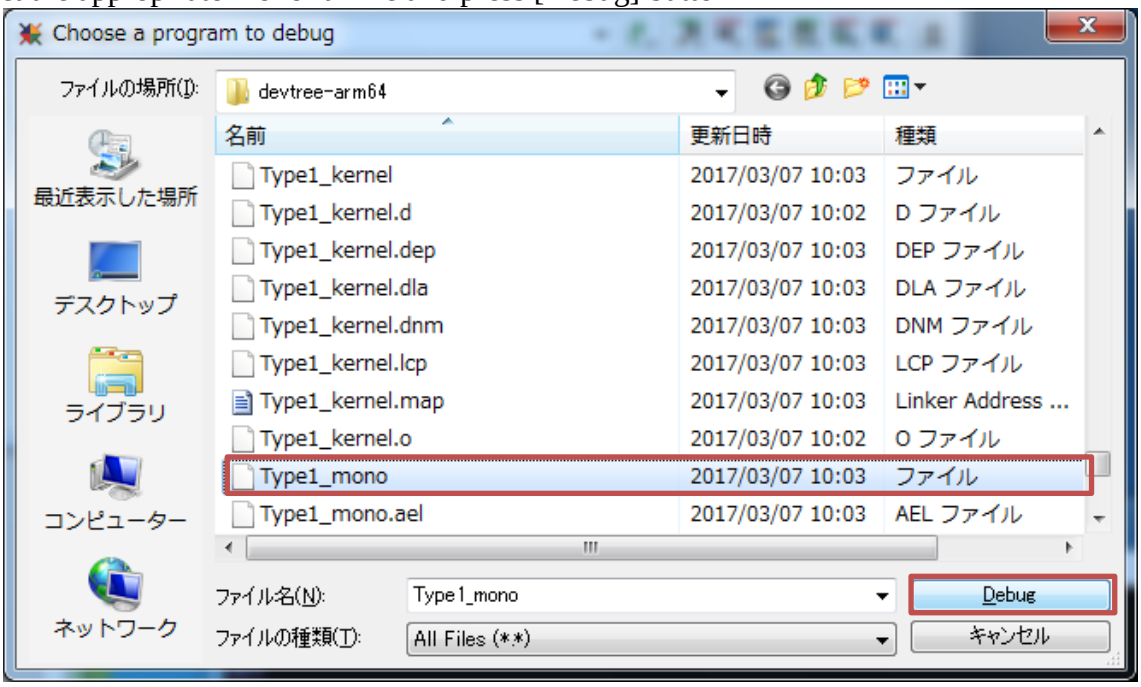


Figure B-3: Select the appropriate monolith file

Please select Type1_mono, Type3_mono or Type4_mono.
The following debug window appears.

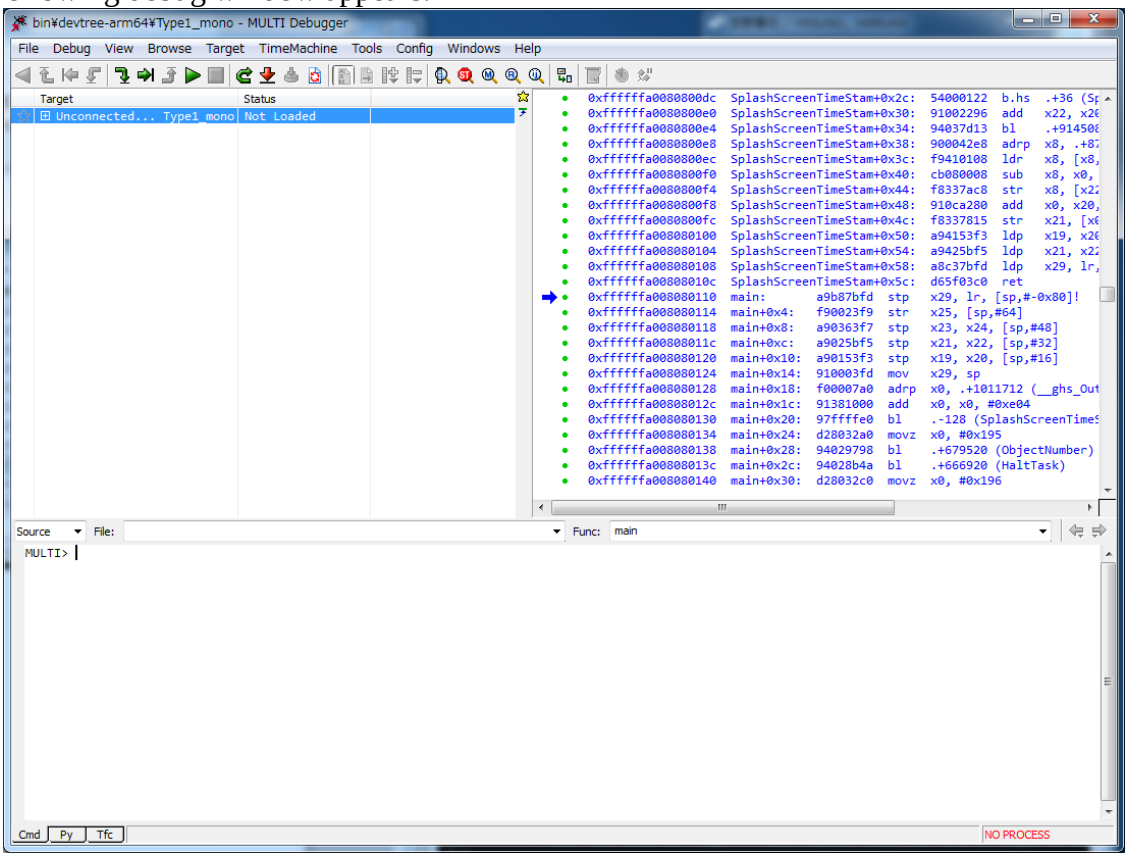


Figure B-4: Debug window

6. Connect to a Target
   Press the [F5] button, select “Green Hills Probe Connection (mpserv) for Renesas R-Car H3/M3 on Salvator-X” and then press the button “Connect”.

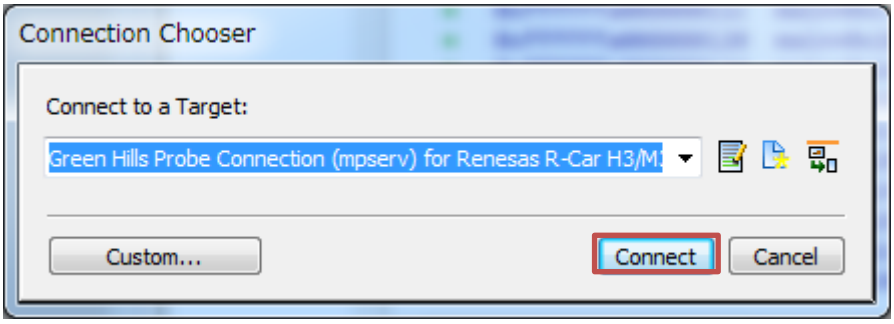


Figure B-5: Connect to a Target

Press the [OK] button on the following window.

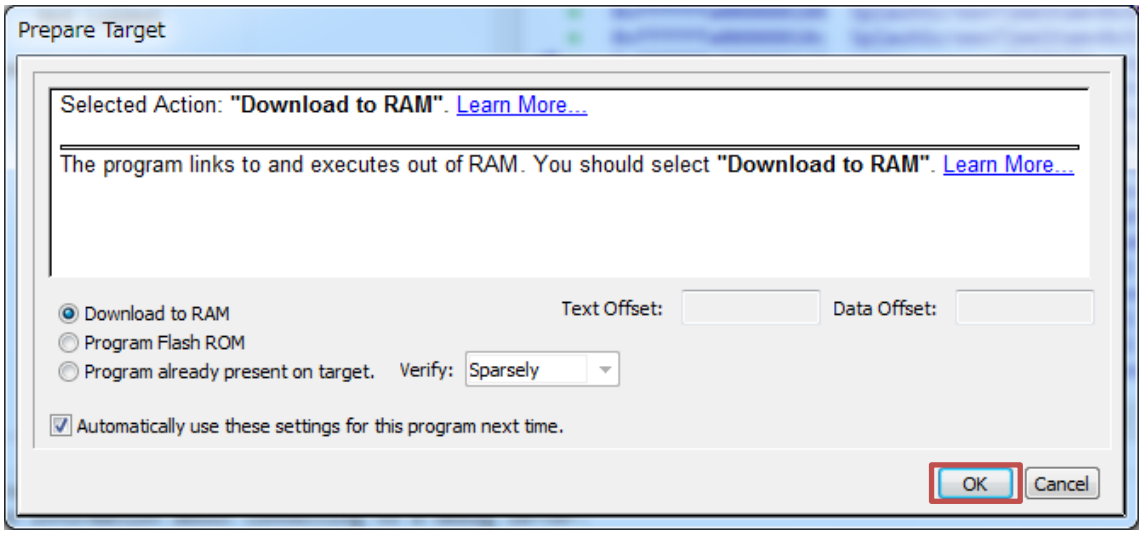


Figure B-6: Prepare Target

The following “Initializing Target” window appears.

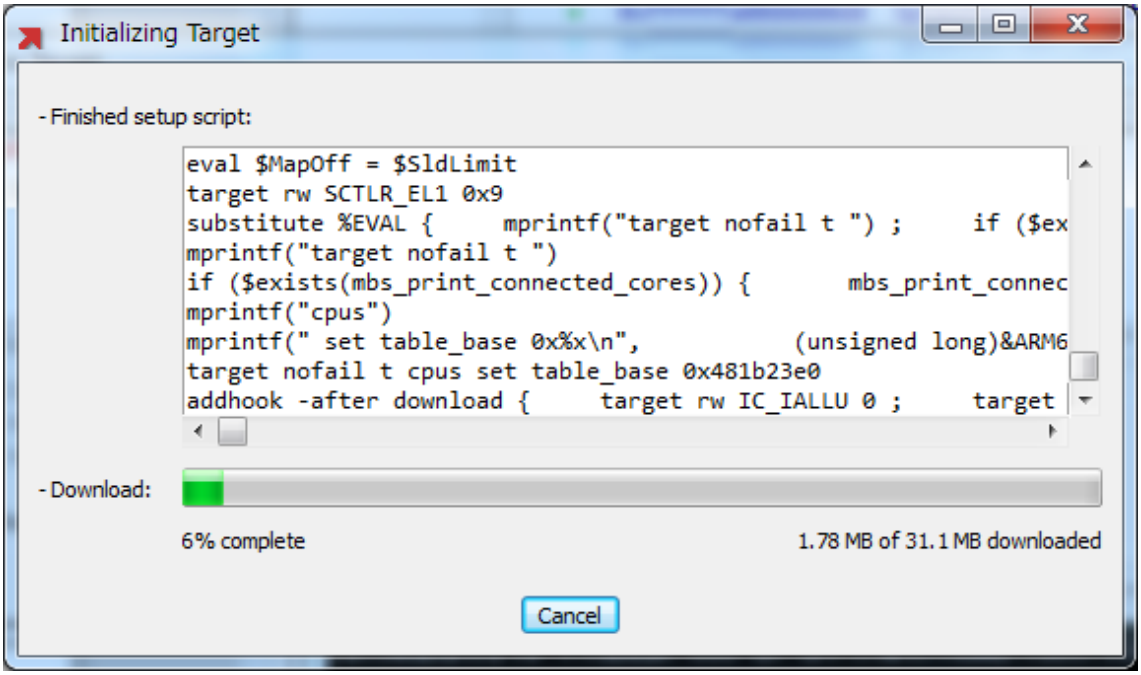


Figure B-7: Initializing Target

After the download completes, finally the following window appears.

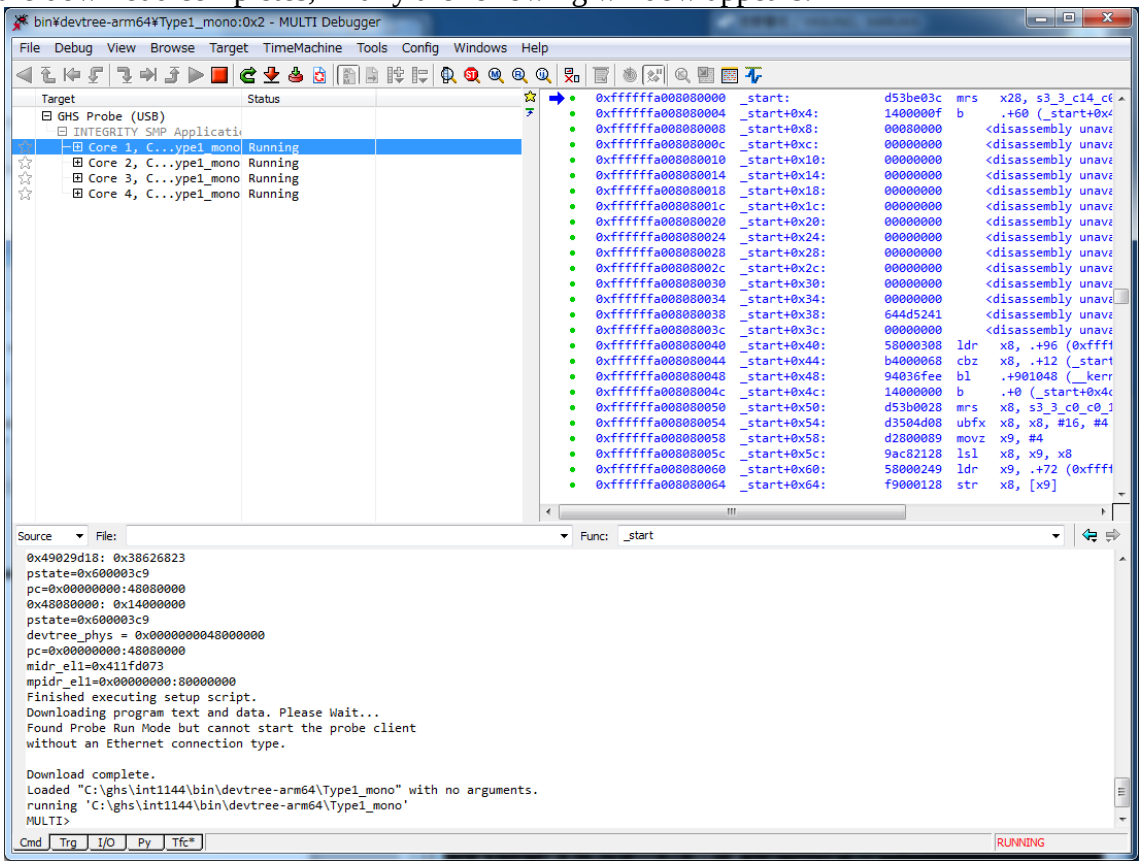


Figure B-8: Running window

INTEGRITY® Virtualization Environment Performance Evaluation Report

Publication Date: Rev.1.00 Feb 28, 2017
Rev.1.10 Mar 17, 2017
Rev.1.20 Mar 21, 2017
Rev.1.30 Apr 20, 2017

Published by: Renesas Electronics Corporation