TI81XX HDMI User Guide

DM816x AM389x DM814x AM387x HDMI User Guide

Linux PSP

IMPORTANT

TI81xx refers to DM816x, DM814x and DM813X.

Read This First

This section show the major updates since last release, please review this section before any SW activities.

- HDMI driver is supported for DM813X(DM385 called in other places) family.

Introduction

HDMI (High Definition Multimedia Interface) is an interface used to transport un-compressed audio / video digital data (along with other control signals). HDMI IP on above class of SoCs implements HDMI Standard Specification 1.3A. Software is not certified for any specifications or compliance. A source is device that generates video / audio content (such as DM816X, DM814x PC, DVD Player, etc…) and sink is a device that consumes the same (such as TV, Monitor, LCD, etc…)

HDMI is controlled transparently from the application through the sysfs entries exposed by the VPSS driver for controlling the displays. HDMI also exposes IOCTLs through standard Linux character driver interface. Control through character driver interface is limited. Users are encouraged to use the sysfs entries for controlling the HDMI like changing of resolution, getting the status like hot plug etc. All of the sysfs entries are explained in later part of the document.

HDMI driver is exposed as the /dev/TI81XX_HDMI under the standard Linux file system and is provide as the Linux loadable module along with the Linux PSP package. HDMI can't be compiled with kernel. It should always be a loadable module. Firmware needs to be loaded using the slave loader before inserting the HDMI Linux module.

Upgrade and Compatibility

Linux character driver interface was provided for the HDMI in all of the earlier releases. Now HDMI is controlled transparently through video driver VPSS sysfs entries. Details about the same is given in the later part of the document. Limited control is provided through the character driver interface. Application using the HDMI will have to now use the SYSFS entries to get the HDMI control. Following are the new features added compared to previous release of the HDMI

- Transperent control of HDMI using the Video driver (VPSS) SYSFS entries.
- Reading of EDID from the sink device
- Hot Plug Detect(HPD) status information.

Build TI81xx on-chip HDMI driver

- Enable TI81xx HDMI in menuconfig

```
$ make ARCH=arm CROSS_COMPILE=PATH_TO_TOOLCHAIN/bin/arm-none-linux-gnueabi- ti8168_evm_defconfig
$ make ARCH=arm CROSS_COMPILE=PATH_TO_TOOLCHAIN/bin/arm-none-linux-gnueabi- menuconfig
```

select Device Drivers

```
   Userspace binary formats  --->
   Power management options  --->
   [*] Networking support  --->
 " Device Drivers  --->    "
   CBUS support  --->
   File systems  --->
```

select graphics support:

```
   Multifunction device drivers  --->
   [ ] Voltage and Current Regulator Support  --->
   < > Multimedia support  --->
 " Graphics support  --->     "
   <*> Sound card support  --->
   [*] HID Devices  --->
```

Select TI81XX HDMI Driver Module (EXPERIMENTAL)

```
   < > Fujitsu MB862xx GDC support
   < > E-Ink Broadsheet/Epson S1D13521 controller support
   < > OMAP frame buffer support (EXPERIMENTAL)
 " <M> TI81XX HDMI Driver Module (EXPERIMENTAL)  ---->"
   < > TI81XX Video Processing Subsystem (EXPERIMENTAL)  --->
   [ ] Backlight & LCD device support  --->
```

Features Supported

Character driver Interface

Support standard Linux character driver control (IOCTL) to provide following features

- Start / Stop HDMI
- Read EDID from the connected Sink
- Query status like hot plug detect, driver started etc.

SYSFS interface

Support standard sysfs to provide following features

- Set the display mode. Supported modes are 1080P60, 1080P30, 1080I60 and 720P60
- Start / stop HDMI
- Read sinks EDID information and display small string

Note :Changing of the resolution on HDMI also requires changing the resolution on encoder. Encoder needs to be disabled before changing the resolution on HDMI. Further all the connected display/graphics path on that Encoder needs to be stopped.

Features not verified

In this release, following features are not supported

- Deep color mode of operation

Supported System Calls

Supported standard system calls are

- *open ()* - To open an instance of HDMI device driver
- *close ()* - To close an instance of HDMI device driver
- *ioctl ()* - To control the driver operations.

Module Parameter

HDMI driver module when 'insmod' can disable/enable the debug logs using the switch *debug=0/1*

Supported display modes

- 1080P60
- 1080I60
- 720P60
- 1080P30

Application Interface

- open
  - Standard character drivers open call, the mode flags of the call is ignored by the driver.
- close
  - Standard character drivers close call.
- ioctl
    - TI81XXHDMI_GET_STATUS- Get current status, whether streaming or not, whether a sync detected or not.
    - TI81XXHDMI_START- Used to turn ON streaming / start display
    - TI81XXHDMI_STOP - Used to turn OFF streaming / stop display
    - TI81XXHDMI_READ_EDID - Used to read the EDID of the HDMI sync.

Dependencies

- Pixel Clock
  - The pixel clock is provided by the VIDEO PLL module, this modules has to be configured to meet the pixel rate required for given resolution. Its beyond the scope of this driver to configure this. If this driver is used along with HDVPSS driver provided by TI, HDVPSS driver ensure that pixel clock is configured for the chosen resolution. The resolution selected on HDMI should match with resolution configured of DISPLAY Driver. (In case of FBDEV, inserting the drivers/modules and connecting graphics pipeline to appropriate VENC does this. Please refer FBDEV user guide for details)
- Discrete Syncs only
  - HDMI expects the VENC to provide discrete syncs

- HDMI driver with default configuration, expects the VENC to provide video data in RGB color space in either 24 bits/pixel or 30 bits/pixel

Typical Sequence

Load the Modules in the following order.

- Load VPSS module, (This module configures and starts the *display 0* VENC in 1080p-60 mode.) :

```
insmod vpss.ko
```

- Load fbdev module :

```
insmod ti81xxfb.ko
```

- Load on-chip HDMI module :

```
insmod ti81xxhdmi.ko
```

To change the mode of the Encoder on which HDMI is connected as well as of the HDMI

- Disable Venc

```
echo 0 > /sys/devices/platform/vpss/display0/enabled
```

- Change mode

```
echo 720p-60 > /sys/devices/platform/vpss/display0/mode
```

- Enable Venc

```
echo 1 > /sys/devices/platform/vpss/display0/enabled
```

- Example to configure the HDMI driver and start streaming of fbdev output through on-chip HDMI is explained in the application **saFbdevHdmiDisplay.c**
- VPSS Video User guide at [| DM814X VPSS VideoUserGuide [2]][| DM816X VPSS VideoUserGuide [3]]
- Before changing the mode in the HDMI driver user needs to set the PLL clock and Venc mode accordingly. This is explained at VPSS Video User Guide shown in above.

EDID

Reading of the EDID from the sink is supported through character driver interface using TI81XXHDMI_READ_EDID. Basic informataion from the EDID is also displayed as a part of syfs entry. Following is the sysfs entry for reading the basics information from EDID

```
cat /sys/devices/platform/vpss/display0/edid
```

EDID

Reading of the EDID from the sink is supported through character driver interface using TI81XXHDMI_READ_EDID. Basic informataion from the EDID is also displayed as a part of syfs entry. Following is the sysfs entry for reading the basics information from EDID

```
cat /sys/devices/platform/vpss/display0/edid
```

TI81XX HDMI Audio Support

Note: TI81XX HDMI audio features are supported from PSP_04.00.01.13 Release onwards. Please Make sure that all VPSS modules [| DM814X VPSS VideoUserGuide [4]] or [| DM816X VPSS VideoUserGuide [3]]are inserted and HDMI Video is enable before testing audio

All the sample rate and audio formats depends on the HDMI TV's capabilities.

Features

This section describes the features supported by ALSA SoC HDMI Audio driver.

- Supports HDMI audio codec on TI81XX in ALSA SoC framework.
- Multiple sample rates support (32KHz, 44.1KHz, 48KHz, 96KHz and 192KHz)playback.
- Supports audio 16bit and 24bit (S16_LE ,S24_LE) audio format.
- Supports audio in stereo mode.
- Supports all audio mode in 1080P@60,1080I@60,720P@60 and 480P@60 video modes

Configuration

To enable/disable audio support, start the *Linux Kernel Configuration* tool:

```
$ make menuconfig
```

Select *Device Drivers* from the main menu.

```
...
    ...
    Power management options --->
    [ ] Networking support --->
    Device Drivers --->
    File systems --->
    Kernel hacking --->
    ...
    ...
```

Select *Sound card support* as shown here:

```
...
    ...
    Multimedia devices --->
    Graphics support --->
<*> Sound card support --->
[*] HID Devices --->
[*] USB support --->
    ...
    ...
```

Select *Advanced Linux Sound Architecture* as shown here:

```
--- Sound card support
<*> Advanced Linux Sound Architecture --->
< > Open Sound System (DEPRECATED) --->
```

Select *ALSA for SoC audio support* as shown here:

```
...
    ...
[*] ARM sound devices --->
[*] SPI sound devices --->
<*> ALSA for SoC audio support --->
```

For TI816x, select *On-chip HDMI audio support for TI81XX EVM* as shown here:

```
--- ALSA for SoC audio support
<*> SoC Audio for the TI81XX chip
<*> SoC Audio support for TI81XX EVM
<*> On-chip HDMI audio support for TI81XX EVM
< > Build all ASoC CODEC drivers (NEW)
```

Note: Soc Audio support for TI81XX EVM option appears if SoC Audio for the TI816X chips is selected Note: Currently TI814x menu options appear with TI81XX instead of TI816x mentioned here.

Note: Only TI816X PG2.0 silicon supports **Auto Mode** configurations and the rest all are is using the **Software/Manual Mode** for generate the ACR packets.

HDMI Auto Mode Configuration

In Auto – mode. An audio clock is provided to HDMI controller, N value is programmed and CTS is computed automatically by the HDMI core. ACR packets are sent out periodically to the Rx using the following relation: MCLK=(128 or 256)*Fs = TMDS*N/CTS

TI816X PG2.0 supports Auto Mode by using the McBSP clock as the HDMI input MCLK. The MCLK frequency need to programs according to the Audio sampling rate requirements. PSP4.00.02.14 release used sysclk22 as the parent of the McBSP clock. The parent clock must not be used for any other modules. User can modify the parent clock by changing the a macro in the HDMI Audio driver and recompile the kernel to achieve this.

- Open the file *sound/soc/davinci/davinci-hdmi.c* and modify the macro shown below .

```
#define HDMI_MCBSP_CLK_PARENT   "sysclk22_ck"   /* select one: sysclk20_ck, sysclk21_ck, sysclk22_ck*/
```

Note: Clock get will fail if the parent clock's use-count is not 0

Load On-Chip HDMI Module

On-Chip HDMI hardware is controlled through another kernel driver, which should be loaded after vpss.ko is loaded in order to work the HDMI audio. Please refer | HDMI user guide [5]for details.

Prerequisites

1. syslink kernel driver module, referred as "syslink.ko"
2. loader user space program, referred as "slaveloader"
3. M3 BIOS Firmware binary, referred as "ti816x_hdvpss.xem3"
4. VPSS kernel driver module, referred as "vpss.ko"
5. FBDev kernel driver module, referred as "ti81xxfb.ko"
6. V4L2 display kernel module, referred as "ti81xxvo.ko"
7. On-Chip HDMI kernel module, referred as "ti81xxhdmi.ko"

TI814X Please use the following firmware for ti814x

1. M3 BIOS Firmware binary, referred as "ti814x_hdvpss_512M or ti814x_hdvpss_1G.xem3"

Please refer [DM816X VPSS VideoUserGuide [6]] for more information about building the above modules

Note: notifyk.vpssm3_sva=0xA0000000 must be added to the bootargs.It is required to use syslink notify driver in kernel. This address is valid for DM816X/AM389X platforms. Refer [DM816X VPSS VideoUserGuide [7]] for more details. For ti814x_hdvpss_1G.xem3 use notifyk.vpssm3_sva=0xA0000000 must be added to the bootargs

- Load VPSS and HDMI Module

```
$ insmod syslink.ko
$ ./slaveloader startup VPSS-M3 ti816x_hdvpss.xem3
$ insmode vpss.ko
$ insmod ti81xxfb.ko
$ insmod ti81xxvo.ko
$ insmod ti81xxhdmi.ko
```

Enable HDMI Display

- Disable Venc

```
echo 0 > /sys/devices/platform/vpss/display0/enabled
```

- Select mode

```
echo 720p-60 > /sys/devices/platform/vpss/display0/mode
```

- Enable Venc

```
echo 1 > /sys/devices/platform/vpss/display0/enabled
```

Note: Available audio sample rate support depends upon the pixel clock frequency, the video format timing, and whether or not content protection re-synchronization is needed. Please restart the audio while changing the video resolution. Refer the Feature list for the supported video resolution and audio sampling rate.

ALSA commands

Display ALSA devices

```
$ aplay -l
```

```
**** List of PLAYBACK Hardware Devices ****
  card 0: EVM [TI81XX EVM], device 0: AIC3X tlv320aic3x-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: EVM [TI81XX EVM], device 1: hdmi HDMI-DAI-CODEC-1 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Audio Playback Test

```
 $ aplay -Dhw:0,1 <filename>
```

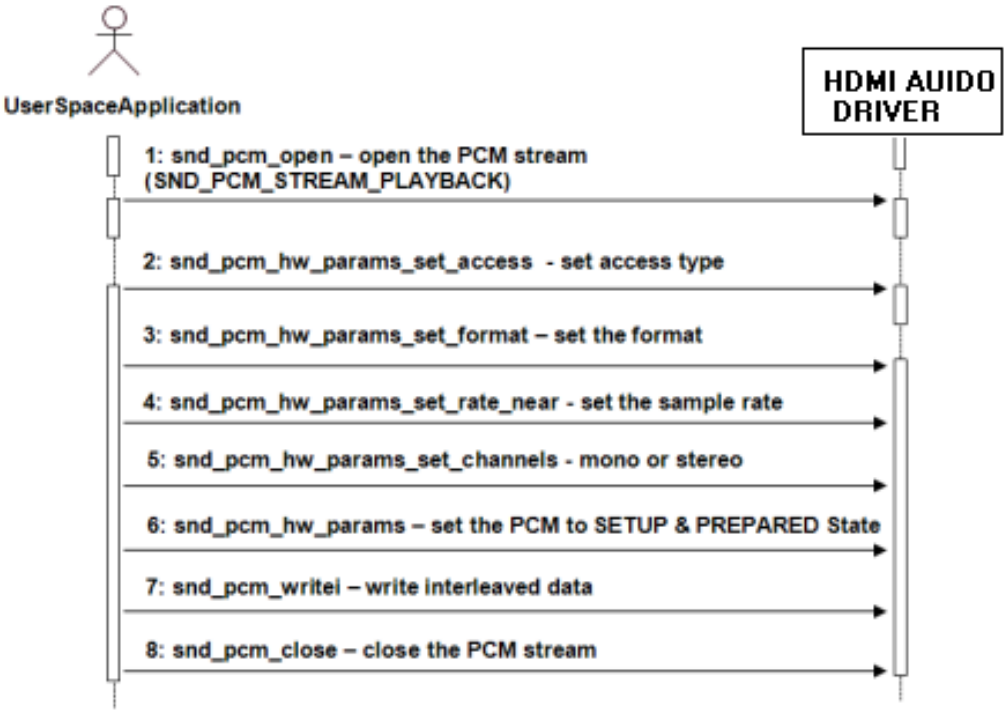
Amixer commands

ALSA mixer commands are not supported by the Hardware.

Note: ALSA softvol plug-in can be used for user level volume control Please find the usage in ALSA wiki http://alsa.opensrc.org/Softvol

User Space Interactions

This section depicts the sequence of operations for a simple playback.



Sample Applications

This chapter describes the audio sample applications provided along with the package. The source for these sample applications are available in the Examples directory of the Release Package folder.

Introduction

Writing an audio application involves the following steps:

- Opening the audio device
- Set the parameters of the device
- Receive audio data from the device or deliver audio data to the device
- Close the device

These steps are explained in detail in this section.

Note

User space ALSA libraries can be downloaded from this link http://www.alsa-project.org/main/index.php/Download [8]. User needs to build and install them before he starts using the ALSA based applications.

A minimal playback application

This program opens an audio interface for playback, configures it for stereo, 16 bit, 44.1kHz, interleaved conventional read/write access. Then its delivers a chunk of random data to it, and exits. It represents about the simplest possible use of the ALSA Audio API, and isn't meant to be a real program.

Opening the audio device

To write a simple PCM application for ALSA, we first need a handle for the PCM device. Then we have to specify the direction of the PCM stream, which can be either playback or capture. We also have to provide some information about the configuration we would like to use, like buffer size, sample rate, pcm data format. So, first we declare:

```
#include <stdio.h>
#include <stdlib.h>
#include <alsa/asoundlib.h>


#define BUFF_SIZE 4096


int main (int argc, char *argv[])
{
  int err;
  short buf[BUFF_SIZE];
  int rate = 44100; /* Sample rate */
  unsigned int exact_rate; /* Sample rate returned by */
  /* Handle for the PCM device */
  snd_pcm_t *playback_handle;
  /* Playback stream */
  snd_pcm_stream_t stream = SND_PCM_STREAM_PLAYBACK;
  /* This structure contains information about the hardware and can be used to specify the configuration to be used for */
  /* the PCM stream. */
  snd_pcm_hw_params_t *hw_params;
```

The most important ALSA interfaces to the PCM devices are the "plughw" and the "hw" interface. If you use the "plughw" interface, you need not care much about the sound hardware. If your sound card does not support the sample rate or sample format you specify, your data will be automatically converted. This also applies to the access type and the number of channels. With the "hw" interface, you have to check whether your hardware supports the configuration you would like to use. Otherwise, user can use the default interface for playback by:

```
/* Name of the PCM device, like plughw:0,0 */
  /* The first number is the number of the soundcard, the second number is the number of the device.  HDMI codec is detected as 2nd device*/


  static char *device = "plughw:0,1"; /* playback device */
```

Now we can open the PCM device:

```
/* Open PCM. The last parameter of this function is the mode. */
  if ((err = snd_pcm_open (&playback_handle, device, stream, 0)) < 0) {
    fprintf (stderr, "cannot open audio device (%s)\n", snd_strerror (err));
    exit (1);
  }
```

Setting the parameters of the device

Now we initialize the variables and allocate the `hwparams` structure:

```
/* Allocate the snd_pcm_hw_params_t structure on the stack. */
  if ((err = snd_pcm_hw_params_malloc (&hw_params)) < 0) {
    fprintf (stderr, "cannot allocate hardware parameters (%s)\n", snd_strerror (err));
    exit (1);
  }
```

Before we can write PCM data to the soundcard, we have to specify access type, sample format, sample rate, number of channels, number of periods and period size. First, we initialize the hwparams structure with the full configuration space of the soundcard:

```
/* Init hwparams with full configuration space */
  if ((err = snd_pcm_hw_params_any (playback_handle, hw_params)) < 0) {
    fprintf (stderr, "cannot initialize hardware parameter structure (%s)\n", snd_strerror (err));
    exit (1);
  }
```

Now configure the desired parameters. For this example, we assume that the soundcard can be configured for stereo playback of 16 Bit Little Endian data, sampled at 44100 Hz. Therefore, we restrict the configuration space to match this configuration only. The access type specifies the way in which multi-channel data is stored in the buffer. For INTERLEAVED access, each frame in the buffer contains the consecutive sample data for the channels. For 16 Bit stereo data, this means that the buffer contains alternating words of sample data for the left and right channel.

```
/* Set access type. */
  if ((err = snd_pcm_hw_params_set_access (playback_handle, hw_params, SND_PCM_ACCESS_RW_INTERLEAVED)) < 0) {
    fprintf (stderr, "cannot set access type (%s)\n", snd_strerror (err));
    exit (1);
  }

  /* Set sample format */
  if ((err = snd_pcm_hw_params_set_format (playback_handle, hw_params, SND_PCM_FORMAT_S16_LE)) < 0) {
    fprintf (stderr, "cannot set sample format (%s)\n", snd_strerror (err));
    exit (1);
  }

  /* Set sample rate. If the exact rate is not supported by the hardware, use nearest possible rate. */
  exact_rate = rate;
  if ((err = snd_pcm_hw_params_set_rate_near (playback_handle, hw_params, &exact_rate, 0)) < 0) {
    fprintf (stderr, "cannot set sample rate (%s)\n", snd_strerror (err));
    exit (1);
  }

  if (rate != exact_rate) {
    fprintf(stderr, "The rate %d Hz is not supported by your hardware.\n ==> Using %d Hz instead.\n", rate, exact_rate);
  }

  /* Set number of channels */
  if ((err = snd_pcm_hw_params_set_channels (playback_handle, hw_params, 2)) < 0) {
```

```
    fprintf (stderr, "cannot set channel count (%s)\n", snd_strerror (err));
    exit (1);
  }
```

Now we apply the configuration to the PCM device pointed to by `pcm_handle` and prepare the PCM device.

```
  /* Apply HW parameter settings to PCM device and prepare device. */
  if ((err = snd_pcm_hw_params (playback_handle, hw_params)) < 0) {
    fprintf (stderr, "cannot set parameters (%s)\n", snd_strerror (err));
    exit (1);
  }

  snd_pcm_hw_params_free (hw_params);

  if ((err = snd_pcm_prepare (playback_handle)) < 0) {
    fprintf (stderr, "cannot prepare audio interface for use (%s)\n", snd_strerror (err));
    exit (1);
  }
```

Writing data to the device

After the PCM device is configured, we can start writing PCM data to it. The first write access will start the PCM playback. For interleaved write access, we use the function:

```
/* Write some junk data to produce sound. */
  if ((err = snd_pcm_writei (playback_handle, buf, BUFF_SIZE/2)) != BUFF_SIZE/2) {
    fprintf (stderr, "write to audio interface failed (%s)\n", snd_strerror (err));
    exit (1);
  } else {
    fprintf (stdout, "snd_pcm_writei successful\n");
  }
```

After the PCM playback is started, we have to make sure that our application sends enough data to the soundcard buffer. Otherwise, a buffer under-run will occur. After such an under-run has occurred, `snd_pcm_prepare` should be called.

Closing the device

After the data has been transferred, the device needs to be closed by calling:

```
  snd_pcm_close (playback_handle);

  exit (0);
}
```

References

[1] http://creativecommons.org/licenses/by-sa/3.0/

[2] http://processors.wiki.ti.com/index.php/DM814X_C6A814X_AM387X_VPSS_Video_Driver_User_Guide_PSP_04.01.00.05

[3] http://processors.wiki.ti.com/index.php/DM816X_C6A816X_AM389X_VPSS_Video_Driver_User_Guide_PSP_04.00.00.12

[4] http://processors.wiki.ti.com/index.php/DM814X_C6A814X_AM387X_VPSS_Video_Driver_User_Guide_PSP_04.01.00.05#Load_VPSS_and_Fbdev_Driver_Modules

[5] http://processors.wiki.ti.com/index.php/DM816x_AM389x_HDMI_User_Guide

[6] http://processors.wiki.ti.com/index.php/DM816X_AM389X_VPSS_Video_Driver_User_Guide_PSP_04.00.00.12#Load_VPSS.2C_Fbdev_and_V4L2_Display_Driver_Modules

[7] http://processors.wiki.ti.com/index.php/DM816X_AM389X_VPSS_Video_Driver_User_Guide_PSP_04.00.00.12

[8] http://www.alsa-project.org/main/index.php/Download

Article Sources and Contributors

TI81XX HDMI User Guide *Source*: http://processors.wiki.ti.com/index.php?oldid=96242 *Contributors*: Deepu.raj, HardikShah, Parth.saxena, SriramAG, Yihe

Image Sources, Licenses and Contributors

Image:TIBanner.png *Source*: http://processors.wiki.ti.com/index.php?title=File:TIBanner.png *License*: unknown *Contributors*: Nsnehaprabha

File:Alsa-hdmi-usecase.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Alsa-hdmi-usecase.png *License*: unknown *Contributors*: Deepu.raj

License

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.