

Enabling the GuC/HuC Firmware for Linux* on New Intel GPU Platforms

Advanced Media Feature Enabling Application Note

February 2019



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

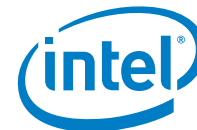
Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: <http://www.intel.com/design/literature.htm>

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at <http://www.intel.com/> or from the OEM or retailer.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright© 2019, Intel Corporation. All rights reserved.



Contents

1.0 Introduction	5
1.1 Terminology	5
1.2 Background.....	5
2.0 GuC/HuC Status Check and Enablement	6
2.1 Preliminaries	6
2.1.1 What is GuC/HuC.....	6
2.1.2 Where to Download.....	6
2.2 Check the GuC/HuC Load Status.....	6
2.3 How to Enable the GuC/HuC	7
2.3.1 Download Platform Firmware Files from www.git.kernel.org	7
2.3.2 Enable GuC/HuC Firmware Loading.....	9
3.0 Final Results	11
Appendix – Enable Low Power (EU-less) Encoding on Gen9+ Intel HD Graphics ..	12

Figures

Figure 1. GuC and HuC Status	7
Figure 2. Local Firmware Path and Download Site.....	8
Figure 3. Download the Firmware File by Clicking “plain”	8
Figure 4. Modify the Kernel Parameters	9
Figure 5. Generate a New Grub Configuration File	10
Figure 6. Update “initramfs”	10
Figure 7. GuC/HuC Successful Loading Status	11
Figure 8. Failure Caused by Lack of GuC/HuC	12
Figure 9. GPU Usage of Normal AVC Encoding	13
Figure 10. GPU Usage of Low Power AVC Encoding	13

Tables

Table 1. Terminology	5
Table 2. Platforms	9
Table 3. Kernel Parameters	9



Revision History

Date	Revision	Description
February 2019	1.0	Initial release.

§



1.0 Introduction

1.1 Terminology

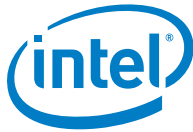
Table 1. Terminology

Term	Description
MSDK	Intel® Media SDK
Gen9	The 9 th Generation GPU Architecture
EU	Execution Unit Compute/Render Engine in Intel GPU

1.2 Background

The new generations of the Intel graphics hardware use firmware that have power, performance benefits, and functionalities, such as scheduling and media offloading. Some advanced GPU features (e.g., low power/EU- less than H.264 encoding in Gen9 and higher GPU platforms) cannot be achieved if the GuC and HuC lack support.

It is important for users to understand how to enable and check the firmware status before using it. The 9th platform of the Intel® Core™ i5-6600K Processor will be used in this paper to enable the GuC HuC on an Ubuntu* 16.04 system.



2.0 GuC/HuC Status Check and Enablement

2.1 Preliminaries

2.1.1 What is GuC/HuC

GuC is designed to perform graphics workload scheduling on the various graphics parallel engines. In this scheduling model, the host software submits work through one of the 256 graphics doorbells. This invokes the scheduling operation on the appropriate graphics engine. Scheduling operations include determining which workload to run next, submitting a workload to a command streamer, pre-empting existing workloads running on an engine, monitoring progress, and notifying the host software when work is done.

HuC is designed to offload some of the media functions from the CPU to GPU. These include bitrate control and header parsing. For example, in the case of bitrate control, the driver invokes the HuC in the beginning of each frame encoding pass. The encode bitrate is adjusted by the calculation from HuC. Both the HuC hardware and the encode hardware reside in the GPU. Using the HuC will save unnecessary CPU-GPU synchronization.

2.1.2 Where to Download

The firmware for the Intel Linux* Graphics is available on www.git.kernel.org. It is sorted by the three letter product code of the processor (e.g., for the Kaby Lake GuC, it may be kbl_guc_ver9_14.bin). The i915 firmware download site for the Linux* Graphics can be found on:

<https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git/tree/i915>

2.2 Check the GuC/HuC Load Status

Run these commands to check the load status of the GuC/HuC firmware:

```
GuC: sudo cat /sys/kernel/debug/dri/0/i915_guc_load_status
```

```
HuC: sudo cat /sys/kernel/debug/dri/0/i915_huc_load_status
```

As shown in Figure 1, both the GuC and HuC are not loaded in this system (with 4.18 kernel).

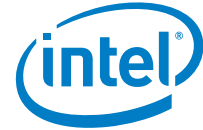


Figure 1. GuC and HuC Status

```

Terminal
root@intel-dbg: /home/intel
root@intel-dbg:/home/intel# uname -r
4.18.0-041800-generic
root@intel-dbg:/home/intel# cat /sys/kernel/debug/dri/0/i915_guc_load_status
GuC firmware: i915/skl_guc_ver9_33.bin
status: fetch NONE, load NONE
version: wanted 9.33, found 0.0
header: offset 0, size 0
uCode: offset 0, size 0
RSA: offset 0, size 0

GuC status 0x00000001:
  Bootrom status = 0x0
  uKernel status = 0x0
  MIA Core status = 0x0

Scratch registers:
  0: 0x0
  1: 0x0
  2: 0x0
  3: 0x0
  4: 0x0
  5: 0x0
  6: 0x0
  7: 0x0
  8: 0x0
  9: 0x0
 10: 0x0
 11: 0x0
 12: 0x0
 13: 0x0
 14: 0x0
 15: 0x0
root@intel-dbg:/home/intel# cat /sys/kernel/debug/dri/0/i915_huc_load_status
HuC firmware: i915/skl_huc_ver01_07_1398.bin
status: fetch NONE, load NONE
version: wanted 1.7, found 0.0
header: offset 0, size 0
uCode: offset 0, size 0
RSA: offset 0, size 0

HuC status 0x00006000:
root@intel-dbg:/home/intel#

```

2.3 How to Enable the GuC/HuC

2.3.1 Download Platform Firmware Files from www.git.kernel.org

The GPU firmware files should be stored in `/lib/firmware/i915`. Ensure that all platform-related files have been downloaded and placed in the directory shown in Figure 2.



Figure 2. Local Firmware Path and Download Site

The terminal window shows the local firmware path: `root@intel-dbg:/home/intel# ll /lib/firmware/i915/`. The output lists various firmware files with their sizes and permissions. A red box highlights the file `skl_huc_ver02_00_1810.bin` with a size of 218688 bytes. The web browser window shows the download site: `https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git`. The file list on the right side of the browser also highlights `skl_huc_ver02_00_1810.bin` with a size of 218688 bytes. A red arrow points from the terminal window to the browser window.

Note: Take note of the binary file size that was downloaded; Certain methods (e.g., using Wget*) may obtain a smaller file size that may cause an unexpected failure. Intel recommends using the “save file” option by opening the file link and clicking “plain.”

Figure 3. Download the Firmware File by Clicking “plain”

The screenshot shows the kernel/git/firmware/linux repository page. The path is `root/i915/kbl_guc_ver9_14.bin`. The blob is `e5b979e2551c363e4e5c966659d00110b717f5a1 (plain)`. A red arrow points to the "plain" option. Below the blob information, there is a hex dump table. A red arrow points to the "Opening kbl_guc_ver9_14.bin" dialog box, which shows the file name, size (139 KB), and source (https://git.kernel.org). The dialog box has "Cancel" and "Save File" buttons.

ofs	hex dump
0000	06 00 00 00 a1 00 00 00 00 01 00 00 00 00 86 80 00 00 1
0020	40 00 00 00 01 00 00 00 13 56 33 00 61 75 74 6f 62 75 69 6c 5
0040	01 00 03 00 0e 00 09 00 00 00 00 00 00 00 00 00 00 00 00 00
0060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080	bc fc
00a0	c0 b2
00c0	e8 b2
00e0	10 b3
0100	32 b3
0120	4e b3
0140	20 37
0160	20 37
0180	20 37
01a0	20 37



Note: There is a possibility that the current Linux* distribution is packaged with the firmware. However, this depends on whether a user has installed a newer graphics driver or has updated to a new Linux* kernel. Visit www.git.kernel.org to get the complete list.

Table 2. Platforms

Prefix Name	Referred Platforms
BXT	Apollo Lake (a.k.a Broxton - previous code name for Apollo Lake, BXT was canceled in 2016.)
GLK	Gemini Lake
SKL	Sky Lake - 6 th Platform of the Intel® Core™ Processor
KBL	Kaby Lake - 7 th Platform of the Intel® Core™ Processor Coffee Lake - 8 th Platform of the Intel® Core™ Processor

2.3.2 Enable GuC/HuC Firmware Loading

Currently, the GuC/HuC is not enabled by default (as of 4.16). Users should add specific kernel parameters to enable it during the system boot. Note that different Linux* kernels have different parameters. Table 3 shows the related kernel parameters.

Table 3. Kernel Parameters

Kernel Version	Parameter
4.15	i915.enable_guc_loading=1 i915.enable_guc_submission=1
4.18, 4.19	I915.enable_guc=2

- Edit “/etc/default/grub” and extend GRUB_CMDLINE_LINUX_DEFAULT with the corresponding parameter. The result is shown in Figure 4.

Figure 4. Modify the Kernel Parameters

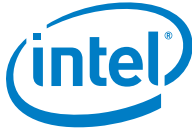
```

root@intel-dbg: /home/intel

root@intel-dbg:/home/intel# cat /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash i915.enable_guc=2"

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"
    
```



- Regenerate the grub configuration file by running “grub-mkconfig -o /boot/grub/grub.cfg.”

Figure 5. Generate a New Grub Configuration File

```
root@intel-dbg: /home/intel
root@intel-dbg:/home/intel# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-4.18.0-041800-generic
Found initrd image: /boot/initrd.img-4.18.0-041800-generic
Found linux image: /boot/vmlinuz-4.15.0-38-generic
Found initrd image: /boot/initrd.img-4.15.0-38-generic
Found linux image: /boot/vmlinuz-4.10.0-28-generic
Found initrd image: /boot/initrd.img-4.10.0-28-generic
Failed to probe /dev/sdb4 for filesystem type
Found Windows Boot Manager on /dev/sda2@/EFI/Microsoft/Boot/bootmgfw.efi
Found CentOS Linux release 7.4.1708 (Core) on /dev/mapper/centos-root
Adding boot menu entry for EFI firmware configuration
done
root@intel-dbg:/home/intel#
```

- Update “initramfs” to ensure that the kernel parameters are fully updated during the boot stage.

```
sudo update-initramfs -u
```

Figure 6. Update “initramfs”

```
root@intel-dbg:/home/intel# update-initramfs -u
update-initramfs: Generating /boot/initrd.img-4.18.0-041800-generic
root@intel-dbg:/home/intel#
```

- After rebooting, the GuC/HuC should be loaded successfully.



3.0 Final Results

As shown in Figure 7, the system status is displayed when the GuC/HuC is correctly loaded. The “found” version and “wanted” version will correspond similarly.

Note: A different Linux* kernel may display a different version number.

Figure 7. GuC/HuC Successful Loading Status

```
Terminal
root@intel-dbg: /home/intel
root@intel-dbg:/home/intel# uname -r
4.18.0-041800-generic
root@intel-dbg:/home/intel# cat /sys/kernel/debug/dri/0/i915_guc_load_status
GuC firmware: i915/skl_guc_ver9_33.bin
status: fetch SUCCESS, load SUCCESS
version: wanted 9.33, found 9.33
header: offset 0, size 128
uCode: offset 128, size 147136
RSA: offset 147264, size 256

GuC status 0x800300ec:
  Bootrom status = 0x76
  uKernel status = 0x0
  MIA Core status = 0x3

Scratch registers:
  0: 0xf0000000
  1: 0x34a5c0
  2: 0x0
  3: 0x5f5e100
  4: 0x0
  5: 0x30afd3
  6: 0x0
  7: 0x8
  8: 0x11
  9: 0x40
 10: 0x0
 11: 0x0
 12: 0x0
 13: 0x0
 14: 0x0
 15: 0x700000

root@intel-dbg:/home/intel# cat /sys/kernel/debug/dri/0/i915_huc_load_status
HuC firmware: i915/skl_huc_ver01_07_1398.bin
status: fetch SUCCESS, load SUCCESS
version: wanted 1.7, found 1.7
header: offset 0, size 128
uCode: offset 128, size 140608
RSA: offset 140736, size 256

HuC status 0x00006080:
root@intel-dbg:/home/intel#
```

Appendix – Enable Low Power (EU-less) Encoding on Gen9+ Intel HD Graphics

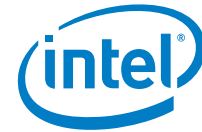
Intel has introduced a brand new fix-function IP (a.k.a VDENC) in Generation 9 GPU architecture. This new IP has the potential to realize low power H.264 video encoding without involving the Execution Unit (EU - the most important GPU engine for computing and rendering).

The Intel® Media SDK contains samples (sample_encode and sample_multi_transcode) to demonstrate how users can enable low power encoding from the application level by using the “-qsv-ff” parameter. However, this function strongly depends on the GuC/HuC enablement. The different possible scenarios for this feature are shown in Figure 8, Figure 9, and Figure 10.

A.1 Low-Power Encoding Will Receive “DEVICE_FAILED” if There is No GuC/HuC Loaded

Figure 8. Failure Caused by Lack of GuC/HuC

```
root@intel-dbg: /home/intel/vpg_driver/MediaSDK/build/_bin/release
root@intel-dbg:/home/intel/vpg_driver/MediaSDK/build/_bin/release# ./sample_encode h264 -i 1080p.yuv -w 1920 -h 1080 -hw -qsv-ff
File output is disabled as -o option isn't specified
libva info: VA-API version 1.3.0
libva info: va_getDriverName() returns 0
libva info: User requested driver 'iHD'
libva info: Trying to open /usr/local/lib/dri/iHD_drv_video.so
libva info: Found init function __vaDriverInit_1_3
libva info: va_openDriver() returns 0
[ERROR], sts=MFX_ERR_DEVICE_FAILED(-17), ResetMFXComponents, n_pnfxENC->Init failed at /home/intel/vpg_driver/MediaSDK/samples/sample_encode/src/pipeline_encode.cpp:1689
[ERROR], sts=MFX_ERR_DEVICE_FAILED(-17), Init, ResetMFXComponents failed at /home/intel/vpg_driver/MediaSDK/samples/sample_encode/src/pipeline_encode.cpp:1419
[ERROR], sts=MFX_ERR_DEVICE_FAILED(-17), main, pPipeline->Init failed at /home/intel/vpg_driver/MediaSDK/samples/sample_encode/src/sample_encode.cpp:1189
root@intel-dbg:/home/intel/vpg_driver/MediaSDK/build/_bin/release#
```



A.2 Normal Encoding vs. Low-Power Encoding (GuC/HuC Enabled) Comparison

The H264 low-power encoding can be implemented after enabling the GuC/HuC. The differences in using this feature are shown in Figure 9 and Figure 10. The EU (shown as “render busy” engine) usage is dropped from 65+% to 0. Thus, the computer program source and silicon power are now saved.

Figure 9. GPU Usage of Normal AVC Encoding

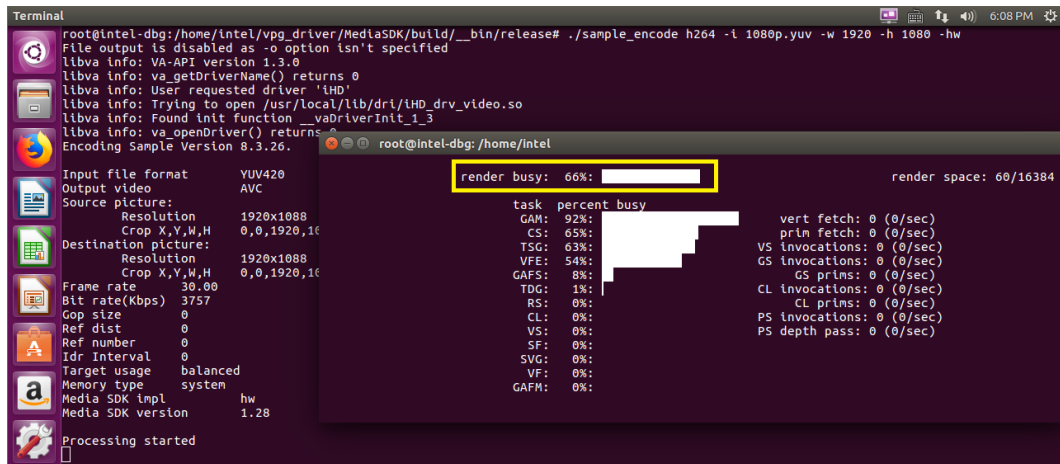


Figure 10. GPU Usage of Low Power AVC Encoding

