

4th improvement plan for resolution and UI size errors

History

- In the SDK released in real mode, interlace resolution cannot be set. Then customer used fixed mode.
 -  [OTT-35909](#) - [SKB] [Smart3] [AMANDROIDS] [AMANDROIDS-13] Resolution Setting CLOSED
- CTS and TVTS fail in fixed mode. So, we provide the way to set interlace resolution in real mode (ACTIVE CONFIG)
 -  [OTT-36791](#) - [SKB] [Smart3S] [AMANDROIDS-21] [Android S] [GTVS] [TVTS] [b/276194241] 32 TvsHdmiHostTestCases fails CLOSED
 -  [OTT-37824](#) - [SKB] [Smart3S] [AMANDROIDS-36] Real mode= FALSE, xTS (CTS-V, TVTS) failed because display class does not distinguish 1080p/1080i CLOSED
 -  [OTT-41524](#) - [SKB] [Smart3S] [AMANDROIDS-21] [Android S] [GTVS] [TVTS] Active config support interlace modes CLOSED
- some problem in real mode, So provide vm override patches.
 -  [OTT-42235](#) - [SKB] [Smart3S] [Realmode] [AMANDROIDS-97] Can't change resolution 720p on SKB's Home-UI CLOSED
 -  [OTT-44026](#) - [SKB] [Smart3S] Add wm reset to fix cold boot UI abnormal issue VERIFIED
 -  [OTT-45136](#) - [SKB] [Smart3S] Web View of ch24 is black when set 720p CLOSED
 -  [OTT-43883](#) - [SKB] [Smart3S] [AMANDROIDS-97] [5.3.7 Full] Boot Animation 1/4 small size issue VERIFIED
 -  [OTT-44273](#) - [SKB] [Smart3S] [AMANDROIDS-112] [5.3.7 Full] Boot Animation 1/4 smaller/black issue CLOSED
- SKB rejected #3. and urged improvement measures without modifying the App
 - There is no solution other than option 3, and it is communicated that the app should follow the Android UI development method.
- Intek removed the patches of #2 and #3 and found an improvement plan for method #1.
 - Rolled back to using Fixed mode, and seems to have found a workaround to pass the CTS verifier in Fixed mode.

Patches

date	jira	patch	path	Intek version reverted
2022/10/28	OTT-35909	franklin: use hwc fixed size mode	device/amlogic/franklin	
2023/04/26	OTT-41524	hwc: support setting interlace mode on AndroidS	hardware/amlogic	
2023/04/26	OTT-41524	systemcontrol: support setting interlace mode on AndroidS	vendor/amlogic/common	
2023/04/26	OTT-41524	TvSettings: support setting interlace mode on AndroidS	vendor/amlogic/common	
2023/05/25	OTT-42235	bootanimation: fix resize window error on android S	frameworks/base	48
2023/05/25	OTT-42235	bootanimation: lower than 1080p, the bootanimation is too large	frameworks/base	48
2023/05/25	OTT-43883	display: sync wm settings when do initial	frameworks/base	50
2023/05/24	OTT-44026	dpi: wm reset in DisplayDensityManager	vendor/amlogic/common	48

Intek's Solution

patch history

common changed from 07b681ffcf152f9b3603aab3e3479af2ee092468 to 0dd4afe09a8c5412cecbe96e3bf17ec63465c942

[+] 0dd4afe09a8c Apply persist. sys. skb. mode

device/amlogic/INTEKS20SI changed from 4ec37f7e0f27a0b9f5828e5ed8b643ce44dc2f8e to d151d0e33d478dcf89b64f762876357d0cd6f85

[+] d151d0e Change HWC to FixedMode

device/amlogic/franklin changed from 1af0b613254f08a0e0295997a2b348316595eb8d to 15e2f324f6699aefdeaec0d72e08db8a825260a

[+] 15e2f32 1080i patch Boardonfig

frameworks/av changed from 3df855f9b5ec47dad1762712b871ccbe1c92c4fc to e95b85395b81b4b34163c78f05d61143375e0dcc

[+] 6a1ca11dda mediacodec: video black, when change hisplayer to liveplayer [1/1]

[+] 2d479be1fb mediacodec: when channel change, block in pushBlankBuffersToNativeWindow [1/1]

frameworks/base changed from 9d941fabfdbf0c629406f9a4d71b65ead3450276 to f5acb2dd6771cdf3156997e9192c5803c6ef74fb

[+] f5acb2dd6771 Revert "bootanimation: lower than 1080p, the bootanimation is too large[1/1]"

[+] 10263284e8ea Revert "bootanimation: fix resize window error on android S"

[+] 7e9268192895 bootanimation: fix resize window error on android S

[+] 6045a773c539 bootanimation: lower than 1080p, the bootanimation is too large[1/1]

hardware/amlogic changed from fef2306ba54615e7a6ff338cdfd38205107290a7 to f282dc065fd783e34cf4b40b8858a8de28cee106

[+] f282dc0 Change HWC to FixedMode

vendor/amlogic/common changed from e3beb80c72e14a054846cfd158efe7dbc6f2f84d to 6187bb4521ebed5c85fd2e5bf48efc14b3acd332
 [+] 6187bb4 Revert "dpi: m reset in DisplayDensityManager [1/1]"
 [+] 53f8d2f Change HWC to FixedMode
 [+] 1ca4150 dpi: wm reset in DisplayDensityManager [1/1]

vendor/intek-service changed from e6250623946dd54b69235bd1f6fec67bb04cb041 to ba6da8194b0e2d15cbf21e9c5754afd34d06daf8
 [+] ba6da81 Change HWC to FixedMode

vendor/intek/device/amlogic changed from 1d8ff14634e371473dc72cc5ba74e87988f8895b to 07900e7448431353debd1b9deda5cc93f2d046ef
 [+] 07900e7 Change HWC to FixedMode

device/amlogic/INTEKS20SI changed from d151d0e33d478dcaf89b64f762876357d0cd6f85 to 0c625d734d25a51d21e2418c25f853522003d0a3
 [+] 0c625d7 [BIDAT20012-85] Set sys.skb.resolution.nonoverride

frameworks/base changed from f5acb2dd6771cdf3156997e9192c5803c6ef74fb to f01dade7809a1234ff34b5973805efa95710e1a9
 [+] f01dade7809a Revert "display: sync wm settings when dc initial [1/1]"
 [+] ffe12968763e display: sync wm settings when dc initial [1/1]

hardware/amlogic changed from f282dc065fd783e34cf4b40b8858a8de28cee106 to 96a0be8164f7fff56e645e51b33940d05bf94099
 [+] 96a0be8 Fixed Mode switching test issue of CTS Verifier

vendor/amlogic/common changed from 6187bb4521ebed5c85fd2e5bf48efc14b3acd332 to b06ee5f471d5017c33e66cf6c2b0a2b43f2b074d
 [+] b06ee5f Fixed Mode switching test issue of CTS Verifier

Fixed Mode switching test issue of CTS Verifier

hardware/amlogic	Author: skkim <skkim@intekdigital.com> Date: Wed May 31 19:26:45 2023 +0900 Fixed Mode switching test issue of CTS Verifier Change-Id: Ifdd2cdd2e36c8a2bf4a9b4291eed8b6b98184f2e <pre>diff --git a/hwcomposer/common/hwc/FixedSizeModeMgr.cpp b/hwcomposer/common/hwc/FixedSizeModeMgr.cpp index 058ccc2..659a60c 100755 --- a/hwcomposer/common/hwc/FixedSizeModeMgr.cpp +++ b/hwcomposer/common/hwc/FixedSizeModeMgr.cpp @@ -199,10 +199,11 @@ int32_t FixedSizeModeMgr::setActiveConfig(uint32_t config) { } drm_mode_info_t cfg = it->second; + mCurConfig = it->first; mConnector->setMode(cfg); + MESON_LOGD("setActiveConfig: %s", cfg.name); std::string dispmode(cfg.name); sc_set_display_mode(dispmode); - mCurConfig = it->first; + } else { + if (config > 0) { + MESON_LOGE("FixedSizeModeMgr dont support config (%d)", config);</pre>
vendor/amlogic /common	Author: skkim <skkim@intekdigital.com> Date: Wed May 31 19:27:58 2023 +0900 Fixed Mode switching test issue of CTS Verifier Change-Id: I2140de0953a9f9eaadb86388180709ffcfe1e975 <pre>diff --git a/frameworks/services/systemcontrol/DisplayMode.cpp b/frameworks/services/systemcontrol/DisplayMode.cpp index d2d85c2..e7081a9 100755 --- a/frameworks/services/systemcontrol/DisplayMode.cpp +++ b/frameworks/services/systemcontrol/DisplayMode.cpp @@ -3702,7 +3702,13 @@ void DisplayMode::setSklDeepColor(char *attr) if (isColorAuto) { if (!strcmp(mHdmdata.final_displaymode, "2160p", 5)) { - for (i=0; i<ARRAY_SIZE(ITK_COLOR_ATTR_4K); i++) { + if (mHdmdata.final_displaymode[5] == '2' mHdmdata.final_displaymode[5] == '3') { + i = 3; + } else { + i = 0; + } + } + for (; i<ARRAY_SIZE(ITK_COLOR_ATTR_4K); i++) { if (strstr(mHdmdata.dc_cap, ITK_COLOR_ATTR_4K[i]) != NULL) { strcpy(attr, ITK_COLOR_ATTR_4K[i]); break;</pre>

Change HWC to FixedMode

device/amlogic /INTEKS20SI	Author: skkim <skkim@intekdigital.com> Date: Thu May 25 16:22:41 2023 +0900 Change HWC to FixedMode Change-Id: I1c4dc09638897f29301fa07960e1ae277f15c17d <pre>diff --git a/BoardConfig.mk b/BoardConfig.mk index 1103352..a3dcce6 100644 --- a/BoardConfig.mk +++ b/BoardConfig.mk @@ -70,7 +70,7 @@ HWC_PRIMARY_CONNECTOR_TYPE := hdmi #HWC_ENABLE_HEADLESS_MODE := true #HWC_ENABLE_SOFTWARE_VSYNC := true HWC_ENABLE_PRIMARY_HOTPLUG := true -HWC_ENABLE_REAL_MODE := true +HWC_ENABLE_REAL_MODE := false HWC_HDMI_FRAC_MODE := 2 #HWC_ENABLE_SECURE_LAYER_PROCESS := true #HWC_DISABLE_CURSOR_PLANE := true</pre>
hardware/amlogic	Author: skkim <skkim@intekdigital.com> Date: Fri May 26 18:17:55 2023 +0900 Change HWC to FixedMode Change-Id: Idee812f0986cc02fd83429ef95f368816fb6f0cb <pre>diff --git a/hwcomposer/common/hwc/FixedSizeModeMgr.cpp b/hwcomposer/common/hwc/FixedSizeModeMgr.cpp index 7b3aaf0..058ccc2 100755 --- a/hwcomposer/common/hwc/FixedSizeModeMgr.cpp +++ b/hwcomposer/common/hwc/FixedSizeModeMgr.cpp @@ -16,6 +16,25 @@ #define DEFAULT_DPI (159) #define DEFAULT_REFRESH_RATE (60.0f) +// __ITK_CODED__ +#include <systemcontrol.h> +#include <math.h> + +static const char *FixedDispModes[] = { + "1080p24hz", + "1080p25hz", + "1080p30hz", + "1080p50hz", + "1080p60hz", + "2160p24hz", + "2160p25hz", + "2160p30hz", + "2160p50hz", + "2160p60hz", + "720p50hz", + "720p60hz" +}; + static drm_mode_info_t fakeInitialMode = { .name = "FAKE_INITIAL_MODE", .dpiX = DEFAULT_DPI, @@ -78,6 +97,17 @@ int32_t FixedSizeModeMgr::update() { strncpy(mCurMode.name, "FAKE_PREVIOUS_MODE", DRM_DISPLAY_MODE_LEN); } +// __ITK_CODED__ +if (mSkbMode) { + if (useFakeMode) { + mModes.clear(); + mModes.emplace(mModes.size(), fakeInitialMode); + mCurConfig = 0; + } else { + updateReal(realMode); + } +} + return 0; } @@ -85,6 +115,99 @@ int32_t FixedSizeModeMgr::getDisplayMode(drm_mode_info_t & mode) { return mCrtc->getMode(mode); } -int32_t FixedSizeModeMgr::getDisplayConfigs(</pre>

```

- uint32_t * outNumConfigs, uint32_t * outConfigs) {
- *outNumConfigs = 1;
- if (outConfigs) {
- *outConfigs = 0;
- }
+int32_t FixedSizeModeMgr::getDisplayConfigs(uint32_t * outNumConfigs, uint32_t * outConfigs) {
+ // __ITK_CODED__
+ if (mSkbMode) {
+ int i = 0;
+
+ *outNumConfigs = mModes.size();
+ if (outConfigs) {
+ for (auto it = mModes.begin(); it != mModes.end(); ++it) {
+ outConfigs[i] = it->first;
+ i++;
+ }
+ } else {
+ *outNumConfigs = 1;
+ if (outConfigs) {
+ *outConfigs = 0;
+ }
+ }
+ return HWC2_ERROR_NONE;
+ }

-int32_t FixedSizeModeMgr::getDisplayAttribute(
- uint32_t config __unused, int32_t attribute, int32_t * outValue,
- int32_t caller __unused) {
- switch (attribute) {
- case HWC2_ATTRIBUTE_WIDTH:
- *outValue = mCurMode.pixelW;
- break;
- case HWC2_ATTRIBUTE_HEIGHT:
- *outValue = mCurMode.pixelH;
- break;
- case HWC2_ATTRIBUTE_VSYNC_PERIOD:
- float refresh_rate;
- if (HwcConfig::isHeadlessMode()) {
- refresh_rate = (float)HwcConfig::headlessRefreshRate();
- } else {
- refresh_rate = mCurMode.refreshRate;
- }
- if (HwcConfig::getMaxRefreshRate() > 0.0f &&
- refresh_rate > HwcConfig::getMaxRefreshRate()) {
- refresh_rate = HwcConfig::getMaxRefreshRate();
- }
- *outValue = 1e9 / refresh_rate;
- break;
- case HWC2_ATTRIBUTE_DPI_X:
- *outValue = mCurMode.dpiX;
- break;
- case HWC2_ATTRIBUTE_DPI_Y:
- *outValue = mCurMode.dpiY;
- break;
- case HWC2_ATTRIBUTE_CONFIG_GROUP:
- *outValue = mCurMode.groupId;
- break;
- default:
- MESON_LOGE("Unkown display attribute(%d)", attribute);
- break;
- }
+int32_t FixedSizeModeMgr::getDisplayAttribute(uint32_t config, int32_t attribute, int32_t * outValue, int32_t
+ caller __unused) {
+ // __ITK_CODED__
+ if (mSkbMode) {
+ return getDisplayAttributeReal(config, attribute, outValue);
+ } else {
+ switch (attribute) {
+ case HWC2_ATTRIBUTE_WIDTH:
+ *outValue = mCurMode.pixelW;
+ break;
+ case HWC2_ATTRIBUTE_HEIGHT:
+ *outValue = mCurMode.pixelH;
+ break;
+ case HWC2_ATTRIBUTE_VSYNC_PERIOD:
+ float refresh_rate;
+ if (HwcConfig::isHeadlessMode()) {
+ refresh_rate = (float)HwcConfig::headlessRefreshRate();
+ } else {
+ refresh_rate = mCurMode.refreshRate;
+ }
+ if (HwcConfig::getMaxRefreshRate() > 0.0f &&
+ refresh_rate > HwcConfig::getMaxRefreshRate()) {

```

```

+         refresh_rate = HwcConfig::getMaxRefreshRate();
+     }
+     *outValue = 1e9 / refresh_rate;
+     break;
+ case HWC2_ATTRIBUTE_DPI_X:
+     *outValue = mCurMode.dpiX;
+     break;
+ case HWC2_ATTRIBUTE_DPI_Y:
+     *outValue = mCurMode.dpiY;
+     break;
+ case HWC2_ATTRIBUTE_CONFIG_GROUP:
+     *outValue = mCurMode.groupId;
+     break;
+ default:
+     MESON_LOGE("Unkown display attribute(%d)", attribute);
+     break;
+ }
+ }
+
+ return HWC2_ERROR_NONE;
+ }

-int32_t FixedSizeModeMgr::getActiveConfig(
- uint32_t * outConfig, int32_t caller __unused) {
- *outConfig = 0;
+int32_t FixedSizeModeMgr::getActiveConfig(uint32_t * outConfig, int32_t caller __unused) {
+ // __ITK_CODED__
+ if (mSkbMode) {
+     *outConfig = mCurConfig;
+ } else {
+     *outConfig = 0;
+ }
+ return HWC2_ERROR_NONE;
+ }

-int32_t FixedSizeModeMgr::setActiveConfig(
- uint32_t config) {
- if (config > 0) {
-     MESON_LOGE("FixedSizeModeMgr dont support config (%d)", config);
- }
+int32_t FixedSizeModeMgr::setActiveConfig(uint32_t config) {
+ // __ITK_CODED__
+ if (mSkbMode) {
+     std::map<uint32_t, drm_mode_info_t>::iterator it = mModes.find(config);
+     if (it == mModes.end()) {
+         MESON_LOGE("setActiveConfig: invalid config=%d", config);
+         return HWC2_ERROR_NOT_VALIDATED;
+     }
+
+     drm_mode_info_t cfg = it->second;
+     mConnector->setMode(cfg);
+     std::string dispmode(cfg.name);
+     sc_set_display_mode(dispmode);
+     mCurConfig = it->first;
+ } else {
+     if (config > 0) {
+         MESON_LOGE("FixedSizeModeMgr dont support config (%d)", config);
+     }
+ }
+ return HWC2_ERROR_NONE;
+ }

@@ -169,3 +232,105 @@ void FixedSizeModeMgr::dump(String8 & dumpstr) {
+ "-----\n");
+ }

+// __ITK_CODED__
+void FixedSizeModeMgr::getSkbMode()
+{
+    int fd;
+    char buf[8] = {0};
+
+    fd = open("/sys/class/itk_debug/skbmode", O_RDONLY);
+    if (fd < 0) {
+        mSkbMode = false;
+        return;
+    }
+
+    read(fd, buf, 8);
+    close(fd);
+
+    if (buf[0] == '1') {
+        mSkbMode = true;
+    } else {

```

```

+         mSkbMode = false;
+     }
+ }
+
+ bool FixedSizeModeMgr::isMatch(const char *mode)
+ {
+     int i;
+
+     for (i=0; i < 12; i++) {
+         if (strcmp(FixedDispModes[i], mode) == 0) {
+             return true;
+         }
+     }
+
+     return false;
+ }
+
+ void FixedSizeModeMgr::updateReal(drm_mode_info_t real)
+ {
+     drm_mode_info_t mode;
+     std::map<uint32_t, drm_mode_info_t> connectorModeList;
+
+     mModes.clear();
+     mConnector->getModes(connectorModeList);
+
+     for (auto it = connectorModeList.begin(); it != connectorModeList.end(); it++) {
+         mode = it->second;
+         if (isMatch(mode.name)) {
+             mModes.emplace(mModes.size(), mode);
+         }
+     }
+
+     for (auto it = mModes.begin(); it != mModes.end(); ++it) {
+         if (strcmp(it->second.name, real.name) == 0) {
+             if (fabs(real.refreshRate - it->second.refreshRate) < 1e-2) {
+                 mCurConfig = it->first;
+             }
+         }
+     }
+ }
+
+ int32_t FixedSizeModeMgr::getDisplayAttributeReal(uint32_t config, int32_t attribute, int32_t* outValue)
+ {
+     drm_mode_info_t curMode;
+     std::map<uint32_t, drm_mode_info_t>::iterator it;
+     it = mModes.find(config);
+
+     if (it == mModes.end()) {
+         return HWC2_ERROR_UNSUPPORTED;
+     }
+
+     curMode = it->second;
+
+     switch (attribute) {
+     case HWC2_ATTRIBUTE_WIDTH:
+         *outValue = curMode.pixelW;
+         break;
+     case HWC2_ATTRIBUTE_HEIGHT:
+         *outValue = curMode.pixelH;
+         break;
+     case HWC2_ATTRIBUTE_VSYNC_PERIOD:
+         if (HwcConfig::isHeadlessMode()) {
+             *outValue = 1e9 / HwcConfig::headlessRefreshRate();
+         } else {
+             *outValue = 1e9 / curMode.refreshRate;
+         }
+         break;
+     case HWC2_ATTRIBUTE_DPI_X:
+         *outValue = curMode.dpiX;
+         break;
+     case HWC2_ATTRIBUTE_DPI_Y:
+         *outValue = curMode.dpiY;
+         break;
+     case HWC2_ATTRIBUTE_CONFIG_GROUP:
+         *outValue = curMode.groupId;
+         break;
+     default:
+         MESON_LOGE("Unkown display attribute(%d)", attribute);
+         break;
+     }
+
+     return HWC2_ERROR_NONE;
+ }

```

```

diff --git a/hwcomposer/common/hwc/HwcModeMgr.cpp b/hwcomposer/common/hwc/HwcModeMgr.cpp
index c6d4c62..adb79e9 100755
--- a/hwcomposer/common/hwc/HwcModeMgr.cpp
+++ b/hwcomposer/common/hwc/HwcModeMgr.cpp
@@ -16,7 +16,9 @@
std::shared_ptr<HwcModeMgr> createModeMgr(
    hwc_modes_policy_t policy) {
    if (policy == FIXED_SIZE_POLICY) {
        HwcModeMgr * modeMgr = (HwcModeMgr *) new FixedSizeModeMgr();
+        FixedSizeModeMgr * fsm = new FixedSizeModeMgr();
+        fsm->getSkbMode();
+        HwcModeMgr * modeMgr = (HwcModeMgr *) fsm;
        return std::shared_ptr<HwcModeMgr>(std::move(modeMgr));
    } else if (policy == FULL_ACTIVE_POLICY) {
        HwcModeMgr * modeMgr = (HwcModeMgr *) new VariableModeMgr();
diff --git a/hwcomposer/common/hwc/include/FixedSizeModeMgr.h b/hwcomposer/common/hwc/include/FixedSizeModeMgr.h
index 375fb27..2479488 100755
--- a/hwcomposer/common/hwc/include/FixedSizeModeMgr.h
+++ b/hwcomposer/common/hwc/include/FixedSizeModeMgr.h
@@ -49,6 +49,9 @@ public:
    return HWC2_ERROR_NONE;
};

+ // __ITK_CODED__
+ void getSkbMode();
+
protected:
    std::shared_ptr<HwDisplayConnector> mConnector;
    std::shared_ptr<HwDisplayCrtc> mCrtc;
@@ -61,6 +64,13 @@ protected:
    uint32_t mFbWidth;
    uint32_t mFbHeight;

+ // __ITK_CODED__
+ bool isMatch(const char *mode);
+ void updateReal(drm_mode_info_t real);
+ int32_t getDisplayAttributeReal(uint32_t config, int32_t attribute, int32_t* outValue);
+ bool mSkbMode;
+ uint32_t mCurConfig;
+
};

#endif /* FIXED_SIZE_MODE_MGR_H */

```

vendor/amlogic/common

```

Author: skkim <skkim@intekdigital.com>
Date: Thu May 25 17:27:29 2023 +0900

Change HWC to FixedMode

Change-Id: I20cd109b14df81ab6c3c44a4e7d25e7fdc59c77b

diff --git a/frameworks/services/systemcontrol/DisplayMode.cpp b/frameworks/services/systemcontrol/DisplayMode.cpp
index 8477033..d2d85c2 100755
--- a/frameworks/services/systemcontrol/DisplayMode.cpp
+++ b/frameworks/services/systemcontrol/DisplayMode.cpp
@@ -702,6 +702,12 @@ void DisplayMode::setSourceOutputMode(const char* outputmode) {
    AutoMutex _l(mLock);
#endif

+ // __ITK_CODED__
+ if (mHdmidata.reason == OUPUT_CHANGE_BY_HWC && strcmp(mHdmidata.final_displaymode, outputmode) == 0) {
+     SYS_LOGI("skip to setSourceOutputMode because of same resolution");
+     return;
+ }
+
    //1. get hdmi data
    mHdmidata.state = OUPUT_MODE_STATE_SWITCH;
    strcpy(mHdmidata.ui_hdmi_mode, outputmode);
@@ -783,7 +789,9 @@ void DisplayMode::applyDisplaySetting(output_mode_state state) {
    std::string cur_ColorAttribute;
    DisplayModeMgr::getInstance().getDisplayAttribute(DISPLAY_HDMI_COLOR_ATTR, cur_ColorAttribute);
    strcpy(cur_ColorAttribute, cur_ColorAttribute.c_str());
-    strcpy(final_deepcolor, mHdmidata.final_deepcolor);
+ // __ITK_CODED__
+ //strcpy(final_deepcolor, mHdmidata.final_deepcolor);
+ setSkbDeepColor(final_deepcolor);
    SYS_LOGI("curDeepcolor[%s] final_deepcolor[%s]\n", cur_ColorAttribute, final_deepcolor);

    if (strstr(cur_ColorAttribute, final_deepcolor) == NULL) {
@@ -3655,7 +3663,62 @@ bool DisplayMode::getPrefHdmiDispMode(char* mode) {
}

```

```

// __ITK_CODED__ : INTEL
-int DisplayMode::setHdcpMode(int mode) {
+int DisplayMode::setHdcpMode(int mode)
+{
+    return pTxAuth->setHdcpMode(mode);
+}

+#define COLOR_AUTO "/btv_home/config/itk_attr.conf"
+
+static const char *ITK_COLOR_ATTR_4K[] = {
+    "420,8bit",
+    "420,10bit",
+    "420,12bit",
+    "422,8bit",
+    "422,10bit",
+    "422,12bit",
+    "444,8bit",
+    "444,10bit",
+    "rgb,8bit"
+};
+
+static const char *ITK_COLOR_ATTR[] = {
+    "422,8bit",
+    "422,10bit",
+    "422,12bit",
+    "444,8bit",
+    "444,10bit",
+    "rgb,8bit"
+};
+
+void DisplayMode::setSkbDeepColor(char *attr)
+{
+    int i;
+    bool isColorAuto = true;
+
+    if (access(COLOR_AUTO, F_OK) == 0) {
+        isColorAuto = false;
+    }
+
+    if (isColorAuto) {
+        if (!strcmp(mHdmdata.final_displaymode, "2160p", 5)) {
+            for (i=0; i<ARRAY_SIZE(ITK_COLOR_ATTR_4K); i++) {
+                if (strstr(mHdmdata.dc_cap, ITK_COLOR_ATTR_4K[i]) != NULL) {
+                    strcpy(attr, ITK_COLOR_ATTR_4K[i]);
+                    break;
+                }
+            }
+        } else {
+            for (i=0; i<ARRAY_SIZE(ITK_COLOR_ATTR); i++) {
+                if (strstr(mHdmdata.dc_cap, ITK_COLOR_ATTR[i]) != NULL) {
+                    strcpy(attr, ITK_COLOR_ATTR[i]);
+                    break;
+                }
+            }
+        }
+    } else {
+        strcpy(attr, mHdmdata.final_deepcolor);
+    }
+}
+
+////////////////////////////////////
+
diff --git a/frameworks/services/systemcontrol/DisplayMode.h b/frameworks/services/systemcontrol/DisplayMode.h
index 2ff2ad4..83646b4 100755
--- a/frameworks/services/systemcontrol/DisplayMode.h
+++ b/frameworks/services/systemcontrol/DisplayMode.h
@@ -709,6 +709,9 @@ private:
     mutable Mutex mLock;
     sp<SystemControlNotify> mNotifyListener;
 #endif
+
+    // __ITK_CODED__
+    void setSkbDeepColor(char *attr);
+};

#endif // ANDROID_DISPLAY_MODE_H

```