Realtek MI Player API
user guide

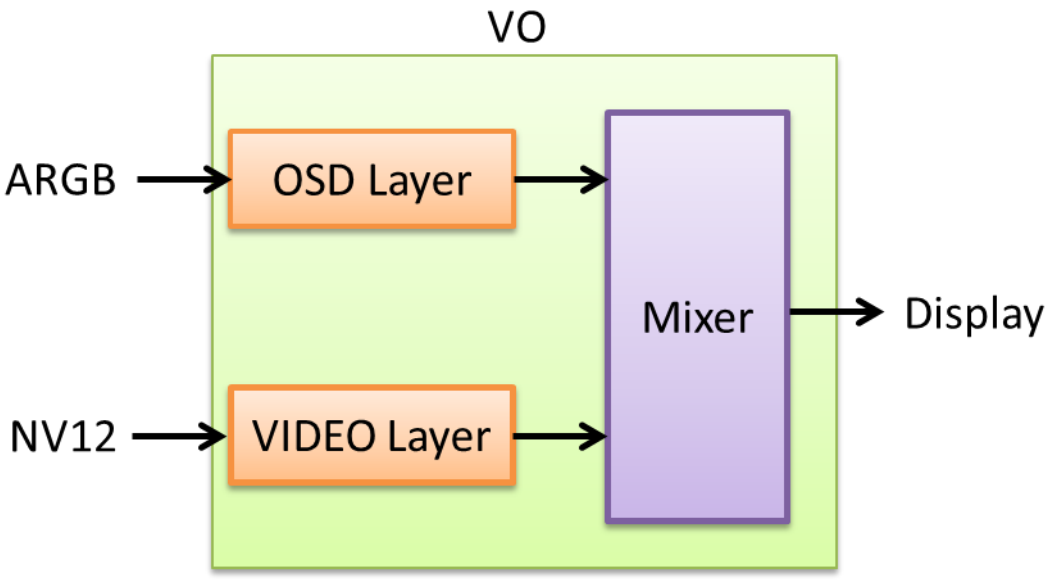
Version Information			
Version	Writer	Description	Date
0	Joshua Yang	First draft	2019/04/08

1. Introduction

This document is the tutorial of how to apply RTK player media interface (MI) into a normal player to display.

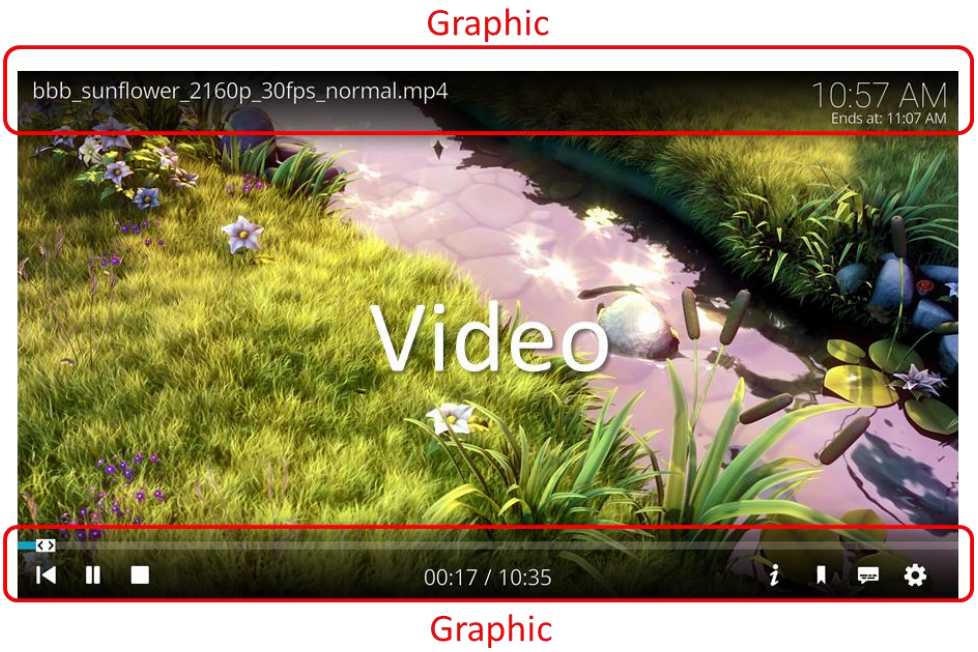
1.1 System structure

Video Output (VO) is the hardware module to mixes input source (video/graphic) into a frame and sends it HDMI or Panel Interface to display. There are two input sources of VO. One is “OSD Layer”, and the other is “VIDEO Layer” (Figure 1.1). The format of video is NV12 , OSD is ARGB . Finally, the “Mixer” processes these different sources and blends them into a picture as output. The “OSD Layer” is on the “VIDEO Layer”. It means that “VIDEO Layer” will not appear if “OSD Layer” has no transparency.



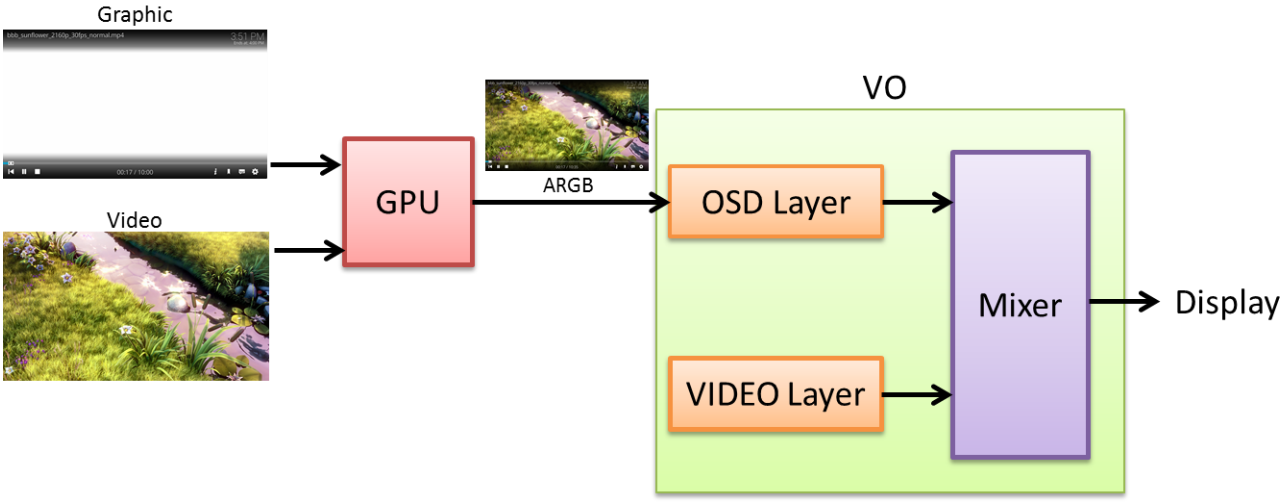
(Figure 1.1)

The screen of a player can be divided into “video” and “graphic”. The “video” means the picture that produced by decoder. The “graphic” can be the control button, time bar, file information or subtitle. (Figure 1.2)



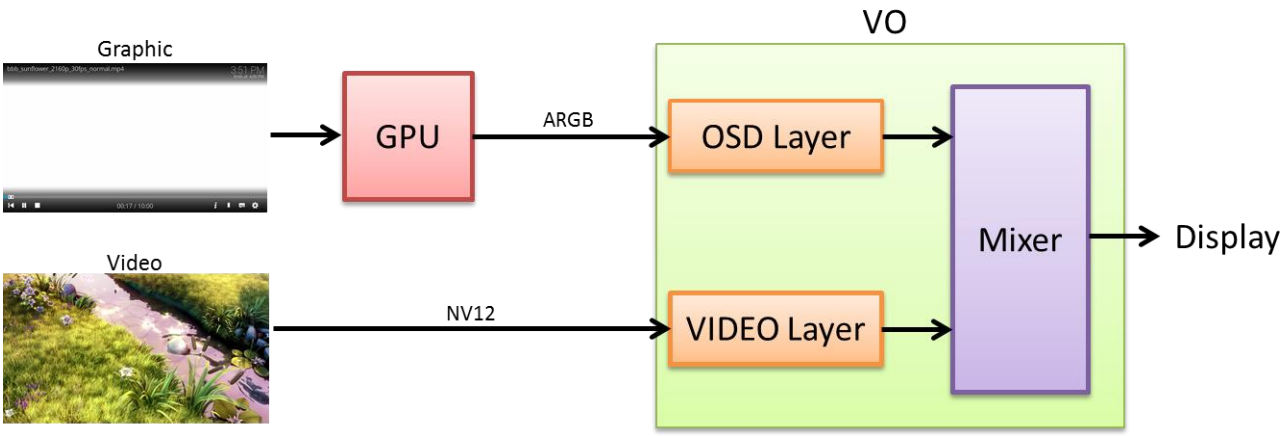
(Figure 1.2)

Player uses GPU to render both video and graphic usually (Figure 1.3). However it takes time for GPU when the resolution of video is high. This phenomenon is more obvious in embedded systems. The render time is bigger than display time of a frame. Player will drop more frames to achieve Audio Video Synchronize (AV sync).



(Figure 1.3)

A RTK Player MI provides a method to flip video directly to “VIDEO layer” of VO. GPU only needs to process the graphic part and can reduce the render time (Figure 1.4). It's worth noting that the player needs to make graphic transparent to make video be displayed.



(Figure 1.4)

2. Basic Data Structures

2.1 RtkMiReturn_e

[Description]

Return value of RTK MI APIs.

[Syntax]

```
typedef enum {
    RTK_MI_SUCCESS = 0,
    RTK_MI_ERR_FAILURE = -1,
    RTK_MI_ERR_INVALID_OPERATION = -2,
    RTK_MI_ERR_INVALID_PARAM = -3,
    RTK_MI_ERR_INSUFFICIENT_RESOURCE = -4,
    RTK_MI_ERR_INVALID_STREAM = -5,
} RtkMiReturn_e;
```

2.2 MediaServiceSel_e

[Description]

There are two types of RTK MI services. One is “PLAYER” , and the other is “NVR”. This document focus on MEDIA_SERVICE_PLAYER .

[Syntax]

```
typedef enum {
    MEDIA_SERVICE_PLAYER,
    MEDIA_SERVICE_NVR,
}
```

[Note]

These two services can’t exist at the same time.

2.3 HDMIOutput_e

[Description]

HDMI Tx output resolution. HDMI_OUTPUT_AUTO means that HDMI Tx will shows the best resolution of monitor automatically.

HDMI_OUTPUT_NOT_SUPPORT means that HDMI Tx can't identify the resolution. HDMI_OUTPUT_NONE means that HDMI Tx is disabled.

[Syntax]

```
typedef enum {
    HDMI_OUTPUT_AUTO,
    HDMI_OUTPUT_720P_50,
    HDMI_OUTPUT_720P_60,
    HDMI_OUTPUT_1080P_50,
    HDMI_OUTPUT_1080P_60,
    HDMI_OUTPUT_4K_25,
    HDMI_OUTPUT_4K_30,
    HDMI_OUTPUT_4K_50,
    HDMI_OUTPUT_4K_60,
    HDMI_OUTPUT_NOT_SUPPORT,
    HDMI_OUTPUT_NONE,
} HDMIOutput_e;
```

2.4 DPOutput_e

[Description]

DP Tx output resolution. DP_OUTPUT_AUTO means that DP Tx will shows the best resolution of monitor. DP_OUTPUT_NOT_SUPPORT means that DP Tx can't identify the resolution. DP_OUTPUT_NONE means that DP Tx is disabled.

[Syntax]

```
typedef enum {
    DP_OUTPUT_AUTO,
    DP_OUTPUT_720P_60,
    DP_OUTPUT_1080P_60,
    DP_OUTPUT_4K_30,
    DP_OUTPUT_NOT_SUPPORT,
    DP_OUTPUT_NONE,
} DPOutput_e;
```

2.5 ScanningType_e

[Description]

The scanning type of video.

[Syntax]

```
typedef enum {
    TYPE_PROGRESSIVE,
    TYPE_INTERLACE,
} ScanningType_e;
```

2.6 HDMITxSt_t

[Description]

The structure describes the output information of HDMI Tx.

[Syntax]

```
typedef struct {
    HDMIOutput_e mode;
    unsigned int width;
    unsigned int height;
    unsigned int fps;
    ScanningType_e type;
}HDMITxSt_t;
```

2.7 DPTxSt_t

[Description]

The structure describes the output information of DP Tx.

[Syntax]

```
typedef struct {
    DpOutput_e mode;
    unsigned int width;
    unsigned int height;
    unsigned int fps;
    ScanningType_e type;
}DpTxSt_t;
```

3. Media Interfaces

3.1 rtk_MI_Init

RtkMiReturn_e rtk_MI_Init(
MediaServiceSel_e sel)

[Description]

Initialize and select the RTK media service. There can only be one media service at the same time.

[Parameters]

sel	RTK media service selection

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: This service doesn't support.
RTK_MI_ERR_INSUFFICIENT_RESOURCE: The service has initialized before or the other service has not destroyed yet.

3.2 rtk_MI_Destroy

RtkMiReturn_e rtk_MI_Destroy(void)

[Description]

Destroy the selected RTK media service.

[Parameters]

NONE	

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: This service doesn't support.
RTK_MI_ERR_INVALID_OPERATION: This service has not initialized before.

3.3 rtk_MI_Player_SetDispWindow

RtkMiReturn_e rtk_MI_Player_SetWindow(
unsigned int offset_x,
unsigned int offset_y,
unsigned int width,
unsigned int height)

[Description]

Specify the position/size of display window. Display window means the display area of video (Figure 3.1). The size of the display window should not exceed the boundary of the video output device, which is determined by rtk_MI_VO_SetOutput().

(Figure 3.1)

[Parameters]

offset_x	X axis of the display window position (the upper left corner indicates 0)
offset_y	Y axis of the display window position (the upper left corner indicates 0)
width	width of the display window
height	height of the display window

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: Setting fail.
RTK_MI_ERR_INVALID_OPERATION: Haven't initialized.
RTK_MI_ERR_INVALID_PARAM: Input parameters exceed the boundary of the video output.

3.4 rtk_MI_Player_GetDispWindow

RtkMiReturn_e rtk_MI_Player_GetWindow(
unsigned int *offset_x,
unsigned int *offset_y,
unsigned int *width,
unsigned int *height)

[Description]

Get the position/size of display window.

[Parameters]

offset_x	X axis of the display window position (the upper left corner indicates 0)
offset_y	Y axis of the display window position (the upper left corner indicates 0)
width	width of the display window
height	height of the display window

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: Setting fail.
RTK_MI_ERR_INVALID_OPERATION: Haven't initialized.
RTK_MI_ERR_INVALID_PARAM: Input parameters exceed the boundary of the video output.

3.5 rtk_MI_Player_DirectFlip

RtkMiReturn_e rtk_MI_Player_DirectFlip(
unsigned char * virtBufAddr,
unsigned int width,
unsigned int height,
long long ptsMs,
void * privData)

[Description]

Inform media service to shows specified frame. This API doesn't control AVSync, the AVSync must be controlled by player itself. It is meant that the time interval between each rtk_MI_Player_DirectFlip() must be a reasonable value.

[Parameters]

virtBufAddr	virtual frame buffer address
width	frame width
height	frame height
ptsMs	frame PTS(Presentation Time Stamp) in millisecond
privData	private data of decoded frame

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: Setting fail.
RTK_MI_ERR_INVALID_OPERATION: Haven't initialized.
RTK_MI_ERR_INVALID_PARAM: Input parameters abnormal.
RTK_MI_ERR_INVALID_STREAM: Frame is not a RTK hardware decoded one.

3.6 rtk_MI_Player_SetBlank

RtkMiReturn_e rtk_MI_Player_SetBlank(
unsigned char enable)

[Description]

Set blank status of display window.

[Parameters]

enable	Display window mute enable

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: Setting fail.
RTK_MI_ERR_INVALID_OPERATION: Haven't initialized.

3.7 rtk_MI_Player_GetBlank

RtkMiReturn_e rtk_MI_Player_GetBlank(
unsigned char *enabled)

[Description]

Get blank status of display window.

[Parameters]

enable	Display window mute enable

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: Setting fail.
RTK_MI_ERR_INVALID_OPERATION: Haven't initialized.
RTK_MI_ERR_INVALID_PARAM: Input parameter is NULL pointer.

3.8 rtk_MI_VO_SetOutput

RtkMiReturn_e rtk_MI_VO_SetOutput(
HDMIOutput_e hdmiOutput,
DPOutput_e dpOutput)

[Description]

Set output resolution of video device.

[Parameters]

hdmiOutput	HDMI Tx resolution
dpOutput	DP Tx resolution

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: Setting fail.
RTK_MI_ERR_INVALID_OPERATION: Haven't initialized.
RTK_MI_ERR_INVALID_PARAM: Input parameter is abnormal.

3.9 rtk_MI_VO_GetOutput

RtkMiReturn_e rtk_MI_VO_GetOutput(
HDMITxSt_t * hdmiSt,
DPTxSt_t * dpSt)

[Description]

Get output information of video device.

[Parameters]

hdmiSt	HDMI Tx information
dpSt	DP Tx information

[Returns]

On success, returns RTK_MI_SUCCESS ; on error it returns an error number:
RTK_MI_ERR_FAILURE: Setting fail.
RTK_MI_ERR_INVALID_OPERATION: Haven't initialized.
RTK_MI_ERR_INVALID_PARAM: Input parameter is NULL pointer.

4. System Control

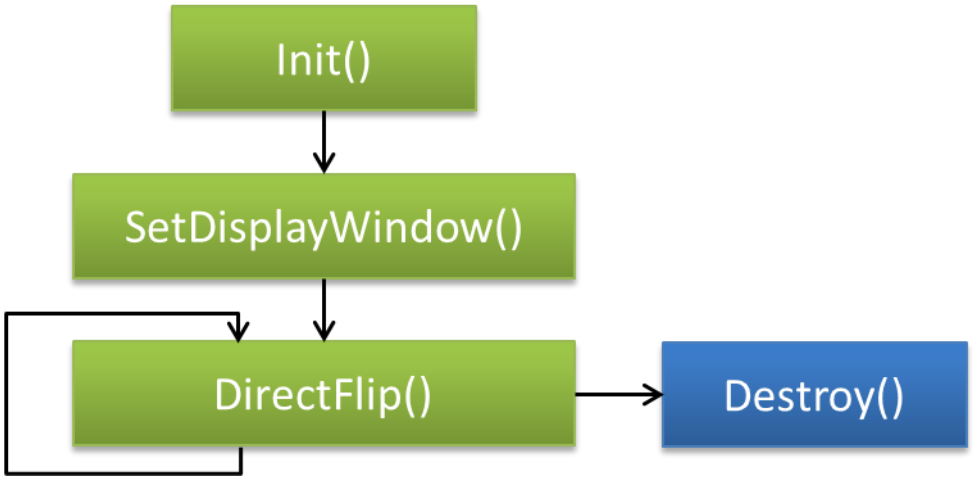
In this chapter, we will introduce the main API flow of RTK Player MI.

4.1 Prepare

You need to make transparent of the graphic part and disable the video render of the GPU to present the video on VIDEO layer behind OSD layer.

4.2 Main Flow

The main flow shows in (Figure 4.1) :



(Figure 4.1)

- The RTK MI must be initialized before the RTK MI services are enabled by the player.
- Player also needs to specify the display window of video to inform VO the display area.
- Collection the frame information (physical buffer address/ width/ height/pts) of each frame decoded from decoder and send these parameters into VO by rtk_MI_Player_DirectFlip() continuously.
- Similarly, the RTK MI must be destroyed (deinitialized) when closing the player to release resources after the RTK MI services are disabled by applications.