

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields		

Data Structures

Here are the data structures with brief descriptions:

DecParam_tag	
DIM_tag	Set width and height
DispParam_Ext1_tag	
dvr_bs_data_tag	Dvr bit-stream data
dvr_dec_channel_param_tag	
dvr_dec_clear_param_tag	
dvr_dec_control_tag	
dvr_dec_queue_get_tag	
dvr_disp_clear_param_tag	
dvr_disp_color_attribute_tag	
dvr_disp_color_key_tag	
dvr_disp_control_tag	
dvr_disp_disp_param_tag	
dvr_disp_plane_param_st_tag	
dvr_disp_plane_param_tag	
dvr_disp_resolution_tag	
dvr_disp_scaler_info_tag	
dvr_disp_update_disp_param_tag	
dvr_disp_update_plane_param_tag	
dvr_disp_vbi_info_tag	
dvr_enc_channel_param_tag	Dvr encode channel parameter
dvr_enc_control_tag	Dvr encode control parameter
dvr_enc_copy_buf_tag	

dvr_enc_queue_get_tag	Get dvr encode buffer
dvr_enc_src_tag	Dvr encode source parameter
dvr_enc_update_channel_param_tag	Update dvr encode channel parameter
dvr_graph_vqueuet_tag	Set dvr graph queue configuration
dvr_rate_tag	
EncParam_Ext1_tag	No longer used, replace by EncParam_Ext4
EncParam_Ext2_tag	No longer used, replace by EncParam_Ext4
EncParam_Ext3_tag	No longer used, replace by EncParam_Ext4
EncParam_Ext4_tag	Encode parameter extension
EncParam_Ext5_tag	
EncParam_tag	Encode parameter
FuncTag_tag	Function tag for videograph level
POS_tag	Set position x and y
QueueMemConfig_tag	Set queue memory configuration
RECT_tag	Set size and position
ReproduceBitStream_tag	Dvr bit stream parameter, include main, sub1, sub2 bit-stream
ROI_ALL_tag	Set ROI position x, y, width, height, and

	enable/disable
ScalerParamtag	Scalar parameter
snapshot_param_tag	Snapshot parameter
dvr_disp_update_disp_param_tag::val_t	
video_process_tag	Dvr video parameter

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

DecParam_tag Struct Reference

Data Fields

```
int output_type
int reserved [2]
```

Detailed Description

Definition at line **14** of file [dvr_dec_api.h](#).

Field Documentation

int output_type

DecoderOutputColorTag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **16** of file [dvr_dec_api.h](#).

int reserved[2]

Definition at line **17** of file [dvr_dec_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

DIM_tag Struct Reference

set width and height More...

Data Fields

int **width**

int **height**

Detailed Description

set width and height

Examples:

[2ch_liveview.c](#), [liveview.c](#), and [pip.c](#).

Definition at line [156](#) of file [dvr_type_define.h](#).

Field Documentation

int width

width in pixel

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [pip.c](#), [playback.c](#), [roi.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [157](#) of file [dvr_type_define.h](#).

int height

height in pixel

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [pip.c](#), [playback.c](#), [roi.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [158](#) of file [dvr_type_define.h](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

DispParam_Ext1_tag Struct Reference

Data Fields

dvr_disp_channel_type chn_type

Detailed Description

Examples:

[pip.c.](#)

Definition at line [216](#) of file [dvr_disp_api.h](#).

Field Documentation

dvr_disp_channel_type chn_type

dvr_disp_channel_type_tag

Examples:

[pip.c.](#)

Definition at line [218](#) of file [dvr_disp_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_bs_data_tag Struct Reference

dvr bit-stream data [More...](#)

Data Fields

```
struct timeval timestamp
    int offset
    int length
    int is_keyframe
    int mv_offset
    int mv_length
    void * p_job
    int stream
    int NonRef
    int reserved [8]
```

Detailed Description

dvr bit-stream data

Definition at line [216](#) of file [dvr_type_define.h](#).

Field Documentation

struct timeval timestamp

Definition at line [217](#) of file [dvr_type_define.h](#).

int offset

bitstream buffer offset.

Examples:

[2ch_playback.c](#), [capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [playback.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [218](#) of file [dvr_type_define.h](#).

int length

bitstream buffer length.

Examples:

[2ch_playback.c](#), [capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [playback.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [219](#) of file [dvr_type_define.h](#).

int is_keyframe

1: current frame is a keyframe. 0: current frame is not a keyframe.

Definition at line [220](#) of file [dvr_type_define.h](#).

int mv_offset

motion vector offset

Examples:

[motion-detection-mpeg4.c](#), and [motion-detection.c](#).

Definition at line [221](#) of file [dvr_type_define.h](#).

int mv_length

motion vector length

Definition at line [222](#) of file [dvr_type_define.h](#).

void* p_job

internal use

Definition at line [225](#) of file [dvr_type_define.h](#).

int stream

internal use

Definition at line [226](#) of file [dvr_type_define.h](#).

int NonRef

0: current P frame was encoded as referenced.

1: current P frame was encoded as non-referenced

Definition at line [229](#) of file [dvr_type_define.h](#).

int reserved[8]

Definition at line [230](#) of file [dvr_type_define.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_dec_channel_param_tag Struct Reference

Data Fields

int	dec_type
int	dec_dest_type
int	channel
int	is_blocked
int	is_use_scaler
DecParam	dec_param
ScalerParam	scl_param
int	reserved [8]

Detailed Description

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line [20](#) of file [dvr_dec_api.h](#).

Field Documentation

`int dec_type`

`EncodeType_tag`

Examples:

`2ch_playback.c`, and `playback.c`.

Definition at line **22** of file `dvr_dec_api.h`.

`int dec_dest_type`

`dvr_dec_dest_type_tag`

Definition at line **24** of file `dvr_dec_api.h`.

`int channel`

Examples:

`2ch_playback.c`, and `playback.c`.

Definition at line **25** of file `dvr_dec_api.h`.

`int is_blocked`

Definition at line **26** of file `dvr_dec_api.h`.

`int is_use_scaler`

Examples:

`2ch_playback.c`, and `playback.c`.

Definition at line **27** of file [dvr_dec_api.h](#).

DecParam dec_param

DecParam_tag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **29** of file [dvr_dec_api.h](#).

ScalerParam scl_param

ScalerParamtag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **31** of file [dvr_dec_api.h](#).

int reserved[8]

Definition at line **32** of file [dvr_dec_api.h](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_dec_clear_param_tag Struct Reference

Data Fields

RECT **win**
unsigned int **pattern**
int **reserved** [2]

Detailed Description

Definition at line [74](#) of file [dvr_dec_api.h](#).

Field Documentation

RECT win

RECT_tag

Definition at line **77** of file [dvr_dec_api.h](#).

unsigned int pattern

Definition at line **78** of file [dvr_dec_api.h](#).

int reserved[2]

Definition at line **79** of file [dvr_dec_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_dec_control_tag Struct Reference

Data Fields

```
dvr_dec_ctrl_cmd command
struct {
    DIM dim
    RECT win
    int bs_rate
    int reserved [2]
}
src_param
struct {
    int plane_id
    RECT win
    int is_display
    int display_rate
    int reserved [2]
}
dst_param
```

Detailed Description

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **50** of file [dvr_dec_api.h](#).

Field Documentation

dvr_dec_ctrl_cmd command

dvr_dec_ctrl_cmd_tag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **52** of file [dvr_dec_api.h](#).

DIM dim

DIM_tag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **56** of file [dvr_dec_api.h](#).

RECT win

RECT_tag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **58** of file [dvr_dec_api.h](#).

int bs_rate

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **59** of file [dvr_dec_api.h](#).

int reserved[2]

Definition at line **60** of file [dvr_dec_api.h](#).

struct { ... } src_param

Examples:

[2ch_playback.c](#), and [playback.c](#).

int plane_id

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **64** of file [dvr_dec_api.h](#).

int is_display

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **67** of file [dvr_dec_api.h](#).

int display_rate

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **68** of file [dvr_dec_api.h](#).

struct { ... } dst_param

Examples:

[2ch_playback.c](#), and [playback.c](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_dec_queue_get_tag Struct Reference

Data Fields

dvr_bs_data bs

int channel

int reserved [1]

Detailed Description

Definition at line [36](#) of file [dvr_dec_api.h](#).

Field Documentation

dvr_bs_data bs

dvr_bs_data_tag

Definition at line **38** of file [dvr_dec_api.h](#).

int channel

Definition at line **39** of file [dvr_dec_api.h](#).

int reserved[1]

Definition at line **40** of file [dvr_dec_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_clear_param_tag Struct Reference

Data Fields

int	plane_id
RECT	win
unsigned int	pattern
int	reserved [2]

Detailed Description

Examples:

[**2ch_liveview.c.**](#)

Definition at line [**340**](#) of file [**dvr_disp_api.h**](#).

Field Documentation

int plane_id

Examples:

[2ch_liveview.c](#).

Definition at line [342](#) of file [dvr_disp_api.h](#).

RECT win

RECT_tag

Examples:

[2ch_liveview.c](#).

Definition at line [344](#) of file [dvr_disp_api.h](#).

unsigned int pattern

Examples:

[2ch_liveview.c](#).

Definition at line [345](#) of file [dvr_disp_api.h](#).

int reserved[2]

Definition at line [346](#) of file [dvr_disp_api.h](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_color_attribute_tag Struct Reference

Data Fields

int	brightness
int	saturation
int	contrast
int	huesin
int	huecos
int	sharpnessk0
int	sharpnessk1
int	sharpness_thres0
int	shaprness_thres1
int	reserved [2]

Detailed Description

Definition at line [32](#) of file [dvr_disp_api.h](#).

Field Documentation

int brightness

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **33** of file [dvr_disp_api.h](#).

int saturation

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **34** of file [dvr_disp_api.h](#).

int contrast

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **35** of file [dvr_disp_api.h](#).

int huesin

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **36** of file [dvr_disp_api.h](#).

int huecos

Examples:

2ch_playback.c, and **playback.c**.

Definition at line **37** of file **dvr_disp_api.h**.

int sharpnessk0

Examples:

2ch_playback.c, and **playback.c**.

Definition at line **38** of file **dvr_disp_api.h**.

int sharpnessk1

Examples:

2ch_playback.c, and **playback.c**.

Definition at line **39** of file **dvr_disp_api.h**.

int sharpness_thres0

Examples:

2ch_playback.c, and **playback.c**.

Definition at line **40** of file **dvr_disp_api.h**.

int shaprness_thres1

Examples:

2ch_playback.c, and **playback.c**.

Definition at line **41** of file **dvr_disp_api.h**.

int reserved[2]

Definition at line **42** of file **dvr_disp_api.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_color_key_tag Struct Reference

Data Fields

int	is_enable
int	color
int	reserved [2]

Detailed Description

Definition at line [45](#) of file [dvr_disp_api.h](#).

Field Documentation

int is_enable

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **46** of file [dvr_disp_api.h](#).

int color

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **47** of file [dvr_disp_api.h](#).

int reserved[2]

Definition at line **48** of file [dvr_disp_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_control_tag Struct Reference

Data Fields

```
int type
int channel
dvr_disp_ctrl_cmd command
int reserved [8]

struct {
    struct {
        int cap_path
        int di_mode
        int mode
        int dma_order
        int scale_indep
        int input_system
        int cap_rate
        int color_mode
        int is_use_scaler
    DIM dim
    RECT win
    video_process vp_param
    ScalerParam scl_param
        int cap_buf_id
        int ext_size
        void * pext_data
        int reserved [2]
    } lv
    struct {
        RECT win
        int rate
        int di_mode
        int mode
        int dma_order
        int scale_indep
        int input_system
    }
}
```

```
int cap_rate
int color_mode
int is_use_scaler
DIM dim
video_process vp_param
ScalerParam scl_param
int reserved [5]
} cas
int reserved [5]
}
struct {
    struct {
        RECT win
        int plane_id
        int reserved [5]
    } lv
    struct {
        int path
        RECT win
        RECT rect0
        RECT rect1
        RECT swc_rect0
        RECT swc_rect1
        RECT bg_rect0
        RECT bg_rect1
        int plane_id
        int reserved [5]
    } cas
    int reserved [5]
}
```

src_param

```
}

}
```

dst_param

Detailed Description

Examples:

[**2ch_liveview.c**](#), [**2ch_playback.c**](#), [**liveview.c**](#), [**pip.c**](#), and [**playback.c**](#).

Definition at line [**230**](#) of file [**dvr_disp_api.h**](#).

Field Documentation

int type

dvr_disp_type

Examples:

[2ch_liveview.c](#), [liveview.c](#), and [pip.c](#).

Definition at line [232](#) of file [dvr_disp_api.h](#).

int channel

Examples:

[2ch_liveview.c](#), [liveview.c](#), and [pip.c](#).

Definition at line [233](#) of file [dvr_disp_api.h](#).

dvr_disp_ctrl_cmd command

dvr_disp_ctrl_cmd_tag

Examples:

[2ch_liveview.c](#), [liveview.c](#), and [pip.c](#).

Definition at line [235](#) of file [dvr_disp_api.h](#).

int reserved[5]

Definition at line [237](#) of file [dvr_disp_api.h](#).

int cap_path

Definition at line [242](#) of file `dvr_disp_api.h`.

int di_mode

LiveviewFrameTypeTag

Definition at line [244](#) of file `dvr_disp_api.h`.

int mode

LiveviewFrameModeTag

Definition at line [246](#) of file `dvr_disp_api.h`.

int dma_order

LiveviewDMAOrderTag

Definition at line [248](#) of file `dvr_disp_api.h`.

int scale_indep

0:fixed aspect-ratio

LiveviewScalerRatioTag

Definition at line [250](#) of file `dvr_disp_api.h`.

int input_system

MCP_VIDEO_NTSC

MCP_VIDEO_PAL

MCP_VIDEO_VGA

Definition at line **254** of file **dvr_disp_api.h**.

int cap_rate

Definition at line **255** of file **dvr_disp_api.h**.

int color_mode

CaptureColorModeTag

Definition at line **257** of file **dvr_disp_api.h**.

int is_use_scaler

Definition at line **258** of file **dvr_disp_api.h**.

DIM dim

DIM_tag

Definition at line **261** of file **dvr_disp_api.h**.

RECT win

RECT_tag

Definition at line **263** of file **dvr_disp_api.h**.

video_process vp_param

video_process_tag

Definition at line **265** of file **dvr_disp_api.h**.

ScalerParam scl_param

if is_use_scaler is TRUE, user needs to fill this structure

ScalerParamtag

Definition at line **267** of file **dvr_disp_api.h**.

int cap_buf_id

Definition at line **268** of file **dvr_disp_api.h**.

int ext_size

display parameter extension size

Definition at line **270** of file **dvr_disp_api.h**.

void* pext_data

point to display parameter extension structure

Definition at line **272** of file **dvr_disp_api.h**.

struct { ... } lv

Examples:

[2ch_liveview.c](#), [liveview.c](#), and [pip.c](#).

int rate

Definition at line **278** of file [dvr_disp_api.h](#).

struct { ... } cas

struct { ... } src_param

Examples:

[2ch_liveview.c](#), [liveview.c](#), and [pip.c](#).

int plane_id

Definition at line **312** of file [dvr_disp_api.h](#).

struct { ... } lv

int path

Definition at line **317** of file [dvr_disp_api.h](#).

RECT rect0

RECT_tag

Definition at line **321** of file [dvr_disp_api.h](#).

RECT rect1

RECT_tag

Definition at line **323** of file **dvr_disp_api.h**.

RECT swc_rect0

RECT_tag

Definition at line **325** of file **dvr_disp_api.h**.

RECT swc_rect1

RECT_tag

Definition at line **327** of file **dvr_disp_api.h**.

RECT bg_rect0

RECT_tag

Definition at line **329** of file **dvr_disp_api.h**.

RECT bg_rect1

RECT_tag

Definition at line **331** of file **dvr_disp_api.h**.

struct { ... } cas

struct { ... } dst_param

Examples:

2ch_liveview.c, liveview.c, and pip.c.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_disp_param_tag Struct Reference

Data Fields

int	disp_num
int	target_id
int	plane_comb
int	output_system
int	output_mode
dvr_disp_color_attribute	color_attrib
dvr_disp_color_key	transparent_color [2]
dvr_disp_vbi_info	vbi_info
dvr_disp_scaler_info	scl_info
DIM	dim
dvr_disp_resolution	res
int	reserved [8]

Detailed Description

Examples:

[**2ch_liveview.c**](#), [**2ch_playback.c**](#), [**liveview.c**](#), [**pip.c**](#), and [**playback.c**](#).

Definition at line **80** of file [**dvr_disp_api.h**](#).

Field Documentation

`int disp_num`

input(write only)

Examples:

`2ch_liveview.c`, `2ch_playback.c`, `liveview.c`, `pip.c`, and `playback.c`.

Definition at line **81** of file `dvr_disp_api.h`.

`int target_id`

where to output. 1:LCD1, 2:LCD2

Definition at line **83** of file `dvr_disp_api.h`.

`int plane_comb`

the number of active plane for this LCD

`dvr_disp_plane_combination_tag`

Examples:

`2ch_playback.c`, and `playback.c`.

Definition at line **85** of file `dvr_disp_api.h`.

`int output_system`

`MCP_VIDEO_NTSC`

`MCP_VIDEO_PAL`

MCP_VIDEO_VGA

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **89** of file [dvr_disp_api.h](#).

int output_mode

LCDOutputModeTag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **91** of file [dvr_disp_api.h](#).

dvr_disp_color_attribute color_attrib

dvr_disp_color_attribute_tag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **93** of file [dvr_disp_api.h](#).

dvr_disp_color_key transparent_color[2]

dvr_disp_color_key_tag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **95** of file [dvr_disp_api.h](#).

dvr_disp_vbi_info vbi_info

dvr_disp_vbi_info_tag

Definition at line **97** of file **dvr_disp_api.h**.

dvr_disp_scaler_info scl_info

dvr_disp_scaler_info_tag

Definition at line **99** of file **dvr_disp_api.h**.

DIM dim

default background dimension

DIM_tag

Examples:

2ch_playback.c, and **playback.c**.

Definition at line **101** of file **dvr_disp_api.h**.

dvr_disp_resolution res

dvr_disp_resolution_tag

Examples:

2ch_playback.c, and **playback.c**.

Definition at line **103** of file **dvr_disp_api.h**.

int reserved[8]

Definition at line **104** of file **dvr_disp_api.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_plane_param_st_tag Struct Reference

Data Fields

int	disp_num
int	plane_num
dvr_disp_plane_param_st	param
int	reserved [2]

Detailed Description

Examples:

[**2ch_liveview.c**](#), [**2ch_playback.c**](#), [**liveview.c**](#), [**pip.c**](#), and [**playback.c**](#).

Definition at line [**166**](#) of file [**dvr_disp_api.h**](#).

Field Documentation

int disp_num

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [168](#) of file [dvr_disp_api.h](#).

int plane_num

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [169](#) of file [dvr_disp_api.h](#).

dvr_disp_plane_param_st param

dvr_disp_plane_param_tag

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [171](#) of file [dvr_disp_api.h](#).

int reserved[2]

Definition at line [172](#) of file [dvr_disp_api.h](#).

doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_plane_param_tag Struct Reference

Data Fields

int	plane_id
RECT	win
int	data_mode
int	color_mode
int	reserved [2]

Detailed Description

Definition at line [155](#) of file [dvr_disp_api.h](#).

Field Documentation

int plane_id

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [156](#) of file [dvr_disp_api.h](#).

RECT win

RECT_tag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line [158](#) of file [dvr_disp_api.h](#).

int data_mode

LCDOutputModeTag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line [160](#) of file [dvr_disp_api.h](#).

int color_mode

LCDOutputColorTypeTag

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **162** of file **dvr_disp_api.h**.

int reserved[2]

Definition at line **163** of file **dvr_disp_api.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_resolution_tag Struct Reference

Data Fields

```
int input_res
int output_type
int reserved [2]
```

Detailed Description

Definition at line [73](#) of file [dvr_disp_api.h](#).

Field Documentation

int input_res

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **74** of file [dvr_disp_api.h](#).

int output_type

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **75** of file [dvr_disp_api.h](#).

int reserved[2]

Definition at line **76** of file [dvr_disp_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_scaler_info_tag Struct Reference

Data Fields

int	is_enable
DIM	dim
int	reserved [2]

Detailed Description

Definition at line **66** of file [dvr_disp_api.h](#).

Field Documentation

int is_enable

Definition at line **67** of file [dvr_disp_api.h](#).

DIM dim

DIM_tag

Definition at line **69** of file [dvr_disp_api.h](#).

int reserved[2]

Definition at line **70** of file [dvr_disp_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields		

Data Structures | Data Fields

dvr_disp_update_disp_param_tag Struct Reference

Data Structures

```
struct val_t
```

Data Fields

	int	disp_num
dvr_disp_disp_param_name	param	
	struct	val
dvr_disp_update_disp_param_tag::val_t		

Detailed Description

Examples:

[**2ch_liveview.c**](#), [**2ch_playback.c**](#), [**liveview.c**](#), [**pip.c**](#), and [**playback.c**](#).

Definition at line [**122**](#) of file [**dvr_disp_api.h**](#).

Field Documentation

int disp_num

input

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [123](#) of file [dvr_disp_api.h](#).

dvr_disp_disp_param_name param

dvr_disp_disp_param_name_tag

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [125](#) of file [dvr_disp_api.h](#).

struct dvr_disp_update_disp_param_tag::val_t val

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_update_plane_param_tag Struct Reference

Data Fields

```
int plane_id  
dvr_disp_plane_param_name param  
int reserved [8]  
union {  
    RECT win  
    int data_mode  
    int color_mode  
}  
val  
int reserved1 [8]
```

Detailed Description

Examples:

[**2ch_liveview.c**](#), [**2ch_playback.c**](#), [**liveview.c**](#), [**pip.c**](#), and [**playback.c**](#).

Definition at line [**184**](#) of file [**dvr_disp_api.h**](#).

Field Documentation

int plane_id

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [186](#) of file [dvr_disp_api.h](#).

dvr_disp_plane_param_name param

dvr_disp_plane_param_name_tag

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [188](#) of file [dvr_disp_api.h](#).

int reserved[8]

Definition at line [189](#) of file [dvr_disp_api.h](#).

RECT win

RECT_tag

Definition at line [192](#) of file [dvr_disp_api.h](#).

int data_mode

LCDOutputModeTag

Examples:

2ch_liveview.c, **2ch_playback.c**, **liveview.c**, **pip.c**, and **playback.c**.

Definition at line **194** of file **dvr_disp_api.h**.

int color_mode

LCDOutputColorTypeTag

Examples:

2ch_liveview.c, **2ch_playback.c**, **liveview.c**, **pip.c**, and **playback.c**.

Definition at line **196** of file **dvr_disp_api.h**.

union { ... } val

Examples:

2ch_liveview.c, **2ch_playback.c**, **liveview.c**, **pip.c**, and **playback.c**.

int reserved1[8]

Definition at line **198** of file **dvr_disp_api.h**.

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_disp_vbi_info_tag Struct Reference

Data Fields

int	filler
int	lineno
int	lineheight
int	fb_offset
int	fb_size
int	reserved [2]

Detailed Description

Definition at line [57](#) of file [dvr_disp_api.h](#).

Field Documentation

int filler

0:driver do it, 1:ap can set values for it(need to fill the following fields)

Definition at line **58** of file [dvr_disp_api.h](#).

int lineno

Definition at line **59** of file [dvr_disp_api.h](#).

int lineheight

Definition at line **60** of file [dvr_disp_api.h](#).

int fb_offset

Definition at line **61** of file [dvr_disp_api.h](#).

int fb_size

Definition at line **62** of file [dvr_disp_api.h](#).

int reserved[2]

Definition at line **63** of file [dvr_disp_api.h](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_enc_channel_param_tag Struct Reference

dvr encode channel parameter. [More...](#)

Data Fields

dvr_enc_src_param src
ReproduceBitStream main_bs

Detailed Description

dvr encode channel parameter.

Examples:

`capture_raw.c`, `main-bitstream-record.c`, `mjpeg-record.c`,
`motion-detection-mpeg4.c`, `motion-detection.c`, `mpeg4-record.c`, `roi.c`, `snapshot.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [264](#) of file `dvr_enc_api.h`.

Field Documentation

`dvr_enc_src_param src`

Examples:

`capture_raw.c`, `main-bitstream-record.c`, `mjpeg-record.c`,
`motion-detection-mpeg4.c`, `motion-detection.c`, `mpeg4-record.c`, `roi.c`, `snapshot.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line **265** of file `dvr_enc_api.h`.

`ReproduceBitStream main_bs`

Examples:

`capture_raw.c`, `main-bitstream-record.c`, `mjpeg-record.c`,
`motion-detection-mpeg4.c`, `motion-detection.c`, `mpeg4-record.c`, `roi.c`, `snapshot.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line **266** of file `dvr_enc_api.h`.

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_enc_control_tag Struct Reference

dvr encode control parameter [More...](#)

Data Fields

	int command
union {	
int * count	
int * repd_bs_num	
}	output
	int stream
dvr_enc_update_channel_param	update_parm
	int reserved [8]

Detailed Description

dvr encode control parameter

Examples:

`capture_raw.c`, `main-bitstream-record.c`, `mjpeg-record.c`,
`motion-detection-mpeg4.c`, `motion-detection.c`, `mpeg4-record.c`, `roi.c`, `snapshot.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [332](#) of file `dvr_enc_api.h`.

Field Documentation

int command

enc control command , such as ENC_START, ENC_STOP, etc.

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [333](#) of file [dvr_enc_api.h](#).

int* count

Examples:

[capture_raw.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), and [snapshot.c](#).

Definition at line [335](#) of file [dvr_enc_api.h](#).

int* repd_bs_num

0: all, 1: SUB1, 2: SUB2.

Definition at line [336](#) of file [dvr_enc_api.h](#).

union { ... } output

Examples:

[capture_raw.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), and [snapshot.c](#).

int stream

stream number 0 : main, 1 : sub stream 1, 2 : sub stream 2

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#),
[motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [338](#) of file [dvr_enc_api.h](#).

dvr_enc_update_channel_param update_parm

dvr_enc_update_channel_param_tag

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [340](#) of file [dvr_enc_api.h](#).

int reserved[8]

Definition at line [341](#) of file [dvr_enc_api.h](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_enc_copy_buf_tag Struct Reference

Data Fields

unsigned int	bs_user_va
unsigned int	mv_user_va
unsigned int	bs_buf_length
unsigned int	mv_buf_length
struct timeval	timestamp
unsigned int	bs_length
unsigned int	mv_length
unsigned int	is_keyframe
int	stream
int	NonRef
int	new_bs
void *	p_job
int	reserved [16]

Detailed Description

Definition at line [344](#) of file `dvr_enc_api.h`.

Field Documentation

unsigned int bs_user_va

bitstream user virtual address

Definition at line [346](#) of file [dvr_enc_api.h](#).

unsigned int mv_user_va

MV user virtual address 0 means nocopy.

Definition at line [347](#) of file [dvr_enc_api.h](#).

unsigned int bs_buf_length

bitstream length by AP prepared, and return the real bitstream length

Definition at line [348](#) of file [dvr_enc_api.h](#).

unsigned int mv_buf_length

motion length by AP prepared, and return the real motion length

Definition at line [349](#) of file [dvr_enc_api.h](#).

struct timeval timestamp

Definition at line [352](#) of file [dvr_enc_api.h](#).

unsigned int bs_length

bitstream length by AP prepared, and return the real bitstream length

Definition at line [353](#) of file [dvr_enc_api.h](#).

unsigned int mv_length

motion length by AP prepared, and return the real motion length

Definition at line [354](#) of file [dvr_enc_api.h](#).

unsigned int is_keyframe

1: current frame is a keyframe. 0: current frame is not a keyframe.

Definition at line [356](#) of file [dvr_enc_api.h](#).

int stream

Definition at line [357](#) of file [dvr_enc_api.h](#).

int NonRef

Definition at line [358](#) of file [dvr_enc_api.h](#).

int new_bs

To indicate whether the packet using new setting.

Definition at line [359](#) of file [dvr_enc_api.h](#).

void* p_job

internal use

Definition at line **360** of file **dvr_enc_api.h**.

int reserved[16]

Definition at line **361** of file **dvr_enc_api.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_enc_queue_get_tag Struct Reference

get dvr encode buffer [More...](#)

Data Fields

dvr_bs_data	bs
int	new_bs
int	mb_len
int	channel

Detailed Description

get dvr encode buffer

Examples:

2ch_playback.c, capture_raw.c, main-bitstream-record.c, mjpeg-record.c, motion-detection-mpeg4.c, motion-detection.c, mpeg4-record.c, playback.c, roi.c, snapshot.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **277** of file **dvr_enc_api.h**.

Field Documentation

dvr_bs_data bs

dvr_bs_data_tag

Examples:

[2ch_playback.c](#), [capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [playback.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [279](#) of file [dvr_enc_api.h](#).

int new_bs

To indicate whether the packet using new setting.

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [280](#) of file [dvr_enc_api.h](#).

int mb_len

Only for debug H.264: D1:103680 HD1:50688 CIF:25344.

Definition at line [281](#) of file [dvr_enc_api.h](#).

int channel

Only for debug H.264: sub ==> bs.stream.

Definition at line **282** of file **dvr_enc_api.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_enc_src_tag Struct Reference

dvr encode source parameter [More...](#)

Data Fields

int	channel
int	enc_src_type
DIM	dim
int	di_mode
int	mode
int	dma_order
int	scale_indep
int	input_system
int	color_mode
video_process	vp_param
int	reserved [8]

Detailed Description

dvr encode source parameter

Definition at line [235](#) of file [dvr_enc_api.h](#).

Field Documentation

`int channel`

channle number 0~7

Examples:

`capture_raw.c`, `main-bitstream-record.c`, `mjpeg-record.c`,
`mpeg4-record.c`, `roi.c`, `snapshot.c`, `sub-bitstream-record.c`,
`update-bitrate.c`, and `update-record-setting.c`.

Definition at line [237](#) of file `dvr_enc_api.h`.

`int enc_src_type`

`dvr_enc_src_type_tag`

Examples:

`main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`,
`sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [239](#) of file `dvr_enc_api.h`.

`DIM dim`

`DIM_tag`

Examples:

`main-bitstream-record.c`, `mjpeg-record.c`, `motion-detection-mpeg4.c`, `motion-detection.c`, `mpeg4-record.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [241](#) of file `dvr_enc_api.h`.

int di_mode

LiveviewFrameTypeTag

Examples:

main-bitstream-record.c, mjpeg-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **243** of file **dvr_enc_api.h**.

int mode

LiveviewFrameModeTag

Examples:

main-bitstream-record.c, mjpeg-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **245** of file **dvr_enc_api.h**.

int dma_order

LiveviewDMAOrderTag

Examples:

main-bitstream-record.c, mjpeg-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **247** of file **dvr_enc_api.h**.

int scale_indep

LiveviewScalerRatioTag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [249](#) of file [dvr_enc_api.h](#).

int input_system

MCP_VIDEO_NTSC

MCP_VIDEO_PAL

MCP_VIDEO_VGA

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [253](#) of file [dvr_enc_api.h](#).

int color_mode

CaptureColorModeTag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [255](#) of file [dvr_enc_api.h](#).

video_process vp_param

`video_process_tag`

Examples:

`main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`,
`sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [257](#) of file `dvr_enc_api.h`.

`int reserved[8]`

Definition at line [258](#) of file `dvr_enc_api.h`.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_enc_update_channel_param_tag Struct Reference

update dvr encode channel parameter [More...](#)

Data Fields

```
struct {
    int di_mode
    int mode
    int scale_indep
    int is_3DI
    int is_denoise
    int reserved [8]
}                      src
                    int stream_enable
                    int frame_rate
                    int bit_rate
                    int ip_interval
                    DIM dim
                    int init_quant
                    int max_quant
                    int min_quant
    unsigned int ext_size
    void * pext_data
```

Detailed Description

update dvr encode channel parameter

Definition at line [298](#) of file [dvr_enc_api.h](#).

Field Documentation

int di_mode

LiveviewFrameTypeTag

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [302](#) of file [dvr_enc_api.h](#).

int mode

LiveviewFrameModeTag

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [304](#) of file [dvr_enc_api.h](#).

int scale_indep

LiveviewScalerRatioTag

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [306](#) of file [dvr_enc_api.h](#).

int is_3DI

TRUE : enable 3D deinterlace , False : disable 3D deinterlace

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [308](#) of file [dvr_enc_api.h](#).

int is_denoise

TRUE : enable 3D denoise , False : disable 3D denoise

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [310](#) of file [dvr_enc_api.h](#).

int reserved

Definition at line [311](#) of file [dvr_enc_api.h](#).

struct { ... } src

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

int stream_enable

0:disable, 1:enable

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [313](#) of file [dvr_enc_api.h](#).

int frame_rate

frame rate per second

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [314](#) of file [dvr_enc_api.h](#).

int bit_rate

Bitrate per second.

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [315](#) of file [dvr_enc_api.h](#).

int ip_interval

The I-P interval frames.

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [316](#) of file [dvr_enc_api.h](#).

DIM dim

DIM_tag

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [318](#) of file [dvr_enc_api.h](#).

int init_quant

The initial quant value.

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [319](#) of file [dvr_enc_api.h](#).

int max_quant

The max quant value.

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [320](#) of file [dvr_enc_api.h](#).

int min_quant

The min quant value.

Examples:

[roi.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [321](#) of file [dvr_enc_api.h](#).

unsigned int ext_size

encode parameter extension size

Examples:

[roi.c](#).

Definition at line [323](#) of file [dvr_enc_api.h](#).

void* pext_data

point to encode parameter extension structure

Examples:

[roi.c](#).

Definition at line [325](#) of file [dvr_enc_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_graph_vqueue_t tag Struct Reference

set dvr graph queue configuration More...

Data Fields

```
struct v_queue_t * que
    int size
    int count
    int ddr
    int limit_count
    FuncTag user_tag
```

Detailed Description

set dvr graph queue configuration

Definition at line [203](#) of file [dvr_type_define.h](#).

Field Documentation

struct v_queue_t* que

Definition at line [205](#) of file [dvr_type_define.h](#).

int size

size of buffer

Definition at line [206](#) of file [dvr_type_define.h](#).

int count

the number of buffers

Definition at line [207](#) of file [dvr_type_define.h](#).

int ddr

the DDR number

Definition at line [208](#) of file [dvr_type_define.h](#).

int limit_count

Max count of in-used buffer.

Definition at line [209](#) of file [dvr_type_define.h](#).

FuncTag user_tag

function tag for videograph level

Definition at line **210** of file **dvr_type_define.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

dvr_rate_tag Struct Reference

Data Fields

```
int numerator
int denominator
int reserved [4]
```

Detailed Description

Definition at line [233](#) of file [dvr_type_define.h](#).

Field Documentation

int numerator

Not in use.

Definition at line [234](#) of file [dvr_type_define.h](#).

int denominator

Not in use.

Definition at line [235](#) of file [dvr_type_define.h](#).

int reserved[4]

Not in use.

Definition at line [236](#) of file [dvr_type_define.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

EncParam_Ext1_tag Struct Reference

no longer used, replace by EncParam_Ext4 [More...](#)

Data Fields

unsigned int **feature_enable**
unsigned int **target_rate_max**
unsigned int **reaction_delay_max**

Detailed Description

no longer used, replace by EncParam_Ext4

Definition at line **18** of file [dvr_enc_api.h](#).

Field Documentation

unsigned int feature_enable

Definition at line **19** of file [dvr_enc_api.h](#).

unsigned int target_rate_max

Definition at line **20** of file [dvr_enc_api.h](#).

unsigned int reaction_delay_max

Definition at line **21** of file [dvr_enc_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

EncParam_Ext2_tag Struct Reference

no longer used, replace by EncParam_Ext4 [More...](#)

Data Fields

unsigned int	feature_enable
unsigned int	target_rate_max
unsigned int	reaction_delay_max
int	enc_type
int	MJ_quality

Detailed Description

no longer used, replace by EncParam_Ext4

Definition at line **27** of file [dvr_enc_api.h](#).

Field Documentation

unsigned int feature_enable

Definition at line **28** of file [dvr_enc_api.h](#).

unsigned int target_rate_max

Definition at line **29** of file [dvr_enc_api.h](#).

unsigned int reaction_delay_max

Definition at line **30** of file [dvr_enc_api.h](#).

int enc_type

Definition at line **31** of file [dvr_enc_api.h](#).

int MJ_quality

Definition at line **32** of file [dvr_enc_api.h](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

EncParam_Ext3_tag Struct Reference

no longer used, replace by EncParam_Ext4 [More...](#)

Data Fields

unsigned int	feature_enable
unsigned int	target_rate_max
unsigned int	reaction_delay_max
int	enc_type
int	MJ_quality
int	watermark_enable
int	watermark_interval
int	watermark_init_pattern
int	watermark_init_interval

Detailed Description

no longer used, replace by EncParam_Ext4

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [38](#) of file [dvr_enc_api.h](#).

Field Documentation

unsigned int feature_enable

this is a bit combination. If you want to involve watermark, then feature_enable |= DVR_ENC_H264_WATERMARK.

[DVR_ENC_ENHANCE_H264_RATECONTROL](#)

[DVR_ENC_MJPEG_FUNCTION](#)

[DVR_ENC_H264_WATERMARK](#)

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [42](#) of file [dvr_enc_api.h](#).

unsigned int target_rate_max

ECBR: usually stays at target bitrate,

produce large bitrate when big motion, but never more than max bitrate

Definition at line [45](#) of file [dvr_enc_api.h](#).

unsigned int reaction_delay_max

max frame count will the rate-control stay when away from the target bitrate, only valid at ECBR

Definition at line [47](#) of file [dvr_enc_api.h](#).

int enc_type

EncodeType_tag

Definition at line **49** of file [dvr_enc_api.h](#).

int MJ_quality

Motion JPEG quality.

1(worst) ~100(best)

Examples:

[mjpeg-record.c](#).

Definition at line **51** of file [dvr_enc_api.h](#).

int watermark_enable

Indicate watermark enable or not

if 0: disable,

if 1: enable, but only insert watermark at intra-mb,

if 3: enable, and insert watermark at intra/inter-mb,

if others: not allowed

Definition at line **57** of file [dvr_enc_api.h](#).

int watermark_interval

specify the interval, to insert watermark each "interval" frame,

valid when watermark function is enabled,
if 1: will be inserted watermark each frames,
if N: will be inserted watermark each N frames

Definition at line **62** of file [dvr_enc_api.h](#).

`int watermark_init_pattern`

The initial pattern for watermark
valid when watermark function is enabled
Definition at line **65** of file [dvr_enc_api.h](#).

`int watermark_init_interval`

specify the interval, to reinit with init_pattern each "watermark"ed frames
valid when watermark function is enabled
if 1: every "watermark"ed frame with init_pattern
if N: initial with init_pattern each N "watermark"ed frame
Definition at line **70** of file [dvr_enc_api.h](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

EncParam_Ext4_tag Struct Reference

encode parameter extension [More...](#)

Data Fields

```
unsigned int feature_enable  
unsigned int target_rate_max  
unsigned int reaction_delay_max  
    int enc_type  
    int MJ_quality  
    int watermark_enable  
    int watermark_interval  
    int watermark_init_pattern  
    int watermark_init_interval  
POS roi_pos
```

Detailed Description

encode parameter extension

Definition at line **76** of file [dvr_enc_api.h](#).

Field Documentation

unsigned int feature_enable

this is a bit combination. If you want to involve watermark, then
feature_enable |= DVR_ENC_H264_WATERMARK.

DVR_ENC_ENHANCE_H264_RATECONTROL

DVR_ENC_MJPEG_FUNCTION

DVR_ENC_H264_WATERMARK

DVR_ENC_ROI_POS

Definition at line **81** of file **dvr_enc_api.h**.

unsigned int target_rate_max

for New rate control

ECBR: usually stays at target bitrate,

produce large bitrate when big motion, but never more than max
bitrate

Definition at line **85** of file **dvr_enc_api.h**.

unsigned int reaction_delay_max

max frame count will the rate-control stay when away from the target
bitrate, only valid at ECBR

Definition at line **87** of file **dvr_enc_api.h**.

int enc_type

for MJPEG

EncodeType_tag

Definition at line **90** of file [dvr_enc_api.h](#).

int MJ_quality

Motion JPEG quality.

1(worst) ~100(best)

Definition at line **92** of file [dvr_enc_api.h](#).

int watermark_enable

for H264 Water Mark

Indicate watermark enable or not

if 0: disable,

if 1: enable, but only insert watermark at intra-mb,

if 3: enable, and insert warermark at intra/inter-mb,

if others: not allowed

Definition at line **99** of file [dvr_enc_api.h](#).

int watermark_interval

specify the interval, to insert watermark each "interval" frame,

valid when watermark function is enabled,
if 1: will be inserted watermark each frames,
if N: will be inserted watermark each N frames

Definition at line [104](#) of file [dvr_enc_api.h](#).

int watermark_init_pattern

The initial pattern for watermark
valid when watermark function is enabled
Definition at line [107](#) of file [dvr_enc_api.h](#).

int watermark_init_interval

specify the interval, to reinit with init_pattern each "watermark"ed frames
valid when watermark function is enabled
if 1: every "watermark"ed frame with init_pattern
if N: initial with init_pattern each N "watermark"ed frame
Definition at line [112](#) of file [dvr_enc_api.h](#).

POS roi_pos

for ROI x, y position
Definition at line [114](#) of file [dvr_enc_api.h](#).

doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

EncParam_Ext5_tag Struct Reference

Data Fields

unsigned int	feature_enable
unsigned int	target_rate_max
unsigned int	reaction_delay_max
int	enc_type
int	MJ_quality
int	watermark_enable
int	watermark_interval
int	watermark_init_pattern
int	watermark_init_interval
POS	roi_pos
ROI_ALL	roi_all

Detailed Description

Examples:

[roi.c.](#)

Definition at line [117](#) of file [dvr_enc_api.h](#).

Field Documentation

unsigned int feature_enable

this is a bit combination. If you want to involve watermark, then
feature_enable |= DVR_ENC_H264_WATERMARK.

DVR_ENC_ENHANCE_H264_RATECONTROL

DVR_ENC_MJPEG_FUNCTION

DVR_ENC_H264_WATERMARK

DVR_ENC_ROI_POS

DVR_ENC_ROI_ALL

Examples:

[roi.c](#).

Definition at line **123** of file [dvr_enc_api.h](#).

unsigned int target_rate_max

for New rate control

ECBR: usually stays at target bitrate,

produce large bitrate when big motion, but never more than max
bitrate

Examples:

[roi.c](#).

Definition at line **127** of file [dvr_enc_api.h](#).

unsigned int reaction_delay_max

max frame count will the rate-control stay when away from the target bitrate, only valid at ECBR

Examples:

[roi.c.](#)

Definition at line [129](#) of file [dvr_enc_api.h](#).

int enc_type

for MJPEG

[EncodeType_tag](#)

Examples:

[roi.c.](#)

Definition at line [132](#) of file [dvr_enc_api.h](#).

int MJ_quality

Motion JPEG quality.

1(worst) ~100(best)

Examples:

[roi.c.](#)

Definition at line [134](#) of file [dvr_enc_api.h](#).

int watermark_enable

for H264 Water Mark

Indicate watermark enable or not

if 0: disable,

if 1: enable, but only insert watermark at intra-mb,

if 3: enable, and insert watermark at intra/inter-mb,

if others: not allowed

Examples:

[roi.c.](#)

Definition at line [141](#) of file [dvr_enc_api.h](#).

int watermark_interval

specify the interval, to insert watermark each "interval" frame,

valid when watermark function is enabled,

if 1: will be inserted watermark each frames,

if N: will be inserted watermark each N frames

Examples:

[roi.c.](#)

Definition at line [146](#) of file [dvr_enc_api.h](#).

int watermark_init_pattern

The initial pattern for watermark

valid when watermark function is enabled

Examples:

[roi.c.](#)

Definition at line [149](#) of file `dvr_enc_api.h`.

`int watermark_init_interval`

specify the interval, to reinit with init_pattern each "watermark"ed frames

valid when watermark function is enabled

if 1: every "watermark"ed frame with init_pattern

if N: initial with init_pattern each N "watermark"ed frame

Examples:

[roi.c](#).

Definition at line [154](#) of file `dvr_enc_api.h`.

`POS roi_pos`

for ROI x, y position

Definition at line [156](#) of file `dvr_enc_api.h`.

`ROI_ALL roi_all`

for update ROI all function, x, y position, width, height, enable/disable

Examples:

[roi.c](#).

Definition at line [158](#) of file `dvr_enc_api.h`.

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

EncParam_tag Struct Reference

encode parameter [More...](#)

Data Fields

```
int input_type
int frame_rate
int bit_rate
int ip_interval
int init_quant
int max_quant
int min_quant
int is_use_ROI
RECT ROI_win
unsigned int ext_size
void * pext_data
int reserved [6]
```

Detailed Description

encode parameter

Definition at line [164](#) of file [dvr_enc_api.h](#).

Field Documentation

int input_type

EncodeType_tag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [166](#) of file [dvr_enc_api.h](#).

int frame_rate

frame rate per second

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [168](#) of file [dvr_enc_api.h](#).

int bit_rate

bit rate per second

Examples:

[main-bitstream-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [170](#) of file [dvr_enc_api.h](#).

int ip_interval

The I-P interval frames

Examples:

main-bitstream-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **172** of file **dvr_enc_api.h**.

int init_quant

The initial quant value

Examples:

main-bitstream-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **174** of file **dvr_enc_api.h**.

int max_quant

The max/min quant value

Examples:

main-bitstream-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **176** of file **dvr_enc_api.h**.

int min_quant

Examples:

main-bitstream-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line [176](#) of file `dvr_enc_api.h`.

`int is_use_ROI`

Enable the ROI function

Examples:

`main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`, `roi.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [178](#) of file `dvr_enc_api.h`.

`RECT ROI_win`

`RECT_tag`

Examples:

`main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`, `roi.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [180](#) of file `dvr_enc_api.h`.

`unsigned int ext_size`

encode parameter extension size

Examples:

`capture_raw.c`, `main-bitstream-record.c`, `mjpeg-record.c`, `motion-detection-mpeg4.c`, `motion-detection.c`, `mpeg4-record.c`, `roi.c`, `snapshot.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [182](#) of file `dvr_enc_api.h`.

void* pext_data

point to encode parameter extension structure

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#),
[motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [184](#) of file [dvr_enc_api.h](#).

int reserved[6]

Definition at line [185](#) of file [dvr_enc_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

FuncTag_tag Struct Reference

function tag for videograph level [More...](#)

Data Fields

int	func	: 28
int	cas_ch	: 4
int	lv_ch	
int	rec_ch	
int	pb_ch	

Detailed Description

function tag for videograph level

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [capture_raw.c](#), [liveview.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [pip.c](#), [playback.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [31](#) of file [dvr_type_define.h](#).

Field Documentation

int func

functional tag

Definition at line **33** of file [dvr_type_define.h](#).

int cas_ch

cascade channel

Definition at line **34** of file [dvr_type_define.h](#).

int lv_ch

liveview channel

Definition at line **35** of file [dvr_type_define.h](#).

int rec_ch

record channel

Definition at line **36** of file [dvr_type_define.h](#).

int pb_ch

playback channel

Definition at line **37** of file [dvr_type_define.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

POS_tag Struct Reference

set position x and y [More...](#)

Data Fields

int **x**

int **y**

Detailed Description

set position x and y

Definition at line [174](#) of file [dvr_type_define.h](#).

Field Documentation

int x

x position

Definition at line [175](#) of file [dvr_type_define.h](#).

int y

y positon

Definition at line [176](#) of file [dvr_type_define.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

QueueMemConfig_tag Struct Reference

set queue memory configuration [More...](#)

Data Fields

int	size
int	count
int	ddr_num
int	limit_count

Detailed Description

set queue memory configuration

Definition at line [193](#) of file [dvr_type_define.h](#).

Field Documentation

int size

size of buffer

Definition at line [194](#) of file [dvr_type_define.h](#).

int count

the number of buffers

Definition at line [195](#) of file [dvr_type_define.h](#).

int ddr_num

the DDR number

Definition at line [196](#) of file [dvr_type_define.h](#).

int limit_count

Max count of in-used buffer.

Definition at line [197](#) of file [dvr_type_define.h](#).

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

RECT_tag Struct Reference

set size and position More...

Data Fields

int **x**
int **y**
int **width**
int **height**

Detailed Description

set size and position

Examples:

[**2ch_liveview.c**](#), [**2ch_playback.c**](#), [**liveview.c**](#), [**pip.c**](#), and [**playback.c**](#).

Definition at line [**164**](#) of file [**dvr_type_define.h**](#).

Field Documentation

int x

x position

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [main-bitstream-record.c](#), [mpeg4-record.c](#), [roi.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [165](#) of file [dvr_type_define.h](#).

int y

y positon

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [main-bitstream-record.c](#), [mpeg4-record.c](#), [roi.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [166](#) of file [dvr_type_define.h](#).

int width

width in pixel

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [main-bitstream-record.c](#), [mpeg4-record.c](#), [roi.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line **167** of file `dvr_type_define.h`.

int height

height in pixel

Examples:

`2ch_liveview.c`, `2ch_playback.c`, `liveview.c`, `main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`, `pip.c`, `playback.c`, `roi.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line **168** of file `dvr_type_define.h`.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

ReproduceBitStream_tag Struct Reference

dvr bit stream parameter, include main, sub1, sub2 bit-stream. [More...](#)

Data Fields

int	enabled
int	out_bs
int	enc_type
int	is_blocked
int	en_snapshot
DIM	dim
EncParam	enc
ScalerParam	scl
snapshot_param	snap
int	reserved [8]

Detailed Description

dvr bit stream parameter, include main, sub1, sub2 bit-stream.

Examples:

[capture_raw.c](#), [roi.c](#), [sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [206](#) of file [dvr_enc_api.h](#).

Field Documentation

`int enabled`

`DVR_ENC_EBST_ENABLE`: enabled

`DVR_ENC_EBST_DISABLE`: disabled

Examples:

`main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [209](#) of file `dvr_enc_api.h`.

`int out_bs`

0: main-bitstream

1: sub-bitstream1

2: sub-bitstream2

Examples:

`capture_raw.c`, `main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`, `roi.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [213](#) of file `dvr_enc_api.h`.

`int enc_type`

`EncodeType_tag`

Examples:

`main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`,

sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **215** of file **dvr_enc_api.h**.

int is_blocked

indicate all system-call for this channel is "blocked" or "non-block" type

Examples:

main-bitstream-record.c, mjpeg-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **217** of file **dvr_enc_api.h**.

int en_snapshot

enable/disable snapshot

DVR_ENC_EBST_ENABLE: enabled

DVR_ENC_EBST_DISABLE: disabled

Examples:

main-bitstream-record.c, mjpeg-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **220** of file **dvr_enc_api.h**.

DIM dim

DIM_tag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [roi.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [222](#) of file [dvr_enc_api.h](#).

EncParam enc

EncParam_tag

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [224](#) of file [dvr_enc_api.h](#).

ScalerParam scl

ScalerParamtag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [226](#) of file [dvr_enc_api.h](#).

snapshot_param snap

snapshot_param_tag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#),

sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line [228](#) of file [dvr_enc_api.h](#).

int reserved[8]

Definition at line [229](#) of file [dvr_enc_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

ROI_ALL_tag Struct Reference

set ROI position x, y, width, height, and enable/disable [More...](#)

Data Fields

```
int is_use_ROI  
RECT win
```

Detailed Description

set ROI position x, y, width, height, and enable/disable

Definition at line [182](#) of file [dvr_type_define.h](#).

Field Documentation

int is_use_ROI

Enable the ROI function, TRUE/FALSE

Examples:

[roi.c.](#)

Definition at line [184](#) of file [dvr_type_define.h](#).

RECT win

RECT_tag

Examples:

[roi.c.](#)

Definition at line [186](#) of file [dvr_type_define.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

ScalerParamtag Struct Reference

scalar parameter [More...](#)

Data Fields

```
int src_fmt
int dst_fmt
int scale_mode
int is_dither
int is_correction
int is_album
int des_level
int reserved [8]
```

Detailed Description

scalar parameter

Definition at line [255](#) of file [dvr_type_define.h](#).

Field Documentation

int src_fmt

source format

ScaleColorModeTag

Examples:

[2ch_playback.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [playback.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [257](#) of file [dvr_type_define.h](#).

int dst_fmt

destination format

ScaleColorModeTag

Examples:

[2ch_playback.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [playback.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [259](#) of file [dvr_type_define.h](#).

int scale_mode

ScaleMethodTag

Examples:

[2ch_playback.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [playback.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [261](#) of file `dvr_type_define.h`.

int is_dither

Not in use.

Examples:

`2ch_playback.c`, `main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`, `playback.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [262](#) of file `dvr_type_define.h`.

int is_correction

Not in use.

Examples:

`2ch_playback.c`, `main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`, `playback.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [263](#) of file `dvr_type_define.h`.

int is_album

Not in use.

Examples:

`2ch_playback.c`, `main-bitstream-record.c`, `mjpeg-record.c`, `mpeg4-record.c`, `playback.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [264](#) of file `dvr_type_define.h`.

int des_level

Not in use.

Examples:

2ch_playback.c, main-bitstream-record.c, mjpeg-record.c, mpeg4-record.c, playback.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **265** of file **dvr_type_define.h**.

int reserved[8]

Definition at line **266** of file **dvr_type_define.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

snapshot_param_tag Struct Reference

snapshot parameter More...

Data Fields

```
int sample
int RestartInterval
int u82D
int quality
int reserved [8]
```

Detailed Description

snapshot parameter

Definition at line [191](#) of file [dvr_enc_api.h](#).

Field Documentation

int sample

JpegEncInputFormatTag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [193](#) of file [dvr_enc_api.h](#).

int RestartInterval

not in use

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [195](#) of file [dvr_enc_api.h](#).

int u82D

JpegEnc420InputFormatTag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [197](#) of file [dvr_enc_api.h](#).

int quality

1(worst) ~100(best)

Examples:

main-bitstream-record.c, mjpeg-record.c, mpeg4-record.c, sub-bitstream-record.c, update-bitrate.c, and update-record-setting.c.

Definition at line **199** of file **dvr_enc_api.h**.

int reserved[8]

Definition at line **200** of file **dvr_enc_api.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields		
dvr_disp_update_disp_param_tag			Data Fields

dvr_disp_update_disp_param_tag::val_t Struct Reference

Data Fields

int	target_id
int	plane_comb
int	output_system
int	output_mode
int	display_rate
dvr_disp_color_attribute	color_attrib
dvr_disp_color_key	transparent_color [2]
dvr_disp_vbi_info	vbi_info
dvr_disp_scaler_info	scl_info
dvr_disp_resolution	res
int	reserved [8]

Detailed Description

Definition at line [127](#) of file `dvr_disp_api.h`.

Field Documentation

int target_id

Definition at line [128](#) of file [dvr_disp_api.h](#).

int plane_comb

[dvr_disp_plane_combination_tag](#)

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [130](#) of file [dvr_disp_api.h](#).

int output_system

[MCP_VIDEO_NTSC](#)

[MCP_VIDEO_PAL](#)

[MCP_VIDEO_VGA](#)

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [134](#) of file [dvr_disp_api.h](#).

int output_mode

[LCDOutputModeTag](#)

Definition at line [136](#) of file `dvr_disp_api.h`.

`int display_rate`

NTSC : 30, PAL : 25.

Examples:

`2ch_liveview.c`, `2ch_playback.c`, `liveview.c`, `pip.c`, and `playback.c`.

Definition at line [137](#) of file `dvr_disp_api.h`.

`dvr_disp_color_attribute color_attrib`

`dvr_disp_color_attribute_tag`

Definition at line [139](#) of file `dvr_disp_api.h`.

`dvr_disp_color_key transparent_color[2]`

`dvr_disp_color_key_tag`

Definition at line [141](#) of file `dvr_disp_api.h`.

`dvr_disp_vbi_info vbi_info`

`dvr_disp_vbi_info_tag`

Definition at line [143](#) of file `dvr_disp_api.h`.

`dvr_disp_scaler_info scl_info`

`dvr_disp_scaler_info_tag`

Definition at line **145** of file **dvr_disp_api.h**.

dvr_disp_resolution res

dvr_disp_resolution_tag

Definition at line **147** of file **dvr_disp_api.h**.

int reserved[8]

Definition at line **148** of file **dvr_disp_api.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields	Data Fields	

video_process_tag Struct Reference

dvr video parameter. More...

Data Fields

```
int is_3DI
int is_denoise
int denoise_mode
int reserved [4]
```

Detailed Description

dvr video parameter.

Definition at line [242](#) of file [dvr_type_define.h](#).

Field Documentation

int is_3DI

TRUE : enable 3D deinterlace , False : disable 3D deinterlace

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [244](#) of file [dvr_type_define.h](#).

int is_denoise

TRUE : enable 3D denoise , False : disable 3D denoise

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [246](#) of file [dvr_type_define.h](#).

int denoise_mode

GM3DIFrameTypeTag

Examples:

[main-bitstream-record.c](#), [mjpeg-record.c](#), [mpeg4-record.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [248](#) of file [dvr_type_define.h](#).

int reserved[4]

Definition at line **249** of file **dvr_type_define.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
doxygen 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- b -

- bg_rect0 : [dvr_disp_control_tag](#)
- bg_rect1 : [dvr_disp_control_tag](#)
- bit_rate : [dvr_enc_update_channel_param_tag](#) , [EncParam_tag](#)
- brightness : [dvr_disp_color_attribute_tag](#)
- bs : [dvr_enc_queue_get_tag](#) , [dvr_dec_queue_get_tag](#)
- bs_buf_length : [dvr_enc_copy_buf_tag](#)
- bs_length : [dvr_enc_copy_buf_tag](#)
- bs_rate : [dvr_dec_control_tag](#)
- bs_user_va : [dvr_enc_copy_buf_tag](#)

Main Page

Data Structures

Files

Examples

File List

Globals

File List

Here is a list of all files with brief descriptions:

source/dvr_common_api.h	[code]
source/dvr_dec_api.h	[code]
source/dvr_dec_ioctl.h	[code]
source/dvr_disp_api.h	[code]
source/dvr_disp_ioctl.h	[code]
source/dvr_enc_api.h	[code]
source/dvr_enc_ioctl.h	[code]
source/dvr_type_define.h	[code]

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

Defines

source/dvr_common_api.h File Reference

[Go to the source code of this file.](#)

Defines

```
#define DVR_IOC_MAGIC 'D'  
#define DVR_COMMON_APPLY _IOW(DVR_IOC_MAGIC, 1, FuncT  
#define DVR_COMMON_DEBUG _IOW(DVR_IOC_MAGIC, 2, Func  
#define DVR_COMMON_DEBUG_WITH_PANIC _IOW(DVR_IOC_M  
3, FuncTag)
```

Define Documentation

```
#define DVR_IOC_MAGIC 'D'
```

Definition at line 7 of file [dvr_common_api.h](#).

```
#define DVR_COMMON_APPLY _IOW(DVR_IOC_MAGIC, 1, FuncT
```

```
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag)
```

- get function tag from user space, and set to videograph level
- parameter :
pointer tag : argument from user space ioctl parameter, it means structure FuncTag

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [capture_raw.c](#), [liveview.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [pip.c](#), [playback.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line 16 of file [dvr_common_api.h](#).

```
#define DVR_COMMON_DEBUG _IOW(DVR_IOC_MAGIC, 2, FuncT
```

Definition at line 17 of file [dvr_common_api.h](#).

```
#define DVR_COMMON_DEBUG_WITH_PANIC _IOW(DVR_IOC_M
```

Definition at line 18 of file [dvr_common_api.h](#).

doxygen 1.7.1

[Main Page](#)[Data Structures](#)[Files](#)[Examples](#)[File List](#)[Globals](#)

[Data Structures](#) | [Typedefs](#) |
[Enumerations](#)

source/dvr_dec_api.h File Reference

[Go to the source code of this file.](#)

Data Structures

struct [DecParam_tag](#)
struct [dvr_dec_channel_param_tag](#)
struct [dvr_dec_queue_get_tag](#)
struct [dvr_dec_control_tag](#)
struct [dvr_dec_clear_param_tag](#)

Typedefs

typedef enum dvr_dec_dest_type_tag	dvr_dec_dest_type
typedef struct DecParam_tag	DecParam
typedef struct dvr_dec_channel_param_tag	dvr_dec_channel_param
typedef struct dvr_dec_queue_get_tag	dvr_dec_queue_get
typedef enum dvr_dec_ctrl_cmd_tag	dvr_dec_ctrl_cmd
typedef struct dvr_dec_control_tag	dvr_dec_control
typedef struct dvr_dec_clear_param_tag	dvr_dec_clear_param

Enumerations

```
enum dvr_dec_dest_type_tag { DEC_TYPE_TO_DISPLAY = 0,
    DEC_TYPE_TO_BUFFER, DEC_TYPE_COUNT }


---

enum dvr_dec_ctrl_cmd_tag { DEC_START, DEC_STOP,
    DEC_UPDATE }
```

TypeDef Documentation

typedef enum dvr_dec_dest_type_tag dvr_dec_dest_type

typedef struct DecParam_tag DecParam

typedef struct dvr_dec_channel_param_tag dvr_dec_channel_param

typedef struct dvr_dec_queue_get_tag dvr_dec_queue_get

typedef enum dvr_dec_ctrl_cmd_tag dvr_dec_ctrl_cmd

typedef struct dvr_dec_control_tag dvr_dec_control

typedef struct dvr_dec_clear_param_tag dvr_dec_clear_param

Enumeration Type Documentation

`enum dvr_dec_dest_type_tag`

Enumerator:

DEC_TYPE_TO_DISPLAY

DEC_TYPE_TO_BUFFER

DEC_TYPE_COUNT

Definition at line **8** of file [dvr_dec_api.h](#).

`enum dvr_dec_ctrl_cmd_tag`

Enumerator:

DEC_START

DEC_STOP

DEC_UPDATE

Definition at line **44** of file [dvr_dec_api.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

Defines

source/dvr_dec_ioctl.h File Reference

[Go to the source code of this file.](#)

Defines

```
#define DVR_DEC_IOC_MAGIC 'H'  
#define DVR_DEC_SET_CHANNEL_PARAM _IOWR(DVR_DEC_IO  
dvr_dec_channel_param)  
#define DVR_DEC_GET_CHANNEL_PARAM _IO(DVR_DEC_IOC_I  
#define DVR_DEC_QUEUE_GET _IOWR(DVR_DEC_IOC_MAGIC, !  
dvr_dec_queue_get)  
#define DVR_DEC_QUEUE_PUT _IOWR(DVR_DEC_IOC_MAGIC, 6  
dvr_dec_queue_get)  
#define DVR_DEC_CONTROL _IOW(DVR_DEC_IOC_MAGIC, 7, dv  
#define DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE _IOWR(DVR  
8, int)  
#define DVR_DEC_CLEAR_WIN _IOW(DVR_DEC_IOC_MAGIC, 18  
dvr_dec_clear_param)  
#define DVR_DEC_CLEAR_WIN2 _IOW(DVR_DEC_IOC_MAGIC, 1  
dvr_dec_clear_param)
```

Define Documentation

```
#define DVR_DEC_IOC_MAGIC 'H'
```

Definition at line [2](#) of file [dvr_dec_ioctl.h](#).

```
#define DVR_DEC_SET_CHANNEL_PARAM _IOWR(DVR_DEC_IO
```

```
ioctl(dec_fd, DVR_DEC_SET_CHANNEL_PARAM, &ch_param)
```

- explanation : get channel parameter from user space ,and set parameter to device driver
- parameter :
pointer ch_param : argument from user space ioctl parameter, it means structure dvr_dec_channel_param

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line [12](#) of file [dvr_dec_ioctl.h](#).

```
#define DVR_DEC_GET_CHANNEL_PARAM _IO(DVR_DEC_IOC_M
```

```
ioctl(dec_fd, DVR_DEC_GET_CHANNEL_PARAM, &ch_param)
```

- explanation : get channel parameter from user space
- parameter :
pointer ch_param : argument from user space ioctl parameter, it means structure dvr_dec_channel_param

Definition at line [21](#) of file [dvr_dec_ioctl.h](#).

```
#define DVR_DEC_QUEUE_GET _IOWR(DVR_DEC_IOC_MAGIC, !
```

ioctl(dec_fd, DVR_DEC_QUEUE_GET, &data)

- explanation : Get buffer to user space. It includes the buffer length, offset.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_dec_queue_get

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line [30](#) of file [dvr_dec_ioctl.h](#).

#define DVR_DEC_QUEUE_PUT _IOWR(DVR_DEC_IOC_MAGIC, 6)

ioctl(dec_fd, DVR_DEC_QUEUE_PUT, &data)

- explanation : get buffer from user space, and release buffer at videograph layer
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_dec_queue_get

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line [39](#) of file [dvr_dec_ioctl.h](#).

#define DVR_DEC_CONTROL _IOW(DVR_DEC_IOC_MAGIC, 7, dvctrl)

ioctl(dec_fd, DVR_DEC_CONTROL, &dec_ctrl)

- explanation : get decode control command from user space, and set command to videograph layer
- parameter :
pointer dec_ctrl : argument from user space ioctl parameter, it

means structure `dvr_dec_control`

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **48** of file [dvr_dec_ioctl.h](#).

```
#define DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE _IOWR(DVR_
```

```
ioctl(dec_fd, DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE,  
&dec_buf_size)
```

- explanation : get output buffer size for dvr decode to user space.
- parameter :
pointer dec_buf_size : argument from user space ioctl parameter, it means request buffer size.

Examples:

[2ch_playback.c](#), and [playback.c](#).

Definition at line **57** of file [dvr_dec_ioctl.h](#).

```
#define DVR_DEC_CLEAR_WIN _IOW(DVR_DEC_IOC_MAGIC, 18
```

```
ioctl(dec_fd, DVR_DEC_CLEAR_WIN, &dec_clear_param)
```

- explanation : to prevent lcd unexpected image, when stop playback, then start another playback,
use flow : `ioctl(dvr_fd,DVR_COMMON_APPLY,&data); //stop current playback`
.....
`ioctl(dec_fd,DVR_DEC_CLEAR_WIN,&data); //must after stop, and before start apply`
`ioctl(dvr_fd,DVR_COMMON_APPLY,&data); //start new playback`
- parameter :
pointer dec_clear_param : argument from user space ioctl parameter, it means structure `dvr_dec_clear_param`.

Definition at line 71 of file [dvr_dec_ioctl.h](#).

```
#define DVR_DEC_CLEAR_WIN2 _IOW(DVR_DEC_IOC_MAGIC, 1)
```

```
ioctl(dec_fd, DVR_DEC_CLEAR_WIN2, &dec_clear_param)
```

- explanation : to prevent lcd unexpected image, when update current playback
use flow : dec_ctrl.command = DEC_UPDATE;
ioctl(dec_fd, DVR_DEC_CONTROL, &dec_ctrl);
ioctl(dec_fd, DVR_DEC_CLEAR_WIN2, &data); // tell VG to clear buffer after apply
ioctl(dvr_fd, DVR_COMMON_APPLY, &data);
- parameter :
pointer dec_clear_param : argument from user space ioctl parameter, it means structure dvr_dec_clear_param.

Definition at line 85 of file [dvr_dec_ioctl.h](#).

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

Data Structures | Defines |
Typedefs | Enumerations

source/dvr_disp_api.h File Reference

[Go to the source code of this file.](#)

Data Structures

```
struct dvr_disp_color_attribute_tag  
struct dvr_disp_color_key_tag  
struct dvr_disp_vbi_info_tag  
struct dvr_disp_scaler_info_tag  
struct dvr_disp_resolution_tag  
struct dvr_disp_disp_param_tag  
struct dvr_disp_update_disp_param_tag  
struct dvr_disp_update_disp_param_tag::val_t  
struct dvr_disp_plane_param_tag  
struct dvr_disp_plane_param_st_tag  
struct dvr_disp_update_plane_param_tag  
struct DispParam_Ext1_tag  
struct dvr_disp_control_tag  
struct dvr_disp_clear_param_tag
```

Defines

```
#define DVR_PLANE_ID(d, p) ((d<<8)|p)
#define GET_DISP_NUM_FROM_PLANE_ID(h) (h>>8)
#define GET_PLANE_NUM_FROM_PLANE_ID(h) (0xFF&h)
#define DVR_PARAM_MAGIC 0x1688
#define DVR_PARAM_MAGIC_SHIFT 16
#define CAP_BUF_ID(id) ((DVR_PARAM_MAGIC << DVR_PARAM_
#define DVR_DISP_MAGIC_ADD_VAL(val) ((DVR_PARAM_MAGIC
#define (val))
#define DVR_DISP_CHECK_MAGIC(val) (((val)>>DVR_PARAM_MA
#define DVR_DISP_GET_VALUE(val) ((val)&((1<<DVR_PARAM_MA
```

Typedefs

typedef struct dvr_disp_color_attribute_tag	dvr_disp_color_attribute
typedef struct dvr_disp_color_key_tag	dvr_disp_color_key
typedef enum dvr_disp_plane_combination_tag	dvr_disp_plane_combination
typedef struct dvr_disp_vbi_info_tag	dvr_disp_vbi_info
typedef struct dvr_disp_scaler_info_tag	dvr_disp_scaler_info
typedef struct dvr_disp_resolution_tag	dvr_disp_resolution
typedef struct dvr_disp_disp_param_tag	dvr_disp_disp_param
typedef enum dvr_disp_disp_param_name_tag	dvr_disp_disp_param_name
typedef struct dvr_disp_update_disp_param_tag	dvr_disp_update_disp_param
typedef struct dvr_disp_plane_param_tag	dvr_disp_plane_param_st
typedef struct dvr_disp_plane_param_st_tag	dvr_disp_plane_param
typedef enum dvr_disp_plane_param_name_tag	dvr_disp_plane_param_name
typedef struct dvr_disp_update_plane_param_tag	dvr_disp_update_plane_param
typedef enum dvr_disp_ctrl_cmd_tag	dvr_disp_ctrl_cmd
typedef enum dvr_disp_channel_type_tag	dvr_disp_channel_type
typedef struct DispParam_Ext1_tag	DispParam_Ext1
typedef struct dvr_disp_control_tag	dvr_disp_control
typedef struct dvr_disp_clear_param_tag	dvr_disp_clear_param

Enumerations

enum	dvr_disp_type { DISP_TYPE_LIVEVIEW, DISP_TYPE_CASCADE, DISP_TYPE_PLAYBACK, DISP_TYPE_ON_BUFFER }
enum	dvr_disp_plane_combination_tag { BG_ONLY = 0, BG_AND_1PLANE = 1, BG_AND_2PLANE = 2 }
	dvr_disp_disp_param_name_tag { DISP_PARAM_TARGET, DISP_PARAM_PLANE_COMBINATION, DISP_PARAM_OUTPUT_MODE,
enum	DISP_PARAM_OUTPUT_SYSTEM, DISP_PARAM_COLOR_ATTRIBUTE, DISP_PARAM_TRANSPARENT_COLOR, DISP_PARAM_RESOLUTION, DISP_PARAM_APPLY }
enum	dvr_disp_plane_param_name_tag { PLANE_PARAM_COLOR_MODE, PLANE_PARAM_WINDOW, PLANE_PARAM_DATA_MODE, PLANE_PARAM_APPLY }
enum	dvr_disp_ctrl_cmd_tag { DISP_START, DISP_STOP, DISP_UPDATE, DISP_RUN }
enum	dvr_disp_channel_type_tag { DISP_NORMAL_CHN, DISP_LAYER0_CHN, DISP_LAYER1_CHN }

Define Documentation

```
#define DVR_PLANE_ID( d,  
                      p  
                    ) ((d<<8)|p)
```

Definition at line [9](#) of file [dvr_disp_api.h](#).

```
#define GET_DISP_NUM_FROM_PLANE_ID( h ) (h>>8)
```

Definition at line [10](#) of file [dvr_disp_api.h](#).

```
#define GET_PLANE_NUM_FROM_PLANE_ID( h ) (0xFF&h)
```

Definition at line [11](#) of file [dvr_disp_api.h](#).

```
#define DVR_PARAM_MAGIC 0x1688
```

Definition at line [221](#) of file [dvr_disp_api.h](#).

```
#define DVR_PARAM_MAGIC_SHIFT 16
```

Definition at line [222](#) of file [dvr_disp_api.h](#).

```
#define CAP_BUF_ID( id ) ((DVR_PARAM_MAGIC << DVR_PAR
```

Definition at line [223](#) of file [dvr_disp_api.h](#).

```
#define DVR_DISP_MAGIC_ADD_VAL( val ) ((DVR_PARAM_MA
```

Examples:

pip.c.

Definition at line [225](#) of file [dvr_disp_api.h](#).

```
#define DVR_DISP_CHECK_MAGIC( val ) (((val)>>DVR_PARAM
```

Definition at line [226](#) of file [dvr_disp_api.h](#).

```
#define DVR_DISP_GET_VALUE( val ) ((val)&((1<<DVR_PARAM
```

Definition at line [227](#) of file [dvr_disp_api.h](#).

Typedef Documentation

typedef struct dvr_disp_color_attribute_tag dvr_disp_color_attribu

typedef struct dvr_disp_color_key_tag dvr_disp_color_key

typedef enum dvr_disp_plane_combination_tag dvr_disp_plane_c

typedef struct dvr_disp_vbi_info_tag dvr_disp_vbi_info

typedef struct dvr_disp_scaler_info_tag dvr_disp_scaler_info

typedef struct dvr_disp_resolution_tag dvr_disp_resolution

typedef struct dvr_disp_disp_param_tag dvr_disp_disp_param

typedef enum dvr_disp_disp_param_name_tag dvr_disp_disp_par

typedef struct dvr_disp_update_disp_param_tag dvr_disp_update

typedef struct dvr_disp_plane_param_tag dvr_disp_plane_param

typedef struct dvr_disp_plane_param_st_tag dvr_disp_plane_para

typedef enum dvr_disp_plane_param_name_tag dvr_disp_plane_p

typedef struct dvr_disp_update_plane_param_tag dvr_disp_update

typedef enum dvr_disp_ctrl_cmd_tag dvr_disp_ctrl_cmd

typedef enum dvr_disp_channel_type_tag dvr_disp_channel_type

typedef struct DispParam_Ext1_tag DispParam_Ext1

typedef struct dvr_disp_control_tag dvr_disp_control

typedef struct dvr_disp_clear_param_tag dvr_disp_clear_param

Enumeration Type Documentation

enum dvr_disp_type

Enumerator:

DISP_TYPE_LIVEVIEW
DISP_TYPE CASCADE
DISP_TYPE_PLAYBACK
DISP_TYPE_ON_BUFFER

Definition at line 14 of file [dvr_disp_api.h](#).

enum dvr_disp_plane_combination_tag

Enumerator:

BG_ONLY only backgraound

BG_AND_1PLANE background and another plane

BG_AND_2PLANE background and another 2 planes

Definition at line 51 of file [dvr_disp_api.h](#).

enum dvr_disp_disp_param_name_tag

Enumerator:

DISP_PARAM_TARGET
DISP_PARAM_PLANE_COMBINATION
DISP_PARAM_OUTPUT_MODE
DISP_PARAM_OUTPUT_SYSTEM

DISP_PARAM_COLOR_ATTRIBUTE
DISP_PARAM_TRANSPARENT_COLOR
DISP_PARAM_RESOLUTION
DISP_PARAM_APPLY

Definition at line [110](#) of file [dvr_disp_api.h](#).

enum dvr_disp_plane_param_name_tag

Enumerator:

PLANE_PARAM_COLOR_MODE
PLANE_PARAM_WINDOW
PLANE_PARAM_DATA_MODE
PLANE_PARAM_APPLY

Definition at line [177](#) of file [dvr_disp_api.h](#).

enum dvr_disp_ctrl_cmd_tag

Enumerator:

DISP_START
DISP_STOP
DISP_UPDATE
DISP_RUN

Definition at line [203](#) of file [dvr_disp_api.h](#).

enum dvr_disp_channel_type_tag

Enumerator:

DISP_NORMAL_CHN
DISP_LAYER0_CHN
DISP_LAYER1_CHN

Definition at line **210** of file **dvr_disp_api.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
doxygen 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

Defines

source/dvr_disp_ioctl.h File Reference

[Go to the source code of this file.](#)

Defines

```
#define DVR_DISP_IOC_MAGIC 'I'  
#define DVR_DISP_INITIATE _IO(DVR_DISP_IOC_MAGIC, 1)  
#define DVR_DISP_TERMINATE _IO(DVR_DISP_IOC_MAGIC, 2)  
#define DVR_DISP_GET_DISP_PARAM _IOWR(DVR_DISP_IOC_M  
dvr_disp_disp_param)  
#define DVR_DISP_SET_DISP_PARAM _IOWR(DVR_DISP_IOC_M  
dvr_disp_disp_param)  
#define DVR_DISP_UPDATE_DISP_PARAM _IOWR(DVR_DISP_IO  
6, dvr_disp_update_disp_param)  
#define DVR_DISP_GET_PLANE_PARAM _IOWR(DVR_DISP_IOC  
dvr_disp_plane_param)  
#define DVR_DISP_SET_PLANE_PARAM _IOWR(DVR_DISP_IOC  
dvr_disp_plane_param)  
#define DVR_DISP_UPDATE_PLANE_PARAM _IOWR(DVR_DISP_9,  
dvr_disp_update_plane_param)  
#define DVR_DISP_CONTROL _IOR(DVR_DISP_IOC_MAGIC, 10,  
dvr_disp_control)  
#define DVR_DISP_CLEAR_WIN _IOWR(DVR_DISP_IOC_MAGIC,  
dvr_disp_clear_param)
```

Define Documentation

```
#define DVR_DISP_IOC_MAGIC 'I'
```

Definition at line [2](#) of file [dvr_disp_ioctl.h](#).

```
#define DVR_DISP_INITIATE _IO(DVR_DISP_IOC_MAGIC, 1)
```

```
ioctl(disp_fd, DVR_DISP_INITIATE, 0)
```

- explanation : not used

Definition at line [10](#) of file [dvr_disp_ioctl.h](#).

```
#define DVR_DISP_TERMINATE _IO(DVR_DISP_IOC_MAGIC, 2)
```

```
ioctl(disp_fd, DVR_DISP_TERMINATE, 0)
```

- explanation : not used

Definition at line [18](#) of file [dvr_disp_ioctl.h](#).

```
#define DVR_DISP_GET_DISP_PARAM _IOWR(DVR_DISP_IOC_M
```

```
ioctl(disp_fd, DVR_DISP_GET_DISP_PARAM, &disp_param)
```

- explanation : get LCD color parameter from user space ,and set parameter to device driver
- parameter :
pointer disp_param : argument from user space ioctl parameter, it means structure dvr_disp_disp_param

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and

[playback.c](#)

Definition at line [28](#) of file [dvr_disp_ioctl.h](#).

```
#define DVR_DISP_SET_DISP_PARAM _IOWR(DVR_DISP_IOC_M
```

```
ioctl(disp_fd, DVR_DISP_SET_DISP_PARAM, &disp_param)
```

- explanation : get LCD color parameter from user space ,and set parameter to device driver
- parameter :
pointer disp_param : argument from user space ioctl parameter, it means structure dvr_disp_disp_param

Examples:

[playback.c](#)

Definition at line [38](#) of file [dvr_disp_ioctl.h](#).

```
#define DVR_DISP_UPDATE_DISP_PARAM _IOWR(DVR_DISP_IO
```

```
ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM,  
&disp_update_param)
```

- explanation : get LCD color parameter from user space ,and update parameter to device driver
- parameter :
pointer disp_update_param : argument from user space ioctl parameter, it means structure dvr_disp_update_disp_param

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line [48](#) of file [dvr_disp_ioctl.h](#).

```
#define DVR_DISP_GET_PLANE_PARAM _IOWR(DVR_DISP_IOC_
```

```
ioctl(disp_fd, DVR_DISP_GET_PLANE_PARAM, &plane_param)
```

- explanation : get LCD plane(window) parameter from user space, and set parameter to device driver
- parameter :
pointer plane_param : argument from user space ioctl parameter, it means structure dvr_disp_plane_param

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line **58** of file [dvr_disp_ioctl.h](#).

```
#define DVR_DISP_SET_PLANE_PARAM _IOWR(DVR_DISP_IOC_
```

```
ioctl(disp_fd, DVR_DISP_SET_PLANE_PARAM,  
&plane_param_set)
```

- explanation : get LCD plane(window) parameter from user space, and set parameter to device driver
- parameter :
pointer plane_param_set : argument from user space ioctl parameter, it means structure dvr_disp_plane_param

Examples:

[playback.c](#).

Definition at line **68** of file [dvr_disp_ioctl.h](#).

```
#define DVR_DISP_UPDATE_PLANE_PARAM _IOWR(DVR_DISP_
```

```
ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PARAM,  
&plane_param_update)
```

- explanation : get LCD plane(window) parameter from user space, and update parameter to device driver
- parameter :

pointer plane_param_update : argument from user space ioctl parameter, it means structure dvr_disp_update_plane_param

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [liveview.c](#), [pip.c](#), and [playback.c](#).

Definition at line **78** of file [dvr_disp_ioctl.h](#).

#define DVR_DISP_CONTROL _IOR(DVR_DISP_IOC_MAGIC, 10, c

ioctl(disp_fd, DVR_DISP_CONTROL, &disp_ctrl)

- explanation : get display control command from user space, and set command to videograph layer
- parameter :

pointer disp_ctrl : argument from user space ioctl parameter, it means structure dvr_disp_control

Examples:

[2ch_liveview.c](#), [liveview.c](#), and [pip.c](#).

Definition at line **88** of file [dvr_disp_ioctl.h](#).

#define DVR_DISP_CLEAR_WIN _IOWR(DVR_DISP_IOC_MAGIC,

ioctl(disp_fd, DVR_DISP_INITIATE, &disp_clear_param)

- explanation : get channel parameter from user space ,and clear lcd buffer
- parameter :

pointer disp_clear_param; : argument from user space ioctl parameter, it means structure dvr_disp_clear_param

Examples:

2ch_liveview.c.

Definition at line **98** of file **dvr_disp_ioctl.h**.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

Data Structures | Defines |
Typedefs | Enumerations

source/dvr_enc_api.h File Reference

[Go to the source code of this file.](#)

Data Structures

struct	EncParam_Ext1_tag
	no longer used, replace by EncParam_Ext4 More...
struct	EncParam_Ext2_tag
	no longer used, replace by EncParam_Ext4 More...
struct	EncParam_Ext3_tag
	no longer used, replace by EncParam_Ext4 More...
struct	EncParam_Ext4_tag
	encode parameter extension More...
struct	EncParam_Ext5_tag
struct	EncParam_tag
	encode parameter More...
struct	snapshot_param_tag
	snapshot parameter More...
struct	ReproduceBitStream_tag
	dvr bit stream parameter, include main, sub1, sub2 bit-stream. More...
struct	dvr_enc_src_tag
	dvr encode source parameter More...
struct	dvr_enc_channel_param_tag
	dvr encode channel parameter. More...
struct	dvr_enc_queue_get_tag
	get dvr encode buffer More...
struct	dvr_enc_update_channel_param_tag
	update dvr encode channel parameter More...
struct	dvr_enc_control_tag
	dvr encode control parameter More...
struct	dvr_enc_copy_buf_tag

Defines

```
#define POLLIN_MAIN_BS 0x0001
#define POLLIN_SUB1_BS (POLLIN_MAIN_BS << 1)
#define POLLIN_SUB2_BS (POLLIN_MAIN_BS << 2)
#define POLLIN_SUB3_BS (POLLIN_MAIN_BS << 3)
#define POLLIN_SUB4_BS (POLLIN_MAIN_BS << 4)
#define POLLIN_SUB5_BS (POLLIN_MAIN_BS << 5)
#define POLLIN_SUB6_BS (POLLIN_MAIN_BS << 6)
#define POLLIN_SUB7_BS (POLLIN_MAIN_BS << 7)
#define POLLIN_SNAP_BS (POLLIN_MAIN_BS << 8)
#define POLLIN_SNAP_BS (1 << DVR_ENC_REPD_BT_NUM)
#define DVR_ENC_MAGIC 0x1689
#define DVR_ENC_MAGIC_SHIFT 16
#define DVR_ENC_MAGIC_ADD_VAL(val) ((DVR_ENC_MAGIC << (val)))
#define DVR_ENC_CHECK_MAGIC(v) (((v)>>DVR_ENC_MAGIC_S))
#define DVR_ENC_GET_VALUE(v) ((v)&((1<<DVR_ENC_MAGIC_S)
#define DVR_ENC_ENHANCE_H264_RATECONTROL 1
#define DVR_ENC_MJPEG_FUNCTION (1 << 1)
#define DVR_ENC_H264_WATERMARK (1 << 2)
#define DVR_ENC_ROI_POS (1 << 3)
#define DVR_ENC_ROI_ALL (1 << 4)
#define DVR_ENC_EBST_ENABLE 0x55887799
#define DVR_ENC_EBST_DISABLE 0x0
```

Typedefs

typedef enum dvr_enc_src_type_tag	dvr_enc_src_type
typedef struct EncParam_Ext1_tag	EncParam_Ext1
typedef struct EncParam_Ext2_tag	EncParam_Ext2
typedef struct EncParam_Ext3_tag	EncParam_Ext3
typedef struct EncParam_Ext4_tag	EncParam_Ext4
typedef struct EncParam_Ext5_tag	EncParam_Ext5
typedef struct EncParam_tag	EncParam
typedef struct snapshot_param_tag	snapshot_param
typedef struct ReproduceBitStream_tag	ReproduceBitStream
typedef struct dvr_enc_src_tag	dvr_enc_src_param
typedef struct dvr_enc_channel_param_tag	dvr_enc_channel_param
typedef enum dvr_enc_channel_param_name_tag	dvr_enc_channel_param_name
typedef struct dvr_enc_queue_get_tag	dvr_enc_queue_get
typedef struct dvr_enc_update_channel_param_tag	dvr_enc_update_channel_param
typedef struct dvr_enc_control_tag	dvr_enc_control
typedef struct dvr_enc_copy_buf_tag	dvr_enc_copy_buf

Enumerations

```
enum dvr_enc_src_type_tag { ENC_TYPE_FROM_CAPTURE = 0,
                           ENC_TYPE_FROM CASCADE,
                           ENC_TYPE_FROM_BUFFER, ENC_SRC_TYPE_COUNT }

enum dvr_enc_channel_param_name_tag {
                           ENC_PARAM_SRC_DIM, ENC_PARAM_DST_WIN }

dvr_enc_ctrl_cmd {
                           ENC_START, ENC_STOP, ENC_SNAP, ENC_UPDATE,
                           ENC_RAW
}
```

Define Documentation

```
#define POLLIN_MAIN_BS 0x0001
```

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [365](#) of file [dvr_enc_api.h](#).

```
#define POLLIN_SUB1_BS (POLLIN_MAIN_BS << 1)
```

Examples:

[capture_raw.c](#), [roi.c](#), [sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [366](#) of file [dvr_enc_api.h](#).

```
#define POLLIN_SUB2_BS (POLLIN_MAIN_BS << 2)
```

Examples:

[sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [367](#) of file [dvr_enc_api.h](#).

```
#define POLLIN_SUB3_BS (POLLIN_MAIN_BS << 3)
```

Definition at line [368](#) of file [dvr_enc_api.h](#).

```
#define POLLIN_SUB4_BS (POLLIN_MAIN_BS << 4)
```

Definition at line [369](#) of file `dvr_enc_api.h`.

```
#define POLLIN_SUB5_BS (POLLIN_MAIN_BS << 5)
```

Definition at line [370](#) of file `dvr_enc_api.h`.

```
#define POLLIN_SUB6_BS (POLLIN_MAIN_BS << 6)
```

Definition at line [371](#) of file `dvr_enc_api.h`.

```
#define POLLIN_SUB7_BS (POLLIN_MAIN_BS << 7)
```

Definition at line [372](#) of file `dvr_enc_api.h`.

```
#define POLLIN_SUB8_BS (POLLIN_MAIN_BS << 8)
```

Definition at line [373](#) of file `dvr_enc_api.h`.

```
#define POLLIN_SNAP_BS (1 << DVR_ENC_REPD_BT_NUM)
```

Examples:

[motion-detection-mpeg4.c](#), [motion-detection.c](#), and [snapshot.c](#).

Definition at line [374](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_MAGIC 0x1689
```

Definition at line [376](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_MAGIC_SHIFT 16
```

Definition at line [377](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_MAGIC_ADD_VAL( val ) ((DVR_ENC_MAGIC
```

Examples:

`capture_raw.c`, `main-bitstream-record.c`, `mjpeg-record.c`,
`motion-detection-mpeg4.c`, `motion-detection.c`, `mpeg4-record.c`, `roi.c`, `snapshot.c`, `sub-bitstream-record.c`, `update-bitrate.c`, and `update-record-setting.c`.

Definition at line [378](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_CHECK_MAGIC( v ) (((v)>>DVR_ENC_MAGI
```

Definition at line [380](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_GET_VALUE( v ) ((v)&((1<<DVR_ENC_MAGI
```

Definition at line [381](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_ENHANCE_H264_RATECONTROL 1
```

Definition at line [382](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_MJPEG_FUNCTION (1 << 1)
```

Definition at line [383](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_H264_WATERMARK (1 << 2)
```

Definition at line [384](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_ROI_POS (1 << 3)
```

Definition at line [385](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_ROI_ALL (1 << 4)
```

Definition at line [386](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_EBST_ENABLE 0x55887799
```

Examples:

[sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [389](#) of file `dvr_enc_api.h`.

```
#define DVR_ENC_EBST_DISABLE 0x0
```

Definition at line [390](#) of file `dvr_enc_api.h`.

Typedef Documentation

typedef enum dvr_enc_src_type_tag dvr_enc_src_type

typedef struct EncParam_Ext1_tag EncParam_Ext1

no longer used, replace by EncParam_Ext4

typedef struct EncParam_Ext2_tag EncParam_Ext2

no longer used, replace by EncParam_Ext4

typedef struct EncParam_Ext3_tag EncParam_Ext3

no longer used, replace by EncParam_Ext4

typedef struct EncParam_Ext4_tag EncParam_Ext4

encode parameter extension

typedef struct EncParam_Ext5_tag EncParam_Ext5

typedef struct EncParam_tag EncParam

encode parameter

typedef struct snapshot_param_tag snapshot_param

snapshot parameter

typedef struct ReproduceBitStream_tag ReproduceBitStream

dvr bit stream parameter, include main, sub1, sub2 bit-stream.

typedef struct dvr_enc_src_tag dvr_enc_src_param

dvr encode source parameter

typedef struct dvr_enc_channel_param_tag dvr_enc_channel_param

dvr encode channel parameter.

typedef enum dvr_enc_channel_param_name_tag dvr_enc_channel_param_name

typedef struct dvr_enc_queue_get_tag dvr_enc_queue_get

get dvr encode buffer

typedef struct dvr_enc_update_channel_param_tag dvr_enc_update_channel_param

update dvr encode channel parameter

typedef struct dvr_enc_control_tag dvr_enc_control

dvr encode control parameter

typedef struct dvr_enc_copy_buf_tag dvr_enc_copy_buf

Enumeration Type Documentation

enum dvr_enc_src_type_tag

Enumerator:

ENC_TYPE_FROM_CAPTURE
ENC_TYPE_FROM CASCADE
ENC_TYPE_FROM_BUFFER
ENC_SRC_TYPE_COUNT

Definition at line 8 of file [dvr_enc_api.h](#).

enum dvr_enc_channel_param_name_tag

Enumerator:

ENC_PARAM_SRC_DIM
ENC_PARAM_DST_WIN

Definition at line 269 of file [dvr_enc_api.h](#).

enum dvr_enc_ctrl_cmd

Enumerator:

ENC_START
ENC_STOP
ENC_SNAP
ENC_UPDATE
ENC_RAW

Definition at line 287 of file [dvr_enc_api.h](#).

doxygen 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

Defines

source/dvr_enc_ioctl.h File Reference

[Go to the source code of this file.](#)

Defines

```
#define DVR_ENC_IOC_MAGIC 'B'  
#define DVR_ENC_SET_CHANNEL_PARAM _IOWR(DVR_ENC_IOC_MAGIC, 0, dvr_enc_channel_param)  
#define DVR_ENC_GET_CHANNEL_PARAM _IOWR(DVR_ENC_IOC_MAGIC, 1, dvr_enc_channel_param)  
#define DVR_ENC_QUEUE_GET _IOWR(DVR_ENC_IOC_MAGIC, 2, dvr_enc_queue_get)  
#define DVR_ENC_QUEUE_PUT _IOWR(DVR_ENC_IOC_MAGIC, 3, dvr_enc_queue_put)  
#define DVR_ENC_CONTROL _IOWR(DVR_ENC_IOC_MAGIC, 7, dvr_enc_control)  
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE _IOWR(DVR_ENC_IOC_MAGIC, 8, dvr_enc_query_output_buffer_size)  
#define DVR_ENC_QUEUE_GET_SNAP _IOWR(DVR_ENC_IOC_MAGIC, 9, dvr_enc_queue_get_snap)  
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP_OFFSET _IOWR(DVR_ENC_IOC_MAGIC, 10, dvr_enc_query_output_buffer_snap_offset)  
#define DVR_ENC_QUEUE_GET_SUB1_BS _IOWR(DVR_ENC_IOC_MAGIC, 11, dvr_enc_queue_get_sub1_bs)  
#define DVR_ENC_QUEUE_GET_SUB2_BS _IOWR(DVR_ENC_IOC_MAGIC, 12, dvr_enc_queue_get_sub2_bs)  
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1_BS_OFFSET _IOWR(DVR_ENC_IOC_MAGIC, 13, dvr_enc_query_output_buffer_sub1_bs_offset)  
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB2_BS_OFFSET _IOWR(DVR_ENC_IOC_MAGIC, 14, dvr_enc_query_output_buffer_sub2_bs_offset)  
#define DVR_ENC_RESET_INTRA _IOR(DVR_ENC_IOC_MAGIC, 15, dvr_enc_reset_intra)  
#define DVR_ENC_SET_SUB_BS_PARAM _IOWR(DVR_ENC_IOC_MAGIC, 16, dvr_enc_set_sub_bs_param)  
#define DVR_ENC_SWAP_INTRA _IOR(DVR_ENC_IOC_MAGIC, 17, dvr_enc_swap_intra)  
#define DVR_ENC_SUB_PATH_DENOISE_CTRL _IOWR(DVR_ENC_IOC_MAGIC, 18, dvr_enc_sub_path_denoise_ctrl)  
#define DVR_ENC_QUEUE_GET_SUB3_BS _IOWR(DVR_ENC_IOC_MAGIC, 19, dvr_enc_queue_get_sub3_bs)  
#define DVR_ENC_QUEUE_GET_SUB4_BS _IOWR(DVR_ENC_IOC_MAGIC, 20, dvr_enc_queue_get_sub4_bs)  
#define DVR_ENC_QUEUE_GET_SUB5_BS _IOWR(DVR_ENC_IOC_MAGIC, 21, dvr_enc_queue_get_sub5_bs)  
#define DVR_ENC_QUEUE_GET_SUB6_BS _IOWR(DVR_ENC_IOC_MAGIC, 22, dvr_enc_queue_get_sub6_bs)  
#define DVR_ENC_QUEUE_GET_SUB7_BS _IOWR(DVR_ENC_IOC_MAGIC, 23, dvr_enc_queue_get_sub7_bs)
```

```
dvr_enc_queue_get)
#define DVR_ENC_QUEUE_GET_SUB8_BS _IOWR(DVR_ENC_IOC_M,dvr_enc_queue_get)
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB3_BS_OFFSET
26, int)
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB4_BS_OFFSET
27, int)
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB5_BS_OFFSET
28, int)
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB6_BS_OFFSET
29, int)
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB7_BS_OFFSET
30, int)
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB8_BS_OFFSET
31, int)
#define DVR_ENC_QUEUE_GET_COPY _IOWR(DVR_ENC_IOC_M,DVR_ENC_QUEUE_GET_SUB1_BS_COPY _IOWR(DVR_E
dvr_enc_copy_buf)
#define DVR_ENC_QUEUE_GET_SUB2_BS_COPY _IOWR(DVR_E
dvr_enc_copy_buf)
#define DVR_ENC_QUEUE_GET_SUB3_BS_COPY _IOWR(DVR_E
dvr_enc_copy_buf)
#define DVR_ENC_QUEUE_GET_SUB4_BS_COPY _IOWR(DVR_E
dvr_enc_copy_buf)
#define DVR_ENC_QUEUE_GET_SUB5_BS_COPY _IOWR(DVR_E
dvr_enc_copy_buf)
#define DVR_ENC_QUEUE_GET_SUB6_BS_COPY _IOWR(DVR_E
dvr_enc_copy_buf)
#define DVR_ENC_QUEUE_GET_SUB7_BS_COPY _IOWR(DVR_E
dvr_enc_copy_buf)
#define DVR_ENC_QUEUE_GET_SUB8_BS_COPY _IOWR(DVR_E
dvr_enc_copy_buf)
```

Define Documentation

```
#define DVR_ENC_IOC_MAGIC 'B'
```

Definition at line 28 of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_SET_CHANNEL_PARAM _IOWR(DVR_ENC_IO
```

```
ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_param)
```

- explanation : get channel parameter from user space ,and set parameter to device driver
- parameter :
pointer ch_param : argument from user space ioctl parameter, it means structure dvr_enc_channel_param

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line 38 of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_GET_CHANNEL_PARAM _IOWR(DVR_ENC_IO
```

```
ioctl(enc_fd, DVR_ENC_GET_CHANNEL_PARAM, &ch_param) :
```

- explanation : get channel parameter from user space
- parameter :
pointer ch_param : argument from user space ioctl parameter, it means structure dvr_enc_channel_param

Definition at line 47 of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET _IOWR(DVR_ENC_IOC_MAGIC, 5)
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET, &data)
```

- explanation : Get buffer to user space. It includes the buffer length, offset.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [56](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_PUT _IOWR(DVR_ENC_IOC_MAGIC, 6)
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data)
```

- explanation : get buffer from user space, and release buffer at videograph layer
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [65](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_CONTROL _IOW(DVR_ENC_IOC_MAGIC, 7, dv)
```

ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl)

- explanation : get encode control command from user space, and set command to videograph layer
- parameter :
pointer enc_ctrl : argument from user space ioctl parameter, it means structure dvr_enc_control

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [74](#) of file [dvr_enc_ioctl.h](#).

#define DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE _IOWR(DVR_

ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE, &enc_buf_size)

- explanation : get output buffer size for dvr encode to user space.
buffer size = main bitstream + sub1 bitstream + sub2 bitstream + snapshot
- parameter :
pointer enc_buf_size : argument from user space ioctl parameter, it means request buffer size.

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [84](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SNAP _IOWR(DVR_ENC_IOC_M
```

```
ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE,  
&enc_buf_size);
```

- explanation : get snapshot buffer to user space
- parameter :
pointer queue_data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Examples:

[motion-detection-mpeg4.c](#), [motion-detection.c](#), and
[snapshot.c](#).

Definition at line [93](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP_OFFSET _I
```

```
ioctl(enc_fd,  
DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP_OFFSET,  
&bs_buf_snap_offset)
```

- explanation : get output buffer offset for snapshot to user space
- parameter :
pointer bs_buf_snap_offset : argument from user space ioctl parameter, it means snapshot offset.

Examples:

[motion-detection-mpeg4.c](#), [motion-detection.c](#), [snapshot.c](#),
[sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [103](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB1_BS _IOWR(DVR_ENC_IOC
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB1_BS, &data) :
```

- explanation : get sub1-bitstream buffer to user space.
- parameter :
 - pointer data** : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Examples:

[capture_raw.c](#), [roi.c](#), [sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [112](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB2_BS _IOWR(DVR_ENC_IOC
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB2_BS, &data)
```

- explanation : get sub2-bitstream buffer to user space.
- parameter :
 - pointer data** : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Examples:

[sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [121](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1_BS_OFFSET
```

```
ioctl(enc_fd,
DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1_BS_OFFSET,
&sub1_bs_buf_offset)
```

- explanation : get output buffer offset for Sub1-bitstream to user space
- parameter :
 - pointer sub1_bs_buf_offset** : argument from user space ioctl parameter, it means sub1 bitstream buffer offset.

Examples:

[capture_raw.c](#), [roi.c](#), [sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [131](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB2_BS_OFFSET
```

```
ioctl(enc_fd,  
DVR_ENC_QUERY_OUTPUT_BUFFER_SUB2_BS_OFFSET,  
&sub2_bs_buf_offset)
```

- explanation : get output buffer offset for Sub2-bitstream to user space
- parameter :
pointer sub2_bs_buf_offset : argument from user space ioctl parameter, it means sub2 bitstream buffer offset.

Examples:

[sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [141](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_RESET_INTRA _IOR(DVR_ENC_IOC_MAGIC, 1
```

```
ioctl(enc_fd, DVR_ENC_RESET_INTRA, &stream_num)
```

- explanation : get i-frame as possible
- parameter :
pointer stream_num : argument from user space ioctl parameter, it means structure main-bitstream or sub1-bitstream or sub2-bitstream

Definition at line [150](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_SET_SUB_BS_PARAM _IOW(DVR_ENC_IOC_M
```

```
ioctl(enc_fd, DVR_ENC_SET_SUB_BS_PARAM, &sub_bitstream)
```

- explanation : get sub-bitstream parameter from user space, and set sub-bitstream parameter to device driver
- parameter :
pointer sub_bitstream : argument from user space ioctl parameter, it means structure ReproduceBitStream

Examples:

[capture_raw.c](#), [roi.c](#), [sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [159](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_SWAP_INTRA _IOR(DVR_ENC_IOC_MAGIC, 18,
```

```
ioctl(enc_fd, DVR_ENC_SWAP_INTRA, &stream_num)
```

- explanation : get i-frame definitely when next switch
- parameter :
pointer stream_num : argument from user space ioctl parameter, it means structure main-bitstream or sub1-bitstream or sub2-bitstream

Definition at line [168](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_SUB_PATH_DENOISE_CTRL _IOW(DVR_ENC_,
```

```
ioctl(enc_fd, DVR_ENC_SUB_PATH_DENOISE_CTRL, &control)
```

- explanation : get denoise option from user space, and set to device driver
- parameter :
pointer control : argument from user space ioctl parameter, it means denoise on/off

Definition at line [177](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB3_BS _IOWR(DVR_ENC_IOC
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB3_BS, &data)
```

- explanation : get sub3-bitstream buffer to user space.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Definition at line [186](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB4_BS _IOWR(DVR_ENC_IOC
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB4_BS, &data)
```

- explanation : get sub4-bitstream buffer to user space.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Definition at line [195](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB5_BS _IOWR(DVR_ENC_IOC
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB5_BS, &data)
```

- explanation : get sub5-bitstream buffer to user space.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Definition at line [204](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB6_BS _IOWR(DVR_ENC_IOC
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB6_BS, &data)
```

- explanation : get sub6-bitstream buffer to user space.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Definition at line [213](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB7_BS _IOWR(DVR_ENC_IOC_ID, 10, struct dvr_enc_queue_get)
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB7_BS, &data)
```

- explanation : get sub7-bitstream buffer to user space.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Definition at line [222](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB8_BS _IOWR(DVR_ENC_IOC_ID, 11, struct dvr_enc_queue_get)
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB8_BS, &data)
```

- explanation : get sub8-bitstream buffer to user space.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get

Definition at line [231](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB3_BS_OFFSET
```

```
ioctl(enc_fd,  
DVR_ENC_QUERY_OUTPUT_BUFFER_SUB3_BS_OFFSET,  
&sub3_bs_buf_offset)
```

- explanation : get output buffer offset for Sub3-bitstream to user space
- parameter :

pointer sub3_bs_buf_offset : argument from user space ioctl parameter, it means sub3 bitstream buffer offset.

Definition at line [241](#) of file [dvr_enc_ioctl.h](#).

#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB4_BS_OFFSET

```
ioctl(enc_fd,
DVR_ENC_QUERY_OUTPUT_BUFFER_SUB4_BS_OFFSET,
&sub4_bs_buf_offset)
```

- explanation : get output buffer offset for Sub4-bitstream to user space
- parameter :

pointer sub4_bs_buf_offset : argument from user space ioctl parameter, it means sub4 bitstream buffer offset.

Definition at line [251](#) of file [dvr_enc_ioctl.h](#).

#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB5_BS_OFFSET

```
ioctl(enc_fd,
DVR_ENC_QUERY_OUTPUT_BUFFER_SUB5_BS_OFFSET,
&sub5_bs_buf_offset)
```

- explanation : get output buffer offset for Sub5-bitstream to user space
- parameter :

pointer sub5_bs_buf_offset : argument from user space ioctl parameter, it means sub5 bitstream buffer offset.

Definition at line [261](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB6_BS_OFFSET
```

```
ioctl(enc_fd,  
DVR_ENC_QUERY_OUTPUT_BUFFER_SUB6_BS_OFFSET,  
&sub6_bs_buf_offset)
```

- explanation : get output buffer offset for Sub6-bitstream to user space
- parameter :
pointer sub6_bs_buf_offset : argument from user space ioctl parameter, it means sub6 bitstream buffer offset.

Definition at line [271](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB7_BS_OFFSET
```

```
ioctl(enc_fd,  
DVR_ENC_QUERY_OUTPUT_BUFFER_SUB7_BS_OFFSET,  
&sub7_bs_buf_offset)
```

- explanation : get output buffer offset for Sub7-bitstream to user space
- parameter :
pointer sub7_bs_buf_offset : argument from user space ioctl parameter, it means sub7 bitstream buffer offset.

Definition at line [281](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB8_BS_OFFSET
```

```
ioctl(enc_fd,  
DVR_ENC_QUERY_OUTPUT_BUFFER_SUB8_BS_OFFSET,  
&sub8_bs_buf_offset)
```

- explanation : get output buffer offset for Sub8-bitstream to user space

- parameter :
pointer sub8_bs_buf_offset : argument from user space ioctl parameter, it means sub8 bitstream buffer offset.

Definition at line [291](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_COPY _IOWR(DVR_ENC_IOC_M
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_COPY, &data)
```

- explanation : Get buffer and copy to user space. It includes the bitstream length.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure `dvr_enc_copy_buf`

Definition at line [300](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB1_BS_COPY _IOWR(DVR_E
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB1_BS_COPY, &data)
```

- explanation : get and copy sub8-bitstream to user space buffer.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure `dvr_enc_copy_buf`

Definition at line [309](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB2_BS_COPY _IOWR(DVR_E
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB2_BS_COPY, &data)
```

- explanation : get and copy sub8-bitstream to user space buffer.
- parameter :
pointer data : argument from user space ioctl parameter, it

means structure `dvr_enc_copy_buf`

Definition at line [318](#) of file `dvr_enc_ioctl.h`.

```
#define DVR_ENC_QUEUE_GET_SUB3_BS_COPY _IOWR(DVR_E
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB3_BS_COPY, &data)
```

- explanation : get and copy sub8-bitstream to user space buffer.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure `dvr_enc_copy_buf`

Definition at line [327](#) of file `dvr_enc_ioctl.h`.

```
#define DVR_ENC_QUEUE_GET_SUB4_BS_COPY _IOWR(DVR_E
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB4_BS_COPY, &data)
```

- explanation : get and copy sub8-bitstream to user space buffer.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure `dvr_enc_copy_buf`

Definition at line [336](#) of file `dvr_enc_ioctl.h`.

```
#define DVR_ENC_QUEUE_GET_SUB5_BS_COPY _IOWR(DVR_E
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB5_BS_COPY, &data)
```

- explanation : get and copy sub8-bitstream to user space buffer.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure `dvr_enc_copy_buf`

Definition at line [345](#) of file `dvr_enc_ioctl.h`.

```
#define DVR_ENC_QUEUE_GET_SUB6_BS_COPY _IOWR(DVR_E
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB6_BS_COPY, &data)
```

- explanation : get and copy sub8-bitstream to user space buffer.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_copy_buf

Definition at line [354](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB7_BS_COPY _IOWR(DVR_E
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB7_BS_COPY, &data)
```

- explanation : get and copy sub8-bitstream to user space buffer.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_copy_buf

Definition at line [363](#) of file [dvr_enc_ioctl.h](#).

```
#define DVR_ENC_QUEUE_GET_SUB8_BS_COPY _IOWR(DVR_E
```

```
ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB8_BS_COPY, &data)
```

- explanation : get and copy sub8-bitstream to user space buffer.
- parameter :
pointer data : argument from user space ioctl parameter, it means structure dvr_enc_copy_buf

Definition at line [373](#) of file [dvr_enc_ioctl.h](#).

doxygen 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

Data Structures | Defines |
Typedefs | Enumerations

source/dvr_type_define.h File Reference

[Go to the source code of this file.](#)

Data Structures

struct **FuncTag_tag**

function tag for videograph level More...

struct **DIM_tag**

set width and height More...

struct **RECT_tag**

set size and position More...

struct **POS_tag**

set position x and y More...

struct **ROI_ALL_tag**

set ROI position x, y, width, height, and enable/disable
More...

struct **QueueMemConfig_tag**

set queue memory configuration More...

struct **dvr_graph_vqueueut_tag**

set dvr graph queue configuration More...

struct **dvr_bs_data_tag**

dvr bit-stream data More...

struct **dvr_rate_tag**

struct **video_process_tag**

dvr video parameter. More...

struct **ScalerParamtag**

scalar parameter More...

Defines

```
#define TRUE 1
#define FALSE 0
#define GMDVR_MEM_CFG_FILE "/mnt/mtd/gmdvr_mem.cfg"
#define GMDVR_MAKE_FOURCC(a, b, c, d) ((int)((a)|(b)<<8|(c)<<16)
#define GMVAL_DO_NOT CARE (-54172099)
#define GMVAL_RATE(val, base) ((base<<16)|val)
#define GMVAL_RT_GET_VAL(rate) (rate&0x0000FFFF)
#define GMVAL_RT_GET_BASE(rate) (rate>>16)
#define GFID_DISP(plane_num) (0x10000+plane_num)
#define GFID_ENC(ch_num) (0x20000+ch_num)
#define GFID_DEC(ch_num) (0x30000+ch_num)
#define MTHD_USE_CMP 0x1
#define FN_NONE 0x0
#define FN_LIVEVIEW 0x1
#define FN_RECORD 0x2
#define FN_PLAYBACK 0x4
#define FN_CASCADE 0x8
#define FN_LCD_PARAM 0x10
#define FN_PLANE_PARAM 0x20
#define FN_SUB1_RECORD 0x40
#define FN_SUB2_RECORD 0x80
#define FN_SUB3_RECORD 0x100
#define FN_SUB4_RECORD 0x200
#define FN_SUB5_RECORD 0x400
#define FN_SUB6_RECORD 0x800
#define FN_SUB7_RECORD 0x1000
#define FN_SUB8_RECORD 0x2000
#define FN_UPDATE_METHOD 0x10000
#define FN_PB_SCL_LINK 0x20000
#define FN_METHOD_USE_CMP 0x8000000
#define FN_SUB_ALL_RECORD (FN_RECORD|FN_SUB1_RECOR
#define FN_RESET_TAG(ptag) { (ptag)->func=0; (ptag)->lv_ch=0; (pt
```

```

#define FN_COPY_TAG(newt, ptag) { (newt)->func=(ptag)->func; (ne
#define FN_SET_LV_CH(ptag, ch_num) { (ptag)->func|=FN_LIVEVIEW;
#define FN_SET_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_REC
#define FN_SET_SUB1_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_
#define FN_SET_SUB2_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_
#define FN_SET_SUB3_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_
#define FN_SET_SUB4_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_
#define FN_SET_SUB5_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_
#define FN_SET_SUB6_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_
#define FN_SET_SUB7_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_
#define FN_SET_SUB8_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_
#define FN_SET_PB_CH(ptag, ch_num) { (ptag)->func|=FN_PLAYBACK;
#define FN_SET_CAS_CH(ptag, ch_num) { (ptag)->func|=FN_CASC
#define FN_SET_FUNC(ptag, fnc) { (ptag)->func|=fnc; }
#define FN_REMOVE_FUNC(ptag, fnc) { (ptag)->func&=(~fnc); }
#define FN_SET_ALL(ptag)
#define FN_IS_UPDATE(ptag) ((ptag)->func&FN_UPDATE_METHOD)
#define FN_COMPARE(ptagA, ptagB) ( ((ptagA)->func==(ptagB)->fu
>cas_ch) )
#define FN_IS_EMPTY(ptag) ( ((ptag)->func==0) && ((ptag)->lv_ch==
#define FN_IS_FN(ptag, fn) ( (ptag)->func&(fn) )
#define FN_REMOVE_LV_CH(ptag, ch) ( (ptag)->lv_ch&=(~ch) )
#define FN_CHECK_MASK(ptagA, ptagB)
#define FN_ITEMS(ptag) (ptag)->func, (ptag)->lv_ch, (ptag)->rec_ch,
#define QNAME_LCD "lcd"
#define QNAME_3DI_SCL "3di_scl"
#define QNAME_LV_SCL "lv_scl"
#define QNAME_ENC_IN "enc_in"
#define QNAME_ENC_OUT "enc_out"
#define QNAME_SS_ENC_IN "ssenc_in"
#define QNAME_SS_ENC_OUT "ssenc_out"
#define QNAME_SUB1_ENC_IN "sub1enc_in"
#define QNAME_SUB1_ENC_OUT "sub1enc_out"
#define QNAME_SUB2_ENC_IN "sub2enc_in"
#define QNAME_SUB2_ENC_OUT "sub2enc_out"
#define QNAME_DEC_IN "dec_in"

```

```
#define QNAME_PB_SCL "pb_scl"
#define DBG_ENTITY_FNC 0x01
#define DBG_ENTITY_JOB_FLOW 0x02
#define DBG_DVR_FNC 0x04
#define DBG_DVR_DATA_FLOW 0x08
#define DBG_GRAPH_FNC 0x10
#define DBG_GRAPH_DATA 0x20
#define MCP_VIDEO_NTSC 0
#define MCP_VIDEO_PAL 1
#define MCP_VIDEO_VGA 2
#define DEFAULT_D1_WIDTH 720
#define DEFAULT_D1_HEIGHT 576
#define DEFAULT_CIF_WIDTH 352
#define DEFAULT_CIF_HEIGHT 288
#define DEFAULT_QCIF_WIDTH 176
#define DEFAULT_QCIF_HEIGHT 144
```

Typedefs

typedef struct FuncTag_tag	FuncTag
typedef enum QueueID_tag	QueueID
typedef struct DIM_tag	DIM
typedef struct RECT_tag	RECT
typedef struct POS_tag	POS
typedef struct ROI_ALL_tag	ROI_ALL
typedef struct QueueMemConfig_tag	QueMemCfg
typedef struct dvr_graph_vqueueut_tag	dvr_graph_vqueueut
typedef struct dvr_bs_data_tag	dvr_bs_data
typedef struct dvr_rate_tag	dvr_ratio
typedef struct video_process_tag	video_process
typedef struct ScalerParamtag	ScalerParam
typedef enum EncodeType_tag	EncodeType
typedef enum LCDOutputColorTypeTag	LCDOutputColorType
typedef enum LCDOutputModeTag	LCDOutputMode
typedef enum LiveviewFrameTypeTag	LiveviewFrameType
typedef enum LiveviewFrameModeTag	LiveviewFrameMode
typedef enum GM3DIFrameTypeTag	GM3DIFrameType
typedef enum LiveviewDMAOrderTag	LiveviewDMAOrder
typedef enum LiveviewScalerRatioTag	LiveviewScalerRatio
typedef enum CaptureColorModeTag	CaptureColorMode
typedef enum EncoderInputFormatTag	EncoderInputFormat
typedef enum DecoderOutputColorTag	DecoderOutputColor
typedef enum JpegEncInputFormatTag	JpegEncInputFormat
typedef enum JpegEnc420InputFormatTag	JpegEnc420InputFormat
typedef enum ScaleMethodTag	ScaleMethod
typedef enum ScaleColorModeTag	ScaleColorMode
typedef enum CapturePathTag	CapturePath
typedef enum LCDResolutionTag	LCDResolution

Enumerations

	QueueID_tag {
	QID_LCD = 10, QID_3DI_SCL, QID_LV_SCL,
	QID_ENC_IN,
	QID_ENC_OUT, QID_SS_ENC_IN, QID_SS_ENC_OUT,
enum	QID_SUB1_ENC_IN,
	QID_SUB1_ENC_OUT, QID_SUB2_ENC_IN,
	QID_SUB2_ENC_OUT, QID_DEC_IN,
	QID_PB_SCL
	}
	EncodeType_tag {
	ENC_TYPE_H264 = 0, ENC_TYPE_MPEG,
enum	ENC_TYPE_MJPEG, ENC_TYPE_YUV422,
	ENC_TYPE_COUNT
	}
	LCDOutputColorTypeTag {
	LCD_COLOR_YUV422, LCD_COLOR_YUV420,
	LCD_COLOR_RGB = 16, LCD_COLOR_ARGB = 16,
enum	LCD_COLOR_RGB888, LCD_COLOR_RGB565,
	LCD_COLOR_RGB555, LCD_COLOR_RGB444,
	LCD_COLOR_RGB8
	}
enum	LCDOutputModeTag { LCD_PROGRESSIVE = 0,
	LCD_INTERLACING = 1 }
enum	LiveviewFrameTypeTag { LVFRAME_EVEN_ODD = 0,
	LVFRAME_ENLARGE_ONE_FIELD = 1,
	LVFRAME_WEAVED_TWO_FIELDS = 2,
	LVFRAME_GM3DI_FORMAT = 3 }
enum	LiveviewFrameModeTag { LVFRAME_FRAME_MODE = 0,
	LVFRAME_FIELD_MODE = 1, LVFRAME_FIELD_MODE2 =
	2 }
enum	GM3DIFrameTypeTag { GM3DI_FIELD = 1, GM3DI_FRAME
	= 2 }
enum	LiveviewDMAOrderTag { DMAORDER_PACKET = 0,
	DMAORDER_3PLANAR = 1, DMAORDER_2PLANAR = 2 }
	LiveviewScalerRatioTag { CAPSCALER_KEEP_RATIO = 0,

```
enum CAPSCALER_NOT_KEEP_RATIO = 1 }

CaptureColorModeTag {
    CAPCOLOR_RGB888 = 0, CAPCOLOR_RGB565 = 1,
enum CAPCOLOR_YUV422 = 2, CAPCOLOR_YUV420_M0 = 3,
    CAPCOLOR_YUV420_M1 = 4
}

EncoderInputFormatTag { ENC_INPUT_H2642D = 0,
enum ENC_INPUT_MP42D = 1, ENC_INPUT_1D420 = 2,
    ENC_INPUT_1D422 = 3 }

DecoderOutputColorTag {
enum DEC_OUTPUT_COLOR_YUV420 = 4,
    DEC_OUTPUT_COLOR_YUV422 = 5 }

JpegEncInputFormatTag {
    JCS_yuv420 = 0, JCS_yuv422 = 1, JCS_yuv211 = 2,
enum JCS_yuv333 = 3,
    JCS_yuv222 = 4, JCS_yuv111 = 5, JCS_yuv400 = 6
}

JpegEnc420InputFormatTag {
    JENC_INPUT_MP42D = 0, JENC_INPUT_1D420 = 1,
enum JENC_INPUT_H2642D = 2, JENC_INPUT_DMAWRP420 = 3,
    JENC_INPUT_1D422 = 4
}

enum ScaleMethodTag { SCALE_LINEAR = 0,
    SCALE_NON_LINEAR, SCALE_METHOD_COUNT }

ScaleColorModeTag {
    SCALE_RGB888 = 0, SCALE_RGB565 = 1,
    SCALE_H264_YUV420_MODE0 = 2,
enum SCALE_H264_YUV420_MODE1 = 3,
    SCALE_YUV444 = 4, SCALE_YUV422 = 5,
    SCALE_MP4_YUV420_MODE0 = 6,
    SCALE_MP4_YUV420_MODE1 = 7
}

enum CapturePathTag { CAPPATH_DEFAULT = 0,
    CAPPATH_PATH_1, CAPPATH_PATH_2 }

LCDResolutionTag {
    LCD_RES_D1, LCD_RES_SVGA, LCD_RES_XGA,
    LCD_RES_XVGA,
```

```
enum  LCD_RES_SXGA, LCD_RES_1360x768,  
      LCD_RES_COUNT  
}
```

Define Documentation

```
#define TRUE 1
```

Definition at line [7](#) of file [dvr_type_define.h](#).

```
#define FALSE 0
```

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), and [pip.c](#).

Definition at line [11](#) of file [dvr_type_define.h](#).

```
#define GMDVR_MEM_CFG_FILE "/mnt/mtd/gmdvr_mem.cfg"
```

Definition at line [14](#) of file [dvr_type_define.h](#).

```
#define GMDVR_MAKE_FOURCC ( a,  
                           b,  
                           c,  
                           d  
                         )      (int)((a)|(b)<<8|(c)<<16|(d)<
```

Definition at line [15](#) of file [dvr_type_define.h](#).

```
#define GMVAL_DO_NOT CARE (-54172099)
```

Definition at line [17](#) of file [dvr_type_define.h](#).

```
#define GMVAL_RATE ( val,
```

```
base  
) ((base<<16)|val)
```

Definition at line [18](#) of file [dvr_type_define.h](#).

```
#define GMVAL_RT_GET_VAL( rate ) (rate&0x0000FFFF)
```

Definition at line [20](#) of file [dvr_type_define.h](#).

```
#define GMVAL_RT_GET_BASE( rate ) (rate>>16)
```

Definition at line [21](#) of file [dvr_type_define.h](#).

```
#define GFID_DISP( plane_num ) (0x10000+plane_num)
```

Definition at line [24](#) of file [dvr_type_define.h](#).

```
#define GFID_ENC( ch_num ) (0x20000+ch_num)
```

Definition at line [25](#) of file [dvr_type_define.h](#).

```
#define GFID_DEC( ch_num ) (0x30000+ch_num)
```

Definition at line [26](#) of file [dvr_type_define.h](#).

```
#define MTHD_USE_CMP 0x1
```

Definition at line [44](#) of file [dvr_type_define.h](#).

```
#define FN_NONE 0x0
```

Definition at line [46](#) of file [dvr_type_define.h](#).

#define FN_LIVEVIEW 0x1

Definition at line [47](#) of file [dvr_type_define.h](#).

#define FN_RECORD 0x2

Definition at line [48](#) of file [dvr_type_define.h](#).

#define FN_PLAYBACK 0x4

Definition at line [49](#) of file [dvr_type_define.h](#).

#define FN CASCADE 0x8

Definition at line [50](#) of file [dvr_type_define.h](#).

#define FN_LCD_PARAM 0x10

Definition at line [51](#) of file [dvr_type_define.h](#).

#define FN_PLANE_PARAM 0x20

Definition at line [52](#) of file [dvr_type_define.h](#).

#define FN_SUB1_RECORD 0x40

Definition at line [53](#) of file [dvr_type_define.h](#).

```
#define FN_SUB2_RECORD 0x80
```

Definition at line 54 of file [dvr_type_define.h](#).

```
#define FN_SUB3_RECORD 0x100
```

Definition at line 55 of file [dvr_type_define.h](#).

```
#define FN_SUB4_RECORD 0x200
```

Definition at line 56 of file [dvr_type_define.h](#).

```
#define FN_SUB5_RECORD 0x400
```

Definition at line 57 of file [dvr_type_define.h](#).

```
#define FN_SUB6_RECORD 0x800
```

Definition at line 58 of file [dvr_type_define.h](#).

```
#define FN_SUB7_RECORD 0x1000
```

Definition at line 59 of file [dvr_type_define.h](#).

```
#define FN_SUB8_RECORD 0x2000
```

Definition at line 60 of file [dvr_type_define.h](#).

```
#define FN_UPDATE_METHOD 0x10000
```

Definition at line **61** of file [dvr_type_define.h](#).

```
#define FN_PB_SCL_LINK 0x20000
```

Definition at line **62** of file [dvr_type_define.h](#).

```
#define FN_METHOD_USE_CMP 0x8000000
```

Definition at line **63** of file [dvr_type_define.h](#).

```
#define FN_SUB_ALL_RECORD (FN_RECORD|FN_SUB1_RECOR
```

Definition at line **64** of file [dvr_type_define.h](#).

```
#define FN_RESET_TAG( ptag ) { (ptag)->func=0; (ptag)->lv_ch
```

Examples:

[2ch_liveview.c](#), [2ch_playback.c](#), [capture_raw.c](#), [liveview.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [pip.c](#), [playback.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line **67** of file [dvr_type_define.h](#).

```
#define FN_COPY_TAG( newt,  
                      ptag  
                  ) { (newt)->func=(ptag)->func; (nev
```

Definition at line **68** of file [dvr_type_define.h](#).

```
#define FN_SET_LV_CH( ptag,  
                      ch_num  
                    ) { (ptag)->func|=FN_LIVEVIEW
```

Examples:

[2ch_liveview.c](#), [liveview.c](#), and [pip.c](#).

Definition at line [69](#) of file [dvr_type_define.h](#).

```
#define FN_SET_REC_CH( ptag,  
                      ch_num  
                    ) { (ptag)->func|=(FN_RECOR
```

Examples:

[capture_raw.c](#), [main-bitstream-record.c](#), [mjpeg-record.c](#), [motion-detection-mpeg4.c](#), [motion-detection.c](#), [mpeg4-record.c](#), [roi.c](#), [snapshot.c](#), [sub-bitstream-record.c](#), [update-bitrate.c](#), and [update-record-setting.c](#).

Definition at line [70](#) of file [dvr_type_define.h](#).

```
#define FN_SET_SUB1_REC_CH( ptag,  
                           ch_num  
                         ) { (ptag)->func|=(FN_S
```

Examples:

[capture_raw.c](#), [roi.c](#), [sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [71](#) of file [dvr_type_define.h](#).

```
#define FN_SET_SUB2_REC_CH( ptag,  
                           ch_num  
                         ) { (ptag)->func|=(FN_S
```

Examples:

[sub-bitstream-record.c](#), and [update-record-setting.c](#).

Definition at line [72](#) of file [dvr_type_define.h](#).

```
#define FN_SET_SUB3_REC_CH( ptag,  
                           ch_num  
                         ) { (ptag)->func|=(FN_S
```

Definition at line [73](#) of file [dvr_type_define.h](#).

```
#define FN_SET_SUB4_REC_CH( ptag,  
                           ch_num  
                         ) { (ptag)->func|=(FN_S
```

Definition at line [74](#) of file [dvr_type_define.h](#).

```
#define FN_SET_SUB5_REC_CH( ptag,  
                           ch_num  
                         ) { (ptag)->func|=(FN_S
```

Definition at line [75](#) of file [dvr_type_define.h](#).

```
#define FN_SET_SUB6_REC_CH( ptag,  
                           ch_num  
                         ) { (ptag)->func|=(FN_S
```

Definition at line [76](#) of file [dvr_type_define.h](#).

```
#define FN_SET_SUB7_REC_CH( ptag,
```

```
        ch_num  
    ) { (ptag)->func|=FN_SET_SUB8_REC_CH; }
```

Definition at line 77 of file [dvr_type_define.h](#).

```
#define FN_SET_SUB8_REC_CH( ptag,  
                           ch_num  
    ) { (ptag)->func|=FN_SET_SUB8_REC_CH; }
```

Definition at line 78 of file [dvr_type_define.h](#).

```
#define FN_SET_PB_CH( ptag,  
                     ch_num  
    ) { (ptag)->func|=FN_PLAYBACK; }
```

Examples:

[2ch_playback.c](#), [liveview.c](#), and [playback.c](#).

Definition at line 79 of file [dvr_type_define.h](#).

```
#define FN_SET_CAS_CH( ptag,  
                      ch_num  
    ) { (ptag)->func|=FN_CASCADE; }
```

Definition at line 80 of file [dvr_type_define.h](#).

```
#define FN_SET_FUNC( ptag,  
                   fnc  
    ) { (ptag)->func|=fnc; }
```

Definition at line 81 of file [dvr_type_define.h](#).

```
#define FN_REMOVE_FUNC( ptag,  
                      fnc  
)          { (ptag)->func&=(~fnc); }
```

Definition at line 82 of file [dvr_type_define.h](#).

```
#define FN_SET_ALL( ptag )
```

Value:

```
{   (ptag)->func=(0xFF); /* Note: 'func' value  
doesn't include FN_UPDATE_METHOD. */ \  
    (ptag)  
->lv_ch=0xFFFFFFFF; (ptag)->rec_ch=0xFFFFFFFF; (p  
tag)->pb_ch=0xFFFFFFFF; (ptag)->cas_ch=0xF; }
```

Definition at line 83 of file [dvr_type_define.h](#).

```
#define FN_IS_UPDATE( ptag ) ((ptag)->func&FN_UPDATE_ME
```

Definition at line 85 of file [dvr_type_define.h](#).

```
#define FN_COMPARE( ptagA,  
                  ptagB  
)          ( ((ptagA)->func==(ptagB)->func)
```

Definition at line 86 of file [dvr_type_define.h](#).

```
#define FN_IS_EMPTY( ptag ) ( ((ptag)->func==0) && ((ptag)->l
```

Definition at line 87 of file [dvr_type_define.h](#).

```
#define FN_IS_FN( ptag,
```

fn
) ((ptag)->func&(fn))

Definition at line 88 of file [dvr_type_define.h](#).

```
#define FN_REMOVE_LV_CH( ptag,  
                         ch  
                     ) ( (ptag)->lv_ch&=(~ch) )
```

Definition at line 89 of file [dvr_type_define.h](#).

```
#define FN_CHECK_MASK( ptagA,  
                      ptagB  
                    )
```

Value:

```
(((((ptagA)->func&(ptagB)->func)==FN_LIVEVIEW)?((p  
tagA)->lv_ch&(ptagB)->lv_ch):  
    (((((ptagA)->func&(ptagB)->func)==FN_R  
ECORD)?((ptagA)->rec_ch&(ptagB)->rec_ch):  
        (((((ptagA)->func&(ptagB)->func)==  
FN_SUB1_RECORD)?((ptagA)->rec_ch&(ptagB)->rec_ch)  
        :  
            (((((ptagA)->func&(ptagB)->fun  
c)==FN_SUB2_RECORD)?((ptagA)->rec_ch&(ptagB)->rec  
_ch):  
                (((((ptagA)->func&(ptagB)->fun  
c)==FN_SUB3_RECORD)?((ptagA)->rec_ch&(ptagB)->rec  
_ch):  
                    (((((ptagA)->func&(ptagB)->fun  
c)==FN_SUB4_RECORD)?((ptagA)->rec_ch&(ptagB)->rec  
_ch):  
                        (((((ptagA)->func&(ptagB)->fun  
c)==FN_SUB5_RECORD)?((ptagA)->rec_ch&(ptagB)->rec
```

```
_ch): \
        (((((ptagA)->func&(ptagB)->fun
c)==FN_SUB6_RECORD)?((ptagA)->rec_ch&(ptagB)->rec
_ch): \
        (((((ptagA)->func&(ptagB)->fun
c)==FN_SUB7_RECORD)?((ptagA)->rec_ch&(ptagB)->rec
_ch): \
        (((((ptagA)->func&(ptagB)->fun
c)==FN_SUB8_RECORD)?((ptagA)->rec_ch&(ptagB)->rec
_ch): \
        (((((ptagA)->func&(ptagB)->func)==FN_PLAYBACK)?((ptagA)->pb_ch&(ptagB)->pb_
ch): \
        (((((ptagA)->func&(ptagB)->func)==FN_CASCADE)?((ptagA)->cas_ch&(ptagB)->cas_
ch): \
        ((ptagA)->func&(ptagB)->f
unc) ))))))))))))
```

Definition at line [92](#) of file [dvr_type_define.h](#).

```
#define FN_ITEMS( ptag ) (ptag)->func, (ptag)->lv_ch, (ptag)->r
```

Definition at line [116](#) of file [dvr_type_define.h](#).

```
#define QNAME_LCD "lcd"
```

Definition at line [120](#) of file [dvr_type_define.h](#).

```
#define QNAME_3DI_SCL "3di_scl"
```

Definition at line [121](#) of file [dvr_type_define.h](#).

```
#define QNAME_LV_SCL "lv_scl"
```

Definition at line [122](#) of file `dvr_type_define.h`.

```
#define QNAME_ENC_IN "enc_in"
```

Definition at line [123](#) of file `dvr_type_define.h`.

```
#define QNAME_ENC_OUT "enc_out"
```

Definition at line [124](#) of file `dvr_type_define.h`.

```
#define QNAME_SS_ENC_IN "ssenc_in"
```

Definition at line [125](#) of file `dvr_type_define.h`.

```
#define QNAME_SS_ENC_OUT "ssenc_out"
```

Definition at line [126](#) of file `dvr_type_define.h`.

```
#define QNAME_SUB1_ENC_IN "sub1enc_in"
```

Definition at line [127](#) of file `dvr_type_define.h`.

```
#define QNAME_SUB1_ENC_OUT "sub1enc_out"
```

Definition at line [128](#) of file `dvr_type_define.h`.

```
#define QNAME_SUB2_ENC_IN "sub2enc_in"
```

Definition at line [129](#) of file `dvr_type_define.h`.

```
#define QNAME_SUB2_ENC_OUT "sub2enc_out"
```

Definition at line [130](#) of file `dvr_type_define.h`.

```
#define QNAME_DEC_IN "dec_in"
```

Definition at line [131](#) of file `dvr_type_define.h`.

```
#define QNAME_PB_SCL "pb_scl"
```

Definition at line [132](#) of file `dvr_type_define.h`.

```
#define DBG_ENTITY_FNC 0x01
```

Definition at line [278](#) of file `dvr_type_define.h`.

```
#define DBG_ENTITY_JOB_FLOW 0x02
```

Definition at line [279](#) of file `dvr_type_define.h`.

```
#define DBG_DVR_FNC 0x04
```

Definition at line [280](#) of file `dvr_type_define.h`.

```
#define DBG_DVR_DATA_FLOW 0x08
```

Definition at line [281](#) of file `dvr_type_define.h`.

```
#define DBG_GRAPH_FNC 0x10
```

Definition at line [282](#) of file [dvr_type_define.h](#).

```
#define DBG_GRAPH_DATA 0x20
```

Definition at line [283](#) of file [dvr_type_define.h](#).

```
#define MCP_VIDEO_NTSC 0
```

video mode : NTSC

Examples:

[2ch_liveview.c](#), and [pip.c](#).

Definition at line [298](#) of file [dvr_type_define.h](#).

```
#define MCP_VIDEO_PAL 1
```

video mode : PAL

Examples:

[2ch_liveview.c](#), and [pip.c](#).

Definition at line [299](#) of file [dvr_type_define.h](#).

```
#define MCP_VIDEO_VGA 2
```

video mode : VGA

Definition at line [300](#) of file [dvr_type_define.h](#).

```
#define DEFAULT_D1_WIDTH 720
```

Definition at line [412](#) of file `dvr_type_define.h`.

```
#define DEFAULT_D1_HEIGHT 576
```

Definition at line [413](#) of file `dvr_type_define.h`.

```
#define DEFAULT_CIF_WIDTH 352
```

Definition at line [414](#) of file `dvr_type_define.h`.

```
#define DEFAULT_CIF_HEIGHT 288
```

Definition at line [415](#) of file `dvr_type_define.h`.

```
#define DEFAULT_QCIF_WIDTH 176
```

Definition at line [416](#) of file `dvr_type_define.h`.

```
#define DEFAULT_QCIF_HEIGHT 144
```

Definition at line [417](#) of file `dvr_type_define.h`.

Typedef Documentation

typedef struct FuncTag_tag FuncTag

function tag for videograph level

typedef enum QueueID_tag QueueID

typedef struct DIM_tag DIM

set width and height

typedef struct RECT_tag RECT

set size and position

typedef struct POS_tag POS

set position x and y

typedef struct ROI_ALL_tag ROI_ALL

set ROI position x, y, width, height, and enable/disable

typedef struct QueueMemConfig_tag QueMemCfg

set queue memory configuration

typedef struct dvr_graph_vqueueut_tag dvr_graph_vqueueut

set dvr graph queue configuration

typedef struct dvr_bs_data_tag dvr_bs_data

dvr bit-stream data

typedef struct dvr_rate_tag dvr_ratio

typedef struct video_process_tag video_process

dvr video parameter.

typedef struct ScalerParamtag ScalerParam

scalar parameter

typedef enum EncodeType_tag EncodeType

typedef enum LCDOutputColorTypeTag LCDOutputColorType

typedef enum LCDOutputModeTag LCDOutputMode

typedef enum LiveviewFrameTypeTag LiveviewFrameType

typedef enum LiveviewFrameModeTag LiveviewFrameMode

typedef enum GM3DIFrameTypeTag GM3DIFrameType

typedef enum LiveviewDMAOrderTag LiveviewDMAOrder

typedef enum [LiveviewScalerRatioTag](#) LiveviewScalerRatio

typedef enum [CaptureColorModeTag](#) CaptureColorMode

typedef enum [EncoderInputFormatTag](#) EncoderInputFormat

typedef enum [DecoderOutputColorTag](#) DecoderOutputColor

typedef enum [JpegEncInputFormatTag](#) JpegEncInputFormat

typedef enum [JpegEnc420InputFormatTag](#) JpegEnc420InputFormat

typedef enum [ScaleMethodTag](#) ScaleMethod

typedef enum [ScaleColorModeTag](#) ScaleColorMode

typedef enum [CapturePathTag](#) CapturePath

typedef enum [LCDResolutionTag](#) LCDResolution

Enumeration Type Documentation

enum QueueID_tag

Enumerator:

QID_LCD
QID_3DI_SCL
QID_LV_SCL
QID_ENC_IN
QID_ENC_OUT
QID_SS_ENC_IN
QID_SS_ENC_OUT
QID_SUB1_ENC_IN
QID_SUB1_ENC_OUT
QID_SUB2_ENC_IN
QID_SUB2_ENC_OUT
QID_DEC_IN
QID_PB_SCL

Definition at line [137](#) of file `dvr_type_define.h`.

enum EncodeType_tag

Enumerator:

ENC_TYPE_H264
ENC_TYPE_MPEG
ENC_TYPE_MJPEG
ENC_TYPE_YUV422
ENC_TYPE_COUNT

Definition at line [269](#) of file `dvr_type_define.h`.

enum LCDOutputColorTypeTag

Enumerator:

LCD_COLOR_YUV422
LCD_COLOR_YUV420
LCD_COLOR_RGB
LCD_COLOR_ARGB
LCD_COLOR_RGB888
LCD_COLOR_RGB565
LCD_COLOR_RGB555
LCD_COLOR_RGB444
LCD_COLOR_RGB8

Definition at line **286** of file **dvr_type_define.h**.

enum LCDOutputModeTag

Enumerator:

LCD_PROGRESSIVE
LCD_INTERLACING

Definition at line **302** of file **dvr_type_define.h**.

enum LiveviewFrameTypeTag

Enumerator:

LVFRAME_EVEN_ODD
LVFRAME_ENLARGE_ONE_FIELD
LVFRAME_WEAVED_TWO_FIELDS
LVFRAME_GM3DI_FORMAT

Definition at line **307** of file **dvr_type_define.h**.

enum LiveviewFrameModeTag

Enumerator:

LVFRAME_FRAME_MODE
LVFRAME_FIELD_MODE
LVFRAME_FIELD_MODE2

Definition at line **314** of file **dvr_type_define.h**.

enum GM3DIFrameTypeTag

Enumerator:

GM3DI_FIELD
GM3DI_FRAME

Definition at line **320** of file **dvr_type_define.h**.

enum LiveviewDMAOrderTag

Enumerator:

DMAORDER_PACKET
DMAORDER_3PLANAR
DMAORDER_2PLANAR

Definition at line **326** of file **dvr_type_define.h**.

enum LiveviewScalerRatioTag

Enumerator:

CAPSCALER_KEEP_RATIO
CAPSCALER_NOT_KEEP_RATIO

Definition at line **332** of file **dvr_type_define.h**.

enum CaptureColorModeTag

Enumerator:

CAPCOLOR_RGB888
CAPCOLOR_RGB565
CAPCOLOR_YUV422
CAPCOLOR_YUV420_M0
CAPCOLOR_YUV420_M1

Definition at line [338](#) of file **dvr_type_define.h**.

enum EncoderInputFormatTag

Enumerator:

ENC_INPUT_H2642D
ENC_INPUT_MP42D
ENC_INPUT_1D420
ENC_INPUT_1D422

Definition at line [346](#) of file **dvr_type_define.h**.

enum DecoderOutputColorTag

Enumerator:

DEC_OUTPUT_COLOR_YUV420
DEC_OUTPUT_COLOR_YUV422

Definition at line [353](#) of file **dvr_type_define.h**.

enum JpegEncInputFormatTag

Enumerator:

JCS_yuv420
JCS_yuv422
JCS_yuv211
JCS_yuv333
JCS_yuv222
JCS_yuv111
JCS_yuv400

Definition at line [358](#) of file `dvr_type_define.h`.

enum JpegEnc420InputFormatTag

Enumerator:

JENC_INPUT_MP42D
JENC_INPUT_1D420
JENC_INPUT_H2642D
JENC_INPUT_DMAWRP420
JENC_INPUT_1D422

Definition at line [368](#) of file `dvr_type_define.h`.

enum ScaleMethodTag

Enumerator:

SCALE_LINEAR
SCALE_NON_LINEAR
SCALE_METHOD_COUNT

Definition at line [376](#) of file `dvr_type_define.h`.

enum ScaleColorModeTag

Enumerator:

```
SCALE_RGB888  
SCALE_RGB565  
SCALE_H264_YUV420_MODE0  
SCALE_H264_YUV420_MODE1  
SCALE_YUV444  
SCALE_YUV422  
SCALE_MP4_YUV420_MODE0  
SCALE_MP4_YUV420_MODE1
```

Definition at line [382](#) of file `dvr_type_define.h`.

enum CapturePathTag

Enumerator:

```
CAPPATH_DEFAULT  
CAPPATH_PATH_1  
CAPPATH_PATH_2
```

Definition at line [394](#) of file `dvr_type_define.h`.

enum LCDResolutionTag

Enumerator:

```
LCD_RES_D1  
LCD_RES_SVGA  
LCD_RES_XGA  
LCD_RES_XVGA  
LCD_RES_SXGA  
LCD_RES_1360x768  
LCD_RES_COUNT
```

Definition at line [401](#) of file `dvr_type_define.h`.

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

Examples

Here is a list of all examples:

- [2ch_liveview.c](#)
 - [2ch_playback.c](#)
 - [capture_raw.c](#)
 - [liveview.c](#)
 - [main-bitstream-record.c](#)
 - [mpeg-record.c](#)
 - [motion-detection-mpeg4.c](#)
 - [motion-detection.c](#)
 - [mpeg4-record.c](#)
 - [pip.c](#)
 - [playback.c](#)
 - [roi.c](#)
 - [snapshot.c](#)
 - [sub-bitstream-record.c](#)
 - [update-bitrate.c](#)
 - [update-record-setting.c](#)
-

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

2ch_liveview.c

```
/**  
 * this sample code implement two channel liveview  
function.  
*  
*/  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
#include <sys/mman.h>  
#include <poll.h>  
  
#include "dvr_common_api.h"  
#include "dvr_disp_api.h"  
#include "dvr_enc_api.h"  
  
int dvr_fd = 0;  
int disp_fd = 0;  
int main_disp_no = 0;  
int main_plane_id = 0;  
int open_flag = 0;  
int is_NTSC = TRUE; //if FALSE, it's PAL  
  
  
char getch(void)  
{  
    int n = 1;
```

```
    unsigned char ch;
    struct timeval tv;
    fd_set rfds;

    FD_ZERO(&rfds);
    FD_SET(0, &rfds);
    tv.tv_sec = 10;
    tv.tv_usec = 0;
    n = select(1, &rfds, NULL, NULL, &tv);
    if (n > 0) {
        n = read(0, &ch, 1);
        if (n == 1)
            return ch;
        return n;
    }
    return -1;
}

int setup_lv_channel(int ch_num, int cmd, int is_use_scaler, int di_mode, int mode, int is_3DI, int is_denoise, int dn_mode, DIM *src_dim, RECT *src_rect, RECT *dst_rect)
{
    int ret;
    dvr_disp_control disp_ctrl;
    memset(&disp_ctrl, 0x0, sizeof(dvr_disp_control));
    disp_ctrl.type = DISP_TYPE_LIVEVIEW;
    disp_ctrl.channel = ch_num;
    disp_ctrl.command = cmd;
    if(cmd != DISP_STOP)
    {
        disp_ctrl.src_param.lv.cap_path = ch_num;

        disp_ctrl.src_param.lv.di_mode = di_mode;
        disp_ctrl.src_param.lv.mode = mode;
```

```
        disp_ctrl.src_param.lv.vp_param.is_3DI = is_3DI;
        disp_ctrl.src_param.lv.vp_param.is_denoise = is_denoise;
        disp_ctrl.src_param.lv.vp_param.denoise_mode = dn_mode;

        disp_ctrl.src_param.lv.dma_order = DMAORDER_PACKET;
        disp_ctrl.src_param.lv.scale_indep = CAPSCALE_NOT_KEEP_RATIO;
        disp_ctrl.src_param.lv.input_system = (is_NTSC)? MCP_VIDEO_NTSC: MCP_VIDEO_PAL;
        disp_ctrl.src_param.lv.cap_rate = (is_NTSC)? 30: 25;

        disp_ctrl.src_param.lv.color_mode = CAPCOL_OR_YUV422;
        disp_ctrl.dst_param.lv.plane_id = main_plane_id;

        disp_ctrl.src_param.lv.is_use_scaler = is_use_scaler;
        disp_ctrl.src_param.lv.dim.width = src_dim->width;
        disp_ctrl.src_param.lv.dim.height = src_dim->height;

        if(is_use_scaler)
        {
            if(src_rect)
            {
                disp_ctrl.src_param.lv.win.x = src_rect->x;
                disp_ctrl.src_param.lv.win.y = src_rect->y;
                disp_ctrl.src_param.lv.win.width =
```

```

    src_rect->width;
                disp_ctrl.src_param.lv.win.height
= src_rect->height;
}
else
{
    disp_ctrl.src_param.lv.win.x = 0;
    disp_ctrl.src_param.lv.win.y = 0;
    disp_ctrl.src_param.lv.win.width =
src_dim->width;
    disp_ctrl.src_param.lv.win.height
= src_dim->height;
}

        disp_ctrl.src_param.lv.scl_param.src_f
mt = SCALE_YUV422;
        disp_ctrl.src_param.lv.scl_param.dst_f
mt = SCALE_YUV422;
        disp_ctrl.src_param.lv.scl_param.scale
_mode = SCALE_LINEAR;
        disp_ctrl.src_param.lv.scl_param.is_di
ther = FALSE;
        disp_ctrl.src_param.lv.scl_param.is_co
rrection = FALSE;
        disp_ctrl.src_param.lv.scl_param.is_al
bum = TRUE;
        disp_ctrl.src_param.lv.scl_param.des_l
evel = 0;
}
        disp_ctrl.dst_param.lv.win.x = dst_rect->x
;
        disp_ctrl.dst_param.lv.win.y = dst_rect->y
;
        disp_ctrl.dst_param.lv.win.width = dst_rec
t->width;
        disp_ctrl.dst_param.lv.win.height = dst_re
ct->height;

```

```
    }

    ret = ioctl(disp_fd, DVR_DISP_CONTROL, &disp_ctrl);

    return ret;
}

int run_lv_command(int ch_num)
{
    int ret;
    FuncTag tag;

    FN_RESET_TAG(&tag);
    FN_SET_LV_CH(&tag, ch_num);

    ret = ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    if(ret<0)
        return -1;

    return ret;
}

int do_liveview_closech(int ch_num)
{
    int ret;

    ret = setup_lv_channel(ch_num, DISP_STOP, 0, 0
    , 0, FALSE, FALSE, 0, NULL, NULL, NULL);
    if(ret<0)
        return -1;

    ret = run_lv_command(ch_num);

    return ret;
}
```

```
int do_disp_startup()
{
    int i, ret;
    dvr_disp_disp_param         disp_param;
    dvr_disp_update_disp_param  disp_update_param;
    dvr_disp_plane_param        plane_param[3];
    dvr_disp_update_plane_param plane_update_pa;
    dvr_disp_control           dsp_ctl;

    open_flag = 1;

    disp_fd = open("/dev/dvr_disp", O_RDWR);

    memset(&disp_param, 0x0, sizeof(dvr_disp_disp_param));
    memset(&disp_update_param, 0x0, sizeof(dvr_disp_update_disp_param));
    memset(plane_param, 0x0, sizeof(dvr_disp_plane_param) * 3);
    memset(&plane_update_pa, 0x0, sizeof(dvr_disp_update_plane_param));
    memset(&dsp_ctl, 0x0, sizeof(dvr_disp_control));
}

main_disp_no = 0;

// query LCD1 information
disp_param.disp_num = main_disp_no;
ret = ioctl(disp_fd, DVR_DISP_GET_DISP_PARAM,
&disp_param);

usleep(100000);

// set BG_AND_2PLANE, which means we need 1 background and another 2 planes
disp_update_param.disp_num = main_disp_no;
disp_update_param.param = DISP_PARAM_PLANE_COM
```

```
BINATION;
    disp_update_param.val.plane_comb = BG_ONLY; //BG_AND_2PLANE;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
    if (ret < 0)
        return -1;

    disp_update_param.disp_num = main_disp_no;
    disp_update_param.param = DISP_PARAM_OUTPUT_SYSTEM;
}

    disp_update_param.val.output_system = MCP_VIDEO_VGA;
    disp_update_param.val.display_rate = is_NTSC?
30 : 25;

    ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
    if (ret < 0)
        return -1;

    disp_update_param.param = DISP_PARAM_APPLY;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
    if (ret < 0)
        return -1;

// query 3 planes information
for(i = 0; i < 3; i++) {
    plane_param[i].disp_num = main_disp_no;
    plane_param[i].plane_num = i;
    ret = ioctl(disp_fd, DVR_DISP_GET_PLANE_PARAM
, &plane_param[i]);
    if (ret < 0)
        return -1;
```

```
        usleep(50000);
        if (i == 0)
            main_plane_id = plane_param[i].param.p
lane_id;
    }

    // set color mode for background plane
    plane_update_pa.plane_id = plane_param[0].param
.plane_id;
    plane_update_pa.param = PLANE_PARAM_COLOR_MODE
;
    plane_update_pa.val.color_mode = LCD_COLOR_YUV
422;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PAR
AM, &plane_update_pa);
    if (ret < 0)
        return -1;

    // set data mode for background plane
    plane_update_pa.plane_id = plane_param[0].param
.plane_id;
    plane_update_pa.param = PLANE_PARAM_DATA_MODE;
    plane_update_pa.val.data_mode = LCD_PROGRESSIV
E;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PAR
AM, &plane_update_pa);
    if (ret < 0)
        return -1;

    plane_update_pa.param = PLANE_PARAM_APPLY;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PAR
AM, &plane_update_pa);
    if (ret < 0)
        return -1;

    return 0;
}
```

```
int do_disp_endup()
{
    if (!open_flag) {
        printf("Multi close\n");
        return -1;
    }

    if (disp_fd > 0)
        close(disp_fd);

    disp_fd = 0;
    open_flag = 0;

    return 0;
}

int open_dvr_common(void)
{
    dvr_fd = open("/dev/dvr_common", O_RDWR);
    if (dvr_fd < 0) {
        perror("Open [/dev/dvr_common] failed:");
        return -1;
    }
    return 0;
}

void close_dvr_common(void)
{
    close(dvr_fd);
}

int main(int argc, char *argv[])
{
    int ret = 0;
    DIM dim;
    RECT win;
```

```
char key;
dvr_disp_clear_param clear_win_st;

open_dvr_common();

do_disp_startup();

memset(&clear_win_st, 0x0, sizeof(dvr_disp_clear_param));
clear_win_st.win.x = 352;
clear_win_st.win.y = 0;
clear_win_st.win.width = 720;
clear_win_st.win.height = 480;
clear_win_st.pattern = 0x10801080;
clear_win_st.plane_id = 0;

ret = ioctl(disp_fd, DVR_DISP_CLEAR_WIN, &clear_win_st);

while(1)
{
    key = getch();
    if (key == 'q' || key == 'Q')
        break;
    switch(key)
    {
        case 'x':
            dim.width = 352;
            dim.height = 240;
            win.x = 0;
            win.y = 0;
            win.width = 352;
            win.height = 240;
            ret=setup_lv_channel(0, DISP_START
, 0, LVFRAME_WEAVED_TWO_FIELDS, LVFRAME_FRAME_MODE
, FALSE, FALSE, 0, &dim, NULL, &win);
```

```
        break;
    case 'y':
        ret=run_lv_command(0);
        break;

    case 'z':
        ret=do_liveview_closech(0);
        break;
    case 'a':
        dim.width = 352;
        dim.height = 240;
        win.x = 352;
        win.y = 0;
        win.width = 352;
        win.height = 240;
        ret=setup_lv_channel(2, DISP_START
, 0, LVFRAME_WEAVED_TWO_FIELDS, LVFRAME_FRAME_MODE
, FALSE, FALSE, 0, &dim, NULL, &win);

        break;
    case 'b':
        ret=run_lv_command(2);
        break;

    case 'c':
        ret=do_liveview_closech(2);
        break;
    }
}
do_disp_endup();

close_dvr_common();
printf("finish\n");
return 0;
}
```

Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

2ch_playback.c

```
/**  
 * this sample code implement playback function, a  
nd playback for 20 seconds.  
 * demo file is CH0_video_0.avi,  
 * please put the demo file at the same directory  
with executed binary file.  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
#include <pthread.h>  
  
#include "dvr_common_api.h"  
#include "dvr_disp_api.h"  
#include "dvr_enc_api.h"  
#include "dvr_dec_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int disp_fd = 0;  
int dec_fd[2] = {0};  
int main_disp_no = 0;  
int main_plane_id = 0;  
int open_flag = 0;
```

```
int is_NTSC = TRUE;

#define FUNCTION_NULL      -1
#define FUNCTION_NORMAL     0

int menu_func = FUNCTION_NULL;

unsigned char *pbbs_buf[2];
int dec_buf_size[2];
pthread_t thr_id_packet_reader[2];
int flag_exit_reader_loop[2]={FALSE};
HANDLE pb_file[2]= {NULL};
AviMainHeader pb_main_header[2];
AviStreamHeader pb_stream_header[2];
GmAviStreamFormat pb_stream_format[2];
int stream_id_pkt[2] = {0};
pthread_mutex_t dec_lock[2];
struct pollfd dec_fds[2];
int is_st_vga = TRUE;

RECT D1_4CH[4] = {
    { 0, 0, 360, 240}, {360, 0, 360, 240},
    { 0,240, 360, 240}, {360,240, 360, 240}
};

RECT VGA_4CH[4] = {
    { 0, 0, 320, 240}, {320, 0, 320, 240},
    { 0,360, 512, 360}, {512,360, 512, 360}
};

char getch(void)
{
    int n = 1;
    unsigned char ch;
    struct timeval tv;
    fd_set rfds;
```

```
    FD_ZERO(&rfds);
    FD_SET(0, &rfds);
    tv.tv_sec = 10;
    tv.tv_usec = 0;
    n = select(1, &rfds, NULL, NULL, &tv);
    if (n > 0) {
        n = read(0, &ch, 1);
        if (n == 1)
            return ch;
        return n;
    }
    return -1;
}

int do_disp_startup()
{
    int i, ret;
    dvr_disp_disp_param          disp_param;
    dvr_disp_update_disp_param   disp_update_param;
    dvr_disp_plane_param         plane_param[3];
    dvr_disp_update_plane_param  plane_update_pa;
    dvr_disp_control             dsp_ctl;

    if (open_flag) {
        printf("multi open\n");
        return -1;
    }
    open_flag = 1;

    disp_fd = open("/dev/dvr_disp", O_RDWR);
    if (disp_fd < 0) {
        perror("Open failed:");
        open_flag = 0;
        return -1;
    }

    memset(&disp_param, 0x0, sizeof(dvr_disp_disp_
```

```
param));
    memset(&disp_update_param, 0x0, sizeof(dvr_disp_update_disp_param));
    memset(plane_param, 0x0, sizeof(dvr_disp_plane_param) * 3);
    memset(&plane_update_pa, 0x0, sizeof(dvr_disp_update_plane_param));
    memset(&dsp_ctl, 0x0, sizeof(dvr_disp_control));
}

main_disp_no = 0;

// query LCD1 information
disp_param.disp_num = main_disp_no;
ret = ioctl(disp_fd, DVR_DISP_GET_DISP_PARAM,
&disp_param);
if (ret < 0)
    return -1;
printf("LCD(%d): dim(%d-%d) res(in=%d,out=%d) plane_comb(%d) system(%d) mode(%d)\n",
       1, disp_param.dim.width, disp_param.dim.height, disp_param.res.input_res, disp_param.res.output_type,
       disp_param.plane_comb, disp_param.output_system, disp_param.output_mode);
printf("          : color attrib.: br(%d)sa(%d)co(%d)-hus(%d)huc(%d)sh0(%d)sh1-(%d)shth0(%d)sh1(%d)\n",
       disp_param.color_attrib.brightness, disp_param.color_attrib.saturation, disp_param.color_attrib.contrast,
       disp_param.color_attrib.huesin, disp_param.color_attrib.huecos, disp_param.color_attrib.sharpnessk0,
       disp_param.color_attrib.sharpnessk1, disp_param.color_attrib.sharpness_thres0, disp_param.color_attrib.sharpness_thres1);
```

```

        printf("      : transparent: color1(%d,%d
) color2(%d,%d)\n",
            disp_param.transparent_color[0].is_ena
ble, disp_param.transparent_color[0].color,
            disp_param.transparent_color[1].is_ena
ble, disp_param.transparent_color[1].color);
            usleep(100000);

        // set BG_AND_2PLANE, which means we need
1 background and another 2 planes
        disp_update_param.disp_num = main_disp_no;
        disp_update_param.param = DISP_PARAM_PLANE
_COMBINATION;
        disp_update_param.val.plane_comb = BG_ONLY;
//BG_AND_2PLANE;
        ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_
PARAM, &disp_update_param);
        if (ret < 0)
            return -1;

        disp_update_param.disp_num = main_disp_no;
        disp_update_param.param = DISP_PARAM_OUTPU
T_SYSTEM;
        disp_update_param.val.output_system = MCP_
VIDEO_VGA;
        disp_update_param.val.display_rate = is_NT
SC? 30 : 25;

        ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_
PARAM, &disp_update_param);
        if (ret < 0)
            return -1;
        disp_update_param.param = DISP_PARAM_APPLY
;
        ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_
PARAM, &disp_update_param);
        if (ret < 0)

```

```
    return -1;

    // query 3 planes information
    for(i = 0; i < 3; i++) {
        plane_param[i].disp_num = main_disp_no
    ;
        plane_param[i].plane_num = i;
        ret = ioctl(disp_fd, DVR_DISP_GET_PLAN
E_PARAM, &plane_param[i]);
        if (ret < 0)
            return -1;
        printf("LCD(%d)-plane(%d): ID(%d) rect
(%d-%d,%d-%d) data_mode(%d) color_mode(%d)\n",
               main_disp_no, plane_param[i].plane
_num, plane_param[i].param.plane_id,
               plane_param[i].param.win.x, plane_
param[i].param.win.y, plane_param[i].param.win.wid
th, plane_param[i].param.win.height,
               plane_param[i].param.data_mode, pl
ane_param[i].param.color_mode
        );
        usleep(50000);
        if (i == 0)
            main_plane_id = plane_param[i].par
am.plane_id;
    }

    // set color mode for background plane
    plane_update_pa.plane_id = plane_param[0].
param.plane_id;
    plane_update_pa.param = PLANE_PARAM_COLOR_
MODE;
    plane_update_pa.val.color_mode = LCD_COLOR
_YUV422;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE
_PARAM, &plane_update_pa);
    if (ret < 0)
```

```
        return -1;

        // set data mode for background plane
        plane_update_pa.plane_id = plane_param[0].param.plane_id;
        plane_update_pa.param = PLANE_PARAM_DATA_MODE;
        plane_update_pa.val.data_mode = LCD_PROGRESSIVE;
        ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PARAM,
                    &plane_update_pa);
        if (ret < 0)
            return -1;

        plane_update_pa.param = PLANE_PARAM_APPLY;
        ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PARAM,
                    &plane_update_pa);
        if (ret < 0)
            return -1;

        return 0;
}

int do_disp_endup()
{
    if (!open_flag) {
        printf("Multi close\n");
        return -1;
    }

    if (disp_fd > 0)
        close(disp_fd);

    disp_fd = 0;
    open_flag = 0;

    return 0;
```

```
}

int open_pb_files(int ch_num, int index)
{
    int strh_count;
    char tmp_str[64];

    sprintf(tmp_str, "CH%d_video_%d.avi", ch_num,
index);

    pb_file[ch_num] = GMAVIOpen(tmp_str, GMAVI_FILEMODE_READ, 0);
    if(!pb_file[ch_num]) {
        printf("Open [%s] failed!\n", tmp_str);
        return -1;
    }
    GMAVIGetAviMainHeader(pb_file[ch_num], &pb_main_header[ch_num]);
    GMAVIGetStreamHeaderNum(pb_file[ch_num], &strh_count);
    GMAVIGetStreamHeader(pb_file[ch_num], 1/*get stream1*/, &pb_stream_header[ch_num], &pb_stream_format[ch_num], &stream_id_pkt[ch_num]);
    printf("dec type:<%c%c%c%c>\n", pb_stream_header[ch_num].fccHandler[0]
                                                , pb_stream_header[ch_num].fccHandler[1]
                                                , pb_stream_header[ch_num].fccHandler[2]
                                                , pb_stream_header[ch_num].fccHandler[3]);

    return 0;
}

void close_pb_files(int ch_num)
{
```

```
    if(pb_file[ch_num]) {
        GMAVICClose(pb_file[ch_num]);
        pb_file[ch_num]=NULL;
    }
}

int do_pb_init(int ch_num)
{
    int ret;
    dvr_dec_channel_param     ch_param;

    memset(&ch_param, 0x0, sizeof(dvr_dec_channel_
param));

    ret = open_pb_files(ch_num, 0);
    if(ret < 0)
        goto DO_PB_INIT_FAILED;

    dec_fd[ch_num] = open("/dev/dvr_dec", O_RDWR);
    if(dec_fd[ch_num] < 0) {
        perror("Open [/dev/dvr_dec] failed:");
        goto DO_PB_INIT_FAILED;
    }

    switch(*(int *)pb_stream_header[ch_num].fccHan
dler) {
        case GMAVI_TYPE_H264:
            ch_param.dec_type = ENC_TYPE_H264;

            break;
        case GMAVI_TYPE_MPEG4:
            ch_param.dec_type = ENC_TYPE_MPEG;

            break;
        case GMAVI_TYPE_MJPEG:
            ch_param.dec_type = ENC_TYPE_MJPEG
;
```

```
        break;
    default: printf("%s:%d <dec_type err. %d>\n", __FUNCTION__, __LINE__, ch_param.dec_type); return -1;
}

ch_param.channel = ch_num;
ch_param.is_use_scaler = 1;
ch_param.dec_param.output_type = DEC_OUTPUT_C0LOR_YUV422;
ch_param.scl_param.src_fmt = SCALE_YUV422;
ch_param.scl_param.dst_fmt = SCALE_YUV422;
ch_param.scl_param.scale_mode = SCALE_LINEAR;
ch_param.scl_param.is_dither = 0;
ch_param.scl_param.is_correction = 0;
ch_param.scl_param.is_album = 1;
ch_param.scl_param.des_level = 0;

ret = ioctl(dec_fd[ch_num], DVR_DEC_SET_CHANNEL_PARAM, &ch_param);
if(ret < 0) {
    perror("Decoder DVR_DEC_SET_CHANNEL_PARAM failed:");
    goto DO_PB_INIT_FAILED;
}

ret = ioctl(dec_fd[ch_num], DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE, &dec_buf_size[ch_num]);
if(ret < 0) {
    perror("Decoder DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE failed:");
    goto DO_PB_INIT_FAILED;
}

pbbs_buf[ch_num] = (unsigned char*) mmap(NULL,
dec_buf_size[ch_num], PROT_READ|PROT_WRITE, MAP_SHARED, dec_fd[ch_num], 0);
```

```

    if(pbbs_buf[ch_num] == MAP_FAILED) {
        perror("Dec mmap failed");
        goto DO_PB_INIT_FAILED;
    }

    return 0;

DO_PB_INIT_FAILED:
    if(dec_fd[ch_num]) {
        if(pbbs_buf[ch_num]){
            munmap((void*)pbbs_buf[ch_num], dec_buf_size[ch_num]);
            pbbs_buf[ch_num] = NULL;
        }
        close(dec_fd[ch_num]);
        dec_fd[ch_num] = 0;
    }

    if(pb_file[ch_num]) {
        GMAVIClose(pb_file[ch_num]);
        pb_file[ch_num]=NULL;
    }
    return -1;
}

#define MAX_RECORD 4096
void *packet_reader(void *arg)
{
    int ret, ch_num;
    int pb_idx = 0, pb_total = 0;
    int pb_offset[MAX_RECORD] = {0}; //backup I frame offset, must deal with overflow and multi channel by customer
    int reverse;
    dvr_enc_queue_get data;
    unsigned char *buf;
    int buf_size = 0,intra = 0, backup_data = 0;

```

```
    unsigned int i = 0;
    int file_idx = 1;
    dvr_dec_control dec_ctrl;
    FuncTag tag;

    memset(&dec_ctrl, 0x0, sizeof(dvr_dec_control));
}

ch_num = (int)arg;

// prepare to select(or poll)
dec_fds[ch_num].fd = dec_fd[ch_num];
dec_fds[ch_num].events = POLLIN;
dec_fds[ch_num].revents = 0;

while(1)
{
    ret =poll(&dec_fds[ch_num], 1, 2000);

    pthread_mutex_lock(&dec_lock[ch_num]);

    if(flag_exit_reader_loop[ch_num]){
        pthread_mutex_unlock(&dec_lock[ch_num])
    }
    break;
}

ret = ioctl(dec_fd[ch_num], DVR_DEC_QUEUE_
GET, &data);
backup_data = data.bs.length;
if(ret < 0) {
    pthread_mutex_unlock(&dec_lock[ch_num])
};

printf("buffer is not ready...\n");
usleep(10000);
```

```

        continue;
    }

    buf = pbbs_buf[ch_num] + data.bs.offset;

    // read bs(start)
    buf_size = backup_data;
    reverse=0;
    ret = GMAVIGetStreamDataAndIndex(pb_file[ch_num],
                                      &stream_id_pkt[ch_num],
                                      buf, &buf_size, &intra, NULL,
                                      0, i, reverse, &pb_offset[pb_idx]);
    if((intra == 1) && (pb_idx < MAX_RECORD))
        pb_total=(pb_idx++);

    if(ret == GMSTS_END_OF_DATA){
        printf("CH(%d,%d) - End of Playback!\n"
               , ch_num, file_idx);

OPEN_PB_FILE AGAIN:
    close_pb_files(ch_num);
    ret = open_pb_files(ch_num, file_idx++)
);
    if(ret < 0) {
        file_idx=0;
        goto OPEN_PB_FILE AGAIN;
    }
    GMAVISeek(pb_file[ch_num], GMAVI_SEEK_TO_BEGINNING, NULL);

    //return job
    data.bs.length = 0;
    ret = ioctl(dec_fd[ch_num], DVR_DEC_QUEUE_PUT, &data);
    if(ret<0)
        printf("put failed when EOF...\\n")
;

```

```

        dec_ctrl.command = DEC_UPDATE;
        dec_ctrl.src_param.dim.width = pb_main
_header[ch_num].dwWidth;
        dec_ctrl.src_param.dim.height = pb_main
_header[ch_num].dwHeight;
        dec_ctrl.src_param.win.x = 0;
        dec_ctrl.src_param.win.y = 0;
        dec_ctrl.src_param.win.width = pb_main
_header[ch_num].dwWidth;
        dec_ctrl.src_param.win.height = pb_main
_header[ch_num].dwHeight;
        dec_ctrl.src_param.bs_rate = (int) (10
00000/(pb_main_header[ch_num].dwMicroSecPerFrame))
;
        dec_ctrl.dst_param.plane_id = GMVAL_D0
_NOT_CARE;
        dec_ctrl.dst_param.win.x = GMVAL_D0_NO
T_CARE;
        dec_ctrl.dst_param.win.y = GMVAL_D0_NO
T_CARE;
        dec_ctrl.dst_param.win.width = GMVAL_D
0_NOT_CARE;
        dec_ctrl.dst_param.win.height = GMVAL_
DO_NOT_CARE;
        dec_ctrl.dst_param.is_display = GMVAL_
DO_NOT_CARE;
        dec_ctrl.dst_param.display_rate = GMVA
L_DO_NOT_CARE;

        ret = ioctl(dec_fd[ch_num], DVR_DEC_CO
NTROL, &dec_ctrl);
        if(ret < 0)
            printf("can't update playback para
meters!");
FN_RESET_TAG(&tag);

```

```
        FN_SET_PB_CH(&tag, ch_num);
        ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
        pthread_mutex_unlock(&dec_lock[ch_num])
    );
    buf_size = 0;
    continue;
}
else if(ret < 0) {
    printf("CH(%d) - read error!\n", ch_num);
}
data.bs.length = buf_size;
// read bs(end)
ret = ioctl(dec_fd[ch_num], DVR_DEC_QUEUE_PUT, &data);
if(ret < 0)
    printf("put failed...\n");

pthread_mutex_unlock(&dec_lock[ch_num]);
}
}

int do_pb_start(int ch_num)
{
    int ret;
    dvr_dec_control dec_ctrl;
    FuncTag tag;

    memset(&dec_ctrl, 0x0, sizeof(dvr_dec_control));
}

if(!pb_file[ch_num]) {
    printf("Open file failed!\n");
    return -1;
}
```

```

    flag_exit_reader_loop[ch_num] = FALSE;

    ret = pthread_create(&thr_id_packet_reader[ch_num], NULL, (void *)packet_reader, (void*)ch_num);
    if(ret < 0) {
        perror("create thread[packet_reader] failed");
        return -1;
    }

    dec_ctrl.command = DEC_START;
    dec_ctrl.src_param.dim.width = pb_main_header[ch_num].dwWidth;
    dec_ctrl.src_param.dim.height = pb_main_header[ch_num].dwHeight;
    dec_ctrl.src_param.win.x = 0; //ROI, after decoder, or the input to scalar
    dec_ctrl.src_param.win.y = 0; //ROI, after decoder, or the input to scalar
    dec_ctrl.src_param.win.width = pb_main_header[ch_num].dwWidth; //ROI, after decoder
    dec_ctrl.src_param.win.height = pb_main_header[ch_num].dwHeight; //ROI, after decoder
    dec_ctrl.src_param.bs_rate=(int)(1000000/(pb_main_header[ch_num].dwMicroSecPerFrame));

    if(is_st_vga) {
        dec_ctrl.dst_param.win.x = VGA_4CH[ch_num].x;
        dec_ctrl.dst_param.win.y = VGA_4CH[ch_num].y;
    }
    else {
        dec_ctrl.dst_param.win.x = D1_4CH[ch_num].x;
        //final position of screen
        dec_ctrl.dst_param.win.y = D1_4CH[ch_num].y
    }
}

```

```
;      //final position of screen
    }
    dec_ctrl.dst_param.win.width = VGA_4CH[ch_num].width;      //final width in screen
    dec_ctrl.dst_param.win.height = VGA_4CH[ch_num].height;    //final height in screen
    dec_ctrl.dst_param.plane_id = main_plane_id;
    dec_ctrl.dst_param.is_display = TRUE;
    dec_ctrl.dst_param.display_rate = (is_NTSC)? 3
0:25;

    ret = ioctl(dec_fd[ch_num], DVR_DEC_CONTROL, &
dec_ctrl);
    if(ret < 0)
        return -1;

    FN_RESET_TAG(&tag);
    FN_SET_PB_CH(&tag, ch_num);
    ret = ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    if(ret < 0)
        return -1;

    pthread_mutex_init(&dec_lock[ch_num], NULL);

    menu_func = FUNCTION_NORMAL;

    return 0;
}

int do_pb_stop(int ch_num)
{
    int ret;
    dvr_dec_control     dec_ctrl;
    FuncTag  tag;

    memset(&dec_ctrl, 0x0, sizeof(dvr_dec_control))
};
```

```
menu_func = FUNCTION_NULL;
pthread_mutex_lock(&dec_lock[ch_num]);

    dec_ctrl.command = DEC_STOP;
    if((ret = ioctl(dec_fd[ch_num], DVR_DEC_CONTROL
, &dec_ctrl)) < 0)
        goto pb_exit;

    FN_RESET_TAG(&tag);
    FN_SET_PB_CH(&tag, ch_num);
    if((ret = ioctl(dvr_fd, DVR_COMMON_APPLY, &tag
)) < 0)
        goto pb_exit;

pb_exit:
    flag_exit_reader_loop[ch_num] = TRUE;
    pthread_mutex_unlock(&dec_lock[ch_num]);

    return 0;
}

int do_pb_exit(int ch_num)
{
    if(dec_fd[ch_num]){
        if(pbbs_buf[ch_num]){
            munmap((void*)pbbs_buf[ch_num], dec_buf_size[ch_num]);
            pbbs_buf[ch_num] = NULL;
        }
        close(dec_fd[ch_num]);
        dec_fd[ch_num] = 0;
    }
    close_pb_files(ch_num);
    return 0;
}

int main(void)
```

```
{  
    char key;  
    int ret;  
  
    //open dvr_common  
    dvr_fd = open("/dev/dvr_common", O_RDWR);  
  
    do_disp_startup();  
  
    while(1)  
    {  
        key = getch();  
        if (key == 'q' || key == 'Q')  
            break;  
        switch(key)  
        {  
            case 'a':  
                ret=do_pb_init(0);  
                break;  
  
            case 'b':  
                ret=do_pb_start(0);  
  
                break;  
            case 'c':  
                ret=do_pb_stop(0);  
                break;  
            case 'd':  
                ret=do_pb_exit(0);  
                break;  
            case 'm':  
                ret=do_pb_init(1);  
                break;  
  
            case 'n':  
                ret=do_pb_start(1);  
        }  
    }  
}
```

```
        break;
    case 'o':
        ret=do_pb_stop(1);
        break;
    case 'p':
        ret=do_pb_exit(1);
        break;
    }
}

do_disp_endup();

//close dvr_common
close(dvr_fd);
return 0;
}
```

Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

capture_raw.c

```
/**  
 * this sample code snapshot a video yuv422 frame,  
 and record for 10 seconds.  
 * trigger snapshot yuv422 per second.  
 * Please use gmdvr_mem_3_3_3_3.cfg for the buffer  
 config.  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int enc_fd = 0;  
  
//test record  
unsigned char *bs_buf;  
HANDLE rec_file;  
int enc_buf_size;  
struct pollfd rec_fds;  
char file_name[128];
```

```

int sub1_bs_buf_offset;

dvr_enc_channel_param    ch_param;
ReproduceBitStream  sub_ch_param;
EncParam_Ext3 enc_param_ext = {0};
dvr_enc_control   enc_ctrl;
FuncTag  tag;

dvr_enc_channel_param    user_rec_ch_setting =
{
{
    3,                                /* channel number */
    ENC_TYPE_FROM_CAPTURE,
    // {1280, 720},           /* channel 0 */
    // {640, 480},            /* channel 1 */
    // {320, 240},            /* channel 2 */
    {160, 112},             /* channel 3 */
    LVFRAME_EVEN_ODD,
    LVFRAME_FRAME_MODE,
    DMAORDER_PACKET,
    CAPSCALER_NOT_KEEP_RATIO,
    MCP_VIDEO_NTSC,
    CAPCOLOR_YUV422,
    { FALSE, FALSE, GM3DI_FIELD }

},
{
    DVR_ENC_EBST_ENABLE,
    0,
    ENC_TYPE_H264,
    FALSE,
    DVR_ENC_EBST_DISABLE,
    // {1280, 720},           /* channel 0 */
    // {640, 480},            /* channel 1 */
    // {320, 240},            /* channel 2 */
    {160, 112},             /* channel 3 */
    {ENC_INPUT_H2642D, 30, 200*1000, 30, 25,
51, 1 , FALSE, {0, 0, 160, 112}},

}

```

```

        {SCALE_YUV422, SCALE_YUV422, SCALE_LINEAR,
FALSE, FALSE, TRUE, 0 },
        {JCS_yuv420, 0, JENC_INPUT_MP42D, 70}

    }

};

ReproduceBitStream    user_rec_sub_ch_setting =
{
    DVR_ENC_EBST_ENABLE, //enabled
    1, // sub1-bitstream
    ENC_TYPE_YUV422, //enc_type, 0: ENC_TYPE_H264,
    1:ENC_TYPE_MPEG, 2:ENC_TYPE_MJPEG, 3:ENC_TYPE_YUV
    422
    FALSE, // is_blocked
    DVR_ENC_EBST_DISABLE, // en_snapshot,
//    {640, 480}, /* channel 1 */
//    {320, 240}, /* channel 2 */
    {160, 112}, /* channel 3 */
    {ENC_INPUT_H2642D, 30, 262144, 15, 25, 51, 1
, FALSE, {0, 0, 160, 112}}, //EncParam
    {SCALE_YUV422, SCALE_YUV422, SCALE_LINEAR, FAL
SE, FALSE, TRUE, 0 }, //ScalerParam
    {JCS_yuv420, 0, JENC_INPUT_MP42D, 70 } //snap
shot_param
};

void do_record_start(void)
{
    memcpy(&ch_param, &user_rec_ch_setting, sizeof
(ch_param)); //main-bitstream
    ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_
    ADD_VAL(sizeof(enc_param_ext));
    ch_param.main_bs.enc.pext_data = &enc_param_ex
t;

    enc_param_ext.feature_enable = 0;
}

```

```
    ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_param);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE
, &enc_buf_size);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1
_BS_OFFSET, &sub1_bs_buf_offset);

    bs_buf = (unsigned char*) mmap(NULL, enc_buf_size,
PROT_READ|PROT_WRITE,
                                MAP_SHARED, enc_fd, 0);

    memcpy(&sub_ch_param, &user_rec_sub_ch_setting
, sizeof(sub_ch_param)); //sub1-raw
    ioctl(enc_fd, DVR_ENC_SET_SUB_BS_PARAM, &sub_ch_param);

//record start
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
    enc_ctrl.command = ENC_START;
    enc_ctrl.stream = 0;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    enc_ctrl.command = ENC_START;
    enc_ctrl.stream = sub_ch_param.out_bs;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

// set function tag parameter to dvr graph level
el
    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_param.src.channel);
    FN_SET_SUB1_REC_CH(&tag, ch_param.src.channel)
;
```

```
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
}

void do_record_stop(void)
{
    //record stop
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
};

    enc_ctrl.stream = sub_ch_param.out_bs;
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    enc_ctrl.stream = 0;
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_param.src.channel);
    FN_SET_SUB1_REC_CH(&tag, ch_param.src.channel)
;
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    munmap((void*)bs_buf, enc_buf_size);
}

/***
 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num;
    dvr_enc_queue_get data;
    unsigned char *buf;
    int buf_size;
    struct timeval t1,t2;
    char tmp_str[128];
```

```
    int raw_no = 1;
    int flag_raw = 0;

    dvr_fd = open("/dev/dvr_common", O_RDWR);      //  

open_dvr_common
    enc_fd = open("/dev/dvr_enc", O_RDWR);        //  

open_dvr_encode
    ch_num = user_rec_ch_setting.src.channel;

    do_record_start();

    sprintf(file_name, "CH%d_video_%d", ch_num, 0)
;
    sprintf(tmp_str, "%s.h264", file_name);

    rec_file = fopen ( tmp_str , "wb+" );
    gettimeofday(&t1, NULL);

    while(1) {
        // prepare to select(or poll)
        rec_fds.fd = enc_fd;
        rec_fds.revents = 0;
        rec_fds.events = (POLLIN_MAIN_BS | POLLIN_
SUB1_BS);

        poll(&rec_fds, 1, 500);

        if (rec_fds.revents & POLLIN_SUB1_BS) {

            ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET_
SUB1_BS, &data);
            if (ret >= 0) {
                FILE * fd;

                buf = bs_buf + sub1_bs_buf_offset
+ data.bs.offset;
                buf_size = data.bs.length;
```

```

                printf("<buf_size=%d>\n", buf_size
);
                sprintf(tmp_str, "CH%d_raw%03d.yuv"
, ch_num, ++flag_raw);
                fd = fopen ( tmp_str , "wb+" );
                fwrite(buf , 1 , buf_size , fd);

                fclose(fd);
                printf("capture raw: output file %
s\n", tmp_str);
                ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
            }
        }

        if (!(rec_fds.revents & POLLIN_MAIN_BS))
            continue;

        // get dataout buffer
        ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET, &da
ta);
        if(ret < 0)
            continue;

        buf = bs_buf + data.bs.offset;
        buf_size = data.bs.length;

        fwrite (buf , 1 , buf_size , rec_file);
        fflush(rec_file);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data);

        gettimeofday(&t2, NULL);

        if ((t2.tv_sec - t1.tv_sec) == 1) {      /
< trigger capture yuv422 per second,
        memset(&enc_ctrl, 0x0, sizeof(dvr_enc_

```

```

control));
        enc_ctrl.command = ENC_RAW;
        enc_ctrl.output.count = (int *)raw_no;
//< do capture yuv422 1 times.
        enc_ctrl.stream = sub_ch_param.out_bs;
//< sub_bitstream number
        ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ct
rl);
        t1.tv_sec = t2.tv_sec;
    }
    if (flag_raw >= 10) {           //< record for
10 seconds. then finish record.
        fclose(rec_file);
        break;
    }
}

do_record_stop();

printf("-----\n");
;
printf(" Record finish\n");
printf("-----\n");
;

close(enc_fd);          //close_dvr_encode
close(dvr_fd);          //close_dvr_common
return 0;
}

```

Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

liveview.c

```
/**  
 * this sample code implement liveview function, a  
nd liveview for 20 seconds.  
*  
*/  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_disp_api.h"  
  
int dvr_fd = 0;  
int disp_fd = 0;  
int main_disp_no = 0;  
int main_plane_id = 0;  
int open_flag = 0;  
  
int main(int argc, char *argv[])  
{  
    int fd, i ;  
    char isp_file[]{"dev/isp"};  
    dvr_disp_disp_param disp_param;  
    dvr_disp_update_disp_param disp_update_param;  
    dvr_disp_plane_param plane_param[3];  
    dvr_disp_update_plane_param plane_update_pa;
```

```
dvr_disp_control dsp_ctl;
dvr_disp_control disp_ctrl;
DIM dim;
RECT win;
FuncTag tag;
struct timeval t1,t2;
gettimeofday(&t1, NULL);
int ch_num = 0;

fd = open(isp_file, O_RDWR);
dvr_fd = open("/dev/dvr_common", O_RDWR);
//do_disp_startup
open_flag = 1;
disp_fd = open("/dev/dvr_disp", O_RDWR);
memset(&disp_param, 0x0, sizeof(dvr_disp_disp_param));
memset(&disp_update_param, 0x0, sizeof(dvr_disp_update_disp_param));
memset(plane_param, 0x0, sizeof(dvr_disp_plane_param) * 3);
memset(&plane_update_pa, 0x0, sizeof(dvr_disp_update_plane_param));
memset(&dsp_ctl, 0x0, sizeof(dvr_disp_control));
);

main_disp_no = 0;
// query LCD1 information
disp_param.disp_num = main_disp_no;
ioctl(disp_fd, DVR_DISP_GET_DISP_PARAM, &disp_param);

usleep(100000);

disp_update_param.disp_num = main_disp_no;
disp_update_param.param = DISP_PARAM_PLANE_COMBINATION;
disp_update_param.val.plane_comb = BG_ONLY; //
```

```
BG_AND_2PLANE;
    ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM, &disp_update_param);

    disp_update_param.disp_num = main_disp_no;
    disp_update_param.param = DISP_PARAM_OUTPUT_SYSTEM;
    disp_update_param.val.output_system = MCP_VIDEO_VGA;
    disp_update_param.val.display_rate = 30;
    ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM, &disp_update_param);

    disp_update_param.param = DISP_PARAM_APPLY;
    ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM, &disp_update_param);

    // query 3 planes information
    for (i = 0; i < 3; i++) {
        plane_param[i].disp_num = main_disp_no;
        plane_param[i].plane_num = i;
        ioctl(disp_fd, DVR_DISP_GET_PLANE_PARAM, &plane_param[i]);

        usleep(50000);
        if (i == 0)
            main_plane_id = plane_param[i].param.plane_id;
    }

    // set color mode for background plane
    plane_update_pa.plane_id = plane_param[0].param.plane_id;
    plane_update_pa.param = PLANE_PARAM_COLOR_MODE;
    plane_update_pa.val.color_mode = LCD_COLOR_YUV
422;
```

```
    ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PARAM, &plane_update_pa);

    // set data mode for background plane
    plane_update_pa.plane_id = plane_param[0].param
    .plane_id;
    plane_update_pa.param = PLANE_PARAM_DATA_MODE;
    plane_update_pa.val.data_mode = LCD_PROGRESSIV
E;
    ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PARAM, &plane_update_pa);

    plane_update_pa.param = PLANE_PARAM_APPLY;
    ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PARAM, &plane_update_pa);
    //do_liveview_t1
    dim.width = 720;
    dim.height = 480;
    win.x = 0;
    win.y = 0;
    win.width = 640;
    win.height = 360;
    //setup_lv_channel
    memset(&disp_ctrl, 0x0, sizeof(dvr_disp_control));
    disp_ctrl.type = DISP_TYPE_LIVEVIEW;
    disp_ctrl.channel = 0;
    disp_ctrl.command = DISP_START;
    disp_ctrl.src_param.lv.cap_path = ch_num;
    disp_ctrl.src_param.lv.di_mode = LVFRAME_WEAVE
D_TWO_FIELDS;
    disp_ctrl.src_param.lv.mode = LVFRAME_FRAME_M0
DE;
    disp_ctrl.src_param.lv.vp_param.is_3DI = FALSE
;
    disp_ctrl.src_param.lv.vp_param.is_denoise = F
ALSE;
```

```

        disp_ctrl.src_param.lv.vp_param.denoise_mode =
    0;
        disp_ctrl.src_param.lv.dma_order = DMAORDER_PACKET;
        disp_ctrl.src_param.lv.scale_indep = CAPSCALER_KEEP_RATIO;
        disp_ctrl.src_param.lv.input_system = MCP_VIDEO_NTSC;
        disp_ctrl.src_param.lv.cap_rate = 30 ;
        disp_ctrl.src_param.lv.color_mode = CAPCOLOR_YUV422;
        disp_ctrl.dst_param.lv.plane_id = main_plane_id;
        disp_ctrl.src_param.lv.is_use_scaler = FALSE;

        disp_ctrl.dst_param.lv.win.x = win.x;
        disp_ctrl.dst_param.lv.win.y = win.y;
        disp_ctrl.dst_param.lv.win.width = win.width;
        disp_ctrl.dst_param.lv.win.height = win.height
;

        ioctl(disp_fd, DVR_DISP_CONTROL, &disp_ctrl);

//run_lv_command
FN_RESET_TAG(&tag);
for (i = 0; i < 16; i++) {
    FN_SET_LV_CH(&tag, i);
    FN_SET_PB_CH(&tag, i);
}
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);

gettimeofday(&t2, NULL);
// set 20 seconds to do liveview
while(t2.tv_sec - t1.tv_sec < 20){
    gettimeofday(&t2, NULL);
}

```

```
//do_liveview_close1ch
memset(&disp_ctrl, 0x0, sizeof(dvr_disp_control));
);
disp_ctrl.type = DISP_TYPE_LIVEVIEW;
disp_ctrl.channel = 0;
disp_ctrl.command = DISP_STOP;
ioctl(disp_fd, DVR_DISP_CONTROL, &disp_ctrl);

//run_lv_command
FN_RESET_TAG(&tag);
for (i = 0; i < 16; i++) {
    FN_SET_LV_CH(&tag, i);
    FN_SET_PB_CH(&tag, i);
}
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);

close(disp_fd);
disp_fd = 0;
open_flag = 0;

close(dvr_fd);
printf("finish\n");
return 0;
}
```

Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

main-bitstream-record.c

```
/**  
 * this sample code implement main bitstream record function, and record for 20 seconds.  
 * (1) for H264 record, file format please indicate xxx.h264  
 *      for MPEG record, file format please indicate xxx.m4v  
 *      for MOTION JPEG record, file format please indicate xxx.jpg  
 * (2) for H264 record, max_quant < 51, 1 < min_quant  
 *      for MPEG record, max_quant < 31, 1 < min_quant  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int enc_fd = 0;
```

```
//test record
unsigned char *bs_buf;
HANDLE rec_file;
int avi_str_id;
int enc_buf_size;
struct pollfd rec_fds;
char file_name[128];

/**
 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num = 0;
    dvr_enc_channel_param    ch_param;
    EncParam_Ext3 enc_param_ext = {0};
    dvr_enc_control    enc_ctrl;
    dvr_enc_queue_get    data;
    unsigned char *buf;
    int buf_size;
    FuncTag tag;
    struct timeval t1,t2;
    char tmp_str[128];

    dvr_fd = open("/dev/dvr_common", O_RDWR); // //open_dvr_common

    enc_fd = open("/dev/dvr_enc", O_RDWR); // //open_dvr_encode

    //set dvr encode source parameter
    ch_param.src.input_system = MCP_VIDEO0_NTSC;
    ch_param.src.channel = ch_num;
    ch_param.src.enc_src_type = ENC_TYPE_FROM_CAPTURE;
```

```

ch_param.src.dim.width = 1280;
ch_param.src.dim.height = 720;

ch_param.src.di_mode = LVFRAME_EVEN_ODD;
ch_param.src.mode = LVFRAME_FRAME_MODE;
ch_param.src.dma_order = DMAORDER_PACKET;
ch_param.src.scale_indep = CAPSCALER_NOT_KEEP_
RATIO;
ch_param.src.input_system = MCP_VIDEO_NTSC;
ch_param.src.color_mode = CAPCOLOR_YUV422;

ch_param.src.vp_param.is_3DI = FALSE;
ch_param.src.vp_param.is_denoise = FALSE;
ch_param.src.vp_param.denoise_mode = GM3DI_FIE
LD;

//set dvr encode main bitstream parameter
ch_param.main_bs.enabled = DVR_ENC_EBST_ENABLE
;
ch_param.main_bs.out_bs = 0;
ch_param.main_bs.enc_type = ENC_TYPE_H264;
ch_param.main_bs.is_blocked = FALSE;
ch_param.main_bs.en_snapshot = DVR_ENC_EBST_DI
SABLE;

ch_param.main_bs.dim.width = 1280;
ch_param.main_bs.dim.height = 720;

//set main bitstream encode parameter
ch_param.main_bs.enc.input_type = ENC_INPUT_H2
642D;
ch_param.main_bs.enc.frame_rate = 30;
ch_param.main_bs.enc.bit_rate = 1048576;
ch_param.main_bs.enc.ip_interval = 30;
ch_param.main_bs.enc.init_quant = 25;
ch_param.main_bs.enc.max_quant = 51;

```

```

ch_param.main_bs.enc.min_quant = 1;
ch_param.main_bs.enc.is_use_ROI = FALSE;
ch_param.main_bs.enc.ROI_win.x = 0;
ch_param.main_bs.enc.ROI_win.y = 0;
ch_param.main_bs.enc.ROI_win.width = 352;
ch_param.main_bs.enc.ROI_win.height = 240;

//set main bitstream scalar parameter
ch_param.main_bs.scl.src_fmt = SCALE_YUV422;
ch_param.main_bs.scl.dst_fmt = SCALE_YUV422;
ch_param.main_bs.scl.scale_mode = SCALE_LINEAR
;

ch_param.main_bs.scl.is_dither = FALSE;
ch_param.main_bs.scl.is_correction = FALSE;
ch_param.main_bs.scl.is_album = TRUE;
ch_param.main_bs.scl.des_level = 0;

//set main bitstream snapshot parameter
ch_param.main_bs.snap.sample = JCS_yuv420;
ch_param.main_bs.snap.RestartInterval = 0;
ch_param.main_bs.snap.u82D = JENC_INPUT_MP42D;

ch_param.main_bs.snap.quality = 70;

//associate the ext. structure

ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_
ADD_VAL(sizeof(enc_param_ext));
ch_param.main_bs.enc.pext_data = &enc_param_ex
t;

enc_param_ext.feature_enable = 0;           //CBR

ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_p
aram);

ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE

```

```
, &enc_buf_size);

    bs_buf = (unsigned char*) mmap(NULL, enc_buf_size, PROT_READ|PROT_WRITE,
                                  MAP_SHARED, enc_fd, 0);
    ///////////////////////////////////////////////////////////////////
    ///////////////////////////////////////////////////////////////////
    //record start
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
    enc_ctrl.command = ENC_START;
    enc_ctrl.stream = 0;
    ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
}

// set function tag parameter to dvr graph level
FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
///////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////
sprintf(file_name, "CH%d_video_%d", 0, 0);

sprintf(tmp_str, "%s.h264", file_name);

rec_file = fopen ( tmp_str , "wb+" );
gettimeofday(&t1, NULL);

while(1) {
    // prepare to select(or poll)
    rec_fds.fd = enc_fd;
    rec_fds.revents = 0;
    rec_fds.events = POLLIN_MAIN_BS;

    poll(&rec_fds, 1, 500);
```

```
        if (!(rec_fds.revents & POLLIN_MAIN_BS))
            continue;

        // get dataout buffer
        ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET,
&data);
        if(ret < 0)
            continue;

        buf = bs_buf + data.bs.offset;
        buf_size = data.bs.length;

        ret = fwrite (buf , 1 , buf_size , rec
_file);
        fflush(rec_file);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data
);

        gettimeofday(&t2, NULL);

        if ((t2.tv_sec - t1.tv_sec) == 20) {
            //<record for 20 seconds. then finish record.

            fclose(rec_file);
            break;
        }
    }

    //record stop
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control)
);
    enc_ctrl.stream = 0;
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
```

```
FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
munmap((void*)bs_buf, enc_buf_size);

printf("-----\n");
printf(" Record finish\n");
printf("-----\n");
;

close(enc_fd);           //close_dvr_encode
close(dvr_fd);           //close_dvr_common

return 0;
}
```

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

mjpeg-record.c

```
/**  
 * this sample code implement main bitstream Motion  
 * JPEG record function, and record for 20 seconds.  
 * (1) for H264 record, file format please indicate xxx.h264  
 *      for MPEG record, file format please indicate xxx.m4v  
 *      for MOTION JPEG record, file format please indicate xxx.jpg  
 * (2) for H264 record, max_quant < 51, 1 < min_q  
 *      for MPEG record, max_quant < 31, 1 < min_q  
 *      quant  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int enc_fd = 0;
```

```

//test record
unsigned char *bs_buf;
HANDLE rec_file;
int avi_str_id;
int enc_buf_size;
struct pollfd rec_fds;
char file_name[128];

/**
 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num = 0;
    dvr_enc_channel_param    ch_param;
    EncParam_Ext3 enc_param_ext = {0};
    dvr_enc_control    enc_ctrl;
    dvr_enc_queue_get    data;
    unsigned char *buf;
    int buf_size;
    FuncTag tag;
    struct timeval t1,t2;
    char tmp_str[128];

    dvr_fd = open( "/dev/dvr_common", 0_RDWR);      // 
open_dvr_common

    enc_fd = open( "/dev/dvr_enc", 0_RDWR);          // 
open_dvr_encode

    //set dvr encode source parameter
    ch_param.src.input_system = MCP_VIDEO0_NTSC;
    ch_param.src.channel = ch_num;
    ch_param.src.enc_src_type = ENC_TYPE_FROM_CAPT
URE;

```

```
ch_param.src.dim.width = 1280;
ch_param.src.dim.height = 720;

ch_param.src.di_mode = LVFRAME_EVEN_ODD;
ch_param.src.mode = LVFRAME_FRAME_MODE;
ch_param.src.dma_order = DMAORDER_PACKET;
ch_param.src.scale_indep = CAPSCALER_NOT_KEEP_
RATIO;
ch_param.src.input_system = MCP_VIDEO_NTSC;
ch_param.src.color_mode = CAPCOLOR_YUV422;

ch_param.src.vp_param.is_3DI = FALSE;
ch_param.src.vp_param.is_denoise = FALSE;
ch_param.src.vp_param.denoise_mode = GM3DI_FIE
LD;

//set dvr encode main bitstream parameter
ch_param.main_bs.enabled = DVR_ENC_EBST_ENABLE
;
ch_param.main_bs.out_bs = 0;
ch_param.main_bs.enc_type = ENC_TYPE_MJPEG;
ch_param.main_bs.is_blocked = FALSE;
ch_param.main_bs.en_snapshot = DVR_ENC_EBST_DI
SABLE;

ch_param.main_bs.dim.width = 1280;
ch_param.main_bs.dim.height = 720;

//set main bitstream encode parameter
ch_param.main_bs.enc.input_type = ENC_INPUT_H2
642D;
ch_param.main_bs.enc.frame_rate = 30;
ch_param.main_bs.enc.is_use_ROI = FALSE;
ch_param.main_bs.enc.ROI_win.x = 0;
ch_param.main_bs.enc.ROI_win.y = 0;
ch_param.main_bs.enc.ROI_win.width = 320;
```

```

ch_param.main_bs.enc.ROI_win.height = 240;

//set main bitstream scalar parameter
ch_param.main_bs.scl.src_fmt = SCALE_YUV422;
ch_param.main_bs.scl.dst_fmt = SCALE_YUV422;
ch_param.main_bs.scl.scale_mode = SCALE_LINEAR
;

ch_param.main_bs.scl.is_dither = FALSE;
ch_param.main_bs.scl.is_correction = FALSE;
ch_param.main_bs.scl.is_album = TRUE;
ch_param.main_bs.scl.des_level = 0;

//set main bitstream snapshot parameter
ch_param.main_bs.snap.sample = JCS_yuv420;
ch_param.main_bs.snap.RestartInterval = 0;
ch_param.main_bs.snap.u82D = JENC_INPUT_MP42D;

ch_param.main_bs.snap.quality = 70;

//associate the ext. structure

ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_
ADD_VAL(sizeof(enc_param_ext));
ch_param.main_bs.enc.pext_data = &enc_param_ext;

enc_param_ext.feature_enable |= DVR_ENC_MJPEG_
FUNCTION;
enc_param_ext.MJ_quality = 50;

ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_param);

ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE
, &enc_buf_size);

bs_buf = (unsigned char*) mmap(NULL, enc_buf_s

```

```
ize, PROT_READ|PROT_WRITE,
                           MAP_SHARED);
ED, enc_fd, 0);
///////////////////////////////
///////////////////
//record start
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
);
enc_ctrl.command = ENC_START;
enc_ctrl.stream = 0;
ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
);

// set function tag parameter to dvr graph level
else
FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
///////////////////////////////
///////////////////
sprintf(file_name, "CH%d_video_%d", 0, 0);

sprintf(tmp_str, "%s.jpg", file_name);

rec_file = fopen ( tmp_str , "wb+" );
gettimeofday(&t1, NULL);

while(1) {
    // prepare to select(or poll)
    rec_fds.fd = enc_fd;
    rec_fds.revents = 0;
    rec_fds.events = POLLIN_MAIN_BS;

    poll(&rec_fds, 1, 500);

    if (!(rec_fds.revents & POLLIN_MAIN_BS))
})
```

```
        continue;

    // get dataout buffer
    ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET,
&data);
    if(ret < 0)
        continue;

    buf = bs_buf + data.bs.offset;
    buf_size = data.bs.length;

    ret = fwrite (buf , 1 , buf_size , rec
_file);
    fflush(rec_file);
    ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data
);

    gettimeofday(&t2, NULL);

    if ((t2.tv_sec - t1.tv_sec) == 20) {
//<record for 20 seconds. then finish record.

        fclose(rec_file);
        break;
    }
}

//record stop
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control)
);
enc_ctrl.stream = 0;
enc_ctrl.command = ENC_STOP;
ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
```

```
    munmap((void*)bs_buf, enc_buf_size);

    printf("-----\n");
;

    printf(" Record finish\n");
    printf("-----\n");
;

close(enc_fd);           //close_dvr_encode
close(dvr_fd);           //close_dvr_common

return 0;
}
```

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

motion-detection-mpeg4.c

```
/**  
 * this sample code implement MPEG4 motion detection function without ISP module, and record for 20 seconds.  
 *  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int enc_fd = 0;  
  
//test record  
unsigned char *bs_buf;  
HANDLE rec_file;  
int avi_str_id;  
int enc_buf_size;  
struct pollfd rec_fds;  
char file_name[128];
```

```
int bs_buf_snap_offset;
int encode_frame_count = 0;

dvr_enc_channel_param    ch_param;
EncParam_Ext3 enc_param_ext = {0};
dvr_enc_control   enc_ctrl;
FuncTag  tag;

//global motion detection structure
#include "fv_motiondet.h"
struct fv_md gfvcmd;
struct md_cfg md_param;
struct md_res active;

dvr_enc_channel_param    user_rec_ch_setting =
{
{
    0,
    ENC_TYPE_FROM_CAPTURE,
    {1280, 720},
    LVFRAME_EVEN_ODD,
    LVFRAME_FRAME_MODE,
    DMAORDER_PACKET,
    CAPSCALER_NOT_KEEP_RATIO,
    MCP_VIDEO_NTSC,
    CAPCOLOR_YUV422,
    { FALSE, FALSE, GM3DI_FIELD }
},
{
    DVR_ENC_EBST_ENABLE,
    0,
    ENC_TYPE_MPEG,
    FALSE,
    DVR_ENC_EBST_ENABLE,
    {1280, 720},
    {ENC_INPUT_H2642D, 30, 1048576, 30, 25, 3
```

```
1, 1 , FALSE, {0, 0, 320, 240}},  
    {SCALE_YUV422, SCALE_YUV422, SCALE_LINEAR,  
FALSE, FALSE, TRUE, 0 },  
    {JCS_yuv420, 0, JENC_INPUT_MP42D, 70}  
}  
};  
  
void do_record_start(int ch_num)  
{  
    memcpy(&ch_param, &user_rec_ch_setting, sizeof  
(ch_param));  
  
    ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_  
ADD_VAL(sizeof(enc_param_ext));  
    ch_param.main_bs.enc.pext_data = &enc_param_ex  
t;  
  
    enc_param_ext.feature_enable = 0;  
  
    ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_p  
aram);  
  
    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE  
, &enc_buf_size);  
  
    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP  
_OFFSET, &bs_buf_snap_offset);  
  
    bs_buf = (unsigned char*) mmap(NULL, enc_buf_s  
ize, PROT_READ|PROT_WRITE,  
                                MAP_SHAR  
ED, enc_fd, 0);  
    //record start  
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control))  
};  
    enc_ctrl.command = ENC_START;  
    enc_ctrl.stream = 0;
```

```

        ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

        // set function tag parameter to dvr graph lev
el
        FN_RESET_TAG(&tag);
        FN_SET_REC_CH(&tag, ch_num);
        ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
}

void do_record_stop(int ch_num)
{
    //record stop
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
    enc_ctrl.stream = 0;
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    munmap((void*)bs_buf, enc_buf_size);
}

void snapshot_trigger(int ch_num)
{
    int snap_no = 1;
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
    enc_ctrl.command = ENC_SNAP;
    enc_ctrl.output.count = (int *)snap_no;           // 
< do snapshot 1 time
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
}

/**
 * @brief main function

```

```

 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num = 0;
    dvr_enc_queue_get    data;
    unsigned char *buf;
    int buf_size;
    struct timeval t1,t2;
    char tmp_str[128];
    int mb_width,mb_height;
    int count;
    int flag_snapshot = 1;

    dvr_fd = open("/dev/dvr_common", O_RDWR);      //  

open_dvr_common

    enc_fd = open("/dev/dvr_enc", O_RDWR);          //  

open_dvr_encode

    do_record_start(ch_num);

    //do motion detection init
    mb_width = (ch_param.src.dim.width + 16 - 1) /  

16;
    mb_height = (ch_param.src.dim.height + 16 - 1)  

/ 16;

    fv_motion_info_init(&md_param, &active, &gfvcmd,  

mb_width, mb_height);

    for (count = 0; count < 1; count++) {
        int i,j;
        md_param.interlace_mode = 0;
        md_param.usesad = 1; //no sad
        md_param.mb_time_th = 4;
        md_param.rolling_prot_th = 160;

```

```

        md_param.alarm_th = 10;
        md_param.sad_offset = 300;

        for (i = 0; i < mb_height; i++)
            for (j = 0; j < mb_width; j++)
                md_param.mb_cell_en[(i*mb_width +
j)] = 1;

        for (i = 0; i < mb_height; i++)
            for (j = 0; j < mb_width; j++)
                md_param.mb_cell_th[(i*mb_width +
j)] = 25;
    }

    sprintf(file_name, "CH%d_video_%d", 0, 0);

    sprintf(tmp_str, "%s.m4v", file_name);

    rec_file = fopen ( tmp_str , "wb+" );
    gettimeofday(&t1, NULL);

    while(1) {
        // prepare to select(or poll)
        rec_fds.fd = enc_fd;
        rec_fds.revents = 0;
        rec_fds.events = (POLLIN_MAIN_BS | POLLIN_
SNAP_BS);

        poll(&rec_fds, 1, 500);

        if (rec_fds.revents & POLLIN_SNAP_BS) {

            ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET_
SNAP, &data);
            if (ret >= 0) {
                FILE * fd;

```

```

                buf = bs_buf + bs_buf_snap_offset
+ data.bs.offset;
                buf_size = data.bs.length;

                -- flag_snapshot;

                sprintf(tmp_str, "CH%d_snapshot%02
d_%03d.jpg", ch_num, flag_snapshot, encode_frame_c
ount);
                fd = fopen ( tmp_str , "wb+" );
                fwrite(buf , 1 , buf_size , fd);

                fclose(fd);

                printf("snapshot: output file %s\n"
, tmp_str);
                ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
            }
        }

        if (!(rec_fds.revents & POLLIN_MAIN_BS))
            continue;

        // get dataout buffer
        ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET, &da
ta);
        if(ret < 0)
            continue;

        encode_frame_count++;
        buf = bs_buf + data.bs.offset;
        buf_size = data.bs.length;

        //do motion detection
        for (count = 0; count < 1; count++) {
            fv_motion_detection((MBK_INFO *)(bs_bu

```

```

f + data.bs.mv_offset), &md_param, &active, &gfvcmd);
                if ((active.active_num >= md_param.alarm_th)) {
                        printf("Motion Alarm @ %s< %d >\n"
, "main_stream", active.active_num);
                        snapshot_trigger(ch_num);
                }
}

fwrite(buf , 1 , buf_size , rec_file);
fflush(rec_file);
ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data);

gettimeofday(&t2, NULL);

        if ((t2.tv_sec - t1.tv_sec) == 20) {
//< record for 20 seconds. then finish record.

                fclose(rec_file);
                break;
        }
}

do_record_stop(ch_num);

printf("-----\n");
printf(" Record finish\n");
printf("-----\n");
;

fv_motion_info_end(&md_param, &active, &gfvcmd);
close(enc_fd);           //close_dvr_encode
close(dvr_fd);           //close_dvr_common

```

```
    return 0;  
}
```

Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

motion-detection.c

```
/**  
 * this sample code implement H264 motion detection function without ISP module, and record for 20 seconds.  
 *  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int enc_fd = 0;  
  
//test record  
unsigned char *bs_buf;  
HANDLE rec_file;  
int avi_str_id;  
int enc_buf_size;  
struct pollfd rec_fds;  
char file_name[128];
```

```
int bs_buf_snap_offset;
int encode_frame_count = 0;

dvr_enc_channel_param    ch_param;
EncParam_Ext3 enc_param_ext = {0};
dvr_enc_control   enc_ctrl;
FuncTag  tag;

//global motion detection structure
#include "favc_motiondet.h"
struct favc_md gfavcmd;
struct md_cfg md_param;
struct md_res active;

dvr_enc_channel_param    user_rec_ch_setting =
{
{
    0,
    ENC_TYPE_FROM_CAPTURE,
    {1280, 720},
    LVFRAME_EVEN_ODD,
    LVFRAME_FRAME_MODE,
    DMAORDER_PACKET,
    CAPSCALER_NOT_KEEP_RATIO,
    MCP_VIDEO_NTSC,
    CAPCOLOR_YUV422,
    { FALSE, FALSE, GM3DI_FIELD }
},
{
    DVR_ENC_EBST_ENABLE,
    0,
    ENC_TYPE_H264,
    FALSE,
    DVR_ENC_EBST_ENABLE,
    {1280, 720},
    {ENC_INPUT_H2642D, 30, 1048576, 30, 25, 5
```

```
1, 1 , FALSE, {0, 0, 320, 240}},  
    {SCALE_YUV422, SCALE_YUV422, SCALE_LINEAR,  
FALSE, FALSE, TRUE, 0 },  
    {JCS_yuv420, 0, JENC_INPUT_MP42D, 70}  
  
}  
};  
  
void do_record_start(int ch_num)  
{  
    memcpy(&ch_param, &user_rec_ch_setting, sizeof  
(ch_param));  
  
    ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_  
ADD_VAL(sizeof(enc_param_ext));  
    ch_param.main_bs.enc.pext_data = &enc_param_ex  
t;  
  
    enc_param_ext.feature_enable = 0;  
  
    ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_p  
aram);  
  
    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE  
, &enc_buf_size);  
  
    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP  
_OFFSET, &bs_buf_snap_offset);  
  
    bs_buf = (unsigned char*) mmap(NULL, enc_buf_s  
ize, PROT_READ|PROT_WRITE,  
                                MAP_SHAR  
ED, enc_fd, 0);  
    //record start  
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control)  
);  
    enc_ctrl.command = ENC_START;
```

```

    enc_ctrl.stream = 0;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    // set function tag parameter to dvr graph lev
el
    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
}

void do_record_stop(int ch_num)
{
    //record stop
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
    enc_ctrl.stream = 0;
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    munmap((void*)bs_buf, enc_buf_size);
}

void snapshot_trigger(int ch_num)
{
    int snap_no = 1;
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
    enc_ctrl.command = ENC_SNAP;
    enc_ctrl.output.count = (int *)snap_no;      // 
< do snapshot 1 time
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
}

/**

```

```

 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num = 0;
    dvr_enc_queue_get    data;
    unsigned char *buf;
    int buf_size;
    struct timeval t1,t2;
    char tmp_str[128];
    int mb_width,mb_height;
    int count;
    int flag_snapshot = 1;

    dvr_fd = open( "/dev/dvr_common", O_RDWR);      //  

open_dvr_common

    enc_fd = open( "/dev/dvr_enc", O_RDWR);          //  

open_dvr_encode

    do_record_start(ch_num);

    //do motion detection init
    mb_width = (ch_param.src.dim.width + 16 - 1) /  

16;
    mb_height = (ch_param.src.dim.height + 16 - 1)  

/ 16;

    favc_motion_info_init(&md_param, &active, &gfa  

vcmd, mb_width, mb_height);

    for (count = 0; count < 1; count++) {
        int i,j;
        md_param.interlace_mode = 0;
        md_param.usesad = 1; //no sad
        md_param.mb_time_th = 3;

```

```

        md_param.rolling_prot_th = 30;
        md_param.alarm_th = 10;

        for (i = 0; i < mb_height; i++)
            for (j = 0; j < mb_width; j++)
                md_param.mb_cell_en[(i*mb_width +
j)] = 1;

        for (i = 0; i < mb_height; i++)
            for (j = 0; j < mb_width; j++)
                md_param.mb_cell_th[(i*mb_width +
j)] = 10;
    }

    sprintf(file_name, "CH%d_video_%d", 0, 0);

    sprintf(tmp_str, "%s.h264", file_name);

    rec_file = fopen ( tmp_str , "wb+" );
    gettimeofday(&t1, NULL);

    while(1) {
        // prepare to select(or poll)
        rec_fds.fd = enc_fd;
        rec_fds.revents = 0;
        rec_fds.events = (POLLIN_MAIN_BS | POLLIN_
SNAP_BS);

        poll(&rec_fds, 1, 500);

        if (rec_fds.revents & POLLIN_SNAP_BS) {

            ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET_
SNAP, &data);
            if (ret >= 0) {
                FILE * fd;

```

```

        buf = bs_buf + bs_buf_snap_offset
+ data.bs.offset;
        buf_size = data.bs.length;

        -- flag_snapshot;

        sprintf(tmp_str, "CH%d_snapshot%02
d_%03d.jpg", ch_num, flag_snapshot, encode_frame_c
ount);
        fd = fopen ( tmp_str , "wb+" );
        fwrite(buf , 1 , buf_size , fd);

        fclose(fd);

        printf("snapshot: output file %s\n"
, tmp_str);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
    }
}

if (!(rec_fds.revents & POLLIN_MAIN_BS))
    continue;

// get dataout buffer
ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET, &da
ta);
if(ret < 0)
    continue;

encode_frame_count++;
buf = bs_buf + data.bs.offset;
buf_size = data.bs.length;

//do motion detection
for (count = 0; count < 1; count++) {

```

```

        favc_motion_detection(bs_buf + data.bs.
mv_offset, &md_param, &active, &gfavcmd);

        if ((active.active_num >= md_param.alarm_th)) {
                printf("Motion Alarm @ %s< %d >\n"
, "main_stream", active.active_num);
                snapshot_trigger(ch_num);
        }
}

fwrite(buf , 1 , buf_size , rec_file);
fflush(rec_file);
ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data);

gettimeofday(&t2, NULL);

if ((t2.tv_sec - t1.tv_sec) == 20) {
//< record for 20 seconds. then finish record.

        fclose(rec_file);
        break;
}
}

do_record_stop(ch_num);

printf("-----\n");
printf(" Record finish\n");
printf("-----\n");
;

close(enc_fd);           //close_dvr_encode
close(dvr_fd);           //close_dvr_common
;
```

```
    return 0;  
}
```



Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

mpeg4-record.c

```
/**  
 * this sample code implement main bitstream MPEG4  
 record function, and record for 20 seconds.  
 * (1) for H264 record, file format please indicate xxx.h264  
 *      for MPEG record, file format please indicate xxx.m4v  
 *      for MOTION JPEG record, file format please indicate xxx.jpg  
 * (2) for H264 record, max_quant < 51, 1 < min_quant  
 *      for MPEG record, max_quant < 31, 1 < min_quant  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int enc_fd = 0;
```

```

//test record
unsigned char *bs_buf;
HANDLE rec_file;
int avi_str_id;
int enc_buf_size;
struct pollfd rec_fds;
char file_name[128];

/**
 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num = 0;
    dvr_enc_channel_param    ch_param;
    EncParam_Ext3 enc_param_ext = {0};
    dvr_enc_control    enc_ctrl;
    dvr_enc_queue_get    data;
    unsigned char *buf;
    int buf_size;
    FuncTag tag;
    struct timeval t1,t2;
    char tmp_str[128];

    dvr_fd = open( "/dev/dvr_common", 0_RDWR);      // 
open_dvr_common

    enc_fd = open( "/dev/dvr_enc", 0_RDWR);          // 
open_dvr_encode

    //set dvr encode source parameter
    ch_param.src.input_system = MCP_VIDEO0_NTSC;
    ch_param.src.channel = ch_num;
    ch_param.src.enc_src_type = ENC_TYPE_FROM_CAPT
URE;

```

```
ch_param.src.dim.width = 1280;
ch_param.src.dim.height = 720;

ch_param.src.di_mode = LVFRAME_EVEN_ODD;
ch_param.src.mode = LVFRAME_FRAME_MODE;
ch_param.src.dma_order = DMAORDER_PACKET;
ch_param.src.scale_indep = CAPSCALER_NOT_KEEP_
RATIO;
ch_param.src.input_system = MCP_VIDEO_NTSC;
ch_param.src.color_mode = CAPCOLOR_YUV422;

ch_param.src.vp_param.is_3DI = FALSE;
ch_param.src.vp_param.is_denoise = FALSE;
ch_param.src.vp_param.denoise_mode = GM3DI_FIE
LD;

//set dvr encode main bitstream parameter
ch_param.main_bs.enabled = DVR_ENC_EBST_ENABLE
;
ch_param.main_bs.out_bs = 0;
ch_param.main_bs.enc_type = ENC_TYPE_MPEG;
ch_param.main_bs.is_blocked = FALSE;
ch_param.main_bs.en_snapshot = DVR_ENC_EBST_DI
SABLE;

ch_param.main_bs.dim.width = 1280;
ch_param.main_bs.dim.height = 720;

//set main bitstream encode parameter
ch_param.main_bs.enc.input_type = ENC_INPUT_H2
642D;
ch_param.main_bs.enc.frame_rate = 30;
ch_param.main_bs.enc.bit_rate = 1048576;
ch_param.main_bs.enc.ip_interval = 30;
ch_param.main_bs.enc.init_quant = 25;
ch_param.main_bs.enc.max_quant = 31;
```

```

ch_param.main_bs.enc.min_quant = 1;
ch_param.main_bs.enc.is_use_ROI = FALSE;
ch_param.main_bs.enc.ROI_win.x = 0;
ch_param.main_bs.enc.ROI_win.y = 0;
ch_param.main_bs.enc.ROI_win.width = 320;
ch_param.main_bs.enc.ROI_win.height = 240;

//set main bitstream scalar parameter
ch_param.main_bs.scl.src_fmt = SCALE_YUV422;
ch_param.main_bs.scl.dst_fmt = SCALE_YUV422;
ch_param.main_bs.scl.scale_mode = SCALE_LINEAR
;

ch_param.main_bs.scl.is_dither = FALSE;
ch_param.main_bs.scl.is_correction = FALSE;
ch_param.main_bs.scl.is_album = TRUE;
ch_param.main_bs.scl.des_level = 0;

//set main bitstream snapshot parameter
ch_param.main_bs.snap.sample = JCS_yuv420;
ch_param.main_bs.snap.RestartInterval = 0;
ch_param.main_bs.snap.u82D = JENC_INPUT_MP42D;

ch_param.main_bs.snap.quality = 70;

//associate the ext. structure

ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_
ADD_VAL(sizeof(enc_param_ext));
ch_param.main_bs.enc.pext_data = &enc_param_ex
t;

enc_param_ext.feature_enable = 0;           //CBR

ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_p
aram);

ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE

```

```

, &enc_buf_size);

    bs_buf = (unsigned char*) mmap(NULL, enc_buf_size,
        PROT_READ|PROT_WRITE,
                                         MAP_SHARED,
        enc_fd, 0);
    ///////////////////////////////////////////////////////////////////
    ///////////////////////////////////////////////////////////////////
    //record start
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
    enc_ctrl.command = ENC_START;
    enc_ctrl.stream = 0;
    ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
}

// set function tag parameter to dvr graph level
FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
///////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////
sprintf(file_name, "CH%d_video_%d", 0, 0);

sprintf(tmp_str, "%s.m4v", file_name);

rec_file = fopen ( tmp_str , "wb+" );
gettimeofday(&t1, NULL);

while(1) {
    // prepare to select(or poll)
    rec_fds.fd = enc_fd;
    rec_fds.revents = 0;
    rec_fds.events = POLLIN_MAIN_BS;

    poll(&rec_fds, 1, 500);
}

```

```
        if (!(rec_fds.revents & POLLIN_MAIN_BS))
            continue;

        // get dataout buffer
        ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET,
&data);
        if(ret < 0)
            continue;

        buf = bs_buf + data.bs.offset;
        buf_size = data.bs.length;

        ret = fwrite (buf , 1 , buf_size , rec
_file);
        fflush(rec_file);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data
);

        gettimeofday(&t2, NULL);

        if ((t2.tv_sec - t1.tv_sec) == 20) {
            //<record for 20 seconds. then finish record.

            fclose(rec_file);
            break;
        }
    }

    //record stop
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control)
);
    enc_ctrl.stream = 0;
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
```

```
FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
munmap((void*)bs_buf, enc_buf_size);

printf("-----\n");
printf(" Record finish\n");
printf("-----\n");
;

close(enc_fd);           //close_dvr_encode
close(dvr_fd);           //close_dvr_common

return 0;
}
```

Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by
doxygen 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

pip.c

```
/**  
 * this sample code implement liveview PIP function and update pip channel's window position & resolution.  
 * step :  
 * push key "a" to start channel 2 as layer0,  
 * push key "x" to start channel 1 as layer1,  
 * push key "1" to update channel 1's window position at (0, 0)  
 * push key "2" to update channel 1's window position at (352, 0)  
 * push key "3" to update channel 1's window position at (0, 240)  
 * push key "4" to update channel 1's window position at (352, 240)  
 * push key "s" to swap channel 1 as layer0, channel 2 as layer1  
 * push key "5" to update channel 2's window position at (0, 0)  
 * push key "6" to update channel 2's window position at (352, 0)  
 * push key "7" to update channel 2's window position at (0, 240)  
 * push key "8" to update channel 2's window position at (352, 240)  
 * push key "b" to stop channel 2,  
 * push key "y" to stop channel 1,  
 * push key "q" to exit program,  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>
```

```
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <poll.h>

#include "dvr_common_api.h"
#include "dvr_disp_api.h"
#include "dvr_enc_api.h"

int dvr_fd = 0;
int disp_fd = 0;
int main_disp_no = 0;
int main_plane_id = 0;
int open_flag = 0;
int is_NTSC = TRUE; //if FALSE, it's PAL

char getch(void)
{
    int n = 1;
    unsigned char ch;
    struct timeval tv;
    fd_set rfds;

    FD_ZERO(&rfds);
    FD_SET(0, &rfds);
    tv.tv_sec = 10;
    tv.tv_usec = 0;
    n = select(1, &rfds, NULL, NULL, &tv);
    if (n > 0) {
        n = read(0, &ch, 1);
        if (n == 1)
            return ch;
    }
    return n;
```

```
    }
    return -1;
}

int setup_lv_channel(int ch_num, int cmd, int is_u
se_scaler, int di_mode, int mode, int is_3DI, int
is_denoise, int dn_mode, DIM *src_dim, RECT *src_r
ect, RECT *dst_rect, int pip)
{
    int ret;
    dvr_disp_control disp_ctrl;
    DispParam_Ext1 disp_param_ext = {0};

    memset(&disp_ctrl, 0x0, sizeof(dvr_disp_control)
);
    disp_ctrl.type = DISP_TYPE_LIVEVIEW;
    disp_ctrl.channel = ch_num;
    disp_ctrl.command = cmd;
    if(cmd != DISP_STOP)
    {
        disp_ctrl.src_param.lv.cap_path = ch_num;

        disp_ctrl.src_param.lv.di_mode = di_mode;
        disp_ctrl.src_param.lv.mode = mode;

        disp_ctrl.src_param.lv.vp_param.is_3DI = i
s_3DI;
        disp_ctrl.src_param.lv.vp_param.is_denoise
= is_denoise;
        disp_ctrl.src_param.lv.vp_param.denoise_mo
de = dn_mode;

        disp_ctrl.src_param.lv.dma_order = DMAORDE
R_PACKET;
        disp_ctrl.src_param.lv.scale_indep = CAPSC
ALER_NOT_KEEP_RATIO;
        disp_ctrl.src_param.lv.input_system = (is_
```

```
NTSC)? MCP_VIDEO_NTSC: MCP_VIDEO_PAL;
        disp_ctrl.src_param.lv.cap_rate = (is_NTSC
)? 30: 25;

        disp_ctrl.src_param.lv.color_mode = CAPCOL
OR_YUV422;
        disp_ctrl.dst_param.lv.plane_id = main_plane_id;

        disp_ctrl.src_param.lv.is_use_scaler = is_
use_scaler;
        disp_ctrl.src_param.lv.dim.width = src_dim
->width;
        disp_ctrl.src_param.lv.dim.height = src_di
m->height;
        disp_ctrl.src_param.lv.ext_size = DVR_DISP
_MAGIC_ADD_VAL(sizeof(disp_param_ext));
        disp_ctrl.src_param.lv.pext_data = &disp_p
aram_ext;

        if(pip == 1) {
            disp_param_ext.chn_type = DISP_LAYER1_
CHN;
        }
        else {
            disp_param_ext.chn_type = DISP_LAYER0_
CHN;
        }

        if(is_use_scaler)
{
            if(src_rect)
{
                disp_ctrl.src_param.lv.win.x = src
_rect->x;
                disp_ctrl.src_param.lv.win.y = src
_rect->y;

```

```

        disp_ctrl.src_param.lv.win.width =
src_rect->width;
        disp_ctrl.src_param.lv.win.height
= src_rect->height;
    }
    else
    {
        disp_ctrl.src_param.lv.win.x = 0;
        disp_ctrl.src_param.lv.win.y = 0;
        disp_ctrl.src_param.lv.win.width =
src_dim->width;
        disp_ctrl.src_param.lv.win.height
= src_dim->height;
    }

        disp_ctrl.src_param.lv.scl_param.src_f
mt = SCALE_YUV422;
        disp_ctrl.src_param.lv.scl_param.dst_f
mt = SCALE_YUV422;
        disp_ctrl.src_param.lv.scl_param.scale
_mode = SCALE_LINEAR;
        disp_ctrl.src_param.lv.scl_param.is_di
ther = FALSE;
        disp_ctrl.src_param.lv.scl_param.is_co
rrection = FALSE;
        disp_ctrl.src_param.lv.scl_param.is_al
bum = TRUE;
        disp_ctrl.src_param.lv.scl_param.des_l
evel = 0;
    }
    else
    {
        disp_ctrl.src_param.lv.win.x = 0;
        disp_ctrl.src_param.lv.win.y = 0;
        disp_ctrl.src_param.lv.win.width = src
_dim->width;
        disp_ctrl.src_param.lv.win.height = sr

```

```
c_dim->height;

        disp_ctrl.src_param.lv.scl_param.src_f
mt = SCALE_YUV422;
        disp_ctrl.src_param.lv.scl_param.dst_f
mt = SCALE_YUV422;
        disp_ctrl.src_param.lv.scl_param.scale
_mode = SCALE_LINEAR;
        disp_ctrl.src_param.lv.scl_param.is_di
ther = FALSE;
        disp_ctrl.src_param.lv.scl_param.is_co
rrection = FALSE;
        disp_ctrl.src_param.lv.scl_param.is_al
bum = TRUE;
        disp_ctrl.src_param.lv.scl_param.des_l
evel = 0;
    }
    disp_ctrl.dst_param.lv.win.x = dst_rect->x
;
    disp_ctrl.dst_param.lv.win.y = dst_rect->y
;
    disp_ctrl.dst_param.lv.win.width = dst_rec
t->width;
    disp_ctrl.dst_param.lv.win.height = dst_re
ct->height;
}

ret = ioctl(disp_fd, DVR_DISP_CONTROL, &disp_c
trl);

return ret;
}

int update_lv_pip_channel(int ch_num, RECT *dst_re
ct)
{
    int ret;
```

```
dvr_disp_control    disp_ctrl;
FuncTag  tag;

    memset(&disp_ctrl, 0x0, sizeof(dvr_disp_control));
}

    disp_ctrl.channel = ch_num;
    disp_ctrl.command = DISP_UPDATE;
    //PIP mode does not support to update the following parameter
    disp_ctrl.src_param.lv.cap_path = GMVAL_DO_NOT_CARE;
    disp_ctrl.src_param.lv.di_mode = GMVAL_DO_NOT_CARE;
    disp_ctrl.src_param.lv.mode = GMVAL_DO_NOT_CARE;

    disp_ctrl.src_param.lv.vp_param.is_3DI = GMVAL_DO_NOT_CARE;
    disp_ctrl.src_param.lv.vp_param.is_denoise = GMVAL_DO_NOT_CARE;
    disp_ctrl.src_param.lv.vp_param.denoise_mode = GMVAL_DO_NOT_CARE;

    disp_ctrl.src_param.lv.dma_order = GMVAL_DO_NOT_CARE;
    disp_ctrl.src_param.lv.scale_indep = GMVAL_DO_NOT_CARE;
    disp_ctrl.src_param.lv.input_system = GMVAL_DO_NOT_CARE;
    disp_ctrl.src_param.lv.cap_rate = GMVAL_DO_NOT_CARE;
    disp_ctrl.src_param.lv.color_mode = GMVAL_DO_NOT_CARE;
    disp_ctrl.dst_param.lv.plane_id = GMVAL_DO_NOT_CARE;
```

```

        disp_ctrl.src_param.lv.is_use_scaler = GMVAL_D
O_NOT_CARE;
        disp_ctrl.src_param.lv.dim.width = GMVAL_DO_NO
T_CARE;
        disp_ctrl.src_param.lv.dim.height = GMVAL_DO_N
OT_CARE;

        disp_ctrl.src_param.lv.win.x = GMVAL_DO_NOT_CA
RE;
        disp_ctrl.src_param.lv.win.y = GMVAL_DO_NOT_CA
RE;
        disp_ctrl.src_param.lv.win.width = GMVAL_DO_NO
T_CARE;
        disp_ctrl.src_param.lv.win.height = GMVAL_DO_N
OT_CARE;

        disp_ctrl.src_param.lv.scl_param.src_fmt = GMV
AL_DO_NOT_CARE;
        disp_ctrl.src_param.lv.scl_param.dst_fmt = GMV
AL_DO_NOT_CARE;
        disp_ctrl.src_param.lv.scl_param.scale_mode =
GMVAL_DO_NOT_CARE;
        disp_ctrl.src_param.lv.scl_param.is_dither = G
MVAL_DO_NOT_CARE;
        disp_ctrl.src_param.lv.scl_param.is_correction
= GMVAL_DO_NOT_CARE;
        disp_ctrl.src_param.lv.scl_param.is_album = GM
VAL_DO_NOT_CARE;
        disp_ctrl.src_param.lv.scl_param.des_level = G
MVAL_DO_NOT_CARE;
        // PIP mode only support to update the followi
ng parameter
        disp_ctrl.dst_param.lv.win.x = dst_rect->x;
        disp_ctrl.dst_param.lv.win.y = dst_rect->y;
        disp_ctrl.dst_param.lv.win.width = dst_rect->w
idth;
        disp_ctrl.dst_param.lv.win.height = dst_rect->

```

```
height;

    ret = ioctl(disp_fd, DVR_DISP_CONTROL, &disp_ctrl);

    FN_RESET_TAG(&tag);
    FN_SET_LV_CH(&tag, ch_num);

    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);

    return ret;
}

int run_lv_command(int ch_num)
{
    int ret;
    FuncTag tag;

    FN_RESET_TAG(&tag);
    FN_SET_LV_CH(&tag, ch_num);

    ret = ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    if(ret<0)
        return -1;

    return ret;
}

int do_liveview_closech(int ch_num)
{
    int ret;

    ret = setup_lv_channel(ch_num, DISP_STOP, 0, 0
    , 0, FALSE, FALSE, 0, NULL, NULL, NULL, 0);
    if(ret<0)
        return -1;
```

```
    ret = run_lv_command(ch_num);

    return ret;
}

int do_disp_startup()
{
    int i, ret;
    dvr_disp_disp_param          disp_param;
    dvr_disp_update_disp_param   disp_update_param;
    dvr_disp_plane_param         plane_param[3];
    dvr_disp_update_plane_param  plane_update_pa;
    dvr_disp_control            dsp_ctl;

    open_flag = 1;

    disp_fd = open("/dev/dvr_disp", O_RDWR);

    memset(&disp_param, 0x0, sizeof(dvr_disp_disp_param));
    memset(&disp_update_param, 0x0, sizeof(dvr_disp_update_disp_param));
    memset(plane_param, 0x0, sizeof(dvr_disp_plane_param) * 3);
    memset(&plane_update_pa, 0x0, sizeof(dvr_disp_update_plane_param));
    memset(&dsp_ctl, 0x0, sizeof(dvr_disp_control));
}

main_disp_no = 0;

// query LCD1 information
disp_param.disp_num = main_disp_no;
ret = ioctl(disp_fd, DVR_DISP_GET_DISP_PARAM,
&disp_param);

// set BG_AND_2PLANE, which means we need 1 ba
```

```
ckground and another 2 planes
    disp_update_param.disp_num = main_disp_no;
    disp_update_param.param = DISP_PARAM_PLANE_COM
BINATION;
    disp_update_param.val.plane_comb = BG_ONLY;//B
G_AND_2PLANE;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
    if (ret < 0)
        return -1;

    disp_update_param.disp_num = main_disp_no;
    disp_update_param.param = DISP_PARAM_OUTPUT_SY
STEM;

    disp_update_param.val.output_system = MCP_VIDE
O_VGA;
    disp_update_param.val.display_rate = is_NTSC?
30 : 25;

    ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
    if (ret < 0)
        return -1;

    disp_update_param.param = DISP_PARAM_APPLY;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
    if (ret < 0)
        return -1;

// query 3 planes information
for(i = 0; i < 3; i++) {
    plane_param[i].disp_num = main_disp_no;
    plane_param[i].plane_num = i;
    ret = ioctl(disp_fd, DVR_DISP_GET_PLANE_PA
RAM, &plane_param[i]);
```

```
        if (ret < 0)
            return -1;

        if (i == 0)
            main_plane_id = plane_param[i].param.p
lane_id;
    }

    // set color mode for background plane
    plane_update_pa.plane_id = plane_param[0].param
.plane_id;
    plane_update_pa.param = PLANE_PARAM_COLOR_MODE
;
    plane_update_pa.val.color_mode = LCD_COLOR_YUV
422;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PAR
AM, &plane_update_pa);
    if (ret < 0)
        return -1;

    // set data mode for background plane
    plane_update_pa.plane_id = plane_param[0].param
.plane_id;
    plane_update_pa.param = PLANE_PARAM_DATA_MODE;
    plane_update_pa.val.data_mode = LCD_PROGRESSIV
E;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PAR
AM, &plane_update_pa);
    if (ret < 0)
        return -1;

    plane_update_pa.param = PLANE_PARAM_APPLY;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PAR
AM, &plane_update_pa);
    if (ret < 0)
        return -1;
```

```
    return 0;
}

int do_disp_endup()
{
    if (!open_flag) {
        printf("Multi close\n");
        return -1;
    }

    if (disp_fd > 0)
        close(disp_fd);

    disp_fd = 0;
    open_flag = 0;

    return 0;
}

int open_dvr_common(void)
{
    dvr_fd = open("/dev/dvr_common", O_RDWR);
    if (dvr_fd < 0) {
        perror("Open [/dev/dvr_common] failed:");
        return -1;
    }
    return 0;
}

void close_dvr_common(void)
{
    close(dvr_fd);
}

int main(int argc, char *argv[])
{
    int ret = 0;
```

```
DIM dim;
RECT win;
char key;

open_dvr_common();

do_disp_startup();

while(1)
{
    key = getch();
    if (key == 'q' || key == 'Q')
        break;
    switch(key)
    {
        case 'x':
            dim.width = 352;
            dim.height = 240;
            win.x = 352;
            win.y = 240;
            win.width = 352;
            win.height = 240;
            setup_lv_channel(1, DISP_START, 0,
LVFRAME_WEAVED_TWO_FIELDS, LVFRAME_FRAME_MODE, FALSE, FALSE, 0, &dim, NULL, &win, 1);

            ret = run_lv_command(1);
            break;

        case 'y':
            ret = do_liveview_closech(1);
            break;
        case 'a':
            dim.width = 720;
            dim.height = 480;
            win.x = 0;
            win.y = 0;
    }
}
```

```
        win.width = 720;
        win.height = 480;
        setup_lv_channel(2, DISP_START, 0,
LVFRAME_WEAVED_TWO_FIELDS, LVFRAME_FRAME_MODE, FALSE,
FALSE, 0, &dim, NULL, &win, 0);
        ret = run_lv_command(2);
        break;

    case 'b':
        ret = do_liveview_closech(2);
        break;
    case '1':
        win.x = 0;
        win.y = 0;
        win.width = 352;
        win.height = 240;
        ret = update_lv_pip_channel(1, &wi
n);
        break;
    case '2':
        win.x = 352;
        win.y = 0;
        win.width = 352;
        win.height = 240;
        ret = update_lv_pip_channel(1, &wi
n);
        break;
    case '3':
        win.x = 0;
        win.y = 240;
        win.width = 352;
        win.height = 240;
        ret = update_lv_pip_channel(1, &wi
n);
        break;
    case '4':
        win.x = 352;
```

```
    win.y = 240;
    win.width = 352;
    win.height = 240;
    ret = update_lv_pip_channel(1, &wi
n);
        break;
case '5':
    win.x = 0;
    win.y = 0;
    win.width = 352;
    win.height = 240;
    ret = update_lv_pip_channel(2, &wi
n);
        break;
case '6':
    win.x = 352;
    win.y = 0;
    win.width = 352;
    win.height = 240;
    ret = update_lv_pip_channel(2, &wi
n);
        break;
case '7':
    win.x = 0;
    win.y = 240;
    win.width = 352;
    win.height = 240;
    ret = update_lv_pip_channel(2, &wi
n);
        break;
case '8':
    win.x = 352;
    win.y = 240;
    win.width = 352;
    win.height = 240;
    ret = update_lv_pip_channel(2, &wi
n);
```

```
        break;
    case 's':
        do_liveview_closech(1);
        do_liveview_closech(2);
        dim.width = 352;
        dim.height = 240;
        win.x = 0;
        win.y = 0;
        win.width = 720;
        win.height = 480;
        setup_lv_channel(1, DISP_START, 0,
LVFRAME_WEAVED_TWO_FIELDS, LVFRAME_FRAME_MODE, FALSE, FALSE, 0, &dim, NULL, &win, 0);
        ret = run_lv_command(1);

        dim.width = 352;
        dim.height = 240;
        win.x = 352;
        win.y = 240;
        win.width = 352;
        win.height = 240;
        setup_lv_channel(2, DISP_START, 0,
LVFRAME_WEAVED_TWO_FIELDS, LVFRAME_FRAME_MODE, FALSE, FALSE, 0, &dim, NULL, &win, 1);
        ret = run_lv_command(2);
        break;
    }
}

do_disp_endup();

close_dvr_common();
printf("finish\n");
return 0;
}
```

Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

playback.c

```
/**  
 * this sample code implement playback function, a  
nd playback for 20 seconds.  
 * demo file is CH0_video_0.avi,  
 * please put the demo file at the same directory  
with executed binary file.  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_disp_api.h"  
#include "dvr_enc_api.h"  
#include "dvr_dec_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int disp_fd = 0;  
int dec_fd = 0 ;  
int main_disp_no = 0;  
int main_plane_id = 0;  
int open_flag = 0;  
int is_NTSC = TRUE;
```

```
#define FUNCTION_NULL      -1
#define FUNCTION_NORMAL     0

int menu_func = FUNCTION_NULL;
#define USE_UPDATE_METHOD   1

unsigned char *pbbs_buf;
int dec_buf_size;
HANDLE pb_file= NULL;
AviMainHeader pb_main_header;
AviStreamHeader pb_stream_header;
GmAviStreamFormat pb_stream_format;
int stream_id_pkt = 0;
struct pollfd dec_fds;
int is_st_vga = TRUE;

RECT D1_4CH[4] = {
    { 0, 0, 360, 240}, {360, 0, 360, 240},
    { 0, 240, 360, 240}, {360, 240, 360, 240}
};

RECT VGA_4CH[4] = {
    { 0, 0, 640, 360}, {512, 0, 640, 360},
    { 0, 360, 512, 360}, {512, 360, 512, 360}
};

int do_disp_startup()
{
    int i, ret;
    dvr_disp_disp_param          disp_param;
    dvr_disp_update_disp_param   disp_update_param;
    dvr_disp_plane_param         plane_param[3];
    dvr_disp_update_plane_param  plane_update_pa;
    dvr_disp_control             dsp_ctl;

    if (open_flag) {
```

```
        printf("multi open\n");
        return -1;
    }
    open_flag = 1;

    disp_fd = open("/dev/dvr_disp", O_RDWR);
    if (disp_fd < 0) {
        perror("Open failed:");
        open_flag = 0;
        return -1;
    }

    memset(&disp_param, 0x0, sizeof(dvr_disp_disp_param));
    memset(&disp_update_param, 0x0, sizeof(dvr_disp_update_disp_param));
    memset(plane_param, 0x0, sizeof(dvr_disp_plane_param) * 3);
    memset(&plane_update_pa, 0x0, sizeof(dvr_disp_update_plane_param));
    memset(&dsp_ctl, 0x0, sizeof(dvr_disp_control));
}

main_disp_no = 0;

// query LCD1 information
disp_param.disp_num = main_disp_no;
ret = ioctl(disp_fd, DVR_DISP_GET_DISP_PARAM,
&disp_param);
if (ret < 0)
    return -1;
printf("LCD(%d): dim(%d-%d) res(in=%d,out=%d)
plane_comb(%d) system(%d) mode(%d)\n",
    1, disp_param.dim.width, disp_param.dim.height,
    disp_param.res.input_res, disp_param.res.output_type,
    disp_param.plane_comb, disp_param.output_s
```

```

        ystem, disp_param.output_mode);
        printf("      : color attrib.: br(%d)sa(%d)co
(%d)-hus(%d)huc(%d)sh0(%d)sh1-(%d)shth0(%d)shth1(%
d)\n",
              disp_param.color_attrib.brightness, disp_p
aram.color_attrib.saturation, disp_param.color_att
rib.contrast,
              disp_param.color_attrib.huesin, disp_param.
color_attrib.huecos, disp_param.color_attrib.sharp
nessk0,
              disp_param.color_attrib.sharpnessk1, disp_
param.color_attrib.sharpness_thres0, disp_param.co
lor_attrib.shaprness_thres1);
        printf("      : transparent: color1(%d,%d) co
lor2(%d,%d)\n",
              disp_param.transparent_color[0].is_enable,
disp_param.transparent_color[0].color,
              disp_param.transparent_color[1].is_enable,
disp_param.transparent_color[1].color);
        usleep(100000);

#if USE_UPDATE_METHOD
    // set BG_AND_2PLANE, which means we need 1 ba
ckground and another 2 planes
    disp_update_param.disp_num = main_disp_no;
    disp_update_param.param = DISP_PARAM_PLANE_COM
BINATION;
    disp_update_param.val.plane_comb = BG_ONLY;//B
G_AND_2PLANE;
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
    if (ret < 0)
        return -1;

    disp_update_param.disp_num = main_disp_no;
    disp_update_param.param = DISP_PARAM_OUTPUT_SY
STEM;

```

```

        disp_update_param.val.output_system = MCP_VIDEO_VGA;
        disp_update_param.val.display_rate = is_NTSC?
30 : 25;

        ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
        if (ret < 0)
            return -1;
        disp_update_param.param = DISP_PARAM_APPLY;
        ret = ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PARAM
, &disp_update_param);
        if (ret < 0)
            return -1;
#else
    // set BG_AND_2PLANE, which means we need 1 background and another 2 planes
    disp_param.plane_comb = DISP_PARAM_PLANE_COMBINATION;
    ret = ioctl(disp_fd, DVR_DISP_SET_DISP_PARAM,
&disp_param);
    if (ret < 0)
        return -1;
#endif
    // query 3 planes information
    for(i = 0; i < 3; i++) {
        plane_param[i].disp_num = main_disp_no;
        plane_param[i].plane_num = i;
        ret = ioctl(disp_fd, DVR_DISP_GET_PLANE_PA
RAM, &plane_param[i]);
        if (ret < 0)
            return -1;
        printf("LCD(%d)-plane(%d): ID(%d) rect(%d-
%d,%d-%d) data_mode(%d) color_mode(%d)\n",
            main_disp_no, plane_param[i].plane_num
, plane_param[i].param.plane_id,

```

```
        plane_param[i].param.win.x, plane_para  
m[i].param.win.y, plane_param[i].param.win.width,  
plane_param[i].param.win.height,  
        plane_param[i].param.data_mode, plane_  
param[i].param.color_mode  
    );  
    usleep(50000);  
    if (i == 0)  
        main_plane_id = plane_param[i].param.p  
lane_id;  
}  
  
#if USE_UPDATE_METHOD  
    // set color mode for background plane  
    plane_update_pa.plane_id = plane_param[0].param  
.plane_id;  
    plane_update_pa.param = PLANE_PARAM_COLOR_MODE  
;  
    plane_update_pa.val.color_mode = LCD_COLOR_YUV  
422;  
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PAR  
AM, &plane_update_pa);  
    if (ret < 0)  
        return -1;  
  
    // set data mode for background plane  
    plane_update_pa.plane_id = plane_param[0].param  
.plane_id;  
    plane_update_pa.param = PLANE_PARAM_DATA_MODE;  
    plane_update_pa.val.data_mode = LCD_PROGRESSIV  
E;  
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PAR  
AM, &plane_update_pa);  
    if (ret < 0)  
        return -1;  
  
    plane_update_pa.param = PLANE_PARAM_APPLY;
```

```
    ret = ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_PARAM,
AM, &plane_update_pa);
    if (ret < 0)
        return -1;

#else /* USE_UPDATE_METHOD */
    plane_param[0].param.color_mode = LCD_COLOR_YUV422;
    plane_param[0].param.data_mode = LCD_PROGRESSIVE;
    ret = ioctl(disp_fd, DVR_DISP_SET_PLANE_PARAM,
&plane_param[0]);
    if (ret < 0)
        return -1;

    ret = ioctl(disp_fd, DVR_DISP_SET_PLANE_PARAM,
&plane_param[1]);
    if (ret < 0)
        return -1;
#endif /* USE_UPDATE_METHOD */
    return 0;
}

int do_disp_endup()
{
    if (!open_flag) {
        printf("Multi close\n");
        return -1;
    }

    if (disp_fd > 0)
        close(disp_fd);

    disp_fd = 0;
    open_flag = 0;

    return 0;
```

```
}

int open_pb_files(int ch_num, int index)
{
    int strh_count;
    char tmp_str[64];

    sprintf(tmp_str, "CH%d_video_%d.avi", ch_num,
index);

    pb_file = GMAVIOpen(tmp_str, GMAVI_FILEMODE_READ, 0);
    if (!pb_file) {
        printf("Open [%s] failed!\n", tmp_str);
        return -1;
    }
    GMAVIGetAviMainHeader(pb_file, &pb_main_header);
    GMAVIGetStreamHeaderNum(pb_file, &strh_count);
    GMAVIGetStreamHeader(pb_file, 1/*get stream1*/
, &pb_stream_header, &pb_stream_format, &stream_id
_pkp);
    printf("dec type:<%c%c%c%c>\n", pb_stream_heade
r.fccHandler[0]
, pb_stream_header.
fccHandler[1]
, pb_stream_header.
fccHandler[2]
, pb_stream_header.
fccHandler[3]);

    return 0;
}

void close_pb_files(int ch_num)
{
    if(pb_file) {
```

```
        GMAVICClose(pb_file);
        pb_file = NULL;
    }
}

#define MAX_RECORD 4096
void packet_reader(int arg)
{
    int ret, ch_num;
    int pb_idx = 0, pb_total = 0;
    int pb_offset[MAX_RECORD] = {0};
    int reverse;
    dvr_enc_queue_get    data;
    unsigned char *buf;
    int buf_size = 0,intra = 0, backup_data = 0;

    unsigned int i = 0;
    int file_idx = 1;
    dvr_dec_control      dec_ctrl;
    FuncTag tag;
    struct timeval t1,t2;

    memset(&dec_ctrl, 0x0, sizeof(dvr_dec_control))
);

    ch_num = arg;

    // prepare to select(or poll)
    dec_fds.fd = dec_fd;
    dec_fds.events = POLLIN;
    dec_fds.revents = 0;

    gettimeofday(&t1, NULL);

    while(1) {
        gettimeofday(&t2, NULL);
```

```

        if ((t2.tv_sec - t1.tv_sec) == 20) {
//<playback for 20 seconds. then finish playback
        break;
    }

    ret = poll(&dec_fds, 1, 2000);

        //((Dual8181) make request for playback data, then will call DVR_DEC_QUEUE_GET and DVR_DEC_QUEUE_PUT from message handler
    ret = ioctl(dec_fd, DVR_DEC_QUEUE_GET, &data);
    backup_data = data.bs.length;
    if (ret < 0) {
        printf("buffer is not ready...\n");
        usleep(10000);
        continue;
    }

    buf = pbbs_buf + data.bs.offset;

    // read bs(start)
    buf_size = backup_data;
    reverse = 0;
    ret = GMAVIGetStreamDataAndIndex(pb_file,
&stream_id_pkt,
                buf, &buf_size, &intra, NULL, 0, i
, reverse, &pb_offset[pb_idx]);
    if((intra == 1) && (pb_idx<MAX_RECORD))
        pb_total = (pb_idx++);

    if (ret == GMSTS_END_OF_DATA) {
        printf("CH(%d,%d) - End of Playback!\n"
, ch_num, file_idx);

OPEN_PB_FILE AGAIN:
    close_pb_files(ch_num);

```

```

        ret = open_pb_files(ch_num, file_idx++);
    }
    if (ret < 0) {
        file_idx = 0;
        goto OPEN_PB_FILE AGAIN;
    }
    GMAVISeek(pb_file, GMAVI_SEEK_TO_BEGIN
NING, NULL);

//return job
data.bs.length = 0;
ret = ioctl(dec_fd, DVR_DEC_QUEUE_PUT,
&data);
if (ret < 0)
    printf("put failed when EOF...\n")
;

dec_ctrl.command = DEC_UPDATE;
dec_ctrl.src_param.dim.width = pb_main
_header.dwWidth;
dec_ctrl.src_param.dim.height = pb_main
_header.dwHeight;
dec_ctrl.src_param.win.x = 0;
dec_ctrl.src_param.win.y = 0;
dec_ctrl.src_param.win.width = pb_main
_header.dwWidth;
dec_ctrl.src_param.win.height = pb_main
_header.dwHeight;
dec_ctrl.src_param.bs_rate = (int) (10
00000/(pb_main_header.dwMicroSecPerFrame));
dec_ctrl.dst_param.plane_id = GMVAL_D0
_NOT_CARE;
dec_ctrl.dst_param.win.x = GMVAL_D0_NO
T_CARE;
dec_ctrl.dst_param.win.y = GMVAL_D0_NO
T_CARE;
dec_ctrl.dst_param.win.width = GMVAL_D

```

```
    O_NOT_CARE;
        dec_ctrl.dst_param.win.height = GMVAL_
DO_NOT_CARE;
        dec_ctrl.dst_param.is_display = GMVAL_
DO_NOT_CARE;
        dec_ctrl.dst_param.display_rate = GMVA
L_DO_NOT_CARE;

        ret = ioctl(dec_fd, DVR_DEC_CONTROL, &
dec_ctrl);
        if (ret < 0)
            printf("can't update playback para
meters!");

        FN_RESET_TAG(&tag);
        FN_SET_PB_CH(&tag, ch_num);
        ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
        buf_size = 0;
        continue;
    }
    else if (ret < 0)
        printf("CH(%d) - read error!\n", ch_nu
m);

    data.bs.length = buf_size;
    // read bs(end)
    ret = ioctl(dec_fd, DVR_DEC_QUEUE_PUT, &da
ta);
    if (ret < 0)
        printf("put failed...\n");

}

int do_pb_init(int ch_num)
{
```

```
int ret;
dvr_dec_channel_param    ch_param;

memset(&ch_param, 0x0, sizeof(dvr_dec_channel_
param));

ret = open_pb_files(ch_num, 0);
if (ret < 0)
    goto DO_PB_INIT_FAILED;

dec_fd = open("/dev/dvr_dec", O_RDWR);
if (dec_fd < 0) {
    perror("Open [/dev/dvr_dec] failed:");
    goto DO_PB_INIT_FAILED;
}

switch(*(int *)pb_stream_header.fccHandler) {

    case GMAVI_TYPE_H264:
        ch_param.dec_type = ENC_TYPE_H264;
        break;
    case GMAVI_TYPE_MPEG4:
        ch_param.dec_type = ENC_TYPE_MPEG;
        break;
    case GMAVI_TYPE_MJPEG:
        ch_param.dec_type = ENC_TYPE_MJPEG;
        break;
    default:
        printf("%s:%d <dec_type err. %d>\n",
FUNCTION__, __LINE__, ch_param.dec_type);
        return -1;
}

ch_param.channel = ch_num;
ch_param.is_use_scaler = 1;
ch_param.dec_param.output_type = DEC_OUTPUT_CO
```

```

LOR_YUV422;
    ch_param.scl_param.src_fmt = SCALE_YUV422;
    ch_param.scl_param.dst_fmt = SCALE_YUV422;
    ch_param.scl_param.scale_mode = SCALE_LINEAR;
    ch_param.scl_param.is_dither = 0;
    ch_param.scl_param.is_correction = 0;
    ch_param.scl_param.is_album = 1;
    ch_param.scl_param.des_level = 0;

    ret = ioctl(dec_fd, DVR_DEC_SET_CHANNEL_PARAM,
&ch_param);
    if (ret < 0) {
        perror("Decoder DVR_DEC_SET_CHANNLE_PARAM
failed:");
        goto DO_PB_INIT_FAILED;
    }

    ret = ioctl(dec_fd, DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE,
&dec_buf_size);
    if (ret < 0) {
        perror("Decoder DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE failed:");
        goto DO_PB_INIT_FAILED;
    }

    pbbs_buf = (unsigned char*) mmap(NULL, dec_buf_size,
PROT_READ|PROT_WRITE, MAP_SHARED, dec_fd, 0
);
    if (pbbs_buf == MAP_FAILED) {
        perror("Dec mmap failed");
        goto DO_PB_INIT_FAILED;
    }
    return 0;

DO_PB_INIT_FAILED:
    if (dec_fd) {
        if (pbbs_buf) {

```

```

        munmap((void*)pbbs_buf, dec_buf_size);
        pbbs_buf = NULL;
    }
    close(dec_fd);
    dec_fd = 0;
}
if (pb_file) {
    GMAVICClose(pb_file);
    pb_file = NULL;
}
return -1;
}

int do_pb_start(int ch_num)
{
    int ret;
    dvr_dec_control      dec_ctrl;
    FuncTag tag;

    memset(&dec_ctrl, 0x0, sizeof(dvr_dec_control));
};

if (!pb_file) {
    printf("Open file failed!\n");
    return -1;
}

dec_ctrl.command = DEC_START;
dec_ctrl.src_param.dim.width = pb_main_header.
dwWidth;
dec_ctrl.src_param.dim.height = pb_main_header.
dwHeight;
dec_ctrl.src_param.win.x = 0; //ROI, after d
ecoder, or the input to scalar
dec_ctrl.src_param.win.y = 0; //ROI, after d
ecoder, or the input to scalar
dec_ctrl.src_param.win.width = pb_main_header.

```

```

dwWidth;           //ROI, after decoder
    dec_ctrl.src_param.win.height = pb_main_header
.dwHeight;       //ROI, after decoder
    dec_ctrl.src_param.bs_rate=(int) (1000000/(pb_
main_header.dwMicroSecPerFrame));

    if(is_st_vga) {
        dec_ctrl.dst_param.win.x = VGA_4CH[ch_num].x;
        dec_ctrl.dst_param.win.y = VGA_4CH[ch_num].y;
    }
    else {
        dec_ctrl.dst_param.win.x = D1_4CH[ch_num].x;
        //final position of screen
        dec_ctrl.dst_param.win.y = D1_4CH[ch_num].y;
        //final position of screen
    }
    dec_ctrl.dst_param.win.width = VGA_4CH[ch_num].width;
    //final width in screen
    dec_ctrl.dst_param.win.height = VGA_4CH[ch_num].height;
    //final width in screen
    dec_ctrl.dst_param.plane_id = main_plane_id;
    dec_ctrl.dst_param.is_display = TRUE;
    dec_ctrl.dst_param.display_rate = (is_NTSC)? 3
0 : 25;

    ret = ioctl(dec_fd, DVR_DEC_CONTROL, &dec_ctrl);
    if (ret < 0)
        return -1;

    FN_RESET_TAG(&tag);
    FN_SET_PB_CH(&tag, ch_num);
    ret = ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    if (ret < 0)
        return -1;

```

```
    menu_func = FUNCTION_NORMAL;

    return 0;
}

int do_pb_stop(int ch_num)
{
    int ret;
    dvr_dec_control     dec_ctrl;
    FuncTag  tag;

    memset(&dec_ctrl, 0x0, sizeof(dvr_dec_control));
}

    menu_func = FUNCTION_NULL;
    dec_ctrl.command = DEC_STOP;
    if ((ret = ioctl(dec_fd, DVR_DEC_CONTROL, &dec
_ctrl)) < 0)
        goto pb_exit;

    FN_RESET_TAG(&tag);
    FN_SET_PB_CH(&tag, ch_num);
    if ((ret = ioctl(dvr_fd, DVR_COMMON_APPLY, &ta
g)) < 0)
        goto pb_exit;

pb_exit:
    return 0;
}

int do_pb_exit(int ch_num)
{
    if (dec_fd) {
        if (pbbs_buf) {
            munmap((void*)pbbs_buf, dec_buf_size);
            pbbs_buf = NULL;
        }
    }
}
```

```
        }
        close(dec_fd);
        dec_fd = 0;
    }
    close_pb_files(ch_num);

    return 0;
}

int main(void)
{
    //open dvr_common
    dvr_fd = open("/dev/dvr_common", O_RDWR);

    do_disp_startup();
    do_pb_init(0);
    do_pb_start(0);

    packet_reader(0);

    do_pb_stop(0);
    do_pb_exit(0);
    do_disp_endup();

    //close dvr_common
    close(dvr_fd);
    return 0;
}
```

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

roi.c

```
/**  
 * This sample code is for ROI function,  
 * main_bitstream record for 20 seconds H264 forma  
t.  
 * sub1_bitstream record for ROI(320x240)/ROI(640x4  
80)/disable_ROI, and dynamically change position p  
er second.  
 * ROI_win.y should be align at 2 pxls, ROI_win.x  
have no alignment limitation.  
 * ROI_win.width/ROI_win.height should be align at  
16,  
 * Please use gmdvr_mem_3_3_3_3.cfg for the buffer  
config.  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
#define ROI_COORDINATE_X_STEP 20  
#define ROI_COORDINATE_Y_STEP 20
```

```
int dvr_fd = 0;
int enc_fd = 0;

//test record
unsigned char *bs_buf;
HANDLE rec_file;
int enc_buf_size;
struct pollfd rec_fds;
char file_name[128];
int sub1_bs_buf_offset;
char sub_rec_filename[64];
HANDLE rec_sub_bs_file;

dvr_enc_channel_param ch_param;
ReproduceBitStream sub_ch_param;
EncParam_Ext5 enc_param_ext = {0};
dvr_enc_control enc_ctrl;
FuncTag tag;

dvr_enc_channel_param user_rec_ch_setting =
{
{
    0,                                /* channel number */
    ENC_TYPE_FROM_CAPTURE,
    {1280, 720},           /* channel 0 */
    LVFRAME_EVEN_ODD,
    LVFRAME_FRAME_MODE,
    DMAORDER_PACKET,
    CAPSCALER_NOT_KEEP_RATIO,
    MCP_VIDEO_NTSC,
    CAPCOLOR_YUV422,
    { FALSE, FALSE, GM3DI_FIELD }
},
{
    DVR_ENC_EBST_ENABLE,
    0,
```

```

        ENC_TYPE_H264,
        FALSE,
        DVR_ENC_EBST_DISABLE,
        {1280, 720}, /* channel 0 */
        {ENC_INPUT_H2642D, 30, 200*1000, 30, 25,
51, 1 , FALSE, {0, 0, 320, 240}},
        {SCALE_YUV422, SCALE_YUV422, SCALE_LINEAR,
FALSE, FALSE, TRUE, 0 },
        {JCS_yuv420, 0, JENC_INPUT_MP42D, 70}

    }
};

ReproduceBitStream    user_rec_sub_ch_setting =
{
    DVR_ENC_EBST_ENABLE, //enabled
    1, // sub1-bitstream
    ENC_TYPE_H264, //enc_type, 0: ENC_TYPE_H264, 1
:ENC_TYPE_MPEG, 2:ENC_TYPE_MJPEG,
    FALSE, // is_blocked
    DVR_ENC_EBST_DISABLE, // en_snapshot,
    {1280, 720}, /* channel 0 */
    {ENC_INPUT_H2642D, 30, 262144, 30, 25, 51, 1
, TRUE, {0, 0, 320, 240}}, //EncParam
    {SCALE_YUV422, SCALE_YUV422, SCALE_LINEAR, FAL
SE, FALSE, TRUE, 0 }, //ScalerParam
    {JCS_yuv420, 0, JENC_INPUT_MP42D, 70 } //snap
shot_param
};

void do_record_start(void)
{
    memcpy(&ch_param, &user_rec_ch_setting, sizeof
(ch_param)); //main-bitstream
    ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_
ADD_VAL(sizeof(enc_param_ext));
    ch_param.main_bs.enc.pext_data = &enc_param_ex
}

```

```
t;

    enc_param_ext.feature_enable = 0;

    ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_param);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE
, &enc_buf_size);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1
_BS_OFFSET, &sub1_bs_buf_offset);

    bs_buf = (unsigned char*) mmap(NULL, enc_buf_size,
PROT_READ|PROT_WRITE,
                                MAP_SHARED, enc_fd, 0);

    // sub1_bitstream ROI setting.
    user_rec_sub_ch_setting.enc.is_use_ROI=TRUE;
    user_rec_sub_ch_setting.enc.ROI_win.x = 0;
    user_rec_sub_ch_setting.enc.ROI_win.y = 0;
    user_rec_sub_ch_setting.enc.ROI_win.width = 32
0;
    user_rec_sub_ch_setting.enc.ROI_win.height = 2
40;
    memcpy(&sub_ch_param, &user_rec_sub_ch_setting
, sizeof(sub_ch_param)); //sub1-roi
    ioctl(enc_fd, DVR_ENC_SET_SUB_BS_PARAM, &sub_c
h_param);

    //record start
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control))
);
    enc_ctrl.command = ENC_START;
    enc_ctrl.stream = 0;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
```

```

    enc_ctrl.command = ENC_START;
    enc_ctrl.stream = sub_ch_param.out_bs;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    // set function tag parameter to dvr graph lev
el
    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_param.src.channel);
    FN_SET_SUB1_REC_CH(&tag, ch_param.src.channel)
;
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
}

void do_record_stop(void)
{
    //record stop
    enc_ctrl.stream = sub_ch_param.out_bs;
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    enc_ctrl.stream = 0; // 0: Stop for all main a
nd sub
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_param.src.channel);
    FN_SET_SUB1_REC_CH(&tag, ch_param.src.channel)
;
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    munmap((void*)bs_buf, enc_buf_size);
}

void do_roi_coordinate_win_update(int ch_num, int
ROI_win_x, int ROI_win_y, int ROI_win_w, int ROI_w
in_h)

```

```

{
    dvr_enc_control enc_update;
    EncParam_Ext5 enc_param_ext = {0};
    int window_w, window_h;

    window_w = ROI_win_x + ROI_win_w;
    window_h = ROI_win_y + ROI_win_h;
    if((window_w > sub_ch_param.dim.width) || (window_h > sub_ch_param.dim.height)) {
        printf("%s: ROI window over! (x=%d, y=%d, width=%d, height=%d) (%dx%d).\n",
               __FUNCTION__,
               ROI_win_x,
               ROI_win_y,
               ROI_win_w,
               ROI_win_h,
               sub_ch_param.dim.width
        ,
               sub_ch_param.dim.height
    );
        return;
    }
    if(ROI_win_y%2) {
        printf("%s: <ROI_win_y=%d should be align at 2 pxls.>\n",
               __FUNCTION__,
               ROI_win_y);
        return;
    }
    printf("ROI_win_x=%d, ROI_win_y=%d \n",
           ROI_win_x,
           ROI_win_y);
    memset(&enc_update, 0x0, sizeof(dvr_enc_control));
}

    enc_update.stream = 1; /* sub1-bistream update */
    enc_update.update_parm.stream_enable = 1;
    enc_update.update_parm.frame_rate = GMVAL_DO_NOT_CARE;
}

```

```
    enc_update.update_parm.bit_rate = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.ip_interval = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.dim.width = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.dim.height = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.di_mode = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.mode = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.scale_indep = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.is_3DI = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.is_denoise = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.init_quant = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.max_quant = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.min_quant = GMVAL_DO_NOT_CARE;

    enc_update.update_parm.ext_size = DVR_ENC_MAGIC_ADD_VAL(sizeof(enc_param_ext));
    enc_update.update_parm.pext_data = &enc_param_ext;

    enc_param_ext.target_rate_max = GMVAL_DO_NOT_CARE;
    enc_param_ext.reaction_delay_max = GMVAL_DO_NOT_CARE;
    enc_param_ext.enc_type = GMVAL_DO_NOT_CARE;
    enc_param_ext.MJ_quality = GMVAL_DO_NOT_CARE;
    enc_param_ext.watermark_enable = GMVAL_DO_NOT_CARE;
```

```

CARE;
    enc_param_ext.watermark_interval = GMVAL_DO_NOT_CARE;
    enc_param_ext.watermark_init_pattern = GMVAL_D0_NOT_CARE;
    enc_param_ext.watermark_init_interval = GMVAL_D0_NOT_CARE;

    // ROI update
    enc_param_ext.feature_enable |= DVR_ENC_ROI_ALL;
    enc_param_ext.roi_all.is_use_ROI = GMVAL_D0_NOT_CARE;
    enc_param_ext.roi_all.win.width = ROI_win_w;
    enc_param_ext.roi_all.win.height = ROI_win_h;
    enc_param_ext.roi_all.win.x = ROI_win_x;
    enc_param_ext.roi_all.win.y = ROI_win_y;
    enc_update.command = ENC_UPDATE;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_update);

    FN_RESET_TAG(&tag);
    FN_SET_SUB1_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);

}

void do_roi_enable_update(int ch_num, int is_use_ROI)
{
    dvr_enc_control enc_update;
    EncParam_Ext5 enc_param_ext = {0};

    printf("is_use_ROI=%d \n", is_use_ROI);
    memset(&enc_update, 0x0, sizeof(dvr_enc_control));
};

    enc_update.stream = 1; /* sub1-bistream updat

```

```
e */

    enc_update.update_parm.stream_enable = 1;
    enc_update.update_parm.frame_rate = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.bit_rate = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.ip_interval = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.dim.width = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.dim.height = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.di_mode = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.mode = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.scale_indep = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.is_3DI = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.src.is_denoise = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.init_quant = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.max_quant = GMVAL_DO_NOT_CARE;
    enc_update.update_parm.min_quant = GMVAL_DO_NOT_CARE;

    enc_update.update_parm.ext_size = DVR_ENC_MAGIC_ADD_VAL(sizeof(enc_param_ext));
    enc_update.update_parm.pext_data = &enc_param_ext;

    enc_param_ext.target_rate_max = GMVAL_DO_NOT_CARE;
```

```

    enc_param_ext.reaction_delay_max = GMVAL_DO_NOT_CARE;
    enc_param_ext.enc_type = GMVAL_DO_NOT_CARE;
    enc_param_ext.MJ_quality = GMVAL_DO_NOT_CARE;
    enc_param_ext.watermark_enable = GMVAL_DO_NOT_CARE;
    enc_param_ext.watermark_interval = GMVAL_DO_NOT_CARE;
    enc_param_ext.watermark_init_pattern = GMVAL_DO_NOT_CARE;
    enc_param_ext.watermark_init_interval = GMVAL_DO_NOT_CARE;

    // ROI update
    enc_param_ext.feature_enable |= DVR_ENC_ROI_ALL;
    enc_param_ext.roi_all.is_use_ROI = is_use_ROI;
    enc_param_ext.roi_all.win.width = GMVAL_DO_NOT_CARE;
    enc_param_ext.roi_all.win.height = GMVAL_DO_NOT_CARE;
    enc_param_ext.roi_all.win.x = GMVAL_DO_NOT_CAR_E;
    enc_param_ext.roi_all.win.y = GMVAL_DO_NOT_CAR_E;

    enc_update.command = ENC_UPDATE;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_update);

    FN_RESET_TAG(&tag);
    FN_SET_SUB1_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);

}

void do_roi_coordinate_update(int ch_num, int ROI_win_x, int ROI_win_y)

```

```

{
    dvr_enc_control enc_update;
    EncParam_Ext5 enc_param_ext = {0};
    int window_w, window_h;

    window_w = ROI_win_x + sub_ch_param.enc.ROI_win
.width;
    window_h = ROI_win_y + sub_ch_param.enc.ROI_win
.height;
    if((window_w > sub_ch_param.dim.width) || (win
dow_h > sub_ch_param.dim.height)) {
        printf("%s: ROI window over! (x=%d, y=%d,
width=%d, height=%d) (%dx%d).\n", __FUNCTION__,
ROI_win_x,
ROI_win_y,
sub_ch_param.enc.ROI_W
in.width,
sub_ch_param.enc.ROI_W
in.height,
sub_ch_param.dim.width
,
sub_ch_param.dim.height
);
        return;
    }
    if(ROI_win_y%2) {
        printf("%s: <ROI_win_y=%d should be align
at 2 pxls.>\n", __FUNCTION__, ROI_win_y);
        return;
    }
    printf("ROI_win_x=%d, ROI_win_y=%d \n", ROI_wi
n_x, ROI_win_y);
    memset(&enc_update, 0x0, sizeof(dvr_enc_control
));

    enc_update.stream = 1; /* sub1-bistream updat
e */
}

```

```
    enc_update.update_parm.stream_enable = 1;
    enc_update.update_parm.frame_rate = GMVAL_DO_N
OT_CARE;
    enc_update.update_parm.bit_rate = GMVAL_DO_NOT
_CARE;
    enc_update.update_parm.ip_interval = GMVAL_DO_
NOT_CARE;
    enc_update.update_parm.dim.width = GMVAL_DO_NO
T_CARE;
    enc_update.update_parm.dim.height = GMVAL_DO_N
OT_CARE;
    enc_update.update_parm.src.di_mode = GMVAL_DO_
NOT_CARE;
    enc_update.update_parm.src.mode = GMVAL_DO_NOT
_CARE;
    enc_update.update_parm.src.scale_indep = GMVAL_
DO_NOT_CARE;
    enc_update.update_parm.src.is_3DI = GMVAL_DO_N
OT_CARE;
    enc_update.update_parm.src.is_denoise = GMVAL_
DO_NOT_CARE;
    enc_update.update_parm.init_quant = GMVAL_DO_N
OT_CARE;
    enc_update.update_parm.max_quant = GMVAL_DO_NO
T_CARE;
    enc_update.update_parm.min_quant = GMVAL_DO_NO
T_CARE;

    enc_update.update_parm.ext_size = DVR_ENC_MAGI
C_ADD_VAL(sizeof(enc_param_ext));
    enc_update.update_parm.pext_data = &enc_param_
ext;

    enc_param_ext.target_rate_max = GMVAL_DO_NOT_C
ARE;
    enc_param_ext.reaction_delay_max = GMVAL_DO_NO
```

```

T_CARE;
    enc_param_ext.enc_type = GMVAL_DO_NOT_CARE;
    enc_param_ext.MJ_quality = GMVAL_DO_NOT_CARE;
    enc_param_ext.watermark_enable = GMVAL_DO_NOT_
CARE;
    enc_param_ext.watermark_interval = GMVAL_DO_NO
T_CARE;
    enc_param_ext.watermark_init_pattern = GMVAL_D
O_NOT_CARE;
    enc_param_ext.watermark_init_interval = GMVAL_
DO_NOT_CARE;

    // ROI update
    enc_param_ext.feature_enable |= DVR_ENC_ROI_AL
L;
    enc_param_ext.roi_all.is_use_ROI = GMVAL_DO_NO
T_CARE;
    enc_param_ext.roi_all.win.width = GMVAL_DO_NOT
_CARE;
    enc_param_ext.roi_all.win.height = GMVAL_DO_NO
T_CARE;
    enc_param_ext.roi_all.win.x = ROI_win_x;
    enc_param_ext.roi_all.win.y = ROI_win_y;

    enc_update.command = ENC_UPDATE;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_update);

    FN_RESET_TAG(&tag);
    FN_SET_SUB1_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);

}

/***
 * @brief main function
 * @return 0 on success, !0 on error
 */

```

```

int main(int argc, char *argv[])
{
    int ret = 0, ch_num, sub_num = 1;
    dvr_enc_queue_get    data;
    unsigned char *buf;
    int buf_size;
    struct timeval t1,t2;
    char tmp_str[128];
    int count = 0, ROI_win_x = 0, ROI_win_y = 0, ROI_win_w = 0, ROI_win_h = 0, is_use_ROI=1;

    dvr_fd = open("/dev/dvr_common", O_RDWR);      // open_dvr_common
    enc_fd = open("/dev/dvr_enc", O_RDWR);          // open_dvr_encode
    ch_num = user_rec_ch_setting.src.channel;

    do_record_start();

    //main bitstream
    sprintf(file_name, "CH%d_video_%d", ch_num, 0)
    ;
    sprintf(tmp_str, "%s.h264", file_name);

    rec_file = fopen ( tmp_str , "wb+" );

    //sub1 bitstream
    sprintf(sub_rec_filename, "CH%d_Sub%d_Video_%03d", ch_num, sub_num, 001);
    sprintf(tmp_str, "%s.h264", sub_rec_filename);

    rec_sub_bs_file = fopen ( tmp_str , "wb+" );

    gettimeofday(&t1, NULL);

    while(1) {
        // prepare to select(or poll)

```

```

        rec_fds.fd = enc_fd;
        rec_fds.revents = 0;
        rec_fds.events = (POLLIN_MAIN_BS | POLLIN_
SUB1_BS);

        poll(&rec_fds, 1, 500);

        if (rec_fds.revents & POLLIN_SUB1_BS) {

            ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET_
SUB1_BS, &data);
            if (ret >= 0) {
                buf = bs_buf + sub1_bs_buf_offset
+ data.bs.offset;
                buf_size = data.bs.length;
                if(data.new_bs == 1) {
                    fclose(rec_sub_bs_file);

                    sprintf(sub_rec_filename, "CH%
d_Sub%d_Video_%03d", ch_num, sub_num, count);
                    sprintf(tmp_str, "%s.h264", su
b_rec_filename);
                    rec_sub_bs_file = fopen ( tmp_
str , "wb+" );
                    printf("%s:%d <file=%s>\n", __F
UNCTION__, __LINE__, tmp_str);
                }

                fwrite (buf , 1 , buf_size , rec_s
ub_bs_file);
                fflush(rec_sub_bs_file);
                ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
            }
        }

        if (!(rec_fds.revents & POLLIN_MAIN_BS))

```

```

        continue;

    // get dataout buffer
    ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET, &data);
    if(ret < 0)
        continue;

    buf = bs_buf + data.bs.offset;
    buf_size = data.bs.length;

    fwrite (buf , 1 , buf_size , rec_file);
    fflush(rec_file);
    ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data);

    gettimeofday(&t2, NULL);

    if ((t2.tv_sec - t1.tv_sec) == 1) {
        if(count == 5) { //< update roi coordinate x, y, width and height
            ROI_win_x = ROI_win_y = 0;
            ROI_win_w = 640;
            ROI_win_h = 480;
            do_roi_coordinate_win_update(ch_num, ROI_win_x, ROI_win_y, ROI_win_w, ROI_win_h);
        } else if (count == 10) {
            is_use_ROI = 0; //< disable ROI
            do_roi_enable_update(ch_num, is_use_ROI);
        } else if (count == 15) {
            is_use_ROI = 1; //< enable ROI
            do_roi_enable_update(ch_num, is_use_ROI);
        } else { //< update roi coordinate x and y,
    }
}

```

```

        if(is_use_ROI) {
            do_roi_coordinate_update(ch_nu
m, ROI_win_x, ROI_win_y);
            ROI_win_x += ROI_COORDINATE_X_
STEP;
            ROI_win_y += ROI_COORDINATE_Y_
STEP;
        }
    }
    t1.tv_sec = t2.tv_sec;
    ++count;
}
if (count >= 20) {      //< record for 10
seconds. then finish record.
    fclose(rec_file);
    fclose(rec_sub_bs_file);
    break;
}
}

do_record_stop();

printf("-----\n");
printf(" Record finish\n");
printf("-----\n");
;

close(enc_fd);          //close_dvr_encode
close(dvr_fd);          //close_dvr_common
return 0;
}

```

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

snapshot.c

```
/**  
 * this sample code implement snapshot function, a  
nd record for 15 seconds.  
 * trigger snapshot at 5th seconds, and do snapsho  
t 3 times.  
 *  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;  
int enc_fd = 0;  
int enc_fd_channel[DVR_RECORD_CHANNEL_NUM] = {0};  
  
//test record  
unsigned char *bs_buf;  
HANDLE rec_file;  
int avi_str_id;  
int enc_buf_size;
```

```
struct pollfd rec_fds;
char file_name[128];

int sub1_bs_buf_offset;
int sub2_bs_buf_offset;
int bs_buf_snap_offset;
int encode_frame_count = 0;

dvr_enc_channel_param ch_param;
EncParam_Ext3 enc_param_ext = {0};
dvr_enc_control enc_ctrl;
FuncTag tag;

dvr_enc_channel_param user_rec_ch_setting =
{
{
    0,                                /* channel number */
    ENC_TYPE_FROM_CAPTURE,
    {1280, 720},           /* channel 0 */
//    {640, 480},            /* channel 1 */
//    {320, 240},            /* channel 2 */
//    {160, 112},             /* channel 3 */
    LVFRAME_EVEN_ODD,
    LVFRAME_FRAME_MODE,
    DMAORDER_PACKET,
    CAPSCALER_NOT_KEEP_RATIO,
    MCP_VIDEO_NTSC,
    CAPCOLOR_YUV422,
    { FALSE, FALSE, GM3DI_FIELD }
},
{
    DVR_ENC_EBST_ENABLE,
    0,
    ENC_TYPE_H264,
    FALSE,
    DVR_ENC_EBST_ENABLE,
    {1280, 720},           /* channel 0 */
}
```



```
//record start
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control))
);
enc_ctrl.command = ENC_START;
enc_ctrl.stream = 0;
ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

// set function tag parameter to dvr graph lev
el
FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_param.src.channel);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
}

void do_record_stop(void)
{
//record stop
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control))
);
enc_ctrl.stream = 0;
enc_ctrl.command = ENC_STOP;
ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_param.src.channel);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
munmap((void*)bs_buf, enc_buf_size);
}

/**
 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num, i;
    dvr_enc_queue_get    data;
```

```
unsigned char *buf;
int buf_size;
struct timeval t1,t2;
char tmp_str[128];
int snap_no = 3, flag = 0;
int flag_snapshot = 0;

dvr_fd = open("/dev/dvr_common", O_RDWR); //  
open_dvr_common

for(i=0; i<DVR_RECORD_CHANNEL_NUM; i++ ) {
    enc_fd_channel[i] = open("/dev/dvr_enc", O_RDWR); //open_dvr_encode
}
ch_num = user_rec_ch_setting.src.channel;
enc_fd = enc_fd_channel[ch_num];

do_record_start();

sprintf(file_name, "CH%d_video_%d", ch_num, 0)
;
sprintf(tmp_str, "%s.h264", file_name);

rec_file = fopen ( tmp_str , "wb+" );
gettimeofday(&t1, NULL);

while(1) {
    // prepare to select(or poll)
    rec_fds.fd = enc_fd;
    rec_fds.revents = 0;
    rec_fds.events = (POLLIN_MAIN_BS | POLLIN_SNAP_BS);

    poll(&rec_fds, 1, 500);

    if (rec_fds.revents & POLLIN_SNAP_BS) {
```

```
        ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET_SNAP, &data);
        if (ret >= 0) {
            FILE * fd;

            buf = bs_buf + bs_buf_snap_offset
+ data.bs.offset;
            buf_size = data.bs.length;

            -- flag_snapshot;

            sprintf(tmp_str, "CH%d_snapshot%02d_%03d.jpg", ch_num, flag_snapshot, encode_frame_count);
            fd = fopen ( tmp_str , "wb+" );
            fwrite(buf , 1 , buf_size , fd);

            fclose(fd);

            printf("snapshot: output file %s\n"
, tmp_str);
            ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
        }
    }

    if (!(rec_fds.revents & POLLIN_MAIN_BS))
        continue;

    // get dataout buffer
    ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET, &da
ta);
    if(ret < 0)
        continue;

    encode_frame_count ++;
```

```

        buf = bs_buf + data.bs.offset;
        buf_size = data.bs.length;

        fwrite (buf , 1 , buf_size , rec_file);
        fflush(rec_file);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data);

        gettimeofday(&t2, NULL);

        if ((t2.tv_sec - t1.tv_sec) == 5) {           /
// trigger snapshot at 5th seconds,
            if (flag == 0){                         /
// and do snapshot 3 times.
                flag = 1;
                memset(&enc_ctrl, 0x0, sizeof(dvr_
enc_control));
                enc_ctrl.command = ENC_SNAP;
                flag_snapshot = snap_no;
                enc_ctrl.output.count = (int *)sna
p_no;      //< do snapshot 3 times.
                ioctl(enc_fd, DVR_ENC_CONTROL, &en
c_ctrl);
            }
        }

        if ((t2.tv_sec - t1.tv_sec) == 15) {           /
// record for 15 seconds. then finish record.

            fclose(rec_file);
            break;
        }
    }

    do_record_stop();

    printf("-----\n")

```

```
;           printf(" Record finish\n");
printf("-----\n")
;

for(i=0; i<DVR_RECORD_CHANNEL_NUM; i++ ) {
    close(enc_fd_channel[i]);           //close_dvr
_encode
}

close(dvr_fd);           //close_dvr_common

return 0;
}
```

Generated on Wed Jun 15 2011 15:50:59 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

sub-bitstream-record.c

```
/**  
 * this sample code implement main bitstream + sub  
1 bitstream + sub2 bitstream record function  
 * and record for 20 seconds.  
 * (1) for H264 record, file format please indicate xxx.h264  
 *      for MPEG record, file format please indicate xxx.m4v  
 *      for MOTION JPEG record, file format please indicate xxx.jpg  
 * (2) for H264 record, max_quant < 51, 1 < min_q  
uant  
 *      for MPEG record, max_quant < 31, 1 < min_q  
uant  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
#include <poll.h>  
#include <sys/ioctl.h>  
#include <sys/time.h>  
  
#include "dvr_common_api.h"  
#include "dvr_enc_api.h"  
#include "gmavi_api.h"  
  
int dvr_fd = 0;
```

```
int enc_fd = 0;

//test record
unsigned char *bs_buf;
HANDLE rec_file;
int enc_buf_size;
int sub1_bs_buf_offset;
int sub2_bs_buf_offset;
int bs_buf_snap_offset;
struct pollfd rec_fds;
char file_name[128];

//sub record
char sub_rec_filename[DVR_ENC_REPD_BT_NUM - 1][64]
;
int multi_bitstream = 0;
HANDLE rec_sub_bs_file[DVR_ENC_REPD_BT_NUM - 1];

/***
 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num = 0, sub1_num = 1, sub2_num = 2;
    dvr_enc_channel_param ch_param;
    ReproduceBitStream sub1_param;
    ReproduceBitStream sub2_param;
    EncParam_Ext3 enc_param_ext = {0};
    EncParam_Ext3 enc_param_ext1 = {0};
    EncParam_Ext3 enc_param_ext2 = {0};
    dvr_enc_control enc_ctrl;
    dvr_enc_queue_get data;
    unsigned char *buf;
    int buf_size;
    FuncTag tag;
```

```
    struct timeval t1,t2;
    char tmp_str[128];

    dvr_fd = open("/dev/dvr_common", O_RDWR);      //  
open_dvr_common

    enc_fd = open("/dev/dvr_enc", O_RDWR);          //  
open_dvr_encode

    //set dvr encode source parameter
    ch_param.src.input_system = MCP_VIDEO_NTSC;
    ch_param.src.channel = ch_num;
    ch_param.src.enc_src_type = ENC_TYPE_FROM_CAPT
URE;

    ch_param.src.dim.width = 1280;
    ch_param.src.dim.height = 720;

    ch_param.src.di_mode = LVFRAME_EVEN_ODD;
    ch_param.src.mode = LVFRAME_FRAME_MODE;
    ch_param.src.dma_order = DMAORDER_PACKET;
    ch_param.src.scale_indep = CAPSCALER_NOT_KEEP_
RATIO;
    ch_param.src.input_system = MCP_VIDEO_NTSC;
    ch_param.src.color_mode = CAPCOLOR_YUV422;

    ch_param.src.vp_param.is_3DI = FALSE;
    ch_param.src.vp_param.is_denoise = FALSE;
    ch_param.src.vp_param.denoise_mode = GM3DI_FIE
LD;
    //////////////////////////////
///////////////
    //set dvr encode main bitstream parameter
    ch_param.main_bs.enabled = DVR_ENC_EBST_ENABLE
;
    ch_param.main_bs.out_bs = 0;
    ch_param.main_bs.enc_type = ENC_TYPE_H264;
```

```
    ch_param.main_bs.is_blocked = FALSE;
    ch_param.main_bs.en_snapshot = DVR_ENC_EBST_DI
SABLE;

    ch_param.main_bs.dim.width = 1280;
    ch_param.main_bs.dim.height = 720;

    //set main bitstream encode parameter
    ch_param.main_bs.enc.input_type = ENC_INPUT_H2
642D;
    ch_param.main_bs.enc.frame_rate = 30;
    ch_param.main_bs.enc.bit_rate = 1048576;
    ch_param.main_bs.enc.ip_interval = 30;
    ch_param.main_bs.enc.init_quant = 25;
    ch_param.main_bs.enc.max_quant = 51;
    ch_param.main_bs.enc.min_quant = 1;
    ch_param.main_bs.enc.is_use_ROI = FALSE;
    ch_param.main_bs.enc.ROI_win.x = 0;
    ch_param.main_bs.enc.ROI_win.y = 0;
    ch_param.main_bs.enc.ROI_win.width = 320;
    ch_param.main_bs.enc.ROI_win.height = 240;

    //set main bitstream scalar parameter
    ch_param.main_bs.scl.src_fmt = SCALE_YUV422;
    ch_param.main_bs.scl.dst_fmt = SCALE_YUV422;
    ch_param.main_bs.scl.scale_mode = SCALE_LINEAR
;

    ch_param.main_bs.scl.is_dither = FALSE;
    ch_param.main_bs.scl.is_correction = FALSE;
    ch_param.main_bs.scl.is_album = TRUE;
    ch_param.main_bs.scl.des_level = 0;

    //set main bitstream snapshot parameter
    ch_param.main_bs.snap.sample = JCS_yuv420;
    ch_param.main_bs.snap.RestartInterval = 0;
    ch_param.main_bs.snap.u82D = JENC_INPUT_MP42D;
```

```
ch_param.main_bs.snap.quality = 70;

//associate the ext. structure

ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_
ADD_VAL(sizeof(enc_param_ext));
ch_param.main_bs.enc.pext_data = &enc_param_ex
t;

enc_param_ext.feature_enable = 0;           //CBR

///////////////////////////////
/////////////////////////////
//sub1 bitstream
memset(&sub1_param, 0x0, sizeof(ReproduceBitSt
ream));

sub1_param.enabled = DVR_ENC_EBST_ENABLE;
sub1_param.out_bs = 1;
sub1_param.enc_type = ENC_TYPE_H264;
sub1_param.is_blocked = FALSE;
sub1_param.en_snapshot = DVR_ENC_EBST_DISABLE;

sub1_param.dim.width = 1280;
sub1_param.dim.height = 720;

sub1_param.enc.input_type = ENC_INPUT_H2642D;
sub1_param.enc.frame_rate = 30;
sub1_param.enc.bit_rate = 262144;
sub1_param.enc.ip_interval = 15;
sub1_param.enc.init_quant = 25;
sub1_param.enc.max_quant = 51;
sub1_param.enc.min_quant = 1;
sub1_param.enc.is_use_ROI = FALSE;
sub1_param.enc.ROI_win.x = 0;
sub1_param.enc.ROI_win.y = 0;
sub1_param.enc.ROI_win.width = 320;
```

```

    sub1_param.enc.ROI_win.height = 240;

    //set sub1 bitstream scalar parameter
    sub1_param.scl.src_fmt = SCALE_YUV422;
    sub1_param.scl.dst_fmt = SCALE_YUV422;
    sub1_param.scl.scale_mode = SCALE_LINEAR;
    sub1_param.scl.is_dither = FALSE;
    sub1_param.scl.is_correction = FALSE;
    sub1_param.scl.is_album = TRUE;
    sub1_param.scl.des_level = 0;

    //set sub1 bitstream snapshot parameter
    sub1_param.snap.sample = JCS_yuv420;
    sub1_param.snap.RestartInterval = 0;
    sub1_param.snap.u82D = JENC_INPUT_MP42D;
    sub1_param.snap.quality = 70;
    //associate the ext. structure

    sub1_param.enc.ext_size = DVR_ENC_MAGIC_ADD_VAL
(sizeof(enc_param_ext1));
    sub1_param.enc.pext_data = &enc_param_ext1;

    enc_param_ext1.feature_enable &= ~DVR_ENC_MJPE
G_FUNCTION;

    /////////////////////////////////
    ///////////////////////////////
    //sub2 bitstream
    memset(&sub2_param, 0x0, sizeof(ReproduceBitSt
ream));

    sub2_param.enabled = DVR_ENC_EBST_ENABLE;
    sub2_param.out_bs = 2;
    sub2_param.enc_type = ENC_TYPE_H264;
    sub2_param.is_blocked = FALSE;
    sub2_param.en_snapshot = DVR_ENC_EBST_DISABLE;

```

```

sub2_param.dim.width = 1280;
sub2_param.dim.height = 720;

sub2_param.enc.input_type = ENC_INPUT_H2642D;
sub2_param.enc.frame_rate = 30;
sub2_param.enc.bit_rate = 131072;
sub2_param.enc.ip_interval = 15;
sub2_param.enc.init_quant = 25;
sub2_param.enc.max_quant = 51;
sub2_param.enc.min_quant = 1;
sub2_param.enc.is_use_ROI = FALSE;
sub2_param.enc.ROI_win.x = 0;
sub2_param.enc.ROI_win.y = 0;
sub2_param.enc.ROI_win.width = 320;
sub2_param.enc.ROI_win.height = 240;

//set sub2 bitstream scalar parameter
sub2_param.scl.src_fmt = SCALE_YUV422;
sub2_param.scl.dst_fmt = SCALE_YUV422;
sub2_param.scl.scale_mode = SCALE_LINEAR;
sub2_param.scl.is_dither = FALSE;
sub2_param.scl.is_correction = FALSE;
sub2_param.scl.is_album = TRUE;
sub2_param.scl.des_level = 0;

//set sub2 bitstream snapshot parameter
sub2_param.snap.sample = JCS_yuv420;
sub2_param.snap.RestartInterval = 0;
sub2_param.snap.u82D = JENC_INPUT_MP42D;
sub2_param.snap.quality = 70;
//associate the ext. structure

sub2_param.enc.ext_size = DVR_ENC_MAGIC_ADD_VAL
(sizeof(enc_param_ext2));
sub2_param.enc.pext_data = &enc_param_ext2;

```

```
    enc_param_ext2.feature_enable &= ~DVR_ENC_MJPE_G_FUNCTION;

    ///////////////////////////////////////////////////
///////////////////////////////////////////////////

    ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_param);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE,
          &enc_buf_size);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1_BS_OFFSET,
          &sub1_bs_buf_offset);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB2_BS_OFFSET,
          &sub2_bs_buf_offset);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP_OFFSET,
          &bs_buf_snap_offset);

    bs_buf = (unsigned char*) mmap(NULL, enc_buf_size,
                                  PROT_READ|PROT_WRITE,
                                  MAP_SHARED, enc_fd, 0);

    ioctl(enc_fd, DVR_ENC_SET_SUB_BS_PARAM, &sub1_param);

    ioctl(enc_fd, DVR_ENC_SET_SUB_BS_PARAM, &sub2_param);
    ///////////////////////////////////////////////////
///////////////////////////////////////////////////

    //main bitstream record start
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
    enc_ctrl.command = ENC_START;
```

```
    enc_ctrl.stream = 0;
    ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl
);
    // set function tag parameter to dvr graph lev
el
    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    /////////////////////////////////
/////////////////
    multi_bitstream = DVR_ENC_EBST_ENABLE;
    //sub1 bitstream record start
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control)
);
    enc_ctrl.command = ENC_START;
    enc_ctrl.stream = 1;
    ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl
);
    // set function tag parameter to dvr graph lev
el
    FN_RESET_TAG(&tag);
    FN_SET_SUB1_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    /////////////////////////////////
/////////////////
    //sub2 bitstream record start
    memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control)
);
    enc_ctrl.command = ENC_START;
    enc_ctrl.stream = 2;
    ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl
);
    // set function tag parameter to dvr graph lev
el
```

```

FN_RESET_TAG(&tag);
FN_SET_SUB2_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
///////////
//main bitstream
sprintf(file_name, "CH%d_video_%d", 0, 0);

sprintf(tmp_str, "%s.h264", file_name);

rec_file = fopen ( tmp_str , "wb+" );
///////////
//sub1 bitstream
sprintf(sub_rec_filename[sub1_num - 1], "CH%d_Sub%d_Video_%03d", ch_num, sub1_num, 001);
sprintf(tmp_str, "%s.h264", sub_rec_filename[sub1_num - 1]);
rec_sub_bs_file[sub1_num - 1] = fopen ( tmp_st
r , "wb+" );
///////////
//sub2 bitstream
sprintf(sub_rec_filename[sub2_num - 1], "CH%d_Sub%d_Video_%03d", ch_num, sub2_num, 001);
sprintf(tmp_str, "%s.h264", sub_rec_filename[s
ub2_num - 1]);
rec_sub_bs_file[sub2_num - 1] = fopen ( tmp_st
r , "wb+" );
///////////
getttimeofday(&t1, NULL);

while(1) {
    // prepare to select(or poll)
    rec_fds.fd = enc_fd;
    rec_fds.revents = 0;

```

```
    rec_fds.events = POLLIN_MAIN_BS;

    if (multi_bitstream == DVR_ENC_EBST_EN
ABLE)
        rec_fds.events |= (POLLIN_SUB1_BS
| POLLIN_SUB2_BS);

    poll(&rec_fds, 1, 500);

    if (rec_fds.revents & POLLIN_SUB1_BS)
    {
        ret = ioctl(enc_fd, DVR_ENC_QUEUE_
GET_SUB1_BS, &data);
        if(ret < 0)
            continue;

        buf = bs_buf + sub1_bs_buf_offset
+ data.bs.offset;
        buf_size = data.bs.length;

        fwrite (buf , 1 , buf_size , rec_s
ub_bs_file[sub1_num - 1]);
        fflush(rec_sub_bs_file[sub1_num -
1]);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
    }

    if (rec_fds.revents & POLLIN_SUB2_BS)
{
        ret = ioctl(enc_fd, DVR_ENC_QUEUE_
GET_SUB2_BS, &data);
        if(ret < 0)
            continue;

        buf = bs_buf + sub2_bs_buf_offset
+ data.bs.offset;
```

```

        buf_size = data.bs.length;

        fwrite (buf , 1 , buf_size , rec_s
ub_bs_file[sub2_num - 1]);
        fflush(rec_sub_bs_file[sub2_num -
1]);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
    }

    if (!(rec_fds.revents & POLLIN_MAIN_BS
))
        continue;

// get dataout buffer
ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET,
&data);
if(ret < 0)
    continue;

buf = bs_buf + data.bs.offset;
buf_size = data.bs.length;

ret = fwrite (buf , 1 , buf_size , rec
_file);
fflush(rec_file);
ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data
);

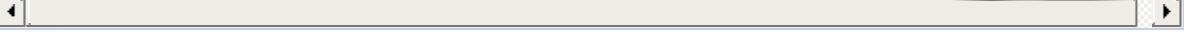
gettimeofday(&t2, NULL);

if ((t2.tv_sec - t1.tv_sec) == 20) {
//<record for 20 seconds. then finish record.

    fclose(rec_file);
    fclose(rec_sub_bs_file[sub1_num -
1]);
}

```

```
        fclose(rec_sub_bs_file[sub2_num -  
1]);  
    break;  
}  
}  
  
//record stop  
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control))  
);  
enc_ctrl.stream = 0; // for all main and su  
b  
enc_ctrl.command = ENC_STOP;  
ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);  
  
FN_RESET_TAG(&tag);  
FN_SET_REC_CH(&tag, ch_num);  
FN_SET_SUB1_REC_CH(&tag, ch_num);  
FN_SET_SUB2_REC_CH(&tag, ch_num);  
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);  
munmap((void*)bs_buf, enc_buf_size);  
  
printf("-----\n");  
;  
printf(" Record finish\n");  
printf("-----\n");  
;  
  
close(enc_fd); //close_dvr_encode  
close(dvr_fd); //close_dvr_common  
  
return 0;  
}
```



Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
doxygen 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

update-bitrate.c

```
/**  
 * this sample code implement main bitstream update bitrate only record function  
 * after record 10 seconds, update main bitstream bit rate.  
 * and record for 20 seconds.  
 *  
 * (1) for H264 record, file format please indicate xxx.h264  
 *      for MPEG record, file format please indicate xxx.m4v  
 *      for MOTION JPEG record, file format please indicate xxx.jpg  
 * (2) for H264 record, max_quant < 51, 1 < min_quant  
 *      for MPEG record, max_quant < 31, 1 < min_quant  
 * (3) for example, if you only update "frame_rate"  
 *,  
 *      paramter "bit_rate" don't need to update,  
 *      please assign enc_parm.update_parm.bit_rate = GMVAL_DO_NOT CARE;  
 *      and so on.  
 */  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>
```

```
#include <sys/mman.h>
#include <poll.h>
#include <sys/ioctl.h>
#include <sys/time.h>

#include "dvr_common_api.h"
#include "dvr_enc_api.h"
#include "gmavi_api.h"

int dvr_fd = 0;
int enc_fd = 0;

//test record
unsigned char *bs_buf;
HANDLE rec_file;
int enc_buf_size;
struct pollfd rec_fds;
char file_name[128];

/**
 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
    int ret = 0, ch_num = 0;
    dvr_enc_channel_param    ch_param;
    EncParam_Ext3 enc_param_ext = {0};
    dvr_enc_control   enc_ctrl;
    dvr_enc_control   enc_update;
    dvr_enc_queue_get   data;
    unsigned char *buf;
    int buf_size;
    FuncTag tag;
    struct timeval t1,t2;
    char tmp_str[128];
    int flag = 0;
```

```

    dvr_fd = open("/dev/dvr_common", O_RDWR);      //  

open_dvr_common

    enc_fd = open("/dev/dvr_enc", O_RDWR);          //  

open_dvr_encode

    //set dvr encode source parameter
    ch_param.src.input_system = MCP_VIDEO_NTSC;
    ch_param.src.channel = ch_num;
    ch_param.src.enc_src_type = ENC_TYPE_FROM_CAPT
URE;

    ch_param.src.dim.width = 1280;
    ch_param.src.dim.height = 720;

    ch_param.src.di_mode = LVFRAME_EVEN_ODD;
    ch_param.src.mode = LVFRAME_FRAME_MODE;
    ch_param.src.dma_order = DMAORDER_PACKET;
    ch_param.src.scale_indep = CAPSCALER_NOT_KEEP_
RATIO;
    ch_param.src.input_system = MCP_VIDEO_NTSC;
    ch_param.src.color_mode = CAPCOLOR_YUV422;

    ch_param.src.vp_param.is_3DI = FALSE;
    ch_param.src.vp_param.is_denoise = FALSE;
    ch_param.src.vp_param.denoise_mode = GM3DI_FIE
LD;
    ///////////////////////////////
///////////////
    //set dvr encode main bitstream parameter
    ch_param.main_bs.enabled = DVR_ENC_EBST_ENABLE
;
    ch_param.main_bs.out_bs = 0;
    ch_param.main_bs.enc_type = ENC_TYPE_H264;
    ch_param.main_bs.is_blocked = FALSE;
    ch_param.main_bs.en_snapshot = DVR_ENC_EBST_DI

```

```

SABLE;

    ch_param.main_bs.dim.width = 1280;
    ch_param.main_bs.dim.height = 720;

    //set main bitstream encode parameter
    ch_param.main_bs.enc.input_type = ENC_INPUT_H2
642D;
    ch_param.main_bs.enc.frame_rate = 30;
    ch_param.main_bs.enc.bit_rate = 4096000;
    ch_param.main_bs.enc.ip_interval = 60;
    ch_param.main_bs.enc.init_quant = 25;
    ch_param.main_bs.enc.max_quant = 51;
    ch_param.main_bs.enc.min_quant = 1;
    ch_param.main_bs.enc.is_use_ROI = FALSE;
    ch_param.main_bs.enc.ROI_win.x = 0;
    ch_param.main_bs.enc.ROI_win.y = 0;
    ch_param.main_bs.enc.ROI_win.width = 320;
    ch_param.main_bs.enc.ROI_win.height = 240;

    //set main bitstream scalar parameter
    ch_param.main_bs.scl.src_fmt = SCALE_YUV422;
    ch_param.main_bs.scl.dst_fmt = SCALE_YUV422;
    ch_param.main_bs.scl.scale_mode = SCALE_LINEAR
;

    ch_param.main_bs.scl.is_dither = FALSE;
    ch_param.main_bs.scl.is_correction = FALSE;
    ch_param.main_bs.scl.is_album = TRUE;
    ch_param.main_bs.scl.des_level = 0;

    //set main bitstream snapshot parameter
    ch_param.main_bs.snap.sample = JCS_yuv420;
    ch_param.main_bs.snap.RestartInterval = 0;
    ch_param.main_bs.snap.u82D = JENC_INPUT_MP42D;

    ch_param.main_bs.snap.quality = 70;

```

```

    //associate the ext. structure

    ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_
ADD_VAL(sizeof(enc_param_ext));
    ch_param.main_bs.enc.pext_data = &enc_param_ext;

    enc_param_ext.feature_enable = 0;           //CBR

    /////////////////////////////////
/////////////////////////////
    //set update recode parameter
    memset(&enc_update, 0x0, sizeof(dvr_enc_control));
    enc_update.stream = 0;
    enc_update.update_parm.stream_enable = 1;
    enc_update.update_parm.frame_rate = GMVAL_DO_N
OT_CARE;
    enc_update.update_parm.bit_rate = 131072;
    enc_update.update_parm.ip_interval = GMVAL_DO_
NOT_CARE;
    enc_update.update_parm.dim.width = GMVAL_DO_NO
T_CARE;
    enc_update.update_parm.dim.height = GMVAL_DO_N
OT_CARE;
    enc_update.update_parm.src.di_mode = GMVAL_DO_
NOT_CARE;
    enc_update.update_parm.src.mode = GMVAL_DO_NOT
_CARE;
    enc_update.update_parm.src.scale_indep = GMVAL_
DO_NOT_CARE;
    enc_update.update_parm.src.is_3DI = GMVAL_DO_N
OT_CARE;
    enc_update.update_parm.src.is_denoise = GMVAL_
DO_NOT_CARE;
    enc_update.update_parm.init_quant = GMVAL_DO_N
OT_CARE;

```

```

    enc_update.update_parm.max_quant = GMVAL_DO_NO
T_CARE;
    enc_update.update_parm.min_quant = GMVAL_DO_NO
T_CARE;

    ///////////////////////////////////////////////////
//////
        ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_p
aram);

        ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE
, &enc_buf_size);

        bs_buf = (unsigned char*) mmap(NULL, enc_buf_s
ize, PROT_READ|PROT_WRITE,
                                MAP_SHAR
ED, enc_fd, 0);
    ///////////////////////////////////////////////////
/////
        //main bitstream record start
        memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control)
);
        enc_ctrl.command = ENC_START;
        enc_ctrl.stream = 0;
        ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl
);

        // set function tag parameter to dvr graph lev
el
        FN_RESET_TAG(&tag);
        FN_SET_REC_CH(&tag, ch_num);
        ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);

    ///////////////////////////////////////////////////
/////
        //main bitstream
        sprintf(file_name, "CH%d_video_%d", 0, 0);

```

```
sprintf(tmp_str, "%s.h264", file_name);
rec_file = fopen ( tmp_str , "wb+" );

gettimeofday(&t1, NULL);

while(1) {
    // prepare to select(or poll)
    rec_fds.fd = enc_fd;
    rec_fds.revents = 0;
    rec_fds.events = POLLIN_MAIN_BS;

    poll(&rec_fds, 1, 500);

    if (!(rec_fds.revents & POLLIN_MAIN_BS))
        continue;

    // get dataout buffer
    ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET,
&data);
    if(ret < 0)
        continue;

    if(data.new_bs) {
        printf("Change file\n");
        fclose(rec_file);
        sprintf(file_name, "CH%d_video_%d_
update", ch_num, ch_num);
        sprintf(tmp_str, "%s.h264", file_n
ame);
        rec_file = fopen ( tmp_str , "wb+" );
    }
    buf = bs_buf + data.bs.offset;
```

```
        buf_size = data.bs.length;

        ret = fwrite (buf , 1 , buf_size , rec
_file);
        fflush(rec_file);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data
);

        gettimeofday(&t2, NULL);

        if ((t2.tv_sec - t1.tv_sec) == 5) {
//<record for 5 seconds. then update record setting

            if(flag == 0){
                flag = 1;
                enc_update.command = ENC_UPDAT
E;
                ioctl(enc_fd, DVR_ENC_CONTROL,
&enc_update);

                FN_RESET_TAG(&tag);
                FN_SET_REC_CH(&tag, ch_num);

                ioctl(dvr_fd, DVR_COMMON_APPLY
, &tag);
            }
        }

        if ((t2.tv_sec - t1.tv_sec) == 20) {
//<record for 20 seconds. then finish record.
            fclose(rec_file);
            break;
        }
    }

//record stop
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control))
```

```
);

    enc_ctrl.stream = 0;      // for all main and su
b
    enc_ctrl.command = ENC_STOP;
    ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

    FN_RESET_TAG(&tag);
    FN_SET_REC_CH(&tag, ch_num);
    ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
    munmap((void*)bs_buf, enc_buf_size);

    printf("-----\n");
;

    printf(" Record finish\n");
    printf("-----\n");
;

    close(enc_fd);          //close_dvr_encode
    close(dvr_fd);          //close_dvr_common

    return 0;
}
```



Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

[Main Page](#)

[Data Structures](#)

[Files](#)

[Examples](#)

update-record-setting.c

```
/**  
 * this sample code implement main bitstream + sub  
1 bitstream + sub2 bitstream update record function  
* after record 10 seconds, update main bitstream  
record.  
* after record 15 seconds, update sub1 bitstream  
record.  
* after record 25 seconds, finish record.  
*  
* (1) for H264 record, file format please indicate xxx.h264  
*      for MPEG record, file format please indicate xxx.m4v  
*      for MOTION JPEG record, file format please indicate xxx.jpg  
* (2) for H264 record, max_quant < 51, 1 < min_quant  
*      for MPEG record, max_quant < 31, 1 < min_quant  
* (3) for example, if you only updat "frame_rate"  
*,  
*      paramter "bit_rate" don't need to update,  
*      please assign enc_parm.update_parm.bit_rate = GMVAL_DO_NOT_CARE;  
*      and so on.  
*  
*/  
  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <poll.h>
#include <sys/ioctl.h>
#include <sys/time.h>

#include "dvr_common_api.h"
#include "dvr_enc_api.h"
#include "gmavi_api.h"

int dvr_fd = 0;
int enc_fd = 0;

//test record
unsigned char *bs_buf;
HANDLE rec_file;
int enc_buf_size;
int sub1_bs_buf_offset;
int sub2_bs_buf_offset;
int bs_buf_snap_offset;
struct pollfd rec_fds;
char file_name[128];

//sub record
char sub_rec_filename[DVR_ENC_REPD_BT_NUM - 1][64];
int multi_bitstream = 0;
HANDLE rec_sub_bs_file[DVR_ENC_REPD_BT_NUM - 1];

/**
 * @brief main function
 * @return 0 on success, !0 on error
 */
int main(int argc, char *argv[])
{
```

```

    int ret = 0, ch_num = 0, sub1_num = 1, sub2_nu
m = 2;
    dvr_enc_channel_param    ch_param;
    ReproduceBitStream sub1_param;
    ReproduceBitStream sub2_param;
    EncParam_Ext3 enc_param_ext = {0};
    EncParam_Ext3 enc_param_ext1 = {0};
    EncParam_Ext3 enc_param_ext2 = {0};
    dvr_enc_control enc_ctrl;
    dvr_enc_control enc_update;
    dvr_enc_control sub_update;
    dvr_enc_queue_get data;
    unsigned char *buf;
    int buf_size;
    FuncTag tag;
    struct timeval t1,t2;
    char tmp_str[128];
    int flag = 0, sub_flag = 0;

    dvr_fd = open("/dev/dvr_common", O_RDWR);      // 
open_dvr_common

    enc_fd = open("/dev/dvr_enc", O_RDWR);          // 
open_dvr_encode

//set dvr encode source parameter
ch_param.src.input_system = MCP_VIDEO_NTSC;
ch_param.src.channel = ch_num;
ch_param.src.enc_src_type = ENC_TYPE_FROM_CAPT
URE;
    ch_param.src.dim.width = 1280;
    ch_param.src.dim.height = 720;

    ch_param.src.di_mode = LVFRAME_EVEN_ODD;
    ch_param.src.mode = LVFRAME_FRAME_MODE;
    ch_param.src.dma_order = DMAORDER_PACKET;

```

```
    ch_param.src.scale_indep = CAPSCALER_NOT_KEEP_
RATIO;
    ch_param.src.input_system = MCP_VIDEO_NTSC;
    ch_param.src.color_mode = CAPCOLOR_YUV422;

    ch_param.src.vp_param.is_3DI = FALSE;
    ch_param.src.vp_param.is_denoise = FALSE;
    ch_param.src.vp_param.denoise_mode = GM3DI_FIE
LD;
    //////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
    //set dvr encode main bitstream parameter
ch_param.main_bs.enabled = DVR_ENC_EBST_ENABLE
;
    ch_param.main_bs.out_bs = 0;
    ch_param.main_bs.enc_type = ENC_TYPE_H264;
    ch_param.main_bs.is_blocked = FALSE;
    ch_param.main_bs.en_snapshot = DVR_ENC_EBST_DI
SABLE;

    ch_param.main_bs.dim.width = 1280;
    ch_param.main_bs.dim.height = 720;

    //set main bitstream encode parameter
ch_param.main_bs.enc.input_type = ENC_INPUT_H2
642D;
    ch_param.main_bs.enc.frame_rate = 15;
    ch_param.main_bs.enc.bit_rate = 1048576;
    ch_param.main_bs.enc.ip_interval = 60;
    ch_param.main_bs.enc.init_quant = 25;
    ch_param.main_bs.enc.max_quant = 51;
    ch_param.main_bs.enc.min_quant = 1;
    ch_param.main_bs.enc.is_use_ROI = FALSE;
    ch_param.main_bs.enc.ROI_win.x = 0;
    ch_param.main_bs.enc.ROI_win.y = 0;
    ch_param.main_bs.enc.ROI_win.width = 320;
    ch_param.main_bs.enc.ROI_win.height = 240;
```

```

//set main bitstream scalar parameter
ch_param.main_bs.scl.src_fmt = SCALE_YUV422;
ch_param.main_bs.scl.dst_fmt = SCALE_YUV422;
ch_param.main_bs.scl.scale_mode = SCALE_LINEAR
;
ch_param.main_bs.scl.is_dither = FALSE;
ch_param.main_bs.scl.is_correction = FALSE;
ch_param.main_bs.scl.is_album = TRUE;
ch_param.main_bs.scl.des_level = 0;

//set main bitstream snapshot parameter
ch_param.main_bs.snap.sample = JCS_yuv420;
ch_param.main_bs.snap.RestartInterval = 0;
ch_param.main_bs.snap.u82D = JENC_INPUT_MP42D;

ch_param.main_bs.snap.quality = 70;

//associate the ext. structure

ch_param.main_bs.enc.ext_size = DVR_ENC_MAGIC_
ADD_VAL(sizeof(enc_param_ext));
ch_param.main_bs.enc.pext_data = &enc_param_ext;

enc_param_ext.feature_enable = 0;           //CBR

///////////////////////////////
/////////////////////////////
//sub1 bitstream
memset(&sub1_param, 0x0, sizeof(ReproduceBitStream));

sub1_param.enabled = DVR_ENC_EBST_ENABLE;
sub1_param.out_bs = 1;
sub1_param.enc_type = ENC_TYPE_H264;
sub1_param.is_blocked = FALSE;

```

```

sub1_param.en_snapshot = DVR_ENC_EBST_DISABLE;

sub1_param.dim.width = 1280;
sub1_param.dim.height = 720;

sub1_param.enc.input_type = ENC_INPUT_H2642D;
sub1_param.enc.frame_rate = 15;
sub1_param.enc.bit_rate = 200000;
sub1_param.enc.ip_interval = 30;
sub1_param.enc.init_quant = 30;
sub1_param.enc.max_quant = 51;
sub1_param.enc.min_quant = 1;
sub1_param.enc.is_use_ROI = FALSE;
sub1_param.enc.ROI_win.x = 0;
sub1_param.enc.ROI_win.y = 0;
sub1_param.enc.ROI_win.width = 320;
sub1_param.enc.ROI_win.height = 240;

//set sub1 bitstream scalar parameter
sub1_param.scl.src_fmt = SCALE_YUV422;
sub1_param.scl.dst_fmt = SCALE_YUV422;
sub1_param.scl.scale_mode = SCALE_LINEAR;
sub1_param.scl.is_dither = FALSE;
sub1_param.scl.is_correction = FALSE;
sub1_param.scl.is_album = TRUE;
sub1_param.scl.des_level = 0;

//set sub1 bitstream snapshot parameter
sub1_param.snap.sample = JCS_yuv420;
sub1_param.snap.RestartInterval = 0;
sub1_param.snap.u82D = JENC_INPUT_MP42D;
sub1_param.snap.quality = 70;
//associate the ext. structure

sub1_param.enc.ext_size = DVR_ENC_MAGIC_ADD_VAL
(sizeof(enc_param_ext1));
sub1_param.enc.pext_data = &enc_param_ext1;

```

```
    enc_param_ext1.feature_enable &= ~DVR_ENC_MJPE
G_FUNCTION;

    //////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//sub2 bitstream
memset(&sub2_param, 0x0, sizeof(ReproduceBitSt
ream));

sub2_param.enabled = DVR_ENC_EBST_ENABLE;
sub2_param.out_bs = 2;
sub2_param.enc_type = ENC_TYPE_H264;
sub2_param.is_blocked = FALSE;
sub2_param.en_snapshot = DVR_ENC_EBST_DISABLE;

sub2_param.dim.width = 1280;
sub2_param.dim.height = 720;

sub2_param.enc.input_type = ENC_INPUT_H2642D;
sub2_param.enc.frame_rate = 15;
sub2_param.enc.bit_rate = 131072;
sub2_param.enc.ip_interval = 30;
sub2_param.enc.init_quant = 25;
sub2_param.enc.max_quant = 51;
sub2_param.enc.min_quant = 1;
sub2_param.enc.is_use_ROI = FALSE;
sub2_param.enc.ROI_win.x = 0;
sub2_param.enc.ROI_win.y = 0;
sub2_param.enc.ROI_win.width = 160;
sub2_param.enc.ROI_win.height = 128;

//set sub2 bitstream scalar parameter
sub2_param.scl.src_fmt = SCALE_YUV422;
sub2_param.scl.dst_fmt = SCALE_YUV422;
sub2_param.scl.scale_mode = SCALE_LINEAR;
```

```

    sub2_param.scl.is_dither = FALSE;
    sub2_param.scl.is_correction = FALSE;
    sub2_param.scl.is_album = TRUE;
    sub2_param.scl.des_level = 0;

    //set sub2 bitstream snapshot parameter
    sub2_param.snap.sample = JCS_yuv420;
    sub2_param.snap.RestartInterval = 0;
    sub2_param.snap.u82D = JENC_INPUT_MP42D;
    sub2_param.snap.quality = 70;
    //associate the ext. structure

    sub2_param.enc.ext_size = DVR_ENC_MAGIC_ADD_VAL
(sizeof(enc_param_ext2));
    sub2_param.enc.pext_data = &enc_param_ext2;

    enc_param_ext2.feature_enable &= ~DVR_ENC_MJPE
G_FUNCTION;

    /////////////////////////////////
    ///////////////////////////////
    //set update recode parameter
    memset(&enc_update, 0x0, sizeof(dvr_enc_control
));
    enc_update.stream = 0;
    enc_update.update_parm.stream_enable = 1;
    enc_update.update_parm.frame_rate = 30;
    enc_update.update_parm.bit_rate = 262144;
    enc_update.update_parm.ip_interval = 60;
    enc_update.update_parm.dim.width = 1280;
    enc_update.update_parm.dim.height = 720;
    enc_update.update_parm.src.di_mode = LVFRAME_E
VEN_ODD;
    enc_update.update_parm.src.mode = LVFRAME_FRAM
E_MODE;
    enc_update.update_parm.src.scale_indep = CAPSC
ALER_NOT_KEEP_RATIO;

```

```
    enc_update.update_parm.src.is_3DI = FALSE;
    enc_update.update_parm.src.is_denoise = FALSE;
    enc_update.update_parm.init_quant = 25;
    enc_update.update_parm.max_quant = 51;
    enc_update.update_parm.min_quant = 1;

    ///////////////////////////////////////////////////
///////////////////////////////////////////////////
//set sub update recode parameter
    memset(&sub_update, 0x0, sizeof(dvr_enc_control));
    sub_update.stream = 1;
    sub_update.update_parm.stream_enable = 1;
    sub_update.update_parm.frame_rate = 30;
    sub_update.update_parm.bit_rate = 131072;
    sub_update.update_parm.ip_interval = 60;
    sub_update.update_parm.dim.width = 1280;
    sub_update.update_parm.dim.height = 720;
    sub_update.update_parm.src.di_mode = LVFRAME_E
VEN_ODD;
    sub_update.update_parm.src.mode = LVFRAME_FRAM
E_MODE;
    sub_update.update_parm.src.scale_indep = CAPSC
ALER_NOT_KEEP_RATIO;
    sub_update.update_parm.src.is_3DI = FALSE;
    sub_update.update_parm.src.is_denoise = FALSE;
    sub_update.update_parm.init_quant = 25;
    sub_update.update_parm.max_quant = 51;
    sub_update.update_parm.min_quant = 1;
    ///////////////////////////////////////////////////
///////////////////////////////////////////////////
    ioctl(enc_fd, DVR_ENC_SET_CHANNEL_PARAM, &ch_p
aram);

    ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE
, &enc_buf_size);
```

```

        ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1
_BS_OFFSET, &sub1_bs_buf_offset);

        ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB2
_BS_OFFSET, &sub2_bs_buf_offset);

        ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP
_OFFSET, &bs_buf_snap_offset);

        bs_buf = (unsigned char*) mmap(NULL, enc_buf_size,
PROT_READ|PROT_WRITE,
MAP_SHARED, enc_fd, 0);

        ioctl(enc_fd, DVR_ENC_SET_SUB_BS_PARAM, &sub1_param);

        ioctl(enc_fd, DVR_ENC_SET_SUB_BS_PARAM, &sub2_param);
        ///////////////////////////////////////////////////
///////////////////////////////////////////////////
//main bitstream record start
        memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
        enc_ctrl.command = ENC_START;
        enc_ctrl.stream = 0;
        ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);
        // set function tag parameter to dvr graph level
        FN_RESET_TAG(&tag);
        FN_SET_REC_CH(&tag, ch_num);
        ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
        ///////////////////////////////////////////////////
///////////////////////////////////////////////////
        multi_bitstream = DVR_ENC_EBST_ENABLE;

```

```
//sub1 bitstream record start
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control))
);
enc_ctrl.command = ENC_START;
enc_ctrl.stream = 1;
ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl
);

// set function tag parameter to dvr graph lev
el
FN_RESET_TAG(&tag);
FN_SET_SUB1_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
///////////
///////////
//sub2 bitstream record start
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control)
);
enc_ctrl.command = ENC_START;
enc_ctrl.stream = 2;
ret = ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl
);

// set function tag parameter to dvr graph lev
el
FN_RESET_TAG(&tag);
FN_SET_SUB2_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
///////////
///////////
//main bitstream
sprintf(file_name, "CH%d_video_%d", 0, 0);

sprintf(tmp_str, "%s.h264", file_name);
rec_file = fopen ( tmp_str , "wb+" );
///////////
///////////
```

```

    //sub1 bitstream
    sprintf(sub_rec_filename[sub1_num - 1], "CH%d_
Sub%d_Video_%03d", ch_num, sub1_num, 001);
    sprintf(tmp_str, "%s.h264", sub_rec_filename[s
ub1_num - 1]);
    rec_sub_bs_file[sub1_num - 1] = fopen ( tmp_st
r , "wb+" );
    /////////////////////////////////
    /////////////////////////////////
    //sub2 bitstream
    sprintf(sub_rec_filename[sub2_num - 1], "CH%d_
Sub%d_Video_%03d", ch_num, sub2_num, 001);
    sprintf(tmp_str, "%s.h264", sub_rec_filename[s
ub2_num - 1]);
    rec_sub_bs_file[sub2_num - 1] = fopen ( tmp_st
r , "wb+" );
    /////////////////////////////////
    /////////////////////////////////
    /////////////////////////////////
    gettimeofday(&t1, NULL);

    while(1) {
        // prepare to select(or poll)
        rec_fds.fd = enc_fd;
        rec_fds.revents = 0;
        rec_fds.events = POLLIN_MAIN_BS;

        if (multi_bitstream == DVR_ENC_EBST_EN
ABLE)
            rec_fds.events |= (POLLIN_SUB1_BS
| POLLIN_SUB2_BS);

        poll(&rec_fds, 1, 500);

        if (rec_fds.revents & POLLIN_SUB1_BS)
{
            ret = ioctl(enc_fd, DVR_ENC_QUEUE_
GET_SUB1_BS, &data);

```

```

        if(ret < 0)
            continue;

        if(data.new_bs) {
            printf("Change file2\n");
            fclose(rec_sub_bs_file[sub1_nu
m - 1]);
            sprintf(sub_rec_filename[sub1_
num - 1], "CH%d_Sub%d_Video_%03d_update", ch_num,
sub1_num, 001);
            sprintf(tmp_str, "%s.h264", su
b_rec_filename[sub1_num - 1]);
            rec_sub_bs_file[sub1_num - 1]
= fopen ( tmp_str , "wb+" );
        }

        buf = bs_buf + sub1_bs_buf_offset
+ data.bs.offset;
        buf_size = data.bs.length;

        fwrite (buf , 1 , buf_size , rec_s
ub_bs_file[sub1_num - 1]);
        fflush(rec_sub_bs_file[sub1_num -
1]);
        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
    }

    if (rec_fds.revents & POLLIN_SUB2_BS)
{
    ret = ioctl(enc_fd, DVR_ENC_QUEUE_
GET_SUB2_BS, &data);
    if(ret < 0)
        continue;

    buf = bs_buf + sub2_bs_buf_offset
+ data.bs.offset;
}

```

```
        buf_size = data.bs.length;

                fwrite (buf , 1 , buf_size , rec_s
ub_bs_file[sub2_num - 1]);
                fflush(rec_sub_bs_file[sub2_num -
1]);
                ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &
data);
}

if (!(rec_fds.revents & POLLIN_MAIN_BS
))
    continue;

// get dataout buffer
ret = ioctl(enc_fd, DVR_ENC_QUEUE_GET,
&data);
if(ret < 0)
    continue;

if(data.new_bs) {
    printf("Change file\n");
    fclose(rec_file);
    sprintf(file_name, "CH%d_video_%d_
update", ch_num, ch_num);
    sprintf(tmp_str, "%s.h264", file_n
ame);
    rec_file = fopen ( tmp_str , "wb"
);
}
buf = bs_buf + data.bs.offset;
buf_size = data.bs.length;

ret = fwrite (buf , 1 , buf_size , rec
_file);
fflush(rec_file);
```

```

        ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data
);

        gettimeofday(&t2, NULL);

        if ((t2.tv_sec - t1.tv_sec) == 10) {
            //<record for 10 seconds. then update record setting
            if(flag == 0){
                flag = 1;
                enc_update.command = ENC_UPDAT
E;
                ioctl(enc_fd, DVR_ENC_CONTROL,
&enc_update);

                FN_RESET_TAG(&tag);
                FN_SET_REC_CH(&tag, ch_num);

                //FN_SET_SUB1_REC_CH(&tag, ch_
num);
                //FN_SET_SUB2_REC_CH(&tag, ch_
num);

                ioctl(dvr_fd, DVR_COMMON_APPLY
, &tag);
            }
        }

        if ((t2.tv_sec - t1.tv_sec) == 15) {
            //<record for 15 seconds. then update sub bitstr
eam record setting
            if(sub_flag == 0){
                sub_flag = 1;

                sub_update.command = ENC_UPDAT
E;
                ioctl(enc_fd, DVR_ENC_CONTROL,

```

```
&sub_update);

FN_RESET_TAG(&tag);

FN_SET_SUB1_REC_CH(&tag, ch_num);

ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
}

}

if ((t2.tv_sec - t1.tv_sec) == 25) {
//<record for 25 seconds. then finish record.
fclose(rec_file);
fclose(rec_sub_bs_file[sub1_num - 1]);
fclose(rec_sub_bs_file[sub2_num - 1]);
break;
}
}

//record stop
memset(&enc_ctrl, 0x0, sizeof(dvr_enc_control));
enc_ctrl.stream = 0; // for all main and sub
enc_ctrl.command = ENC_STOP;
ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ctrl);

FN_RESET_TAG(&tag);
FN_SET_REC_CH(&tag, ch_num);
FN_SET_SUB1_REC_CH(&tag, ch_num);
FN_SET_SUB2_REC_CH(&tag, ch_num);
ioctl(dvr_fd, DVR_COMMON_APPLY, &tag);
munmap((void*)bs_buf, enc_buf_size);
```

```
    printf("-----\n");
};

printf(" Record finish\n");
printf("-----\n");

close(enc_fd);           //close_dvr_encode
close(dvr_fd);           //close_dvr_common

return 0;
}
```



Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
b c d e f g j l m p q r s t v				

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- b -

- BG_AND_1PLANE : [dvr_disp_api.h](#)
- BG_AND_2PLANE : [dvr_disp_api.h](#)
- BG_ONLY : [dvr_disp_api.h](#)

Main Page

Data Structures

Files

Examples

File List

Globals

source/dvr_dec_api.h

Go to the documentation of this file.

```
00001 #ifndef __DVR_DEC_API_H__
00002 #define __DVR_DEC_API_H__
00003
00004 #include <linux/ioctl.h>
00005 #include "dvr_type_define.h"
00006 #include "dvr_dec_ioctl.h"
00007
00008 typedef enum dvr_dec_dest_type_tag {
00009     DEC_TYPE_TO_DISPLAY=0,
00010     DEC_TYPE_TO_BUFFER,
00011     DEC_TYPE_COUNT
00012 }dvr_dec_dest_type;
00013
00014 typedef struct DecParam_tag {
00015     /*! #DecoderOutputColorTag */
00016     int      output_type;
00017     int      reserved[2];
00018 } DecParam;
00019
00020 typedef struct dvr_dec_channel_param_tag {
00021     /*! #EncodeType_tag */
00022     int      dec_type;
00023     /*! #dvr_dec_dest_type_tag */
00024     int      dec_dest_type;
00025     int      channel;
00026     int      is_blocked;
00027     int      is_use_scaler;
00028     /*! #DecParam_tag */
00029     DecParam    dec_param;
00030     /*! #ScalerParamtag */
00031     ScalerParam scl_param;
00032     int      reserved[8];
```

```
00033 }dvr_dec_channel_param;
00034
00035
00036 typedef struct dvr_dec_queue_get_tag {
00037     /*! #dvr_bs_data_tag */
00038     dvr_bs_data bs;
00039     int         channel; //for debug only
00040     int         reserved[1];
00041 }dvr_dec_queue_get;
00042
00043
00044 typedef enum dvr_dec_ctrl_cmd_tag {
00045     DEC_START,
00046     DEC_STOP,
00047     DEC_UPDATE
00048 }dvr_dec_ctrl_cmd;
00049
00050 typedef struct dvr_dec_control_tag {
00051     /*! #dvr_dec_ctrl_cmd_tag */
00052     dvr_dec_ctrl_cmd    command;
00053     struct
00054     {
00055         /*! #DIM_tag */
00056         DIM      dim;
00057         /*! #RECT_tag */
00058         RECT    win;
00059         int      bs_rate; //bitstream frame rate
00060         int      reserved[2];
00061     }src_param;
00062     struct
00063     {
00064         int      plane_id;
00065         /*! #RECT_tag */
00066         RECT    win;
00067         int      is_display;
00068         int      display_rate;
```

```
00069         int      reserved[2];
00070     }dst_param;
00071 }dvr_dec_control;
00072
00073
00074 typedef struct dvr_dec_clear_param_tag
00075 {
00076     /*! #RECT_tag */
00077     RECT        win;
00078     unsigned int pattern;
00079     int         reserved[2];
00080 }dvr_dec_clear_param;
00081
00082 #endif /* __DVR_DEC_API_H__ */
00083
```

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

source/dvr_type_define.h

Go to the documentation of this file.

```
00001 #ifndef __DVR_TYPE_DEFINE_H__
00002 #define __DVR_TYPE_DEFINE_H__
00003
00004 #include "dvr_scenario_define.h"
00005
00006 #ifndef TRUE
00007 #define TRUE    1
00008 #endif
00009
00010 #ifndef FALSE
00011 #define FALSE   0
00012 #endif
00013
00014 #define GMDVR_MEM_CFG_FILE  "/mnt/mtd/gmdvr_
mem.cfg"
00015 #define GMDVR_MAKE_FOURCC(a,b,c,d)      (
int)((a)|(b)<<8|(c)<<16|(d)<<24)
00016
00017 #define GMVAL_DO_NOT CARE      (-54172099)
00018 #define GMVAL_RATE(val,base)      ((base<<16) |
val)
00019
00020 #define GMVAL_RT_GET_VAL(rate)  (rate&0x0000
FFFF)
00021 #define GMVAL_RT_GET_BASE(rate) (rate>>16)
00022
00023 //Graph ID
00024 #define GFID_DISP(plane_num)(0x10000+plane_n
um)
00025 #define GFID_ENC(ch_num)        (0x20000+ch_num)
00026 #define GFID_DEC(ch_num)        (0x30000+ch_num)
00027
```

```

00028 /**
00029  * @brief function tag for videograph level
00030 */
00031 typedef struct FuncTag_tag{
00032     //bit24~27 is reserved
00033     int func : 28;           ///< functional
tag
00034     int cas_ch : 4;          ///< cascade cha
nnel
00035     int lv_ch;              ///< liveview cha
nnel
00036     int rec_ch;             ///< record cha
nnel
00037     int pb_ch;              ///< playback cha
nnel
00038 }FuncTag;
00039 /*
00040
func cas_ch
    rec_ch
    pb
    _ch
00041
    |
    |
    |
00042     xxxx xxxx xxxx   xx      xx      xxxx xxxx xx
xx xxxx      xxxx xxxx xxxx xxxx      xxxx xxxx xxxx x
xxx
00043 */
00044 #define MTHD_USE_CMP          0x1
00045
00046 #define FN_NONE               0x0
00047 #define FN_LIVEVIEW            0x1
00048 #define FN_RECORD              0x2
00049 #define FN_PLAYBACK            0x4
00050 #define FN CASCADE             0x8
00051 #define FN_LCD_PARAM          0x10
00052 #define FN_PLANE_PARAM         0x20
00053 #define FN_SUB1_RECORD         0x40

```

```

00054 #define FN_SUB2_RECORD          0x80
00055 #define FN_SUB3_RECORD          0x100
00056 #define FN_SUB4_RECORD          0x200
00057 #define FN_SUB5_RECORD          0x400
00058 #define FN_SUB6_RECORD          0x800
00059 #define FN_SUB7_RECORD          0x1000
00060 #define FN_SUB8_RECORD          0x2000
00061 #define FN_UPDATE_METHOD        0x10000
00062 #define FN_PB_SCL_LINK          0x20000
00063 #define FN_METHOD_USE_CMP       0x8000000
00064 #define FN_SUB_ALL_RECORD        (FN_RECORD|FN_SU
B1_RECORD|FN_SUB2_RECORD|FN_SUB3_RECORD|FN_SUB4_RE
CORD|FN_SUB5_RECORD|FN_SUB6_RECORD|FN_SUB7_RECORD|
FN_SUB8_RECORD)
00065
00066
00067 #define FN_RESET_TAG(ptag)           { (pt
ag)->func=0; (ptag)->lv_ch=0; (ptag)->rec_ch=0; (p
tag)->pb_ch=0; (ptag)->cas_ch=0; }
00068 #define FN_COPY_TAG(newt,ptag)        { (ne
wt)->func=(ptag)->func; (newt)->lv_ch=(ptag)->lv_c
h; (newt)->rec_ch=(ptag)->rec_ch; (newt)->pb_ch=(p
tag)->pb_ch; (newt)->cas_ch=(ptag)->cas_ch; }
00069 #define FN_SET_LV_CH(ptag, ch_num)   { (pt
ag)->func|=FN_LIVEVIEW; (ptag)->lv_ch|=(0x1<<ch_
num); }
00070 #define FN_SET_REC_CH(ptag, ch_num)  { (pt
ag)->func|=(FN_RECORD); (ptag)->rec_ch|=(0x1<<ch_n
um); }
00071 #define FN_SET_SUB1_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_SUB1_RECORD); (ptag)->rec_ch|=
(0x1<<ch_num); }
00072 #define FN_SET_SUB2_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_SUB2_RECORD); (ptag)->rec_ch|=
(0x1<<ch_num); }
00073 #define FN_SET_SUB3_REC_CH(ptag, ch_num) { (ptag)->func|=(FN_SUB3_RECORD); (ptag)->rec_ch|=

```

```

(0x1<<ch_num); }

00074 #define FN_SET_SUB4_REC_CH(ptag, ch_num) {
    (ptag)->func|=(FN_SUB4_RECORD); (ptag)->rec_ch|=
(0x1<<ch_num); }

00075 #define FN_SET_SUB5_REC_CH(ptag, ch_num) {
    (ptag)->func|=(FN_SUB5_RECORD); (ptag)->rec_ch|=
(0x1<<ch_num); }

00076 #define FN_SET_SUB6_REC_CH(ptag, ch_num) {
    (ptag)->func|=(FN_SUB6_RECORD); (ptag)->rec_ch|=
(0x1<<ch_num); }

00077 #define FN_SET_SUB7_REC_CH(ptag, ch_num) {
    (ptag)->func|=(FN_SUB7_RECORD); (ptag)->rec_ch|=
(0x1<<ch_num); }

00078 #define FN_SET_SUB8_REC_CH(ptag, ch_num) {
    (ptag)->func|=(FN_SUB8_RECORD); (ptag)->rec_ch|=
(0x1<<ch_num); }

00079 #define FN_SET_PB_CH(ptag, ch_num) { (pt
ag)->func|=FN_PLAYBACK; (ptag)->pb_ch|=(0x1<<ch_
num); }

00080 #define FN_SET_CAS_CH(ptag, ch_num) { (pt
ag)->func|=FN CASCADE; (ptag)->cas_ch|=(0x1<<ch
_num); }

00081 #define FN_SET_FUNC(ptag, fnc) { (pt
ag)->func|=fnc; }

00082 #define FN_REMOVE_FUNC(ptag, fnc) { (pt
ag)->func&=(~fnc); }

00083 #define FN_SET_ALL(ptag) { (pt
ag)->func=(0xFF); /* Note: 'func' value doesn't
include FN_UPDATE_METHOD. */ \
00084
( ptag)->lv_ch=0xFFFFFFFF; (ptag)->rec_ch=0xFFFFFFFF
; (ptag)->pb_ch=0xFFFFFFFF; (ptag)->cas_ch=0xF; }

00085 #define FN_IS_UPDATE(ptag) ((ptag)
->func&FN_UPDATE_METHOD)

00086 #define FN_COMPARE(ptagA, ptagB) ( ((pta
gA)->func==(ptagB)->func) && ((ptagA)->lv_ch==(pta
gB)->lv_ch) && ((ptagA)->rec_ch==(ptagB)->rec_ch)

```

```

&& ((ptagA)->pb_ch==(ptagB)->pb_ch) && ((ptagA)->c
as_ch==(ptagB)->cas_ch) )
00087 #define FN_IS_EMPTY(ptag) ( ((pta
g)->func==0) && ((ptag)->lv_ch==0) && ((ptag)->rec
_ch==0) && ((ptag)->pb_ch==0) && ((ptag)->cas_ch==
0) )
00088 #define FN_IS_FN(ptag, fn) ( (ptag
)->func&(fn) )
00089 #define FN_REMOVE_LV_CH(ptag, ch) ( (ptag)-
>lv_ch&=(~ch) )
00090 /* FN_CHECK_MASK uses AND(&) operation on pt
agA with ptagB, and check channel values for LV,RE
C,PB,CAS */
00091 #if 1
00092 #define FN_CHECK_MASK(ptagA, ptagB) \
00093 (((((ptagA)->func&(ptagB)->func)==FN_
LIVEVIEW)?((ptagA)->lv_ch&(ptagB)->lv_ch):
 \
00094 (((((ptagA)->func&(ptagB)->func)=
=FN_RECORD)?((ptagA)->rec_ch&(ptagB)->rec_ch):
 \
00095 (((((ptagA)->func&(ptagB)->fu
nc)==FN_SUB1_RECORD)?((ptagA)->rec_ch&(ptagB)->rec
_ch): \
00096 (((((ptagA)->func&(ptagB)
->func)==FN_SUB2_RECORD)?((ptagA)->rec_ch&(ptagB)-
>rec_ch): \
00097 (((((ptagA)->func&(ptagB)
->func)==FN_SUB3_RECORD)?((ptagA)->rec_ch&(ptagB)-
>rec_ch): \
00098 (((((ptagA)->func&(ptagB)
->func)==FN_SUB4_RECORD)?((ptagA)->rec_ch&(ptagB)-
>rec_ch): \
00099 (((((ptagA)->func&(ptagB)
->func)==FN_SUB5_RECORD)?((ptagA)->rec_ch&(ptagB)-
>rec_ch): \
00100 (((((ptagA)->func&(ptagB)

```

```

->func)==FN_SUB6_RECORD)?((ptagA)->rec_ch&(ptagB)-
>rec_ch): \
00101          (((ptagA)->func&(ptagB)
->func)==FN_SUB7_RECORD)?((ptagA)->rec_ch&(ptagB)-
>rec_ch): \
00102          (((ptagA)->func&(ptagB)
->func)==FN_SUB8_RECORD)?((ptagA)->rec_ch&(ptagB)-
>rec_ch): \
00103          (((ptagA)->func&(pt
agB)->func)==FN_PLAYBACK)?((ptagA)->pb_ch&(ptagB)-
>pb_ch): \
00104          (((ptagA)->func
&(ptagB)->func)==FN CASCADE)?((ptagA)->cas_ch&(pta
gB)->cas_ch): \
00105          ((ptagA)->func&(ptag
B)->func) )))))))))))))
00106 #else
00107 #define FN_CHECK_MASK(ptagA, ptagB) \
00108          (((ptagA)->func&(ptagB)->func)==FN_
LIVEVIEW)?((ptagA)->lv_ch&(ptagB)->lv_ch):
 \
00109          (((ptagA)->func&(ptagB)->func)=
=FN_RECORD)?((ptagA)->rec_ch&(ptagB)->rec_ch):
 \
00110          (((ptagA)->func&(ptagB)->fu
nc)==FN_SUB1_RECORD)?((ptagA)->rec_ch&(ptagB)->rec
_ch): \
00111          (((ptagA)->func&(ptagB)
->func)==FN_SUB2_RECORD)?((ptagA)->rec_ch&(ptagB)-
>rec_ch): \
00112          (((ptagA)->func&(pt
agB)->func)==FN_PLAYBACK)?((ptagA)->pb_ch&(ptagB)-
>pb_ch): \
00113          (((ptagA)->func
&(ptagB)->func)==FN CASCADE)?((ptagA)->cas_ch&(pta
gB)->cas_ch): \
00114          ((ptagA)->func&(ptag

```

```

B)->func) )))))
00115 #endif
00116 #define FN_ITEMS(ptag) (ptag)-
>func, (ptag)->lv_ch, (ptag)->rec_ch, (ptag)->pb_c
h, (ptag)->cas_ch
00117
00118 /* Queue name.
00119     Note => the length of queue must be small
er than MAX_NAME_SIZE. */
00120 #define QNAME_LCD           "lcd"
00121 #define QNAME_3DI_SCL        "3di_scl"
00122 #define QNAME_LV_SCL         "lv_scl"
00123 #define QNAME_ENC_IN         "enc_in"
00124 #define QNAME_ENC_OUT        "enc_out"
00125 #define QNAME_SS_ENC_IN      "ssenc_in"
00126 #define QNAME_SS_ENC_OUT     "ssenc_out"
00127 #define QNAME_SUB1_ENC_IN    "sub1enc_in"
00128 #define QNAME_SUB1_ENC_OUT   "sub1enc_out"
00129 #define QNAME_SUB2_ENC_IN    "sub2enc_in"
00130 #define QNAME_SUB2_ENC_OUT   "sub2enc_out"
00131 #define QNAME_DEC_IN         "dec_in"
00132 #define QNAME_PB_SCL         "pb_scl"
00133 #if TWO_STAGES_SCALER
00134 #define QNAME_PB_SCL2       "pb_scl2"
00135 #endif
00136
00137 typedef enum QueueID_tag {
00138     QID_LCD=10,
00139     QID_3DI_SCL,
00140     QID_LV_SCL,
00141     QID_ENC_IN,
00142     QID_ENC_OUT,
00143     QID_SS_ENC_IN,
00144     QID_SS_ENC_OUT,
00145     QID_SUB1_ENC_IN,
00146     QID_SUB1_ENC_OUT,
00147     QID_SUB2_ENC_IN,

```

```
00148     QID_SUB2_ENC_OUT,
00149     QID_DEC_IN,
00150     QID_PB_SCL
00151 }QueueID;
00152
00153 /**
00154 * @brief set width and height
00155 */
00156 typedef struct DIM_tag {
00157     int width;      ///< width in pixel
00158     int height;     ///< height in pixel
00159 }DIM;
00160
00161 /**
00162 * @brief set size and position
00163 */
00164 typedef struct RECT_tag {
00165     int x;          ///< x position
00166     int y;          ///< y positon
00167     int width;      ///< width in pixel
00168     int height;     ///< height in pixel
00169 }RECT;
00170
00171 /**
00172 * @brief set position x and y
00173 */
00174 typedef struct POS_tag {
00175     int x;          ///< x position
00176     int y;          ///< y positon
00177 }POS;
00178
00179 /**
00180 * @brief set ROI position x, y, width, height, and enable/disable
00181 */
00182 typedef struct ROI_ALL_tag {
00183     /*! Enable the ROI function, TRUE/FALSE
```

```
/*
00184     int      is_use_ROI;
00185     /*! #RECT_tag                                */
00186     RECT      win;
00187 }ROI_ALL;
00188
00189
00190 /**
00191 * @brief set queue memory configuration
00192 */
00193 typedef struct QueueMemConfig_tag {
00194     int size;           ///< size of buffer
00195     int count;          ///< the number of b
uffers
00196     int ddr_num;        ///< the DDR number
00197     int limit_count;    ///< Max count of in
-used buffer
00198 }QueMemCfg;
00199
00200 /**
00201 * @brief set dvr graph queue configuration
00202 */
00203 typedef struct dvr_graph_vqueueut_tag
00204 {
00205     struct v_queue_t    *que;
00206     int size;           ///< size of buf
fer
00207     int count;          ///< the number
of buffers
00208     int ddr;             ///< the DDR num
ber
00209     int limit_count;    ///< Max count o
f in-used buffer
00210     FuncTag user_tag;   ///< function ta
g for videograph level
00211 }dvr_graph_vqueueut;
00212
```

```
00213 /**
00214  * @brief dvr bit-stream data
00215 */
00216 typedef struct dvr_bs_data_tag {
00217     struct timeval timestamp;
00218     int offset;           ///< bitstream buffer offset.
00219     int length;          ///< bitstream buffer length.
00220     int is_keyframe;     ///< 1: current frame is a keyframe. 0: current frame is not a key frame.
00221     int mv_offset;       ///< motion vect or offset
00222     int mv_length;       ///< motion vect or length
00223
00224     //internal use
00225     void *p_job;         ///< internal use
00226     int stream;          ///< internal use
00227     /*! 0: current P frame was encoded as referenced. */
00228     /*! 1: current P frame was encoded as no n-referenced */
00229     int NonRef;
00230     int reserved[8];
00231 }dvr_bs_data;
00232
00233 typedef struct dvr_rate_tag {
00234     int numerator;        ///< Not in use
00235     int denominator;      ///< Not in use
00236     int reserved[4];       ///< Not in use
00237 }dvr_ratio;
00238
```

```
00239 /**
00240  * @brief dvr video parameter.
00241 */
00242 typedef struct video_process_tag {
00243     /*! TRUE : enable 3D deinterlace , False
00244      : disable 3D deinterlace */
00245     int is_3DI;
00246     /*! TRUE : enable 3D denoise , False :
00247      disable 3D denoise */
00248     int is_denoise;
00249     /*! #GM3DIFrameTypeTag */
00250     int denoise_mode;
00251     int reserved[4];
00252 }video_process;
00253
00254 /**
00255  * @brief scalar parameter
00256 */
00257 typedef struct ScalerParamtag {
00258     /*! #ScaleColorModeTag */
00259     int src_fmt;           ///< source form
00260     /*! #ScaleColorModeTag */
00261     int dst_fmt;          ///< destination
00262     /*! #ScaleMethodTag */
00263     int scale_mode;
00264     int is_dither;        ///< Not in
00265     /*! #ScaleMethodTag */
00266     int is_correction;    ///< Not in
00267     /*! #ScaleMethodTag */
00268     int is_album;         ///< Not in
00269     /*! #ScaleMethodTag */
00270     int des_level;        ///< Not in
00271     int reserved[8];
00272 } ScalerParam;
```

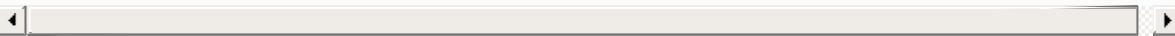
```
00268
00269 typedef enum EncodeType_tag {
00270     ENC_TYPE_H264 = 0,
00271     ENC_TYPE_MPEG,
00272     ENC_TYPE_MJPEG,
00273     ENC_TYPE_YUV422,
00274     ENC_TYPE_COUNT
00275 }EncodeType;
00276
00277 /* debug level */
00278 #define      DBG_ENTITY_FNC          0x01
00279 #define      DBG_ENTITY_JOB_FLOW    0x02
00280 #define      DBG_DVR_FNC            0x04
00281 #define      DBG_DVR_DATA_FLOW     0x08
00282 #define      DBG_GRAPH_FNC          0x10
00283 #define      DBG_GRAPH_DATA        0x20
00284
00285
00286 typedef enum LCDOutputColorTypeTag {
00287     LCD_COLOR_YUV422,
00288     LCD_COLOR_YUV420,
00289     LCD_COLOR_RGB = 16,
00290     LCD_COLOR_ARGB = 16,
00291     LCD_COLOR_RGB888,
00292     LCD_COLOR_RGB565,
00293     LCD_COLOR_RGB555,
00294     LCD_COLOR_RGB444,
00295     LCD_COLOR_RGB8
00296 }LCDOutputColorType;
00297
00298 #define      MCP_VIDEO_NTSC   0    ///< video mode : NTSC
00299 #define      MCP_VIDEO_PAL    1    ///< video mode : PAL
00300 #define      MCP_VIDEO_VGA    2    ///< video mode : VGA
00301
```

```
00302 typedef enum LCDOutputModeTag {
00303     LCD_PROGRESSIVE = 0,
00304     LCD_INTERLACING = 1
00305 }LCDOutputMode;
00306
00307 typedef enum LiveviewFrameTypeTag {
00308     LVFRAME_EVEN_ODD=0,
00309     LVFRAME_ENLARGE_ONE_FIELD=1,
00310     LVFRAME_WEAVED_TWO_FIELDS=2,
00311     LVFRAME_GM3DI_FORMAT=3
00312 }LiveviewFrameType;
00313
00314 typedef enum LiveviewFrameModeTag {
00315     LVFRAME_FRAME_MODE=0,
00316     LVFRAME_FIELD_MODE=1,
00317     LVFRAME_FIELD_MODE2=2 /* Treat every field as a complete frame, must work with LVFRAME_ENLARGE_ONE_FIELD */
00318 }LiveviewFrameMode;
00319
00320 typedef enum GM3DIFrameTypeTag {
00321     GM3DI_FIELD=1,
00322     GM3DI_FRAME=2
00323 }GM3DIFrameType;
00324
00325
00326 typedef enum LiveviewDMAOrderTag {
00327     DMAORDER_PACKET=0,
00328     DMAORDER_3PLANAR=1,
00329     DMAORDER_2PLANAR=2
00330 }LiveviewDMAOrder;
00331
00332 typedef enum LiveviewScalerRatioTag {
00333     CAPSCALER_KEEP_RATIO=0,
00334     CAPSCALER_NOT_KEEP_RATIO=1
00335 }LiveviewScalerRatio;
00336
```

```
00337
00338 typedef enum CaptureColorModeTag {
00339     CAPCOLOR_RGB888=0,
00340     CAPCOLOR_RGB565=1,
00341     CAPCOLOR_YUV422=2,
00342     CAPCOLOR_YUV420_M0=3,
00343     CAPCOLOR_YUV420_M1=4
00344 }CaptureColorMode;
00345
00346 typedef enum EncoderInputFormatTag {
00347     ENC_INPUT_H2642D=0,
00348     ENC_INPUT_MP42D=1,
00349     ENC_INPUT_1D420=2,
00350     ENC_INPUT_1D422=3
00351 }EncoderInputFormat;
00352
00353 typedef enum DecoderOutputColorTag {
00354     DEC_OUTPUT_COLOR_YUV420=4,
00355     DEC_OUTPUT_COLOR_YUV422=5
00356 }DecoderOutputColor;
00357
00358 typedef enum JpegEncInputFormatTag{
00359     JCS_yuv420 = 0,
00360     JCS_yuv422 = 1,
00361     JCS_yuv211 = 2,
00362     JCS_yuv333 = 3,
00363     JCS_yuv222 = 4,
00364     JCS_yuv111 = 5,
00365     JCS_yuv400 = 6
00366 } JpegEncInputFormat;
00367
00368 typedef enum JpegEnc420InputFormatTag{
00369     JENC_INPUT_MP42D=0,
00370     JENC_INPUT_1D420=1,
00371     JENC_INPUT_H2642D=2,
00372     JENC_INPUT_DMAWRP420=3,
```

```
00373             JENC_INPUT_1D422=4
00374 } JpegEnc420InputFormat;
00375
00376 typedef enum ScaleMethodTag {
00377     SCALE_LINEAR=0,
00378     SCALE_NON_LINEAR,
00379     SCALE_METHOD_COUNT
00380 }ScaleMethod;
00381
00382 typedef enum ScaleColorModeTag {
00383     SCALE_RGB888=0,
00384     SCALE_RGB565=1,
00385     SCALE_H264_YUV420_MODE0=2,
00386     SCALE_H264_YUV420_MODE1=3,
00387     SCALE_YUV444=4,
00388     SCALE_YUV422=5,
00389     SCALE_MP4_YUV420_MODE0=6,
00390     SCALE_MP4_YUV420_MODE1=7
00391 }ScaleColorMode;
00392
00393
00394 typedef enum CapturePathTag {
00395     CAPPATH_DEFAULT=0, //If Liveview, use path1, If Record, use path2
00396     CAPPATH_PATH_1,
00397     CAPPATH_PATH_2
00398 }CapturePath;
00399
00400
00401 typedef enum LCDResolutionTag {
00402     LCD_RES_D1,          /* 720x576 or 720x480 */
00403     LCD_RES_SVGA,         /* 800x600 */
00404     LCD_RES_XGA,          /* 1024x768 */
00405     LCD_RES_XVGA,         /* 1280x960 */
00406     LCD_RES_SXGA,         /* 1280x1024 */
00407     LCD_RES_1360x768,      /* 1360x768 */
```

```
00408     LCD_RES_COUNT  
00409 }LCDResolution;  
00410  
00411  
00412 #define      DEFAULT_D1_WIDTH      720  
00413 #define      DEFAULT_D1_HEIGHT     576  
00414 #define      DEFAULT_CIF_WIDTH     352  
00415 #define      DEFAULT_CIF_HEIGHT    288  
00416 #define      DEFAULT_QCIF_WIDTH   176  
00417 #define      DEFAULT_QCIF_HEIGHT  144  
00418  
00419 #endif /* __DVR_TYPE_DEFINE_H__ */  
00420  
00421  
00422  
00423  
00424
```



Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

source/dvr_disp_api.h

Go to the documentation of this file.

```
00001 #ifndef __DVR_DISP_API_H__
00002 #define __DVR_DISP_API_H__
00003
00004 #include <linux/ioctl.h>
00005
00006 #include "dvr_type_define.h"
00007 #include "dvr_disp_ioctl.h"
00008
00009 #define DVR_PLANE_ID(d,p) ((d<<8)|p)
00010 #define GET_DISP_NUM_FROM_PLANE_ID(h) (h
>>8)
00011 #define GET_PLANE_NUM_FROM_PLANE_ID(h) (0
xFF&h)
00012
00013
00014 enum dvr_disp_type {
00015     DISP_TYPE_LIVEVIEW,
00016     DISP_TYPE CASCADE,
00017     DISP_TYPE_PLAYBACK,
00018     DISP_TYPE_ON_BUFFER,
00019 };
00020
00021 /* TODO: the following should be replaced by
   including capture header file.... */
00022 /*
00023 #define VIDEO_MODE_PAL      0
00024 #define VIDEO_MODE_NTSC    1
00025 #define VIDEO_MODE_SECAM   2
00026 #define VIDEO_MODE_AUTO    3
00027 */
00028 /**
00029
```

```
00030
00031 //### Declaration for DVR_DISP_GET_DISP PARA
M and DVR_DISP_SET_DISP_PARAM ###
00032 typedef struct dvr_disp_color_attribute_tag
{
00033     int brightness;
00034     int saturation;
00035     int contrast;
00036     int huesin;
00037     int huecos;
00038     int sharpnessk0;
00039     int sharpnessk1;
00040     int sharpness_thres0;
00041     int shaprness_thres1;
00042     int reserved[2];
00043 }dvr_disp_color_attribute;
00044
00045 typedef struct dvr_disp_color_key_tag {
00046     int is_enable;
00047     int color;
00048     int reserved[2];
00049 }dvr_disp_color_key;
00050
00051 typedef enum dvr_disp_plane_combination_tag
{
00052     BG_ONLY          =0,    ///< only background
00053     BG_AND_1PLANE   =1,    ///< background and another plane
00054     BG_AND_2PLANE   =2,    ///< background and another 2 planes
00055 }dvr_disp_plane_combination;
00056
00057 typedef struct dvr_disp_vbi_info_tag {
00058     int filler;        ///< 0:driver do it,
                           1:ap can set values for it(need to fill the following fields)
```

```
00059     int lineno;
00060     int lineheight;
00061     int fb_offset;
00062     int fb_size;
00063     int reserved[2];
00064 }dvr_disp_vbi_info;
00065
00066 typedef struct dvr_disp_scaler_info_tag {
00067     int is_enable;
00068     /*! #DIM_tag      */
00069     DIM dim;
00070     int reserved[2];
00071 }dvr_disp_scaler_info;
00072
00073 typedef struct dvr_disp_resolution_tag {
00074     int input_res;
00075     int output_type;
00076     int reserved[2];
00077 }dvr_disp_resolution;
00078
00079
00080 typedef struct dvr_disp_disp_param_tag {
00081     int disp_num;    ///< input(write only)
00082
00083     int target_id;    ///< where to output
. 1:LCD1, 2:LCD2
00084     /*! #dvr_disp_plane_combination_tag */
00085     int plane_comb;    ///< the number of active plane for this LCD
00086     /*! #MCP_VIDEO_NTSC */
00087     /*! #MCP_VIDEO_PAL */
00088     /*! #MCP_VIDEO_VGA */
00089     int output_system;
00090     /*! #LCDOutputModeTag */
00091     int output_mode;
00092     /*! #dvr_disp_color_attribute_tag */
00093     dvr_disp_color_attribute color_attrib;
```

```
00094     /*! #dvr_disp_color_key_tag */
00095     dvr_disp_color_key transparent_color[2];
00096     /*! #dvr_disp_vbi_info_tag */
00097     dvr_disp_vbi_info vbi_info;
00098     /*! #dvr_disp_scaler_info_tag */
00099     dvr_disp_scaler_info scl_info;
00100    /*! #DIM_tag */
00101    DIM dim;           ///< default background
und dimension
00102    /*! #dvr_disp_resolution_tag */
00103    dvr_disp_resolution res;
00104    int reserved[8];
00105 }dvr_disp_disp_param;
00106
00107
00108
00109 //### Declaration for DVR_DISP_UPDATE_DISP_P
ARAM ###
00110 typedef enum dvr_disp_disp_param_name_tag {
00111     DISP_PARAM_TARGET,
00112     DISP_PARAM_PLANE_COMBINATION,
00113     DISP_PARAM_OUTPUT_MODE,
00114     DISP_PARAM_OUTPUT_SYSTEM,
00115     DISP_PARAM_COLOR_ATTRIBUTE,
00116     DISP_PARAM_TRANSPARENT_COLOR,
00117     DISP_PARAM_RESOLUTION,
00118     DISP_PARAM_APPLY      /* Use this to apply all parameters. */
00119 }dvr_disp_disp_param_name;
00120
00121
00122 typedef struct dvr_update_disp_param_tag
{
00123     int disp_num;    ///< input
00124     /*! #dvr_disp_disp_param_name_tag */
00125     dvr_disp_disp_param_name param;
```

```
00126     // output
00127     struct val_t {
00128         int target_id;
00129         /*! #dvr_disp_plane_combination_tag
*/
00130         int plane_comb;
00131         /*! #MCP_VIDEO_NTSC */
00132         /*! #MCP_VIDEO_PAL */
00133         /*! #MCP_VIDEO_VGA */
00134         int output_system;
00135         /*! #LCDOutputModeTag */
00136         int output_mode;
00137         int display_rate;    ///< NTSC : 30,
PAL : 25
00138         /*! #dvr_disp_color_attribute_tag
*/
00139         dvr_disp_color_attribute color_at
trib;
00140         /*! #dvr_disp_color_key_tag */
00141         dvr_disp_color_key transparent_color
[2];
00142         /*! #dvr_disp_vbi_info_tag */
00143         dvr_disp_vbi_info vbi_info;
00144         /*! #dvr_disp_scaler_info_tag */
00145         dvr_disp_scaler_info scl_info;
00146         /*! #dvr_disp_resolution_tag */
00147         dvr_disp_resolution res;
00148         int reserved[8];
00149     }val;
00150 }dvr_disp_update_disp_param;
00151
00152
00153
00154 //### Declaration for DVR_DISP_GET_PLANE_PAR
AM and DVR_DISP_SET_PLANE_PARAM ###
00155 typedef struct dvr_disp_plane_param_tag {
00156     int plane_id;
```

```
00157     /*! #RECT_tag */  
00158     RECT  win;  
00159     /*! #LCDOutputModeTag */  
00160     int   data_mode;  
00161     /*! #LCDOutputColorTypeTag */  
00162     int   color_mode;  
00163     int   reserved[2];  
00164 }dvr_disp_plane_param_st;  
00165  
00166 typedef struct dvr_disp_plane_param_st_tag {  
00167     //input  
00168     int disp_num;  
00169     int plane_num;  
00170     /*! #dvr_disp_plane_param_tag */  
00171     dvr_disp_plane_param_st param;  
00172     int reserved[2];  
00173 }dvr_disp_plane_param;  
00174  
00175  
00176 //### Declaration for DVR_DISP_UPDATE_PLANE_  
PARAM ###  
00177 typedef enum dvr_disp_plane_param_name_tag {  
00178     PLANE_PARAM_COLOR_MODE,  
00179     PLANE_PARAM_WINDOW,  
00180     PLANE_PARAM_DATA_MODE,  
00181     PLANE_PARAM_APPLY  
00182 }dvr_disp_plane_param_name;  
00183  
00184 typedef struct dvr_disp_update_plane_param_t  
ag {  
00185     //input  
00186     int   plane_id;  
00187     /*! #dvr_disp_plane_param_name_tag */  
00188     dvr_disp_plane_param_name param;  
00189     int reserved[8];  
00190     union {  
00191         /*! #RECT_tag */  
00192     }
```

```
00192         RECT  win;
00193         /*! #LCDOutputModeTag      */
00194         int   data_mode;
00195         /*! #LCDOutputColorTypeTag  */
00196         int   color_mode;
00197     }val;
00198     int reserved1[8];
00199 }dvr_disp_update_plane_param;
00200
00201
00202 //### Declaration for DVR_DISP_CONTROL ###
00203 typedef enum dvr_disp_ctrl_cmd_tag {
00204     DISP_START,
00205     DISP_STOP,
00206     DISP_UPDATE,
00207     DISP_RUN,
00208 }dvr_disp_ctrl_cmd;
00209 //+++ stanley add
00210 typedef enum dvr_disp_channel_type_tag {
00211     DISP_NORMAL_CHN,
00212     DISP_LAYER0_CHN,
00213     DISP_LAYER1_CHN,
00214 }dvr_disp_channel_type;
00215
00216 typedef struct DispParam_Ext1_tag {
00217     /*! #dvr_disp_channel_type_tag */
00218     dvr_disp_channel_type chn_type;
00219 }DispParam_Ext1;
00220 //---
00221 #define DVR_PARAM_MAGIC 0x1688
00222 #define DVR_PARAM_MAGIC_SHIFT    16
00223 #define CAP_BUF_ID(id) ((DVR_PARAM_MAGIC <<
00224     DVR_PARAM_MAGIC_SHIFT)|(id))
00224 //+++ stanley add
00225 #define DVR_DISP_MAGIC_ADD_VAL(val) ((DVR_P
ARAM_MAGIC << DVR_PARAM_MAGIC_SHIFT)|(val))
```

```

00226 #define DVR_DISP_CHECK_MAGIC(val) (((val)>>
DVR_PARAM_MAGIC_SHIFT)==DVR_PARAM_MAGIC)
00227 #define DVR_DISP_GET_VALUE(val) ((val)&((
1<<DVR_PARAM_MAGIC_SHIFT)-1))
00228 //---
00229
00230 typedef struct dvr_disp_control_tag {
00231     /*! #dvr_disp_type */
00232     int type;
00233     int channel;
00234     /*! #dvr_disp_ctrl_cmd_tag */
00235     dvr_disp_ctrl_cmd command;
00236
00237     int reserved[8];
00238
00239     struct
00240     {
00241         struct {
00242             int cap_path;
00243             /*! #LiveviewFrameTypeTag */
00244             int di_mode;
00245             /*! #LiveviewFrameModeTag */
00246             int mode;
00247             /*! #LiveviewDMAOrderTag */
00248             int dma_order;
00249             /*! #LiveviewScalerRatioTag */
00250             int scale_indep;      ///< 0:f
fixed aspect-ratio
00251             /*! #MCP_VIDEO_NTSC */
00252             /*! #MCP_VIDEO_PAL */
00253             /*! #MCP_VIDEO_VGA */
00254             int input_system;
00255             int cap_rate;
00256             /*! #CaptureColorModeTag */
00257             int color_mode;
00258             int is_use_scaler;    //if TRUE,
E, capture as "src_para.lv.dim" and scale it to "d

```

```
es_param.lv.win"
00259                                     //if FALSE,
SE, capture as "des_param.lv.win" and display it directly
00260             /*! #DIM_tag      */
00261             DIM    dim;
00262             /*! #RECT_tag      */
00263             RECT   win;
00264             /*! #video_process_tag */
00265             video_process vp_param;
00266             /*! #ScalerParamtag */
00267             ScalerParam scl_param; //if is_
use_scaler is TRUE, user needs to fill this structure
00268             int    cap_buf_id;
00269             /*! display parameter extension
size */
00270             int    ext_size;
00271             /*! point to display parameter e
xtension structure */
00272             void   *pext_data;
00273             int    reserved[2];
00274         }lv;
00275     struct {
00276             /*! #RECT_tag      */
00277             RECT   win;
00278             int    rate;
00279             /*! #LiveviewFrameTypeTag */
00280             int    di_mode;
00281             /*! #LiveviewFrameModeTag */
00282             int    mode;
00283             /*! #LiveviewDMAOrderTag */
00284             int    dma_order;
00285             /*! #LiveviewScalerRatioTag */
00286             int    scale_indep;
00287             /*! #MCP_VIDEO_NTSC */
00288             /*! #MCP_VIDEO_PAL */
```

```

00289             /*! #MCP_VIDEO_VGA */
00290             int    input_system;
00291             int    cap_rate;
00292             /*! #CaptureColorModeTag */
00293             int    color_mode;
00294             int    is_use_scaler; //if TRUE,
E, capture as "src_para.lv.dim" and scale it to "des_param.lv.win"
00295                                         //if FALSE,
SE, capture as "des_param.lv.win" and display it directly
00296             /*! #DIM_tag */
00297             DIM    dim;
00298             /*! #video_process_tag */
00299             video_process vp_param;
00300             /*! #ScalerParamtag */
00301             ScalerParam scl_param; //if
is_use_scaler is TRUE, user needs to fill this structure
00302             int    reserved[5];
00303         }cas;
00304         int    reserved[5];
00305     }src_param;
00306
00307     struct
00308     {
00309         struct {
00310             /*! #RECT_tag */
00311             RECT   win;
00312             int    plane_id;
00313             int    reserved[5];
00314         }lv;
00315
00316         struct {
00317             int    path;
00318             /*! #RECT_tag */
00319             RECT   win;

```

```

00320             /*! #RECT_tag */  

00321             RECT rect0;  

00322             /*! #RECT_tag */  

00323             RECT rect1;  

00324             /*! #RECT_tag */  

00325             RECT swc_rect0;  

00326             /*! #RECT_tag */  

00327             RECT swc_rect1;  

00328             /*! #RECT_tag */  

00329             RECT bg_rect0;  

00330             /*! #RECT_tag */  

00331             RECT bg_rect1;  

00332             int plane_id;  

00333             int reserved[5];  

00334         }cas;  

00335         int reserved[5];  

00336     }dst_param;  

00337 }dvr_disp_control;  

00338  

00339  

00340 typedef struct dvr_disp_clear_param_tag  

00341 {  

00342     int plane_id;  

00343     /*! #RECT_tag */  

00344     RECT win;  

00345     unsigned int pattern;  

00346     int reserved[2];  

00347 }dvr_disp_clear_param;  

00348  

00349 #endif /* __DVR_DISP_API_H__ */  

00350  

00351

```



[Main Page](#)[Data Structures](#)[Files](#)[Examples](#)[File List](#)[Globals](#)

source/dvr_enc_api.h

Go to the documentation of this file.

```
00001 #ifndef __DVR_ENC_API_H__
00002 #define __DVR_ENC_API_H__
00003
00004 #include <linux/ioctl.h>
00005 #include "dvr_type_define.h"
00006 #include "dvr_enc_ioctl.h"
00007
00008 typedef enum dvr_enc_src_type_tag {
00009     ENC_TYPE_FROM_CAPTURE=0,
00010     ENC_TYPE_FROM CASCADE,
00011     ENC_TYPE_FROM_BUFFER,
00012     ENC_SRC_TYPE_COUNT
00013 }dvr_enc_src_type;
00014
00015 /**
00016 * @brief no longer used, replace by EncPara
m_Ext4
00017 */
00018 typedef struct EncParam_Ext1_tag{
00019     unsigned int feature_enable;
00020     unsigned int target_rate_max;
00021     unsigned int reaction_delay_max;
00022 }EncParam_Ext1;
00023
00024 /**
00025 * @brief no longer used, replace by EncPara
m_Ext4
00026 */
00027 typedef struct EncParam_Ext2_tag{
00028     unsigned int feature_enable;
00029     unsigned int target_rate_max;
00030     unsigned int reaction_delay_max;
```

```
00031     int          enc_type;
00032     int          MJ_quality;
00033 }EncParam_Ext2;
00034
00035 /**
00036 * @brief no longer used, replace by EncPara
m_Ext4
00037 */
00038 typedef struct  EncParam_Ext3_tag{
00039     /*! #DVR_ENC_ENHANCE_H264_RATECONTROL
*/
00040     /*! #DVR_ENC_MJPEG_FUNCTION
*/
00041     /*! #DVR_ENC_H264_WATERMARK
*/
00042     unsigned int    feature_enable;    ///<
this is a bit combination. If you want to involve
watermark, then feature_enable |= DVR_ENC_H264_WAT
ERMARK.
00043     /*! ECBR: usually stays at target bitrat
e, */
00044     /*! produce large bitrate when big motio
n, but never more than max bitrate */
00045     unsigned int    target_rate_max;
00046     /*! max frame count will the rate-contro
l stay when away from the target bitrate, only val
id at ECBR */
00047     unsigned int    reaction_delay_max;
00048     /*! #EncodeType_tag */
00049     int            enc_type;
00050     /*! 1(worst) ~100(best) */
00051     int            MJ_quality;      ///< Moti
on JPEG quality
00052     /*! Indicate watermark enable or not
*/
00053     /*! if 0: disable,
*/
```

```
00054     /*! if 1: enable, but only insert watermark at intra-mb, */
00055     /*! if 3: enable, and insert watermark at intra/inter-mb, */
00056     /*! if others: not allowed */
00057     int             watermark_enable;
00058     /*! specify the interval, to insert watermark each "interval" frame, */
00059     /*! valid when watermark function is enabled, */
00060     /*! if 1: will be inserted watermark each frames, */
00061     /*! if N: will be inserted watermark each N frames */
00062     int             watermark_interval;
00063     /*! The initial pattern for watermark */
00064     /*! valid when watermark function is enabled */
00065     int             watermark_init_pattern;
00066     /*! specify the interval, to reinit with init_pattern each "watermark"ed frames */
00067     /*! valid when watermark function is enabled */
00068     /*! if 1: every "watermark"ed frame with init_pattern */
00069     /*! if N: initial with init_pattern each N "watermark"ed frame */
00070     int             watermark_init_interval;
00071 }EncParam_Ext3;
00072 /**
00073 * @brief encode parameter extension
00074 */
00075 typedef struct EncParam_Ext4_tag{
```

```
00077     /*! #DVR_ENC_ENHANCE_H264_RATECONTROL
*/
00078     /*! #DVR_ENC_MJPEG_FUNCTION
*/
00079     /*! #DVR_ENC_H264_WATERMARK
*/
00080     /*! #DVR_ENC_ROI_POS          */
00081     unsigned int    feature_enable;    ///<
this is a bit combination. If you want to involve
watermark, then feature_enable |= DVR_ENC_H264_WAT
ERMARK.
00082 /*! for New rate control */
00083     /*! ECBR: usually stays at target bitrat
e, */
00084     /*! produce large bitrate when big motio
n, but never more than max bitrate */
00085     unsigned int    target_rate_max;
00086     /*! max frame count will the rate-contro
l stay when away from the target bitrate, only val
id at ECBR */
00087     unsigned int    reaction_delay_max;
00088 /*! for MJPEG */
00089     /*! #EncodeType_tag          */
00090     int            enc_type;
00091     /*! 1(worst) ~100(best)      */
00092     int            MJ_quality;    ///< Moti
on JPEG quality
00093 /*! for H264 Water Mark */
00094     /*! Indicate watermark enable or not
*/
00095     /*! if 0: disable,
*/
00096     /*! if 1: enable, but only insert waterm
ark at intra-mb, */
00097     /*! if 3: enable, and insert warermark a
t intra/inter-mb, */
00098     /*! if others: not allowed */
```

```
00099     int           watermark_enable;
00100     /*! specify the interval, to insert wate
rmark each "interval" frame,      */
00101     /*! valid when watermark function is ena
bled,                                */
00102     /*! if 1: will be inserted watermark eac
h frames,                         */
00103     /*! if N: will be inserted watermark eac
h N frames                         */
00104     int           watermark_interval;
00105     /*! The initial pattern for watermark
*/
00106     /*! valid when watermark function is ena
bled   */
00107     int           watermark_init_pattern;
00108     /*! specify the interval, to reinit with
 init_pattern each "watermark"ed frames */
00109     /*! valid when watermark function is ena
bled   */
00110     /*! if 1: every "watermark"ed frame with
 init_pattern                         */
00111     /*! if N: initial with init_pattern each
 N "watermark"ed frame                */
00112     int           watermark_init_interval;

00113 /*! for ROI x, y position */
00114     POS           roi_pos;
00115 }EncParam_Ext4;
00116
00117 typedef struct EncParam_Ext5_tag{
00118     /*! #DVR_ENC_ENHANCE_H264_RATECONTROL
*/
00119     /*! #DVR_ENC_MJPEG_FUNCTION
*/
00120     /*! #DVR_ENC_H264_WATERMARK
*/
```

```
00121     /*! #DVR_ENC_ROI_POS */  
00122     /*! #DVR_ENC_ROI_ALL */  
00123     unsigned int    feature_enable;    ///<  
this is a bit combination. If you want to involve  
watermark, then feature_enable |= DVR_ENC_H264_WAT  
ERMARK.  
00124 /*! for New rate control */  
00125     /*! ECBR: usually stays at target bitrat  
e, */  
00126     /*! produce large bitrate when big motio  
n, but never more than max bitrate */  
00127     unsigned int    target_rate_max;  
00128     /*! max frame count will the rate-contro  
l stay when away from the target bitrate, only val  
id at ECBR */  
00129     unsigned int    reaction_delay_max;  
00130 /*! for MJPEG */  
00131     /*! #EncodeType_tag */  
00132     int            enc_type;  
00133     /*! 1(worst) ~100(best) */  
00134     int            MJ_quality;    ///< Moti  
on JPEG quality  
00135 /*! for H264 Water Mark */  
00136     /*! Indicate watermark enable or not  
*/  
00137     /*! if 0: disable,  
*/  
00138     /*! if 1: enable, but only insert waterm  
ark at intra-mb, */  
00139     /*! if 3: enable, and insert warermark a  
t intra/inter-mb, */  
00140     /*! if others: not allowed */  
  
00141     int            watermark_enable;  
00142     /*! specify the interval, to insert wate  
rmark each "interval" frame, */  
00143     /*! valid when watermark function is ena
```

```

bled,                                */
00144     /*! if 1: will be inserted watermark each frames,           */
00145     /*! if N: will be inserted watermark each N frames           */
00146     int          watermark_interval;
00147     /*! The initial pattern for watermark */
00148     /*! valid when watermark function is enabled */
00149     int          watermark_init_pattern;
00150     /*! specify the interval, to reinit with init_pattern each "watermark"ed frames */
00151     /*! valid when watermark function is enabled */
00152     /*! if 1: every "watermark"ed frame with init_pattern */
00153     /*! if N: initial with init_pattern each N "watermark"ed frame */
00154     int          watermark_init_interval;

00155 /*! for ROI x, y position */
00156     POS          roi_pos;
00157 /*! for update ROI all function, x, y position, width, height, enable/disable */
00158     ROI_ALL      roi_all;
00159 }EncParam_Ext5;
00160
00161 /**
00162 * @brief encode parameter
00163 */
00164 typedef struct EncParam_tag {
00165     /*! #EncodeType_tag           */
00166     int      input_type;
00167     /*! frame rate per second   */
00168     int      frame_rate;
00169     /*! bit rate per second     */

```

```
00170     int      bit_rate;
00171     /*! The I-P interval frames */
00172     int      ip_interval;
00173     /*! The initial quant value */
00174     int      init_quant;
00175     /*! The max/min quant value */
00176     int      max_quant, min_quant;
00177     /*! Enable the ROI function */
00178     int      is_use_ROI;
00179     /*! #RECT_tag */
00180     RECT    ROI_win;
00181     /*! encode parameter extension size */
00182     unsigned int ext_size;
00183     /*! point to encode parameter extension
structure */
00184     void    *pext_data;
00185     int      reserved[6];
00186 } EncParam;
00187
00188 /**
00189  * @brief snapshot parameter
00190 */
00191 typedef struct snapshot_param_tag {
00192     /*! #JpegEncInputFormatTag */
00193     int sample;
00194     /*! not in use */
00195     int RestartInterval;
00196     /*! #JpegEnc420InputFormatTag */
00197     int u82D;
00198     /*! 1(worst) ~100(best) */
00199     int quality;
00200     int reserved[8];
00201 } snapshot_param;
00202
00203 /**
00204  * @brief dvr bit stream parameter, include
main, sub1, sub2 bit-stream.
```

```
00205  */
00206 typedef struct ReproduceBitStream_tag {
00207     /*! #DVR_ENC_EBST_ENABLE: enabled */
00208     /*! #DVR_ENC_EBST_DISABLE: disabled */
00209     int enabled;
00210     /*! 0: main-bitstream */
00211     /*! 1: sub-bitstream1 */
00212     /*! 2: sub-bitstream2 */
00213     int out_bs;
00214     /*! #EncodeType_tag */
00215     int enc_type;
00216     /*! indicate all system-call for this channel is "blocked" or "non-block" type */
00217     int is_blocked;
00218     /*! #DVR_ENC_EBST_ENABLE: enabled */
00219     /*! #DVR_ENC_EBST_DISABLE: disabled */
00220     int en_snapshot;    ///< enable/disable snapshot
00221     /*! #DIM_tag */
00222     DIM dim;
00223     /*! #EncParam_tag */
00224     EncParam enc;
00225     /*! #ScalerParamtag */
00226     ScalerParam scl;
00227     /*! #snapshot_param_tag */
00228     snapshot_param snap;
00229     int reserved[8];
00230 } ReproduceBitStream;
00231
00232 /**
00233 * @brief dvr encode source parameter
00234 */
00235 typedef struct dvr_enc_src_tag {
00236     /*! channle number 0~7 */
00237     int channel;
00238     /*! #dvr_enc_src_type_tag */
00239     int enc_src_type;
```

```
00240     /*! #DIM_tag      */
00241     DIM      dim;
00242     /*! #LiveviewFrameTypeTag
 * /
00243     int      di_mode;
00244     /*! #LiveviewFrameModeTag          */
00245     int      mode;
00246     /*! #LiveviewDMAOrderTag          */
00247     int      dma_order;
00248     /*! #LiveviewScalerRatioTag        */
00249
00250     int      scale_indep;
00251     /*! #MCP_VIDEO_NTSC   */
00252     /*! #MCP_VIDEO_PAL    */
00253     /*! #MCP_VIDEO_VGA    */
00254     int      input_system;
00255     /*! #CaptureColorModeTag        */
00256     int      color_mode;
00257     /*! #video_process_tag   */
00258     video_process vp_param;
00259     int reserved[8];
00260 } dvr_enc_src_param;
00261 /**
00262 * @brief dvr encode channel parameter.
00263 */
00264 typedef struct dvr_enc_channel_param_tag {
00265     dvr_enc_src_param src;
00266     ReproduceBitStream main_bs;
00267 }dvr_enc_channel_param;
00268
00269 typedef enum dvr_enc_channel_param_name_tag
{
00270     ENC_PARAM_SRC_DIM,
00271     ENC_PARAM_DST_WIN
00272 }dvr_enc_channel_param_name;
00273
```

```
00274 /**
00275  * @brief get dvr encode buffer
00276 */
00277 typedef struct dvr_enc_queue_get_tag {
00278     /*! #dvr_bs_data_tag */
00279     dvr_bs_data bs;
00280     int new_bs;      ///< To indicate whether
the packet using new setting
00281     int mb_len;      ///< Only for debug H.26
4: D1:103680 HD1:50688 CIF:25344
00282     int channel;    ///< Only for debug H.26
4: sub ==> bs.stream
00283 }dvr_enc_queue_get;
00284
00285
00286 // Declaration for DVR_ENC_CONTROL
00287 enum dvr_enc_ctrl_cmd {
00288     ENC_START,
00289     ENC_STOP,
00290     ENC_SNAP,
00291     ENC_UPDATE,
00292     ENC_RAW
00293 };
00294
00295 /**
00296  * @brief update dvr encode channel parameter
00297 */
00298 typedef struct dvr_enc_update_channel_param_
tag {
00299     struct
00300     {
00301         /*! #LiveviewFrameTypeTag
*/
00302         int     di_mode;
00303         /*! #LiveviewFrameModeTag           */
00304         int     mode;
```

```
00305     /*! #LiveviewScalerRatioTag          */
00306     int      scale_indep;
00307     /*! TRUE : enable 3D deinterlace , False
00308      : disable 3D deinterlace   */
00309     int      is_3DI;
00310     /*! TRUE :  enable 3D denoise , False :
00311      disable 3D denoise   */
00312     int      is_denoise;
00313     int      reserved[8];
00314 } src;
00315     int      stream_enable; ///< 0:disable,
00316      1:enable
00317     int      frame_rate;    ///< frame rate
00318      per second
00319     int      bit_rate;     ///< Bitrate per
00320      second
00321     int      ip_interval; ///< The I-P int
00322      erval frames
00323     /*! #DIM_tag      */
00324     DIM      dim;
00325     int      init_quant; ///< The initial
00326      quant value
00327     int      max_quant; ///< The max qua
00328      nt value
00329     int      min_quant; ///< The min qua
00330      nt value
00331     /*! encode parameter extension size */
00332
00333     unsigned int ext_size;
00334     /*! point to encode parameter extension
00335      structure */
00336     void      *pext_data;
00337     int      reserved[8];
00338 } dvr_enc_update_channel_param;
00339 /**
```

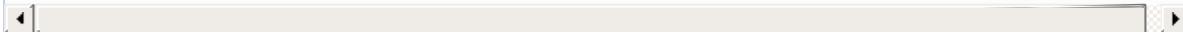
```
00330 * @brief dvr encode control parameter
00331 */
00332 typedef struct dvr_enc_control_tag {
00333     int      command;           ///< enc control
00334     command , such as ENC_START, ENC_STOP, etc.
00335     union {
00336         int *count;
00337         int *repd_bs_num;    ///< 0: all, 1:
00338         SUB1, 2: SUB2.
00339         }output;
00340         int          stream;   ///< stream numb
00341 er 0 : main, 1 : sub stream 1, 2 : sub stream 2
00342     /*! #dvr_enc_update_channel_param_tag
00343 */
00344     dvr_enc_update_channel_param      update_p
00345 arm;
00346     int reserved[8];
00347 }dvr_enc_control;
00348
00349
00350
00351     /* return value by driver */
00352     struct timeval timestamp;
00353     unsigned int bs_length;    ///< bitstream
length by AP prepared, and return the real bitstr
eam length
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
00999
```

```
00354     unsigned int mv_length; ///< motion length by AP prepared, and return the real motion length
00355
00356     unsigned int is_keyframe; ///< 1: current frame is a keyframe. 0: current frame is not a keyframe.
00357     int stream;
00358     int NonRef;
00359     int new_bs; ///< To indicate whether the packet using new setting
00360     void *p_job; ///< internal use
00361     int reserved[16];
00362 } dvr_enc_copy_buf;
00363
00364
00365 #define POLLIN_MAIN_BS      0x0001
00366 #define POLLIN_SUB1_BS      (POLLIN_MAIN_BS <
00367 < 1)
00368 #define POLLIN_SUB2_BS      (POLLIN_MAIN_BS <
00369 < 2)
00370 #define POLLIN_SUB3_BS      (POLLIN_MAIN_BS <
00371 < 3)
00372 #define POLLIN_SUB4_BS      (POLLIN_MAIN_BS <
00373 < 4)
00374 #define POLLIN_SUB5_BS      (POLLIN_MAIN_BS <
00375 < 5)
00376 #define POLLIN_SUB6_BS      (POLLIN_MAIN_BS <
00377 < 6)
00378 #define POLLIN_SUB7_BS      (POLLIN_MAIN_BS <
00379 < 7)
00380 #define POLLIN_SUB8_BS      (POLLIN_MAIN_BS <
00381 < 8)
00382 #define POLLIN_SNAP_BS      (1 << DVR_ENC_REP_D_BT_NUM)
00383
```

```

00376 #define DVR_ENC_MAGIC          0x1689
00377 #define DVR_ENC_MAGIC_SHIFT    16
00378 #define DVR_ENC_MAGIC_ADD_VAL(val) ((DVR_EN
C_MAGIC << DVR_ENC_MAGIC_SHIFT)|(val))
00379
00380 #define DVR_ENC_CHECK_MAGIC(v) (((v)>>DVR_E
NC_MAGIC_SHIFT)==DVR_ENC_MAGIC)
00381 #define DVR_ENC_GET_VALUE(v)     ((v)&((1<<DV
R_ENC_MAGIC_SHIFT)-1))
00382 #define DVR_ENC_ENHANCE_H264_RATECONTROL 1
00383 #define DVR_ENC_MJPEG_FUNCTION        (1
<< 1)
00384 #define DVR_ENC_H264_WATERMARK       (1
<< 2)
00385 #define DVR_ENC_ROI_POS             (1
<< 3)
00386 #define DVR_ENC_ROI_ALL             (1
<< 4)
00387
00388
00389 #define DVR_ENC_EBST_ENABLE         0x55887799
00390 #define DVR_ENC_EBST_DISABLE        0x0
00391 #endif /* __DVR_ENC_API_H__ */
00392
00393

```



Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples
Data Structures	Data Fields		
All	Variables		
b c d e f h i l m n o p q r s t u v w x y			

- b -

- bg_rect0 : [dvr_disp_control_tag](#)
- bg_rect1 : [dvr_disp_control_tag](#)
- bit_rate : [dvr_enc_update_channel_param_tag](#) , [EncParam_tag](#)
- brightness : [dvr_disp_color_attribute_tag](#)
- bs : [dvr_enc_queue_get_tag](#) , [dvr_dec_queue_get_tag](#)
- bs_buf_length : [dvr_enc_copy_buf_tag](#)
- bs_length : [dvr_enc_copy_buf_tag](#)
- bs_rate : [dvr_dec_control_tag](#)
- bs_user_va : [dvr_enc_copy_buf_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- C -

- cap_buf_id : [dvr_disp_control_tag](#)
- cap_path : [dvr_disp_control_tag](#)
- cap_rate : [dvr_disp_control_tag](#)
- cas : [dvr_disp_control_tag](#)
- cas_ch : [FuncTag_tag](#)
- channel : [dvr_dec_queue_get_tag](#) , [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#) , [dvr_enc_queue_get_tag](#) , [dvr_dec_channel_param_tag](#)
- chn_type : [DispParam_Ext1_tag](#)
- color : [dvr_disp_color_key_tag](#)
- color_attrib : [dvr_disp_update_disp_param_tag::val_t](#) , [dvr_disp_disp_param_tag](#)
- color_mode : [dvr_disp_plane_param_tag](#) , [dvr_disp_update_plane_param_tag](#) , [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#)
- command : [dvr_disp_control_tag](#) , [dvr_dec_control_tag](#) , [dvr_enc_control_tag](#)
- contrast : [dvr_disp_color_attribute_tag](#)
- count : [QueueMemConfig_tag](#) , [dvr_enc_control_tag](#) , [dvr_graph_vqueueut_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- d -

- data_mode : [dvr_disp_plane_param_tag](#) , [dvr_disp_update_plane_param_tag](#)
- ddr : [dvr_graph_vqueue_tag](#)
- ddr_num : [QueueMemConfig_tag](#)
- dec_dest_type : [dvr_dec_channel_param_tag](#)
- dec_param : [dvr_dec_channel_param_tag](#)
- dec_type : [dvr_dec_channel_param_tag](#)
- denoise_mode : [video_process_tag](#)
- denominator : [dvr_rate_tag](#)
- des_level : [ScalerParamtag](#)
- di_mode : [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#) , [dvr_enc_update_channel_param_tag](#)
- dim : [dvr_disp_disp_param_tag](#) , [dvr_disp_control_tag](#) , [ReproduceBitStream_tag](#) , [dvr_disp_scaler_info_tag](#) , [dvr_enc_update_channel_param_tag](#) , [dvr_enc_src_tag](#) , [dvr_dec_control_tag](#)
- disp_num : [dvr_disp_plane_param_st_tag](#) , [dvr_disp_disp_param_tag](#) , [dvr_disp_update_disp_param_tag](#)
- display_rate : [dvr_dec_control_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#)
- dma_order : [dvr_enc_src_tag](#) , [dvr_disp_control_tag](#)
- dst_fmt : [ScalerParamtag](#)
- dst_param : [dvr_dec_control_tag](#) , [dvr_disp_control_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- e -

- en_snapshot : [ReproduceBitStream_tag](#)
- enabled : [ReproduceBitStream_tag](#)
- enc : [ReproduceBitStream_tag](#)
- enc_src_type : [dvr_enc_src_tag](#)
- enc_type : [EncParam_Ext5_tag](#) , [ReproduceBitStream_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext2_tag](#) , [EncParam_Ext3_tag](#)
- ext_size : [dvr_disp_control_tag](#) , [dvr_enc_update_channel_param_tag](#) , [EncParam_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- f -

- fb_offset : [dvr_disp_vbi_info_tag](#)
- fb_size : [dvr_disp_vbi_info_tag](#)
- feature_enable : [EncParam_Ext2_tag](#) , [EncParam_Ext3_tag](#) , [EncParam_Ext1_tag](#) , [EncParam_Ext5_tag](#) , [EncParam_Ext4_tag](#)
- filler : [dvr_disp_vbi_info_tag](#)
- frame_rate : [EncParam_tag](#) , [dvr_enc_update_channel_param_tag](#)
- func : [FuncTag_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- h -

- height : [DIM_tag](#) , [RECT_tag](#)
- huecos : [dvr_disp_color_attribute_tag](#)
- huesin : [dvr_disp_color_attribute_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	I	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- i -

- init_quant : [EncParam_tag](#) , [dvr_enc_update_channel_param_tag](#)
- input_res : [dvr_disp_resolution_tag](#)
- input_system : [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#)
- input_type : [EncParam_tag](#)
- ip_interval : [dvr_enc_update_channel_param_tag](#) , [EncParam_tag](#)
- is_3DI : [dvr_enc_update_channel_param_tag](#) , [video_process_tag](#)
- is_album : [ScalerParamtag](#)
- is_blocked : [dvr_dec_channel_param_tag](#) , [ReproduceBitStream_tag](#)
- is_correction : [ScalerParamtag](#)
- is_denoise : [video_process_tag](#) , [dvr_enc_update_channel_param_tag](#)
- is_display : [dvr_dec_control_tag](#)
- is_dither : [ScalerParamtag](#)
- is_enable : [dvr_disp_color_key_tag](#) , [dvr_disp_scaler_info_tag](#)
- is_keyframe : [dvr_enc_copy_buf_tag](#) , [dvr_bs_data_tag](#)
- is_use_ROI : [EncParam_tag](#) , [ROI_ALL_tag](#)
- is_use_scaler : [dvr_disp_control_tag](#) , [dvr_dec_channel_param_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	I	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- I -

- length : [dvr_bs_data_tag](#)
- limit_count : [QueueMemConfig_tag](#) , [dvr_graph_vqueueut_tag](#)
- lineheight : [dvr_disp_vbi_info_tag](#)
- lineno : [dvr_disp_vbi_info_tag](#)
- lv : [dvr_disp_control_tag](#)
- lv_ch : [FuncTag_tag](#)

Main Page	Data Structures	Files	Examples																
Data Structures	Data Fields																		
All	Variables																		
b	c	d	e	f	h	i	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- m -

- main_bs : [dvr_enc_channel_param_tag](#)
- max_quant : [EncParam_tag](#) , [dvr_enc_update_channel_param_tag](#)
- mb_len : [dvr_enc_queue_get_tag](#)
- min_quant : [EncParam_tag](#) , [dvr_enc_update_channel_param_tag](#)
- MJ_quality : [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#) , [EncParam_Ext2_tag](#)
- mode : [dvr_enc_update_channel_param_tag](#) , [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#)
- mv_buf_length : [dvr_enc_copy_buf_tag](#)
- mv_length : [dvr_enc_copy_buf_tag](#) , [dvr_bs_data_tag](#)
- mv_offset : [dvr_bs_data_tag](#)
- mv_user_va : [dvr_enc_copy_buf_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- n -

- new_bs : [dvr_enc_queue_get_tag](#) , [dvr_enc_copy_buf_tag](#)
- NonRef : [dvr_bs_data_tag](#) , [dvr_enc_copy_buf_tag](#)
- numerator : [dvr_rate_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- o -

- offset : [dvr_bs_data_tag](#)
- out_bs : [ReproduceBitStream_tag](#)
- output : [dvr_enc_control_tag](#)
- output_mode : [dvr_disp_disp_param_tag](#) ,
[dvr_disp_update_disp_param_tag::val_t](#)
- output_system : [dvr_disp_update_disp_param_tag::val_t](#) ,
[dvr_disp_disp_param_tag](#)
- output_type : [dvr_disp_resolution_tag](#) , [DecParam_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- p -

- p_job : [dvr_enc_copy_buf_tag](#) , [dvr_bs_data_tag](#)
- param : [dvr_disp_plane_param_st_tag](#) ,
[dvr_disp_update_plane_param_tag](#) ,
[dvr_disp_update_disp_param_tag](#)
- path : [dvr_disp_control_tag](#)
- pattern : [dvr_disp_clear_param_tag](#) ,
[dvr_dec_clear_param_tag](#)
- pb_ch : [FuncTag_tag](#)
- pext_data : [dvr_disp_control_tag](#) , [EncParam_tag](#) ,
[dvr_enc_update_channel_param_tag](#)
- plane_comb : [dvr_disp_disp_param_tag](#) ,
[dvr_disp_update_disp_param_tag::val_t](#)
- plane_id : [dvr_disp_control_tag](#) , [dvr_dec_control_tag](#) ,
[dvr_disp_update_plane_param_tag](#) ,
[dvr_disp_plane_param_tag](#) , [dvr_disp_clear_param_tag](#)
- plane_num : [dvr_disp_plane_param_st_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- q -

- quality : [snapshot_param_tag](#)
- que : [dvr_graph_vqueueut_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- r -

- rate : [dvr_disp_control_tag](#)
- reaction_delay_max : [EncParam_Ext1_tag](#) , [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext2_tag](#) , [EncParam_Ext5_tag](#)
- rec_ch : [FuncTag_tag](#)
- rect0 : [dvr_disp_control_tag](#)
- rect1 : [dvr_disp_control_tag](#)
- repd_bs_num : [dvr_enc_control_tag](#)
- res : [dvr_disp_disp_param_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#)
- reserved : [dvr_enc_control_tag](#) , [dvr_enc_update_channel_param_tag](#) , [dvr_bs_data_tag](#) , [dvr_dec_control_tag](#) , [snapshot_param_tag](#) , [EncParam_tag](#) , [dvr_disp_clear_param_tag](#) , [ReproduceBitStream_tag](#) , [dvr_disp_update_plane_param_tag](#) , [dvr_disp_plane_param_st_tag](#) , [dvr_enc_src_tag](#) , [dvr_dec_clear_param_tag](#) , [dvr_disp_disp_param_tag](#) , [dvr_disp_resolution_tag](#) , [dvr_disp_scaler_info_tag](#) , [ScalerParamtag](#) , [dvr_disp_color_attribute_tag](#) , [dvr_enc_copy_buf_tag](#) , [dvr_disp_color_key_tag](#) , [dvr_rate_tag](#) , [dvr_dec_queue_get_tag](#) , [dvr_disp_control_tag](#) , [DecParam_tag](#) , [dvr_disp_vbi_info_tag](#) , [video_process_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#) , [dvr_disp_plane_param_tag](#) , [dvr_dec_channel_param_tag](#)
- reserved1 : [dvr_disp_update_plane_param_tag](#)
- RestartInterval : [snapshot_param_tag](#)
- roi_all : [EncParam_Ext5_tag](#)

- roi_pos : **EncParam_Ext4_tag** , **EncParam_Ext5_tag**
 - ROI_win : **EncParam_tag**
-

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- s -

- sample : [snapshot_param_tag](#)
- saturation : [dvr_disp_color_attribute_tag](#)
- scale_indep : [dvr_enc_src_tag](#) , [dvr_enc_update_channel_param_tag](#) , [dvr_disp_control_tag](#)
- scale_mode : [ScalerParamtag](#)
- scl : [ReproduceBitStream_tag](#)
- scl_info : [dvr_disp_disp_param_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#)
- scl_param : [dvr_dec_channel_param_tag](#) , [dvr_disp_control_tag](#)
- sharprness_thres1 : [dvr_disp_color_attribute_tag](#)
- sharpness_thres0 : [dvr_disp_color_attribute_tag](#)
- sharpnessk0 : [dvr_disp_color_attribute_tag](#)
- sharpnessk1 : [dvr_disp_color_attribute_tag](#)
- size : [QueueMemConfig_tag](#) , [dvr_graph_vqueueut_tag](#)
- snap : [ReproduceBitStream_tag](#)
- src : [dvr_enc_update_channel_param_tag](#) , [dvr_enc_channel_param_tag](#)
- src_fmt : [ScalerParamtag](#)
- src_param : [dvr_disp_control_tag](#) , [dvr_dec_control_tag](#)
- stream : [dvr_enc_copy_buf_tag](#) , [dvr_enc_control_tag](#) , [dvr_bs_data_tag](#)
- stream_enable : [dvr_enc_update_channel_param_tag](#)
- swc_rect0 : [dvr_disp_control_tag](#)
- swc_rect1 : [dvr_disp_control_tag](#)

doxygen 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- t -

- target_id : [dvr_disp_disp_param_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#)
- target_rate_max : [EncParam_Ext2_tag](#) , [EncParam_Ext3_tag](#) , [EncParam_Ext1_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)
- timestamp : [dvr_enc_copy_buf_tag](#) , [dvr_bs_data_tag](#)
- transparent_color : [dvr_disp_update_disp_param_tag::val_t](#) , [dvr_disp_disp_param_tag](#)
- type : [dvr_disp_control_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- u -

- u82D : [snapshot_param_tag](#)
- update_parm : [dvr_enc_control_tag](#)
- user_tag : [dvr_graph_vqueue_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- v -

- val : [dvr_disp_update_disp_param_tag](#) , [dvr_disp_update_plane_param_tag](#)
- vbi_info : [dvr_disp_update_disp_param_tag::val_t](#) , [dvr_disp_disp_param_tag](#)
- vp_param : [dvr_enc_src_tag](#) , [dvr_disp_control_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- w -

- watermark_enable : [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)
- watermark_init_interval : [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)
- watermark_init_pattern : [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#) , [EncParam_Ext3_tag](#)
- watermark_interval : [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)
- width : [DIM_tag](#) , [RECT_tag](#)
- win : [ROI_ALL_tag](#) , [dvr_disp_control_tag](#) , [dvr_dec_control_tag](#) , [dvr_disp_plane_param_tag](#) , [dvr_dec_clear_param_tag](#) , [dvr_disp_update_plane_param_tag](#) , [dvr_disp_clear_param_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- x -

- x : [RECT_tag](#) , [POS_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Here is a list of all struct and union fields with links to the structures/unions they belong to:

- y -

- y : [RECT_tag](#) , [POS_tag](#)

Main Page

Data Structures

Files

Examples

File List

Globals

source/dvr_common_api.h

Go to the documentation of this file.

```
00001 #ifndef __DVR_COMMON_API_H__
00002 #define __DVR_COMMON_API_H__
00003
00004 #include <linux/ioctl.h>
00005 #include "dvr_type_define.h"
00006
00007 #define DVR_IOC_MAGIC 'D'
00008
00009 /**
00010  * \b ioctl(dvr_fd, DVR_COMMON_APPLY, &tag)
00011  *
00012  * \arg get function tag from user space, and set to videograph level
00013  * \arg parameter :
00014  * \n \b \e pointer \b \e tag : argument from user space ioctl parameter, it means structure FuncTag
00015 */
00016 #define DVR_COMMON_APPLY           _IOW
(DVR_IOC_MAGIC, 1, FuncTag)
00017 #define DVR_COMMON_DEBUG          _IOW
(DVR_IOC_MAGIC, 2, FuncTag)
00018 #define DVR_COMMON_DEBUG_WITH_PANIC _IOW
(DVR_IOC_MAGIC, 3, FuncTag)
00019
00020 #endif /* __DVR_COMMON_API_H__ */
```

Main Page

Data Structures

Files

Examples

File List

Globals

source/dvr_dec_ioctl.h

Go to the documentation of this file.

```
00001
00002 #define DVR_DEC_IOC_MAGIC    'H'
00003
00004 /**
00005  * \b ioctl(dec_fd, DVR_DEC_SET_CHANNEL_PARAM,
00006  *           &ch_param)
00007  * \arg explanation : get channel parameter
00008  * from user space ,and set parameter to device driver
00009  * \arg parameter :
00010  * \n \b \e pointer \b \e ch_param : argument
00011  * from user space ioctl parameter, it means structure dvr_dec_channel_param
00012 #define DVR_DEC_SET_CHANNEL_PARAM
00013 _IOWR(DVR_DEC_IOC_MAGIC, 2, dvr_dec_channel_param)
00014 /**
00015  * \b ioctl(dec_fd, DVR_DEC_GET_CHANNEL_PARAM,
00016  *           &ch_param)
00017  * \arg explanation : get channel parameter
00018  * from user space
00019  * \arg parameter :
00020  * \n \b \e pointer \b \e ch_param : argument
00021 #define DVR_DEC_GET_CHANNEL_PARAM
```

```
_IO(DVR_DEC_IOC_MAGIC, 3)
00022
00023 /**
00024 * \b ioctl(dec_fd, DVR_DEC_QUEUE_GET, &data)

00025 *
00026 * \arg explanation : Get buffer to user space. It includes the buffer length, offset.
00027 * \arg parameter :
00028 * \n \b \e pointer \b \e data : argument from user space ioctl parameter, it means structure dvr_dec_queue_get
00029 */
00030 #define DVR_DEC_QUEUE_GET
_IOWR(DVR_DEC_IOC_MAGIC, 5, dvr_dec_queue_get)
00031
00032 /**
00033 * \b ioctl(dec_fd, DVR_DEC_QUEUE_PUT, &data)

00034 *
00035 * \arg explanation : get buffer from user space, and release buffer at videograph layer
00036 * \arg parameter :
00037 * \n \b \e pointer \b \e data : argument from user space ioctl parameter, it means structure dvr_dec_queue_get
00038 */
00039 #define DVR_DEC_QUEUE_PUT
_IOWR(DVR_DEC_IOC_MAGIC, 6, dvr_dec_queue_get)
00040
00041 /**
00042 * \b ioctl(dec_fd, DVR_DEC_CONTROL, &dec_ctl)
00043 *
00044 * \arg explanation : get decode control command from user space, and set command to videograph layer
```

```
00045 * \arg parameter :  
00046 * \n \b \e pointer \b \e dec_ctrl : argument from user space ioctl parameter, it means structure dvr_dec_control  
00047 */  
00048 #define DVR_DEC_CONTROL  
_IOW(DVR_DEC_IOC_MAGIC, 7, dvr_dec_control)  
00049  
00050 /**  
00051 * \b ioctl(dec_fd, DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE, &dec_buf_size)  
00052 *  
00053 * \arg explanation : get output buffer size for dvr decode to user space.  
  
00054 * \arg parameter :  
00055 * \n \b \e pointer \b \e dec_buf_size : argument from user space ioctl parameter, it means request buffer size.  
00056 */  
00057 #define DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE  
_IOWR(DVR_DEC_IOC_MAGIC, 8, int)  
00058  
00059 /**  
00060 * \b ioctl(dec_fd, DVR_DEC_CLEAR_WIN, &dec_clear_param)  
00061 *  
00062 * \arg explanation : to prevent lcd unexpected image, when stop playback, then start another playback,  
00063 * \n use flow : ioctl(dvr_fd,DVR_COMMON_APP  
LY,&data); //stop current playback  
00064 * \n .....  
00065 * \n ioctl(dec_fd,DVR_DEC_CLEAR_  
WIN,&data); //must after stop, and before start a  
pply  
00066 * \n ioctl(dvr_fd,DVR_COMMON_APP
```

```

LY,&data); //start new playback
00067 * \arg parameter :
00068 * \n \b \e pointer \b \e dec_clear_param :
argument from user space ioctl parameter, it means
structure dvr_dec_clear_param.
00069 *
00070 */
00071 #define DVR_DEC_CLEAR_WIN
_IOW(DVR_DEC_IOC_MAGIC, 18, dvr_dec_clear_param)
00072
00073 /**
00074 * \b ioctl(dec_fd, DVR_DEC_CLEAR_WIN2, &dec
_clear_param)
00075 *
00076 * \arg explanation : to prevent lcd unexpec
ted image, when update current playback
00077 * \n use flow : dec_ctrl.command = DEC_UPD
ATE;
00078 * \n
          ioctl(dec_fd, DVR_DEC_CONT
ROL, &dec_ctrl);
00079 * \n
          ioctl(dec_fd, DVR_DEC_CLEA
R_WIN2, &data); // tell VG to clear buffer after a
pply
00080 * \n
          ioctl(dvr_fd, DVR_COMMON_A
PPPLY, &data);
00081 * \arg parameter :
00082 * \n \b \e pointer \b \e dec_clear_param :
argument from user space ioctl parameter, it means
structure dvr_dec_clear_param.
00083 *
00084 */
00085 #define DVR_DEC_CLEAR_WIN2
_IOW(DVR_DEC_IOC_MAGIC, 19, dvr_dec_clear_param)

```

[Main Page](#)[Data Structures](#)[Files](#)[Examples](#)[File List](#)[Globals](#)

source/dvr_disp_ioctl.h

Go to the documentation of this file.

```
00001
00002 #define DVR_DISP_IOC_MAGIC 'I'
00003
00004 /**
00005 * \b ioctl(disp_fd, DVR_DISP_INITIATE, 0)
00006 *
00007 * \arg explanation : not used
00008 *
00009 */
00010 #define DVR_DISP_INITIATE _IO(DVR_
DISP_IOC_MAGIC, 1)
00011
00012 /**
00013 * \b ioctl(disp_fd, DVR_DISP_TERMINATE, 0)
00014 *
00015 * \arg explanation : not used
00016 *
00017 */
00018 #define DVR_DISP_TERMINATE _IO(DVR_
DISP_IOC_MAGIC, 2)
00019
00020 /**
00021 * \b ioctl(disp_fd, DVR_DISP_GET_DISP_PARAM
, &disp_param)
00022 *
00023 * \arg explanation : get LCD color parameter from user space ,and set parameter to device driver
00024 * \arg parameter :
00025 * \n \b \e pointer \b \e disp_param : argument from user space ioctl parameter, it means structure dvr_disp_disp_param
```

```
00026  *
00027  */
00028 #define DVR_DISP_GET_DISP_PARAM      _IOWR(DV
R_DISP_IOC_MAGIC, 4, dvr_disp_disp_param)
00029
00030 /**
00031 * \b ioctl(disp_fd, DVR_DISP_SET_DISP_PARAM
, &disp_param)
00032 *
00033 * \arg explanation : get LCD color parameter from user space ,and set parameter to device driver
00034 * \arg parameter :
00035 * \n \b \e pointer \b \e disp_param : argument from user space ioctl parameter, it means structure dvr_disp_disp_param
00036 *
00037 */
00038 #define DVR_DISP_SET_DISP_PARAM      _IOWR(DV
R_DISP_IOC_MAGIC, 5, dvr_disp_disp_param)
00039
00040 /**
00041 * \b ioctl(disp_fd, DVR_DISP_UPDATE_DISP_PA
RAM, &disp_update_param)
00042 *
00043 * \arg explanation : get LCD color parameter from user space ,and update parameter to device driver
00044 * \arg parameter :
00045 * \n \b \e pointer \b \e disp_update_param : argument from user space ioctl parameter, it means structure dvr_disp_update_disp_param
00046 *
00047 */
00048 #define DVR_DISP_UPDATE_DISP_PARAM _IOWR(DV
R_DISP_IOC_MAGIC, 6, dvr_disp_update_disp_param)
00049
```

```
00050 /**
00051  * \b ioctl(disp_fd, DVR_DISP_GET_PLANE_PARA
00052  * M, &plane_param)
00053  * \arg explanation : get LCD plane(window)
00054  * parameter from user space, and set parameter to de
00055  * vice driver
00056  * \arg parameter :
00057  * \n \b \e pointer \b \e plane_param : argu
00058  * ment from user space ioctl parameter, it means str
00059  * ucture dvr_disp_plane_param
00060 /**
00061  * \b ioctl(disp_fd, DVR_DISP_SET_PLANE_PARA
00062  * M, &plane_param_set)
00063  * \arg explanation : get LCD plane(window)
00064  * parameter from user space, and set parameter to de
00065  * vice driver
00066  * \arg parameter :
00067  * \n \b \e pointer \b \e plane_param_set : 
00068  * argument from user space ioctl parameter, it means
00069  * structure dvr_disp_plane_param
00070 /**
00071  * \b ioctl(disp_fd, DVR_DISP_UPDATE_PLANE_P
00072  * ARAM, &plane_param_update)
00073  * \arg explanation : get LCD plane(window)
```

```
parameter from user space, and update parameter to
device driver
00074 * \arg parameter :
00075 * \n \b \e pointer \b \e plane_param_update
: argument from user space ioctl parameter, it me
ans structure dvr_disp_update_plane_param
00076 *
00077 */
00078 #define DVR_DISP_UPDATE_PLANE_PARAM _IOWR(DV
R_DISP_IOC_MAGIC, 9, dvr_disp_update_plane_param)
00079
00080 /**
00081 * \b ioctl(disp_fd, DVR_DISP_CONTROL, &disp
_ctrl)
00082 *
00083 * \arg explanation : get display control co
mmand from user space, and set command to videogra
ph layer
00084 * \arg parameter :
00085 * \n \b \e pointer \b \e disp_ctrl : argume
nt from user space ioctl parameter, it means struc
ture dvr_disp_control
00086 *
00087 */
00088 #define DVR_DISP_CONTROL _IOR(DVR
_DISP_IOC_MAGIC, 10, dvr_disp_control)
00089
00090 /**
00091 * \b ioctl(disp_fd, DVR_DISP_INITIATE, &dis
p_clear_param)
00092 *
00093 * \arg explanation : get channel parameter
from user space ,and clear lcd buffer
00094 * \arg parameter :
00095 * \n \b \e pointer \b \e disp_clear_param;
: argument from user space ioctl parameter, it mea
ns structure dvr_disp_clear_param
```

```
00096  *
00097  */
00098 #define DVR_DISP_CLEAR_WIN _IOWR(DV
R_DISP_IOC_MAGIC, 11, dvr_disp_clear_param)
```

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page

Data Structures

Files

Examples

File List

Globals

source/dvr_enc_ioctl.h

Go to the documentation of this file.

```
00001 /*! \mainpage GM8126 dvr encode, decode and  
00002   display ioctl functions  
00003   *  
00004   * dvr_enc_ioctl.h describe dvr encode ioctl  
00005   functions behavior  
00006   * \n dvr_dec_ioctl.h describe dvr decode io  
00007   ctl functions behavior  
00008   * \n dvr_disp_ioctl.h describe dvr display  
00009   ioctl functions behavior  
00010  *  
00011  * \example main-bitstream-record.c  
00012  * \example sub-bitstream-record.c  
00013  * \example update-record-setting.c  
00014  * \example mpeg4-record.c  
00015  * \example mjpeg-record.c  
00016  * \example snapshot.c  
00017  * \example motion-detection.c  
00018  * \example update-bitrate.c  
00019  * \example playback.c  
00020  * \example roi.c  
00021  * \example capture_raw.c  
00022  * \example liveview.c  
00023  * \example 2ch_liveview.c  
00024  * \example 2ch_playback.c  
00025  * \example motion-detection-mpeg4.c  
00026  *  
00027 */
```

```
00028 #define DVR_ENC_IOC_MAGIC 'B'
00029
00030 /**
00031 * \b ioctl(enc_fd, DVR_ENC_SET_CHANNEL PARA
M, &ch_param)
00032 *
00033 * \arg explanation : get channel parameter
from user space ,and set parameter to device driver
00034 * \arg parameter :
00035 * \n \b \e pointer \b \e ch_param : argument
t from user space ioctl parameter, it means struct
ure dvr_enc_channel_param
00036 *
00037 */
00038 #define DVR_ENC_SET_CHANNEL_PARAM
        _IOWR(DVR_ENC_IOC_MAGIC, 2, dvr_enc_channe
l_param)
00039
00040 /**
00041 * \b ioctl(enc_fd, DVR_ENC_GET_CHANNEL PARA
M, &ch_param) :
00042 *
00043 * \arg explanation : get channel parameter
from user space
00044 * \arg parameter :
00045 * \n \b \e pointer \b \e ch_param : argument
t from user space ioctl parameter, it means struct
ure dvr_enc_channel_param
00046 */
00047 #define DVR_ENC_GET_CHANNEL_PARAM
        _IOWR(DVR_ENC_IOC_MAGIC, 3, dvr_enc_channe
l_param)
00048
00049 /**
00050 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET, &data)
```

```
00051  *
00052  * \arg explanation : Get buffer to user space. It includes the buffer length, offset.
00053  * \arg parameter :
00054  * \n \b \e pointer \b \e data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get
00055  */
00056 #define DVR_ENC_QUEUE_GET
        _IOWR(DVR_ENC_IOC_MAGIC, 5, dvr_enc_queue_
get)
00057
00058 /**
00059  * \b ioctl(enc_fd, DVR_ENC_QUEUE_PUT, &data)

00060  *
00061  * \arg explanation : get buffer from user space, and release buffer at videograph layer
00062  * \arg parameter :
00063  * \n \b \e pointer \b \e data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get
00064  */
00065 #define DVR_ENC_QUEUE_PUT
        _IOWR(DVR_ENC_IOC_MAGIC, 6, dvr_enc_queue_
get)
00066
00067 /**
00068  * \b ioctl(enc_fd, DVR_ENC_CONTROL, &enc_ct
rl)
00069  *
00070  * \arg explanation : get encode contorl com
mand from user space, and set command to videograp
h layer
00071  * \arg parameter :
00072  * \n \b \e pointer \b \e enc_ctrl : argume
nt from user space ioctl parameter, it means struc
```

```
ture dvr_enc_control
00073 */
00074 #define DVR_ENC_CONTROL
        _IOW(DVR_ENC_IOC_MAGIC, 7, dvr_enc_control)

00075
00076 /**
00077 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE, &enc_buf_size)
00078 *
00079 * \arg explanation : get output buffer size
00080 *                      buffer size = main bit
00081 *                      stream + sub1 bitstream + sub2 bitstream + snapshot
00082 * \arg parameter :
00083 * \n \b \e pointer \b \e enc_buf_size : argument from user space ioctl parameter, it means request buffer size.
00084 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE
        _IOWR(DVR_ENC_IOC_MAGIC, 8, int)
00085
00086 /**
00087 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE, &enc_buf_size);
00088 *
00089 * \arg explanation : get snapshot buffer to user space
00090 * \arg parameter :
00091 * \n \b \e pointer \b \e queue_data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get
00092 */
00093 #define DVR_ENC_QUEUE_GET_SNAP
        _IOWR(DVR_ENC_IOC_MAGIC, 9, dvr_enc_queue_get)
```

```
00094
00095 /**
00096  * ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER
00097 _SNAP_OFFSET, &bs_buf_snap_offset)
00097 *
00098  * \arg explanation : get output buffer offset for snapshot to user space
00099  * \arg parameter :
00100  * \n \b \e pointer \b \e bs_buf_snap_offset
00100  : argument from user space ioctl parameter, it means snapshot offset.
00101 *
00102 */
00103 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP_OFFSET
00103 _IOR(DVR_ENC_IOC_MAGIC, 10, int)
00104
00105 /**
00106  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB1_BS, &data) :
00107 *
00108  * \arg explanation : get sub1-bitstream buffer to user space.
00109  * \arg parameter :
00110  * \n \b \e pointer \b \e data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get
00111 */
00112 #define DVR_ENC_QUEUE_GET_SUB1_BS
00112 _IOWR(DVR_ENC_IOC_MAGIC, 11, dvr_enc_queue
00113 _get)
00113
00114 /**
00115  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB2_BS, &data)
00116 *
00117  * \arg explanation : get sub2-bitstream buffer to user space.
```

```
00118 * \arg parameter :
00119 * \n \b \e pointer \b \e data : argument fr
om user space ioctl parameter, it means structure
dvr_enc_queue_get
00120 */
00121 #define DVR_ENC_QUEUE_GET_SUB2_BS
    _IOWR(DVR_ENC_IOC_MAGIC, 12, dvr_enc_queue
_get)
00122
00123 /**
00124 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BU
FER_SUB1_BS_OFFSET, &sub1_bs_buf_offset)
00125 *
00126 * \arg explanation : get output buffer offs
et for Sub1-bitstream to user space
00127 * \arg parameter :
00128 * \n \b \e pointer \b \e sub1_bs_buf_offset
: argument from user space ioctl parameter, it me
ans sub1 bitstream buffer offset.
00129 *
00130 */
00131 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1_BS_
OFFSET _IOR(DVR_ENC_IOC_MAGIC, 13, int)
00132
00133 /**
00134 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BU
FER_SUB2_BS_OFFSET, &sub2_bs_buf_offset)
00135 *
00136 * \arg explanation : get output buffer offs
et for Sub2-bitstream to user space
00137 * \arg parameter :
00138 * \n \b \e pointer \b \e sub2_bs_buf_offset
: argument from user space ioctl parameter, it me
ans sub2 bitstream buffer offset.
00139 *
00140 */
00141 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB2_BS_
```

```
OFFSET _IOR(DVR_ENC_IOC_MAGIC, 14, int)
00142 /**
00143  * \b ioctl(enc_fd, DVR_ENC_RESET_INTRA, &stream_num)
00144  *
00145  * \arg explanation : get i-frame as possible
00146
00147  * \arg parameter :
00148  * \n \b \e pointer \b \e stream_num : argument from user space ioctl parameter, it means structure main-bitstream or sub1-bitstream or sub2-bitstream
00149 */
00150 #define DVR_ENC_RESET_INTRA
        _IOR(DVR_ENC_IOC_MAGIC, 15, int)
00151
00152 /**
00153  * \b ioctl(enc_fd, DVR_ENC_SET_SUB_BS_PARAM
00154  , &sub_bitstream)
00155  *
00156  * \arg explanation : get sub-bitstream parameter from user space, and set sub-bitstream parameter to device driver
00157  * \n \b \e pointer \b \e sub_bitstream : argument from user space ioctl parameter, it means structure ReproduceBitStream
00158 */
00159 #define DVR_ENC_SET_SUB_BS_PARAM
        _IOW(DVR_ENC_IOC_MAGIC, 16, int)
00160
00161 /**
00162  * \b ioctl(enc_fd, DVR_ENC_SWAP_INTRA, &stream_num)
00163  *
00164  * \arg explanation : get i-frame definitely
```

```
when next switch
00165 * \arg parameter :
00166 * \n \b \e pointer \b \e stream_num : argument from user space ioctl parameter, it means structure main-bitstream or sub1-bitstream or sub2-bitstream
00167 */
00168 #define DVR_ENC_SWAP_INTRA
        _IOR(DVR_ENC_IOC_MAGIC, 18, int)
00169
00170 /**
00171 * \b ioctl(enc_fd, DVR_ENC_SUB_PATH_DENOISE_CTRL, &control)
00172 *
00173 * \arg explanation : get denoise option from user space, and set to device driver
00174 * \arg parameter :
00175 * \n \b \e pointer \b \e control : argument from user space ioctl parameter, it means denoise on/off
00176 */
00177 #define DVR_ENC_SUB_PATH_DENOISE_CTRL
        _IOW(DVR_ENC_IOC_MAGIC, 19, int)
00178
00179 /**
00180 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB3_BS, &data)
00181 *
00182 * \arg explanation : get sub3-bitstream buffer to user space.
00183 * \arg parameter :
00184 * \n \b \e pointer \b \e data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get
00185 */
00186 #define DVR_ENC_QUEUE_GET_SUB3_BS
        _IOWR(DVR_ENC_IOC_MAGIC, 20, dvr_enc_queue
```

```
_get)
00187
00188 /**
00189  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB4_BS,
00190  *           &data)
00191  * \arg explanation : get sub4-bitstream buffer to user space.
00192  * \arg parameter :
00193  * \n \b \e pointer \b \e data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get
00194 */
00195 #define DVR_ENC_QUEUE_GET_SUB4_BS
00196         _IOWR(DVR_ENC_IOC_MAGIC, 21, dvr_enc_queue
00197 _get)
00198
00199 /**
00200  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB5_BS,
00201  *           &data)
00202  * \arg explanation : get sub5-bitstream buffer to user space.
00203  * \arg parameter :
00204  * \n \b \e pointer \b \e data : argument from user space ioctl parameter, it means structure dvr_enc_queue_get
00205 */
00206 /**
00207  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB6_BS,
00208  *           &data)
00209  * \arg explanation : get sub6-bitstream buf
```

```
fer to user space.  
00210 * \arg parameter :  
00211 * \n \b \e pointer \b \e data : argument fr  
om user space ioctl parameter, it means structure  
dvr_enc_queue_get  
00212 */  
00213 #define DVR_ENC_QUEUE_GET_SUB6_BS  
        _IOWR(DVR_ENC_IOC_MAGIC, 23, dvr_enc_queue  
_get)  
00214  
00215 /**  
00216 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB7_B  
S, &data)  
00217 *  
00218 * \arg explanation : get sub7-bitstream buf  
fer to user space.  
00219 * \arg parameter :  
00220 * \n \b \e pointer \b \e data : argument fr  
om user space ioctl parameter, it means structure  
dvr_enc_queue_get  
00221 */  
00222 #define DVR_ENC_QUEUE_GET_SUB7_BS  
        _IOWR(DVR_ENC_IOC_MAGIC, 24, dvr_enc_queue  
_get)  
00223  
00224 /**  
00225 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB8_B  
S, &data)  
00226 *  
00227 * \arg explanation : get sub8-bitstream buf  
fer to user space.  
00228 * \arg parameter :  
00229 * \n \b \e pointer \b \e data : argument fr  
om user space ioctl parameter, it means structure  
dvr_enc_queue_get  
00230 */  
00231 #define DVR_ENC_QUEUE_GET_SUB8_BS
```

```
        _IOWR(DVR_ENC_IOC_MAGIC, 25, dvr_enc_queue
_>get)
00232
00233 /**
00234 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB3_BS_OFFSET, &sub3_bs_buf_offset)
00235 *
00236 * \arg explanation : get output buffer offset for Sub3-bitstream to user space
00237 * \arg parameter :
00238 * \n \b \e pointer \b \e sub3_bs_buf_offset
00239 : argument from user space ioctl parameter, it means sub3 bitstream buffer offset.
00240 *
00241 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB3_BS_
00242 OFFSET _IOR(DVR_ENC_IOC_MAGIC, 26, int)
00243 /**
00244 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB4_BS_OFFSET, &sub4_bs_buf_offset)
00245 *
00246 * \arg explanation : get output buffer offset for Sub4-bitstream to user space
00247 * \arg parameter :
00248 * \n \b \e pointer \b \e sub4_bs_buf_offset
00249 : argument from user space ioctl parameter, it means sub4 bitstream buffer offset.
00250 *
00251 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB4_BS_
00252 OFFSET _IOR(DVR_ENC_IOC_MAGIC, 27, int)
00253 /**
00254 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB5_BS_OFFSET, &sub5_bs_buf_offset)
00255 *
```

```
00256 * \arg explanation : get output buffer offset for Sub5-bitstream to user space
00257 * \arg parameter :
00258 * \n \b \e pointer \b \e sub5_bs_buf_offset
  : argument from user space ioctl parameter, it means sub5 bitstream buffer offset.
00259 *
00260 */
00261 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB5_BS_
OFFSET _IOR(DVR_ENC_IOC_MAGIC, 28, int)
00262
00263 /**
00264 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB6_BS_OFFSET, &sub6_bs_buf_offset)
00265 *
00266 * \arg explanation : get output buffer offset for Sub6-bitstream to user space
00267 * \arg parameter :
00268 * \n \b \e pointer \b \e sub6_bs_buf_offset
  : argument from user space ioctl parameter, it means sub6 bitstream buffer offset.
00269 *
00270 */
00271 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB6_BS_
OFFSET _IOR(DVR_ENC_IOC_MAGIC, 29, int)
00272
00273 /**
00274 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BUFFER_SUB7_BS_OFFSET, &sub7_bs_buf_offset)
00275 *
00276 * \arg explanation : get output buffer offset for Sub7-bitstream to user space
00277 * \arg parameter :
00278 * \n \b \e pointer \b \e sub7_bs_buf_offset
  : argument from user space ioctl parameter, it means sub7 bitstream buffer offset.
00279 *
```

```
00280  */
00281 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB7_BS_
OFFSET _IOR(DVR_ENC_IOC_MAGIC, 30, int)
00282
00283 /**
00284 * \b ioctl(enc_fd, DVR_ENC_QUERY_OUTPUT_BU
FER_SUB8_BS_OFFSET, &sub8_bs_buf_offset)
00285 *
00286 * \arg explanation : get output buffer offs
et for Sub8-bitstream to user space
00287 * \arg parameter :
00288 * \n \b \e pointer \b \e sub8_bs_buf_offset
: argument from user space ioctl parameter, it me
ans sub8 bitstream buffer offset.
00289 *
00290 */
00291 #define DVR_ENC_QUERY_OUTPUT_BUFFER_SUB8_BS_
OFFSET _IOR(DVR_ENC_IOC_MAGIC, 31, int)
00292
00293 /**
00294 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_COPY,
&data)
00295 *
00296 * \arg explanation : Get buffer and copy to
user space. It includes the bitstream length.
00297 * \arg parameter :
00298 * \n \b \e pointer \b \e data : argument fr
om user space ioctl parameter, it means structure
dvr_enc_copy_buf
00299 */
00300 #define DVR_ENC_QUEUE_GET_COPY
        _IOWR(DVR_ENC_IOC_MAGIC, 35, dvr_enc_copy_
buf)
00301
00302 /**
00303 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB1_B
S_COPY, &data)
```

```
00304  *
00305  * \arg explanation : get and copy sub8-bits
00306  * \arg parameter :
00307  * \n \b \e pointer \b \e data : argument fr
00308  * om user space ioctl parameter, it means structure
00309  * dvr_enc_copy_buf
00310  */
00311 #define DVR_ENC_QUEUE_GET_SUB1_BS_COPY
00312     _IOWR(DVR_ENC_IOC_MAGIC, 36, dvr_enc_copy_
00313 buf)
00314 /**
00315  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB2_B
00316  * S_COPY, &data)
00317  *
00318  * \arg explanation : get and copy sub8-bits
00319  * \arg parameter :
00320  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB3_B
00321  * S_COPY, &data)
00322  *
00323  * \arg explanation : get and copy sub8-bits
00324  * \arg parameter :
00325  * \n \b \e pointer \b \e data : argument fr
00326  * om user space ioctl parameter, it means structure
00327  * dvr_enc_copy_buf
```

```
00326  */
00327 #define DVR_ENC_QUEUE_GET_SUB3_BS_COPY
        _IOWR(DVR_ENC_IOC_MAGIC, 38, dvr_enc_copy_
buf)
00328
00329 /**
00330 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB4_B
S_COPY, &data)
00331 *
00332 * \arg explanation : get and copy sub8-bits
tream to user space buffer.
00333 * \arg parameter :
00334 * \n \b \e pointer \b \e data : argument fr
om user space ioctl parameter, it means structure
dvr_enc_copy_buf
00335 */
00336 #define DVR_ENC_QUEUE_GET_SUB4_BS_COPY
        _IOWR(DVR_ENC_IOC_MAGIC, 39, dvr_enc_copy_
buf)
00337
00338 /**
00339 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB5_B
S_COPY, &data)
00340 *
00341 * \arg explanation : get and copy sub8-bits
tream to user space buffer.
00342 * \arg parameter :
00343 * \n \b \e pointer \b \e data : argument fr
om user space ioctl parameter, it means structure
dvr_enc_copy_buf
00344 */
00345 #define DVR_ENC_QUEUE_GET_SUB5_BS_COPY
        _IOWR(DVR_ENC_IOC_MAGIC, 40, dvr_enc_copy_
buf)
00346
00347 /**
00348 * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB6_B
```

```
S_COPY, &data)
00349  *
00350  * \arg explanation : get and copy sub8-bits
tream to user space buffer.
00351  * \arg parameter :
00352  * \n \b \e pointer \b \e data : argument fr
om user space ioctl parameter, it means structure
dvr_enc_copy_buf
00353  */
00354 #define DVR_ENC_QUEUE_GET_SUB6_BS_COPY
        _IOWR(DVR_ENC_IOC_MAGIC, 41, dvr_enc_copy_
buf)
00355 /**
00356  */
00357  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB7_B
S_COPY, &data)
00358  *
00359  * \arg explanation : get and copy sub8-bits
tream to user space buffer.
00360  * \arg parameter :
00361  * \n \b \e pointer \b \e data : argument fr
om user space ioctl parameter, it means structure
dvr_enc_copy_buf
00362  */
00363 #define DVR_ENC_QUEUE_GET_SUB7_BS_COPY
        _IOWR(DVR_ENC_IOC_MAGIC, 42, dvr_enc_copy_
buf)
00364
00365
00366 /**
00367  * \b ioctl(enc_fd, DVR_ENC_QUEUE_GET_SUB8_B
S_COPY, &data)
00368  *
00369  * \arg explanation : get and copy sub8-bits
tream to user space buffer.
00370  * \arg parameter :
00371  * \n \b \e pointer \b \e data : argument fr
```

```
om user space ioctl parameter, it means structure
dvr_enc_copy_buf
00372 */
00373 #define DVR_ENC_QUEUE_GET_SUB8_BS_COPY
        _IOWR(DVR_ENC_IOC_MAGIC, 43, dvr_enc_copy_
buf)
00374
00375
00376
```



Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
c d e f g j l p q r s v				

- C -

- CaptureColorMode : [dvr_type_define.h](#)
- CapturePath : [dvr_type_define.h](#)

- d -

- DecoderOutputColor : [dvr_type_define.h](#)
- DecParam : [dvr_dec_api.h](#)
- DIM : [dvr_type_define.h](#)
- DispParam_Ext1 : [dvr_disp_api.h](#)
- dvr_bs_data : [dvr_type_define.h](#)
- dvr_dec_channel_param : [dvr_dec_api.h](#)
- dvr_dec_clear_param : [dvr_dec_api.h](#)
- dvr_dec_control : [dvr_dec_api.h](#)
- dvr_dec_ctrl_cmd : [dvr_dec_api.h](#)
- dvr_dec_dest_type : [dvr_dec_api.h](#)
- dvr_dec_queue_get : [dvr_dec_api.h](#)
- dvr_disp_channel_type : [dvr_disp_api.h](#)
- dvr_disp_clear_param : [dvr_disp_api.h](#)
- dvr_disp_color_attribute : [dvr_disp_api.h](#)
- dvr_disp_color_key : [dvr_disp_api.h](#)
- dvr_disp_control : [dvr_disp_api.h](#)
- dvr_disp_ctrl_cmd : [dvr_disp_api.h](#)
- dvr_disp_disp_param : [dvr_disp_api.h](#)
- dvr_disp_disp_param_name : [dvr_disp_api.h](#)
- dvr_disp_plane_combination : [dvr_disp_api.h](#)
- dvr_disp_plane_param : [dvr_disp_api.h](#)
- dvr_disp_plane_param_name : [dvr_disp_api.h](#)
- dvr_disp_plane_param_st : [dvr_disp_api.h](#)

- dvr_disp_resolution : **dvr_disp_api.h**
- dvr_disp_scaler_info : **dvr_disp_api.h**
- dvr_disp_update_disp_param : **dvr_disp_api.h**
- dvr_disp_update_plane_param : **dvr_disp_api.h**
- dvr_disp_vbi_info : **dvr_disp_api.h**
- dvr_enc_channel_param : **dvr_enc_api.h**
- dvr_enc_channel_param_name : **dvr_enc_api.h**
- dvr_enc_control : **dvr_enc_api.h**
- dvr_enc_copy_buf : **dvr_enc_api.h**
- dvr_enc_queue_get : **dvr_enc_api.h**
- dvr_enc_src_param : **dvr_enc_api.h**
- dvr_enc_src_type : **dvr_enc_api.h**
- dvr_enc_update_channel_param : **dvr_enc_api.h**
- dvr_graph_vqueueut : **dvr_type_define.h**
- dvr_ratio : **dvr_type_define.h**

- e -

- EncoderInputFormat : **dvr_type_define.h**
- EncodeType : **dvr_type_define.h**
- EncParam : **dvr_enc_api.h**
- EncParam_Ext1 : **dvr_enc_api.h**
- EncParam_Ext2 : **dvr_enc_api.h**
- EncParam_Ext3 : **dvr_enc_api.h**
- EncParam_Ext4 : **dvr_enc_api.h**
- EncParam_Ext5 : **dvr_enc_api.h**

- f -

- FuncTag : **dvr_type_define.h**

- g -

- GM3DIFrameType : **dvr_type_define.h**

- j -

- JpegEnc420InputFormat : **dvr_type_define.h**
- JpegEncInputFormat : **dvr_type_define.h**

- l -

- LCDOutputColorType : **dvr_type_define.h**
- LCDOutputMode : **dvr_type_define.h**
- LCDResolution : **dvr_type_define.h**
- LiveviewDMAOrder : **dvr_type_define.h**
- LiveviewFrameMode : **dvr_type_define.h**
- LiveviewFrameType : **dvr_type_define.h**
- LiveviewScalerRatio : **dvr_type_define.h**

- p -

- POS : **dvr_type_define.h**

- q -

- QueMemCfg : **dvr_type_define.h**
- QueueID : **dvr_type_define.h**

- r -

- RECT : **dvr_type_define.h**
- ReproduceBitStream : **dvr_enc_api.h**
- ROI_ALL : **dvr_type_define.h**

- s -

- ScaleColorMode : **dvr_type_define.h**
- ScaleMethod : **dvr_type_define.h**
- ScalerParam : **dvr_type_define.h**
- snapshot_param : **dvr_enc_api.h**

- v -

- video_process : **dvr_type_define.h**

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines

- CaptureColorModeTag : [dvr_type_define.h](#)
- CapturePathTag : [dvr_type_define.h](#)
- DecoderOutputColorTag : [dvr_type_define.h](#)
- dvr_dec_ctrl_cmd_tag : [dvr_dec_api.h](#)
- dvr_dec_dest_type_tag : [dvr_dec_api.h](#)
- dvr_disp_channel_type_tag : [dvr_disp_api.h](#)
- dvr_disp_ctrl_cmd_tag : [dvr_disp_api.h](#)
- dvr_disp_disp_param_name_tag : [dvr_disp_api.h](#)
- dvr_disp_plane_combination_tag : [dvr_disp_api.h](#)
- dvr_disp_plane_param_name_tag : [dvr_disp_api.h](#)
- dvr_disp_type : [dvr_disp_api.h](#)
- dvr_enc_channel_param_name_tag : [dvr_enc_api.h](#)
- dvr_enc_ctrl_cmd : [dvr_enc_api.h](#)
- dvr_enc_src_type_tag : [dvr_enc_api.h](#)
- EncoderInputFormatTag : [dvr_type_define.h](#)
- EncodeType_tag : [dvr_type_define.h](#)
- GM3DIFrameTypeTag : [dvr_type_define.h](#)
- JpegEnc420InputFormatTag : [dvr_type_define.h](#)
- JpegEnInputFormatTag : [dvr_type_define.h](#)
- LCDOutputColorTypeTag : [dvr_type_define.h](#)
- LCDOutputModeTag : [dvr_type_define.h](#)
- LCDResolutionTag : [dvr_type_define.h](#)
- LiveviewDMAOrderTag : [dvr_type_define.h](#)
- LiveviewFrameModeTag : [dvr_type_define.h](#)
- LiveviewFrameTypeTag : [dvr_type_define.h](#)
- LiveviewScalerRatioTag : [dvr_type_define.h](#)
- QueueID_tag : [dvr_type_define.h](#)
- ScaleColorModeTag : [dvr_type_define.h](#)
- ScaleMethodTag : [dvr_type_define.h](#)

doxygen 1.7.1

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
b c d e g j l p q s				

- b -

- BG_AND_1PLANE : [dvr_disp_api.h](#)
- BG_AND_2PLANE : [dvr_disp_api.h](#)
- BG_ONLY : [dvr_disp_api.h](#)

- c -

- CAPCOLOR_RGB565 : [dvr_type_define.h](#)
- CAPCOLOR_RGB888 : [dvr_type_define.h](#)
- CAPCOLOR_YUV420_M0 : [dvr_type_define.h](#)
- CAPCOLOR_YUV420_M1 : [dvr_type_define.h](#)
- CAPCOLOR_YUV422 : [dvr_type_define.h](#)
- CAPPATH_DEFAULT : [dvr_type_define.h](#)
- CAPPATH_PATH_1 : [dvr_type_define.h](#)
- CAPPATH_PATH_2 : [dvr_type_define.h](#)
- CAPSCALER_KEEP_RATIO : [dvr_type_define.h](#)
- CAPSCALER_NOT_KEEP_RATIO : [dvr_type_define.h](#)

- d -

- DEC_OUTPUT_COLOR_YUV420 : [dvr_type_define.h](#)
- DEC_OUTPUT_COLOR_YUV422 : [dvr_type_define.h](#)
- DEC_START : [dvr_dec_api.h](#)
- DEC_STOP : [dvr_dec_api.h](#)
- DEC_TYPE_COUNT : [dvr_dec_api.h](#)
- DEC_TYPE_TO_BUFFER : [dvr_dec_api.h](#)
- DEC_TYPE_TO_DISPLAY : [dvr_dec_api.h](#)
- DEC_UPDATE : [dvr_dec_api.h](#)
- DISP_LAYER0_CHN : [dvr_disp_api.h](#)
- DISP_LAYER1_CHN : [dvr_disp_api.h](#)

- DISP_NORMAL_CHN : [dvr_disp_api.h](#)
- DISP_PARAM_APPLY : [dvr_disp_api.h](#)
- DISP_PARAM_COLOR_ATTRIBUTE : [dvr_disp_api.h](#)
- DISP_PARAM_OUTPUT_MODE : [dvr_disp_api.h](#)
- DISP_PARAM_OUTPUT_SYSTEM : [dvr_disp_api.h](#)
- DISP_PARAM_PLANE_COMBINATION : [dvr_disp_api.h](#)
- DISP_PARAM_RESOLUTION : [dvr_disp_api.h](#)
- DISP_PARAM_TARGET : [dvr_disp_api.h](#)
- DISP_PARAM_TRANSPARENT_COLOR : [dvr_disp_api.h](#)
- DISP_RUN : [dvr_disp_api.h](#)
- DISP_START : [dvr_disp_api.h](#)
- DISP_STOP : [dvr_disp_api.h](#)
- DISP_TYPE CASCADE : [dvr_disp_api.h](#)
- DISP_TYPE_LIVEVIEW : [dvr_disp_api.h](#)
- DISP_TYPE_ON_BUFFER : [dvr_disp_api.h](#)
- DISP_TYPE_PLAYBACK : [dvr_disp_api.h](#)
- DISP_UPDATE : [dvr_disp_api.h](#)
- DMAORDER_2PLANAR : [dvr_type_define.h](#)
- DMAORDER_3PLANAR : [dvr_type_define.h](#)
- DMAORDER_PACKET : [dvr_type_define.h](#)

- e -

- ENC_INPUT_1D420 : [dvr_type_define.h](#)
- ENC_INPUT_1D422 : [dvr_type_define.h](#)
- ENC_INPUT_H2642D : [dvr_type_define.h](#)
- ENC_INPUT_MP42D : [dvr_type_define.h](#)
- ENC_PARAM_DST_WIN : [dvr_enc_api.h](#)
- ENC_PARAM_SRC_DIM : [dvr_enc_api.h](#)
- ENC_RAW : [dvr_enc_api.h](#)
- ENC_SNAP : [dvr_enc_api.h](#)
- ENC_SRC_TYPE_COUNT : [dvr_enc_api.h](#)
- ENC_START : [dvr_enc_api.h](#)
- ENC_STOP : [dvr_enc_api.h](#)
- ENC_TYPE_COUNT : [dvr_type_define.h](#)
- ENC_TYPE_FROM_BUFFER : [dvr_enc_api.h](#)
- ENC_TYPE_FROM_CAPTURE : [dvr_enc_api.h](#)
- ENC_TYPE_FROM_CASCADE : [dvr_enc_api.h](#)
- ENC_TYPE_H264 : [dvr_type_define.h](#)

- ENC_TYPE_MJPEG : [dvr_type_define.h](#)
- ENC_TYPE_MPEG : [dvr_type_define.h](#)
- ENC_TYPE_YUV422 : [dvr_type_define.h](#)
- ENC_UPDATE : [dvr_enc_api.h](#)

- g -

- GM3DI_FIELD : [dvr_type_define.h](#)
- GM3DI_FRAME : [dvr_type_define.h](#)

- j -

- JCS_yuv111 : [dvr_type_define.h](#)
- JCS_yuv211 : [dvr_type_define.h](#)
- JCS_yuv222 : [dvr_type_define.h](#)
- JCS_yuv333 : [dvr_type_define.h](#)
- JCS_yuv400 : [dvr_type_define.h](#)
- JCS_yuv420 : [dvr_type_define.h](#)
- JCS_yuv422 : [dvr_type_define.h](#)
- JENC_INPUT_1D420 : [dvr_type_define.h](#)
- JENC_INPUT_1D422 : [dvr_type_define.h](#)
- JENC_INPUT_DMAWRP420 : [dvr_type_define.h](#)
- JENC_INPUT_H2642D : [dvr_type_define.h](#)
- JENC_INPUT_MP42D : [dvr_type_define.h](#)

- l -

- LCD_COLOR_ARGB : [dvr_type_define.h](#)
- LCD_COLOR_RGB : [dvr_type_define.h](#)
- LCD_COLOR_RGB444 : [dvr_type_define.h](#)
- LCD_COLOR_RGB555 : [dvr_type_define.h](#)
- LCD_COLOR_RGB565 : [dvr_type_define.h](#)
- LCD_COLOR_RGB8 : [dvr_type_define.h](#)
- LCD_COLOR_RGB888 : [dvr_type_define.h](#)
- LCD_COLOR_YUV420 : [dvr_type_define.h](#)
- LCD_COLOR_YUV422 : [dvr_type_define.h](#)
- LCD_INTERLACING : [dvr_type_define.h](#)
- LCD_PROGRESSIVE : [dvr_type_define.h](#)
- LCD_RES_1360x768 : [dvr_type_define.h](#)

- LCD_RES_COUNT : [dvr_type_define.h](#)
- LCD_RES_D1 : [dvr_type_define.h](#)
- LCD_RES_SVGA : [dvr_type_define.h](#)
- LCD_RES_SXGA : [dvr_type_define.h](#)
- LCD_RES_XGA : [dvr_type_define.h](#)
- LCD_RES_XVGA : [dvr_type_define.h](#)
- LVFRAME_ENLARGE_ONE_FIELD : [dvr_type_define.h](#)
- LVFRAME_EVEN_ODD : [dvr_type_define.h](#)
- LVFRAME_FIELD_MODE : [dvr_type_define.h](#)
- LVFRAME_FIELD_MODE2 : [dvr_type_define.h](#)
- LVFRAME_FRAME_MODE : [dvr_type_define.h](#)
- LVFRAME_GM3DI_FORMAT : [dvr_type_define.h](#)
- LVFRAME_WEAVED_TWO_FIELDS : [dvr_type_define.h](#)

- p -

- PLANE_PARAM_APPLY : [dvr_disp_api.h](#)
- PLANE_PARAM_COLOR_MODE : [dvr_disp_api.h](#)
- PLANE_PARAM_DATA_MODE : [dvr_disp_api.h](#)
- PLANE_PARAM_WINDOW : [dvr_disp_api.h](#)

- q -

- QID_3DI_SCL : [dvr_type_define.h](#)
- QID_DEC_IN : [dvr_type_define.h](#)
- QID_ENC_IN : [dvr_type_define.h](#)
- QID_ENC_OUT : [dvr_type_define.h](#)
- QID_LCD : [dvr_type_define.h](#)
- QID_LV_SCL : [dvr_type_define.h](#)
- QID_PB_SCL : [dvr_type_define.h](#)
- QID_SS_ENC_IN : [dvr_type_define.h](#)
- QID_SS_ENC_OUT : [dvr_type_define.h](#)
- QID_SUB1_ENC_IN : [dvr_type_define.h](#)
- QID_SUB1_ENC_OUT : [dvr_type_define.h](#)
- QID_SUB2_ENC_IN : [dvr_type_define.h](#)
- QID_SUB2_ENC_OUT : [dvr_type_define.h](#)

- s -

- SCALE_H264_YUV420_MODE0 : [dvr_type_define.h](#)
- SCALE_H264_YUV420_MODE1 : [dvr_type_define.h](#)
- SCALE_LINEAR : [dvr_type_define.h](#)
- SCALE_METHOD_COUNT : [dvr_type_define.h](#)
- SCALE_MP4_YUV420_MODE0 : [dvr_type_define.h](#)
- SCALE_MP4_YUV420_MODE1 : [dvr_type_define.h](#)
- SCALE_NON_LINEAR : [dvr_type_define.h](#)
- SCALE_RGB565 : [dvr_type_define.h](#)
- SCALE_RGB888 : [dvr_type_define.h](#)
- SCALE_YUV422 : [dvr_type_define.h](#)
- SCALE_YUV444 : [dvr_type_define.h](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples				
File List	Globals						
All	Typedefs	Enumerations	Enumerator				
c	d	f	g	m	p	q	t

- C -

- CAP_BUF_ID : [dvr_disp_api.h](#)

- d -

- DBG_DVR_DATA_FLOW : [dvr_type_define.h](#)
- DBG_DVR_FNC : [dvr_type_define.h](#)
- DBG_ENTITY_FNC : [dvr_type_define.h](#)
- DBG_ENTITY_JOB_FLOW : [dvr_type_define.h](#)
- DBG_GRAPH_DATA : [dvr_type_define.h](#)
- DBG_GRAPH_FNC : [dvr_type_define.h](#)
- DEFAULT_CIF_HEIGHT : [dvr_type_define.h](#)
- DEFAULT_CIF_WIDTH : [dvr_type_define.h](#)
- DEFAULT_D1_HEIGHT : [dvr_type_define.h](#)
- DEFAULT_D1_WIDTH : [dvr_type_define.h](#)
- DEFAULT_QCIF_HEIGHT : [dvr_type_define.h](#)
- DEFAULT_QCIF_WIDTH : [dvr_type_define.h](#)
- DVR_COMMON_APPLY : [dvr_common_api.h](#)
- DVR_COMMON_DEBUG : [dvr_common_api.h](#)
- DVR_COMMON_DEBUG_WITH_PANIC : [dvr_common_api.h](#)
- DVR_DEC_CLEAR_WIN : [dvr_dec_ioctl.h](#)
- DVR_DEC_CLEAR_WIN2 : [dvr_dec_ioctl.h](#)
- DVR_DEC_CONTROL : [dvr_dec_ioctl.h](#)
- DVR_DEC_GET_CHANNEL_PARAM : [dvr_dec_ioctl.h](#)
- DVR_DEC_IOC_MAGIC : [dvr_dec_ioctl.h](#)
- DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE : [dvr_dec_ioctl.h](#)
- DVR_DEC_QUEUE_GET : [dvr_dec_ioctl.h](#)
- DVR_DEC_QUEUE_PUT : [dvr_dec_ioctl.h](#)
- DVR_DEC_SET_CHANNEL_PARAM : [dvr_dec_ioctl.h](#)

- DVR_DISP_CHECK_MAGIC : [dvr_disp_api.h](#)
- DVR_DISP_CLEAR_WIN : [dvr_disp_ioctl.h](#)
- DVR_DISP_CONTROL : [dvr_disp_ioctl.h](#)
- DVR_DISP_GET_DISP_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_GET_PLANE_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_GET_VALUE : [dvr_disp_api.h](#)
- DVR_DISP_INITIATE : [dvr_disp_ioctl.h](#)
- DVR_DISP_IOC_MAGIC : [dvr_disp_ioctl.h](#)
- DVR_DISP_MAGIC_ADD_VAL : [dvr_disp_api.h](#)
- DVR_DISP_SET_DISP_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_SET_PLANE_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_TERMINATE : [dvr_disp_ioctl.h](#)
- DVR_DISP_UPDATE_DISP_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_UPDATE_PLANE_PARAM : [dvr_disp_ioctl.h](#)
- DVR_ENC_CHECK_MAGIC : [dvr_enc_api.h](#)
- DVR_ENC_CONTROL : [dvr_enc_ioctl.h](#)
- DVR_ENC_EBST_DISABLE : [dvr_enc_api.h](#)
- DVR_ENC_EBST_ENABLE : [dvr_enc_api.h](#)
- DVR_ENC_ENHANCE_H264_RATECONTROL : [dvr_enc_api.h](#)
- DVR_ENC_GET_CHANNEL_PARAM : [dvr_enc_ioctl.h](#)
- DVR_ENC_GET_VALUE : [dvr_enc_api.h](#)
- DVR_ENC_H264_WATERMARK : [dvr_enc_api.h](#)
- DVR_ENC_IOC_MAGIC : [dvr_enc_ioctl.h](#)
- DVR_ENC_MAGIC : [dvr_enc_api.h](#)
- DVR_ENC_MAGIC_ADD_VAL : [dvr_enc_api.h](#)
- DVR_ENC_MAGIC_SHIFT : [dvr_enc_api.h](#)
- DVR_ENC_MJPEG_FUNCTION : [dvr_enc_api.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP_OFFSET :
[dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1_BS_OFFSET :
[dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB2_BS_OFFSET :
[dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB3_BS_OFFSET :
[dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB4_BS_OFFSET :
[dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB5_BS_OFFSET :
[dvr_enc_ioctl.h](#)

dvr_enc_ioctl.h

- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB6_BS_OFFSET : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB7_BS_OFFSET : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB8_BS_OFFSET : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SNAP : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB1_BS : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB1_BS_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB2_BS : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB2_BS_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB3_BS : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB3_BS_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB4_BS : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB4_BS_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB5_BS : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB5_BS_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB6_BS : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB6_BS_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB7_BS : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB7_BS_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB8_BS : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_GET_SUB8_BS_COPY : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_QUEUE_PUT : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_RESET_INTRA : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_ROI_ALL : [**dvr_enc_api.h**](#)
- DVR_ENC_ROI_POS : [**dvr_enc_api.h**](#)
- DVR_ENC_SET_CHANNEL_PARAM : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_SET_SUB_BS_PARAM : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_SUB_PATH_DENOISE_CTRL : [**dvr_enc_ioctl.h**](#)
- DVR_ENC_SWAP_INTRA : [**dvr_enc_ioctl.h**](#)
- DVR_IOC_MAGIC : [**dvr_common_api.h**](#)
- DVR_PARAM_MAGIC : [**dvr_disp_api.h**](#)
- DVR_PARAM_MAGIC_SHIFT : [**dvr_disp_api.h**](#)
- DVR_PLANE_ID : [**dvr_disp_api.h**](#)

- f -

- FALSE : [dvr_type_define.h](#)
- FN CASCADE : [dvr_type_define.h](#)
- FN CHECK_MASK : [dvr_type_define.h](#)
- FN COMPARE : [dvr_type_define.h](#)
- FN COPY_TAG : [dvr_type_define.h](#)
- FN IS_EMPTY : [dvr_type_define.h](#)
- FN IS_FN : [dvr_type_define.h](#)
- FN IS_UPDATE : [dvr_type_define.h](#)
- FN ITEMS : [dvr_type_define.h](#)
- FN LCD_PARAM : [dvr_type_define.h](#)
- FN LIVEVIEW : [dvr_type_define.h](#)
- FN METHOD_USE_CMP : [dvr_type_define.h](#)
- FN NONE : [dvr_type_define.h](#)
- FN PB_SCL_LINK : [dvr_type_define.h](#)
- FN PLANE_PARAM : [dvr_type_define.h](#)
- FN PLAYBACK : [dvr_type_define.h](#)
- FN RECORD : [dvr_type_define.h](#)
- FN REMOVE_FUNC : [dvr_type_define.h](#)
- FN REMOVE_LV_CH : [dvr_type_define.h](#)
- FN RESET_TAG : [dvr_type_define.h](#)
- FN SET_ALL : [dvr_type_define.h](#)
- FN SET_CAS_CH : [dvr_type_define.h](#)
- FN SET_FUNC : [dvr_type_define.h](#)
- FN SET_LV_CH : [dvr_type_define.h](#)
- FN SET_PB_CH : [dvr_type_define.h](#)
- FN SET_REC_CH : [dvr_type_define.h](#)
- FN SET_SUB1_REC_CH : [dvr_type_define.h](#)
- FN SET_SUB2_REC_CH : [dvr_type_define.h](#)
- FN SET_SUB3_REC_CH : [dvr_type_define.h](#)
- FN SET_SUB4_REC_CH : [dvr_type_define.h](#)
- FN SET_SUB5_REC_CH : [dvr_type_define.h](#)
- FN SET_SUB6_REC_CH : [dvr_type_define.h](#)
- FN SET_SUB7_REC_CH : [dvr_type_define.h](#)
- FN SET_SUB8_REC_CH : [dvr_type_define.h](#)
- FN SUB1_RECORD : [dvr_type_define.h](#)
- FN SUB2_RECORD : [dvr_type_define.h](#)
- FN SUB3_RECORD : [dvr_type_define.h](#)

- FN_SUB4_RECORD : [dvr_type_define.h](#)
- FN_SUB5_RECORD : [dvr_type_define.h](#)
- FN_SUB6_RECORD : [dvr_type_define.h](#)
- FN_SUB7_RECORD : [dvr_type_define.h](#)
- FN_SUB8_RECORD : [dvr_type_define.h](#)
- FN_SUB_ALL_RECORD : [dvr_type_define.h](#)
- FN_UPDATE_METHOD : [dvr_type_define.h](#)

- g -

- GET_DISP_NUM_FROM_PLANE_ID : [dvr_disp_api.h](#)
- GET_PLANE_NUM_FROM_PLANE_ID : [dvr_disp_api.h](#)
- GFID_DEC : [dvr_type_define.h](#)
- GFID_DISP : [dvr_type_define.h](#)
- GFID_ENC : [dvr_type_define.h](#)
- GMDVR_MAKE_FOURCC : [dvr_type_define.h](#)
- GMDVR_MEM_CFG_FILE : [dvr_type_define.h](#)
- GMVAL_DO_NOT_CARE : [dvr_type_define.h](#)
- GMVAL_RATE : [dvr_type_define.h](#)
- GMVAL_RT_GET_BASE : [dvr_type_define.h](#)
- GMVAL_RT_GET_VAL : [dvr_type_define.h](#)

- m -

- MCP_VIDEO_NTSC : [dvr_type_define.h](#)
- MCP_VIDEO_PAL : [dvr_type_define.h](#)
- MCP_VIDEO_VGA : [dvr_type_define.h](#)
- MTHD_USE_CMP : [dvr_type_define.h](#)

- p -

- POLLIN_MAIN_BS : [dvr_enc_api.h](#)
- POLLIN_SNAP_BS : [dvr_enc_api.h](#)
- POLLIN_SUB1_BS : [dvr_enc_api.h](#)
- POLLIN_SUB2_BS : [dvr_enc_api.h](#)
- POLLIN_SUB3_BS : [dvr_enc_api.h](#)
- POLLIN_SUB4_BS : [dvr_enc_api.h](#)
- POLLIN_SUB5_BS : [dvr_enc_api.h](#)
- POLLIN_SUB6_BS : [dvr_enc_api.h](#)

- POLLIN_SUB7_BS : [dvr_enc_api.h](#)
- POLLIN_SUB8_BS : [dvr_enc_api.h](#)

- q -

- QNAME_3DI_SCL : [dvr_type_define.h](#)
- QNAME_DEC_IN : [dvr_type_define.h](#)
- QNAME_ENC_IN : [dvr_type_define.h](#)
- QNAME_ENC_OUT : [dvr_type_define.h](#)
- QNAME_LCD : [dvr_type_define.h](#)
- QNAME_LV_SCL : [dvr_type_define.h](#)
- QNAME_PB_SCL : [dvr_type_define.h](#)
- QNAME_SS_ENC_IN : [dvr_type_define.h](#)
- QNAME_SS_ENC_OUT : [dvr_type_define.h](#)
- QNAME_SUB1_ENC_IN : [dvr_type_define.h](#)
- QNAME_SUB1_ENC_OUT : [dvr_type_define.h](#)
- QNAME_SUB2_ENC_IN : [dvr_type_define.h](#)
- QNAME_SUB2_ENC_OUT : [dvr_type_define.h](#)

- t -

- TRUE : [dvr_type_define.h](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples											
File List	Globals													
All	Typedefs	Enumerations	Enumerator	Defines										
b	c	d	e	f	g	j	l	m	p	q	r	s	t	v

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- C -

- CAP_BUF_ID : [dvr_disp_api.h](#)
- CAPCOLOR_RGB565 : [dvr_type_define.h](#)
- CAPCOLOR_RGB888 : [dvr_type_define.h](#)
- CAPCOLOR_YUV420_M0 : [dvr_type_define.h](#)
- CAPCOLOR_YUV420_M1 : [dvr_type_define.h](#)
- CAPCOLOR_YUV422 : [dvr_type_define.h](#)
- CAPPATH_DEFAULT : [dvr_type_define.h](#)
- CAPPATH_PATH_1 : [dvr_type_define.h](#)
- CAPPATH_PATH_2 : [dvr_type_define.h](#)
- CAPSCALER_KEEP_RATIO : [dvr_type_define.h](#)
- CAPSCALER_NOT_KEEP_RATIO : [dvr_type_define.h](#)
- CaptureColorMode : [dvr_type_define.h](#)
- CaptureColorModeTag : [dvr_type_define.h](#)
- CapturePath : [dvr_type_define.h](#)
- CapturePathTag : [dvr_type_define.h](#)

Main Page	Data Structures	Files	Examples											
File List	Globals													
All	Typedefs	Enumerations	Enumerator	Defines										
b	c	d	e	f	g	j	l	m	p	q	r	s	t	v

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- d -

- DBG_DVR_DATA_FLOW : [dvr_type_define.h](#)
- DBG_DVR_FNC : [dvr_type_define.h](#)
- DBG_ENTITY_FNC : [dvr_type_define.h](#)
- DBG_ENTITY_JOB_FLOW : [dvr_type_define.h](#)
- DBG_GRAPH_DATA : [dvr_type_define.h](#)
- DBG_GRAPH_FNC : [dvr_type_define.h](#)
- DEC_OUTPUT_COLOR_YUV420 : [dvr_type_define.h](#)
- DEC_OUTPUT_COLOR_YUV422 : [dvr_type_define.h](#)
- DEC_START : [dvr_dec_api.h](#)
- DEC_STOP : [dvr_dec_api.h](#)
- DEC_TYPE_COUNT : [dvr_dec_api.h](#)
- DEC_TYPE_TO_BUFFER : [dvr_dec_api.h](#)
- DEC_TYPE_TO_DISPLAY : [dvr_dec_api.h](#)
- DEC_UPDATE : [dvr_dec_api.h](#)
- DecoderOutputColor : [dvr_type_define.h](#)
- DecoderOutputColorTag : [dvr_type_define.h](#)
- DecParam : [dvr_dec_api.h](#)
- DEFAULT_CIF_HEIGHT : [dvr_type_define.h](#)
- DEFAULT_CIF_WIDTH : [dvr_type_define.h](#)
- DEFAULT_D1_HEIGHT : [dvr_type_define.h](#)
- DEFAULT_D1_WIDTH : [dvr_type_define.h](#)
- DEFAULT_QCIF_HEIGHT : [dvr_type_define.h](#)
- DEFAULT_QCIF_WIDTH : [dvr_type_define.h](#)
- DIM : [dvr_type_define.h](#)
- DISP_LAYER0_CHN : [dvr_disp_api.h](#)
- DISP_LAYER1_CHN : [dvr_disp_api.h](#)
- DISP_NORMAL_CHN : [dvr_disp_api.h](#)

- DISP_PARAM_APPLY : [dvr_disp_api.h](#)
- DISP_PARAM_COLOR_ATTRIBUTE : [dvr_disp_api.h](#)
- DISP_PARAM_OUTPUT_MODE : [dvr_disp_api.h](#)
- DISP_PARAM_OUTPUT_SYSTEM : [dvr_disp_api.h](#)
- DISP_PARAM_PLANE_COMBINATION : [dvr_disp_api.h](#)
- DISP_PARAM_RESOLUTION : [dvr_disp_api.h](#)
- DISP_PARAM_TARGET : [dvr_disp_api.h](#)
- DISP_PARAM_TRANSPARENT_COLOR : [dvr_disp_api.h](#)
- DISP_RUN : [dvr_disp_api.h](#)
- DISP_START : [dvr_disp_api.h](#)
- DISP_STOP : [dvr_disp_api.h](#)
- DISP_TYPE CASCADE : [dvr_disp_api.h](#)
- DISP_TYPE_LIVEVIEW : [dvr_disp_api.h](#)
- DISP_TYPE_ON_BUFFER : [dvr_disp_api.h](#)
- DISP_TYPE_PLAYBACK : [dvr_disp_api.h](#)
- DISP_UPDATE : [dvr_disp_api.h](#)
- DispParam_Ext1 : [dvr_disp_api.h](#)
- DMAORDER_2PLANAR : [dvr_type_define.h](#)
- DMAORDER_3PLANAR : [dvr_type_define.h](#)
- DMAORDER_PACKET : [dvr_type_define.h](#)
- dvr_bs_data : [dvr_type_define.h](#)
- DVR_COMMON_APPLY : [dvr_common_api.h](#)
- DVR_COMMON_DEBUG : [dvr_common_api.h](#)
- DVR_COMMON_DEBUG_WITH_PANIC : [dvr_common_api.h](#)
- dvr_dec_channel_param : [dvr_dec_api.h](#)
- dvr_dec_clear_param : [dvr_dec_api.h](#)
- DVR_DEC_CLEAR_WIN : [dvr_dec_ioctl.h](#)
- DVR_DEC_CLEAR_WIN2 : [dvr_dec_ioctl.h](#)
- DVR_DEC_CONTROL : [dvr_dec_ioctl.h](#)
- dvr_dec_control : [dvr_dec_api.h](#)
- dvr_dec_ctrl_cmd : [dvr_dec_api.h](#)
- dvr_dec_ctrl_cmd_tag : [dvr_dec_api.h](#)
- dvr_dec_dest_type : [dvr_dec_api.h](#)
- dvr_dec_dest_type_tag : [dvr_dec_api.h](#)
- DVR_DEC_GET_CHANNEL_PARAM : [dvr_dec_ioctl.h](#)
- DVR_DEC_IOC_MAGIC : [dvr_dec_ioctl.h](#)
- DVR_DEC_QUERY_OUTPUT_BUFFER_SIZE : [dvr_dec_ioctl.h](#)
- DVR_DEC_QUEUE_GET : [dvr_dec_ioctl.h](#)
- dvr_dec_queue_get : [dvr_dec_api.h](#)

- DVR_DEC_QUEUE_PUT : [dvr_dec_ioctl.h](#)
- DVR_DEC_SET_CHANNEL_PARAM : [dvr_dec_ioctl.h](#)
- dvr_disp_channel_type : [dvr_disp_api.h](#)
- dvr_disp_channel_type_tag : [dvr_disp_api.h](#)
- DVR_DISP_CHECK_MAGIC : [dvr_disp_api.h](#)
- dvr_disp_clear_param : [dvr_disp_api.h](#)
- DVR_DISP_CLEAR_WIN : [dvr_disp_ioctl.h](#)
- dvr_disp_color_attribute : [dvr_disp_api.h](#)
- dvr_disp_color_key : [dvr_disp_api.h](#)
- DVR_DISP_CONTROL : [dvr_disp_ioctl.h](#)
- dvr_disp_control : [dvr_disp_api.h](#)
- dvr_disp_ctrl_cmd : [dvr_disp_api.h](#)
- dvr_disp_ctrl_cmd_tag : [dvr_disp_api.h](#)
- dvr_disp_disp_param : [dvr_disp_api.h](#)
- dvr_disp_disp_param_name : [dvr_disp_api.h](#)
- dvr_disp_disp_param_name_tag : [dvr_disp_api.h](#)
- DVR_DISP_GET_DISP_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_GET_PLANE_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_GET_VALUE : [dvr_disp_api.h](#)
- DVR_DISP_INITIATE : [dvr_disp_ioctl.h](#)
- DVR_DISP_IOC_MAGIC : [dvr_disp_ioctl.h](#)
- DVR_DISP_MAGIC_ADD_VAL : [dvr_disp_api.h](#)
- dvr_disp_plane_combination : [dvr_disp_api.h](#)
- dvr_disp_plane_combination_tag : [dvr_disp_api.h](#)
- dvr_disp_plane_param : [dvr_disp_api.h](#)
- dvr_disp_plane_param_name : [dvr_disp_api.h](#)
- dvr_disp_plane_param_name_tag : [dvr_disp_api.h](#)
- dvr_disp_plane_param_st : [dvr_disp_api.h](#)
- dvr_disp_resolution : [dvr_disp_api.h](#)
- dvr_disp_scaler_info : [dvr_disp_api.h](#)
- DVR_DISP_SET_DISP_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_SET_PLANE_PARAM : [dvr_disp_ioctl.h](#)
- DVR_DISP_TERMINATE : [dvr_disp_ioctl.h](#)
- dvr_disp_type : [dvr_disp_api.h](#)
- dvr_disp_update_disp_param : [dvr_disp_api.h](#)
- DVR_DISP_UPDATE_DISP_PARAM : [dvr_disp_ioctl.h](#)
- dvr_disp_update_plane_param : [dvr_disp_api.h](#)
- DVR_DISP_UPDATE_PLANE_PARAM : [dvr_disp_ioctl.h](#)
- dvr_disp_vbi_info : [dvr_disp_api.h](#)

- dvr_enc_channel_param : [dvr_enc_api.h](#)
- dvr_enc_channel_param_name : [dvr_enc_api.h](#)
- dvr_enc_channel_param_name_tag : [dvr_enc_api.h](#)
- DVR_ENC_CHECK_MAGIC : [dvr_enc_api.h](#)
- dvr_enc_control : [dvr_enc_api.h](#)
- DVR_ENC_CONTROL : [dvr_enc_ioctl.h](#)
- dvr_enc_copy_buf : [dvr_enc_api.h](#)
- dvr_enc_ctrl_cmd : [dvr_enc_api.h](#)
- DVR_ENC_EBST_DISABLE : [dvr_enc_api.h](#)
- DVR_ENC_EBST_ENABLE : [dvr_enc_api.h](#)
- DVR_ENC_ENHANCE_H264_RATECONTROL : [dvr_enc_api.h](#)
- DVR_ENC_GET_CHANNEL_PARAM : [dvr_enc_ioctl.h](#)
- DVR_ENC_GET_VALUE : [dvr_enc_api.h](#)
- DVR_ENC_H264_WATERMARK : [dvr_enc_api.h](#)
- DVR_ENC_IOC_MAGIC : [dvr_enc_ioctl.h](#)
- DVR_ENC_MAGIC : [dvr_enc_api.h](#)
- DVR_ENC_MAGIC_ADD_VAL : [dvr_enc_api.h](#)
- DVR_ENC_MAGIC_SHIFT : [dvr_enc_api.h](#)
- DVR_ENC_MJPEG_FUNCTION : [dvr_enc_api.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SIZE : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SNAP_OFFSET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB1_BS_OFFSET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB2_BS_OFFSET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB3_BS_OFFSET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB4_BS_OFFSET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB5_BS_OFFSET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB6_BS_OFFSET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB7_BS_OFFSET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUERY_OUTPUT_BUFFER_SUB8_BS_OFFSET : [dvr_enc_ioctl.h](#)
- dvr_enc_queue_get : [dvr_enc_api.h](#)

- DVR_ENC_QUEUE_GET : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SNAP : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB1_BS : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB1_BS_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB2_BS : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB2_BS_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB3_BS : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB3_BS_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB4_BS : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB4_BS_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB5_BS : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB5_BS_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB6_BS : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB6_BS_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB7_BS : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB7_BS_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB8_BS : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_GET_SUB8_BS_COPY : [dvr_enc_ioctl.h](#)
- DVR_ENC_QUEUE_PUT : [dvr_enc_ioctl.h](#)
- DVR_ENC_RESET_INTRA : [dvr_enc_ioctl.h](#)
- DVR_ENC_ROI_ALL : [dvr_enc_api.h](#)
- DVR_ENC_ROI_POS : [dvr_enc_api.h](#)
- DVR_ENC_SET_CHANNEL_PARAM : [dvr_enc_ioctl.h](#)
- DVR_ENC_SET_SUB_BS_PARAM : [dvr_enc_ioctl.h](#)
- dvr_enc_src_param : [dvr_enc_api.h](#)
- dvr_enc_src_type : [dvr_enc_api.h](#)
- dvr_enc_src_type_tag : [dvr_enc_api.h](#)
- DVR_ENC_SUB_PATH_DENOISE_CTRL : [dvr_enc_ioctl.h](#)
- DVR_ENC_SWAP_INTRA : [dvr_enc_ioctl.h](#)
- dvr_enc_update_channel_param : [dvr_enc_api.h](#)
- dvr_graph_vqueue : [dvr_type_define.h](#)
- DVR_IOC_MAGIC : [dvr_common_api.h](#)
- DVR_PARAM_MAGIC : [dvr_disp_api.h](#)
- DVR_PARAM_MAGIC_SHIFT : [dvr_disp_api.h](#)
- DVR_PLANE_ID : [dvr_disp_api.h](#)
- dvr_ratio : [dvr_type_define.h](#)

doxygen 1.7.1

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
b c d e f g j l m p q r s t v				

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- e -

- ENC_INPUT_1D420 : [dvr_type_define.h](#)
- ENC_INPUT_1D422 : [dvr_type_define.h](#)
- ENC_INPUT_H2642D : [dvr_type_define.h](#)
- ENC_INPUT_MP42D : [dvr_type_define.h](#)
- ENC_PARAM_DST_WIN : [dvr_enc_api.h](#)
- ENC_PARAM_SRC_DIM : [dvr_enc_api.h](#)
- ENC_RAW : [dvr_enc_api.h](#)
- ENC_SNAP : [dvr_enc_api.h](#)
- ENC_SRC_TYPE_COUNT : [dvr_enc_api.h](#)
- ENC_START : [dvr_enc_api.h](#)
- ENC_STOP : [dvr_enc_api.h](#)
- ENC_TYPE_COUNT : [dvr_type_define.h](#)
- ENC_TYPE_FROM_BUFFER : [dvr_enc_api.h](#)
- ENC_TYPE_FROM_CAPTURE : [dvr_enc_api.h](#)
- ENC_TYPE_FROM CASCADE : [dvr_enc_api.h](#)
- ENC_TYPE_H264 : [dvr_type_define.h](#)
- ENC_TYPE_MJPEG : [dvr_type_define.h](#)
- ENC_TYPE_MPEG : [dvr_type_define.h](#)
- ENC_TYPE_YUV422 : [dvr_type_define.h](#)
- ENC_UPDATE : [dvr_enc_api.h](#)
- EncoderInputFormat : [dvr_type_define.h](#)
- EncoderInputFormatTag : [dvr_type_define.h](#)
- EncodeType : [dvr_type_define.h](#)
- EncodeType_tag : [dvr_type_define.h](#)
- EncParam : [dvr_enc_api.h](#)
- EncParam_Ext1 : [dvr_enc_api.h](#)
- EncParam_Ext2 : [dvr_enc_api.h](#)

- EncParam_Ext3 : **dvr_enc_api.h**
 - EncParam_Ext4 : **dvr_enc_api.h**
 - EncParam_Ext5 : **dvr_enc_api.h**
-

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
b c d e f g j l m p q r s t v				

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- f -

- FALSE : [dvr_type_define.h](#)
- FN CASCADE : [dvr_type_define.h](#)
- FN CHECK_MASK : [dvr_type_define.h](#)
- FN COMPARE : [dvr_type_define.h](#)
- FN COPY_TAG : [dvr_type_define.h](#)
- FN IS_EMPTY : [dvr_type_define.h](#)
- FN IS_FN : [dvr_type_define.h](#)
- FN IS_UPDATE : [dvr_type_define.h](#)
- FN ITEMS : [dvr_type_define.h](#)
- FN_LCD_PARAM : [dvr_type_define.h](#)
- FN_LIVEVIEW : [dvr_type_define.h](#)
- FN_METHOD_USE_CMP : [dvr_type_define.h](#)
- FN_NONE : [dvr_type_define.h](#)
- FN_PB_SCL_LINK : [dvr_type_define.h](#)
- FN_PLANE_PARAM : [dvr_type_define.h](#)
- FN_PLAYBACK : [dvr_type_define.h](#)
- FN_RECORD : [dvr_type_define.h](#)
- FN_REMOVE_FUNC : [dvr_type_define.h](#)
- FN_REMOVE_LV_CH : [dvr_type_define.h](#)
- FN_RESET_TAG : [dvr_type_define.h](#)
- FN_SET_ALL : [dvr_type_define.h](#)
- FN_SET_CAS_CH : [dvr_type_define.h](#)
- FN_SET_FUNC : [dvr_type_define.h](#)
- FN_SET_LV_CH : [dvr_type_define.h](#)
- FN_SET_PB_CH : [dvr_type_define.h](#)
- FN_SET_REC_CH : [dvr_type_define.h](#)
- FN_SET_SUB1_REC_CH : [dvr_type_define.h](#)

- FN_SET_SUB2_REC_CH : [dvr_type_define.h](#)
- FN_SET_SUB3_REC_CH : [dvr_type_define.h](#)
- FN_SET_SUB4_REC_CH : [dvr_type_define.h](#)
- FN_SET_SUB5_REC_CH : [dvr_type_define.h](#)
- FN_SET_SUB6_REC_CH : [dvr_type_define.h](#)
- FN_SET_SUB7_REC_CH : [dvr_type_define.h](#)
- FN_SET_SUB8_REC_CH : [dvr_type_define.h](#)
- FN_SUB1_RECORD : [dvr_type_define.h](#)
- FN_SUB2_RECORD : [dvr_type_define.h](#)
- FN_SUB3_RECORD : [dvr_type_define.h](#)
- FN_SUB4_RECORD : [dvr_type_define.h](#)
- FN_SUB5_RECORD : [dvr_type_define.h](#)
- FN_SUB6_RECORD : [dvr_type_define.h](#)
- FN_SUB7_RECORD : [dvr_type_define.h](#)
- FN_SUB8_RECORD : [dvr_type_define.h](#)
- FN_SUB_ALL_RECORD : [dvr_type_define.h](#)
- FN_UPDATE_METHOD : [dvr_type_define.h](#)
- FuncTag : [dvr_type_define.h](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
b c d e f g j l m p q r s t v				

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- g -

- GET_DISP_NUM_FROM_PLANE_ID : [dvr_disp_api.h](#)
- GET_PLANE_NUM_FROM_PLANE_ID : [dvr_disp_api.h](#)
- GFID_DEC : [dvr_type_define.h](#)
- GFID_DISP : [dvr_type_define.h](#)
- GFID_ENC : [dvr_type_define.h](#)
- GM3DI_FIELD : [dvr_type_define.h](#)
- GM3DI_FRAME : [dvr_type_define.h](#)
- GM3DIFrameType : [dvr_type_define.h](#)
- GM3DIFrameTypeTag : [dvr_type_define.h](#)
- GMDVR_MAKE_FOURCC : [dvr_type_define.h](#)
- GMDVR_MEM_CFG_FILE : [dvr_type_define.h](#)
- GMVAL_DO_NOT CARE : [dvr_type_define.h](#)
- GMVAL_RATE : [dvr_type_define.h](#)
- GMVAL_RT_GET_BASE : [dvr_type_define.h](#)
- GMVAL_RT_GET_VAL : [dvr_type_define.h](#)

Main Page	Data Structures	Files	Examples											
File List	Globals													
All	Typedefs	Enumerations	Enumerator	Defines										
b	c	d	e	f	g	j	l	m	p	q	r	s	t	v

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- j -

- JCS_yuv111 : [dvr_type_define.h](#)
- JCS_yuv211 : [dvr_type_define.h](#)
- JCS_yuv222 : [dvr_type_define.h](#)
- JCS_yuv333 : [dvr_type_define.h](#)
- JCS_yuv400 : [dvr_type_define.h](#)
- JCS_yuv420 : [dvr_type_define.h](#)
- JCS_yuv422 : [dvr_type_define.h](#)
- JENC_INPUT_1D420 : [dvr_type_define.h](#)
- JENC_INPUT_1D422 : [dvr_type_define.h](#)
- JENC_INPUT_DMAWRP420 : [dvr_type_define.h](#)
- JENC_INPUT_H2642D : [dvr_type_define.h](#)
- JENC_INPUT_MP42D : [dvr_type_define.h](#)
- JpegEnc420InputFormat : [dvr_type_define.h](#)
- JpegEnc420InputFormatTag : [dvr_type_define.h](#)
- JpegEnclnputFormat : [dvr_type_define.h](#)
- JpegEnclnputFormatTag : [dvr_type_define.h](#)

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
b c d e f g j I m p q r s t v				

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- I -

- LCD_COLOR_ARGB : [dvr_type_define.h](#)
- LCD_COLOR_RGB : [dvr_type_define.h](#)
- LCD_COLOR_RGB444 : [dvr_type_define.h](#)
- LCD_COLOR_RGB555 : [dvr_type_define.h](#)
- LCD_COLOR_RGB565 : [dvr_type_define.h](#)
- LCD_COLOR_RGB8 : [dvr_type_define.h](#)
- LCD_COLOR_RGB888 : [dvr_type_define.h](#)
- LCD_COLOR_YUV420 : [dvr_type_define.h](#)
- LCD_COLOR_YUV422 : [dvr_type_define.h](#)
- LCD_INTERLACING : [dvr_type_define.h](#)
- LCD_PROGRESSIVE : [dvr_type_define.h](#)
- LCD_RES_1360x768 : [dvr_type_define.h](#)
- LCD_RES_COUNT : [dvr_type_define.h](#)
- LCD_RES_D1 : [dvr_type_define.h](#)
- LCD_RES_SVGA : [dvr_type_define.h](#)
- LCD_RES_SXGA : [dvr_type_define.h](#)
- LCD_RES_XGA : [dvr_type_define.h](#)
- LCD_RES_XVGA : [dvr_type_define.h](#)
- LCDOutputColorType : [dvr_type_define.h](#)
- LCDOutputColorTypeTag : [dvr_type_define.h](#)
- LCDOutputMode : [dvr_type_define.h](#)
- LCDOutputModeTag : [dvr_type_define.h](#)
- LCDResolution : [dvr_type_define.h](#)
- LCDResolutionTag : [dvr_type_define.h](#)
- LiveviewDMAOrder : [dvr_type_define.h](#)
- LiveviewDMAOrderTag : [dvr_type_define.h](#)
- LiveviewFrameMode : [dvr_type_define.h](#)

- LiveviewFrameModeTag : [dvr_type_define.h](#)
- LiveviewFrameType : [dvr_type_define.h](#)
- LiveviewFrameTypeTag : [dvr_type_define.h](#)
- LiveviewScalerRatio : [dvr_type_define.h](#)
- LiveviewScalerRatioTag : [dvr_type_define.h](#)
- LVFRAME_ENLARGE_ONE_FIELD : [dvr_type_define.h](#)
- LVFRAME_EVEN_ODD : [dvr_type_define.h](#)
- LVFRAME_FIELD_MODE : [dvr_type_define.h](#)
- LVFRAME_FIELD_MODE2 : [dvr_type_define.h](#)
- LVFRAME_FRAME_MODE : [dvr_type_define.h](#)
- LVFRAME_GM3DI_FORMAT : [dvr_type_define.h](#)
- LVFRAME_WEAVED_TWO_FIELDS : [dvr_type_define.h](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples											
File List	Globals													
All	Typedefs	Enumerations	Enumerator	Defines										
b	c	d	e	f	g	j	l	m	p	q	r	s	t	v

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- m -

- MCP_VIDEO_NTSC : [dvr_type_define.h](#)
- MCP_VIDEO_PAL : [dvr_type_define.h](#)
- MCP_VIDEO_VGA : [dvr_type_define.h](#)
- MTHD_USE_CMP : [dvr_type_define.h](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples											
File List	Globals													
All	Typedefs	Enumerations	Enumerator	Defines										
b	c	d	e	f	g	j	l	m	p	q	r	s	t	v

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- p -

- PLANE_PARAM_APPLY : [dvr_disp_api.h](#)
- PLANE_PARAM_COLOR_MODE : [dvr_disp_api.h](#)
- PLANE_PARAM_DATA_MODE : [dvr_disp_api.h](#)
- PLANE_PARAM_WINDOW : [dvr_disp_api.h](#)
- POLLIN_MAIN_BS : [dvr_enc_api.h](#)
- POLLIN_SNAP_BS : [dvr_enc_api.h](#)
- POLLIN_SUB1_BS : [dvr_enc_api.h](#)
- POLLIN_SUB2_BS : [dvr_enc_api.h](#)
- POLLIN_SUB3_BS : [dvr_enc_api.h](#)
- POLLIN_SUB4_BS : [dvr_enc_api.h](#)
- POLLIN_SUB5_BS : [dvr_enc_api.h](#)
- POLLIN_SUB6_BS : [dvr_enc_api.h](#)
- POLLIN_SUB7_BS : [dvr_enc_api.h](#)
- POLLIN_SUB8_BS : [dvr_enc_api.h](#)
- POS : [dvr_type_define.h](#)

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
b c d e f g j l m p q r s t v				

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- q -

- QID_3DI_SCL : [dvr_type_define.h](#)
- QID_DEC_IN : [dvr_type_define.h](#)
- QID_ENC_IN : [dvr_type_define.h](#)
- QID_ENC_OUT : [dvr_type_define.h](#)
- QID_LCD : [dvr_type_define.h](#)
- QID_LV_SCL : [dvr_type_define.h](#)
- QID_PB_SCL : [dvr_type_define.h](#)
- QID_SS_ENC_IN : [dvr_type_define.h](#)
- QID_SS_ENC_OUT : [dvr_type_define.h](#)
- QID_SUB1_ENC_IN : [dvr_type_define.h](#)
- QID_SUB1_ENC_OUT : [dvr_type_define.h](#)
- QID_SUB2_ENC_IN : [dvr_type_define.h](#)
- QID_SUB2_ENC_OUT : [dvr_type_define.h](#)
- QNAME_3DI_SCL : [dvr_type_define.h](#)
- QNAME_DEC_IN : [dvr_type_define.h](#)
- QNAME_ENC_IN : [dvr_type_define.h](#)
- QNAME_ENC_OUT : [dvr_type_define.h](#)
- QNAME_LCD : [dvr_type_define.h](#)
- QNAME_LV_SCL : [dvr_type_define.h](#)
- QNAME_PB_SCL : [dvr_type_define.h](#)
- QNAME_SS_ENC_IN : [dvr_type_define.h](#)
- QNAME_SS_ENC_OUT : [dvr_type_define.h](#)
- QNAME_SUB1_ENC_IN : [dvr_type_define.h](#)
- QNAME_SUB1_ENC_OUT : [dvr_type_define.h](#)
- QNAME_SUB2_ENC_IN : [dvr_type_define.h](#)
- QNAME_SUB2_ENC_OUT : [dvr_type_define.h](#)
- QueMemCfg : [dvr_type_define.h](#)

- QueueID : [dvr_type_define.h](#)
 - QueueID_tag : [dvr_type_define.h](#)
-

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page Data Structures Files Examples

File List Globals

All Typedefs Enumerations Enumerator Defines

b c d e f g j l m p q **r** s t v

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- r -

- RECT : [dvr_type_define.h](#)
- ReproduceBitStream : [dvr_enc_api.h](#)
- ROI_ALL : [dvr_type_define.h](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples	
File List	Globals			
All	Typedefs	Enumerations	Enumerator	Defines
b c d e f g j l m p q r s t v				

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- S -

- SCALE_H264_YUV420_MODE0 : [dvr_type_define.h](#)
- SCALE_H264_YUV420_MODE1 : [dvr_type_define.h](#)
- SCALE_LINEAR : [dvr_type_define.h](#)
- SCALE_METHOD_COUNT : [dvr_type_define.h](#)
- SCALE_MP4_YUV420_MODE0 : [dvr_type_define.h](#)
- SCALE_MP4_YUV420_MODE1 : [dvr_type_define.h](#)
- SCALE_NON_LINEAR : [dvr_type_define.h](#)
- SCALE_RGB565 : [dvr_type_define.h](#)
- SCALE_RGB888 : [dvr_type_define.h](#)
- SCALE_YUV422 : [dvr_type_define.h](#)
- SCALE_YUV444 : [dvr_type_define.h](#)
- ScaleColorMode : [dvr_type_define.h](#)
- ScaleColorModeTag : [dvr_type_define.h](#)
- ScaleMethod : [dvr_type_define.h](#)
- ScaleMethodTag : [dvr_type_define.h](#)
- ScalerParam : [dvr_type_define.h](#)
- snapshot_param : [dvr_enc_api.h](#)

>Main Page Data Structures Files Examples

File List Globals

All Typedefs Enumerations Enumerator Defines

b c d e f g j l m p q r s t v

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- t -

- TRUE : [dvr_type_define.h](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

>Main Page Data Structures Files Examples

File List Globals

All Typedefs Enumerations Enumerator Defines

b c d e f g j l m p q r s t v

Here is a list of all functions, variables, defines, enums, and typedefs with links to the files they belong to:

- v -

- video_process : [dvr_type_define.h](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- C -

- cap_buf_id : [dvr_disp_control_tag](#)
- cap_path : [dvr_disp_control_tag](#)
- cap_rate : [dvr_disp_control_tag](#)
- cas : [dvr_disp_control_tag](#)
- cas_ch : [FuncTag_tag](#)
- channel : [dvr_dec_queue_get_tag](#) , [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#) , [dvr_enc_queue_get_tag](#) , [dvr_dec_channel_param_tag](#)
- chn_type : [DispParam_Ext1_tag](#)
- color : [dvr_disp_color_key_tag](#)
- color_attrib : [dvr_disp_update_disp_param_tag::val_t](#) , [dvr_disp_disp_param_tag](#)
- color_mode : [dvr_disp_plane_param_tag](#) , [dvr_disp_update_plane_param_tag](#) , [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#)
- command : [dvr_disp_control_tag](#) , [dvr_dec_control_tag](#) , [dvr_enc_control_tag](#)
- contrast : [dvr_disp_color_attribute_tag](#)
- count : [QueueMemConfig_tag](#) , [dvr_enc_control_tag](#) , [dvr_graph_vqueueut_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- d -

- data_mode : [dvr_disp_plane_param_tag](#) , [dvr_disp_update_plane_param_tag](#)
- ddr : [dvr_graph_vqueue_tag](#)
- ddr_num : [QueueMemConfig_tag](#)
- dec_dest_type : [dvr_dec_channel_param_tag](#)
- dec_param : [dvr_dec_channel_param_tag](#)
- dec_type : [dvr_dec_channel_param_tag](#)
- denoise_mode : [video_process_tag](#)
- denominator : [dvr_rate_tag](#)
- des_level : [ScalerParamtag](#)
- di_mode : [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#) , [dvr_enc_update_channel_param_tag](#)
- dim : [dvr_disp_disp_param_tag](#) , [dvr_disp_control_tag](#) , [ReproduceBitStream_tag](#) , [dvr_disp_scaler_info_tag](#) , [dvr_enc_update_channel_param_tag](#) , [dvr_enc_src_tag](#) , [dvr_dec_control_tag](#)
- disp_num : [dvr_disp_plane_param_st_tag](#) , [dvr_disp_disp_param_tag](#) , [dvr_disp_update_disp_param_tag](#)
- display_rate : [dvr_dec_control_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#)
- dma_order : [dvr_enc_src_tag](#) , [dvr_disp_control_tag](#)
- dst_fmt : [ScalerParamtag](#)
- dst_param : [dvr_dec_control_tag](#) , [dvr_disp_control_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- e -

- en_snapshot : **ReproduceBitStream_tag**
- enabled : **ReproduceBitStream_tag**
- enc : **ReproduceBitStream_tag**
- enc_src_type : **dvr_enc_src_tag**
- enc_type : **EncParam_Ext5_tag** , **ReproduceBitStream_tag** ,
EncParam_Ext4_tag , **EncParam_Ext2_tag** ,
EncParam_Ext3_tag
- ext_size : **dvr_disp_control_tag** ,
dvr_enc_update_channel_param_tag , **EncParam_tag**

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- f -

- fb_offset : [dvr_disp_vbi_info_tag](#)
- fb_size : [dvr_disp_vbi_info_tag](#)
- feature_enable : [EncParam_Ext2_tag](#) , [EncParam_Ext3_tag](#) , [EncParam_Ext1_tag](#) , [EncParam_Ext5_tag](#) , [EncParam_Ext4_tag](#)
- filler : [dvr_disp_vbi_info_tag](#)
- frame_rate : [EncParam_tag](#) , [dvr_enc_update_channel_param_tag](#)
- func : [FuncTag_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- h -

- height : [DIM_tag](#) , [RECT_tag](#)
- huecos : [dvr_disp_color_attribute_tag](#)
- huesin : [dvr_disp_color_attribute_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- i -

- init_quant : [EncParam_tag](#) , [dvr_enc_update_channel_param_tag](#)
- input_res : [dvr_disp_resolution_tag](#)
- input_system : [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#)
- input_type : [EncParam_tag](#)
- ip_interval : [dvr_enc_update_channel_param_tag](#) , [EncParam_tag](#)
- is_3DI : [dvr_enc_update_channel_param_tag](#) , [video_process_tag](#)
- is_album : [ScalerParamtag](#)
- is_blocked : [dvr_dec_channel_param_tag](#) , [ReproduceBitStream_tag](#)
- is_correction : [ScalerParamtag](#)
- is_denoise : [video_process_tag](#) , [dvr_enc_update_channel_param_tag](#)
- is_display : [dvr_dec_control_tag](#)
- is_dither : [ScalerParamtag](#)
- is_enable : [dvr_disp_color_key_tag](#) , [dvr_disp_scaler_info_tag](#)
- is_keyframe : [dvr_enc_copy_buf_tag](#) , [dvr_bs_data_tag](#)
- is_use_ROI : [EncParam_tag](#) , [ROI_ALL_tag](#)
- is_use_scaler : [dvr_disp_control_tag](#) , [dvr_dec_channel_param_tag](#)

Main Page	Data Structures	Files	Examples																		
Data Structures	Data Fields																				
All	Variables																				
b	c	d	e	f	h	i	I	m	n	o	p	q	r	s	t	u	v	w	x	y	

- I -

- length : [dvr_bs_data_tag](#)
- limit_count : [QueueMemConfig_tag](#) , [dvr_graph_vqueueut_tag](#)
- lineheight : [dvr_disp_vbi_info_tag](#)
- lineno : [dvr_disp_vbi_info_tag](#)
- lv : [dvr_disp_control_tag](#)
- lv_ch : [FuncTag_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																
Data Structures	Data Fields																		
All	Variables																		
b	c	d	e	f	h	i	m	n	o	p	q	r	s	t	u	v	w	x	y

- m -

- main_bs : [dvr_enc_channel_param_tag](#)
- max_quant : [EncParam_tag](#) , [dvr_enc_update_channel_param_tag](#)
- mb_len : [dvr_enc_queue_get_tag](#)
- min_quant : [EncParam_tag](#) , [dvr_enc_update_channel_param_tag](#)
- MJ_quality : [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#) , [EncParam_Ext2_tag](#)
- mode : [dvr_enc_update_channel_param_tag](#) , [dvr_disp_control_tag](#) , [dvr_enc_src_tag](#)
- mv_buf_length : [dvr_enc_copy_buf_tag](#)
- mv_length : [dvr_enc_copy_buf_tag](#) , [dvr_bs_data_tag](#)
- mv_offset : [dvr_bs_data_tag](#)
- mv_user_va : [dvr_enc_copy_buf_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- n -

- new_bs : [dvr_enc_queue_get_tag](#) , [dvr_enc_copy_buf_tag](#)
- NonRef : [dvr_bs_data_tag](#) , [dvr_enc_copy_buf_tag](#)
- numerator : [dvr_rate_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- O -

- offset : [dvr_bs_data_tag](#)
- out_bs : [ReproduceBitStream_tag](#)
- output : [dvr_enc_control_tag](#)
- output_mode : [dvr_disp_disp_param_tag](#) ,
[dvr_disp_update_disp_param_tag::val_t](#)
- output_system : [dvr_disp_update_disp_param_tag::val_t](#) ,
[dvr_disp_disp_param_tag](#)
- output_type : [dvr_disp_resolution_tag](#) , [DecParam_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- p -

- p_job : [dvr_enc_copy_buf_tag](#) , [dvr_bs_data_tag](#)
- param : [dvr_disp_plane_param_st_tag](#) ,
[dvr_disp_update_plane_param_tag](#) ,
[dvr_disp_update_disp_param_tag](#)
- path : [dvr_disp_control_tag](#)
- pattern : [dvr_disp_clear_param_tag](#) ,
[dvr_dec_clear_param_tag](#)
- pb_ch : [FuncTag_tag](#)
- pext_data : [dvr_disp_control_tag](#) , [EncParam_tag](#) ,
[dvr_enc_update_channel_param_tag](#)
- plane_comb : [dvr_disp_disp_param_tag](#) ,
[dvr_disp_update_disp_param_tag::val_t](#)
- plane_id : [dvr_disp_control_tag](#) , [dvr_dec_control_tag](#) ,
[dvr_disp_update_plane_param_tag](#) ,
[dvr_disp_plane_param_tag](#) , [dvr_disp_clear_param_tag](#)
- plane_num : [dvr_disp_plane_param_st_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- q -

- quality : [snapshot_param_tag](#)
- que : [dvr_graph_vqueue_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- r -

- rate : [dvr_disp_control_tag](#)
- reaction_delay_max : [EncParam_Ext1_tag](#) , [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext2_tag](#) , [EncParam_Ext5_tag](#)
- rec_ch : [FuncTag_tag](#)
- rect0 : [dvr_disp_control_tag](#)
- rect1 : [dvr_disp_control_tag](#)
- repd_bs_num : [dvr_enc_control_tag](#)
- res : [dvr_disp_disp_param_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#)
- reserved : [dvr_enc_control_tag](#) , [dvr_enc_update_channel_param_tag](#) , [dvr_bs_data_tag](#) , [dvr_dec_control_tag](#) , [snapshot_param_tag](#) , [EncParam_tag](#) , [dvr_disp_clear_param_tag](#) , [ReproduceBitStream_tag](#) , [dvr_disp_update_plane_param_tag](#) , [dvr_disp_plane_param_st_tag](#) , [dvr_enc_src_tag](#) , [dvr_dec_clear_param_tag](#) , [dvr_disp_disp_param_tag](#) , [dvr_disp_resolution_tag](#) , [dvr_disp_scaler_info_tag](#) , [ScalerParamtag](#) , [dvr_disp_color_attribute_tag](#) , [dvr_enc_copy_buf_tag](#) , [dvr_disp_color_key_tag](#) , [dvr_rate_tag](#) , [dvr_dec_queue_get_tag](#) , [dvr_disp_control_tag](#) , [DecParam_tag](#) , [dvr_disp_vbi_info_tag](#) , [video_process_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#) , [dvr_disp_plane_param_tag](#) , [dvr_dec_channel_param_tag](#)
- reserved1 : [dvr_disp_update_plane_param_tag](#)
- RestartInterval : [snapshot_param_tag](#)
- roi_all : [EncParam_Ext5_tag](#)
- roi_pos : [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)

- ROI_win : **EncParam_tag**

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by
doxygen 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- S -

- sample : [snapshot_param_tag](#)
- saturation : [dvr_disp_color_attribute_tag](#)
- scale_indep : [dvr_enc_src_tag](#) , [dvr_enc_update_channel_param_tag](#) , [dvr_disp_control_tag](#)
- scale_mode : [ScalerParamtag](#)
- scl : [ReproduceBitStream_tag](#)
- scl_info : [dvr_disp_disp_param_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#)
- scl_param : [dvr_dec_channel_param_tag](#) , [dvr_disp_control_tag](#)
- sharprness_thres1 : [dvr_disp_color_attribute_tag](#)
- sharpness_thres0 : [dvr_disp_color_attribute_tag](#)
- sharpnessk0 : [dvr_disp_color_attribute_tag](#)
- sharpnessk1 : [dvr_disp_color_attribute_tag](#)
- size : [QueueMemConfig_tag](#) , [dvr_graph_vqueueut_tag](#)
- snap : [ReproduceBitStream_tag](#)
- src : [dvr_enc_update_channel_param_tag](#) , [dvr_enc_channel_param_tag](#)
- src_fmt : [ScalerParamtag](#)
- src_param : [dvr_disp_control_tag](#) , [dvr_dec_control_tag](#)
- stream : [dvr_enc_copy_buf_tag](#) , [dvr_enc_control_tag](#) , [dvr_bs_data_tag](#)
- stream_enable : [dvr_enc_update_channel_param_tag](#)
- swc_rect0 : [dvr_disp_control_tag](#)
- swc_rect1 : [dvr_disp_control_tag](#)

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- t -

- target_id : [dvr_disp_disp_param_tag](#) , [dvr_disp_update_disp_param_tag::val_t](#)
- target_rate_max : [EncParam_Ext2_tag](#) , [EncParam_Ext3_tag](#) , [EncParam_Ext1_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)
- timestamp : [dvr_enc_copy_buf_tag](#) , [dvr_bs_data_tag](#)
- transparent_color : [dvr_disp_update_disp_param_tag::val_t](#) , [dvr_disp_disp_param_tag](#)
- type : [dvr_disp_control_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- u -

- u82D : [snapshot_param_tag](#)
- update_parm : [dvr_enc_control_tag](#)
- user_tag : [dvr_graph_vqueuet_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- v -

- val : [dvr_disp_update_disp_param_tag](#) ,
[dvr_disp_update_plane_param_tag](#)
- vbi_info : [dvr_disp_update_disp_param_tag::val_t](#) ,
[dvr_disp_disp_param_tag](#)
- vp_param : [dvr_enc_src_tag](#) , [dvr_disp_control_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- W -

- watermark_enable : [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)
- watermark_init_interval : [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)
- watermark_init_pattern : [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#) , [EncParam_Ext3_tag](#)
- watermark_interval : [EncParam_Ext3_tag](#) , [EncParam_Ext4_tag](#) , [EncParam_Ext5_tag](#)
- width : [DIM_tag](#) , [RECT_tag](#)
- win : [ROI_ALL_tag](#) , [dvr_disp_control_tag](#) , [dvr_dec_control_tag](#) , [dvr_disp_plane_param_tag](#) , [dvr_dec_clear_param_tag](#) , [dvr_disp_update_plane_param_tag](#) , [dvr_disp_clear_param_tag](#)

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- X -

- x : **RECT_tag** , **POS_tag**

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

doxygen 1.7.1

Main Page	Data Structures	Files	Examples																	
Data Structures	Data Fields																			
All	Variables																			
b	c	d	e	f	h	i	l	m	n	o	p	q	r	s	t	u	v	w	x	y

- y -

- y : **RECT_tag** , **POS_tag**

Generated on Wed Jun 15 2011 15:51:00 for This describe GM8126 ioctl functions by

[doxygen](#) 1.7.1