H264 RATE CONTROL：

The H264 rate control module contains the following part:

• favc_rc.ko

This is the H264 rate control core. It includes the H264 rate control algorithm and H264 encoder communication layer.

```
/ # insmod favc_rc.ko
H264 rate control version: 1.1.9, built @ Oct 16 2014 15:00:56
```

1.3.1 rc_converge_method

This module parameter will influence the algorithm of limiting bitrate.

0: Original algorithm (Default value)

1: Strictly limit bitrate

2.1 /proc/videograph/favce_rc/info

Users can use /proc/videograph/favce_rc/info to get the rate control parameters and driver version of each channel.

Usage:

• Get the rate control parameters and the driver version of each channel

cat /proc/videograph/favce_rc/info

```
H264 rate control version: 1.1.10
chn mode fps bitrate max.br init.q min.q max.q qp I.qp
==== ==== ====== ===== ==== === ==== ==== === ===
0 CBR 30/1 1024 0 25 15 51 23 21
…
```

2.2 /proc/videograph/favce_rc/level

Users can use /proc/videograph/favce_rc/level to set the debug level. Higher debug level will dump more information to the background log.

Usage:

• Get the debug level

cat /proc/videograph/favce_rc/level

```
Log level = 2 (0: emergy, 1: error, 2: warning, 3: debug, 4: info)
```

Set the debug level

Echo [level] /proc/videograph/favce_rc/level

[level]:

0: Emergency message

1: Error message

2: Warning message

3: Debug message

4: Information

For example, if the current debug level is 2, the driver will dump the emergency message, error message, and warning message.

H264 enconder

1.1 Overview

The H264 high-profile encoder is a high-performance hardware video encoder based on the MPEG4 AVC/JVT/H.264 video coding standard. The encoder is designed to compress a sequence of YcbCr 4:2:2 pictures into a compressed video bitstream. The supported resolutions are up to 4096x4096. The features of H264 encoder include the following MPEG4 AVC/JVT/H.264 (ISO/IEC 14496-10) video coding standard:

• High profile level 4.1
• Supports I, B, and P frame encodings
• Supports CAVLC and CABAC entropy coding
• Includes de-interlace and de-noise preprocessor
• Configures horizontal and vertical search ranges as 16pixels ~ 256pixels with configuration steps

of 8pixels

- CBR and VBR (Rate control by firmware)
- Supports user-defined quantization matrix
- Supports mono (4:0:0) encoding
- Supports 4x4 and 8x8 transform coding
- Supports 4x4 and 8x8 intra predictions

1.2 Related Document

The reference document includes:

- H264_Rate_Control_User_Guide_xxx.pdf

2.1 Driver Module

The H264 encoder module contains the following part:

- favc_enc.ko

This is the H264 encoder core. It includes the H264 encoder hardware control layer and middle ware (GM_graph) communication layer.

The following example shows the operation procedure:

```
/ # insmod favc_enc.ko h264e_max_b_frame=0 h264e_max_width=1920 h264e_max_height=1088
FAVC Encoder v0.1.105, built @ Sep 30 2014 17:39:25 (GM8136)
```

2.2 Module Parameter

Table 2-1 lists and describes the module parameters when inserting the H264 encoder driver.

Table 2-1. Module Parameters

Name	**Default Value**	**Description**
h264e_max_width	0	Maximal width of the encoded frame
h264e_max_height	0	Maximal height of the encoded frame
h264e_max_chn	128	Maximal number of the channel ID
h264e_snapshot_chn	0	Number of the snapshot channels
h264e_max_b_frame	0	Maximal number of b frame in gop 0: Not support b frame
h264e_yuv_swap	0	Input YUV422 format 0: CbYCrY 1: YCbYCr
use_ioremap_wc	0	Register of mcp280 remap type 0: ncnb 1: ncb
config_path	“/mnt/mtd”	Specify the configure path
h264e_slice_offset	0	Output slice offset 0: Not output the position of echo slice on the bitstream 1: Output the position of echo slice on the bitstream (At most, first four slices)
h264e_tight_buf	0	Allocate exact number of the reference buffers
h264e_one_ref_buf	0	Reduce number of the reference buffers 0: Reference buffer and reconstructed buffer are different buffers. 1: Reference buffer and reconstructed buffer are the same buffers.
h264e_user_config	0	Using “favce_param.cfg” to be the encode setting for each channel
pwm	0	Control clock ratio

This chapter contains the following sections:

- 3.1 /proc/videograph/h264e/info
- 3.2 /proc/videograph/h264e/chn_info
- 3.3 /proc/videograph/h264e/utilization
- 3.4 /proc/videograph/h264e/callback_period
- 3.5 /proc/videograph/h264e/level
- 3.6 /proc/videograph/h264e/property

• 3.7 /proc/videograph/h264e/job
• 3.8 /proc/videograph/h264e/param
• 3.9 /proc/videograph/h264e/didn
• 3.10 /proc/videograph/h264e/ref_info
• 3.11 /proc/videograph/h264e/q_matrix
• 3.12 /proc/videograph/h264e/mcnr

The H264 encoder module provides several proc nodes. Users can read the information of the H264 encoder or setup the configuration through the nodes.

3.1 /proc/videograph/h264e/info

Users can use /proc/videograph/h264e/info to get the driver information, including the versions of the drivers and the values of the module parameters.

Usage:

• Get the version of the driver and value of the module parameters

cat /proc/videograph/h264e/info

```
FAVC Encoder v0.1.105, built @ Oct 13 2014 15:04:11 (GM8136)
module parameter
==================== ======
h264e_max_width 1920
h264e_max_height 1088
h264e_max_b_frame 0
h264e_max_chn 128
h264e_snapshot_chn 0
h264e_yuv_swap 0
pwm 0
config_path "/mnt/mtd"
use_ioremap_wc 0
h264e_slice_offset 1
h264e_one_ref_buf 1
h264e_tight_buf 1
h264e_user_config 0
```

3.2 /proc/videograph/h264e/chn_info

Users can use /proc/videograph/h264e/chn_info to get the settings of each encoded channel, including the resolution, frame rate, gop, bitrate, mode of the rate control, and current quant value.

Usage:

• Get the settings of each channel

cat /proc/videograph/h264e/chn_info

```
chn resolution buf.type gop mode fps bitrate max.br init.q min.q max.q qp
=== ====== ===== ==== ==== ======= ===== ===== ==== ==== ==== ===
0 704x480 D1 60 CB
R
900/3
0
1024 0 30 15 51 29
1 352x244 cif 60 CB
R
900/3
0
512 0 30 15 51 32
```

3.3 /proc/videograph/h264e/utilization

Users can use /proc/videograph/h264e/utilization to get the percentage of the hardware utilization and set the time period of the measured utilization.

Usage:

- Get the hardware utilization

cat /proc/videograph/h264e/utilization

```
HW Utilization Period=5(sec) Utilization=78
```

- Set the period of the hardware utilization measurement

echo [sec] > /proc/videograph/h264e/utilization

3.4 /proc/videograph/h264e/callback_period

Users can use /proc/videograph/h264e/callback_period to get and set the callback period.

Usage:

- Get the callback period

cat /proc/videograph/h264e/callback_period

```
Callback Period = 3 (msecs)
```

- Set the callback period

echo [msec] > /proc/videograph/h264e/callback_period

3.5 /proc/videograph/h264e/level

Users can use /proc/videograph/h264e/level to set the debug level. Higher debug level will dump more information to the background log.

Usage:

- Get the debug level

cat /proc/videograph/h264e/level

```
Log level = 2 (0: emergy, 1: error, 2: warning, 3: debug, 4: info)
```

- Set the debug level

Echo [level] /proc/videograph/h264e/level

[level]:

0: Emergency message

1: Error message

2: Warning message

3: Debug message

4: Information

For example, if the current debug level is 2, the driver will dump the emergency message, error message, and warning message.

3.6 /proc/videograph/h264e/property

Users can use /proc/videograph/h264e/property to get the input property of the specified channel.

Usage:

- Get the input property of the specified channel

cat /proc/videograph/h264e/property

```
usage: echo [chn] > /proc/videograph/h264e/property
FAVCE favce ch0 job 14257793
=============================================================
ID Name(string) Value(hex) Readme
2 src_xy 00240050 roi xy
4 src_dim 00900060 encode resolution
47 init_quant 0000001e initial quant
41 bitrate 000000e6 target bitrate (Kb)
43 idr_interval 00000036 I frame interval
38 didn_mode 00000000 didn mode
…
```

- Set the specified channel ID to get property

echo [chn] > /proc/videograph/h264e/property

3.7 /proc/videograph/h264e/job

Users can use /proc/videograph/h264e/job to get the information of job in the encoder job list.

Usage:

- Get the information of the job list

cat /proc/videograph/h264e/job

```
usage: echo [chn] > /proc/videograph/h264e/job ([chn] = 999: means dump all job)
current [chn] = 999
Engine Minor Job_ID Status Puttime
=======================================
0 41 15838469 STANDBY 0xb3f1
0 35 15841020 ONGOING 0x8e75
```

- Set the specified channel to get the job information

echo [chn] > /proc/videograph/h264e/job

[chn]: 999 means to get job information in job list

3.8 /proc/videograph/h264e/param

Users can use /proc/videograph/h264e/param to get and set the encode parameters.

Usage:

- Get the encode parameters

cat /proc/videograph/h264e/param

Parameter name Value Note

DefaultCfg 1 0: Light quality (Between performance and quality)

1: Performance

2: Quality

3: User definition

SymbolMode 0 0: CAVLC

1: CABAC

ROIQPType 1 0: Disable ROI QP

1: Delta QP

2: Fixed QP

ROIDeltaQP -4 ROI QP = Frame QP - Delta QP

ROIFixedQP 20 ROI QP = Fixed QP

ResendSPSPPS 1 0: Packing sps and pps (Only the first IDR frame)

1: Packing sps and pps (Each I frame_

2: Packing sps and pps (Each frame)

CbQPOffset 6 Cb QP offset

CrQPOffset 6 Cr QP offset

DFDisableIdc 0 Deblock idc

0: Strong

1: Disable

2: Weak

DFAlpha 6 H264 deblock coefficient

DFBeta 6 H264 deblock coefficient

DiDnMode 0 DiDn enable

-1: Using property input

Bit0: Spatial de-interlace

Bit1: Temporal de-interlace

Bit2: Spatial denoise

Bit3: Temporal denoise

PRef0SearchRangeX 32 Search range of X

PRef0SearchRangeY 16 Search range of Y

DisableCoeff 0 Threshold residual coefficient

0: Enable coefficient threshold

1: Disable coefficient threshold

LumaCoeffThd 4 Threshold coefficient of luma

ChromaCoeffThd 4 Threshold coefficient of chroma

DeltaQPWeight 5 Delta QP of each MB

5: Disable Delta QP

4: Enable Delta QP by image variance

DeltaQPStrength 19 Coefficient of Delta QP

DeltaQPThd 231 Coefficient of Delta QP

MaxDeltaQP 5 Max. Delta QP of MB

Transform8x8 0 0: Disable 8x8 transform

1: Enable 8x8 transform

InterDefaultTransformSize 0 Inter hardware transform size

0: 4x4

1: 8x8

DisablePInterPartition 6 Disable inter prediction mode of P frame

Bit 0: 8x8

Bit 1: 8x16

Bit 2: 16x8

Bit 3: 16x16

DisableBInterPartition 14 Disable inter prediction mode of B frame

Bit 0: 8x8

Bit 1: 8x16

Bit 2: 16x8

Bit 3: 16x16

DisableIntra8x8 1 Disable intra 8x8 prediction mode

IntraMode 0 Intra 4x4 prediction mode

0: 5 modes

1: 9 modes

FastIntra4x4 1 Fast algorithm of intra prediction

DisableIntra16x16Plane 1 Disable intra 16x16 plane prediction

DisableIntraInInter 0 Disable intra prediction of P/B frame

DisableIntra4x4 0 Disable intra 4x4 prediction mode of I frame

DisableIntra16x16 0 Disable intra 16x16 prediction mode of I frame

IPOffset 2 QP offset of I/P frame

PBOffset 2 QP offset of P/B frame

QPStep 1 QP step

MinQuant 1 Minimal QP

MaxQuant 51 Maximal QP

IntraCostRatio 0 Intra cost weight

ForceMV0Thd 0 Force MV to be zero by image variance

CABACInitMode 0 CABAC init idc

CostEarlyTerminate 0 Early termination by cost

0: Disable

PCycleEarlyTerminate 4095 Early termination by cycle of P frame: 4095 disable

BCycleEarlyTerminate 4095 Early termination by cycle of B frame: 4095 disable

ScalingListEnable 0 Scaling matrix

0: Disable

1: Enable

MCNREnable 0 MCNR

0: Disable

1: Enable

MCNRShift 2 MCNR parameter

MCNRMVThd 4 MCNR parameter

Profile 100 Default profile

66: Baseline profile

77: Main profile

100: High profile

LevelIdc 0 Level idc

```
0: Using the default setting
Others: level_idc = LevelIdc/10
```

• Set the encode parameters

echo [parameter name] [value(dec)] > /proc/videograph/h264e/param

3.9 /proc/videograph/h264e/didn

Users can use /proc/videograph/h264e/didn to get and set the didn parameters.

Usage:

• Get the didn parameters

cat /proc/videograph/h264e/didn

• Set the didn parameters

echo [parameters name] [value(dec)] > /proc/videograph/h264e/didn

3.10 /proc/videograph/h264e/ref_info

Users can use /proc/videograph/h264e/ref_info to get the number of the allocated buffers and usage of buffer.

Usage:

• Get the number of the allocated buffers

echo 0 > /proc/videograph/h264e/ref_info

cat /proc/videograph/h264e/ref_info

```
dump ref buffer flag = 0 (0: dump pool number, 1: dump ref pool, 2: dump chn pool)
allocate reference buffer va0xbb000000/pa0x20000000, size 27800064
D1: unit 622080, num 36, size 22394880
CIF: unit 158976, num 34, size 5405184
sys info size 11520, mvinfo size 130560, l1col size 261120
total size 28203264 byte (26.896M)
```

Get the number of the allocated buffer

echo 1 > /proc/videograph/h264e/ref_info

cat /proc/videograph/h264e/ref_info

```
Reference Pool
Avail:
id addr_virt addr_phy size
0 0xbb686a00 0x20686a00 622080
1 0xbc8b1600 0x218b1600 158976
Allocated:
id addr_virt addr_phy size
2 0xbb556e00 0x20556e00 622080
3 0xbbaadc00 0x20aadc00 622080
4 0xbc94ca00 0x2194ca00 158976
5 0xbc99a400 0x2199a400 158976
```

• Get the used buffer of each channel

echo 2 > /proc/videograph/h264e/ref_info

cat /proc/videograph/h264e/ref_info

```
Channel used pool
chn res s.res addr_virt addr_phy size
=== ==== ==== ========= ========= ======
0 CIF CIF 0xbc5f6c00 0x215f6c00 158976
1 D1 D1 0xbc09c800 0x2109c800 622080
2 CIF CIF 0xbc692000 0x21692000 158976
3 D1 D1 0xbb12fc00 0x2012fc00 622080
```

3.11 /proc/videograph/h264e/q_matrix

Users can use /proc/videograph/h264e/q_matrix to get and set the scaling matrix.

• Get the scaling matrix

cat /proc/videograph/h264e/q_matrix

• Set the scaling matrix

```
echo [matrix idx] [idx] [value] > /proc/videograph/h264e/q_matrix
```

3.12 /proc/videograph/h264e/mcnr

Users can use /proc/videograph/h264e/mcnr to get and set the mcnr matrix

□ Get the mcnr matrix

```
cat /proc/videograph/h264e/mcnr
```

□ Set the mcnr matrix

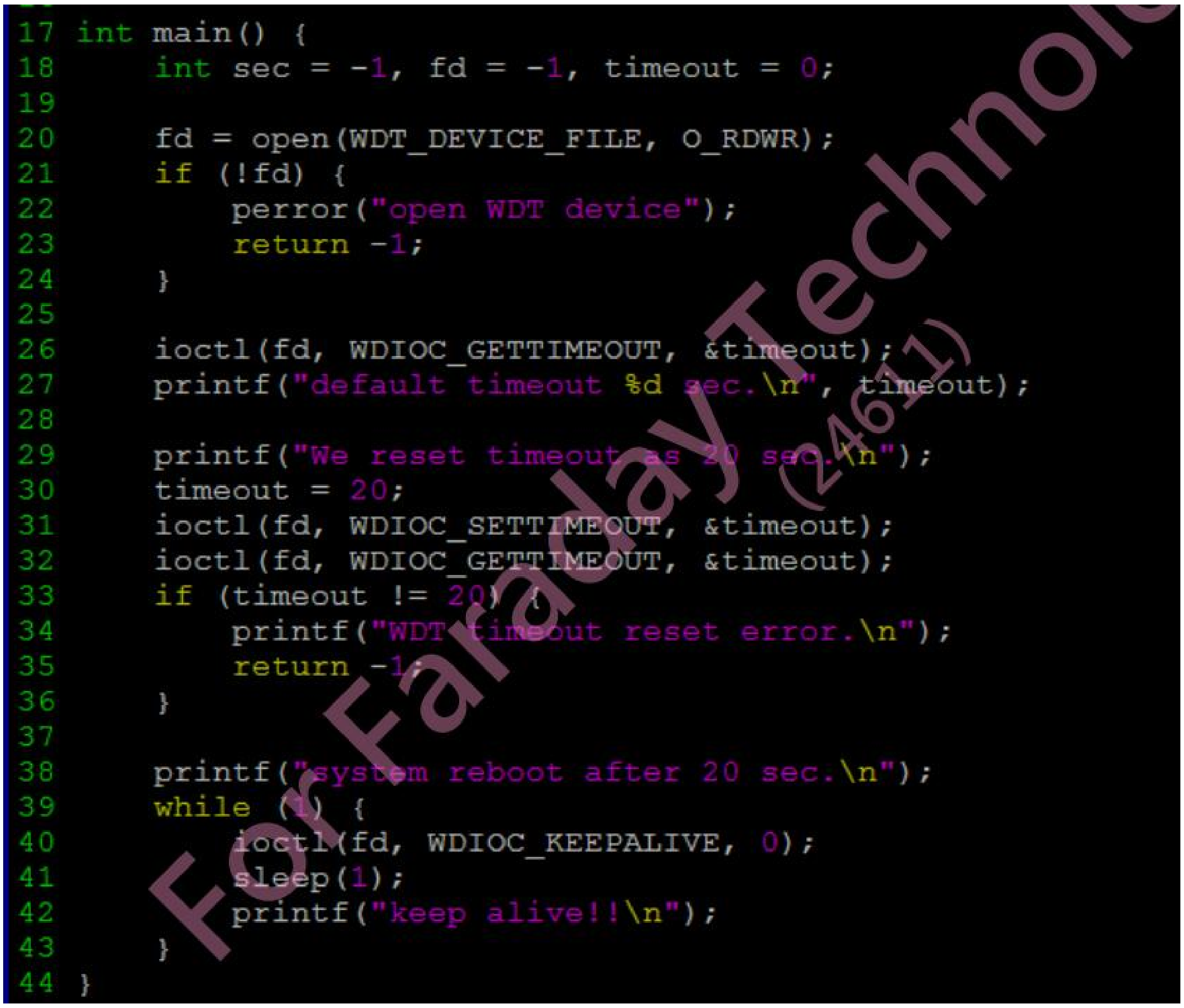
```
echo [H/L] [idx] [value] > /proc/videograph/h264e/mcnr
```

watchdog

In Figure 1-1, it shows the items which should be chosen in menu configuration of Linux kernel.

The steps are Device Drivers ---> Watchdog Timer Support ---> FTWDT010 watchdog

```
ls /dev/watchdog
```



```
int main() {
    int sec = -1, fd = -1, timeout = 0;

    fd = open(WDT_DEVICE_FILE, O_RDWR);
    if (!fd) {
        perror("open WDT device");
        return -1;
    }

    ioctl(fd, WDIOC_GETTIMEOUT, &timeout);
    printf("default timeout %d sec.\n", timeout);

    printf("We reset timeout as 20 sec.\n");
    timeout = 20;
    ioctl(fd, WDIOC_SETTIMEOUT, &timeout);
    ioctl(fd, WDIOC_GETTIMEOUT, &timeout);
    if (timeout != 20) {
        printf("WDT timeout reset error.\n");
        return -1;
    }

    printf("system reboot after 20 sec.\n");
    while (1) {
        ioctl(fd, WDIOC_KEEPALIVE, 0);
        sleep(1);
        printf("keep alive!!\n");
    }
}
```

3DNR

1.1 Overview

3DNR in GM8136 is used to de-noise the frames. The de-noise process is critical for enhancing the scene quality and reducing the encoded bitrate for network transmission. In a real-time IP camera system, the de-noise function is considerably useful in eliminating the Gauss-distribution noise. With this function, the encoder only needs to make little effort to encode the source scene. On the other hand, the encoded files will be smaller and thus reduce the bitrate transferred in the network.

1.2 Features

3DNR contains the following features:

- Supports spatial noise reduction
- Supports temporal noise reduction
- Supports temporal edge enhancement
- Supports temporal strength learning
- Supports source and destination YCbCr swap

In the GM8136 SDK release package, users can find the source code and kernel module of 3DNR from "/module/ft3dnr200".

The 3DNR driver module contains the following part:

- ft3dnr200.ko

This is the 3DNR core. It includes the 3DNR hardware control layer, middleware (GM_Graph), communication layer, and export library layer.

This chapter contains the following section:

- 3.1 /proc/thdnr200/dma Proc Node

The 3DNR driver module provides the proc node. Users can read the information of 3DNR or setup the configuration through the node. The current useful node is listed below.

```
/ # ls /proc/thdnr200/
dma
```

Figure 3-1. Proc Node of 3DNR Driver Module

Table 3-1 shows the proc node component of the 3DNR driver module.

Table 3-1. Proc Node Component of 3DNR Driver Module

/proc/thdnr200 dma param

3.1 /proc/thdnr200/dma Proc Node

The 3DNR hardware supports the setting for the wait intervals of the DMA read and write channels. Users can use the proc node to set the wait intervals of the DMA read and write channels.

3.1.1 param

Users can use the /proc/thdnr200/dma/param node to get and set the parameter for wait intervals of the DMA read and write channels

Usage:

- Get all current values

cat /proc/thdnr200/dma/param

```
/# cat /proc/thdnr200/dma/param
=== DMA Parameter ===
[00]WC_WAIT_VALUE (0x0~0xffff) : 0x0
[01]RC_WAIT_VALUE (0x0~0xffff) : 0x0
```

Usage:

- Set the param_id value to change the parameter value of DMA

echo [param_id] [value] > /proc/thdnr200/dma/param

param_id:

0: WC_WAIT_VALUE is the wait interval of the DMA write channel for each DMA write burst.

1: RC_WAIT_VALUE is the wait interval of the DMA read channel for each DMA read burst.

4.1 max_minors Module Parameter

Users can specify the maximum number of the channels used by the middleware. The usage is:

```
# insmod ft3dnr200.ko max_minors=16
```

If this parameter is not specified, the default value will be used.

4.2 res_cfg Module Parameter

The 3DNR hardware needs the memory space to store the output data depending on the specification of the product. Users can specify various resolutions as the module parameter. The format of the parameter

is "resolution_keywords/channels"; if many resolutions are specified, they should be separated by a comma.
For example, users need the driver to provide one channel of 2M resolution, one channel of VGA resolution, and one channel of CIF resolution:

```
# insmod ft3dnr200.ko res_cfg="2M/1,VGA/1,CIF/1"
```

If this parameter is not specified, the default value will be extracted from the middleware configuration file, "gmlib.cfg", under the [ENCODE_DIDN] section.

4.3 Other Module Parameters

For other module parameters that are not described above, it is not allowed to be modified by users; it is only used by the factory configuration.

REAL TIME CLOCK

IN KERNEL make menuconfig add:
Device Drivers

- Real Time Clock
  - /sys/class/rtc/rtcN (sysfs)
  - /proc/driver/rtc (procfs for rtc0)
  - /dev/rtcN (character devices)

After completing the configuration, users should switch the current directory to arm-linux-3.3/module/RTC/FTRTC011 and make a copy of the RTC kernel module, rtc-ftrtc011.ko. Please use "insmod" to insert this module into kernel and use "mdev –s" to generate a RTC device node as shown as in below

```
/lib/modules # insmod rtc-ftrtc011.ko
ftrtc011 ftrtc011: rtc core: registered ftrtc011 as rtc0
/lib/modules # mdev -s
/lib/modules # ls -lh /dev/rtc0
crw-rw---- 1 root root 254, 0 Jan 1 00:35 /dev/rtc0
/lib/modules #

/lib/modules # date -s 2010.12.06-11:40
Mon Dec 6 11:40:00 UTC 2010
/lib/modules # hwclock --help
BusyBox v1.13.4 (2010-12-06 10:32:54 CST) multi-call binary
Usage: hwclock [-r|--show] [-s|--hctosys] [-w|--systohc] [-l|--localtime]
[-u|--utc] [-f FILE]
Query and set hardware clock (RTC)
Options:
-r Show hardware clock time
-s Set system time from hardware clock
-w Set hardware clock to system time
-u Hardware clock is in UTC
-l Hardware clock is in local time
-f FILE Use specified device (e.g. /dev/rtc2)
/lib/modules # hwclock -w
/lib/modules # hwclock
Thu Nov 31 11:40:43 2013 0.000000 seconds
/lib/modules #
```

By using the RTC related files, users may understand the underneath operations. All paths are related to arm-linux-3.3/.
This article goes with the Linux kernel. It describes RTC in details in Linux.

• Linux-3.3-fa/Documentation/rtc.txt

This implements the GM8136 RTC provided by Grain Media.

• module/RTC/FTRTC011/ftrtc011.c

Scaler300

Scaler300 in GM8136 is used to resize and enhance the video data.

Scaler300 contains the following features:

 Supports maximum image size of 4096x4096
 Supports cropping function for front-end image source
 Supports cropping function for back-end image
 Supports scaling-down/up function with two output resolutions per scan line and bypass function with two output resolutions per scan line
 Supports input formats of YCbCr 4:2:2
 Supports 1-D false color suppression
 Supports 1-D denoise
 Supports smooth and sharpness
 Supports field scaling (Top/Bottom offsets)
 Supports frame-to-field extraction
 Supports up to eight mask windows in one frame with transparency degree control
 User programmable 8-type palette color for mask

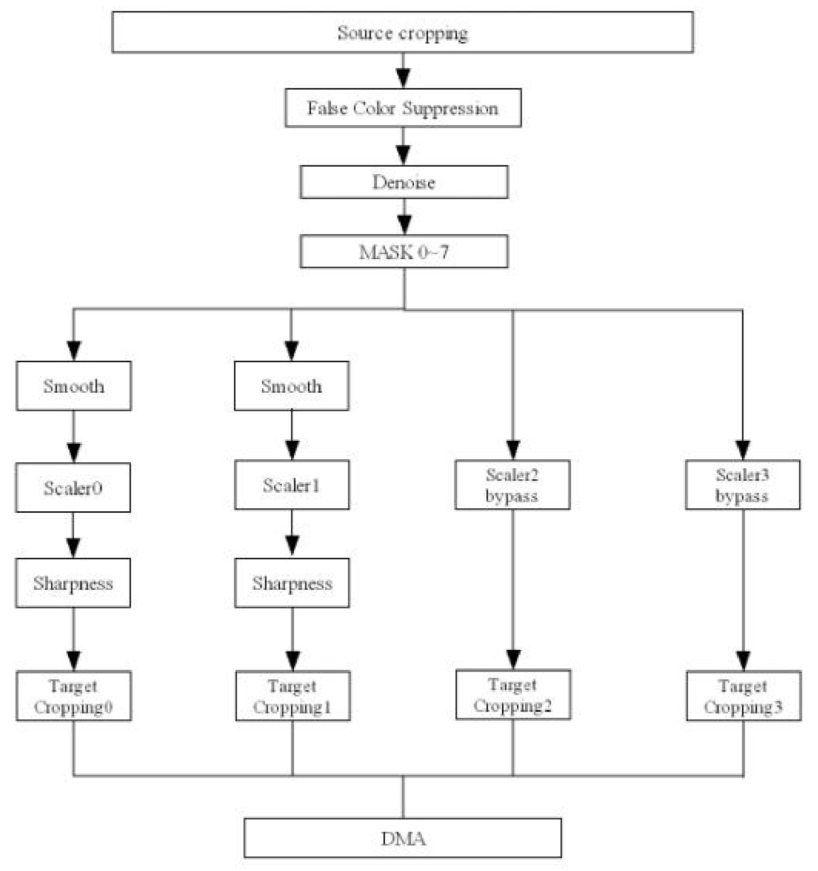


Figure 1.1 Block Diagram of Scaler300

In the GM8136 SDK release package, users can find the source code and kernel module of scaler300 from “/module/scaler300”.

The scaler300 driver module contains the following part:

• fscaler300.ko

This is the scaler300 core. It includes the scaler300 hardware control layer, middleware (GM_Graph),

communication layer, and export library layer.
The scaler300 module provides the proc nodes. Users can read the information of scaler300 or setup the configuration through these nodes. A sample of these nodes is listed below.

```
/ # ls /proc/scaler300/
denoise dma fcs sharpness smooth
```

Figure 3-1. Proc Nodes of Scaler300 Module

Table 3-1 shows the proc node components of the scaler300 module.

Table 3-1. Proc Node Components of Scaler300 Module

/proc/scaler300 fcs enable
param
denoise enable
param
smooth param
sharpness param
dma param

3.1 /proc/scaler300/fcs Proc Node

The scaler300 hardware supports the false-color suppression. Users can use these fcs proc nodes to adjust the fcs parameter for reducing the color noise.

3.1.1 Enable

Users can use the /proc/scaler300/fcs/enable node to control the fcs function.
Usage:

- Get the current value

cat /proc/scaler300/fcs/enable

```
/ # cat /proc/scaler300/fcs/enable
FCS Enable : 0
```

- Set the value to enable/disable fcs

echo [value] > /proc/scaler300/fcs/enable
Value: 0 for disable, 1 for enable

sar_adc

1.1 SAR ADC Overview

Because software can only handle discrete digital signals, all analog signals are required to be translated into this form. SAR (Successive Approximation) ADC provides a method to convert the analog signals into the digital signals. Grain Media uses this common technology to acquire the ADC values for the keypad, battery detection, and Plug-In or Plug-Out of the CVBS signals.
GM8139 implements the SAR ADC technology to detect the keypad input, battery voltage, and CVBS signals. Users can refer to the GM8139 data sheet for more detailed information for the register settings of SAR ADC. By inserting the kernel module provided by Grain Media, users can acquire the ADC values via the corresponding device file described in the following section.

1.2 SAR ADC Basic Usages

The Linux kernel does not provide the driver middleware for ADC. Grain Media uses the basic characteristics of the device driver for implementation; therefore, users will not need to set any option in kernel.
The SAR ADC driver supports three running options. One is the keypad (0x01), the other is the battery voltage (0x02), and another is the CVBS signals (0x04). If users only run CVBS signal detection, please input "insmod sar_adc.ko run_mode=0x04". If users run the keypad and battery voltage detection, please input "insmod sar_adc.ko run_mode=0x03". The default running mode is "all". Users should insert the precompiled kernel module and generate the device file as shown in Figure 1-1.
The SAR ADC driver supports the keypad interrupt limited range. For example, users insert the SAR ADC module as "Insmod sar_adc.ko gain0_lim=0xe6,0xe8 (low value, high value)" , it means that if the value is below 0xe6 and above 0xe8, it will trigger the interrupt.

```
/lib/modules # insmod sar_adc.ko run_mode=0x7
Welcome to use sar_adc_drv
* module_init_func
* register_driver * driver_probe
dev_data_alloc_specific done
set_pinmux set as 0xB8 0xFFEC7638 OK * register_cdev
driver_probe done, io_vadr 0x90C76000, io_padr 0x90A00000 0x832433E0
/lib/modules # mdev -s
/lib/modules # ls -al /dev/sar_adc_drv
crw-rw---- 1 root root 253, 0 Jan 1 00:00 /dev/sar_adc_drv
/lib/modules #
```

sar_adc.ko is the SAR ADC driver module. The device file in /dev/sar_adc_drv contains the physical devices based on SAR ADC. Users can acquire the ADC values by reading this file. Please refer to SAR_ADC/test for a demo program of the keypad detection function. The demo program will detect whether the keypad is pressed. It will show the returned value, which indicates the pushed button. The demo program will also detect the change of the battery voltage. If the KEY ADC status is 4, it means that the keypad is pushed .If the KEY ADC status is 8, it means that the voltage is changed. On the other hand, the SAR ADC driver will automatically detect the CVBS signals to turn off components in order to save power.
The SAR ADC driver module provides repeat duration setting to prevent keypad from repeating inputs. The unit is millisecond. The ioctl parameter is "SAR_ADC_KEY_SET_REPEAT_DURATION".

```
Start SAR ADC Keypad Test
AP get KEY ADC status =4 val = e7
AP get KEY ADC status =4 val = e7
AP get KEY ADC status =8 val = 11
AP get KEY ADC status =8 val = 1
AP get KEY ADC status =8 val = 4
```

1.3 Reference

- arm-linux-3.3/module/SAR_ADC/

All needed information is located in this directory, which contains the kernel module implementation and test program for the user space.

OSG

1.1 OSG Overview

OSG is called "On-Screen Graphic", and it is used to show various graphics on screen. GM8136 OSG only supports the RGB1555 format and transforms it into the black and white YUV422 format. OSG uses THINK2D to complete these jobs.

1.2 OSG Basic Usage

"sw_osg.ko" is the OSG driver module between the middleware and 2D engine driver. If users want to use the OSG function, users need to insert "sw_osg.ko" and "think2d.ko" kernel modules. Please remember to insert "think2d.ko" prior to "sw_osg.ko".
"sw_osg.ko" supports two memory manager modes: Dynamic and static. If users insert "sw_osg.ko" without any module parameter, the dynamic mode will dynamically allocate the memory when setting RGB1555 graphic as the blitting source. Figure1-1 shows how to insert the dynamic mode, "sw_osg.ko". If users insert "sw_osg.ko " with the "idn" and "mem" module parameters, the static mode will allocate the memory by the "mem" parameter when inserting the "sw_osg.ko" module. Figure 1-2 shows how to insert the static mode, "sw_osg.ko".

Figure 1-1. Insert Dynamic Mode

```
/lib/modules # insmod think2d.ko
```

```
think2d_module_init
Think2dGE uses hclk/2.
** T2D VER:1.5 ** think2d_drv_probe done, vbase 0x908AC000, pbase 0x92200000 0x8F706240
/lib/modules # insmod sw_osg.ko
osg canvas is dynammic allocate mode
SWOSG ver 0.1.7
```

Figure 1-2. Insert Static Mode

```
/lib/modules # insmod think2d.ko
think2d_module_init
Think2dGE uses hclk/2.
** T2D VER:1.5 ** think2d_drv_probe done, vbase 0x908AC000, pbase 0x92200000 0x8F706240
/lib/modules # insmod sw_osg.ko idn=132 mem=8192
osg canvas is static allocate mode
SWOSG ver 0.1.7
```

When inserting "swosg.ko" with the "idn" and "mem" module parameters, "mem" refers to how many memories that users need to allocate, the unit is Kbyte (For example, if setting "mem=8192" , it refers to 8Mbytes). "idn" refers to how many pools of the blitting sources that users want, for example, if setting "idn=132", it refers to '128'.

After inserting "swosg.ko" as the static mode, users should use the "echo" command to plan the pool memory.

For example:

"echo 4 64 > /proc/swosg/canvas_static_info" , it refers to set pool of index 4 pool as 64Kbyte.

1.3 OSG Proc Usage

In the "/proc/swosg/" folder, there are two files, one is "canvas_info" which supports read-only, the other is "canvas_static_info" which supports read and write.

- canvas_static_info

If "sw_osg.ko" is inserted as the dynamic mode, this file will be invalid. If "sw_osg.ko" is inserted as the static mode, this file will be valid. In the static mode, users need to use the written function to set the size of the pool memory, and use the reading function to check the allocated memory size for every pool. Figure 1-3 shows the detailed information of "canvas_static_info".

```
/lib/modules # echo 4 64 > /proc/swosg/canvas_static_info
/lib/modules # cat /proc/swosg/canvas_static_info
runmode:static
tnum-alnum: 8-1
alsz-resz-tsz: 65536-8323072-8388608
frambuf info:91000000-3810000-8388608-91781258
id(0) info:0-0K-0-0
id(1) info:0-0K-0-0
id(2) info:0-0K-0-0
id(3) info:0-0K-0-0
id(4) info:91000000-64K-0-1
id(5) info:0-0K-0-0
id(6) info:0-0K-0-0
id(7) info:0-0K-0-0
```

Figure 1-3. Usage of canvas_static_info

- canvas_info

If users set the blitting source via the middleware, users can read this file to check the blitting source information. Figure 1-4 shows the detailed information of "canvas_info".

Figure 1-4. Usage of canvas_info

```
/lib/modules # cat /proc/swosg/canvas_info
osg dma allocate count=0
osg canvas idx: 4
osg canvas usage: 1
osg canvas vaddr: 0x92000000
osg canvas paddr: 0x03810000
osg canvas size: 65536
osg canvas width: 320
osg canvas heigh: 72
osg canvas idx: 6
osg canvas usage: 1
osg canvas vaddr: 0x92010000
osg canvas paddr: 0x03820000
osg canvas size: 65536
osg canvas width: 320
osg canvas heigh: 72
```

GPIO

1.1 GPIO Overview

General-Purpose Input/Output (a.k.a. GPIO) is a common interface of SoC. Users can configure a GPIO pin as an input or output. When the GPIO pin is configured as an output, this pin can be set to the high voltage or low voltage. When the GPIO pin is configured as an input, this pin can be used to detect the voltage level. Moreover, an input pin may be configured as an external interrupt source.
GM8136 provides two ports for GPIO. Each port has 32 pins, which can be individually configured. Please make sure that the pins are well multiplexed before using them. Figure 1-1 illustrates the control flow when a kernel GPIO function is called by the driver.

Figure 1-1. Control Flow of GPIO
When the driver calls one of the functions, gpio_get_value, of the kernel GPIO, kernel will call the corresponding GPIO chips based on the GPIO pin number. GM8136 has two GPIO ports. The first port controls pins[31:0] and the second port controls pins[63:32]. Users should use pins[95:0] to configure the pin and the corresponding port. Specific GPIO chip will call a callback function that has registered in advance. This implementation is a specific GPIO implementation of GM8136.

1.2 Options about GPIO in Linux Kernel

The GPIO function on SoC is FTGPIO010. The GPIO module is built-in in the Linux kernel image. Users can use the menu configuration (make menuconfig) to choose the items for FTGPIO010. As shown in Figure 1-2, the items are chosen in menu configuration of Linux kernel.
The steps are: Device Drivers ---> GPIO Support ---> FTGPIO010 GPIO support
Figure 1-2. Build FTGPIO010
After building GPIO module in the kernel image, users can boot Linux and find sysfs of FTGPIO010, whose path is "/sys/devices/platform/ftgpio010.0". In Figure 1-3, users can find sysfs and make sure that the driver is already built in.
Figure 1-3. sysfs of FTGPIO010

1.3 Examples of GPIO Usage

The usage of the Linux GPIO function is descried in Section 1.3. If users want to set GPIO port 0 pin0 of GM8136 as an output with high voltage, please check the following example code:

```
/* the procedure of setting gpio 0 high */
int ret = -1;
if ((ret = gpio_request(0, "gpio0")) != 0) { return ret; }
if ((ret = gpio_direction_output(0, 1)) != 0) { return ret; }
```

The input pin of GPIO should be used as an interrupt source to prevent the kernel from checking the pin status by using a periodic timer. The Linux GPIO middleware does not provide the GPIO interrupt interface; however, Grain Media has added these functions for the user convenience. Figure 1-4 is a snip example of using the GPIO input as an interrupt code, in which pin0 is set as an interrupt.

Figure 1-4. Example of Using GPIO Input as Interrupt Code

Please note that users must use gpio_request() to control the pin.

In the test_gpio_interrupt function, Grain Media constructs a mode variable, which specifies that this pin should be triggered from high to low. Then, request_irq() is used to register ISR. Please take the IRQF_SHARED flag into consideration. Please use the GPIO input as the interrupt for all pins of the same GPIO port that have the same IRQ number. Users should use the last parameter in request_irq(), dev, to distinguish the pin from being triggered in ISR.

1.4 GPIO in User Space

If users use the Linux GPIO subsystem, GPIO can be set in the user space for kernel sysfs.

First, users should make sure that the following menu configuration is set.

Device Drivers ---> GPIO Support ---> /sys/class/gpio/... (sysfs interface)

Figure 1-5. sysfs Setting for GPIO

After building the kernel image and booting the system, users should follow the steps below for operating GPIO. (The blue words mean results.)

/ # ls /sys/class/gpio/ <<- To see if GPIO is correct or not

export gpiochip0 gpiochip32 gpiochip64 unexport

/ # echo 0 > /sys/class/gpio/export <<- Request port0 pin0. (If using port1 pin0, echo 32 > /sys/class/gpio/export)

/ # ls /sys/class/gpio/ <<- Generate a new folder(gpio0) and its attributes is in the folder

export gpio0 gpiochip0 gpiochip32 gpiochip64 unexport

/ # cat /sys/class/gpio/gpio0/direction <<- Check input of output

in

/ # cat /sys/class/gpio/gpio0/value <<- Above result is input and check its voltage. 1 means high.

1

/ # echo out > /sys/class/gpio/gpio0/direction <<- Change to output

/ # cat /sys/class/gpio/gpio0/direction <<- Check result

out

/ # cat /sys/class/gpio/gpio0/value <<- Check voltage of output

0

/ # echo 1 > /sys/class/gpio/gpio0/value <<- Set voltage to high

/ # cat /sys/class/gpio/gpio0/value <<- Check result

1

Important: Before using this user space function, users must set **pinmux** for the GPIO pin.

1.5 Related File

Documentation/gpio.txt